News Express

Distribution Software Hardware Headline Event

ディストリビューション戦国時代

締め切り間際に、Red Hatと五橋研究所のパートナー契約が突然打ち切られたというニュースが飛び込んできた。日本語redhat Linuxを開発／販売していた五橋研究所は、今後Red Hatのロゴと名称を使えないことになった。その結果、五橋研究所は新会社のレーザーファイブから新たにLASER5 Linuxをリリースした。

7月から8月にかけて、Debian GNU／Linuxをベースにした新しいディストリビューション「Dice Linux」と「でびまる」が発表された。最近、Red Hatベースのディストリビューションばかり目立っていたが、Debianから新しい試みを始めようという動きだ。

Red Hatが新たな日本語版Red Hat Linuxをリリースするのは間違いない。Caldera OpenLinuxは本格的な日本進出の構えを見せているし、カナダのStorm Linuxにはすでに日本語マニュアルがあるようだ。SuSE Linuxといった海外のディストリビューションも、日本へ進出してくるかもしれない。

新しいディストリビューションが参入すれば、既存のディストリビューションはバージョンアップで立ち向かう。これからますます開発競争が激しくなっていくだろう。いよいよ戦国時代である。

今月のイベント

イベント	Linux West（COMMUNET '99）
内容	有料セミナー、オープンセミナー(無料)の開催、Linux関連企業による、Linuxを使ってのシステム構築等の展示など。
期間	1999年9月14日（火）～9月16日（木）
会場	ATCホール（大阪市住之江区）
お問い合わせ	COMMUNET 事務局 TEL 06-6612-3773 http://www.jma.or.jp/communet/

イベント	LinuxWorld Expo/Tokyo'99
内容	コンファレンス、ワークショップ、展示会
期間	1999年9月29日（水）～9月30日（木）
会場	東京ファッションタウン（江東区有明）
お問い合わせ	LinuxWorld運営事務局 TEL 03-5276-3751 http://www.idgexpo.com/linuxexpo/

Distribution 発売日 1999年9月17日

かな漢字変換にWnn6 Ver.3をバンドルした「Vine Linux 1.1CR with Wnn6」

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

発売 技術評論社
TEL 03-3225-3481
価格 9800円

技術評論社は、6月18日に発売したVine Linux 1.1 CR Official製品版に、Wnn6 Ver.3をバンドルした「Vine Linux 1.1 CR with Wnn6」を、9月17日より限定2000本で発売する。また、同時にVine Linux 1.1CR Official製品版の登録ユーザーに対してバージョンアップ版の販売（直販のみ）も行われる。

「Vine Linux 1.1 CR with Wnn6」は、オムロンソフトウェアより7月29日に発売されたかな漢字変換システム「Wnn6 for Linux/BSD Ver3.0」を、Vine Linux用にカスタマイズしてバンドルしたもの。

Software 発売日 1999年9月24日

Linux用のフォント全12書体「Linuxの美しいフォント集」

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

発売 ネットビレッジ
TEL 0426-70-7771
価格 3800円

ネットビレッジから、LinuxのX-TTサーバ環境に対応したTruetypeフォントデータ集「Linuxの美しいフォント集」が発売される。名人「下村芳流」新規書き下ろしの美しい「ペン字」2書体を含めた全12書体を収録している。

対応するOSは、TurboLinux、redhat Linux、Debian GNU/Linux、Vine Linux、Plamo Linux、FreeBSDなどのX-TTサーバ対応環境、およびWindows95/98/NT。

収録Truetypeフォントは、芳流行書（ペン字行書体）、芳流行書（ペン字楷書体）、鎌倉（太行書体）、かんてい流（中かんてい流）、東山（極太角ゴシック体）、桃山（極太丸ゴシック体）、北山（極太角ポップ体）、斑鳩（中細明朝体）、東山（太角ゴシック体）、飛鳥（中教科書体）、鎌倉（中太行書体）、桃山（中細丸ゴシック体）の全12書体。

Hardware 発売日 1999年8月20日

19インチラックマウントPCサーバ「Trus-1」、「Trus-2」

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

発売 ぷらっとホーム
TEL 03-3251-2600
価格 24万8000円〜

ぷらっとホームからISP向けの19インチラックマウントPCサーバ「Trus-1」、「Trus-2」が発売された。Trus-1は高さ44mm、Trus-2は高さ88mmと、従来製品と比べ4分の1もしくは2分の1の薄さとなっている。

Trus-1は、CPUがCeleron-433MHz、メモリ64Mバイト（最大768Mバイト）、ハードディスクは9GバイトのUltraWide SCSIタイプ、EthernetコントローラにIntel82558を搭載。IDEハードディスク8Gバイトを搭載した「Trus-1i」も用意されている。Trus-1が31万8000円、Trus-1iが24万8000円。

Trus-2は、CPUにPentium III 600MHzを1基搭載した「Trus-2S」と、同CPUをデュアル構成にした「Trus-2D」の2モデルが用意されている。メインメモリは128Mバイト（最大1Gバイト）、ハードディスクは9GバイトのUltraWide SCSIタイプ、EthernetコントローラにIntel82558を搭載。Trus-2Sが39万8000円、Trus-2Dが49万8000円。

Hardware 発売日 1999年9月1日

世界最小サイズのLinuxサーバ「ブルーグラス」

URL http://www.aqua-computer.com/

発売 アクアリウムコンピューター
TEL 06-6569-1006
価格 22万8000円

アクアリウムコンピューターから、日本語redhat Linux 6.0をプリインストールした小型サーバ「BlueGrass」が発売される。本体サイズが幅92×高さ218×奥行185mmと非常に小さいことが特徴で、CPUがMediaGXM-233MHz、メモリは64Mバイト(最大128Mバイト)、ハードディスクは6.4Gバイト、100Base-TのEthernetを搭載している。マザーボードは台湾ADVANTECHのPCM-5820。キーボード、マウス、モニタが接続可能なインターフェイスを内蔵している。

今回発売される「BlueGrass」は、プロトタイプバージョンで限定30台の発売となる。10月より正式出荷開始。すでに、メールマガジン専門サイト「まぐまぐ」の一部のサーバとして導入されるなど実際に運用も開始されている。9月14日〜16日まで大阪ATCホールにて開催されるLinuxWestと、9月29日〜30日まで東京ファッションタウンで開催されるLinuxWorld Expo/Tokyo99で展示される予定。

出荷開始は9月30日。販売総代理店はデジタルデザイン、Web上での販売 (http://aqua.mag2.com/) はまぐまぐが行う。

Hardware

発売日　1999年10月

SGI初のIntelアークテクチャのサーバ「SGI 1400L Server for Linux」

URL　http://www.sgi.co.jp/

発売	日本SGI
TEL	0120-161086
価格	米国価格7935USドル（エントリーモデル）から

SGIから、オペレーティングシステムにLinuxを採用したエンタープライズサーバ「SGI 1400L」が発売される。CPUは、Pentium III Xeon 500MHz、4プロセッサまでのSMP（Symmetric MultiProcessing：対称型マルチプロセッシング）に対応する。メモリは256MバイトECC付き（最大4Gバイト）、SCSIインターフェイスカードを2枚内蔵し、1枚はNarrowコネクタで主に低速な周辺機器用、もう1枚は、LVD（低電圧差動）SCSI、Ultra SCSI、Ultra2 SCSIに対応し、高速ハードディスクを接続する。ハードディスクは、最大6台で109Gバイトまで内蔵可能で、ホットスワップにも対応する。

400Wの電源を3台搭載していて、1台が故障しても残りの2台から供給され、故障した電源を通電中に交換できるリダンダント機能がある。ケースは、幅311×高さ483×奥行635mmで、タワー型のほかに、ラックマウントタイプも用意されている。

SGI Linux Environment with Rea Hat Linux 6.0は、Red Hat Linux 6.0をベースに、SGIがプラットフォームに合わせていくつかの機能を拡張している。

なお、SGI 1400L/1400Mはすでに米国では発売されており、日本国内での発売は、10月からの予定。

Hardware

発売日　1999年8月4日

Linuxに対応したバックアップ用DATドライブ「HP SureStore DAT24eU」

URL　http://www.jpn.hp.com/go/tape/

発売	日本ヒューレット・パッカード
TEL	03-3335-8333
価格	23万5000円

日本ヒューレット・パッカードから、Windows、Linuxに対応したDATドライブ「HP SureStore DAT24eU」が発売された。

HP SureStore DAT24eUは、DDS1、DDS2、DDS3に対応し、容量は最大24Gバイト(非圧縮時12Gバイト)、転送速度は毎秒2Mバイト(1時間に7.2Gバイト)。3時間で最大24Gバイトのデータバックアップが可能となっている。

インターフェイスは、Single-Ended Narrow SCSI-2。また、ドライブ自身が常にドライブとテープの状況を監視し、定期的なメンテナンスが必要な場合や故障時に警告メッセージを対応したバックアップソフトウェアを通じてユーザーに通知する「TapeAlert機能」や、アナログの波形データを正確に認識し、エラーレートの低減を図る「フィードフォワードイコライザ」を搭載している。

対応OSは、Red Hat Linux 5.2のほか、Windows 95/98、Windows NT Server 3.51/4.0、Windows NT Workstation 4.0、Sun Solaris 2.3以上、IBM AIX 3.2.5/4.X、SGI IRIX 5.1以上、IBM OS/2 Warpなど。

Hardware

発売日　1999年8月1日

Pentium III 600MHzデュアルプロセッサ搭載の「POLY Dual 1200シリーズ」

URL　http://www.arcbrain.co.jp/

発売	アークブレイン
TEL	03-3375-8968
価格	43万8000円〜

アークブレインから、Pentium III 600MHzデュアルプロセッサを搭載した「POLY Dual 1200シリーズ」が発売された。

OSにTurboLinux日本語版4.0を搭載したPOLY Dual 1200BDU-LNXは、CPUにPentium III 600MHz×2、メモリ128MバイトECC付き（最大1Gバイト）、ハードディスクは9GバイトのUltra2 SCSIタイプを搭載している。メタルフルタワーケースは、幅191×高さ610×奥行572mmで、10個の5インチベイと400W電源を装備している。価格は64万8000円。

同時に、Pentium III 600MHzシングルプロセッサの「POLY 1200B5シリーズ」も発売された。

OSにTurboLinux日本語版4.0を搭載したPOLY 1200B5/U2W-LNXは、CPUにPentium III 600MHz、メモリ128MバイトECC付き（最大1Gバイト）、ハードディスクは9GバイトのUltra2 SCSIタイプを搭載している。AXPミドルタワーケースは、幅197×高さ508×奥行432mmで、3個の5インチベイと4個の3.5インチベイ、250W電源を装備している。価格は43万8000円。

OSは、TurboLinux日本語版のほか、WindowsNT Workstation 4.0のモデルも用意されている。ハードディスクはメタルフルタワーケースの場合、最大で18Gバイトのドライブを7台まで対応可能となっている。

全モデルとも、100BASE-TX/10BASE-T対応ネットワークカードを標準搭載した。

News Headline from 日刊アスキーLinux

www.Linux24.com

Linux対応のFlash Playerが公開

1999年8月25日

米Macromediaは、9月1日より「Flash Player for UNIX」の無償公開を開始すると発表した。Linux、Sun MicrosystemsのSolaris、SGIのIRIXに対応したソフトウェアが用意され、Netscape Navigatorのプラグインとして動作する。Netscape Navigatorのバージョン3以降に対応する。

FlashはMacromediaが開発したグラフィック、動画、MP3オーディオ、入力フォーム、インタラクティブ性を実現するファイルフォーマットであり、Linux版、Solaris版に関しては今までベータテストが行われていた。

Macromedia (http://www.macromedia.com/)

日本HP、Linuxサービスを本格始動

1999年8月25日

日本ヒューレット・パッカードは、LinuxをプリインストールしたPCサーバ「HP Linux E60」を9月1日に発売する。エントリークラスPCサーバ「HP NetServer E60」に、日本語redhat Linux 5.2をプリインストールしたもの。HP NetServerシリーズを取り扱っている販売会社が、HP NetServer E60と、日本HPのサポートおよびプリインストールサービス、そして販売会社オリジナルのサービスを組み合わせたうえで販売する。価格は販売各社により異なるが、HPによると50万円以下になるという。

HP Linux E60シリーズでは、Linuxサポートが付いている「HP Linux E60 S-Pack」とLinuxサポートなしの「HP Linux E60 I-Pack」の2製品が提供され、サポート付きの「HP Linux E60 S-Pack」は、本体「HP NetServer E60」の3年間の「ハードウェア・パック」（翌日訪問のメインテナンス）のほか、1年間のWeb、メールによる技術支援、障害発生時に電話による切り分け支援のサポートなどが提供される。

FreeTypeライブラリがAppleの特許を侵害している?

1999年8月19日

フリーなTrueTypeラスタライザ「FreeType」を開発している「FreeTypeプロジェクト」は、FreeTypeで使われている手法に関連した、Appleの持つ特許を発見したと警告した。同プロジェクトでは、これが直ちに特許侵害にあたるかは分からないが、FreeTypeを使うことがアメリカや日本などで違法である可能性もあるとしている。現存のところ、Appleの法律部門にメールを送り、その返事を待っているところだという。

TrueTypeラスタライザとは、ベクトルデータであるTrueTypeフォントを、実際に表示される形式のビットマップデータに変換するもの。FreeTypeはX-TTやVFlibなどに使われている。

オムロンソフトウェア「Wnn」（うんぬ）のJava版をWeb上で無償公開

1999年8月16日

オムロンソフトウェアは、マルチプラットフォームで動作する「Scalable Wnn」（スケーラブルウンヌ）をJavaで開発、9月から同社のWebページで評価版のダウンロードが開始される。

マルチプラットフォーム化するために、WnnをJavaで書き直し、Wnnの各機能を「Plug」と呼ばれるモジュールに分割、これによりPDAのようにハードウェアリソースの限られた環境でも、必要な機能をセレクト／組み合わせて実装できる。PDAでの利用を考えると、ネットワークに接続することにより、サーバ側の言語処理環境と協調し、辞書の共有や高度な変換、多数の文字フォント使用など、ハードウェアが持っていない機能も使うことができるようになるのがメリットとなる。また、中国語、韓国／朝鮮語などの言葉が、日本語と統一的に扱えるマルチリンガルであることも、特徴の1つ。

Wnnは、クライアント／サーバ方式のほか、アプリケーションから直接変換エンジンを制御する方式をサポートしている。Scalable Wnnでは、クライアント／サーバ方式の場合はWnn4、Wnn6同様、クライアントとしてMuleやxwnmoが利用可能だ。アプリケーションからの直接制御では、Javaアプリケーション、Java Appletからの利用ができる。

オムロンソフトウェア
(http://www.omronsoft.co.jp/)

米SGI、XFSのソースコードの一部をGPLで公開

1999年8月16日

米SGIは、ファイルシステム「XFS」のソースコードの一部をLinuxに移植して、GPLのもとで公開した。XFSは同社のOS「IRIX」で使われているファイルシステム。XFSをLinuxに移植するということは、すでに5月に発表されていた。今回公開したのはソースコードのごく一部分であるため、これだけではXFSを使用することができない。

XFSは、ファイルシステムを更新する際にログをとる（ジャーナリングファイルシステム）ため、突然のシステムクラッシュでもファイルシステムの修復が極めて速い。現在Linuxで主流のファイルシステムext2では、fsckによる修復に時間がかかってしまううえ32ビットファイルシステムであるため2Gバイトを超えるファイルを扱うことができないのに対し、XFSは64ビットファイルシステムであるため非常に大きなファイルを扱うことができる。

Open Source XFS
(http://oss.sgi.com/projects/xfs/)

Oracle8i for Intel-Linux ダウンロード開始

1999年8月12日

米OracleのWebページで、Oracleの最新バージョンであるOracle8iのLinux対応版ダウンロードが始まった。開発者用ライセンスとして無料でダウンロード可能だ。

Oracle8iは、Oracle8にJava、EJB（Enterprise Java Beans）、CORBAといったテクノロジーを実装し、同社が開発したJDK1.1準拠のJVM（β）を搭載、「SQLJ」と呼ばれる機能により、Javaのコード内にSQL文を埋め込んだ「Javaストアドプロシージャ」を実行できる。Javaストアドプロシージャは、Visual Cafe、Borland JBuilderといったJavaの開発ツールで作成できる。

同社ではこのほか、Oracle8iによるWebアプリケーションを作成するツール「Oracle WebDB」もダウンロードできるようにした。

Oracleのテクニカルサイト
(http://technet.oracle.com/tech/linux/)

Red Hatが株式を公開、株価は3倍以上に

1999年8月12日

大手のLinuxディストリビューターのひとつ、米Red Hatが11日、初の株式公開を行った。Red Hat株はNASDAQでの公開直後の46ドルから順調に値を上げ、最高値56.75ドルに達したあと、52.06ドルで取引を終えた。公開前の株価14ドルからは38.06ドル高、272%も値を上げた。Red Hat社長のRobert Young氏は9100万株を所有しており、公開によって一度に3億ドル近くを手にしたことになる。

この株式公開は、オープンソース製品をメインに据えた企業としては初めてのものであり、値動きが注目されていた。

Red Hat (http://www.redhat.com/)

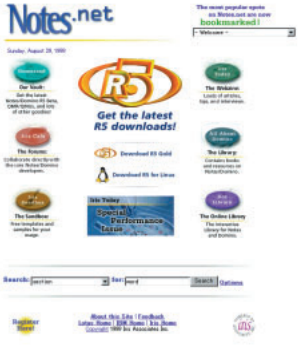
Domino R5 for Linux Sneak Previewが登場

1999年8月11日

米ロータス社のサイトに、Webアプリケーションサーバ「Domino R5」のLinux版がアップロードされた。今回は「Sneak Preview」ということで、正式版のGold Versionまではまだまだといったところだが、インストールしたテスターから、早くもフィードバックが返ってきて、反響の大きさを伺わせる。

ただし、Lotusはそのリリースノートの中で、このバージョンはあくまでもSneak（コッソリと、密かに） Previewであり、実稼働環境へのインストールは推奨しないことを強調している。Lotusは、Domino R5のベータ版を秋口には登場させたい意向。

Domino公式サイト
(http://www.notes.net/)



「GCC 2.95」がリリース

1999年8月1日

GNU ProjectとGCC／EGCSの開発チームは、7月31日（現地時間）、コンパイラスイート「GCC 2.95」をリリースしたと発表した。GCCとEGCSの開発を、GCC Steering Committeeが統合して行っていくという4月の発表以来、初めてのバージョンアップとなる。

これまで、GCCの開発はGNUプロジェクトを推進するFree Software Foundationが行っていた。一方、GCCから派生したEGCSの開発は、Cygnus Solutionsが運営するEGCS Steering Committeeが行っていた。GCCとEGCSの完全な統合を目指して、EGCSの開発を行っていたEGCS Steering CommitteeがGCC Steering Committeeと名称を変更して、開発の責任を負っている。

機能としては、安定版リリースのGCCに、性能が高いとされているEGCSの機能の統合が計られている。具体的には、オプティマイザの性能の向上、Javaのフロントエンドの統合など、サポート言語に対する機能追加、実験段階だが国際化のサポートなどの機能も組み込まれている。

今回のバージョンアップでは同時に、「GCC」の正式名称「GNU C Compiler」を「GNU Compiler Collection」と変更。これは、C言語以外に、C++、Objective C、Ada 95、Fortran 77、Pascalなど複数の言語をサポートしている現状に合わせたものとなっている。略称はこれまでどおりの「GCC」のままとなる。

IBM、PCサーバ「Netfinity」でLinuxサポートを強化

1999年7月29日

米IBMは27日（現地時間）、同社が販売するPCサーバ「Netfinity」シリーズ向けに、Linuxのサポートを強化すると発表した。これによれば、同シリーズにおける、Red Hat、Caldera、SuSE、TurboLinuxについて、無償で90日間の標準サポートが提供されるという。

同社では、DB2のLinuxへの移植を7月末に、またLotus Notes/DominoのLinux移植を年内に予定している。

NEC、NECパーソナルシステムと共同でコンシューマー向けLinuxサービスを開始

1999年7月19日

NECは、同社の販売会社であるNECパーソナルシステムと共同で、コンシューマー向けデスクトップパソコン「ValueStar NX」シリーズ、ノートパソコン「LaVie NX」シリーズ向けに、Linuxサポートサービスを開始すると発表した。サービス自体はすべて、NECパーソナルシステムが行う。

提供されるサービスは、Linuxのインストールや、ネットワークセットアップ、周辺機器の設定、Linuxインストール後のサポートなど。これには、Windows 98とLinuxのマルチブート環境の設定や、同社で動作確認済みの外付けモデム、モデムカード、ネットワークカードの設定などが含まれる。

対応する製品は、5月以降に発売された「ValueStar NX」シリーズと「LaVie NX」シリーズで、対応ディストリビューションはTurboLinux 日本語版 4.0のみ。

Distribution ▶▶▶

Windowsパーティションにインストールできる「Phat Linux 3.0」リリース

米Phatは8月7日、Phat Linux 3.0をリリースした。Phat Linux 3.0はFAT16とFAT32ファイルシステムに対応しており、FAT上の1ファイルを、Linuxのext2ファイルシステムとして、ループバックデバイスによりマウントする。これにより、Linux用にパーティションを切り直す必要がなくなるほか、FAT上でUNIXのファイルを扱えるようにしたファイルシステムUMSDOSと比べて、ファイルへのアクセスが速い。Phat Linux 3.0には新たにKDEがプリインストールされ、Windowsユーザーにとってより使いやすくなっている。パッケージシステムにはRPMを採用している。

Phat Linux 3.0のCD-ROMは、メールにより注文することが可能(下記webページ参照)。価格は20ドル。また、Phat Linuxのサイトよりダウンロードすることもできる。

Phat Linuxのホームページ (http://www.phatlinux.com/)

Debian GNU/Linuxをベースとしたディストリビューション「Dice Linux」

Dice Linuxは、Debian GNU/Linuxのサブセットとして開発されたもので、同ディストリビューションと100%の互換性を持つ。開発チームのProject Diceは、Debian GNU/Linuxのシステムの導入方法や配布形態のありかたを探究するために組織されたボランティアベースのプロジェクト。

Dice Linuxは、Debian GNU/Linuxの数千からなる膨大な収録パッケージを厳選し、ソース、バイナリを含むCD-ROMで計2枚というコンパクトな容量とした。また、Debian JP ProjectがDebianに提供した日本語環境なども収録されている。

そのほか簡単インストーラにより、dselectを使用することなくインストールできる。また、エディタ「dedit」、メーラー「dmail」、やPPPツール「dppxp」などを含む、「Dice Tools(略称 dtools)」と呼ばれるツールを収録する。dtoolsはGPLのライセンスに従って配布される。

Debianベースの新ディストリビューション「でびまる 0.1」リリース

Debian GNU/Linuxベースの新ディストリビューション「でびまる GNU/Linux 0.1」が7月末にリリースされた。今回のリリースはパッケージ選別の方向性やこれに対するユーザーの意見を聞くための評価版。

でびまるは、Dice Linuxとは別のディストリビューションで、「でびまる制作委員会(仮称)」が開発を行なう。

でびまるには、dselectを使用しない簡易インストーラが付属する。

また、Debian GNU/Linuxの公式CD-ROMは、Debianフリーソフトウェアガイドラインに適合しない「non-free」といわれるソフトウェアは、再配布に問題がないパッケージでも収録していないが、でびまるは、CD-ROM配布での便宜を計って、再配布に問題のないnon-freeパッケージを収録していくという。CD-ROMはソース込みで多くとも2枚におさめる予定。

SuSE、VMware 1.0トライアル版をバンドルした「SuSE Linux 6.2」を発売

ドイツのSuSEから、「SuSE Linux 6.2」が発売された。新バージョンではカーネル2.2.10、glibc 2.1、XFree86 3.3.4を採用している。デスクトップ環境はKDE 1.1.1となっており、前バージョン6.1で採用されたGNOMEは今回バンドルされていない。

また、今回新たに、1台のPCで同時に複数のOSを稼動させるVirtual Platform「VMware 1.0」の30日間トライアル版をバンドルする。このほかに、RealPlayer 5.0、StarOffice 5.1(Personal Edition)、WordPerfect 8(Download Edition)、IBM ViaVoice Developer's Kit、Runtime Kit Beta 1.0などが付属する。

価格はパッケージ版が46ユーロ(約5600円)。アップデート版が38ユーロ(約4600円)。同社のWebページで購入できる。

カナダから新しいディストリビューション「Storm Linux」リリース

カナダのStormix Technologiesは、7月6日、Debian GNU/Linuxをベース(Kernelは2.2系)としたディストリビューション「Storm Linux」のアルファ版を公開した。

Storm Linuxで、もっとも目を引くのが「Storm Administration System(SAS)」だ。SASはシステム管理者向けのツールで、クライアントソフトからネットワークからでも操作することができる。

同社Webサイトには、すでに日本語インストールマニュアルが公開されている。

Stormix Technologiesは、1999年2月に、Linuxを対象としたアプリケーションプログラムやツールを提供する目的で設立された。Storm Linuxは、オープンソースで開発を進めること、すべての製品に関してGPLを使うことを明言している。

Stormix Technology Inc. (http://www.stormix.com/)

Red HatとLASER5、突然の契約打ち切り

▶ 日本語redhat Linux 6.0はLASER5 Linux 6.0として発売へ

Red Hat日本代理店の五橋研究所は8月26日、米国Red Hat社とのパートナー契約を解除したと発表した。現在進行中の日本語redhat Linux 6.0の開発は中止し、それに代わるLASER5 Linux 6.0を9月17日に発売するという。LASER5 Linux 6.0は開発中だった日本語redhat Linux 6.0とほぼコンパチブルになるもよう。

五橋研究所は、1997年2月より米国Red Hat社とパートナー契約を結んでおり、日本語redhat Linuxの開発、販売を行ってきたが、今後はRed Hat Linuxをベースに、日本語ローカライズした独自ブランド「LASER5 Linux」として開発していくという。

五橋研究所では、8月31日に全額出資の子会社としてレーザーファイブ株式会社を設立し、日本語redhat Linuxを開発していたOS事業部を全営業権とともに移管する。レーザーファイブは設立時、従業員16名で資本金は1000万円となる見込み。今後は、レーザーファイブがLaser5 Linuxの開発、プロモーション、販売、サポートを行うことになる。

すでに発売中である日本語redhat Linux 6.0 Server EditionなどのRed Hat製品についてのサポートは、レーザーファイブが引き続き継続して行う。また、Red Hat製品の輸入販売は継続して行い、日本語版redhat Linux 5.2のFTP版は、1999年8月24日より180日間だけ配布を続けるが、その後は配布を中止するとのことだ。

関係者の話によると、Red Hat社はパッケージ1本あたり3000円のライセンス料を要求しており、五橋研究所がこれに応じなかったところ、8月24日、突然契約解除の通告が届いたという。これにより五橋研究所はRed Hatのロゴマーク（シャドーマン）とRed Hat Linuxの名称を使用できなくなった。

なお、Red Hat社会長ボブ・ヤング氏は9月8日から幕張メッセで開かれるWORLD PCフォーラム99に合わせて来日する予定であり、その際に日本法人設立の発表があるとも言われている。

この件について編集部からの問い合わせに対し、Red Hat社の広報担当者はノーコメントと回答している。

レーザーファイブのホームページ（http://www.cdrom.co.jp/）

LASER5 Linux 6.0

▶ Red Hat Linux 6.0を日本語化したディストリビューション

レーザーファイブから、LASER5 Linux 6.0が9月17日に発売される。LASER5 Linux 6.0は、英語版のRed Hat Linux 6.0をベースに日本語化したもので、価格は1万2800円。90日間3件までのサポートを電話、FAX、電子メールで受けられる。

また、サポートと各種付属ソフトウェアを省いた廉価版として、LASER5 Linux 6.0 日本語入力キットが6800円で発売される。これには、ATOK12 SEとWnn6 Ver.3が付属している。

LASER5 Linux 6.0はカーネル2.2.xを採用しているので、複数CPUを搭載したシステムでのSMP（対称型マルチプロセッシング）に対応し、RAID、UPS、DATなどのサーバ周辺装置もサポートを広げた。デスクトップ環境として、日本語化したGNOMEとKDEを採用していて、WindowsライクなGUI環境を使用することができる。

日本語入力には、ATOK12 SE for LinuxとWnn6 Ver.3をバンドルしている。リョービの商用フォント5書体と、WindowsとLinuxのデュアルブートを簡単に行えるSystem Commander Lite、3DCGアニメーションソフトBlender、暗号化telnetのSSH、バックアップソフトのBRU、ウイルス対策ソフト、Open Sound Systemなどが付属する。

製品名	LASER5 Linux 6.0
価格:	1万2800円
問い合わせ先:	レーザーファイブ 03-5296-0670 http://www.cdrom.co.jp/

Distribution ▶▶▶

TurboLinux PRO 日本語版 4.2

▶ オフィススイートApplixware 日本語版 4.4.2をバンドル

TurboLinux PRO
日本語版 4.2

価格:	2万9800円
問い合わせ先:	ターボリナックス ジャパン 03-3469-8388 http://www.pht.co.jp/

ターボリナックスジャパンから、TurboLinux PRO 日本語版 4.2が9月10日に発売される。TurboLinux PRO 日本語版 4.2は、7月9日に発売されたTurboLinux 日本語版 4.0を一部改良し、日本語オフィススイートのApplixware 日本語版 4.4.2をバンドルしたものである。

TurboLinux PRO 日本語版 4.2の価格は2万9800円で、90日間3件までのオンラインサポートが付いている。従来のTurboLinux PRO 2.0、PRO 3.0、スタンダード版4.0の利用者は、Webからオンラインで申し込むと1万9800円でアップグレードできる。また、学生、学校教職員、教育機関向けにアカデミックプライス版1万9800円も発売される。

■Applixware日本語版 4.4.2

Applixwareは、米applix社が開発しているオフィススイートで、その日本語版4.4.2は、日本語ワープロによる文書や報告書の作成、データ分析、メールの送受信、イラスト描画、図面作成などの機能を持っている。インストーラ、メニュー、ヘルプ、オンラインブック、マニュアルのすべてを日本語化し、インラインで日本語入力ができる。

Applixwareは、**表1**のような機能を統合していて、共通の操作環境で作業が行えるようになっている。それぞれの機能を連携できるので、たとえばワープロ文書中に表計算の結果を貼り付けたり、グラフィックスツールで作成した画像をプレゼンテーションツールで利用することが可能である。

また、フィルタ機能が拡張され、WindowsのMicrosoft Office 97のWordやExcel、PowerPointで作成したデータを読み込むことや、GIFやJPEG、TIFFなどのさまざまな画像フォーマットに対応した。

■商用ソフトウェア

日本語入力には、ATOK12 SE for LinuxとWnn6 Ver.3をバンドルしている。リョービの商用フォント5書体と、WindowsとLinuxのデュアルブートを簡単に行えるSystem Commander Lite、多くのサウンドカードに対応するOpen Sound Systemも付属する。

■TurboLinux 日本語版 4.2

ベースとなっているTurboLinux日本語版が4.0から4.2にバージョンアップされた。カーネルのバージョン番号は2.2.9と変わらないが、セキュリティ対策のパッチが当てられ、チューニングも行われたようだ。

XFree86は3.3.1から3.3.4になり、Intel i740、SiS530/620、3Dfx Voodoo（banshee、Vodoo3）、S3 Trio3D、Matrox G400、nVIDIA RIVA TNT2、RAGE LT PROといった新しいグラフィックスカードに対応した。

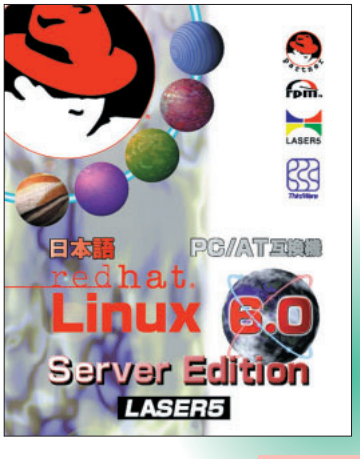
そのほか、MS-DOSフォーマットのフロッピーディスクをマウントせずに使えるようにするNtoolsと、MP3プレーヤソフトのxmmsを追加した。

名称	機能
Applix Words	日本語ワープロ
Applix Spreadsheets	表計算
Applix Presents	プレゼンテーションツール
Applix Graphics	グラフィックスツール（ドロー／ペイント）
Applix Mail	メール送受信
Applix Data	データベースアクセス
Applix Builder	カスタムアプリケーション作成
Filters	データ変換（MS Office、画像データ）

表1 Applixware機能表

日本語redhat Linux 6.0 Server Edition

▶ インターネット、イントラネットサーバを簡単に構成できる



日本語redhat Linux 6.0 Server Edition

価格:	4万9800円
問い合わせ先:	レーザーファイブ 03-5296-0670 http://www.cdrom.co.jp/

レーザーファイブから、日本語redhat Linux 6.0 Server Editionが8月24日に発売された。日本語redhat Linux 6.0 Server Editionは、英語版のRed Hat Linux 6.0をベースにサーバ専用の日本語インストーラを追加している。開発は、レーザーファイブとサードウェア（http://www.3ware.co.jp/）が共同で行った。

日本語redhat Linux 6.0 Server Editionの価格は4万9800円。180日間3件までのサポートを電話、FAX、電子メールで受けることができ、インストール以外の質問も受け付ける。それ以降は延長サポート（1年間5件、12万円）、追加サポート（2件につき5万円）などの有料サポートが用意されている。

■簡単インストール

インストール時に、「インターネットサーバ」と「LANサーバ」を選ぶことで、それぞれに必要な機能が自動的にインストールされ、**表1**のような設定ユーティリティを使って、メニューによる対話式で入力していくことで簡単にLinuxサーバを構成することができる。また、インストール終了後にコマンドラインから同じ設定ユーティリティを起動することができるので安心だ。

インターネットサーバの場合、IPマスカレードや、ファイアウォールとしての設定が簡単に行える。

LANサーバの場合、ファイル／プリンタ共有（Samba）を自動インストール。Sambaの設定では、WindowsNTサーバとの共存も考慮されている。

サーバではパフォーマンス、信頼性のためにRAIDハードディスクを使うことが多いが、日本語redhat Linux 6.0 Server Editionでは、**表2**のRAIDコントローラを自動認識する。ソフトウェアRAIDも設定でき、RAID0（ストライピング）、RAID1（ミラー）、RAID4、RAID5に対応する。

カーネル2.2.5を採用しており、複数CPUを搭載したシステムでのSMP（対称型マルチプロセッシング）に対応する。

付属のセキュリティマネージャでは、パケットフィルタリングによるファイアウォールを設定でき、Web、DNS、SMTPなどの必要なポート以外を塞ぐことが簡単にできる。

ネットワーク（ネットワークカード2枚差し対応、IPマスカレード設定を含む）
Samba（WindowsNT サーバとの共存を含む）
メールサーバ（中継専用サーバ設定を含む）
DNS（DNSレコードの追加設定可能）
DHCP設定
セキュリティ（SSHインストール、ファイアウォール設定を含む）
起動サーバ選択

表1　設定ユーティリティ

BRU 2000 backup soft
Phoenix　Adaptive Firewall
Objectivity/DB
Empress
Sybase Adaptive Server Enterprise
サイボウズOffice
iOffice2000
OfficeExchange
Active! mail
F-SECURE Anti-Virus
Sophos Anti-virus
System Walker
富士通 Fortran&C++
L@mail
FREQSHIP-mini UPS soft
APC UPS Daemon
Roxen Challenger Web Server
SSH

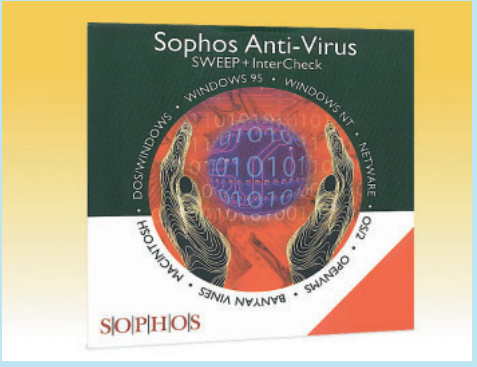
表3　バンドルソフトウェア

IBM ServerRAIDシリーズ
DPT SmartRAIDシリーズ
Mylex DAC960
Compaq Smart-2/P RAID Controller
AMU MegaRAID

表2　対応するハードウェアRAIDコントローラ

Products

ネットワーク時代のウイルス対策



Sophos Anti-Virus for Linux

大規模LANの全マシンに、アンチウイルスソフトをインストールして回るのは非常に大変だ。サーバで集中管理できるSophos Anti-Virusがあれば、手間をかけずにLAN全体をウイルスから守ることができる。

製品名	Sophos Anti-Virus
価格	Linux版は個人使用に限って無償。ファイルサーバライセンスは、9万9000円（サーバ1台、クライアント1〜10台）から
問い合わせ先	シー・エス・イー プロダクツ販売部 03-3463-5633 http://www.cseltd.co.jp/sweep/

ネットワーク対応のウイルス対策システム

Sophos Anti-Virus（以下、SAV）は、ネットワーク環境に対応したウイルス対策システムだ。ネットワークへの負荷を最小限に抑えた、集中的なウイルス管理を可能にする。

インターネットの普及によって、我々の生活や仕事は便利になったが、反面、ウイルスによる被害も増加しており、今や他人事として無視できないものとなっている。ウイルスは、毎月数百件に及ぶ新種、亜種が発見されており、ウイルスを特定するデータファイルは頻繁に更新する必要があるが、多くのマシンが接続された大規模LANでの更新作業は、非常に手間がかかる。

機能を分離して柔軟な運用が可能に

SAVは、ウイルス検索エンジンであるSWEEPと、ウイルスチェックの必要性を判断するクライアント・サーバシステムであるInterCheckとを分離することで、集中管理を実現している。InterCheckクライアントには、環境に応じてスタンドアローン型とネットワーク型の2種類がある。

スタンドアローン型は、コマンドの実行、マクロウイルスの危険性があるデータファイルへのアクセス、フロッピーディスクの挿入を監視しており、それらが起こるとSWEEPを起動してウイルスチェックを行い、検査結果をチェックサムファイルに保存する。同じファイルに再アクセスする際、チェックサムを照合して変更のないことが確認されれば、不必要なチェックは行われない（**図1**）。

対してネットワーク型は、ローカルに保存してあるチェックサムファイルに加えて、サーバ上のセントラルチェックサムファイルも参照する。

ここで、チェックの必要ありと判断すると、InterCheckサーバにファイルを転送し、サーバ上のSWEEPにウイルスチェックを行わせる（**図2**）。

サーバ上でウイルスチェックを行う場合は、ファイルの転送が起きるため、トラフィックの増加が心配になる。しかし、セントラルチェックサムファイルは、すべてのクライアントで共用されるため、あるクライアントが使用し安全が確認されたファイルは、ほかのクライアントからもチェックなしでアクセス可能だ。このように不要なファイル転送を抑えることで、ネットワークへの負荷を低減している。

また、SWEEPとInterCheckが分離されていることで、いろいろな形式での運用が可能になっている。大規模LANでは、ネットワーク型InterCheckクライアントを各クライアントマシンにインストールし、ウイルスチェックはすべてサーバで行うことで、クライアントマシンの性能低下を防ぐことができる。InterCheckクライアントのインストールやアップデートは、サーバから実行できるため、全クライアントマシンを渡り歩く必要はない。

ノート型マシンの場合、クライアント側にもSWEEPをインストールしておけば、社外で使うときもウイルスチェックが可能だ。対応プラットフォームは、サーバ、クライアントとも非常に多岐にわたるため、多種のマシンが混在した環境にも対応できる（**表1**）。

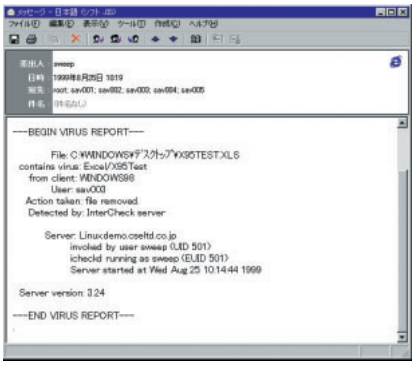
Linux／FreeBSD版を無償で提供

シー・エス・イーでは、SAVのLinux（Intel）版とFreeBSD版を、個人使用の場合に限って無償で提供しており、同社のWebサイトからダウンロードできるようになっている。無償提供版の場合、サポートは受けられない。しかし新種のウイルス発見後、すぐに発表される緊急データファイルは、製品版と同様に利用できるうえ、月に一度の頻度で新しい版が提供されるので、日々増え続けるウイルスへの対応という点では問題ない。

付録CD-ROMに、最新のSophos Anti-Virus for Linux（Intel）無償提供版を収録した。

無償提供版には、ネットワーク対応のInterCheckクライアントは含まれていないため、LAN上の他のマシンに対するウイルスチェックはできない。だがLinuxマシンでSambaを動作させ、Windowsマシンのファイルサーバとして利用しているようなSOHO環境では、Linuxマシン上の共有ディレクトリを定期的にウイルスチェックするといった使い方でも、かなり効果があるだろう。

もちろん、より厳密なウイルスチェックが必要な場合や、大規模な環境で管理コスト削減が求められる場合には、製品版が有効である。SAV for LinuxのInterCheckサーバは、一般的なLinuxのデーモンとして機能するため、今までの管理の一環としてウイルス管理ができる。たとえばログ記録はsyslogを利用するし、またウイルス発見時には管理者にメールで通知させることも可能だ（**画面1**）。



画面1　ウイルス発見の通知メール

サーバ	Linux(Intel、Alpha)、FreeBSD、WindowsNT(Intel、Alpha)、NetWare、OS/2、AS400、Solaris(SPARC、Intel)、ほか
クライアント	Windows3.1、Windows95/98、WindowsNT Workstation、DOS、Macintosh、OS/2

表1　対応プラットフォーム

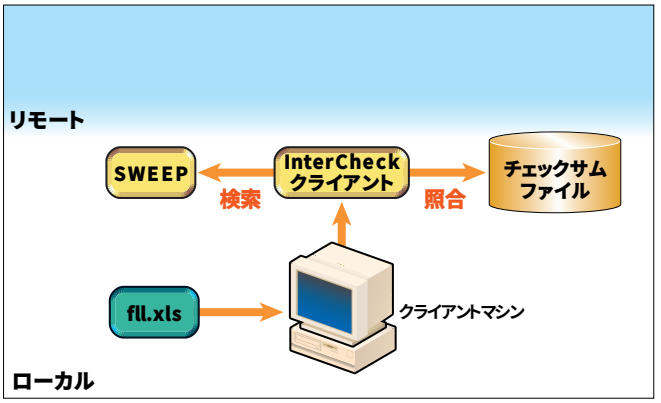


図1　スタンドアローン型InterCheckクライアントの動作

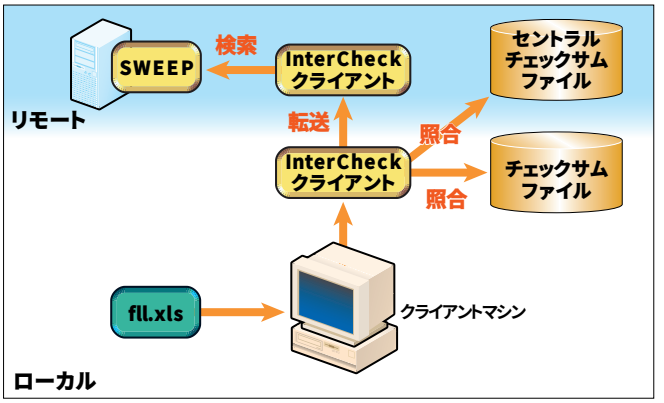


図2　ネットワーク型InterCheckクライアントの動作

サーバに最適なDual Pentium III搭載マシン

Personal Station DP500

Webサーバやファイルサーバとして Linuxを利用している人も多いだろう。ネットワークやクライアントマシンの性能が上がってくると、サーバの能力不足を痛感する。Pentium III 500MHzを2つ搭載した高速デュアルCPUマシンはそのような条件でも余裕をもって対応できる。

製品名	Personal Station DP500
価格	39万8000円
問い合わせ先	ぷらっとホーム 03-3251-2600 http://www.plathome.co.jp/

高速、デュアルCPUのサーバ向けマシン

ぷらっとホームから、ミドルタワーの本格サーバ向けマシン「Personal Station DP500」（以下DP500）が発売された。CPUにPentium III 500MHzを2個搭載したデュアルプロセッサのマシンで、メモリは256Mバイトを標準搭載している。そのほかのハードウェアスペックは**表1**をご覧いただきたい。

まず、ADAPTECのAHA-2940UWというSCSIインターフェイスを標準装備していて、ハードディスクとCD-ROMがSCSIで接続されていることに注目したい。

AHA-2940UWは、UltraWide SCSI対応で、最大40Mバイト／秒の速度でデータを転送できる。IDEインターフェイスでもUltra DMAいう規格では、最大33Mバイト／秒や66Mバイト／秒でデータを転送できるが、ドライブが対応しているだけでなく、OS側のサポートも必要である。

LinuxはUltra DMAに対応しているが、最新のカーネル2.2.xでも、IDEのデフォルト設定がPIO転送になっているように、まだ完全に対応しているとはいえないようだ。そういう意味でも、よく使われている安価なIDEではなく、SCSIインターフェイスのドライブを使用している点はサーバ向きである。

ハードディスクはIBMのDDRS-39130Wで、回転数が7200rpmと高速のものを使用している。CD-ROMは、PLEXTORのPX-40TWという40倍速のドライブだ。CD-ROMドライブでUltraWide SCSIに対応したものは珍しい（**写真3**）。

電源部は、250Wの2重化したリダンダント電源を使用しているため、もし片方の電源が故障しても本体に影響なく、通電中にホットスワップ交換が行える。通常のATX電源と同じサイズの場所にリダンダント電源が入っているのには驚いてしまった（**写真4**）。

なお、注文する際に必要に応じて、CPUをPentium IIIの550MHzに、メモリは最大1Gバイトまで拡張することができる。そのほかにオプションとして、モニタ、UPS無停電電源装

写真1
ケース内部のようす。Pentium IIIが2つ見える。ハードディスクはUltraWide SCSIの9Gバイトを搭載。

写真2
ケース背面パネル。電源部のほかに背面パネルにもファンが取り付けられていてCPUの冷却を助けている。

置、DATバックアップテープ装置、Accelerated-X商用Xサーバソフトウェアが用意されている。また、ケースはラックマウントタイプも用意されている。

OSは自分で選ぶ

DP500ではオペレーティングシステムがオプションとなっているが、Linuxのほかに、FreeBSD、Solaris、Windows 95/98/NT、BeOSの中から選ぶことができる。もっとも、DP500はデュアルCPUなので、Windows 95/98といったシングルCPUにしか対応していないOSを使う必然性はないだろう。

Linuxの場合は、Vine Linux 1.1CR、TurboLinux日本語版4.0、Slackware 3.6J、日本語redhat Linux5.2、英語版Red Hat Linux 5.2（あるいは6.0）というように、ほとんどのディストリビューションを選べる。

今回は、7月に発売されたTurboLinux日本語版4.0を、プレインストールしたDP500を試用した。

気になるデュアルCPUの性能は？

さて、DP500の簡単なベンチマークを行ってみた。CPUとハードディスクの性能を測るために、Linuxカーネルのコンパイル（make）時間を測定した結果が**表1**だ。

TurboLinux日本語4.0のLinuxカーネルは、/usr/local/linuxディレクトリに入っている。それを別の２つのディレクトリにコピーして、それぞれを別のコンソールからログインしmakeした。また、Makefileの“MAKE=make”行に、オプションとして-j3を付けて、2プロセス同時実行によるmakeも測定した。

1台のコンソールの時は、2プロセス同時実行の方が、1プロセスだけよりも1.8倍ほど速くmakeが終了する。2台のコンソールの時は、オプションの有無による違いはほとんどない。そして、1台のコンソールで1プロセスの場合（片方のCPUが遊んでいる）と比べて、わずか10秒ほど余計な時間で、2つ分のmakeを終えている。すなわち5%ほどのオーバーヘッドはあるが、2人で2台のシステムを使っているときと同様の仕事が行えるということだ。

このテストは、DP500をプログラム開発用サーバとして、複数のプロジェクトを同時にコンパイルしたときを想定したものである。結果を見ると、2人が同時に使ったとしても、ほとんどパフォーマンスが落ちていないのが分かる。

Webサーバやファイルサーバなどで、複数の人がコンテンツを共有して扱う場合には、1台のシステムで済ませたいが、性能の高いマシン、レスポンスの速いマシンが必要だ。そういう用途にDP500のようなデュアルCPUマシンが向いているだろう。

makeコマンド	コンソール1	コンソール2
make	4分12秒	なし
make -j3 2プロセス同時	2分19秒	なし
make	4分21秒	4分21秒
make -j3 2プロセス同時	4分23秒	4分23秒

表1　Linuxカーネル2.2.9のmake実行時間

写真4
電源部は２重化されていて、もし片方が故障しても通電状態のまま交換可能。

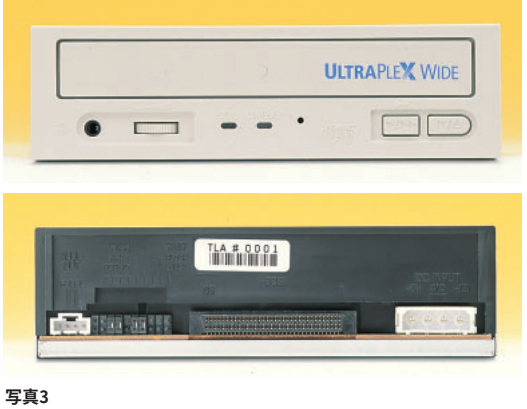


写真3
CD-ROMドライブは、一般的なATAPI（IDE）ではなくUltraWide SCSIで接続される。

CPU	Pentium III 500MHz×2
メモリ	256MバイトSDRAM（PC/100）
マザーボード	TYAN S1832DL Tiger100
ハードディスク	9Gバイト（UltraWide SCSI）
CD-ROM	40倍速 PLEXTOR PX-40TSi/UW（UltraWide SCSI）
フロッピードライブ	3.5インチ TEAC FD-235HG
ビデオカード	Matrox Millennium G200 AGP 8Mバイト SGRAM
ネットワーク	10/100Mbps PCI DEX21140-AF
SCSI I/F	ADAPTEC AHA-2940UW
キーボード	109日本語キーボード
マウス	Logitech MouseMan（3ボタン）
ケース	ミドルタワー SONGCHEER TQ-700RS
外形寸法	185（W）×480（H）×430（D）
電源	250W/ATX リダンダント
OS	オプション
価格	39万8000円

Personal Station DP500のハードウェアスペック

Webブラウザで Linuxの システム管理を 簡単に！

Free Products **HDE-WUI**

Linuxマシンでネットワークサーバを立ち上げたいが、viを使いこなして設定ファイルを変更できる管理者がいない。こういう場合でも、HDE-WUIを使えば、Webブラウザで簡単にサーバの設定が可能だ。

HDE-WUIは、ホライズン・デジタル・エンタープライズ（HDE）社が開発した、Linux上で動作する、GUIによるサーバ管理ツールである。HDE-WUIによって、ユーザー管理やデータのバックアップ、Web、POP、FTP等各種サーバ機能の起動・停止といったサーバ管理者の日常業務が、Webブラウザ上のグラフィカルな環境で行えるようになる。

Linuxは、サーバOSとして、「安定」「少ないリソースのマシンでも動く」といった特長を持つ。しかし、各種サーバプログラムの設定や条件変更は、テキストの設定ファイルをエディタで修正しながら行わなければならない。また設定ファイルの書式も、それぞれのサーバプログラムで異なっている。そのため、Linuxサーバの管理には、豊富な経験に裏打ちされた高い技術力が必要であった。HDE-WUIは、これら多岐にわたるLinuxサーバの管理を、統一されたGUI上で行うツールである。

対応ディストリビューションは、Red Hat Linux 5.X、 6.0、 TurboLinux 4.0、Vine Linux 1.1である。

■インストールは容易

HDE-WUIは、機能別に分割されたRPMパッケージで提供されているため、インストールはrpmコマンドを用いて、簡単に行える。パッケージの構成を表1に示した。wuiserverは、Apacheそのものであるが、HDE-WUI専用にポート番号6951を使用する。その上でHDE-WUIの本体であるwuiが動作し、さらにその上で各モジュールが動作している。セキュリティ保持のために、HDE-WUIを用いるシステムは、シャドウパスワード方式を使用しなければならない。

HDE-WUIを起動し、Webブラウザで http://ホスト名:6591/admin/にアクセスし、認証に成功すると管理用メニューが表示される。そこから「初期設定ウィザード」を選択すると、サーバマシンに関する初期設定が行える。画面1からも分かるように、Windowsマシンと同じ感覚で操作が可能だ。ヘルプが必要な項目には、「？」アイコンが付いており、マウスカーソルをアイコン上に移動すると、説明が表示される。

```
wuiserver（Apache）
└─wui
    ├─wui-admin（管理者用パッケージのベース）
    │    ├─wui-accesslog      Webサーバアクセスログ
    │    ├─wui-account        ユーザー管理
    │    ├─wui-backup         バックアップ
    │    ├─wui-maintenance    サーバの起動・停止、マシンの負荷チェックなど
    │    ├─wui-ml             メーリングリスト管理
    │    ├─wui-network        IPアドレス、ホスト名設定
    │    ├─wui-sysaccount     特殊ユーザー（root、bin等）管理
    │    ├─wui-syslog         ログファイル閲覧
    │    └─wui-wizard         サーバ設定ウィザード
    │
    ├─wui-service（サービスモジュールのベース）
    │    ├─wui-mod_atalk.init  Netatalk（AppleTalkファイル共有）管理用モジュール
    │    ├─wui-mod_dhcpd       DHCPサーバ管理用モジュール
    │    ├─wui-mod_ftp         FTPサーバ管理用モジュール
    │    ├─wui-mod_httpd       Webサーバ管理用モジュール
    │    ├─wui-mod_named       ネームサーバ管理用モジュール
    │    ├─wui-mod_sendmail    MTA管理用モジュール
    │    ├─wui-mod_smb         Samba（Windowsファイル共有）管理用モジュール
    │    └─wui-mod_tcpd        アクセス制限設定用モジュール
    │
    └─wui-user（一般ユーザー用）
```

表1:HDE-WUIの構成

画面1　セットアップウイザードのネットワーク設定

画面2　ユーザーアカウントの管理

■統一的管理が可能

初期設定が終わり、サーバ運用中に必要な操作も、管理用メニューから行う。新規ユーザーの登録を行うには、管理用メニュー左端の「ユーザアカウント」をクリックする（画面2）。同様に、「メンテナンス」から負荷状態、ディスク、メモリの使用状況の確認（画面3）、「システムアカウント」からrootなど特殊ユーザーのパスワード変更（画面4）、「システムログ」から/var/log以下にあるログファイルの閲覧ができる（画面5）。各種サービスの設定についても、現在の状態の確認や、始動、停止、次回起動時にサービスを動作させるかどうかの設定を、統一的に行うことができる（画面6）。

以上のようなサーバ管理業務以外にも、インストールされているrpmパッケージのリストアップ、各パッケージに含まれている情報の確認、パッケージの削除も行える（画面7）。

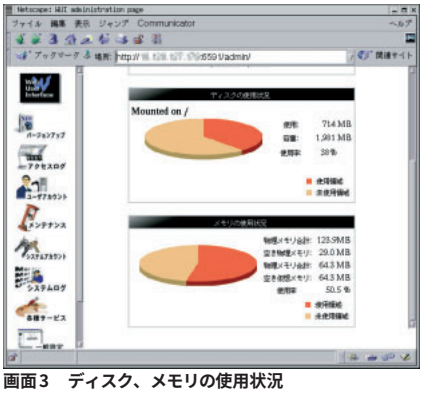
また一般ユーザー用パッケージ（wui-user）をインストールしている場合、http://ホスト名:6591/user/にアクセスし、認証に成功すると、画面8の一般ユーザー用の画面が表示される。ここでは、パスワードの変更とメールの転送先指定が行える。Linuxに不慣れなユーザーを抱える管理者にとっては、ありがたい機能である。

最近、HDE-WUIと同様の機能を持ったソフトウェアが海外でいくつか発表されているが、HDE-WUIには日本語の分かりやすいインストールマニュアルが提供されており、また画面表示も日本語であるため、導入・運用はスムーズに行えるであろう。このような「簡単にネットワークサーバを立ち上げ、設定・管理はWebブラウザで」という思想は、Linux magazine No.2で紹介したCobalt Qube2のそれと非常に似通っている。いわば、「ソフトウェアCobalt Qube」である。こういった思想の製品が現れるのは、やはりこの分野の需要が大きいためであろう。

インターネットサーバの管理に関しては、Windows NTはGUIを使えるが、リモート管理が難しく、Linuxはリモート管理が可能だが、テキストファイルを修正しなければならなかった。しかしHDE-WUIをLinuxと組み合わせることで、一般的なWebブラウザのインターフェイスを用いての、リモート管理が可能になった。「Windows NTの管理者をしているので、インターネットの理論的なことは分かるのだが、viやEmacsに慣れてないから、Linuxの管理はしたくない」というような人々に対して、Linux導入の敷居を下げる効果があるだろう。

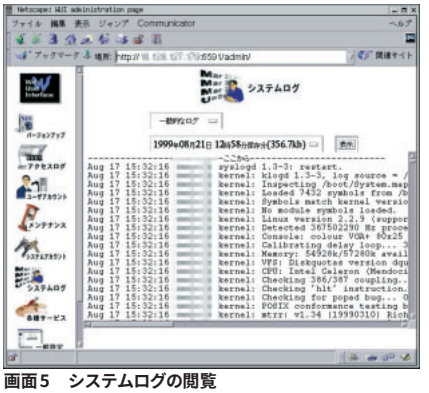
HDE-WUIの正式版は、9月上旬よりWebサイト（http://www.hde-wui.com/）から、無料でダウンロードできるようになる予定である。今回はベータ版のHDE-WUIをTurboLinux 4.0上で使用した。付録CD-ROMに、記事で使用したHDE-WUIを収録した。正式版とは一部異なる可能性があることをお断りしておく。

HDE社は、HDE-WUIをフリーソフトウェアとして公開し、さらに時期は未定だがオープンソース化も検討しているそうである。同社のオープンソース運動への姿勢と、HDE-WUIの能力とがあいまって、正式公開後には多くの支持を受けると思われる。



画面3　ディスク、メモリの使用状況

画面4　特殊なユーザーアカウントの管理



画面5　システムログの閲覧

画面6　ネットワークサービスの設定

画面7　rpmパッケージの管理

画面8　ユーザー用パスワード変更とメール転送の設定

Report

LinuxWorld Conference & Expo

文・写真：宮原 徹
Text • Photo : Tohru Miyahara

8月9日～12日、“LinuxWorld Conference & Expo”が、米San JoseのSan Jone Convention Centerで開催された。このイベントは、Linuxに関連のあるコンファレンス、チュートリアル、およびLinux関連製品やコミュニティの展示会からなる本格的なトレードショーとなっている。コンファレンスには、Linuxをはじめとするオープンソースソフトウェア開発者のコアメンバーが集まり、会場には自由闊達な雰囲気がただよった。

展示会は、出展した企業やコミュニティが158を数え、今年3月に同地で開催された前回と比べて倍の広さに拡大した。

以下、このイベントを訪れた日本オラクルの宮原氏が、出席した基調講演のようすと展示会場で目についたものをレポートする。

基調講演

動くモノがすべてを語る　Sean Moloney （Intel Corp.上級副社長）

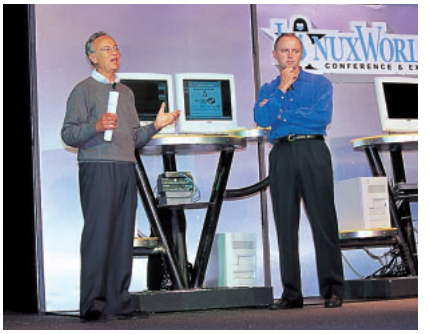
Moloney氏の基調講演は、いくつかのデモンストレーションを中心に進められた。「Google」（http://www.google.com/）は新しいタイプの検索エンジン。Yahoo!のようにキーワードにマッチするリンクがたくさん出てくるのではなく、キーワードに最もふさわしいと思われるページへ直接ジャンプする「I'm feeling lucky」ボタンが特徴。デモンストレーションでは、「Simpsons」「yeah baby」「Linus」などのキーワードで検索してみせた。どんな検索結果になるかは、自分で試してほしい。

このほか、X Window Systemでのデュアルモニタサポート、MP3、16ノードクラスタマシンを使ったCGレンダリング、クラスタ管理システムが披露された。

最後のデモンストレーションには、Intel社のAndrew Grove会長が登場し、IA-64の性能をPR。IA-64で動作するLinuxの応用例として高性能なWebサーバの可能性を示した。デモンストレーションでは、Pentium IIIでIA-64エミュレータを動作させ、その上でLinux on IA-64を動作させたが、結果的にわかりにくいものになってしまった。

デモンストレーションにはVA LiNUX Systems（http://www.valinux.com/）がかなりの協力をしていたもようで、何人かの技術者をデモンストレータとして壇上にまで送り込んでおり、まるでVA LiNUX Systemsの基調講演のようだったのも印象的だ。


Intel社上級副社長Sean Moloney


Intel社のAndrew Grove会長も登場してLinux on IA-64をPR。

カーネル2.4はクリスマスプレゼント？ Linus Torvalds

まず最初に壇上に上がったVA LiNUX SystemsのLarry Augustin氏が会場に「この中でオープンソースソフトウェアにコントリビュートしたことがある人は？」と問いかけると、数十名の手が挙がった。立ち上がった彼らに対して、会場から惜しみない拍手が送られた。さらに「この中でMicrosoftのソフトウェアにコントリビュートしたことがある人は？」と問いかけると、失笑が漏れた。オープンソースムーブメントを歓迎するこうした雰囲気の中、Linus氏が招かれて登場した。

Linus氏は技術的な話をするのが一番エキサイティングだから、と切り出して、まずカーネル2.2.11のリリースを報告した。今回から管理がAlan Cox氏に引き継がれ、「自分はうまくリリースできるようにと祈っただけ」とおどけて見せた。

Linus氏の目下の作業は次の安定バージョンであるカーネル2.4であり、数週間以内に仕様をリリース、4カ月後の12月を目標にソフトウェアをリリースするもよう。カーネル2.0から2.2までのリリース間隔よりかなり短期間だが、これは大幅な変更をしないで、安定と高速化、2.2で積み残した課題に絞ったためということであった。順調に進めば、楽しみなクリスマスプレゼントになるだろう。

カーネル2.4で注目したいのは、USBのサポート強化である。現在のPCはほとんどがUSBインターフェイスを備えており、ノートPCではUSBしかないものもある。当面サポートされるのは、キーボードやマウス、カメラやスキャナといった基本的な周辺機器のみだが、

技術的な話をするのが一番エキサイティングだから、と切り出したLinus Torvalds氏。

いずれにせよ歓迎すべきことである。

続くQ&Aセッションでは、踏み込んだ内容の質疑が数多くやり取りされた。「キーノートスピーチ」らしからぬ光景かもしれないが、これもまた、ハッキング好きなオープンソースソフトウェアの雰囲気にはぴったりかもしれない。

Free Software Foundation、コミュニティ賞を受賞

最後に、コミュニティの貢献者に贈られる「コミュニティ賞」が発表された。今回は、フリーウェアの先駆者であるFree Software Foundationが選ばれた。壇上に上がった代表のRichard Stallman氏は、いつものようにGNUの始まりとフリーソフトの意義を語り始めたが、壇上ではしゃぎまわるLinus氏の子どもたちに毒気を抜かれたか、話をほどほどに切り上げて賞金の2万5000ドルを受け取った。

賞金2万5000ドルの特大の小切手を受け取るFree Software FoundationのStallman氏（右端）。

パネルディスカッション いかにしてオープンソースソフトウェアを共同開発するか

VA LiNUX SystemsのLarry Augustin社長をモデレーターに、Linux、Samba、Apache、XFree86、FreeBSDなど有力なオープンソースソフトウェアのコア開発者が集まって、開発手法や考え方をテーマにパネルディスカッションが開催された。

「どのようにして複数のプログラマーが開発を進めているのか」について、Linus氏は「（結局のところ）お互いが信頼しあって開発を進めている」と述べた。前日の基調講演では「今まではうまくいかなかったが、カーネル2.4では大勢の開発者による共同開発で混乱をきたさないように、ソースコントロールをきちんと行いたい」といったことを発言していたが、それ以上に大切な「気持ち」の問題を言い表していたのではないだろうか。

Linus氏を始め、Samba、Apache、XFree86、FreeBSDのコア開発者が一同に会したパネルディスカッション。

Corel

Corelブースでは、「Corel LINUX」（3月の発表時はCorel Desktop Linux）がお目見え。親しみやすいGUIが特徴で（写真右上）、Qtで作られたインストーラはWindowsと見紛うばかり。入力が必要なのはユーザー名やIPアドレスなど4項目だけだ。起動後の操作環境も、KDEをベースに環境設定用のコントロールセンターなどが追加され、WindowsユーザーならLinuxを知らなくてもすんなり使えそう。当初はIntel版が提供され、その後NetWinder用にStrongARM対応版がリリースされるという。
http://www.corel.com/

Caldera Systems

Caldera Systemsはさきごろ、SPARC版「Open Linux」をリリースした。また、MCG（Motorola Computer Group）、LINEO社（旧Caldera Thin Clients社）と共同で組み込み系のLinuxソリューションを発表しており、この分野のビジネスをリードすると目される。Motorolaとの関係から、当然PowerPCにも対応すると思われ、Open Linuxは、Intel、SPARC、PowerPCの多様なアーキテクチャに対応しそうだ。写真は、ブースで催された現金つかみ取り。
http://www.calderasystems.com/

Stormix Technologies

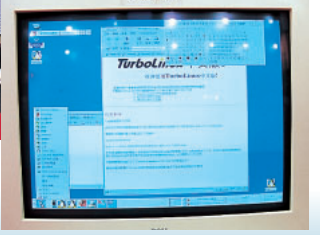
Stormix Technologiesは、リリース前のまだ無名のディストリビューション「Strom Linux」を出展。開発はカナダで行われているが、コアメンバーに日本から池田篤司氏が加わっている。ちょうどブースにいた同氏の説明を聞くことができた。Strom Linuxは、Debianパッケージを使用し、インストーラにはGTKを使用、ターゲットはエンドユーザーだという。年内に英語版を出荷、日本語にもすぐに対応できるように現在開発中で、来年には日本語版を出荷したいとのこと。
http://www.stormix.com/
（池田氏は日刊アスキーLinuxにコラムを掲載中：
http://www.ascii.co.jp/linux/）

SuSE

SuSEは、最新版の「SuSE Linux6.2」を展示。SuSEといえば、ヨーロッパで普及しているディストリビューションの感があるが、米国でも着々とパートナーを増やしているようだ。日本への進出も伝えられているだけに、今後の動向に注目したい。
http://www.suse.com/

TurboLinux

TurboLinuxは、クラスタサーバを中心に展示。日本語版、中国版の「TurboLinux Workstation 4.0」も並ぶなか、英語版の展示が3.6だったのはなぜだろうか。他社のソフトウェアにTurboLinux対応の記載があるものが増えてきており、前回と比べて確実に認知度が上がっていると感じられた。
http://www.turbolinux.co.jp/



Red Hat Software

Red Hat Softwareブースでは、特に目を引く新しい展示や発表はなく、株式公開ばかりが話題になった。Bob Young氏を始めとする同社の幹部が会場を訪れることもなく、記者会見も開かれなかった。これでは来場者に、コミュニティを放ってマネーゲームに奔走しているかのような印象を与えかねないと思う。株価は加熱しているようだが、かなり醒めた目で見られていたのではないか。
http://www.redhat.com/

展示会場 Hardware

VA LiNUX Systems

VA LiNUX Systemsは、社長のLarry Augustin氏が今回のイベントの実行委員長を務め、会場に用意された来場者用のメール用端末（同社のローエンドマシン）すべてを提供している。展示ブースでは、ラックマウント型のサーバ機も展示され、充実したラインナップぶりを見せた。
http://www.valinux.com

Compaq Computer

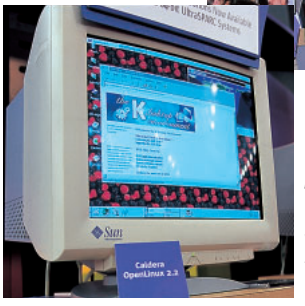
Compaq Computerは、Alpha、Intelアーキテクチャの幅広いソリューションを展示していたが、特にEコマース向けの高可用性、高パフォーマンスのサーバソリューションに力を入れていた。この分野は買収したDECのAlphaマシンの戦略と重なるが、今後Linuxが大きな位置を占めると推測されるだけに、これからの動向が注目される。
http://www.compaq.com/

SGI

SGIは、O2などのグラフィックワークステーションで有名だが、今回のイベントに合わせて企業向けLinuxサーバを発表した。ファイルサーバ、インターネットサーバ、データベースサーバなど、ローエンドからミドルレンジをターゲットにしたLinuxビジネスを展開することになりそうだ。
http://www.sgi.com/

Penguin Computing

Penguin Computingのマシンは「エンペラー」「アデリー」「ロックホッパー」などペンギンシリーズの製品名になっている。今回は、AMDのAthlonを使った「キング」シリーズを発表した。会場では、会期中に発表された最新CPU Athlon 650MHzを使ったLinuxシステムをデモンストレーションした（写真右）。AthlonのLevel2キャッシュは最大8Mバイトで、IntelのXeonプロセッサより大容量だ。これがサーバでどの程度のパフォーマンス差となるのか、大いに興味を引かれる。
http://www.penguincomputing.com/



Sun Microsystems

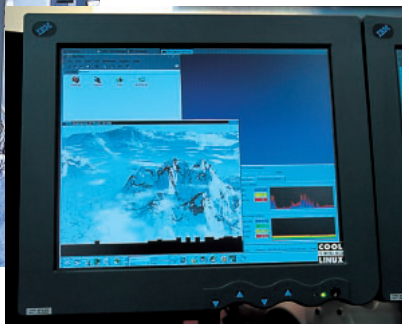
Sun Microsystemsは、SPARC用のRedHat Linux、Caldere OpenLinuxを展示したほか、LinuxとSolaris間のソースコード／バイナリ互換性のデモンストレーションを行った。Linuxについていえば、SPARC以上にJavaに興味があるのだが、今回はJavaに関する特別な発表や展示もなく、少し残念だ。
http://www.sun.com/

サイバネテック

日本から出展したサイバネテック社は、小型Linuxサーバ「PathFinder」「PathExplorer」を出展。もともとNTPサーバとして開発されただけあって、非常にコンパクトで、プリンタサーバボックスのような感じだ。

PFU America

PFU Americaは、おなじみ「Happy Hacking Keyboard」の普及版HHK Liteを特価59ドルで販売。インターフェイスはPS/2のみで、DIPスイッチによるキーアサインの変更などをサポートしている。このほか、CELL COMPUTING社のPCMCIAカードと同サイズのマザーボードの展示もあって、こちらはオンボードでMMX Pentium 233MHzを搭載。これなら世界最小のクラスタマシンが作れそう。
http://www.pfuca.com/



IBM

IBMブースで目を引いたのは、マシンではなく、Lotus Notes/Domino for LinuxのSneak Preview版だろう。会期に合わせてインターネットで配付を開始したこともあり、かなりの来場者が関心を寄せていた。現在はデータベースアクセス機能とORB機能がないが、製品版では他のプラットホームと同様に実装される。サーバ用途のキラーアプリケーションとなり得るだけに、製品版の早い出荷が期待される。
http://www.ibm.com/

TreLOS

TreLOSは、Linux上でWindowsを動作させる「Win4Lin」を出展。VMwareと比べて、Windows95/98に特化している分、非常に高速である。Windowsのファイルを直接Linuxのファイルシステムにインストールし、シェルから「win &」と入力すると、Windowsが独立したウィンドウで起動する。現在は英語版のみで、8月下旬からβ版を同社のサイトで配付、9月末に50ドル程度で出荷の予定とのこと。日本語版の開発も検討中だが、発売時期や価格などは未定。
http://www.trelos.com/

Cygnus Solutions

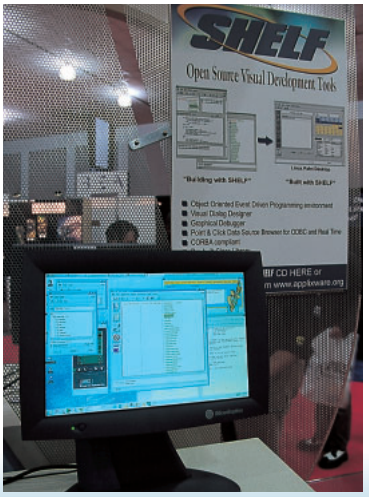
Cygnus Solutionsは、GNUソフトウェアのソリューションビジネスを提供している会社だが、今回は統合開発環境「CODE FUSION」をリリースした。高速なC/C++コンパイラとデバッガ、クラスブラウザなどが統合されており、Oracleなどの開発にも利用されているとのことであった。
http://www.cygnus.com/index.html

Metrowerks

Metrowerksは、統合開発環境「Code Warrior」をデモンストレーション。これまでのRedHat版に加えて、SuSE版もリリースした。しかし、どちらもGCCを利用するGNU版の展示だけで、同社独自のコンパイラを組み入れたProfessional版がなかったのは残念。
http://www.metrowerks.com/

AbiSource

AbiSourceは、オープンソースのワードプロセッサ「AbiWord」の開発元である。今回はSourceGearというスポンサーがつき、コミュニティスペースではなく大きなブースを構えての出展となり、驚かされた。今後、両者がどのような関係で提携していくのか詳細は不明だが、オープンソース活動のひとつのモデルとして面白いと思う。AbiWordは最初からクロスプラットフォームを意図し、ブースではiMacでも動作していた。
http://www.abisource.com/

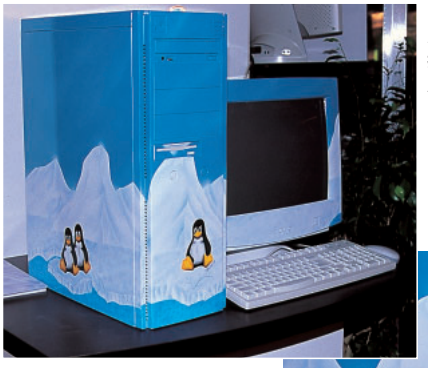


Applix

Applixは、オフィススイートの「Applixware」で知られるが、今回注目されたのは統合開発環境「SHELF」。MetrowerksやCygnus Solutionsと違うのは、GUIデザイナーを備えていること。ウィンドウにボタンやテキストボックスなどを配置し、それぞれのオブジェクトにCのコードを記述できるので、感覚的にはVisualBasicに近い。3Com社のPalmとPCを連動させる「PalmDesktop for Linux」の開発にも使われ、かなり本格的なソフト開発にも利用できるようだ。
http://www.applix.com/

展示会場 Amusement

パネルディスカッションに「本物の」ペンギンが参加。白熱した議論もしばし休みになるのでした。



なんとも涼しげな筐体のマシンを発見。エアブラシで南極のペンギンが描かれている。売り物かと思ったら、展示用の1点ものでした。残念。（Alpha Computersブースで）

前回は新型ビートルの抽選会を催したLinuxCare社だが、今回はバン。残念ながら、応募資格が米国在住なのでした。（LinuxCareブースで）

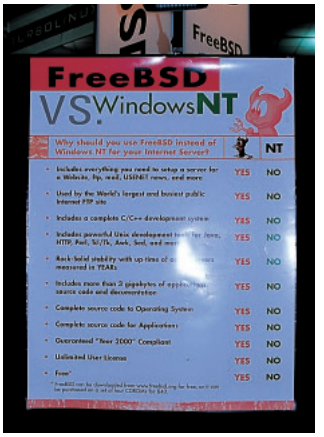
客寄せのロボット。どうやらどこかで操作しているらしいが、作りも動きもナイスな感じ。AIBOもいいけど、こんなの1台欲しい？　（SGI社のブースで）

あれ？　前回と同じ小人（GNOME）が寄付を求めてますよ。（Free Software Foundationのブースで）

場内を歩き回るペンギン着ぐるみ其の壱。なんだか見た目がアニメのペ○ペンに似てはいないか？　（Sybaseのブースで）

子豚さんが出迎えてくれるBRICK House社のブース。なぜ子豚かというと、セキュリティ重視のWebサーバを「三匹の子豚」の「レンガの家」になぞらえているのでした。

ペンギン着ぐるみ其の弐。胸には「Good morning, Mr.Gates. I'll be your server today!」のメッセージ。某社のコピー「Where do you want to go today?」に対する答えになっている。（Penguin Computingブースで）



FreeBSDとWindows NTに関する11のYESとNO。もちろん、NOばかりなのは……。（FreeBSDのブースで）

実

ディストリビューション

Caldera OpenLinux 2.2の全貌

本国では死ぬほどメジャーなのに、日本ではナゼかウケないミュージシャンは多い。そういう人が来日するときは、決まって「最後の大物」とか「玄人好みの実力」とかいった言葉で形容されるものだ。そして、日本のLinux界にもやってきたのだ、「最後の大物」が!! しかも、ド派手なインストーラ、ビジュアルなデスクトップ画面、わかりやすすぎる管理ツール、高機能アプリケーションの搭載で、ちっとも「玄人好み」じゃない。そう、わかりやすくて、凄い機能満載なのだ。どうやら日本語も通じるらしい。名前は「Caldera OpenLinux 2.2」。その実力のほどをとくとご覧いただきたい。

Caldera OpenLinuxとは？

Caldera OpenLinux（以下、OpenLinux）は、日本国内での知名度はお世辞にも高いとはいえないが、米国ではRed Hat Linuxに次ぐシェアを持つといわれる人気ディストリビューションである。さらに前バージョンのOpenLinux 1.3が、米*Network Computing Online*の「1999 Well-Connected Award」において、Sun MicrosystemsのSolaris 7やNovellのNetWare 5を抑えて「Best Network Operating System」に選ばれるなど、その評価は高い。OpenLinuxはどんなディストリビューションなのか？　また、この人気の秘密は何か？　では、この未知のディストリビューション、OpenLinux 2.2が持つ実力の全貌を探ってみよう。

What's OpenLinux 2.2?

OpenLinuxは、米国Caldera Opensystemsによって'95年10月より開発・販売・サポートがなされているLinuxディストリビューションで、日本国内では、アミュレットが販売と日本語環境のサポートを行っている。最新バージョンは'99年4月にリリースされた2.2で、Linux kernel2.2とglibc 2.1、そして統合デスクトップ環境にKDE1.1を採用している。また、パッケージ管理方法にRPMを用いているため、Red Hat系と思われるかもしれないが、インストーラ、管理ツールなど多くの独自性を持ったディストリビューションである。

写真　Caldera OpenLinux 2.2
価格：9,800円
販売代理店：アミュレット（http://www.amulet.co.jp/）

■いち早いKernel 2.2とglibc 2.1のサポート

Linux kernel 2.2がリリースされたのは記憶に新しいところだ。本格的なSMPのサポートなど、Linux kernel 2.2は多くの機能が実装された最新の安定版カーネルである。Red Hat Linux 6.0のリリースによって初めてLinux kernel 2.2に触れた読者も多いだろう。しかし、実は最初にLinux kernel 2.2を組み込んで製品リリースを行ったのは、このOpenLinux 2.2であり、米国での注目度も高かった。glibc 2.1についても同様で、日本語ディストリビューションの多くが、アプリケーションなどの日本語環境の互換性などの理由からglibc 2.0を利用しているが、OpenLinuxはいち早くglibc 2.1をサポートしている。

■一貫したKDEのサポート

現在では、KDEを標準のデスクトップ環境としてサポートしているディストリビューションは少なくない。しかし、ほんの1年前は、SuSE Linux（欧州で人気の高いディストリビューション）とLinuxPPC、そしてこのOpenLinuxくらいしかKDEをサポートしているディストリビューションはなかった。このような一貫したKDEサポートの経験により、KDEインターフェイスはOpenLinuxの中で自然になじんだ環境となっている。

■美しく、使いやすさを追求したインストーラ

今回、OpenLinuxを評価してみて最も驚かされたのが、「LIZARD」と呼ばれるインストーラである。詳細については、次ページからの紹介記事を参照してほしいが、GUIツールキットとしてQtを利用したこのインストーラはとにかく美しく、簡単なのだ。これまで持っていたLinuxインストーラへのイメージをまったく覆してしまうほどの出来で、Windowsアプリケーションとも引けをとらない。また、Windowsユーザーが同じディスク内にOpenLinuxをインストールすることを考慮して、PartitionMagicやBootMagic（いずれも機能限定版）を同梱するなど、さまざまな工夫がなされている。

■独自のGUI管理ツールを搭載

OpenLinuxは、管理ツールとして独自のGUIツール「COAS」を利用するようになっている。このCOASは複数のモジュールからなる管理ツールの総称で、管理に必要な機能の多くをGUIベースで設定することができるようになっている。

では、これらの特徴の中でも特にOpenLinux独自の部分を中心に次ページから紹介していこう。

驚異のインストーラ「LIZARD」

ディストリビューションの評価において、インストーラの占める割合は大きい。魅力的なアプリケーションパッケージや初期設定の豊富さなどももちろん重要な要素だが、それらはユーザーの好みなど主観的な部分が大きい。何といっても、（バージョンの違いこそあれ）同一のカーネルを使い、動かしているアプリケーションの多くがGNUツールという現状では大きな差は生まれにくい。依然として、Linuxの壁はインストールなのである。では、このOpenLinuxのインストーラはどうだろうか？

美＋易＝「LIZARD」

前バージョンまでのインストーラ「LISA」（Linux Installation & System Administrator）から一新されたインストーラ、「LIZARD」（LInux wiZARD）はOpenLinux 2.2の目玉といえる強力なインストーラである（LIZARDが起動しない環境用にLISAも利用可能）。LIZARDの特徴は、端的にいえば次の2つに絞れるだろう。

■Qtを用いた美しい画面

まずは、**画面1**を見てほしい。これは、Xの設定が終わってからのインストール画面ではなく、ブート直後からこのようなグラフィカルな画面でインストールが行なえるのである。これは、KDEで利用されていることで有名なGUIツールキット「Qt」を利用したもので、起動時のアニメーションも含めてかなり美しい出来である。これは、カーネルが読み込まれるまでは、専用のグラフィック描画ルーチンを用いて画面を作り、カーネルがロードされたあとは、XのVGAサーバーを利用してQtを呼び出していると予想される。

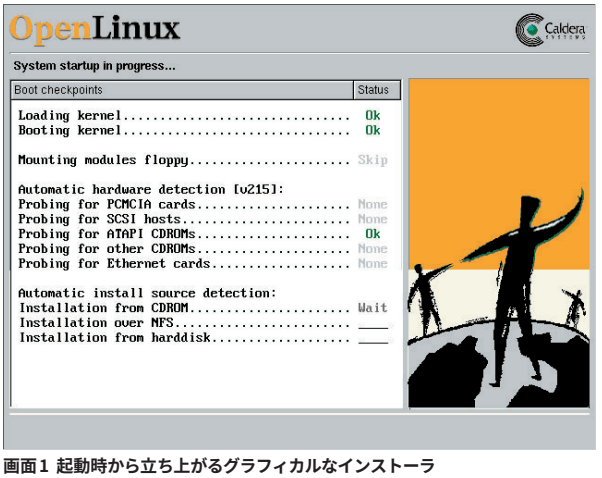
■設定項目を極力減らし、簡単インストール

ビジュアル要素に加えて、初心者がインストールにつまずかないようにするため、ユーザーの入力項目を極力減らしている。入力が必要な項目で主なものは、次のとおりである。

- **ディスクのパーティション設定**
- **キーボード設定**
- **ディスプレイ設定**
- **rootのパスワード設定**
- **一般ユーザー登録**
- **ネットワーク設定**

これらのうち、キーボード設定やrootのパスワード設定などは知識を要求されるものではないし、ネットワーク設定もDHCPを利用できる環境なら難しくはない。したがって、多少の知識が要求されるのはパーティション設定（**画面2**）くらいで、悩むところはほとんどない。

以上がLIZARDの主な特徴だが、ハードウェア検出などインストーラに求められる機能は全て備えている。おそらく足りない機能は日本語表示くらいではないだろうか（英語のほかに、ドイツ語、フランス語、イタリア語をサポートしている）。このあたりは今後に期待したいところだ。もっとも、GUIで直観的に操作できるため、一度Linuxのインストール経験のある人なら、英語表示であってもさほど戸惑うことなくインストールできるだろう。



画面1 起動時から立ち上がるグラフィカルなインストーラ

画面2　パーティション設定画面

■工夫されたインストール待ち時間

LIZARDでは、インストールの待ち時間にテトリス風のゲームをやって時間をつぶすことができる（**画面3**）。あまり魅力を感じない人もいるかもしれないが、こんな細かいところまで配慮されているのはうれしい。何よりもこういった配慮こそがこれまでのLinuxに欠けていた部分だろう。

Windowsからのインストールも考慮

OpenLinuxはビジネス利用を強く打ち出したディストリビューションで、現在のビジネス環境で多くのシェアを持つWindowsとの共存についてもよく考慮されている。具体的には、Windowsアプリケーションのような感覚でOpenLinuxをインストールできるのだ。

WindowsマシンのCD-ROMドライブにInstall CD-ROMを入れると、AutoRun機能により**画面4**のインストール画面が起動する。ここから、PowerQuest（http://www.powerquest.com/）が開発したディスク管理ツール「PartitionMagic」の機能限定版がインストールされ、Windowsが使用しているFAT16やFAT32領域から、GUI操作でLinuxが利用するext2領域を作り出すことができる。ext2領域ができてしまえば、あとは通常のインストールと同様の手順で、OpenLinuxを新しく作り出したext2領域へインストールできる。

なお、WindowsとOpenLinuxの起動の切り替えは、Windowsからのインストール手順の最後にインストールされるPowerQuestのブートセレクタ「BootMagic」（機能限定版）によって行う。

LIZARDの課題は？

さて、ここまでLIZARDの利点を説明してきた。では、LIZARDには問題点はまったくないのだろうか？　結論からいうといくつかある。今回の評価中に感じた不便なところをいくつか次に列挙してみよう。

■日本語表示ができない

LIZARDは、日本語表示の機能を持っていない。これは、海外のディストリビューションなので仕方のないことかもしれない。しかし、LIZARDは多国語対応のインストーラで、ベースとなっているQt2.0もマルチバイト対応なので、実装はそれほど難しくないはずだ。日本語表示の機能はぜひともほしい。

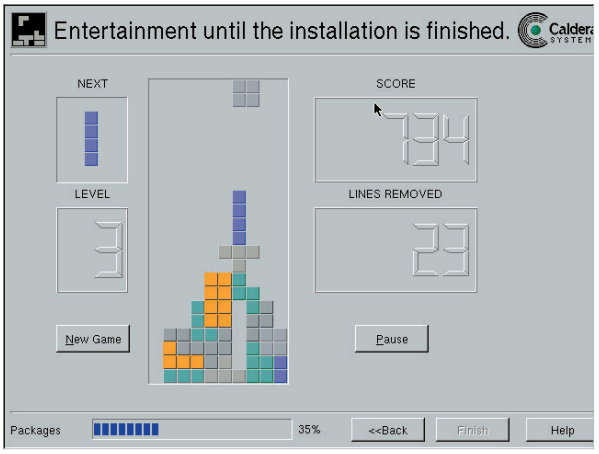
■LILOをどこに書き込むかわからない

LIZARDではLILOをどこに書き込むか質問してこない。これは、ディスクに関する知識がない人には難しい質問であり、怖い作業でもあるので、このあたりを考慮しなくてもいいようにという初心者向けの配慮なのだろう。

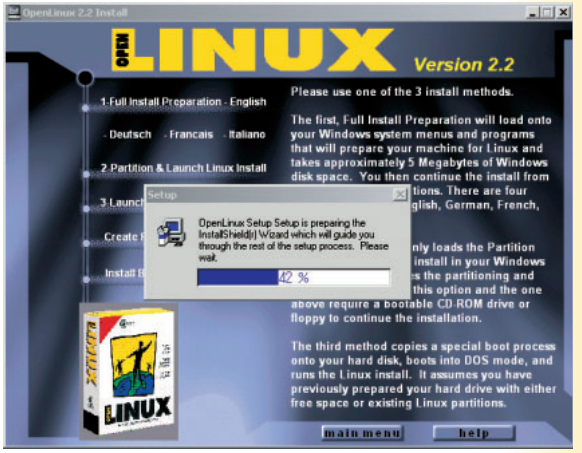
しかし、1台のハードディスクに複数ディストリビューションをインストールして、ブートセレクタなどでOSを切り替えている人には、この指定がないとかえって不安になる。しかも、緊急ディスクも作成しないので、場合によっては、インストール後により複雑な作業を強いられることもあるだろう。どうやらLinuxパーティションの先頭セクタに書き込まれるようだが、このあたりはオプションで選べるようにしてほしいところだ。ただし、Windowsと共存するインストールを行ったときは、BootMagicがMBRにインストールされるので問題はない。

■やはり重い

インストール時間が長く、重い感じがするのは否めない。ただ、このような人のために前バージョンまでのインストーラ「LISA」も残されている。古いマシンにインストールするときなどは、一度考慮してもいいだろう。



画面3　インストールの待ち時間にできるゲーム



画面4　Windowsからの自動起動

使いやすさを追求したデスクトップ環境

OpenLinuxのインストーラLIZARDがいかに今までのLinuxインストーラの常識を覆すものかは前節で述べたとおりである。当然ながら、ユーザーの使いやすさを追及した設計思想は、インストール後の使い勝手の中にもいたるところに反映されている。

具体的には、一般ユーザーにはKDE 1.1、管理者にはGUIベースの管理ツール「COAS」が提供されており、旧来のLinuxの作法に不慣れなユーザーでも戸惑わずに利用できるようになっている。

ここでは、このKDEとCOASについて触れる。なお、KDEについては、すでにLinux magazine No.2の32ページからの特集「KDE vs. GNOME」で詳しく解説したので、ここでは概略だけを示す。

直観的なインターフェイス「KDE」

KDE（the K Desktop Environment）は、GUIツールキットとして「Qt」を採用した統合デスクトップ環境で、OpenLinux以外にも欧州で人気の高いディストリビューション「SuSE Linux」などにも標準のデスクトップ環境として採用されている。

最近では、「Red Hat Linux 6.0」や「TurboLinux 日本語版4.0」といった最新ディリビューションでも採用されるなど、KDEは人気が高まりつつあるが、OpenLinuxは以前のバージョンからKDEを標準採用しており、システムとの一体感も高い。

今回紹介するOpenLinux 2.2の搭載されているのはKDE 1.1（後述の「かぼす」をインストールするとKDE 1.1.1となる）で、次のような特徴を持っている。

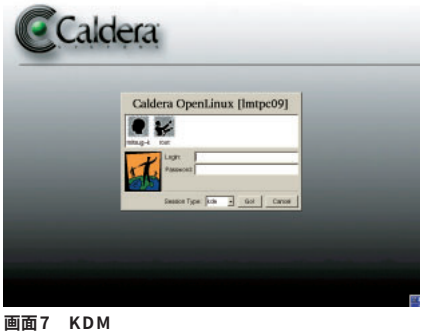
■Windowsライクなインターフェイス

画面5を一見してわかるとおり、スタートメニューのようなメニューからアプリケーションを起動するところや画面下のパネルは、かなりWindowsを意識した作りとなっている。

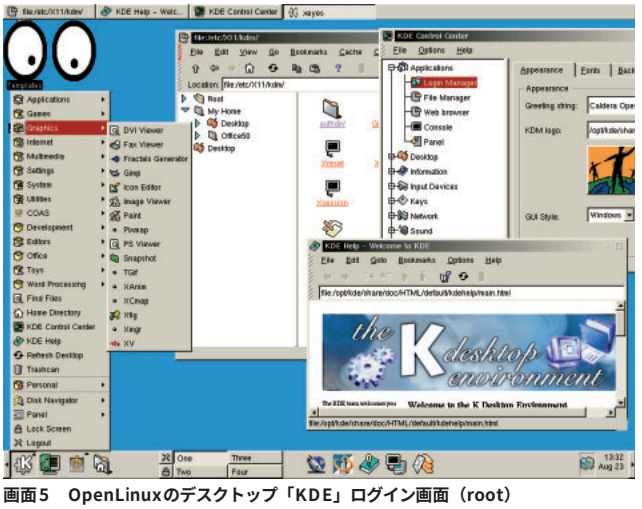
さらに、システム設定を行うツール「Control Center」も、壁紙の変更方法などWindowsと酷似している。Windowsユーザーなら、おそらくすぐに使いこなせるはずである。

■テーマウィザードによるLook & Feelの変更

OpenLinuxでは、各ユーザーの初回ログイン時に「The KDE Wizard」が自動的に起動され、ここからテーマ選択やアイコン設定などが行えるようになっている（KDEメニューの［Utilities］-［KDE Configuration Wizard］を実行しても起動する）。ここから選択できるテーマには、KDE標準テーマのほか、Windowsテーマ、MacOSテーマ（**画面6**）、BeOSテーマ



画面7　KDM


画面5　OpenLinuxのデスクトップ「KDE」ログイン画面（root）


画面6　MacOS風のデスクトップ

の4つから選択できる。ただし、お世辞にも似ているとはいいがたい。これは、KDEのテーマ設定がタイトルバーやデスクトップの背景の配色程度しか変更できないためである（GNOMEはGUIツールキットレベルでテーマに対応している）。なお、テーマについての詳細は、http://kde.themes.org/を参照してほしい。

■KDMの採用

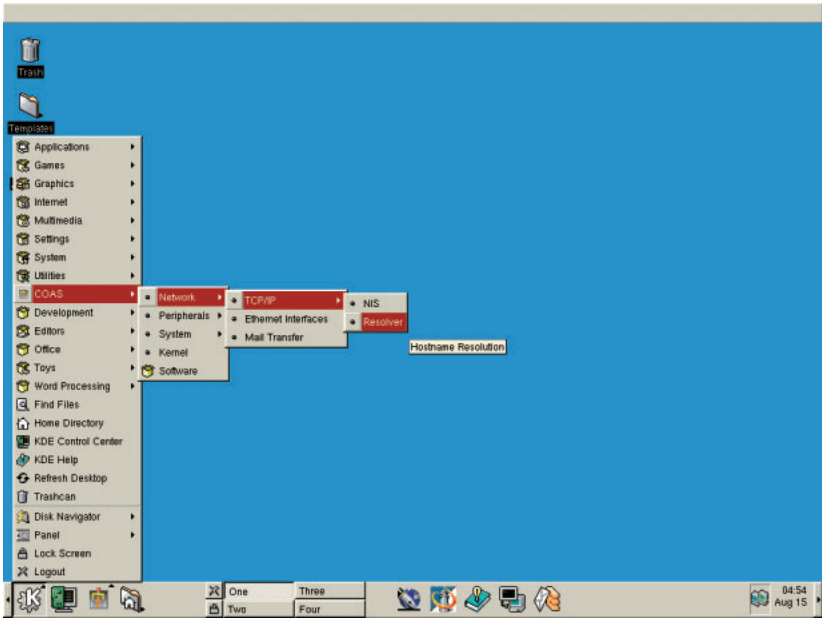
OpenLinuxは、標準でグラフィカルログインを採用している。このログイン画面もXDMではなく、KDEによって提供されるXDMの代替モジュール「KDM」（**画面7**）を利用しており、美しいログイン画面が現れる。

GUI管理ツール群「COAS」

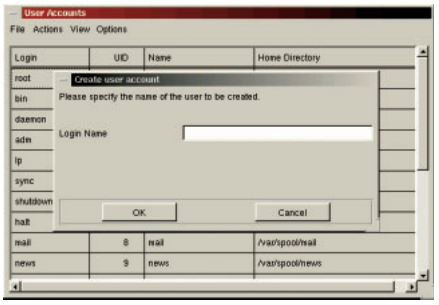
OpenLinuxは、管理ツールもGUIベースのものを提供し、クライアントだけでなく管理者にも使い勝手の向上を図るような配慮がなされている。これは、「COAS」（Caldera Open Administration System）と呼ばれるもので、**表1**に示したような管理機能を提供してくれるGUIモジュール群で、米国で人気の高いスクリプト言語「Python」で記述されている。COASは、機能ごとのモジュールに分かれており、linuxconfのような統合的なコンテナ環境がないため、必要な設定ごとに対応するモジュールを起動して利用する（**画面8**）。COASを起動するには、**画面9**のようにKDEのメニューから必要なモジュールを選んで、起動すればよい。

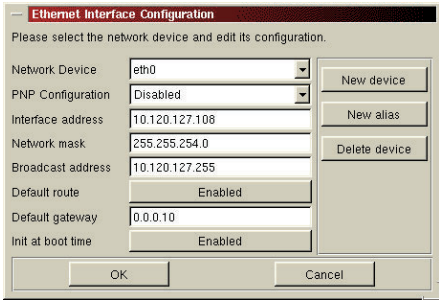
さらに、Webブラウザからのアクセスも行なえるようだ。

ただし、COASはいろいろ便利な面を持つことも確かだが、やはりそれぞれのモジュールをメニューから選ばないといけないのは面倒である（しかも、1つ起動するごとにダイアログボックスが開き、［OK］ボタンを押さないといけない）。さらに、このようなGUI管理ツールは最近のディストリビューションには付属していることが多く、このCOASもそれらと比べて特別に使いやすいというわけではない。



画面9　KDEメニューからCOASを起動する

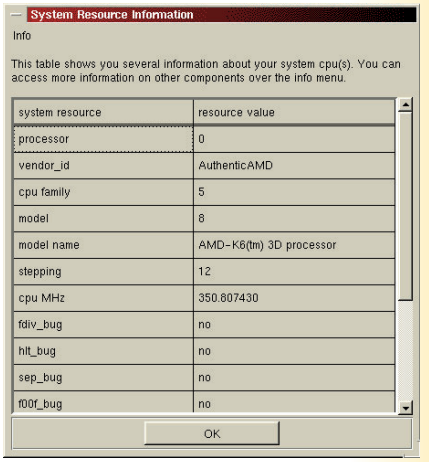




画面8　GUI管理モジュール「COAS」

ネットワーク管理
TCP/IP（NIS、リゾルバ）
イーサネットインターフェイス
メール転送
デバイス管理
キーボード
マウス
プリンタ
システム管理
アカウント
デーモン
ファイルシステム
ホスト名
リソース情報
日付／時刻／タイムゾーン
カーネル設定
ソフトウェアインストール

表1　COASが持つ管理モジュール



同梱の商用アプリケーション

商用版OpenLinuxには、いくつかの商用アプリケーションが付属している。dosemu上で動作するDR-DOSやLinux magazine No.3の195ページで紹介した多機能バックアップツール「BRU」などが同梱されているが、ここでは目玉ともいえる「WordPerfect 8」と「StarOffice 5.0」の2つについて紹介する。

Word Perfect 8

Word Perfectは、Corel（http://www.corel.com）によって開発／販売されているワープロである。

Linuxでは、Microsoft Wordや一太郎のようなWindowsのワープロに匹敵する機能を持ったものは、まだ出ていない（TurboLinux 4.0に同梱されている「一太郎Ark」も、機能は少なくHTMLエディタといったほうが近い）。その点、Windows用のWord Perfectとデータ互換で、機能もほとんど変わりのない、このWord PerfectはLinuxとしては画期的なアプリケーションといえよう（**画面10**）。

StarOffice 5.0

StarOfficeは、Star Division（http://www.stardivision.com/）が開発したアプリケーションスイートで、**表2**に示すような機能を持っている。**画面11**を見ればわかるように、かなりMicrosoft Officeを意識したアプリケーション構成となっている。さらにインターフェイスもツールバーの構成などまでもMicrosoft Office酷似している。

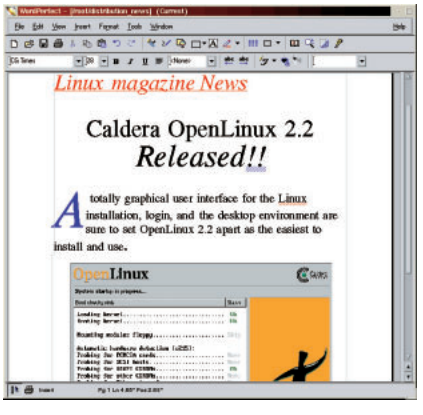
ただ、Microsoft Officeと異なり、各アプリケーションを個別に起動するのではなく、各アプリケーションは「StarDesktop 5.0」と呼ばれるコンテナアプリケーション上で起動するような仕様となっている。これは、ある目的のドキュメントを作成しようとすると、それに応じたアプリケーションが呼ばれるようになっており、ユーザーがアプリケーションの役割をあまり意識せずに、ドキュメント中心にデータ作成できるように配慮した結果だろう。感覚的には、Microsoft Office95に同梱されていた［バインダー］機能に近い。

しかし…

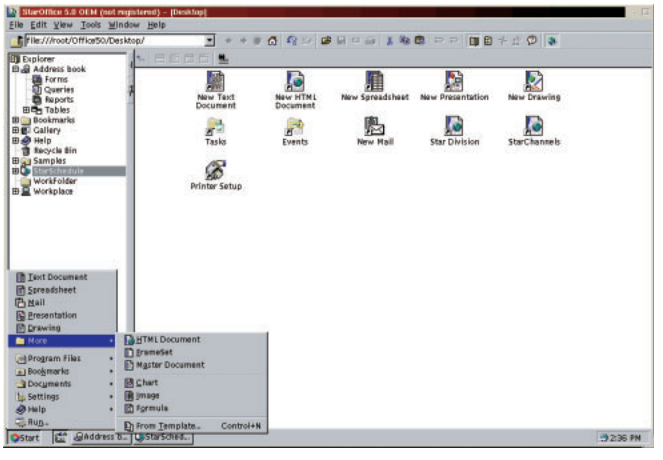
上記の商用アプリケーションは、かなり有用なアプリケーションで、使えるビジネスアプリケーションが少ないといわれるLinuxに大きな恩恵をもたらしてくれるだろう。しかし、肝心の日本語が使えないのだ（後述の「かぼす」をインストールしても同様）。そのため、これらの高機能な商用アプリケーションも、おそらく日本ではあまり利用価値はない。これは、今後に期待するしかない。

モジュール名	機能
StarDesktop 5.0	各アプリケーションを統合するワークスペースコンテナ
StarWriter 5.0	ワードプロセッサ
StarCalc 5.0	表計算アプリケーション
StarImpress 5.0	プレゼンテーションツール
StarDraw 5.0	ドローアプリケーション
StarBase 5.0	データ管理ツール
StarMail 5.0	電子メールクライアント
StarDiscussion 5.0	NetNewsクライアント
StarChart 5.0	グラフ作成ツール
StarImage 5.0	イメージエディタ
StarMath 5.0	数式作成ツール
StarSchedule 5.0	スケジュール管理ツール

表2　StarOffice 5.0で動作するモジュール群



画面10　ワープロアプリケーション「WordPerfect 8」



画面11　アプリケーションスイート「StarOffice 5.0」

日本語環境への対応

ここまで紹介してきたように、OpenLinuxは非常に高機能なディストリビューションなのだが、国産ディストリビューションではないため、日本語環境の取り扱いが気になるところだろう。これは、次に紹介する日本語化ツールによって対応がなされている。

■日本語化キット「かぼす」

日本語環境については、国内で販売代理を行うアミュレット（http://www.amulet.co.jp）が、Open Linuxに日本語環境を構築する追加パッケージ集「かぼす」を作成しており、これをインストールすることにより、**画面12**のように日本語の入力／表示が可能となる（本誌付録CD-ROM Disk2に収録）。なお、「かぼす」のインストールで実現できる詳細な項目は**表3**のとおりである。この「かぼす」は、「OpenLinux 2.2 英語版」パッケージに無償で添付されている。また、FTPサイトでも公開されている（ftp://ftp.amulet.co.jp/pub/kabosu/）。

■「かぼす」のインストール

「かぼす」は日本語化RPMバイナリの集まりなので、rpmコマンドを使って個々にインストールすることも可能だが、対話的なインターフェイスを持ったインストーラ（シェルスクリプト）が用意されているので、特に理由がない限りは、このインストーラを利用することをお勧めする。たとえば、本誌付録CD-ROM Disk2を利用してインストールするなら、CD-ROMをドライブにセットして、

```
$su -
Passward

# mount /mnt/cdrom
# cd /mnt/cdrom/KABOSU
# sh ./setup
```

とする。あとは、表示された説明にしたがって、リターンキーを押していくだけでよい（**画面13**）。

[重要]

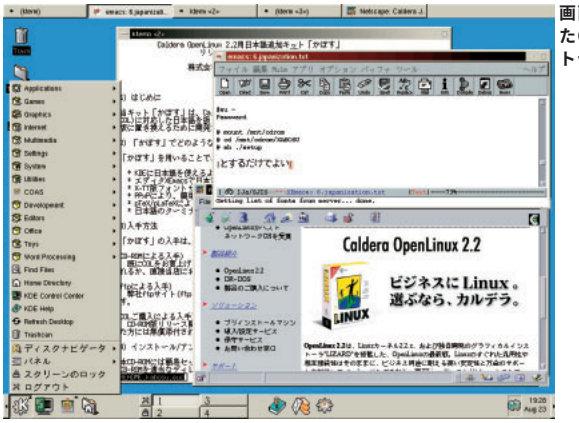
この「かぼす」をインストールすると、StarOffice 5.0が起動しなくなるという現象が編集部にて発見されました。「かぼす」開発元のアミュレットでは原因を調査中です。インストールする際は、この点を了解のうえ、自己責任においてインストールを行ってください。

■日本語マニュアル

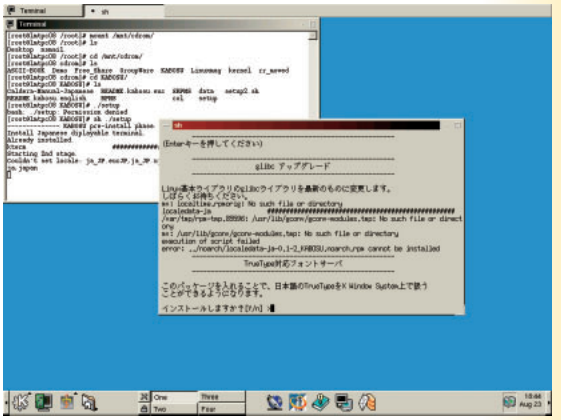
OpenLinuxには日本語マニュアルが存在しない。ただし、「かぼす」をインストールすれば、「OpenLinux 2.2 英語版」パッケージ同梱の英文マニュアル「GETTING STARTED GUIDE」の日本語訳が付属（HTML形式）しているし、http://www.amulet.co.jp/caldera/colman/index.htmlでも日本語訳の一部（Part1～2）が公開されている。

統合デスクトップ環境「KDE」の日本語化
日本語FEP「Canna」によるかな漢字変換入力
エディタ「XEmacs」での日本語利用
「X-TT版フォントサーバ」による日本語
TrueTypeフォントの利用
「PPxP」による簡単かつ高速なダイアルアップ接続
「pTeX/pLaTeX」による組版
「GhostScript」経由による印刷
日本語のターミナルやコードコンバータなどの利用が可能
「OpenLinux 2.2 英語版」パッケージ同梱の英文マニュアル「GETTING STARTED GUIDE」の日本語訳が付属（HTML形式）

表3　「かぼす」によって実現される日本語化機能



画面12　日本語化されたOpenLinuxのデスクトップ環境



画面13　「かぼす」のインストール画面

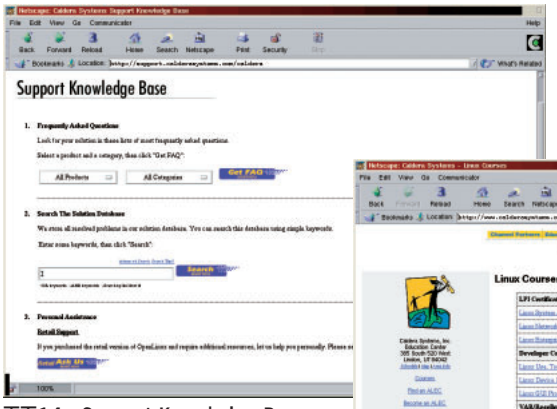
サポート環境＆Specification

ビジネス分野での使用を強く意識しているOpenLinuxは、サポートやトレーニングなどにも積極的に取り組んでいる。

■サポート

OpenLinuxは、Caldera Systemsにユーザー登録を行ってから90日間、もしくは5件までのインストールサポートが行われている。

また、Support Knowledge Base（http://support.calderasystems.com/）で、FAQを検索することもできる。



画面14　Support Knowledge Base

画面15　Calderaが提供するトレーニングコース

■トレーニング

さらに、管理者や開発者向けに豊富なトレーニングコースを設けている。詳細については、http://www.calderasystems.com/education/courses/を参照してほしい。

そのほか

これらの豊富なサポートやトレーニングコースも残念ながら英語しか利用することができない。当面は、販売元のアミュレットが提供しているLinuxセミナーやメーリングリストを利用するしかできない（詳細については、http://www.amulet.co.jp/caldera/support.htmlを参照してほしい）。

このあたりは、国産ディストリビューションに比べると大きく劣っているといわざるを得ないところだ。OpenLinuxが日本で普及していくためには不可欠な要素には違いないので、今後のサポートの充実に期待したい。

■Specification

ディストリビューション名	Caldera OpenLinux
対応ハードウェア	PC/AT互換機
開発元	Caldera Systems, Inc.
最新バージョン	2.2.5
ベースディストリビューション	Red Hat Linux
収録ソフトウェア	
カーネルバージョン	2.2.5
libc（glibc）バージョン	glibc2.1※1
XFree86バージョン	3.3.3.1
日本語対応アプリケーション	△※2
X-TT	△※3
インストーラ	
日本語表示	×
ハードウェアの自動識別	△(PCIカード＋一部のISAカード)
パッケージ方式	
パッケージ形式	rpm
依存関係チェック	○
設定ツール（GUI）	
ユーザー管理	○（COAS）
ネットワーク管理	○（COAS）
パッケージ管理	○（kpackage）
デスクトップ環境	○（control center）
ランレベル／実行サービス	×／○（COAS）
標準設定ファイルの有無	
bash	○
X／標準ウィンドウマネージャ	○/KDE 1.1※4
入手方法など	
オフィシャルサイト	http://www.calderasystems.com/
ミラーサイト（日本）	—
販売CD-ROM	OpenLinux 2.2 英語版ボックス（販売元：アミュレット、9,800円）
備考	※1　「かぼす」インストール後はglibc2.1.1+localedata-ja ※2　「かぼす」をインストールすることにより可能 ※3　「かぼす」に収録 ※4　「かぼす」をインストールするとKDE 1.1.1となる

OpenLinuxの普及と可能性

これまでのOpenLinux

OpenLinuxは、人気のある米国とは異なり、日本では知名度、シェアともに低迷していた。紹介記事では触れなかったが、OpenLinuxにはNetWareクライアントの機能がある（バインダリ、NDSの両接続に対応）。これが、米国では大きなメリットとなったが、NetWareの普及率があまり高くなかった日本では、これはたいした利点とはいえなかったということも一因としてあるだろう。

また、前バージョンのOpenLinux 1.3に対応した日本語追加キット「ゆず」を販売元のアミュレットが作るまで、日本語化は個人が行わなければならなかったことも普及を妨げていた。

国内普及に向けて

前バージョン対応の「ゆず」、そしてOpenLinux 2.2に対応した日本語追加キット「かぼす」の登場によって状況はある程度改善されている。

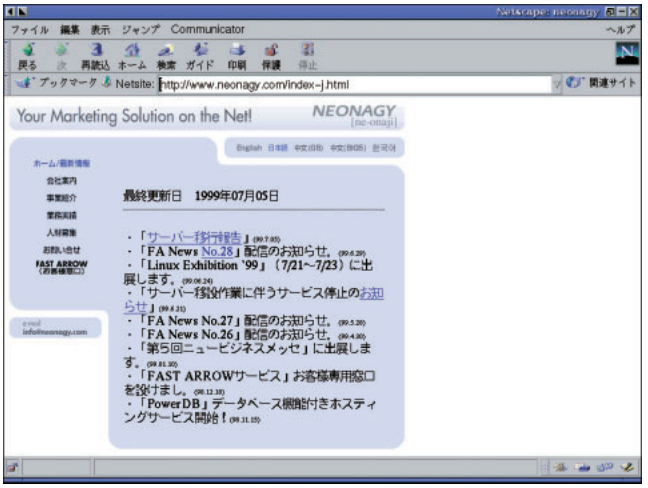
さらに、OpenLinux 2.2を利用したソリューションサービスが、ネオナジー（http://www.neonagy.com/）によって提供されている（**画面16**）。OpenLinuxや各種アプリケーションのインストールからインターネットサーバーの構築やネットワーク導入支援といった「ネットワーク導入サービス」、PostgreSQLを用いたシステム開発およびメンテナンスを行う「データベース開発サービス」、障害発生の開発支援やシステム保守を行う「サポート保守サービス」といった3つのコースが設けられている。このようなソリューションベンダーの登場は、ユーザーに対する安心感を与え、OpenLinux普及への好材料となるだろう。

あとは、インストーラ、アプリケーション、サポート、トレーニングなどの日本語対応が強く望まれるところだが、60ページからのRansom Love氏のインタビューにもあるとおり、開発元のCaldera Systemsも日本市場にも強い関心を持っており、日本市場参入へ向けていろいろな動きを見せている。ディストリビューションの日本語対応はもとより、日本語によるサポートやトレーニングなども、そう遠くないうちに実現しそうである。

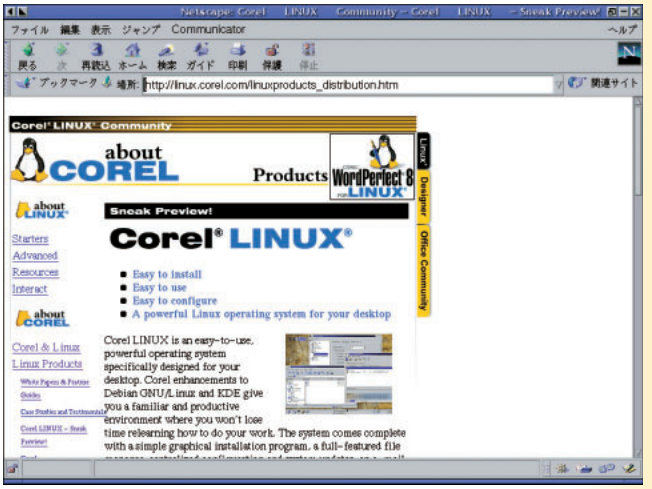
ディストリビューションの新潮流

OpenLinux 2.2は、VineLinuxなどと比べると日本語への対応度については及ぶべくもないが、インストーラの出来などは新しいディストリビューションの時代が近いことを感じさせる。

折りしも、先日開催されたLinux World Conferenc & Expoで参考出展された、Corel（http://linux.corel.com/）のコンシューマー向けディストリビューション「Corel LINUX」（**画面17**）は、GNU/Debian Linuxをベースとしているものの、KDE採用やQtを採用したグラフィカルで簡単なインストーラという点がOpenLinuxと同様であり、グラフィカルなインストーラが今後の大きな流れになっていきそうである。



画面16　ネオナジー（http://www.neonagy.com/index-j.html）



画面17　Corel LINUX（http://linux.corel.com/linuxproducts_distribution.htm）

付録CD-ROM収録!!
Caldera OpenLinux 2.2のインストール

ここでは、本誌付属CD-ROM Disk1に収録されているOpenLinux 2.2のインストール方法を説明します。なお、このOpenLinuxは英語版であり、基本的に日本語環境を考慮しておりません。あらかじめご注意ください。

インストールは、CD-ROMからのブートで開始されますが、CD-ROMブートが利用できない場合は、次の手順でインストールを始める前にブートディスクを作成する必要があります。Windows環境の場合、次の手順のとおりです。

1.フロッピーディスクドライブに初期化済みのフロッピーディスクを挿入する。

2.CD-ROMをドライブに挿入して、Auto Runで起動するインストール画面から、「Install Products」—「Create Floppy Install Diskettes」を選択する。

3.さらに「Create Lizard Install diskette」を選択する。

4.「Enter destination drive:」の質問に、フロッピーディスクを挿入したドライブ（通常はA）を入力し、ディスクの作成を開始します。

これで、インストールディスクが作成されました。また、インストール作業中にモジュールディスクの挿入を求められることがあります。その際は、上記手順の3.で「Create Lizard Modules diskette」を選択し、同様の手順でモジュールディスクを作成してください。

また、CD-ROMのcol/launch/floppyディレクトリにあるinstall.bat（インストールディスク）やmodules.bat（モジュールディスク）を実行しても作成できます。

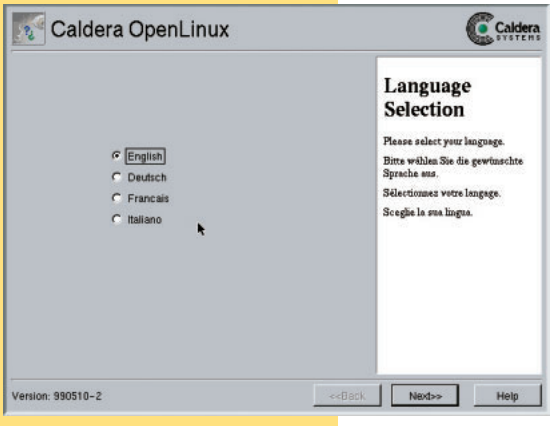
なお、既存のLinux環境では、CD-ROMをマウントしてから、col/launch/floppyディレクトリに移り、それぞれ次のコマンドを実行してください。

```
$ dd if=install.144 of=/dev/fd0
$ dd if=modules.144 of=/dev/fd0
```

さあ、次世代のインストーラ、LIZARDを堪能してください!!

Windowsからのブートディスクの作成

LIZARD Start!

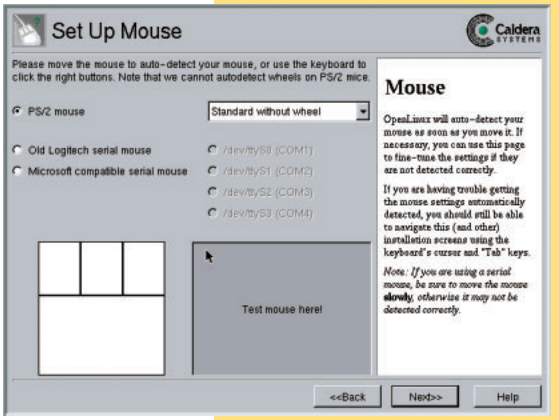


言語の選択

インストール時の言語を選択します。日本語はありませんので、「English」のままでよいでしょう。この画面でマウスを動かすと、自動的にマウスの種類が識別されますので、「Next>>」ボタンを押してください。もしマウスが自動で識別されず、マウスカーソルが動かない場合は、タブキーを押して「Next>>」ボタンにフォーカスを移動し、リターンキーを押してください。

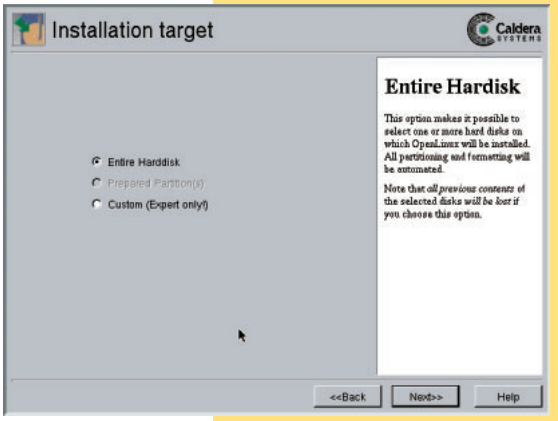
マウスの設定

前の画面でマウスが正しく認識されなかった場合は、キーボードのタブキーとカーソルキーを用いて、マウスの種類を選択します。



インストールするディスクの指定

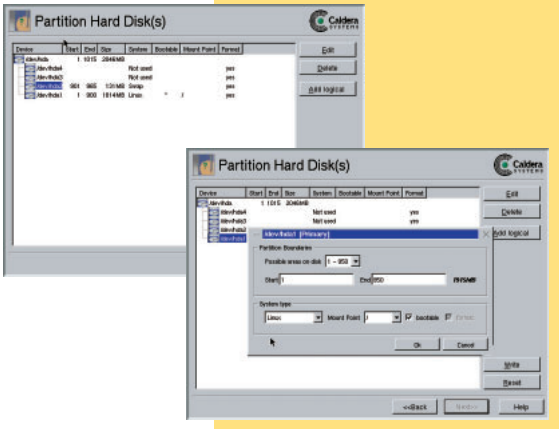
インストールするディスクとパーティションを設定します。ディスクに以前作成したLinux用のパーティションがある場合は、「Prepared Partition(s)」が自動的に選択されます。新たにLinux用パーティションとスワップパーティションを作成したい場合は、「Custom (Expert only!)」を選択します。なお、ハードディスク全体をLinuxで使用する際は「Entire Harddisk」を選択します。



パーティションの作成

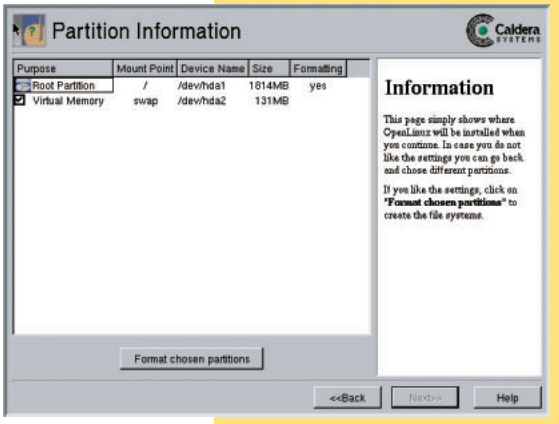
ディスク1台につき、4つの領域が表示されます。拡張パーティションがある場合は、論理ディスクも表示されています。パーティションを選択して「Edit」ボタンを押すと、ダイアログが表示されるので、各種条件を設定します。Linux用のパーティションは、System typeを「Linux」、Mount Pointを「／」にして、bootableにチェックを入れてください。スワップパーティションは、System typeを「Swap」にしてください。

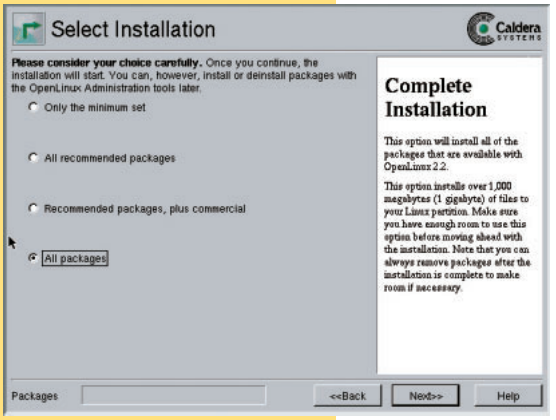
設定後「Write」ボタンを押して、パーティション情報を書き込みます。ただし、以前作成したパーティションの条件（サイズなど）を変更している場合、そのパーティションの中身は消えてしまいます。本当に消してもかまわないパーティションなのか、十分に確認してください。



パーティションのフォーマット

Linux用のパーティション（Root Partition）をフォーマットします。Root Partitionを選択し、「Format chosen partitions」ボタンを押します。

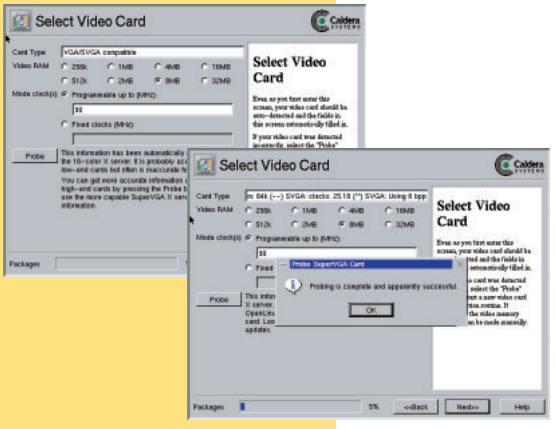




インストールするパッケージの指定

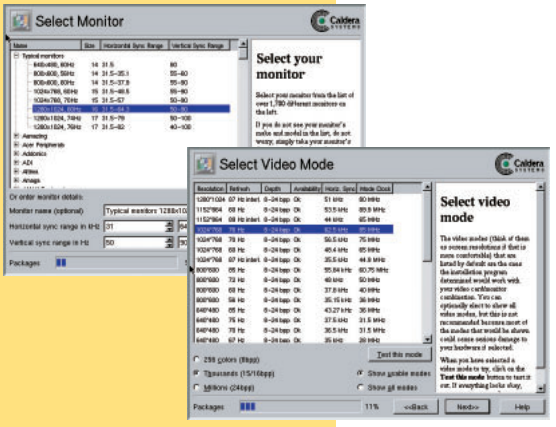
インストールするパッケージを指定します。選択肢は、上から「最小限」、「お勧め」、「お勧め＋商用製品」、「全部」の4種類です。ここでインストールしなかったソフトウェアも、あとで追加インストールできるので、あまり深く考える必要はありません。

「Next>>」ボタンを押すと、ファイルのコピーが始まります。これ以降の設定作業中、バックグラウンドでコピーが行われています。



ビデオカードの設定

使用するビデオカードを設定します。すでにCard Type欄に正しい名前が表示されている場合は、そのまま次の画面に進みます。VGA/SVGA compatibleと表示されている場合は、左端の「Probe」ボタンを押して、ビデオカードの検出を行います。

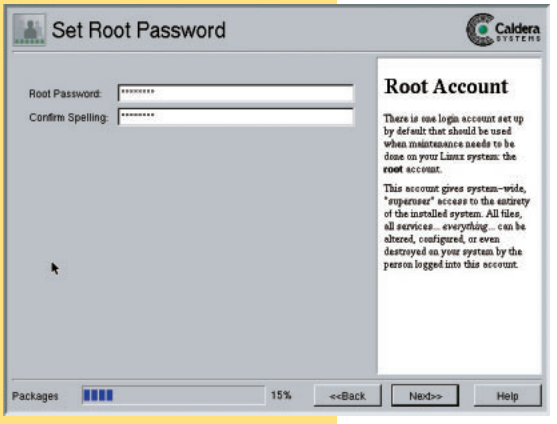


ディスプレイの設定

使用するディスプレイを設定します。リスト内に自分のディスプレイがあれば、それを指定します。見つからない場合は、「Typical monitors」から指定します。水平・垂直周波数の同期範囲は、モニターの説明書などで確認してください。

画面モードの設定

画面の解像度、色数、リフレッシュレートを設定します。使用可能なモードが表示されていますので、使用したいモードを選んでください。モードを選択後、「Test this mode」ボタンを押すことで、実際にそのモードで表示できるかどうか、確認できます。



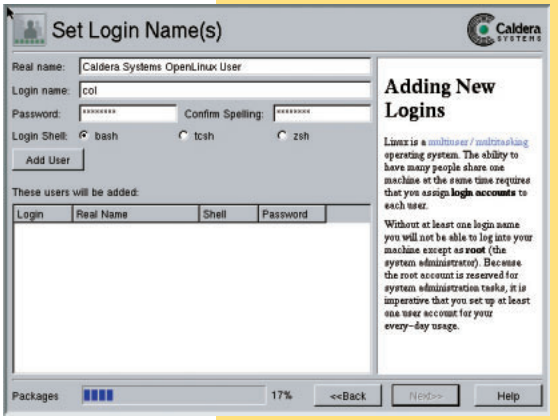
rootユーザーのパスワードの設定

rootユーザーのパスワードを設定します。Root Password欄に入力後、Confirm Spelling欄に再度同じパスワードを入力します。

ユーザー名の設定

マシンを使用するユーザー名を設定します。本名、ログイン名を入力し、ルートと同様にパスワードを入力します。シェルは特に好みがなければ、デフォルトのbashのままでかまわないでしょう。「Add User」ボタンを押すと、登録され、下のリストに表示されます。2人以上のユーザー名を設定したい場合は、本名の入力から繰り返します。

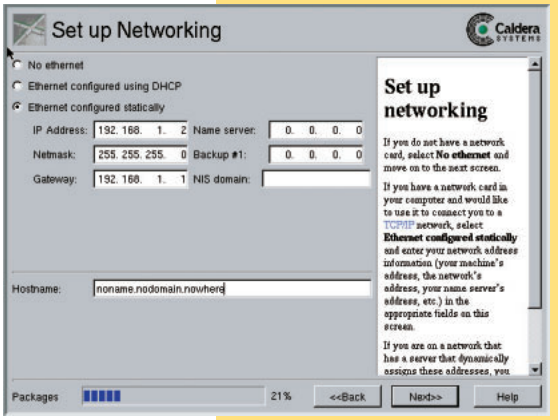
この画面では、最低1人のユーザー名を設定しないと、「Next>>」ボタンが押せないようになっています。



ネットワークの設定

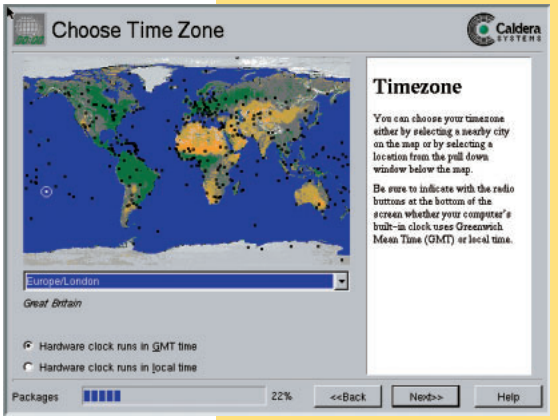
ネットワーク関係の各種設定を行います。ネットワークシステムに応じて、「No ethernet」、「Ethernet configured using DHCP」、「Ethernet configured statically」を選択します。

以下「Ethernet configured statically」を選択した場合は、IPアドレス、ネットマスク、ゲートウェイ、ネームサーバ（DNS）、ホストネームをそれぞれ設定します。予備のネームサーバがある場合は、Backup #1欄に入力します。



タイムゾーンの設定

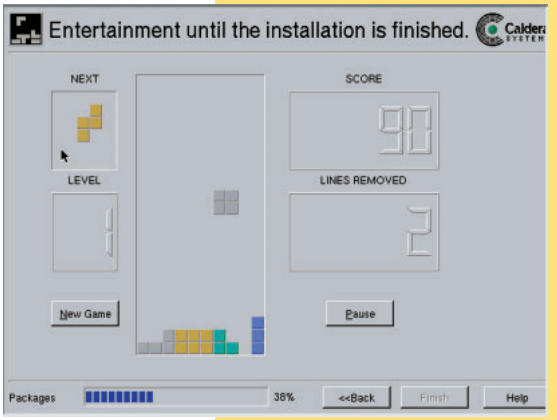
リストから「Asia/Tokyo」を選択するか、世界地図上で東京付近をクリックして、タイムゾーンを設定します。下のラジオボタンは、「Hardware clock runs in local time」に設定します。



インストール終了まで

ファイルコピー中は「Finish」ボタンが押せませんが、インストールが終わると、「Finish」ボタンが押せるようになります。

インストール終了まで、テトリスライクな落ちゲーで遊べますが、夢中になりすぎると、インストール終了に気がつかないことがありますので注意して下さい。



Caldera Systems CEO,Inc.

Ransom H. Love氏インタビュー

高機能な商用ディストリビューション「Caldera OpenLinux」の開発はもとより、日本市場への展開や、組み込みLinux「Lineo Embedix」へのライセンスなど、このところ活発な動きを見せている、Caldera SystemsのCEO、Ransom H. Love氏に話を聞いた。

Caldera Systemsについて

――まず、Caldera Systemsのバックグランドについて教えてください。

Love：弊社は、95年10月に設立されました。もともとは、Novell内のアドバンスドリサーチグループとして、Linux関連の製品開発に携わっていました。しかし、その後就任したCEO、Bob Frankenburgが、Linuxに興味を示さなかったので、Novellを出てCalderaを設立しました。

――現在のCalderaはどのようなビジネス展開をされているのでしょうか？

Love：現在のCalderaには2つの部門があります。1つは、Linux関連の企画・開発を行っているCaldera Systemsで、その成果となる製品が「Caldera OpenLinux」（以下、OpenLinux）です。弊社のLinuxへの取り組み姿勢は、はじめからビジネス用途をターゲットとした製品展開を考えており、デベロッパー関連の人たちを対象としたものではないという点で、ユニークな進め方をしてきたと思っています。

そして、もう1つの部門が組み込み系のビジネスを行っているLineo（旧Caldera Thin Client）です。

――DR-DOSをリリースしている会社ですね。

Love：そうです。DR-DOS以外にもEmbrowser（旧Web Spider）という組み込み用途のWebブラウザ製品もリリースしています。そして、このLineoにOpenLinuxのライセンスを提供することになり、先日発表したところです。したがって、今後は組み込み用途でもOpenLinuxの技術が利用されていきます（編注：現在、Lineo Embedixとして製品リリースがなされている）。

Caldera OpenLinuxのセールスポイント

――Linux人気の高まりとともに、さまざまな特徴を持ったディストリビューションが次々にリリースされています。その中でOpenLinuxが持つセールスポイントとはどんなところでしょうか？

Love： 第一の特徴は、弊社でしっかり管理されているディストリビューションであることです。つまり、バラバラな寄せ集めの製品として提供するのではなく、いろいろなLinux関連のテクノロジーを1つのソースとして集めて、ソリューションとして製品化を行っています。

――それは、どのような意味を持ちますか？

Love：サポートが充実すること、さらにシステムインテグレートを行うにあたって最適化を行って、それに基づいてソリューションを提供していくことができるということが利点です。さらに、セキュリティ面でも優れています。それは、Webからただかき集めただけのソリューションではなく、弊社が手を加えた製品として提供しているからです。

――ユーザーが安心して利用できるというわけですね。

Love：そうです。さらに、もうひとつの重要なセールスポイントとして「使いやすさ」が挙げられます。OpenLinux 2.2のLIZARDインストーラはグラフィカルなもので、Linuxの難しさをある程度緩和することに成功していると思います。Windowsから移行してきたユーザーには特に効果を発揮するでしょう。

――LIZARDは日本のユーザーにも大きなインパクトがあるインストーラだと思います。インストーラ以外にも「使いやすさ」を体感できる機能はありますか？

Love：システム管理についても同様のことがいえます。これは、COAS（Caldera Open Administration System）が重要な役割を果たします。このCOASは、KDEからの起動はもとより、Webブラウザを介したシステム管理ができるということが大きな利点です。

――たしかに、OpenLinuxを実際に使ってみると、いたるところで「使いやすさ」という一貫したポリシーは強く感じますね。

Love：また、ターゲットを絞ったパッケージングを行っているということもセールスポイントといえます。つまり、サーバ、組み込み市場のように、ある業務に特化したソリューションにも対応できるようなパッケージを提供しているのです。最初のセールスポイントと重なりますが、ただ単にまとめただ

けのものを渡しているのではなく、企業の環境を構築するために使えるパッケージを作って、それを統合できるようにしてから提供しているからこそできることといえるでしょう。

――サポートはどうでしょうか？

Love：もちろん、サポートも重要なセールスポイントです。弊社は、トレーニングとサポートという両方のインフラを提供することを目指しています。なぜなら、クライアントがLinuxの知識を持たないことには、それを利用することができないからです。したがって、この状況を打破する意味でもトレーニングの提供は不可欠なのです。さらに、製品サポートをとおして、世界中に販売を広げていくための知識を蓄積していくのです。このサポートは、IBMなどのビジネスパートナーとの連携しながら、より良いものを提供していくつもりです。

日本市場への展開

――世界的な視点から見ると、日本語を利用する日本は、かなり特殊性を持った市場であると思います。今後、日本での展開にあたって、どのような対応をお考えですか？

Love： 多くのクライアントは信頼できる会社から製品を買いたいと考えているでしょう。ですから、まず信頼できるパートナーを見つけて、密なパートナーシップを築きます。そして、そのパートナーが教育からサポート、日本語化までを行うようにしたいと思います。さらに、そのパートナーがクライアントが必要とするソリューションを提示し、それに基づいて日本の市場が必要とする製品を投入していきます。そのためにも、まずより良いパートナーシップの締結が先決です。

――既存の日本語ディストリビューションに対するアドバンテージはどのような形で提供していきますか？

Love：まず、日本語化を進め、品質の優れた製品を出していくことが前提です。さらに、クライアントが満足できる製品をしっかりしたチャネルを通して販売し、サポートしていくことでアドバンテージを出していきたいと思います。弊社は、IBMを通して販売しているサポートや20コース用意しているトレーニングには非常に自信を持っており、これらも日本語で対応できるようにしていく予定です。

Linuxのビジネス展開に向けて

――サポートやトレーニングの強化など、日本でもビジネス色の強い展開をされていくのですね？

Love：はい。ただし、弊社は日本市場に限らず、一貫してビジネス展開を考えたLinuxの開発を行っています。Linuxに対する関心は、日本でも、そして世界中で盛り上がっています。しかし、Linuxをビジネスの中でどのように展開していくかということについては、まだ世界的にも始まったばかりで、各社が評価を進めているところです。私は、キーとなるクライアントに対して製品を提供するところからビジネスは発展すると考えています。そのためには、それを提供するためのパートナーシップが非常に大切になります。そして、そのパートナーにはソリューションやテクノロジーがすでにあるということが前提です。それは、Windows NTからの置き換えを考えているクライアントに満足してもらえるような製品を提供しなければならないからです。そのためのパートナーシップを築いていって、今後の展開を図るための努力を進めているところです。

（聞き手：編集部 北野）

単なる「寄せ集め」の製品ではなく、Linux関連のテクノロジーを1つに集めた「ソリューション」としての製品がOpenLinuxです。

Linux vs. NT

LinuxとWindows NTを徹底比較。それぞれのメカニズムを探る！

Photo:Shuichi Mito (Dee)

１９９９年に入ってから、日本でのLinuxの勢力は急速に拡大してきた。メディアには毎日のようにLinux関連ニュースが登場しており、またLinux関連製品も分野を問わず多数発表されている。そんな中、最近の傾向として感じるのは、仕事でLinuxを使う環境が急速に整ってきていることだ。たとえば、今号の第2特集で扱っているグループウェアでも、最近はLinux対応が目立つ。もちろんグループウェアはビジネス市場向けのソフトウェアである。また、大手PCサーバメーカーも、これまでは自社のPCサーバがLinuxで動作することを確認するだけにとどまることが多かったが、最近、サポートはもちろんLinuxをPCサーバにプレインストールして販売するようになってきた。これならばシステムの導入も比較的容易であり、またサポートもあるから、安心して仕事に利用できる。

PCサーバのOSといえば、現在はマイクロソフトのWindows NTが幅を利かせている分野である。Windows NTは、名前に「Windows」と入ってはいるものの、MS-DOSを祖とするWindows 95／98とはまったく別の系統で開発されたOSである。基本的に1種類のアーキテクチャでサーバとワークステーションそれぞれの分野をカバーし、ビジネス市場で大きなシェアを誇るOSである。特にWindows NTのシェアが大きいのは、ローエンドからミッドレンジのPCサーバ用途だ。

一方のLinuxは、特にインターネット関連のサーバに採用されているほか、OS自体のコストが低いことからローエンドのPCサーバと組み合わせて販売される例が増えてきている（前述のプレインストールPCサーバもこれ該当する）。今後ミッドレンジまでLinuxが伸びるようだと、LinuxはWindows NTと激しく競合することになる。両者はお互いに市場のシェアを奪い合うライバルとなるだろう。今後は両者が混在するシステムが多くなったり、両者のうちどちらを導入するか選択しなければならない場面も増えたりすることが予想される。つまりLinuxユーザーでも、Windows NTのことをある程度知っておくほうが望ましいわけだ。

そこで今回は、LinuxをWindows NTを技術面・機能面で比較することで、それぞれの特徴や得手・不得手など明らかにしていく。第1部では、現在の「モダン」なOSに必要とされる機能を解説するとともに、LinuxとWindows NTのカーネルアーキテクチャに触れる。第2部では、サーバOSの基本的なサービスについて、両者を機能面で比較する。それでは「敵を知り、己を知れば、百戦危うからず」を実践してみよう。

オペレーティングシステムの機能

文：デジタルアドバンテージ　打越浩幸
Text : Hiroyuki Uchikoshi

プロセス管理

ユーザーが実行するプログラムは、実行形式のファイルに格納されて通常はディスク上に置かれている。プログラムを実行するには、この実行形式のファイルをメモリ上へ読み込んできて、CPUやメモリリソースなどの環境（「実行コンテキスト」という）を用意し、実際にCPU時間を割り当てる必要がある。これを行うのがプロセス管理である。

LinuxやWindows NTのようなマルチタスクOSでは、通常、1つのシステム上で複数のプログラムが実行されることになるが、この場合、各プロセスにどのように時間を割り振るか（スケジューリングするか）とか、その優先度の管理、メモリリソースなどが不足した場合のプロセス全体のスワップアウトやスワップインなどの処理なども行われる。

また最近のOSでは、プロセスよりもさらに小さい実行単位としてスレッド（thread）という機能を用意しており、プロセス管理のオーバーヘッドの軽減や、マルチプロセッサシステムにおける全体的な性能の向上を目指している（**図「プロセスとスレッド」**参照）。

この場合、プロセスはプログラムをロード・実行するためのアドレス空間の割り当て単位となり、スレッドは、CPUに対する実行時間の割り当ての単位となる。1つのプロセスの中には、複数のスレッドが含まれ、その内の1つがメインスレッドとして最初に実行が開始される（つまり、プロセスの中には、少なくとも1つのスレッドが存在する）。そして、そのメインスレッドから、必要ならばさらに補助的なスレッドが起動される。グローバルなコードやデータ領域はプロセス全体で共有されるが、これとは別に、スレッドごとにCPUなどのレジスタリソースやスレッド独自のスタック領域などが含まれた、「スレッドコンテキスト」という実行環境が用意される。OSは、プログラムのロードやアドレス空間の割り当てはプロセス単位で行い、実際のCPU時間の割り当ては、スレッド単位で行うことになる。

プログラムの実行をスレッド単位で行えるようになるメリットは大きい。たとえば複雑なユーザーインターフェイスや外部との通信を含む、ワードプロセッサやHTMLエディタのようなアプリケーションを考えてみよう。このようなアプリケーションでは、エディタとしてユーザーの入力を受け付けて画面に表示する必要があるが、この場合、ユーザーの入力に対して1文字ずつ画面を描画していると、非常に遅くて使いづらいものになる可能性がある。

そこで、ユーザーの入力を受け付けるスレッドや、それに基づいて内部のデータ構造を管理したり、画面の描画指示を出したりするスレッド、ウィンドウシステムからの指示によって画面の再描画などを行うスレッドなど、複数のスレッドを使うように設計しておくと、プログラムの応答性が向上するし、開発の手間も削減される。

スレッド化しておけば各機能は独立することになるので、描画が完了しなくても、次の入力を行うことができる。また、検索／置換機能や、ファイルの保存、印刷、HTMLエディタならばWebサーバとのアップロード／ダウンロード機能などもスレッド化しておけば、バックグラウンドでの処理やインクリメンタルな処理なども容易に実装できるであろう。

また、スレッドを使うもう1つの目的として、マルチプロセッサシステムなどにおける処理性能の向上が挙げられる。スレッドをサポートしているOSでは、通常はスレッド単位にCPUを割り当てるようにスケジューリングを行う。このため、プログラムが複数のスレッドを使うようにできている場合（マルチスレッド対応）、密結合の対称型マルチプロセッサシステム（SMP、物理メモリ空間を共有して動作するマルチプロセッサシステム。各CPUの役割は対等で、すべてのCPUから同じように主記憶にアクセスできる）ならば、複数のCPU上で同時にスレッドが動作するため、全体的な処理性能を向上させることができる。

実際には、主記憶へのアクセスが競合するなどのオーバーヘッドがあるので、CPUの数に完全に比例して処理性能が上がるわけではないが、たいていの場合は効果がある。特に、多数のクライアントからの処理を同時に処理しなければならないサーバ用途や、画像処理アプリケーションにおけるフィル

タ処理のような用途では、その効果は大きい。

このように、スレッドを複数のCPUで効率よく分担して処理するためには、OS側の対応が欠かせないが、OSによってはまだ十分サポートしていない場合がある。最初にシングルCPUを前提にして作られたOSは、（Linuxを含む、ほとんどのUNIX系OSは、当初はマルチプロセッサシステムをサポートしていなかった）、マルチプロセッサシステム上で動作させると、リソースの競合や不整合が起こってしまうのである。

これに対処するためには、たとえばリソースに対してはすべて排他的なロックをかけてからアクセスするなどの対策を完全に行わなければならない。しかしこれは非常に手間がかかる作業である。

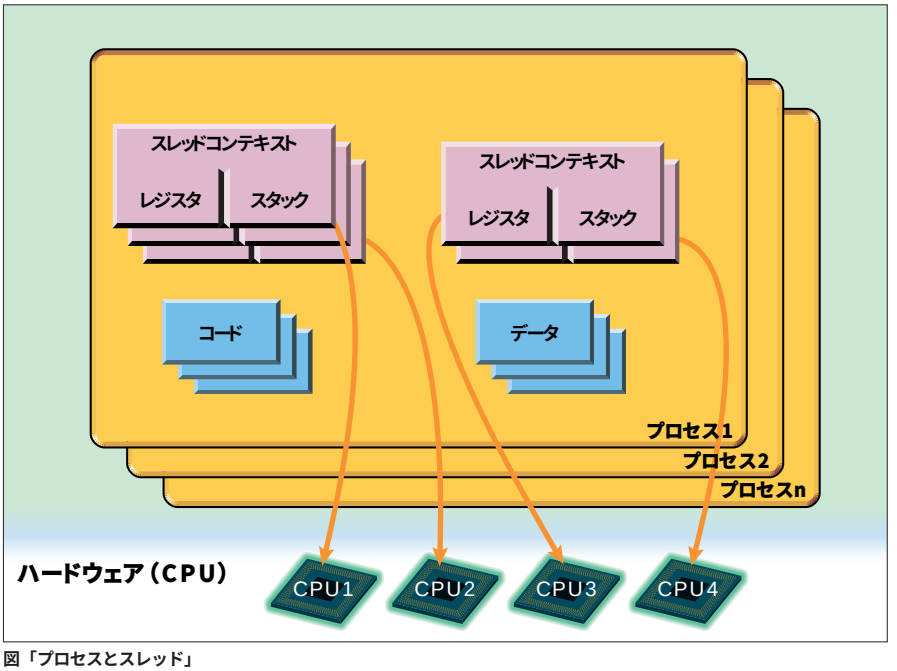
そのため暫定的に、カーネルはそのままにしておいて、ユーザープログラムでのみマルチスレッドを使えるようにしたシステムが使われることがある。このようなスレッドを「ユーザーモードスレッド」とか「ライトウェイトスレッド（軽量スレッド）」といい、ユーザープログラム内のスレッドライブラリで擬似的にスレッド環境を構築、スケジューリングしている。この場合のスレッド間でのコンテキストスイッチや、各プロセッサへのスレッドの割り当てなどはユーザープログラム側で行う必要がある（実際には、スレッドライブラリの内部で処理されるので、ユーザーが細かくコンテキストスイッチングを制御したり、スケジューリングアルゴリズムを管理したりする必要はないが）。

UNIX系OSでは、POSIX標準に基づいたpthreadというユーザーモードスレッドのライブラリが使われている。これを利用すると、ユーザープログラム内では複数のスレッドを複数のCPUに割り当てて同時に実行することができる。しかし、システムコールなどはどれか1つのスレッドが実行していると、他のスレッドはブロックされてしまう（待たされる）システムが多い（システムから見ると、あくまでも1つのプロセスにすぎないから）。そのため、完全なスレッドサポートをしたOSと比べると、システムのパフォーマンスが劣ることになる。

ただしLinuxにはclone()というシステムコールが用意されており、複数のスレッドからのシステムコールを同時に処理することもできる。clone()はfork()のような機能を持っているが(fork()を呼び出すと、呼び出し元のプロセスとまったく同じメモリ空間を持つ、別のプロセスが生成される)、元のプロセスとアドレス空間などを共有している。つまり、異なるプロセスIDを持つ2つのプロセスに分離するが、リソースは共有していて、通常のスケジューリング規則が適用される。これはまさにスレッドそのものであるが、psコマンドなどでは2つのプロセスに見える（詳細は後述）。

Windows NTは、もともとマルチプロセッサ、マルチスレッド対応のOSなので、Win32 APIのひとつとしてスレッドAPIを備えている。これによって生成された各スレッドは、NTカーネル自身がスケジューリングして実行する。マルチプロセッサシステムにおける分散処理も、カーネル自身が行う。

なおWindows NTには、スレッド以外に、ファイバ（fiber）という機能も用意されている。これは、いわゆるユーザーモードスレッドであり、スレッドコンテキストの中で実行される。スケジューリングやコンテキストスイッチングなどは、ユーザーのプログラム自身で行う必要がある（カーネルサポートのないユーザーモードスレッドやライトウェイトスレッドに相当する）。



図「プロセスとスレッド」
プロセス（プログラム）とスレッド（プログラムの実行単位）を分離して、1つのプロセスを複数のスレッドで処理できるようにしている。

メモリ／バッファ／アドレス空間管理

メモリ管理とは、システムに搭載されている物理メモリを、プログラムやOSカーネルからの要求に応じて、割り当てたり、不要になったものを回収したりして後の用途に備えるなどの処理を行う。

割り当てられた物理的なメモリは、各プロセスの実行のために使われたりするほか（プログラムコードや動的／静的なデータ領域、スレッドコンテキストなど）、OSカーネル内部で、プロセスやスレッドを管理するための各種の構造体や、ディスクやネットワーク、デバイスドライバとの入出力のための各種のバッファなどにも利用される。

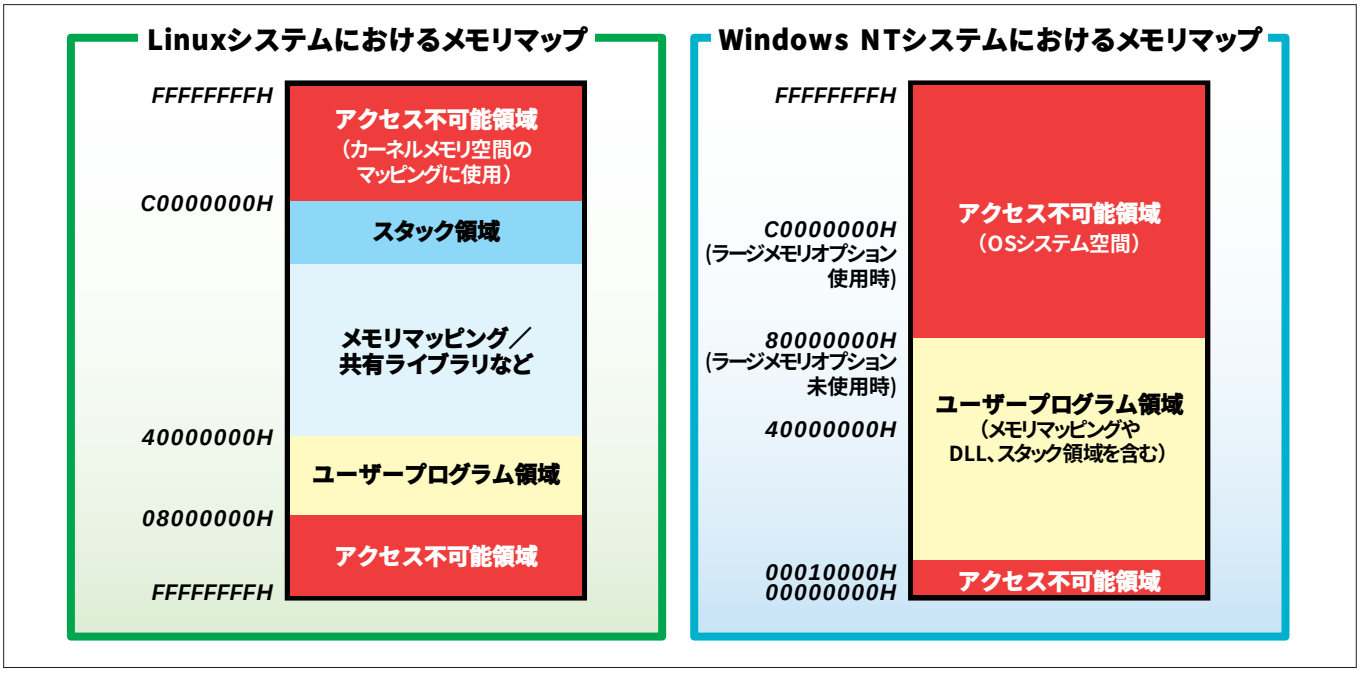
現在の一般的な32ビット OS環境では、通常は1プロセス当たり2～4Gバイト程度のメモリアドレス空間が割り当てられることが多い（プロセスごとに4Gバイトのアドレス空間が割り当てられるので、アドレス空間の総計は、実際のプロセッサでアクセス可能な物理メモリサイズよりも大きくなる）。このアドレス空間内には実行ファイルから読み込まれた、プログラムのコードや初期データのほか、実行中に動的に増減するデータ領域（ヒープ領域）や、スレッドごとのデータやスタック領域などが配置される。また共有ライブラリなどもこのアドレス空間内に配置される。**図「プロセスのアドレス空間」**に、LinuxとWindows NTにおける、各ユーザープロセスのアドレス空間を示しておく。

ところで、以上の処理は物理的なメモリをどう管理するかという処理であったが、より進んだ現代のOSでは、あらゆるオブジェクトにアドレス（アドレス空間）を割り当てて、OSカーネルやプログラムから直接アクセスできるようにするという方法が進んでいる。

たとえば、メモリマップドファイルなどはこの典型的な例といえよう。これは、ファイルをアドレス空間内のどこかにマッピングしておくと、そのアドレス空間をポインタで参照するだけで、ファイルの内容にアクセスできる、という機能である。いちいちバッファに少しずつ読み出してきたりする手間をかけずにファイルにアクセスすることが可能となる。

この手法では、ユーザープログラムでのバッファリングの管理などが不要になるというメリットがある。従来、ユーザープログラムでは、ファイル入出力のパフォーマンスを向上させるために、ある程度大きなサイズ（実際にプログラムで必要なデータは、そこからさらに小さく分けて使用する）を単位としてデータを読み書きするバッファリング処理を自前で用意していた。

ユーザープログラム側であらかじめ



図「プロセスのアドレス空間」
LinuxとWindows NTそれぞれのユーザープログラムからみたメモリアドレス空間のレイアウト。いずれのOSでも、アドレス0から始まる領域は、NULLポインタ割り当て検出などのためにアクセス不可能領域となっている。Linuxでは、サポートする物理メモリの最大サイズによって、プログラムから利用できる上限が変わる。これは最大1Gバイトまでサポートする場合のレイアウトである。Windows NTでは、標準では最大2Gバイトのアドレス空間が利用できるが、Windows NT Server Enterprise Editionのラージメモリサポート機能を使うと、最大3Gバイトまで利用可能となる。

バッファリング処理用のバッファなどを用意する必要がないので、必要なメモリの量が減少する。もちろんこの場合でも、OS内部ではメモリマップドファイルのために、実際にはデータのバッファリング処理を行っていたりするが（そうしないとパフォーマンスが出ないから）、システム内の他のプロセスとの兼ね合いや物理メモリの空き具合などを勘案して、最適に決定される。

後述する最近のマイクロカーネルベースのOSでは、このような考え方をさらに徹底的に押し進めて、物理メモリは単にアドレス空間（ディスクや各種のリソースが含まれる）のためのキャッシュに過ぎない、という概念を採用しているものがある。プログラムの実行に必要となる各種のオブジェクトは巨大な（仮想的な）アドレス空間内に散在しており、実際にCPUが必要とした時点でそれらが物理的なメモリ上に呼び出されてくるのである。

物理メモリにマッピングされる単位は、一般的には各CPUの物理的なページサイズ（4Kバイトないし8Kバイトのものが多い）が最小単位となるが、実際にはワーキングセットサイズを考慮して、適切なサイズが選ばれる。ワーキングセットサイズとは、よくアクセスされる、ある特定範囲のアドレス空間のことで、このサイズが大きいとヒットする確率が高くなるが、メモリをそれだけ大量に使用することになる。

また空きメモリがなくなった場合は、各プロセスやスレッドの優先度、アクセス頻度などを考慮して、不要なオブジェクトやページがディスクなどにスワップアウトされたり、（読み出しのみのプログラムコードページならば）内容が破棄されたりする。

この方式のメリットとしては、システム全体を通して最適なメモリ割り当てができるということが挙げられる。あらかじめ固定的なバッファ領域を用意しておく必要はないし、同じオブジェクトはシステム全体を通して一カ所でしかキャッシュされていないので、無駄にキャッシュ内で重複することがない。これはプログラミング方法にもよるが、たとえば、カーネル内とユーザープログラム内の両方で、同じファイルが同じようにバッファリングされることがなくなり、メモリが有効に利用できるようになる。

デバイス管理（デバイスドライバ）

システムで利用できる各種のハードウェアデバイス（ディスク型デバイス、キャラクタ型デバイス、各種入出力装置、ネットワークデバイス、グラフィックスデバイスほか）を制御するのがデバイス管理である。

一般的なOSでは、最低限のデバイスとの入出力ルーチンと割り込みハンドラさえ用意すれば、あとは上位に用意されたファイルシステムドライバやネットワークドライバなどがそれらの機能を使って、ファイルシステムやネットワーク入出力、グラフィックス描画などを行ってくれるようになっている。

デバイス管理におけるもうひとつの大事な機能としては、各種のハードウェアアーキテクチャに依存するような機能を抽象化して、さまざまなハードウェア上でOSを実行できるようにするということが挙げられる。

現在では、ほとんどのプログラムやOSはCやC++言語のような高級言語で記述されるため、CPUやマシンのアーキテクチャなどが異なっていてもあまり依存することはないが、OSの核となる部分ではどうしてもCPUに依存した処理を行わざるを得ない。

このようなハードウェア依存の項目としては、デバイスとの入出力や割り込みハンドリング、物理メモリの管理や仮想記憶のためのページング処理、ブート処理などがあり、これらはアーキテクチャごとに個別に開発される。たとえば、現在のLinuxでは、x86アーキテクチャのほか、Sparc、Alpha、arm、mc68000、PowerPCなどがサポートされている。Windows NTでは、x86とAlphaアーキテクチャがサポートされている。いずれのOSでも、アーキテクチャの違いはなるべく局所化するように設計されており、少なくともC言語のような高級言語で記述している限りは、各プロセッサアーキテクチャに依存しないで済むようになっている。

割り込み／例外処理

一般に、割り込みや例外は、ハードウェアによって引き起こされるが、場合によってはソフトウェアから擬似的に発生させることもある。各種の割り込みや例外を統一的に処理できるように仮想化するのもOSに求められる機能である。

また、リアルタイム用途などに向けて、プロセスや割り込み処理の優先度を上げて、必ずサービスが実行されるようにしたり、ある一定時間以内の割り込み応答を保証したりすることもある。

プロセス間通信

プロセス間通信機能とは、2つ以上のプロセスの間で、データをやり取りしたり、同期を取ったり、イベントを送るなど、協調して処理を行うために必要な機能である。通信の相手は同じ

システム上のプロセスだけではなく、他のシステム上で動作しているプロセスと同期を取ることもできる。これにより、複数のプロセスが共同して処理を進めたり、異なるシステム間で通信をしたりすることができる。

代表的なプロセス間通信のための機能としては、シグナル（イベントの通知）や共有メモリ、プロセス間パイプ、名前付きパイプ、IPC（Inter Process Communication）、RPC（Remote Procedure Call）、メッセージキュー、セマフォ、ストリーム、ネットワークコネクションなどがあり、これらは用途によって使い分けられている。

IPCやRPCは、異なるプロセス間でも、サブルーチンを呼び出すような（一見すると）手軽な方法で、プロセス間通信を行うための手段である。一般に、異なるCPUアーキテクチャ間でプロセス間通信を行うと、多バイトデータのバイトオーダー（ビッグエンディアンかリトルエンディアン）などの影響を受けないように、データのバイトオーダーの変更なども行われる。

ファイルシステムサポート

ファイルシステムとは、プログラムやデータを格納するためのファイルを階層的に構築、管理するための機能であるが、それ以外にも名前空間としての役割（データを格納するだけでなく、各種のリソースを識別するための機能。デバイスファイルやロック用ファイル、共有リソースの管理などにも使われる）も持っている。

また、単にファイルを独自のフォーマットで管理するだけでなく、他のOSシステムとのデータ交換などの用途も考えると、広く普及しているMS-DOSフォーマットのディスクや、メジャーな他のUNIXのファイルシステム、CD-ROMのISO9660などの広く普及しているファイルシステムなどへもアクセスできる必要がある。さらに、物理的なローカルディスク上のファイルだけではなく、ネットワーク経由で他のシステム上のファイルシスムへもアクセスするためのネットワークファイルサービス機能も、最近のOSでは必須といえる。

このような多様なファイルシステムを実装するために、OS内部では、仮想的なファイルシステムのマウント機構を用意して、複数のファイルシステムを同時に、柔軟に混在させて利用できるようにしている。

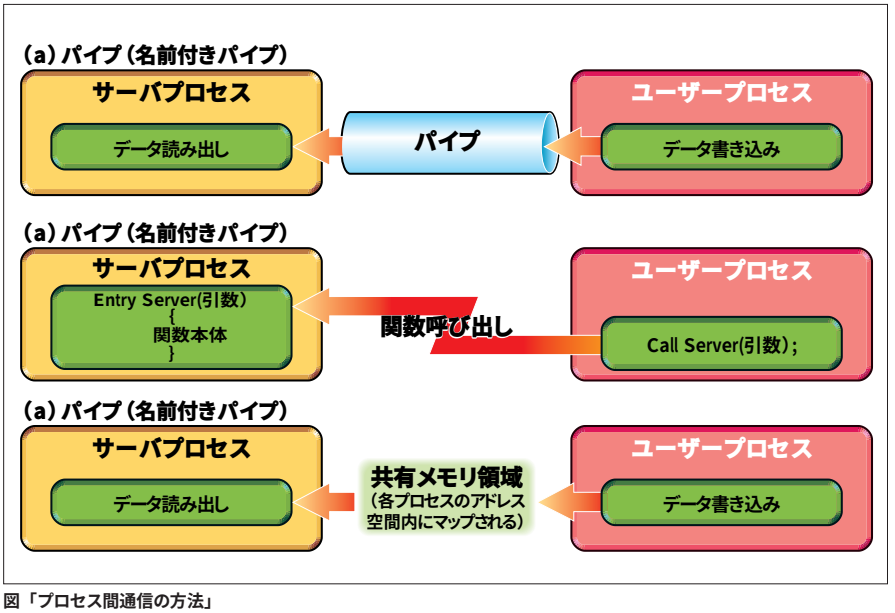
現在一般的に広く使われているファイルシステムとしては、UNIXなどのufsやext2ファイルシステム、Minixファイルシステム、MS-DOSやWindows NTのFAT、VFAT、NTFS、HPFSファイルシステム、CD-ROMのISO9660ファイルシステムなどのほか、ネットワークファイルシステムとしてNFSやSMB（Samba）、AppleShare（netatalk）などがある。

ユーザー管理／セキュリティ機能

システムにアクセスするユーザーを識別したり、そのユーザーの権限に応じた操作のみを実行できるように管理したりするのもOSに必要な機能である。

UNIXでは、歴史的に、ユーザー名とグループ名だけを使ってこの処理を行っている。商用UNIXでは、さらに細かいユーザー権限などを管理する機能を持っているが、Linuxではそのような拡張は行っていない。必要ならば、アプリケーションが個別に独自のセキュリティ機能を提供している。

Windows NTでは、基本的には、システム内のすべてのオブジェクトに対して、それを操作したりアクセスしたりする権限を個別に設定して管理している。オブジェクトへアクセスする場合は、ユーザーやグループ名に基づ



図「プロセス間通信の方法」
2つのプロセス間で通信するには、2つのプロセス間で通信チャネルを開設するパイプを使ったり、LPC/RPC（手続き呼び出し）でサーバ側のサービスエントリを呼び出したり、共有メモリで直接データを読み書きしたりする方法がある。また、シグナルやイベントで相手側に通知したり、スレッド間でセマフォやミューテックスなどを使って、同期／排他制御をする方法もある。

いてまずアクセス権の検査が行われ、それをパスしない限りアクセスしたり、操作したりすることができない。さらに監査機能（オブジェクトへのアクセスの記録を取っておく機能）も備えている。Linuxの場合は、監査機能が必要ならアプリケーションレベルで対応する必要がある。

グラフィックス機能

最近では、グラフィカルユーザーインターフェイスの普及により、OS全体の機能としても、グラフィックス機能のサポートが求められるようになってきている。

もともとUNIX系OSでは、グラフィックス機能はOSのサポート範囲外であり、必要ならば各ユーザープログラム側で個別に対応していた。が、最近ではOSの基本機能として、X Window Systemを備えていることが多い。この場合、グラフィックス機能として求められるのは、ユーザーインターフェイス用途なら単純な2次元描画機能でもよいが、最近ではゲームやOpenGLを使うアプリケーションなどの要求に応えて、グラフィックスカードの持つ3次元描画機能などもサポートされている。ただしGUIの不要なサーバ用途などでは、X Window Systemは不要であり、Windowsと違ってシステムに必要なメモリが少なくてすむ。

これに対してWindows NTの場合は、システム自体がウィンドウシステムをベースとしたOSであるため、Win32 APIセットの中にグラフィックス描画機能が含まれている。また、ゲームやグラフィックス向けアプリケーションのために、DirectXやOpenGLなどのAPIセットも備えている。最近のグラフィックスカードは高度な3次元描画機能をサポートしているが、それらを有効に使うためには、ポリゴンデータやモデリングデータなどを直接グラフィックスチップに渡し、なるべくチップの持つ機能（ポリゴン単位での描画や座標計算、テクスチャマップ、アルファブレンディング、アンチエイリアシングほか）を有効活用する必要がある。そのためにも、DirectXやOpenGLなどのAPIセットのサポートが必要なのである（Win32 APIだけでは単純な2次元描画しかできない）。

ネットワーク機能

ネットワーク機能は、現在のOSシステムでは必須の機能といえる。複数のコンピュータ間でデータをやり取りしたり、インターネットにアクセスしたりするのに、ネットワークの機能は欠かせないからだ。

OSに求められるネットワーク機能としてはさまざまなものが考えられるが、現実的には、TCP/IPをベースにした一連のプロトコルや各種のサービスが求められている。あとは各OSや環境ごとに独自のプロトコルやサービス（Windows環境のSMBプロトコルや、MacintoshのAppleShare、NetWareのファイルサービスなど）も用意されていれば、他のシステムとの相互運用性が高くなる。

OSに求められるネットワーク機能としては、各種のネットワークデバイスに対する接続機能のほか、TCP/IPなどのコアプロトコル、およびその上で動作する各種のネットワークサービス（Web、メール、FTP、telnet、NFSほか多数）がある。どれだけネットワークサービスが充実しているかだけでなく、その安定性やパフォーマンスなども重要な要素といえる。

64ビットCPUアーキテクチャのサポート

コンピューティング性能の向上に伴って、近い将来CPUも64ビット化が進むことになる。64ビット化することにより、データの処理幅が広がって演算能力が向上するというメリットもあるが、当面は、より広いアドレス空間が利用できるということの意義が大きいと思われる。現在の32ビット OSでも、ファイルマッピング機能などを使おうとすると、32ビットアドレス空間（4Gバイト）では明らかに不足しているからだ。ただしこのためには、OSのサポートするアドレス空間やAPIセットが、64ビット化される必要がある。

商用UNIXではすでに64ビット化が行われており、LinuxでもSparcプロセッサのために64ビットシステムコールの実装が行われている（Mercedプロセッサ用の開発も進んでいる）。Windows NTでも、現在のWin32を拡張してWin64 APIの仕様策定と実装が行われようとしているが、実際に利用できるようになるのはまだまだ先のようである。

●

これら以外にも、OSには、その導入や運用のしやすさ、スケーラビリティ、パフォーマンス、耐障害性、異種アーキテクチャサポートなど、さまざまな機能が求められている。デスクトップや組み込み用途からエンタープライズ環境まで、1種類のOSですべてカバーできるのが理想であろうが、現実的にはこれは無理といえる。ユーザーは1種類のOSだけにこだわることなく、その機能や将来性なども加味して、複数のOSシステムやマシンアーキテクチャを使ってシステムを構築したり、維持管理したりする必要があるだろう。

モノリシックカーネルOSとマイクロカーネルOS

文：デジタルアドバンテージ　打越浩幸
Text : Hiroyuki Uchikoshi

OSシステムの構造には、大きく分けると、モノリシックカーネル構造を採用したものとマイクロカーネル構造をベースにしたものの2つの種類がある。前者は伝統的なシステムで使われている手法であり、マイクロカーネルは比較的最近のOSで採用されているアーキテクチャである。

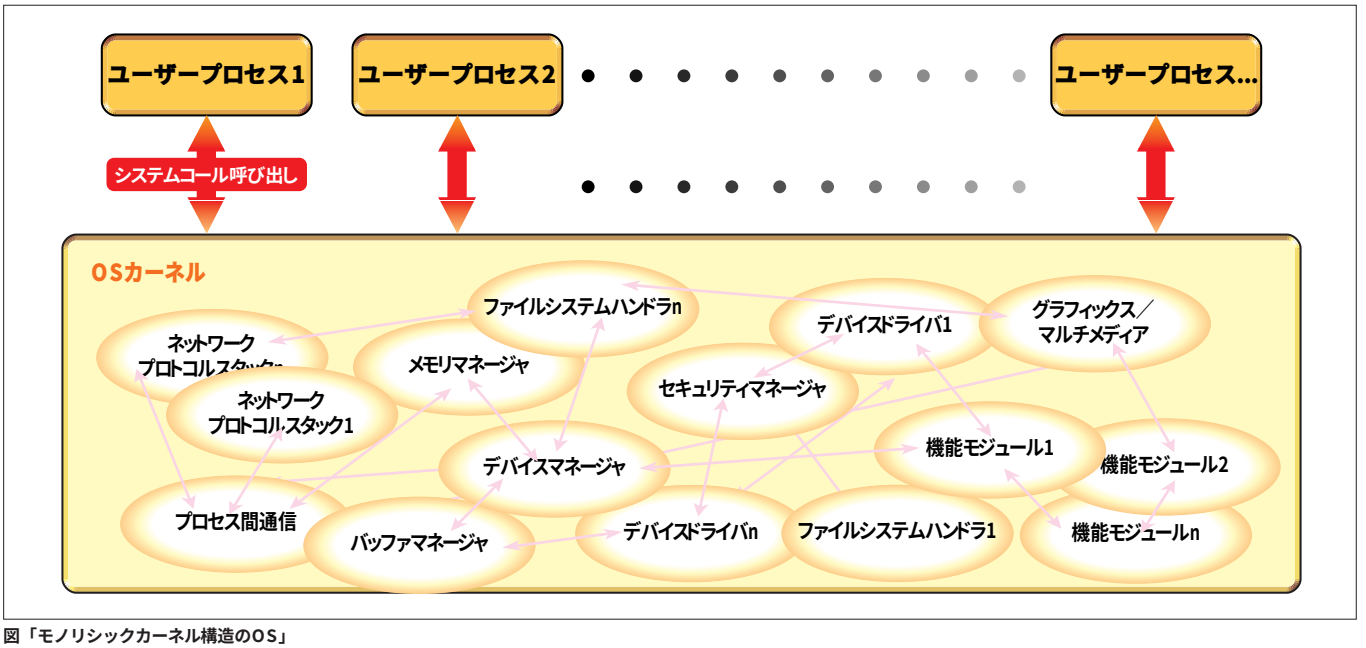
モノリシックカーネルOS

モノリシック（monolithic、一体型）カーネルは、カーネル内部に各種のさまざまな機能やサービスを組み込んで、全体としては巨大な1つの構造となるように作られているOSのことである。最初に作られたUNIXを始め、伝統的なシングルCPU向けのOS（4.4BSDまでのUNIXや、Windows 95/98、MS-DOSなど）はすべてこのモノリシック構造のOSとして作られている。

当初Minixを参考に作られたとされるLinuxも、このモノリシック構造のOSである。このような構造のシステムは特に珍しいものではない。最初は簡単な機能しか持たなかった基本システム（もしくはプロトタイプ）に対して、ネットワーク機能や高性能な入出力機能、グラフィックス機能、新しいデバイスに対するサポートなどを次々に追加していくと、どんなソフトウェアでも巨大化の一途をたどり、その内部は相互に複雑に関連する、多数の機能モジュールで構成されるようになる（**図「モノリシックカーネル構造のOS」**参照）。OSに対する機能追加要求がなくならない限り、このような巨大化、複雑化の傾向はずっと続くことになる。

しかしシステム全体の性能を向上させるために、マルチプロセッサシステムや分散システムなどをサポートしようとすると、このような構造が性能向上の足かせとなることは明らかであろう。先に述べたように、OSカーネルをマルチプロセッサ／マルチスレッドで動作させたくても、もともと排他制御などを考慮していないプログラムでは、マルチスレッドで動作させることはできないからだ。もしマルチスレッドで動作させると、内部的なデータ構造に不整合が生じてしまう可能性がある。

たとえばあるプロセスを生成する処理を考えてみよう。このためには、プロセス用のデータ構造体を割り当て、



図「モノリシックカーネル構造のOS」
モノリシックカーネルのOSでは、1つのカーネルコードの中に各モジュールがすべて含まれており（しばしば、巨大な実行ファイルになっている）、それらはデータ構造や各種のルーチンを共有しながら密接に絡み合って構築されている。これらはすべて1つのアドレス空間の元に置かれ、データを共有しながら動作している。場合によっては、カーネル内部で複数のスレッドが動作している。

関連する各種のリソース（シグナル、スレッドコンテキスト、新しいプロセスIDほか）など確保する必要がある。途中で他のシステムコールに割り込まれてしまうと、これらのデータ構造の整合性が保てなくなり、システムが破綻するのである。

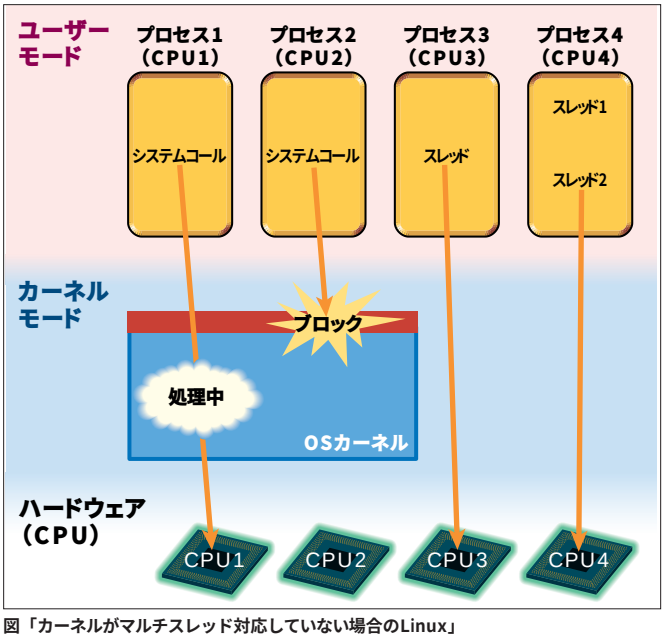
そこでシステムのマルチスレッド化のためには、まずユーザーモードスレッドをサポートしてユーザープログラムのスケーラビリティを確保しておき、その後カーネル内部をマルチスレッド実行できるようにOS内の各部を修正・変更するという手法が取られる。カーネル内部のマルチスレッド化が完了するまでは、カーネルは常に単一のCPUでしか動作しないように、排他制御が行われる。つまり、複数のプロセスから同時にシステムコール呼び出しが発生しても、現在処理中のシステムコールが完了する（もしくは、自発的に他のシステムコールの処理にスイッチする）まで、後続のシステムコールの処理を待たせておくのである。システムコールの処理性能はプロセッサ1台分に限定されるが、ユーザープロセスはマルチCPUの恩恵を受けることができる。

Linuxでは、2.2よりも前のバージョンのカーネルがちょうどこの状態に相当している（**図「カーネルがマルチスレッド対応していない場合のLinux」**参照）。さらにLinuxにはclone()システムコールがあるので、ユーザープログラムではシステムコールを発行しない限り、複数のCPU上で同時に動作することができる。

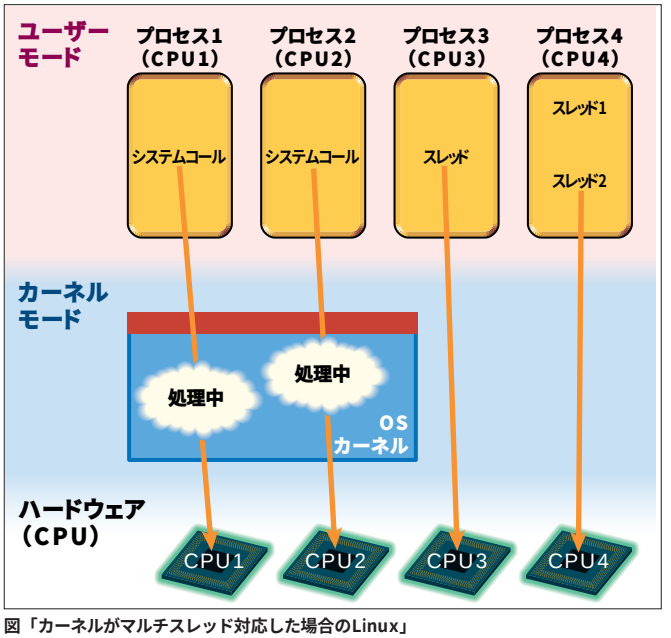
そしてLinuxのカーネル 2.2以降ではこれよりもやや進化している。カーネル全体でシステムコールを1つずつ排他的に処理するのではなく、カーネルの中の限定的な部分（タイマーリストを処理する部分、runqueueを処理する部分、物理ページを割り当てる部分など）ごとに排他制御を行うようになっている。このため、複数のプロセスからのシステムコール呼び出しを同時にカーネル内部で処理することができるようになった（**図「カーネルがマルチスレッド対応した場合のLinux」**参照）。CPUが複数あれば、それらは同時に処理されるので、カーネルコードのスケーラビリティも高くなっている。

マイクロカーネルOS

マイクロカーネルOSとは、従来のモノリシック型のOSにおける構造を再構築して、非常にコンパクトにまとめられた「マイクロカーネル（microkernel）」と、サービスを実現するための「サーバプロセス」を分離した構造のOSである。ユーザープロセスから見たOSカーネルのインターフェイスや機能は同じであり、従来のプログラムをそのまま動かすこともできるが、その内部構造は大きく変更されている（**図「マイクロカーネル構造のOS」**参



図「カーネルがマルチスレッド対応していない場合のLinux」
4つのプロセスが4つのCPU上でそれぞれ実行されているとき、2つのプロセスから同時にシステムコールを呼び出しても、後から呼び出したほうは、カーネルの入り口でブロックされ（スリープ状態になる）、最初のシステムコールの処理が完了（もしくはそれが自発的にsleep）するまで、待たされる。システムコールに関係のないユーザーモードプロセスは各CPU上でそのまま実行される。



図「カーネルがマルチスレッド対応した場合のLinux」
4つのプロセスが4つのCPU上でそれぞれ実行されているとき、2つのプロセスから同時にシステムコールを呼び出しても、カーネルの入り口で待たされることなく、両方のシステムコールの処理が異なるCPU上で同時に実行される。これによりパフォーマンスが向上する。なお、システムコールに関係のないユーザーモードプロセスはそのまま各CPU上で実行される。

照)。

マイクロカーネル方式では、カーネル内部には必要最低限の機能しか実装されていない。プロセス／スレッド管理、メモリ管理、プロセス間通信機能、割り込み／例外処理、デバイスドライバ（ハードウェア）インターフェイスなどである。このほかにも必要に応じて、オブジェクト管理や、セキュリティ機能なども入れられることがあるが、基本的にはカーネル内部では非常に限られた処理しか行わず、モノリシック型のOSで提供されていたような各種のシステムコールはすべて、ユーザーモードで動作している（つまり、ユーザープログラムと同じ特権レベルで動作している)、カーネル外部のサーバプロセスにおいて実行される。

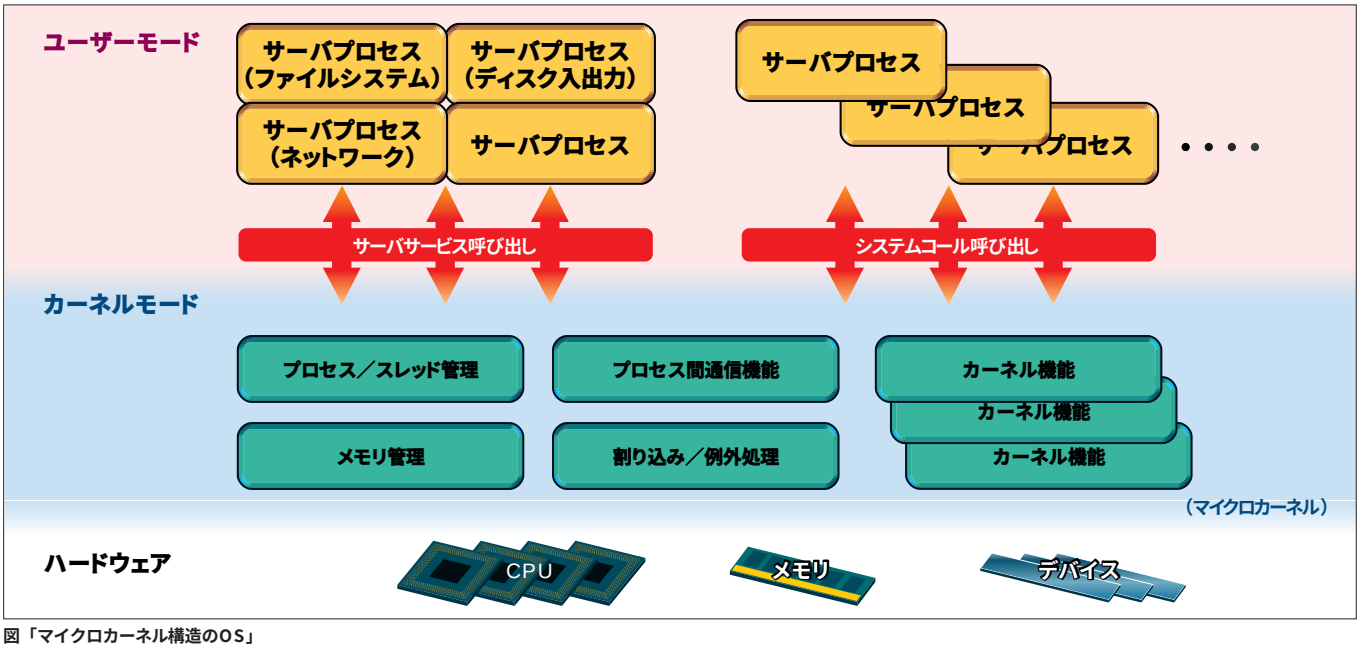
たとえば、ディスク上のファイルシステムへのアクセスや読み書き、バッファリング処理、ネットワークプロトコルの処理などもすべて、外部のサーバプロセスが担当する。また、仮想記憶システムにおけるページフォルト処理（物理メモリがマップされていないアドレス空間へアクセスした場合に、そのページの内容を外部のディスクからメモリへ読み込んでくるなどの処理）は、すべて仮想記憶管理のためのサーバプロセスで行い、デバイス入出力や割り込み処理などもカーネルの外部で処理される。

このようにマイクロカーネル方式のOSでは、OSの提供するサービスをすべてクライアント／サーバ形式で処理することが大きな特徴となっている。この場合の処理の流れは次のようになる。まずアプリケーションプログラムが呼び出したシステムコール（たとえば、ファイルのopenやread／writeなど）は、マイクロカーネル部へ渡される。その後、あらかじめ登録されているサーバプロセスへ引数と共に渡され、システムコールが処理される。そして、処理の結果は逆の順序をたどって、サーバプロセスからカーネルを経由して、最初に呼び出しを行ったアプリケーションプログラムへと戻される（**図「マイクロカーネルOSにおけるサービスの呼び出し」参照**)。

実際には、これらのマイクロカーネルやサーバプロセスに対する呼び出しは、プロセス間通信機能やRPC（リモート手続き呼び出し）によって行われる。そのためマイクロカーネル方式のOSでは、カーネルのサービスとして、プロセス間通信機能があらかじめ用意されている。

モノリシックカーネル方式に比べると、マイクロカーネル方式では、アプリケーションプログラムとサーバプロセス間でのモード遷移や、パラメータ渡しのオーバーヘッドが大きくなっている。それにも関わらず、このようなサーバ／クライアント方式を使ってOSの機能を実現しているのは、機能拡張の容易さや、モジュラリティ（モジュール性)、スケーラビリティなどの能力が優れているからである。

機能拡張が容易であるというのは、OSのサービスを各種サーバプロセスで実現しているため、特定のOSの機能



図「マイクロカーネル構造のOS」
マイクロカーネル方式のOSでは、カーネル内部には必要最低限の機能しか実装されていない。カーネル内部では非常に限られた処理しか行わず、各種のシステムコールはすべて、ユーザーモードで動作しているカーネル外部のサーバプロセスにおいて実行される。

にとらわれず、各種のサービスを容易に追加、変更、拡張することができるということを意味している。どれか特定のサービス（たとえばLinuxのシステムコール）だけにとどまらず、BSD UNIXや、POSIXのインターフェイスに準じたUNIXサービスを提供し、しかもユーザープログラムごとに異なるサービスを利用することもできる。たとえばWindows NTでは、Win32だけでなく、OS/2やPOSIX準拠のインターフェイスも同時に実現している。もちろん常に単一のインターフェイスだけを実現してもよいのであるが、必要ならば複数のサービスを、同時に実現する事ができるのがマイクロカーネル方式のOSの大きな特徴である。

モノリシックカーネルのOSでは、（歴史的に）システム全体がある特定のサービスやシステムコールに向けて最適化されており、まったく異なる新しいサービスやアプリケーションインターフェイスを実装するのは困難なのが普通である。OS内部のデータ領域や各種のサービスのためのルーチンが相互に複雑に関連していて、ある特定の機能だけを取り出して変更したり、機能追加したりすることができないからだ。

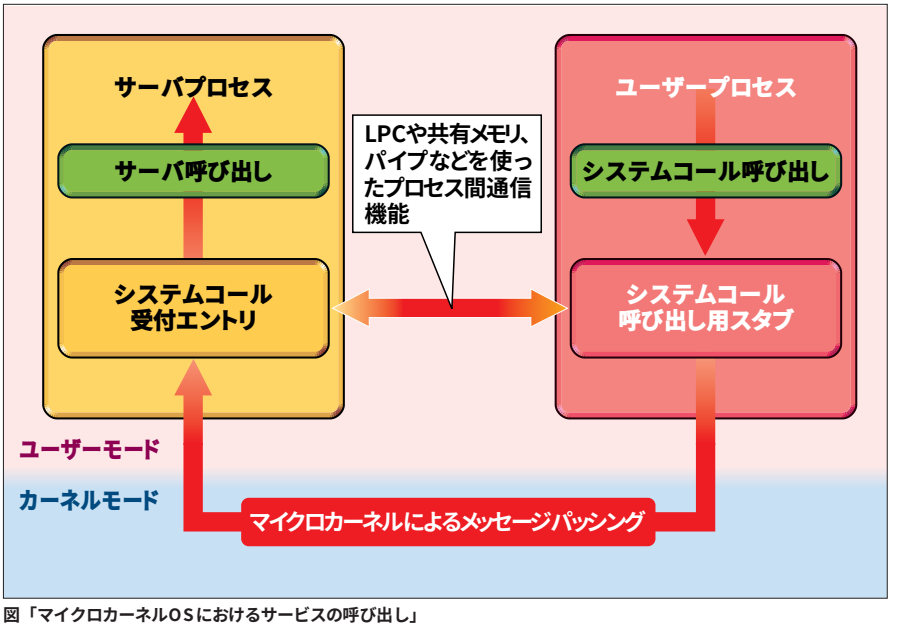
これに対してマイクロカーネル方式のOSでは、各サービスが独立したサービスプロセスとして実現されているので、機能拡張などに伴う修正や追加が比較的容易であり、その分、開発のための工数やデバッグ作業なども少なくてすむということになる。ただし、実際にはクライアント／サーバ型で実装しなければならないため、このようなプログラミングスタイルに馴れていなければ、逆に開発効率が低くなるかもしれない。そのため無理に各サービスを分けて実装するのではなく、たとえば当面は既存のUNIXカーネルを1つのプロセスにしてしまって、シングルサーバ方式でUNIXを実装するということも可能であるし、またよく行われている。このような柔軟性も、マイクロカーネル方式の利点といえる。

スケーラビリティが優れているというのは、マルチプロセッサシステムなどを使えば、システムのパフォーマンスを向上させることができる、ということを指す。OSのサービスを提供するサーバプロセスが複数に分かれているため、マルチプロセッサシステムならばこれらのプロセスを各CPUで分散処理することにより、ユーザーアプリケーションだけでなく、カーネルの性能もスケーラブルに拡張することができるのである。これは大規模なマルチプロセッサシステムでは、大きなメリットといえる。

このほかにも、各サービスプロセスが分離独立しているので、あるサービスが障害を起こしても、その影響が他へ及ばないというメリットもある（もっとも、サービスが利用できなくなることには代わりがないので、これはあまりメリットとはいえないかもしれないが）。

マイクロカーネル方式のデメリット

以上のようにスケーラビリティなどに優れたマイクロカーネル方式であるが、ユーザープログラムとサーバプロセス間の通信などのせいで（プロセッサの特権モードの変更などが頻発するから）、オーバーヘッドが大きくなるという欠点もある。そこで、現実のマイクロカーネル方式のOSでは、いくつかのサーバプロセス（たとえばWindows NTの描画ルーチンなど）をカーネル内に実装し、余計なオーバーヘッドを避けるようになっているのが普通である。



図「マイクロカーネルOSにおけるサービスの呼び出し」
ユーザープロセスがシステムコールを発行すると、それはマイクロカーネルを経由して、サーバプロセスへと送られる。実際には、ユーザープロセスとサーバプロセス間で通信するためのチャネル（LPC：Local Procedure CallやRPC：Remote Procedure Call、共有メモリ、パイプなど、さまざまな方法を使って実現できる）を確保し、それを使ってサーバとクライアントが通信を行う。この方式では、モノリシックカーネル方式よりもコンテキストスイッチのオーバーヘッドが多くなるため、パフォーマンス的には不利になることが多い。

モノリシック構造のLinuxカーネル

それでは、LinuxとNTについて、その内部構造を見てみよう。

すでに述べたように、Linuxのカーネルは現在のところはモノリシック型の構造を採用している。各種のデバイスドライバやサービス（ネットワークサービスやファイルサービス、キャッシュ管理ほか）などはあらかじめカーネル内部に組み込まれており、それらがすべて1つのアドレス空間内でデータ構造などを共有しながら動作しているのである。そのためシステムコールや割り込み処理を実行するときは、カーネル内部でのリソースへのアクセスに対する競合を避けるため、排他制御が行われている。

マルチプロセッサシステムの恩恵を最大限に受けるためには、カーネル内部のサービスなどを積極的に分離独立させて、複数のプロセスやサービスに分割するのが一番良いと思われるが、実際には開発工数などを考慮して、現在の所はまだ部分的なリソースの排他制御にとどまっている。

Linuxにおけるシステムコールの排他制御の方式は比較的単純で、当初はカーネル全体をロックするようになっていた。その後カーネル2.2以降では、局所的なリソースごとに排他制御をする方式に改められている。

これにより、マルチプロセッサシステムの場合には同時に複数のシステムコールが処理できるようになっている。ただし現在でも、システムコールの処理中に他のシステムコールの処理が割り込むことはできないようになっているので、優先度の高いプロセスであっても、現在処理中のシステムコールが完了するか、ウェイト状態に入るまで、処理が待たされることになる。

■UNIXにおけるシステムコール処理

ここで、Linux（や伝統的なUNIX）におけるシステムコールの処理方法を説明しておこう。システムコールを発行するには、x86アーキテクチャのLinuxの場合は、引数を各レジスタにセットしたあと「INT 80H（ソフトウェア割り込みの0x80番）」命令を実行する（**図「x86 Linuxにおけるシステムコール処理の概要」**参照）。すると今まで「ユーザーモード」で実行されていたプロセスのコンテキストが「カーネルモード」に切り替わり、カーネル内部へと制御が進んでいく。Linux（や伝統的なUNIX）では、ユーザープログラムからシステムコールを発行すると、このようにユーザーコンテキストの延長としてカーネル内部へと制御が移行し、処理が行われる。

ネットワークのクライアント／サーバ方式のアプリケーションなどのように、複数のプロセスからの（システムコールの）処理要求をその入り口でキューイングしておいて（待ち行列に溜めておいて）、それを順次取り出しながら、1つずつ順番に処理する、というような形態にはなっていない。いってみれば、カーネルというコードやデータを、マルチスレッドで各ユーザープロセスが共有しながらそれぞれ実行しているといえるだろう。だから、1つのプロセスで同時に発行できる（同時に処理できる）システムコールの数は、1つまでしか許されない。カーネルサポートのないユーザーモードスレッド（軽量スレッド）を使っても、複数のスレッドからシステムコールを同時に発行することはできない。実行できるスレッドコンテキストは、常にどれか1つしかないからだ。

ただしLinuxでは、clone()システムコールを使って1つのプロセス中に2つ以上のスレッドを生成することができる。この場合のスレッドは、カーネル側から見ると単に複数のプロセスのように見えるので（実際そのように扱われている）、各スレッドが独立してシステムコールを発行することができる。

ところで、プロセスがシステムコールを処理中かどうかは、psやtopコマンドで表示されるWCHAN（ウェイトチャネル。プロセスがどういう要因で停止しているかを示すイベント）フィールドを見れば分かる。ここにカーネル内部のルーチン名（もしくはアドレス数値）が表示されていれば、システムコールを処理中である。もっとも、ここに表示されている停止要因は、ほとんどの場合、（デバイスや他のプロセスなどからの）データの入出力待ちか、シグナルによる外部からの停止指示によるものである。

さて以上のような仕組みのため、UNIXではカーネル内部にある各種のデータ構造体などは、当初から（マルチプロセッサをサポートする前から）、各所にリソースを排他的に制御する機能が組み込まれていた（複数のプロセスから共有されているため。スレッド対応プログラムを書いたことがある人なら分かるであろう）。そしてまず基本的な方針として、システムコールの処理は途中で中断されないように作られている。

ただし、たとえばファイル読み出しのシステムコールではディスクからの入力待ちが生じるので、このような場合は、自発的にプロセス自身がsleep()して、他のプロセスに積極的にCPU時間を渡すというふうにインプリメントされている（ユーザープロセスはプリエンプティブなマルチタスクだが、カ

ーネル内部は協調的マルチタスク方式）。だから、カーネル内でのプロセスの停止は、必ずsleep()や、ほかのシグナル待ちなどでしか起こらないということになる（WCHANのアドレスが常にsleepなどの「きり」のよい値になっているのはこのためである）。オリジナルのUNIXのインプリメンテーションがリアルタイム処理に向いていないのは、このような事情による。割り込みが発生しても、現在実行中のシステムコールなどの処理が終わらない限りタスクスイッチが行われないので、一定時間以内のタスクスイッチを保証することができないのである。

なお、もっと細かい単位での排他制御では（例：プロセステーブルの更新とページテーブルの更新を連続して不可分で行うなど）、一時的に割り込みを禁止にしたり（割り込み禁止フラグを立てる）、割り込み許可レベルを高くしたりする（緊急度の高い割り込みだけはとりあえず受け付けることができるようにするため）などの手法が使われている。

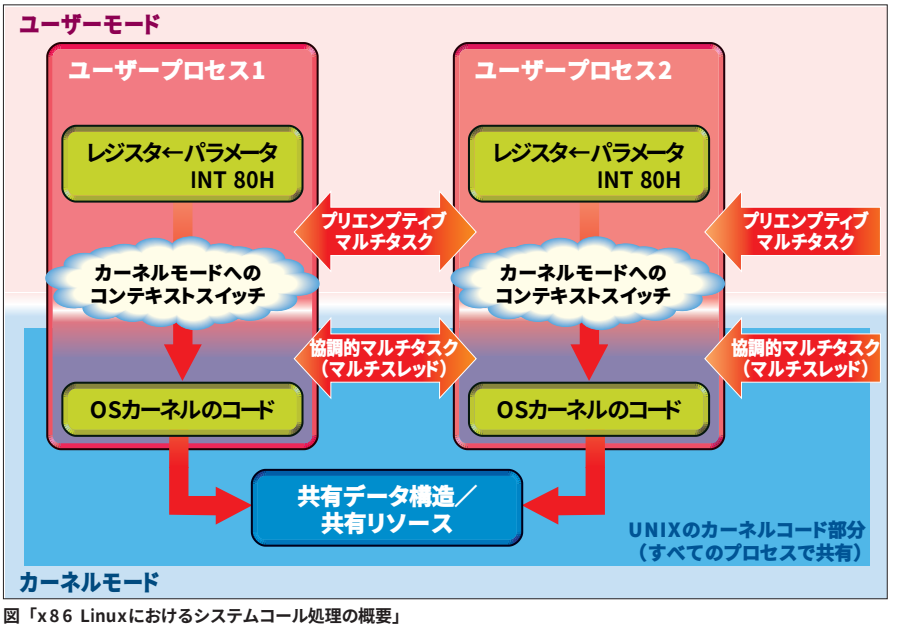
しかしマルチプロセッサシステムでは、これらだけでは十分とはいえない。割り込み禁止などの措置は、それを実行しているプロセッサでのみ有効な方法であり、他のプロセッサからのメモリアクセスを禁止することはできないからだ。だからマルチプロセッサシステムでカーネルコードを同時に実行させると、データ構造の一貫性が壊れる可能性がある。

そこでLinuxの2.0では、とりあえずカーネルコード全体を排他的に処理するという方法を採用した。つまり、複数のプロセスからシステムコールが発行されても、どれか1つのプロセッサでのみ逐次的に処理するのである。パフォーマンスを考えるとあまり望ましい方法ではないが、開発工数を抑えてマルチプロセッサ対応させるという点では、有効な方式だといえる。もともとカーネルコードの実行時間はユーザープログラムに比べると非常に短いので、システムコールの頻度が多くない環境では十分実用的である。

ただ、ネットワークやディスク入出力、タスクスイッチなどの頻度が高くなる用途（エンタープライズ向け用途など）ではやはり望ましくないので、Linuxの2.2以降では、カーネルコードをマルチスレッドで実行できるように改良が施された。

まず排他制御の方式として、マルチプロセッサシステムで使われているハードウェアスピンロック方式を取り入れている。スピンロックとは、条件が満たされるまでその場でループして待つための命令のことであるが、排他制御したい部分が小さければ、十分実用的な方法である。これを実現するためには、メモリの読み出しと値のセットというメモリアクセスサイクルを、他のプロセッサに割り込まれずに不可分に実行する必要があるが、CPUにはこのための命令が用意されている（Test and Set命令など）。これを使って排他制御したいコード部分を保護する。

さらに、排他制御の単位を細かく管理して、複数のプロセッサ上で不必要にお互いを待つことがないようにしている。例えば、プロセステーブルとi-nodeテーブルを同時に操作しないということがあらかじめ分かっていれば、これらを排他制御するために異なるスピンロック変数を使うことができる。もし同じスピンロック変数を使って排他制御を行うと、プロセステーブルを操作する処理によって、まったく関係のないi-nodeテーブルの処理ルーチンが待たされてしまうことになるからだ（シングルプロセッサのときはこれでも問題ない。もともと同時に実行される



図「x86 Linuxにおけるシステムコール処理の概要」
ユーザープロセスでシステムコールを発行すると（x86アーキテクチャでは、レジスタにパラメータをセットしてINT 80Hを実行する）、プロセスのコンテキストが「ユーザーモード」から「カーネルモード」に切り換わり、カーネル内のコードへと制御が移行する。カーネル内では、複数の（カーネルモードの）プロセスがリソースを共有しながら、マルチスレッドで実行している。そのため、各所にカーネルリソースの競合を防ぐための排他制御機構が組み込まれている。ユーザープロセスは任意の時点で割り込まれるプリエンプティブマルチタスクであるが、カーネル内部ではシステムコールの処理が終了するか、入出力の完了待ちなどの時点でのみプロセスのコンテキストスイッチが行われる。

ことはないからだ)。

以上のような処理によって、カーネルの2.2以降では、カーネルコードもマルチプロセッサで動作するようになっている。

■カーネルスレッドのサポート

Linuxではこのほか、カーネルサービスをサポートするために、カーネルモードでいくつかのスレッド(カーネルモードスレッド)が動作している(**図「Linuxのプロセス例(カーネルモードスレッド)」**)。init(初期プロセス)のほか、kflushd(ディスクなどバッファの書き出しのためのデーモン)、kswapd(プロセスの自動的なページアウトのためのデーモン)などである。

これらは、定期的な処理や、システムコール処理を補助するためのルーチンであるが、プロセスとして実装した方が望ましいので(ユーザープログラムの状態に関わらず、常に動作している必要があるため。プロセス形式になっていないと、システムコールの処理時やタイマー割り込み時などに付随的に動かすことしかできないので、非常に扱いづらいし、プログラミングも困難になる)、特別にカーネルモードで動作するプロセスとして実装されている。

一般のユーザープロセスとの違いは、コードがカーネル内部に置かれていて、カーネルとデータ構造を共有しているという点にある。カーネル内部のデータへアクセスする場合のオーバーヘッドがなくなるし、カーネル内のルーチンを呼び出す場合でも重いコンテキストスイッチが不要になるので、高速、軽量なサービスを実現することができる。

マイクロカーネル方式のWindows NT

Windows NTは、このマイクロカーネル構造をベースにしたOSである。ただし純粋なマイクロカーネル方式ではなく、パフォーマンス向上のために、かなりの部分を特権モードで動作するNT Executiveという形で実装している(**図「Windows NTのOS構造」**)。

カーネル側で各種のサービス(ファイルシステムドライバやウィンドウマネージャ、GDIサービスなど)が動作しているのは、すでに述べたように、性能低下に対する対策である。開発当初のNTでは、GDI、すなわち描画ルーチンはユーザープロセスとして実装されていたが、現在ではその一部がカーネル側に移されるなど、さらにチューニングが進んでいる。

図「NTのタスクマネージャ」に示したのは、あるNT Serverマシンにおけるタスクマネージャの表示例である。このうち、「System」として表示されているのがカーネルモードで動作しているOS部分を表している。カーネル内部だけでも数10個のスレッドが動いているが、それ以外にもカーネル外部でシステムをサポートするために数多くのサービスプロセスが動作している。

グラフィックス/マルチメディア機能の比較

次に、両OSにおける、グラフィックスなどのサポートについて見ておこう。

■Windows NTのグラフィックス機能サポート

Windows NTは、もともとWindows 3.1やWindows 95などから継承したWin32 APIセットをサポートしているため、2Dや3Dのグラフィックス機能をサポートしている。Win32だけでは足りない用途に対しては、OpenGLやDirectXなどのグラフィックス用途に最適化されたAPIセットも用意されている。グラフィックスチップの持つ機能も積極的にサポートしているし、場合によっては次々と仕様を拡張して、機能強化を図っている。もちろん、相次ぐ仕様の追加や変更はプログラマー

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0	0	768	76	?	S	17-Aug	1:22	init
root	2	0	0	0	0	?	SW	17-Aug	0:10	[kflushd]
root	3	0	0	0	0	?	SW	17-Aug	0:00	[kpiod]
root	4	0	0	0	0	?	SW	17-Aug	1:02	[kswapd]
root	118	0	0	260	48	?	S	17-Aug	0:00	/sbin/dhcpcd-c
bin	163	0	0	776	0	?	SW	17-Aug	0:00	[portmap]
root	175	0	0.1	824	292	?	S	17-Aug	1:30	syslogd
root	182	0	0	1084	0	?	SW	17-Aug	0:00	[klogd]
daemon	191	0	0	804	108	?	S	17-Aug	0:00	/usr/sbin/atd
root	200	0	0	812	128	?	S	17-Aug	0:26	crond
root	209	0	0	796	76	?	S	17-Aug	0:00	inetd
root	228	0	0.8	5064	2220	?	S	17-Aug	0:01	/usr/bin/atok12x
root	240	0	0	1204	64	?	S	17-Aug	0:00	sshd
bin	245	0	0	1416	48	?	S	17-Aug	0:01	/usr/sbin/cannase
root	259	0	0	744	0	tty5	SW	17-Aug	0:00	[mingetty]
root	260	0	0	744	0	tty6	SW	17-Aug	0:00	[mingetty]
root	262	0	0	736	24	?	S	17-Aug	1:54	update(bdflush)
wnn	26736	0	0	1268	0	?	SW	18-Aug	0:00	[jserver]
wnn	26737	0	0	19096	0	?	SW	18-Aug	0:00	[jserver]
root	29757	0	0	744	0	tty4	SW	18-Aug	0:00	[mingetty]
root	3013	0	0	744	0	tty2	SW	20-Aug	0:00	[mingetty]
root	12163	0	0.5	3464	1396	?	S	21-Aug	1:50	/usr/X11R6/bin/xt
tadayo-m	12164	0	0	1836	0	ttyp0	SW	21-Aug	0:00	[bash]
root	7771	0	0	744	0	tty3	SW	21-Aug	0:00	[mingetty]
root	7811	0	0	1840	0	?	SW	21-Aug	0:00	[smbd]
root	7818	0	0.1	1476	372	?	S	21-Aug	0:35	nmbd#NAME?
root	7823	0	0	744	0	tty1	SW	21-Aug	0:00	[mingetty]
root	4981	0	0	1452	0	ttyp0	SW	22-Aug	0:00	[su]
root	4982	0	0	1372	0	ttyp0	SW	22-Aug	0:00	[bash]
root	4990	0	0	1448	0	ttyp0	SW	22-Aug	0:00	[su]
root	4991	0	0.2	1924	608	ttyp0	S	22-Aug	0:00	#NAME?
root	5003	0	0	1168	0	ttyp0	TW	22-Aug	0:00	[pppd]
root	13661	0	0.3	2416	1012	?	S	23-Aug	0:00	xdm
root	13663	0	1.1	13292	2936	?	S	23-Aug	0:00	/usr/X11R6/bin/X
root	13664	0	0.5	2312	1292	?	S	23-Aug	0:00	-:0
root	15213	0	0.3	2176	780	ttyp0	R	23-Aug	0:00	ps aux

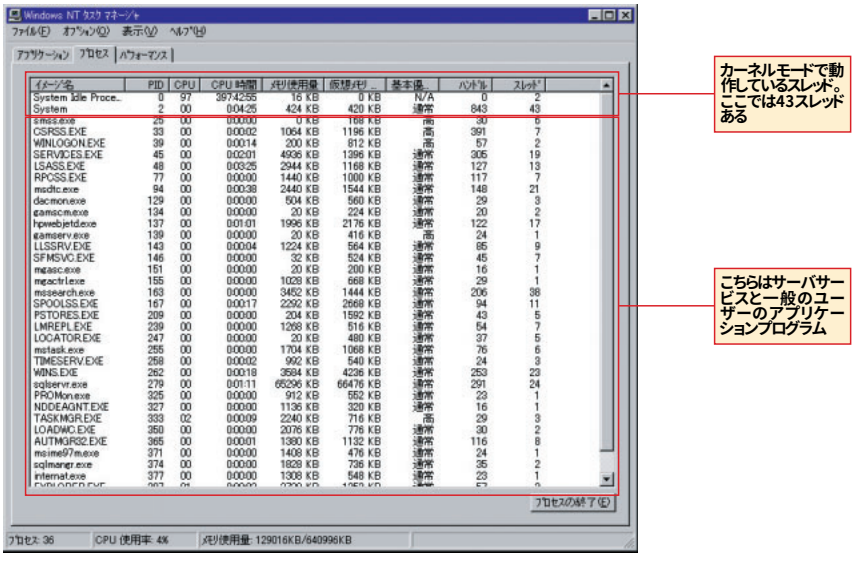
カーネルモードスレッドの例。ここには3つ(PID=2～4)ある。独立したユーザープロセスではないので、VSZやRSSフィールドは0になっている

こちらはサーバサービスや一般のユーザーのアプリケーションプログラム

図「Linuxのプロセス例(カーネルモードスレッド)」
あるLinuxマシンにおけるカーネルモードスレッドの例。カーネルモードスレッドとしてkflushd、kpiod、kswapdが動作しているのがわかる。

にしてみればあまり嬉しくないのかもしれないが、たとえば現状のDirectX 6.0だけではチップの持つ高性能な3次元描画機能を有効に活かすことができないことを考えると、これを避けるのは難しい。

グラフィックスチップの機能強化はこれからもさらに進み、たとえば3Dグラフィックスならば、ポリゴンデータの座標計算なども描画パイプラインにマージされるであろう。しかし一方、CPU側でなるべく多くの計算処理を担当するというシステムもあるはずなので、さまざまなコンフィギュレーションに対応できる、より柔軟なAPIセットが求められることになるだろう。



図「NTのタスクマネージャ」
あるWindows NT Server 4.0マシンにおける、タスクマネージャの例。カーネルモードで動作している各種のExecutiveサービスなどは、すべて「System」というプロセス（プロセスIDは2）の所に表示されている。ユーザーモードで動作しているサーバサービスと、一般のユーザープログラムの区別は特にない。

■Linuxのグラフィックス機能サポート

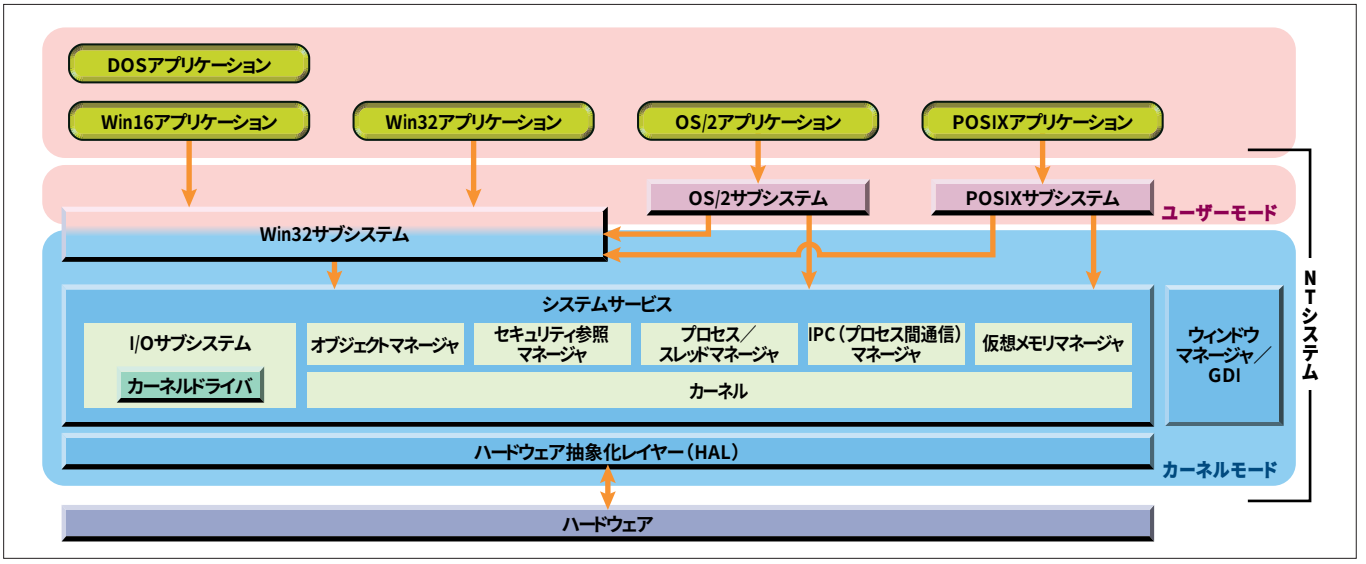
Windowsと同じような環境を求めようとすると、Linuxでは、X Window Systemをベースにすることになるであろう。

しかし実際には、特に日本語環境を考えると、ワープロや表計算などの用途ではLinuxはあまり向いていないといえる。アプリケーションはいくつかあるものの、既存のワープロ製品の置き換えやデータの互換性などを考えると、Linuxだけで既存のWindowsを置き換えるにはまだ不十分というのが実状といえる。製品の選択肢が少ないだけでなく、機能もまだ十分ではない（使うかどうかは別にして）。

また、X Window Systemは印刷環境に対しては何のサポートもしてくれないので、データを印刷するためには、現状では各アプリケーションがプリンタの制御コードを独自に出力するしかないようである。

となると、サポートできるプリンタの種類にも限界があるし、今後も急激に環境が改善される見込みはあまりないように思える。

もしWindowsなどの環境をLinux上でも実現したいのならば、高性能なWindows環境エミュレータが開発されるのを待った方がいいかもしれない。



図「Windows NTのOS構造」

ユーザーの管理・認証

編集部

複数のユーザーが使用する（マルチユーザー）OSの場合、そのユーザー群をどのように管理するかということは、非常に重要なOSの機能である。今日ではネットワークサポートがOSにとって当たり前になったため、ネットワーク接続されたコンピュータ全体でのユーザー管理機能も必要とされている。

Linuxのユーザー管理・認証

Linuxでのユーザー管理は、一般的なUNIX同様、使用者ごとに「ユーザー」というアカウントを発行して管理する。ユーザーアカウントは、パスワードで認証して使用者を確認し、利用できるリソースを制限したり、使用者のプライベートな設定を有効にしたりするのに使われる。

またユーザーを集団ごとに管理するために「グループ」というアカウントが存在する。グループを使えば、リソースへのアクセス制限などの設定が容易になる。

ユーザー／グループアカウントには、それぞれUIDとGIDというユニークな番号が割り当てられる。これはrootアカウントといくつかのウェルノウンアカウントを除けば、管理者が任意に番号を指定できる。

アカウント情報の保管先は、/etc/passwd（ユーザー）、/etc/groups（グループ）である。最近は、シャドウパスワード方式が採用されているケースが多く、この場合、パスワードはrootしか読めない/etc/shadowに別保存される。これは、/etc/passwdが一般ユーザーでも読み出せるためセキュリティ面で好ましくないことと、読み出しを禁止してしまうとホームディレクトリやシェルなどを参照する多くの機能で不具合が生じてしまうことからだ。

以上は単体のLinux環境でのユーザー管理だったが、複数のLinux環境ではNISなどのディレクトリサービスを利用することになる（**コラム「NIS（Network Information Service）」**を参照）。

Windows NTのユーザー管理・認証

Linuxを含むUNIX系OSは、多少の違いはあるが、ユーザーの管理方式はよく似ている。これに対してWindows NTのユーザー管理は、似ている点もなくはないが、UNIX系OSとはかなり異なる印象を受ける。設計思想が異なるということだろう。以下では、Windows NTのユーザー管理・認証機能のうち、LinuxなどUNIX系OSとは異なる独自の部分に焦点を当ててみた。

■アカウントを識別するSID

LinuxでもWindows NTでも、ユーザーやグループなどのアカウントを作成すると、必ずIDと呼ばれる数値が割り当てられる。システム内部では、アカウント名より扱いやすいこのIDで管理されている。

双方のOSで異なるのは、このIDの

Column

NIS（Network Information Service）

Linuxを含むUNIX系OSにも、NTのドメインのようなネットワーク上のコンピュータを論理的なグループにまとめ上げる機能がある。代表的なのはNIS（Network Information Service）やNIS＋である。どちらもSun MicrosystemsのUNIXに初めて実装されたもので、NIS＋はNISに対して性能やセキュリティなどを強化したバージョンである。ノベルのNDSのようなディレクトリサービスの一種といえる。

NISの目的は、UNIX系OSで動作する複数のコンピュータ間で、ユーザー名やグループ名、パスワードなどの情報を共通に保つことである。NISを導入すると、NISドメインと呼ばれる論理的なグループに参加しているコンピュータには、どれも同じユーザー名／パスワードでログインできる。

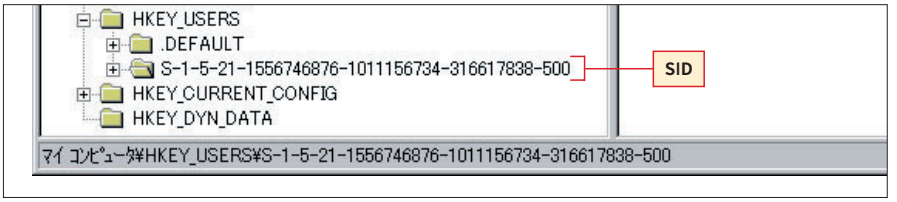
NTのドメインとNISの間で大きく異なるのは、NTドメインではユーザー認証までドメインコントローラに集中していたのに対して、NISでは各コンピュータでユーザー認証が行われるという点である。NISのクライアントは、ユーザー情報などNISドメインで共通の情報をNISサーバに要求して入手し、ユーザーの認証を行う。

またNISが共通に保つファイルには、hostsなどユーザー管理と直接関係のないファイルも含まれるし、異なるOS同士でもNISにさえ対応していれば同じNISドメインで情報の共通化が図れるなど、NTにはない特徴も多い。

つけ方である。Linuxではroot以外、管理者がIDの数値を指定できるし、その値も単純な数値でよい（rootは必ず0である）。Windows NTの場合、IDはSID（Security Identification、セキュリティ識別子）と呼ばれ、システムから唯一無二の数列が自動的に割り当てられる。つまり管理者にもSIDの数列は自由に指定できない。SIDの実体は、**画面「Windows NTのSIDの例」**のように非常に長い数列である。

SIDが割り当てられるのは、実はユーザーアカウントだけではない。Windows NTが取り扱うオブジェクトのうち、保護される必要があるもの、つまりセキュリティが考慮されるオブジェクトに対しては、固有のSIDが割り当てられる。当然グループアカウントにもSIDがつけられる。また最初から作成される予約済みのオブジェクトには、どこでも同じSIDが割り当てられている。

SIDが唯一無二の固有なIDであり手動で指定できないという特徴が、LinuxとWindows NTの間で、ユーザー／グループアカウントの取り扱い方の差を生んでいる。たとえば、ある既存のユーザーアカウントを削除し、再び同じ名前でユーザーアカウントを作成したとしよう。Windows NTの場合、削除前のユーザーアカウントで所有していたファイルに対して、同じユーザー名でアクセスしても、その権利がないとされてアクセスできないのだ。これは削除前と削除後で、アカウントのSIDが異なるため、システムが別のユーザーと認識するためである。Linuxなら、削除前のユーザー名とユーザーIDを使ってユーザーアカウントを作成すれば、アカウントは復活して以前使っていたリソースにもアクセスできるはずだ。



画面「Windows NTのSIDの例」

また、複数のマシンを同じユーザーが使用する場合、Linuxならユーザー名とユーザーID、グループID、パスワードなどを統一して各マシンにアカウントを作成すればよい。しかし、Windows NTでは同一のSIDを持つユーザーを作成できないので、複数のマシンにまったく同じユーザーアカウントを用意するのは不可能だ。そこでアカウント情報は一箇所に集中してユーザー認証もそこで行い、リソースを持つマシンはそこへユーザー認証を依頼する、というスタイルをとる（これが後述のドメインによるユーザー認証である）。

■アカウント情報の格納先

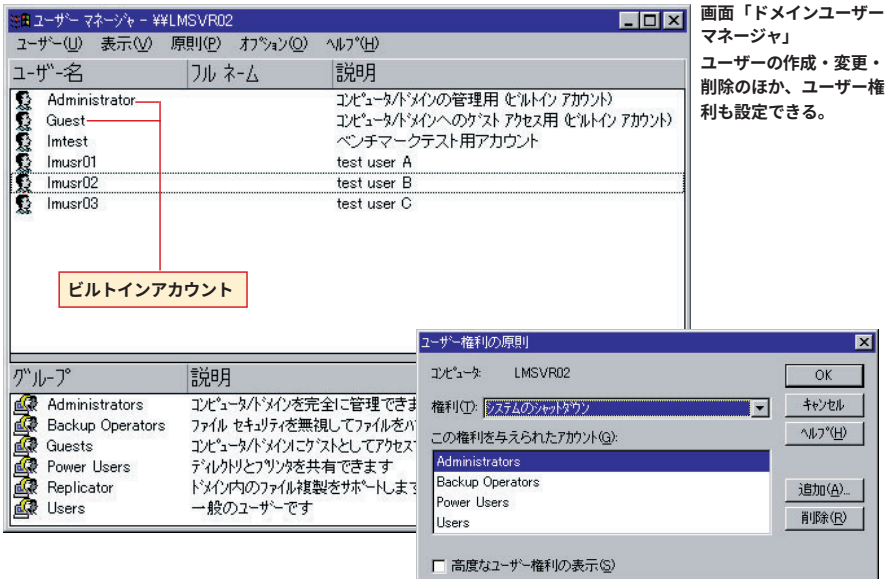
Linuxの場合、ユーザーアカウント情報は/etc/passwd（シャドウパスワードは/etc/shadow）に、またグループ情報は/etc/groupに保存される。いずれもプレーンテキストファイルであり、/etc/shadow以外は一般ユーザーでも容易に読み出せる。

これに対してWindows NTでは、ユーザー／グループの情報はSAM（Security Account Manager）データベースに保存される。これはアカウントのパスワードをキーとして暗号化されたデータをまとめたバイナリファイルだ（実体は<WinNTディレクトリ>\system32\configにあるSAMというファイル）。一般ユーザーはSAMデータベースをオープンできないし、Service Pack 4以降ではSAM全体をさらに暗号化できるようになった。これでSAMデータベースをバックアップしたファイルをクラックして、パスワードを得るといったことも難しくなる。

図「Windows NTのセキュリティに関わるコンポーネント」は、Windows NTのセキュリティ関連コンポーネントの連携を表している。ユーザーが対象のシステムにログオンしようとすると、そのアカウント名でSAMデータベースが検索され、パスワードが照合される（ユーザー認証）。パスワードが合っていたらSAMデータベースから関連するSIDが抽出され、アクセストークンと呼ばれる情報のセットに格納される。以後、そのユーザーがファイルなどにアクセスしようとすると、アクセストークン内のSIDとファイルのアクセス権のリスト中にあるSIDとが照合され、アクセス可能かどうかチェックされる。SAMとSAMデータベースはログオン時の要として働くわけだ。そのため、セキュリティ面では前述の暗号化のように注意が払われている。

■ビルトインアカウント

画面「ドメインユーザーマネージャ」は、Windows NTでユーザーやグループのアカウントを作成・設定するツールである。この画面のユーザー名の欄にあるAdministratorとGuestというユーザーアカウントは、システムによって最初から組み込まれているビルトインアカウントである。これらのアカウントは、Windows NTをインストール



画面「ドメインユーザーマネージャ」
ユーザーの作成・変更・削除のほか、ユーザー権利も設定できる。

すると必ず作成される。

AdministratorはUNIX系OSのrootに相当するユーザーアカウントで、システムの管理に利用する。もう一方のGuestは、アカウントを持っていないユーザーからのアクセスを制御するのに使われる。アカウントのないユーザーからのアクセスには、このGuestアカウントが使われるので、Guestアカウントからのアクセスを許可すれば、誰からでもアクセス可能なリソースを用意できる。インターネットなど外部接続のない小規模なLAN環境では、Guestアカウントを使うとアクセス権の設定が容易になるなどのメリットはあるものの、セキュリティホールにもなり得るので、デフォルトではGuestアカウントは無効になっている。

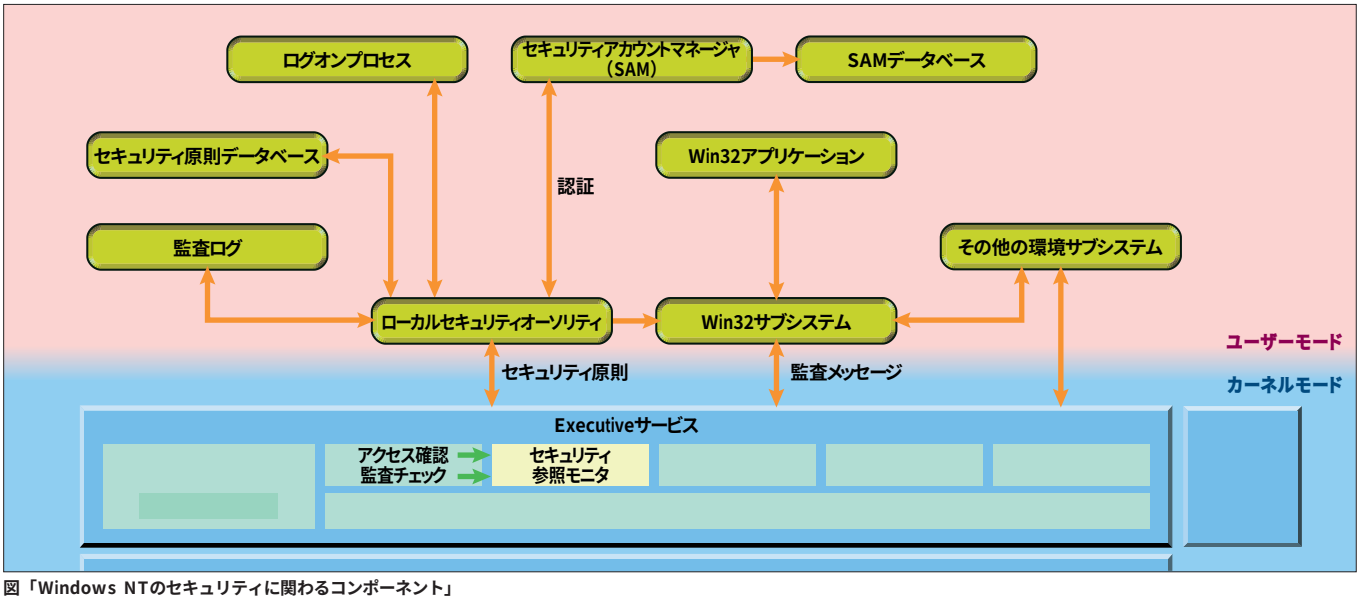
グループについても、特殊な機能をもったビルトインアカウントがいくつかある。たとえば、システム管理者のためのグループとしてAdministratorsグループが、またファイルのバックアップを担当するグループとしてBackup Operatorsグループが用意されている、といった具合だ。必要に応じて、ユーザーアカウントをこれらのグループに入れれば、各ビルトイングループに与えられた機能が利用できるようになる。

このほかにも、システムが提供する各種機能をユーザーやグループに「権利」として与えることができる。**画面「ドメインユーザーマネージャ」**の右下のように、システムをシャットダウンしたり、システム時刻を変更したりするといった権利を、ユーザーやグループに与えられる。実際には、グループを作成して権利を与え、その権利を必要とするユーザーをグループに入れる、という方法が推奨される。

なおLinuxにもrootをはじめとするユーザーやグループが最初から用意される（標準ユーザー／標準グループと呼ばれる）。しかし、Windows NTのビルトインアカウントのほうがより細かく定義されている。

Windowsネットワークでのユーザー管理・認証

以上はWindows NT単体でのユーザー管理について記したが、もともと



図「Windows NTのセキュリティに関わるコンポーネント」

Windows NTには、Windowsネットワーク全体のユーザー管理を担当する能力がある。次は、Windows NTがどのようにWindowsネットワークでのユーザー管理・認証に関わるかを説明してみる。なお、ここで使っている「ドメイン」とはWindows NT独自の概念であり、DNSにおける「ドメイン」とはまったく異なることに注意していただきたい。

■論理的なグループの必要性

Windowsネットワークでは、複数のコンピュータを束ねる論理的なグループとして、ワークグループとドメインの2種類が用意されている。論理的なグループとは、ネットワーク構造に依存しないグループのことで、複数のサブネットをまたいでグループを形成したり、WAN経由のリモートコンピュータも加えたりすることができる。ワークグループやドメインという論理的なグループの目的は、実際の組織構成に適した形で複数のコンピュータをグループ化することと、それにより各種リソースの共有やユーザー情報の管理などを効率よく行うことが挙げられる。

■ワークグループ

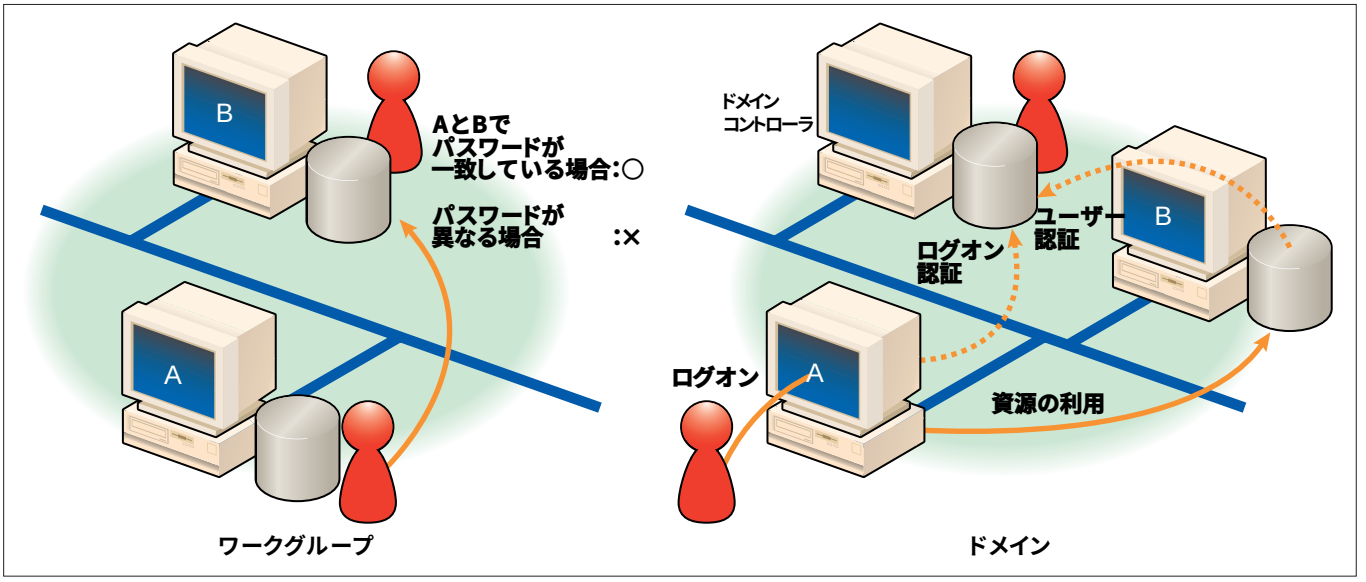
ワークグループはコンピュータ数もそれほど多くない小規模なWindowsネットワークに適した論理グループである。なぜなら、ユーザーやリソース、セキュリティの管理を一元化して効率を上げるという機能がワークグループにはほとんど備わっていないからだ。実際の組織構成に合わせて、ファイル／プリンタ共有などのリソースをグループ分けすることだけが目的といってもよい。

ワークグループで各コンピュータのリソースにアクセスするには、基本的にコンピュータごとに配置されたユーザー情報をもとにユーザー認証が必要になる。つまりコンピュータごとにいちいちログオンする必要がある。この手間を省くには、ワークグループ内のすべてのコンピュータでアカウント名とパスワードの組を同じものに統一する必要がある（ログオンが透過的に行われる）。逆にいえば、こうしたユーザー管理の手間が許容できる程度の台数しか、現実的にワークグループでは管理しきれないといえる。

■ドメイン

ワークグループに対するドメインの特徴は、ユーザー管理を一元化したことである。ドメインでは、ドメインに参加するコンピュータ全体に通用するユーザー／グループアカウントが提供される。

ドメインのユーザー情報は、ドメインコントローラと呼ばれるコンピュータで保持・管理される。ドメインコントローラとなったWindows NTマシンでは、**図「Windows NTのセキュリティに関わるコンポーネント」**のSAMデータベースにドメインのユーザー情報が蓄積されるほか、ドメインのユーザーに対する認証も行われる。リソースを公開する各コンピュータは、そのリソースにアクセスする権利をドメインのユーザー／グループに与えることで、ユーザー認証自体はドメインコントローラに任せることができる（**図「ドメインとワークグループでのユーザー認証の違い」**）。このようにユーザーの管理・認証はドメインコントローラに集中するため、ワークグループに比べてその管理ははるかに容易になる。もちろん、ユーザー数やコンピュータ数も



図「ドメインとワークグループでのユーザー認証の違い」　認証に使われるユーザー情報のありかが異なる

ワークグループより多くすることができる。

複数のドメイン同士で連携することも可能だ。ドメイン同士で信頼関係を結ぶと、一方のドメインにある共有リソースに対して、他方のドメインユーザーがアクセスする権利を持たせられる。ただし、**コラム「ドメインによる管理の限界」**にあるように、Windows NT 4.0の複数ドメインの取り扱いには制限が多く、あまり多くのドメインを扱うのは現実的ではない。

ドメインで集中管理できる情報の種類にも注意したい。ドメイン内の各コンピュータが公開している共有リソースに関わる情報（共有名やアクセス権、ファイルへのパスなど）は、そのリソースを持つコンピュータに保存される。集中管理されるのは、あくまでドメインユーザー情報である。したがって、いくらドメインコントローラの保持するファイルやデータをバックアップしても、リソースを公開しているコンピュータのシステムが破壊されると、共有名やアクセス権などの情報が失われてしまう。

Linux＋Sambaでのユーザー管理・認証

Linuxのカーネル自体には、ネットワーク全体に関わるユーザー管理機能は含まれていない。しかし、Linuxのディストリビューションなら NISと呼ばれるディレクトリサービスが含まれていることが多く、これを用いればネットワーク全体のユーザー管理をLinuxが担当できる（NISについては**コラム「NIS（Network Information Service）」**を参照していただきたい）。

NISに対応しているLinuxやUNIX系OSだけしかネットワーク上に存在しないなら、NISを利用すればよい。しかし実際には、サーバがLinuxであってもクライアントはWindowsというネットワークのほうが、現時点では大多数だろう。標準ではWindowsはNISクライアントになれないので、NISによる管理は非現実的だ。

一方、Windowsネットワークにおいてサーバ でWindowsがクライアントという構成なら、Sambaによるファイル共有サービスは必須といえる。LinuxとSambaを組み合わせた場合、Windowsネットワークでのユーザー管理にはSambaが深く関わる。そこで以下では、Sambaによるユーザー管理の特徴を見ていく。なおSambaのファイル／プリンタ共有サービスについては、84ページからの記事を参照していただきたい。

■多様なユーザー認証

Sambaによって公開されているファイル共有に対して、ユーザー認証は次の3種類から選択できる。

①Linuxのユーザー認証を使う
②Samba独自のユーザー認証を使う
③Windows NTサーバ／Windows NTドメインのユーザー認証を使う

このように3種類もの認証方法が提供されているのは、Windowsの暗号化パスワードが原因の1つである。Windows 95 OSR1以前やWindows NT 4.0 Service Pack 2以前では、ネットワーク経由でログオンする際、パスワードは暗号化されない平文で送出されていた。この場合、Sambaが受け取るパスワードも平文なので、①Linuxのユーザー認証がそのまま利用できる（telnetで平文パスワードを送って認証されるのと同じ感覚である）。

ところがWindowsのバージョンが上がると、セキュリティの強化策としてネットワークに送るパスワードが暗号化されるようになった。LinuxとWindowsの間で、パスワードの暗号化アルゴリズムに互換性がないため、Linuxの認証ロジックをそのまま利用することはできない。そこで、Samba

Column

ドメインによる管理の限界

ユーザー数やドメイン数が増えてくると、より効率的にユーザーを管理するような体制にしないと、ユーザー管理の手間は増えるばかりである。グループを活用するのは、その手間を減らす手の1つである。しかし、Windows NTのグループの扱いには制限がある。たとえば、ドメインで管理するグループにはそのドメインのユーザーしか追加できない。つまりグループに別のグループを入れたり、別のドメインのユーザー／グループを入れたりすることはできない。そのため、複数の部と課にそれぞれグループを割り当てて管理するなどといった、実際の組織構成に合わせたユーザー管理が難しい。

ドメインの扱いにも同様の欠点があり、ツリー状に複数のドメインを束ねたりすることはできない。またドメイン間でユーザーの移動もできない。

このように現時点のWindows NT 4.0では、組織構成が3階層以上あるようなある程度大きい組織だと、単体で管理するのが難しい。次期製品のWindows 2000ではこうしたドメインやグループの制限が少なくなるはずである。またWindows NT 4.0でも、ノベルのNDS（Novell Directory Service）などのディレクトリ管理ソフトウェアと組み合わせるといった解決法もある。いずれにせよ、ユーザー管理の面では、単体のWindows NTは比較的規模の小さい組織での運用に向いているといえる。

が独自に認証ロジックを用意し、Windowsの暗号化パスワードでも認証できるようになった。

③のWindows NT サーバ／ドメインのユーザー認証は、すでにWindows NTドメインがある環境でSambaによるファイルサーバを運用する場合、他のWindows NT サーバと同様、ドメインユーザー情報が使えるので便利だ。また、ワークグループの環境でもWindows NT Server／Workstationマシンがあれば、そのマシンにSambaとWindows 95／98のユーザー認証を任せ、ユーザー管理をより集中させることも可能だ。

ここで問題になるのは、②と③のユーザー認証を用いる場合、LinuxとSambaあるいはWindows NTとの間でユーザーアカウントを二重に管理しなければならない点である。一応、Sambaのツール（smbpasswd）でパスワードを変更すると自動的にLinuxのパスワードも自動的に更新する、という仕組みがある。しかし、smbpasswdを介さずにLinuxやWindows NTで直接パスワードを変更したら意味がない。運用面での注意が必要である。エンドユーザーには**画面「GUIによるSambaのパスワード設定」**のようにWWWブラウザ経由でパスワードを変更させるとよいだろう。

なお②の場合、Windowsクライアントのレジストリを操作することで、ネットワークに送出するパスワードを暗号から平文に変更することができる。すべてのクライアントにこの処理を行えば、単純な①の認証方式で済むというメリットはある。しかし、実際には複数台あるクライアントのレジストリを操作するのは面倒であり、お勧めできない（ミスするとブートできなくなる可能性がある）。もちろんネットワークでパスワードを盗聴される危険性が高いこともデメリットといえる。

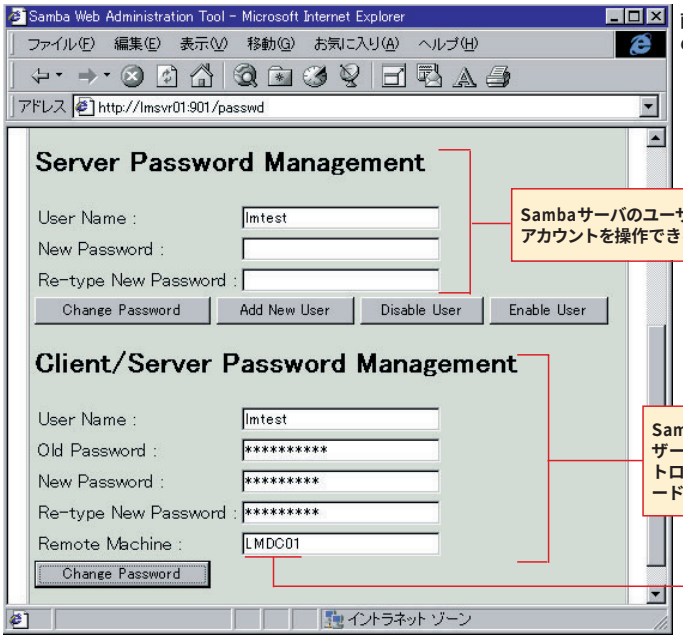
■ドメインログオンのサポート

以上のほかに、Sambaがサポートするユーザー管理機能としては、Windowsクライアントからのドメインログオンが挙げられる。これはSambaサーバがドメインコントローラとして働くことを意味する。また、ユーザープロファイルやログオンスクリプト、システムポリシーといった、もともとはWindows NTのもとで実現されていた機能も、Sambaで実現できる。

ユーザープロファイルとは、ユーザーがWindowsマシンにログオンしたときに表示されるデスクトップの作業環境を保存したデータである。具体的には、画面のデザインや壁紙、ネットワーク／プリンタ接続などといった作業環境の設定がユーザープロファイルという形で保存される。このユーザープロファイルをサーバに保存しておき、ユーザーがドメインにログオンしたらそのWindowsクライアントにユーザープロファイルをロードして、前の作業環境を再現する、というのが移動ユーザープロファイルの機能だ。これにより、どのWindowsクライアントにログオンしても、自前の作業環境をある程度再現することができる（原理的にアプリケーション環境の再現までは無理）。

ログオンスクリプトは、ユーザーがドメインにログオンした直後にWindowsクライアントで自動実行されるバッチファイルである。またシステムポリシーは、Windowsクライアントのシステム構成を強制的に変更する機能で、勝手にWindowsクライアントの設定を変更できないようにする、といった使い道がある。

この3つの機能はいずれもWindows NT Serverによるドメインコントローラに実装されている機能だ。後述するように、現時点のSambaが持つドメインコントローラの機能はまだ限定的だが、単なるWindows用ファイルサーバから着実に進化しているのは間違いない。



画面「GUIによるSambaのパスワード設定」

ファイル／プリンタ共有サービス

編集部

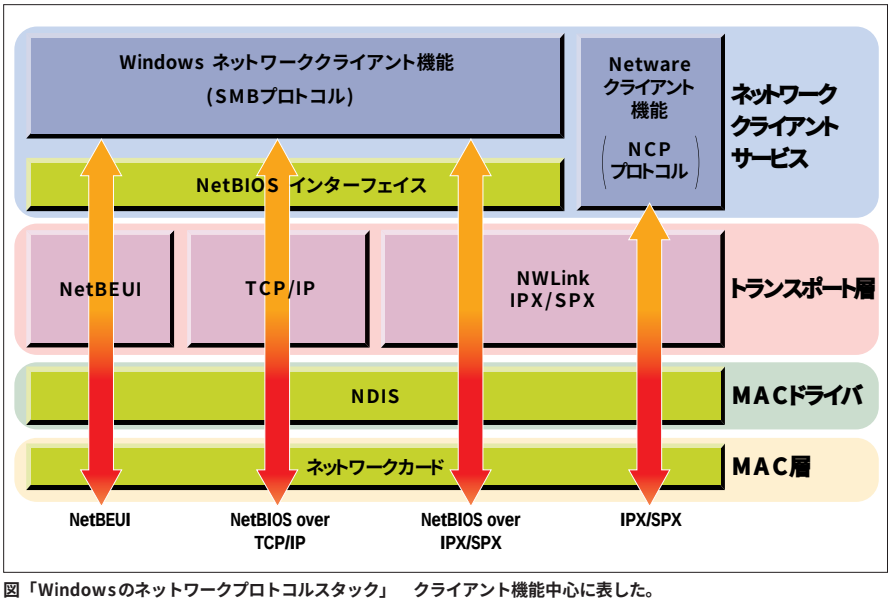
SOHOや企業内の部門クラスの組織でサーバマシンを業務に使っている場合、必ず利用されているサービスといえば、やはりファイル／プリンタ共有ではないだろうか。データベースやグループウェアよりは原始的なサービスだが、Windows 95／98が標準でファイル／プリンタ共有のクライアントおよびサーバとなれるので、運用や扱いが容易な分、導入しやすいのだろう。

このようによく使われているファイル／プリンタ共有というサービスにおいて、LinuxとWindows NTがどのような機能を提供できるのか比較してみる。ここで想定している環境は、Windowsクライアントが複数存在するネットワークである。

Sambaとは？

Windows NTでは、Windowsクライアントに対するファイル／プリンタ共有サービスが標準的に組み込まれている。これに対してLinuxのカーネルには組み込まれていない。しかし、現在ほとんどのLinuxディストリビューションには、SambaというWindowsのファイル／プリンタ共有を実現するソフトウェアが入っており、LinuxをWindows用ファイル／プリントサーバに変身させることができる。

より正確にいえば、Sambaは単なるWindowsのファイル／プリンタ共有サービスではなく、Windowsネットワーク環境においてWindows NTが提供するファイル／プリンタ共有を含む各種サービス・機能を、別のOSで実現するためのソフトウェアである。Samba自体は、Linuxだけではなく、そのほかのUNIX系OSやOS/2などのプラットフォームで動作するフリーウェアでもある。



図「Windowsのネットワークプロトコルスタック」　クライアント機能中心に表した。

Samba以外にもWindowsのファイル／プリンタ共有を実現できるソフトウェアは存在する。しかしLinuxでは現在Sambaが標準的な地位を得ているので、ここではLinux＋Sambaで提供されるサービスをWindows NTと比べてみる。

Windowsネットワークの構築

ファイル／プリンタ共有サービスを実現するには、Windowsネットワークが正しく稼働していることが前提となる。それにはWindows NTまたはLinux＋Sambaが提供するサービスや機能も影響を与える。まず、双方のOSがWindowsネットワークの構築にどう役立つのか探ってみよう。

■ネットワークプロトコル

Windowsネットワークでのファイル／プリンタ共有には、SMB（Server Message Block）という上位プロトコルが用いられる。下位のプロトコルについては、**図「Windowsのネットワークプロトコルスタック」**のように、WindowsならTCP/IPだけではなくNetBEUIやIPX/SPXというプロトコルも使える。これに対してLinux＋Sambaがファイル／プリンタ共有で使えるのは、TCP/IPだけである。しかし、現状ではWindowsクライアントはTCP/IPを必要とする場面が非常に多く、逆に他の2つのプロトコルはWindowsだけの環境では廃れてきているため、TCP/IPだけのサポートでも

実用上問題にはならない。

■IPアドレスの管理

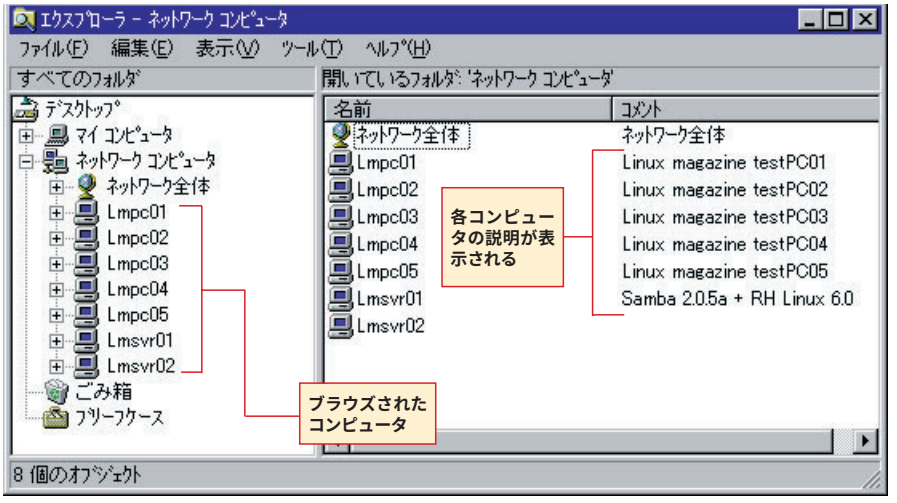
TCP/IPネットワークでは、各コンピュータ（ノード）に対して、IPアドレスを重複なく適切に割り当てる必要がある。それにはDHCPサーバを用いて、自動的にIPアドレスを割り当てたり、各種パラメータ（デフォルトゲートウェイやDNSサーバのIPアドレスなど）を自動配布したりするのが一般的だ。LinuxもWindows NT ServerもDHCPサーバ／クライアント機能を標準装備しているので問題はない。

■名前解決

TCP/IPネットワーク上にある各コンピュータには、固有のIPアドレスとともに、ホスト名あるいはNetBIOS名（Windowsの世界のホスト名）といった固有の名前も割り当てられるのが一般的だ。アプリケーションレベルでは、わかりにくいIPアドレスではなく名前を使ってコンピュータを特定するが、実際にその名前のコンピュータと通信するには、IPアドレスを知る必要がある。そこで「名前」からIPアドレスを得る（あるいはその逆）という作業が生じる。これを名前解決（Name Resolution）と呼ぶ。

NetBIOS名を解決する方法はいくつかある。TCP/IPの世界で一般的なIPブロードキャストやDNSサーバへの問い合わせは、NetBIOS名の解決にも利用されることがある。静的な方法としては、NetBIOS名とIPアドレスの対応を記したlmhostsファイル（あるいはhostsファイル）が使える。しかしWindows NT LANの環境で一番推奨されるのはWINS（Windows Internet Name Service）である。Windows NT Serverに標準装備のWINSサーバは、WINSクライアントになったWindowsマシンからそのIPアドレスとNetBIOS名などの情報を取得し、逆にWINSクライアントからの名前解決の問い合わせに応える。動的にクライアントの登録が可能なDNSサーバと、その動作は似ている。

以上の名前解決の手段は、いずれもLinux＋Sambaで実現できる。Windows NT ServerのWINSサーバに相当するのはnmbdというプログラムだ。ただし現状では完全に純正のWINSサーバと同じ機能を持っているわけではなく、他のWINSサーバと連携できない、同一セグメントに複数のnmbdによるWINSサーバを起動できない、といった制限がある。もっとも、これはWINSサーバ間の通信プロトコルが公開されていないせいでもあるのだが。



画面 「ブラウザにより得られたコンピュータのリスト」

■ブラウザ（コンピュータブラウザ）

Windowsマシンでエクスプローラを起動し、ネットワークコンピュータの中を参照すると、**画面「ブラウザにより得られたコンピュータのリスト」**のようにWindowsネットワークに参加しているコンピュータの一覧が表示される。このように、ネットワーク上で利用できるリソースの一覧（ブラウズリスト）を表示することをブラウジングといい、それを実現するのがブラウザである。Windows NTではComputer Browserサービスが、またSambaではnmbdというプログラムがブラウザとして働く。

ブラウザにはマスタブラウザとバックアップブラウザの2種類があり、前者がブラウズリストを収集して保持し、後者がクライアントにブラウズリストを配布する。マスタブラウザについては、TCP/IPネットワークの場合、複数のサブネットをブラウズするためにセグメントマスタブラウザとドメインマスタブラウザの2種類に細分化される。Windows NTならどのブラウザにもなれるが、Sambaはバックアップブラウザにはなれない。とはいえ、Windows 95／98が代わりのバックアップブラウザとして動作できるので、実用上は大した問題ではない。

逆にSambaがブラウザにならないと困る場合もある。それは、ワークグループを複数のサブネットにまたがって構成する場合で、Sambaをブラウザとしたときだけ全サブネットを正しくブ

ラウズできる。

■ドメイン／ワークグループ

Windowsネットワークではドメインとワークグループという2種類の論理的なグループが利用できる。このうちワークグループは、ネットワーク上の共有リソースをグループ分けする程度の意味合いしかなく、各コンピュータの関係もほぼ対等だ。Linux＋Sambaのファイルサーバをワークグループに参加させるのは難しくない。

一方、ドメインについては状況が異なる。ドメインコントローラというユーザーやドメインの管理を担当する特殊なサーバが存在するからだ。とはいっても、Linux＋Sambaサーバをドメインコントローラではなく単なるドメインのメンバーにするのは可能である（多少設定に手間がかかるが）。同様にドメインに参加しているWindows NT Serverと比べると、NT標準の管理ツール（サーバマネージャ）からの操作で一部エラーが生じるなど細かい差違はあるものの、ファイルサーバとしてはほぼ同等に扱える。ユーザー認証も、NT Server同様ドメインユーザーアカウントを利用できる。

逆にLinux＋Sambaサーバをドメインコントローラとする場合、Windows NT Serverベースのドメインコントローラと比べて、その機能はかなり限定されるのが現状だ。Windows 95／98のドメインログオンはサポートするものの、1台目のドメインコントローラ（PDC：プライマリドメインコントローラ）にしかなれず、またバックアップ用のドメインコントローラ（BDC：バックアップドメインコントローラ）を別途用意することもできない。Windows NTのドメインログオンもまだ実験段階のようだ。

ファイル共有サービスの違い

Linux＋ SambaとWindows NTでは、ベースのOSのアーキテクチャが異なるので、挙動が若干異なる部分もある。以下では、そうした違いに注目して探ってみる。

■ファイル名の扱い

Windows 95／98とWindows NT、そしてLinuxでは、主に使われるファイルシステムが異なるせいもあって、ファイル名の取り扱い方がそれぞれ異なる。しかし、Windows 95／98とWindows NTは、同じWindowsファミリーということもあってか、ファイル名の取り扱い方はほぼ同じである。そのため、Windows NTのファイルサーバとWindows 95／98クライアントとの間で、ファイル名に関する問題はほとんどない。

一方、LinuxとWindowsを比べると、無視しきれない違いがある。それは、Linuxではファイルを識別する際にファイル名の大文字／小文字を区別するのに対して、Windowsでは区別しないことだ。つまり「test.dat」「TEST.dat」「Test.dat」という3つのファイルを、Linuxは異なる3つのファイルとして、またWindowsは同一のファイルとして扱うわけだ。

Linux＋Sambaのファイルサーバと Windowsクライアントの間でも、同じ問題が生じる。**画面「ファイル名に関するトラブルの例」**は、LinuxからSambaの共有にファイル名の大文字／小文字が異なる3つのファイルを書き込み、Sambaを通じてWindows 98クライアントからエクスプローラで参照しているところだ。一見、3つのファイルとして識別されているように見えるが、ファイルコピーの操作をしようとすると、意図したファイルにアクセスできないことに気づく。Windows側では3つのファイルのうち2つは実体にアクセスできない。ディレクトリ中に見えるだけである。

この問題はLinuxとWindowsの仕様の違いが原因なので、いかんともしがたい。ただし、Sambaはファイル名の取り扱いを柔軟に変更できるので、Windowsからアクセスされたファイルでも、大文字／小文字を識別するように設定を変更できる。もちろん、Windowsのローカルなファイルと扱いが変わるので注意が必要だが。

■ファイル属性

ここでいうファイル属性とは、Windowsでファイルを作成すると割り当てられる4種類の属性だ。すなわちシステム（S）／不可視（H）／読み出しのみ（R）／アーカイブ（A）である（ディレクトリ属性は省いている）。もともとはFATファイルシステムでのファイル属性である。

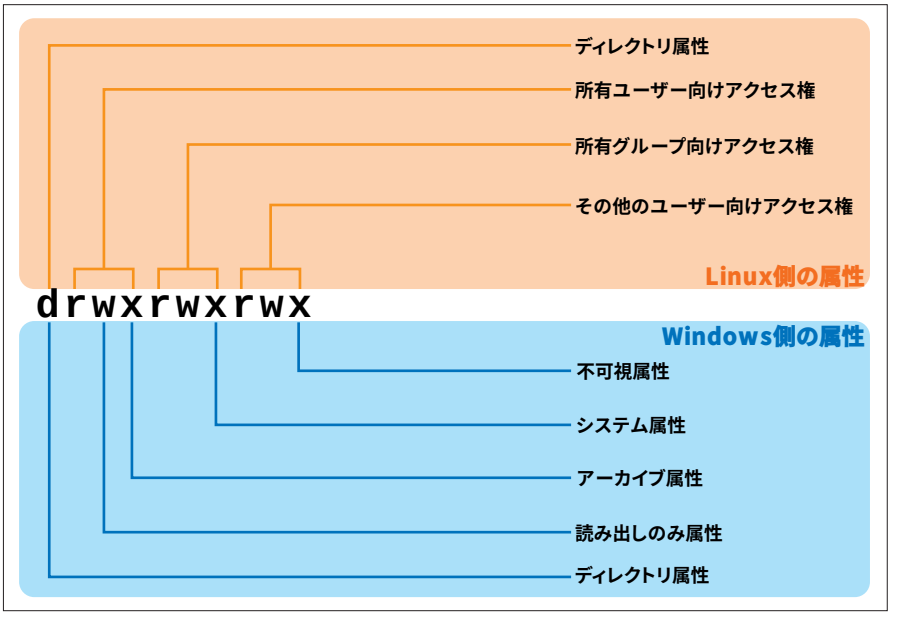
Windows NTのファイル共有サー

画面 「ファイル名に関するトラブルの例」

ビスでは、これら4種類のファイル属性はそのまま保存される。しかしLinux＋Sambaのファイル共有サービスでは、使用するファイルシステムが異なるため、Sambaの設定次第ではファイル属性が失われてしまう場合がある。

Linuxのファイル属性は、所有ユーザー／所有グループ／その他のユーザーという3種類の区分に対して、それぞれ読み出し／書き込み／実行可能という3種類の属性がある（そのほかディレクトリ属性などもあるが、ここでは触れない）。Windowsのシステム／不可視／アーカイブに対応する各属性は、Linuxにはない。一方、Windows側には実行可能属性がない（NTFSのアクセス権には、類似の属性が存在するが）。そこでSambaは、Windowsのシステム／不可視／アーカイブの各属性を、Linuxの実行可能属性に割り当てることができるようになっている（**図「SambaによるLinuxとWindows間のファイル属性のマッピング」**）。なお、読み出しのみ属性はもちろん所有ユーザーの読み出し／書き込み属性に反映される。

このようにして属性を保存すること自体はできるが、Sambaで公開しているファイルやフォルダをLinux側から参照した場合、本来の意味とは異なる形で属性がつけられているため、混乱を招きかねない。LinuxとWindowsの双方から共有ボリュームを参照する場合には注意が必要だろう。



図「SambaによるLinuxとWindows間のファイル属性のマッピング」
不可視属性とシステム属性はデフォルトではマッピングされない。

■アクセスの制御（アクセス権）

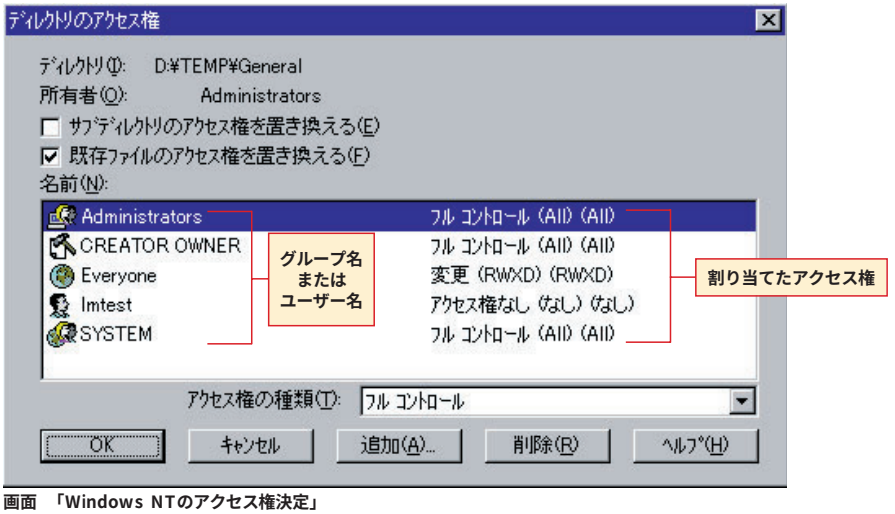
ファイル共有サービスの重要な機能の1つとして、格納されるファイルやディレクトリの保護、すなわちセキュリティが挙げられる。セキュリティを維持するためには、状況に応じてファイルやディレクトリへのアクセスを制限する必要がある。ここでいうアクセス権とは、ファイルなどへアクセスするための権利を指す。どのようなアクセス権を設定できるかが、ファイル共有サービスのポイントの1つといえる。

アクセス権を設定する対象には、ファイルやディレクトリといったアクセス対象そのものと、公開している共有そのものがある。Windowsクライアントからは、まず「共有」という入口で、次にファイルシステムのレベルでそれぞれアクセスが制限されるという二段構えに見える。Linux＋Sambaでは、共有でのアクセス制限はSambaが、ファイルシステムはLinuxが担当する。

共有でのアクセス制限は、ユーザー／グループによる制限とそれ以外に分かれる。前者による制限はSambaもWindows NTもほぼ同等の設定が可能で、ユーザー／グループごとに読み出し専用あるいは読み書き可能といったアクセスの制限ができる。ユーザー／グループ以外の制限方法はWindows NTにはなく、Sambaにはクライアント PC（ホスト）ごとに制限するといった機能がある（詳細は88ページの**コラム「Linux＋Samba独特の機能」**を参照）。

ファイルシステムでのアクセス制限は、Windows NTのほうがきめ細かく設定できる（ただしファイルシステムにNTFSを使う場合に限られる）。Windows NTでは、ファイルやディレクトリに対して任意のユーザー／グループでアクセス権を設定できる（88ページの**画面「Windows NTのアクセス権設定」**）。それに対してLinuxでは、対象のファイルまたはディレクトリの所有ユーザー／所有グループ／それ以外のユーザーという3種類の項目だけにしか、アクセス権を設定できない。しかし、前述のようにSambaでは共有レベルで細かいアクセス権の設定が可能であり、それと組み合わせれば、実用上はWindows NTと遜色のないアクセス制御が実現できるだろう。

ところでWindows NTにないLinux＋Sambaの機能の1つに、ユーザー認証を必要とせず共有そのものに



画面 「Windows NTのアクセス権決定」

設定されたパスワードで認証するファイル共有モードがある。これはWindows 95／98が持つファイル共有機能と同等で、共有自体に設定されている読み出し専用およびフルアクセス可能な2種類のパスワードでアクセスを制限する。ユーザーがその共有に接続しようとした場合、どちらかのパスワードを入力すればそれを利用できる。ユーザー認証の設定が不要なため、小規模で外部に接続されていないネットワークでは手軽に扱えて便利だろう。

ただし、これもWindows 95／98同様だが、1つのLinux＋Sambaサーバでは、ユーザー認証を必要とするファイル共有モードとは共存できず、二者択一となる。

■パスワード管理

ユーザー認証について詳細は76ページからを参照していただくとして、基本的にはLinux＋Sambaのほうがパスワードの管理により注意を払う必要がある。LinuxとSambaそれぞれでユーザー認証を行うため、二重にパスワードを管理しなければならない（SambaからLinuxへのパスワード同期機能はある）。場合によってはWindows NTのパスワードも絡んでくるので厄介だ。Windows NTのファイルサーバなら、ドメインまたはローカルのユーザー情報を管理するだけで済む。

Column

Linux＋Samba独特の機能

Linux＋Sambaは、Windows NTにはない機能も備えている。そうした機能が必要とされた理由はさまざまだが、何にせよLinux＋Sambaの利点であることには間違いない。以下ではそうした機能に焦点を当ててみた。

■ホストによるアクセスの制限

Sambaで公開している共有ボリュームは、特定のクライアントマシン（ホスト）だけにアクセスを許可または禁止できる。たとえば、ある1台のホストだけはアクセスを禁止したり、サブネット内部からのアクセスだけを許可したり、あるDNSドメインに存在するマシンだけに許可／禁止をしたり、といったことが可能である。

Windows NTで同様の機能を実現しようとすると、TCP/IPレベルで設定するしかなく、ファイル共有以外のサービスにまでアクセス許可／禁止の影響が出てしまう。

■共有名やファイルなどの表示／非表示

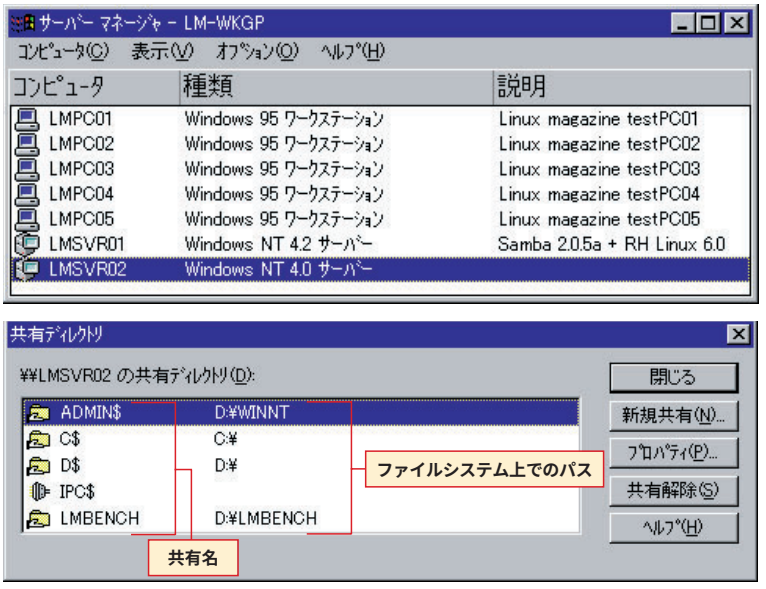
通常、エクスプローラなどでファイルサーバをブラウズすると、利用可能な共有名が表示されるが、Sambaではこの共有を表示するか否かを個別に設定できる。Windows NTで共有名を隠すには、共有名の末尾に「$」をつけるしかない。

表示されない共有でも、アクセス権さえあれば接続できる。セキュリティや使い勝手の点から、ブラウズされたくない共有が必要とされる場合もある。Sambaなら共有名を変えずに対処できる。

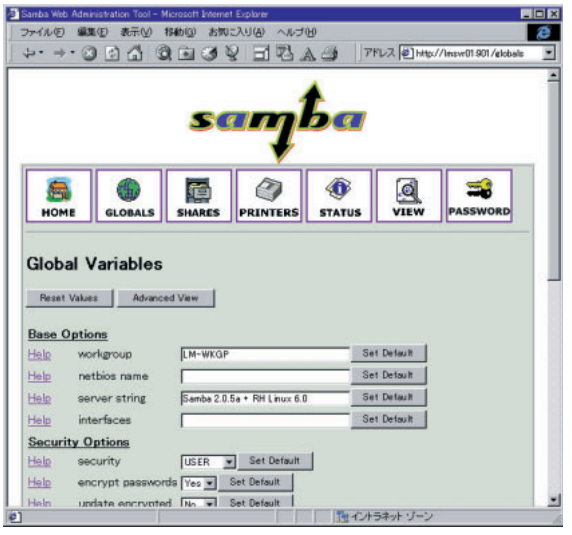
Sambaは共有名のほかファイルやディレクトリも表示しない、つまりクライアントから参照できないように設定できる。こうした機能が用意されているのは、Linuxのような UNIX系OSではファイルシステムからデバイスを直接参照できる（/dev以下のファイルなど）ため、Sambaで公開してクライアントからアクセスされるのは危険なためだ。また、Windowsクライアントには余計なLinuxの設定ファイル（ファイル名の1文字目が「.」のファイルなど）を隠蔽する役割もある。どちらかといえば、LinuxとWindowsの違いによる副作用を防ぐのが目的といえる。

■実効ユーザーの変更

WindowsクライアントからWindows NTのファイルサーバにアクセスする場合、必ずユーザー認証されたユーザーでアクセスが行われる。一方、Sambaの場合は、実際にファイルなどをアクセスする実効ユーザーを変えることができる。あるユーザーAで共有にアクセスしても、ファイルサーバ内部ではユーザーBとしてアクセスする、という具合にユーザーをマッピングできる。この機能により、ファイル所有者を統一してユーザー間のファイル相互利用を容易にしたり、あるいはサーバとクライアントの間でユーザー名が食い違っているままの環境でファイルサーバへのアクセスを実現したりできる。



画面 「Windows NTのサーバマネージャ」 サーバマネージャ（上）とその共有設定メニュー（下）



画面 「SWAT」
SWATでSambaの設定を参照しているところ。リモートコンピュータに必要なソフトウェアはWebブラウザだけで済む。

■リモート管理機能

ここでいうリモート管理機能とは、ファイルサーバ以外のマシンからネットワーク経由でファイルサーバを管理（各種設定を変更・確認したり、ステータスを調べたり）する機能を指す。Windows NTをファイルサーバとした場合、**画面「Windows NTのサーバマネージャ」**のサーバマネージャでリモート管理が可能で、共有の新規作成・削除・設定変更や、接続ユーザーの確認などの機能を持つ。NTのサービスを開始・停止することもできる。

サーバマネージャをはじめとするWindows NT Serverの管理ツールは、Windows 95／98／NT Workstation用も用意されており、Windows NT Server CD-ROMからインストールできる。しかしWindows以外のプラットフォームでは利用できない。NT Serverリソースキット付属のWeb Administration kitを Windows NT Serverにインストールすれば、Webブラウザからでも管理機能を利用できるが、それにはIISのインストールが必要になるなど、やや敷居も高い。

一方、SambaではSWAT（Samba Web Administration Tool）と呼ばれるWebブラウザ経由でリモート管理できるツールが用意されている（**画面「SWAT」**）。これにより共有の新規作成・削除・設定変更からSambaのほとんどの設定まで、Web経由で簡単に操作できる。もし、SWATで設定を変更できない場合でも、telnetなどでリモートマシンからSambaの設定ファイル（smb.conf）を操作するという手もある。リモート管理の可能な範囲はLinux＋Sambaの方がWindows NTより明らかに広い（Windows NTでもシマンテックpcAnywhereなどのソフトウェアで、コンソールをリモートマシンにリダイレクトして管理するという手もあるが、一般的とは言い難い）。

■ユーザー使用容量の制限（クォータ）

ファイルサーバにユーザーがファイルを詰め込み過ぎて、ディスク容量が足りなくなるというのはよく聞く話である。これを防ぐには、1ユーザーの使用できる最大容量を制限するクォータ(Quota）という機能を利用するのが一般的だ。

Linuxの場合、カーネルバージョンが少なくとも1.3.8x以降なら、クォータ機能が内蔵されている。SambaもLinuxのクォータ機能には対応しているので、組み合わせて使うことができる。一方、Windows NTには標準でクォータ機能はなく、別途クォータソフトウェアを購入するなどしてカバーしなければならない。

■ファイルのリアルタイム圧縮と暗号化

ファイルサーバのハードディスクを効率よく使うには、ファイルを圧縮してから格納するという手法がある。Windows NTはファイルまたはディレクトリ単位で、リアルタイムに圧縮・解凍する機能をファイルシステム（NTFS）のレベルでサポートしている。ファイル共有サービスでも、この圧縮機能は利用できる。Linuxにはまだ標準的な圧縮機能はないものの、開発は進められている。

もう1つの暗号化は、セキュリティを高めるためにファイルを格納する際

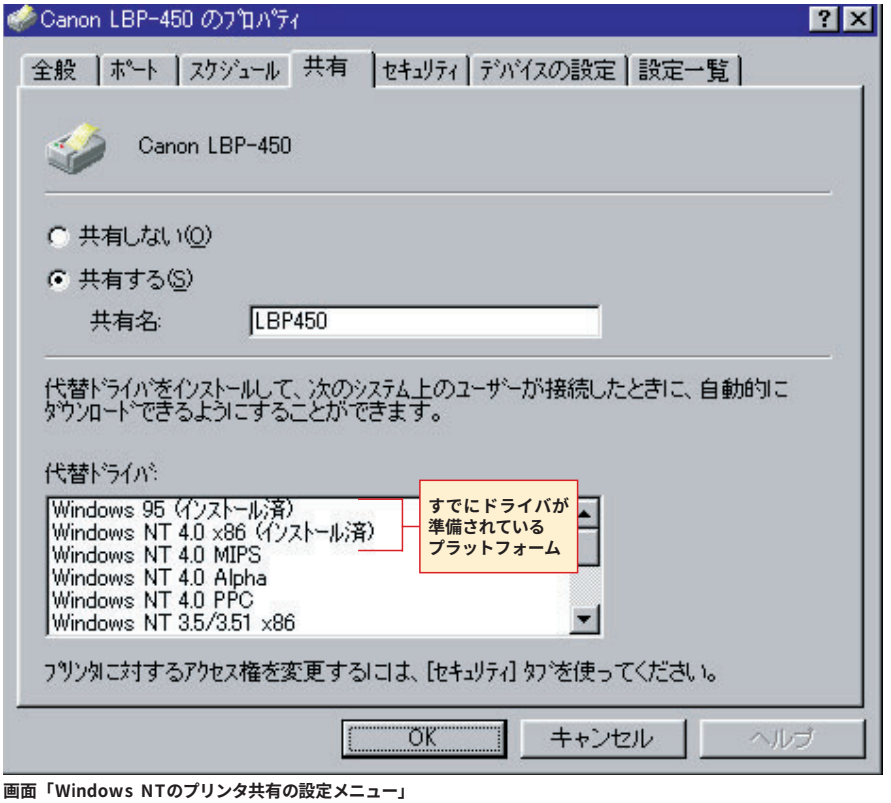
に暗号化を施し、逆にファイルを読み出す際には復号化するという機能だ。現時点ではLinuxもWindows NTも対応していないが、Windows NTの次期バージョンには組み込まれるようだ。

プリンタ共有

Windows NTやLinux＋Sambaのサーバは、ファイル共有サービスと同様に、複数のWindowsクライアントで1台のプリンタを共有するサービス（プリンタ共有）もサポートしている。プリンタ共有のサーバは、Windowsクライアントに対してプリントジョブを一時的に溜め込むスプールを提供し、そこからサーバ直結か、あるいはネットワーク接続のプリンタへジョブを順次送出する、という役割を果たす。共有可能なプリンタは、Windows NTなら自分自身で利用可能なプリンタ（ドライバがなくてもよい）が、またLinuxならlprなどで印刷できるプリンタである。ファイル共有より比較的機能は単純だが、やはりWindows NTとLinux＋Sambaでは細かい点で異なるところがある。以下、そうした差違をチェックしてみよう。

■サーバでもレンダリングできるWindows NT

Windowsクライアントから共有プリンタへ印刷する場合、一般的にはWindowsクライアント内でレンダリング、すなわち出力先のプリンタが直接印字できるプリントデータへの変換（描画）が行われる。そのため、サーバはデータの中身には触れず、単にクライアントからプリンタへデータの橋渡しをするにとどまる。サーバが何であっても印刷結果には影響しないわけだ。



画面「Windows NTのプリンタ共有の設定メニュー」
「代替ドライバ」を事前に組み込んでおくと、クライアントPCが共有に接続したとき自動的にドライバをサーバからダウンロードできる。Windows 98用ドライバはWindows 95と共通に扱われる。

しかし、クライアントとサーバがどちらもWindows NT 4.0の場合には、EMF（Enhanced Meta File）という中間ファイルを用いて、サーバにレンダリングさせることができる（もちろんサーバ側にも正しいプリンタドライバを組み込む必要がある）。その際、クライアントからサーバへ送られるデータ容量が小さくなり、高速化されるというメリットがある。

この機能はLinux＋Sambaにはない。とはいえ、Windows 95／98クライアントまで含めた場合、EMFを使うと印刷結果が乱れるなどの問題が生じやすいこともあり、あまり役立っていないのが現状のようだ。

■プリンタドライバの自動配布

プリンタ共有では、Windowsクライアントでレンダリングを行うため、プリンタドライバもWindowsクライアントに組み込まなければならない。そこで、Windows NTのプリンタ共有サービスでは、ドライバ組み込みの手間をなくすために、Windowsクライアントへ自動的にドライバをダウンロードさせる機能を備えている（**画面「Windows NTのプリンタ共有の設定メニュー」**）。Windows NT 4.0はもちろん、Windows 95／98に対してもドライバを配布できる（ただしサーバにService Pack 4以降が当たっていないと、ドライバの仕様によって自動配布ができない場合がある）。

一方、Linux＋Sambaでも、Windows 95／98に対しては、適切な設定を行えばドライバの自動配布を実現できる。が、現状ではWindows NTに対してはできないようだ。

ファイル共有サービスの性能比較

編集部

技術面でOSを比較する場合、その機能と性能が重要なファクタであることは間違いない。ここでは、Linux＋SambaとWindows NTそれぞれのファイル共有サービスのパフォーマンスを調べるために、簡単なベンチマークテストを行ってみた。

テスト環境について

このテストで用意したテスト環境は、SOHOなど小規模なオフィスで用いる単一セグメントのネットワークを想定している。すなわち1台のサーバマシンと数台のクライアントPC、そしてそれらを相互接続するEthernetのネットワークである。以下、用意した機材や設定などを記す。

■サーバマシンとサーバOS

サーバには、コンパックコンピュータ製PCサーバのProLiant 1600を用いた。詳細は93ページの**コラム「テストに使用したPCサーバ」**を参照していただきたい。

コラム中の**表1**で、ハードディスクを4台も装備しているのは、各ドライブにWindows NTのシステムとデータ、Linuxのシステムとデータをそれぞれ別個に格納するためだ。これにより、パーティションのサイズや配置などといった条件を、両OS間でなるべく同じにすることができた。

このPCサーバにインストールしたLinuxとWindows NTのバージョンは、**表1**のとおりである。なお、Windows NTの欄にあるOption Packとは、WebサーバなどWindows NTにとってオプショナルな機能を集めたソフトウェア集で、マイクロソフトより無償公開されている。今回はFTPサーバを利用するためにOption Packをインストールしている。

テスト環境のWindowsネットワークはワークグループである。Linux＋Sambaサーバは、まだ完全なドメインコントローラとしての機能を持っておらず、ドメイン環境でテストする段階ではないと判断した。Linux＋SambaサーバもWindows NT Serverも、ドメインコントローラではなくスタンドアロンサーバとして稼働させている。

またLinux＋Sambaサーバのユーザー認証はUSERモード、つまりSamba側で暗号化パスワードを認証するモードを選んでいる。これがスタンドアロンサーバであるWindows NT Serverのユーザー認証にもっとも近いからだ。

Sambaの設定は、2.0.5aのデフォルトからチューニング関連の設定を変更している。具体的には、Socketのオプションに「TCP_NODELAY」を指定し、さらに先読み（read prediction）の有効、ログレベルの低下、といった対処を行った。そのほか、Sambaの実行ファイルsmbdとnmbdはデーモンとして起動している。

■クライアントPC

Windows 98をインストールしたクライアントPCを5台用意した。5台のハードウェア仕様はまったく同じではなく、プロセッサの種類もMMX Pentium-166MHzから Pentium II-400MHzと幅広い。ただし、メモリ容量は64Mバイト以上でネットワークは100BASE-TX対応という2点は、どのPCも満たしている。

クライアントPCのネットワーク設定は、**表2**に準じている。なお、バックアップブラウザをWindows 98マシンで実行するため、マイクロソフト共有サービスも組み込んでいる。

■ネットワーク接続

サーバマシンと5台のクライアントPCを100BASE-TXリピータハブで接続している。10BASE-Tではなく100BASE-TXを選んだのは、ネットワークがボトルネックになってOSの性能差が目立たなくなるのを避けたいからだ。また現実の作業環境と異なるのは、インターネットなど外部ネットワークとは接続せず、独立したネットワークでテストしていることである。これは、外部からの予期せぬアクセスでベンチマークテストの測定結果が乱れるのを防ぐためだ。

そのほかのネットワークの設定は表2のとおりである。IPアドレスを決め打ちにし、名前解決をlmhostsファイルという静的な解決法にしたのは、ダイナミックにDHCPサーバやWINSサーバ（nmbd）が稼働して、そのときの測定値に影響するのを防ぎたかったからである。一般的な環境なら、DHCPサーバやWINSサーバを運用するだろう。

下位プロトコル	TCP/IPのみ
IPアドレス	全マシンで決め打ち（DHCP非使用）
Windosの名前解決	lmhostsファイルを全マシンに配布
マスタブラウザ	PCサーバ
バックアップブラウザ	クライアントPCのうちの1台
ユーザー認証	PCサーバ上のユーザー情報を使用
Windowsネットワーク	ワークグループ

表2　テスト時の各種ネットワークの設定

クライアントPC×1台のテスト

最初に測定したのは、1台のクライアントPCとサーバの間で、XCOPYコマンドでファイルを転送したときの速度である。比較のために、ほぼ同じ条件でFTPの転送レートも測定してみた。SMBに比べFTPはオーバーヘッドが非常に小さいため、双方の速度を比べれば、性能のボトルネックの手がかりを得やすい。

このテストでは、サーバとクライアントPCとの間で160Mバイトの単一ファイルを、Windows 98標準のXCOPYおよびFTPクライアントで転送したときの速度を測定している。またクライアントPCのハードディスクが性能のボトルネックにならないよう、ディスクキャッシュを利用してディスクアクセスを抑えている。逆にサーバ側ではディスクアクセスが生じるため、結果にはサーバのディスクとネットワークの両方の性能が含まれる。

■クライアント1台のテスト結果

測定結果はグラフ1のとおりである。LinuxはFTPが速く、Windows NTはXCOPY（つまりSMBプロトコルによる転送）で速いという傾向が表れている。とはいっても、SMBでの性能差はそれほど大きくはない。むしろWindows NTのFTPによる書き込みが大きく落ち込んでいるのが気になる。

なお、FTPクライアントのTCPウィンドウサイズをデフォルト（4Kバイト）から増やしたところ、16Kバイト以上ではさらに遅くなる傾向が見られた。

Linuxについては、FTPでの書き込みで100BASE-TXの限界に近い性能を発揮している。プリミティブなネットワークの性能に問題はないようだ。

クライアントPC×5台のテスト

グラフ1の結果を踏まえて次に行ったのが、5台のクライアントPCとサーバの間で同時にファイルコピーを実行するというテストである。テスト方法は、サイズが10Kバイトのファイル3000個と1Mバイトのファイル30個をクライアントPCごとにサーバに用意し、5台同時にXCOPYコマンドでファイルを転送して速度を測定するというものだ。これも読み出し／書き込みの両方を測定した。アクセス中、5台のクライアントは別個のファイルをアクセスするので、同一ファイルへのアクセスの競合は生じない。

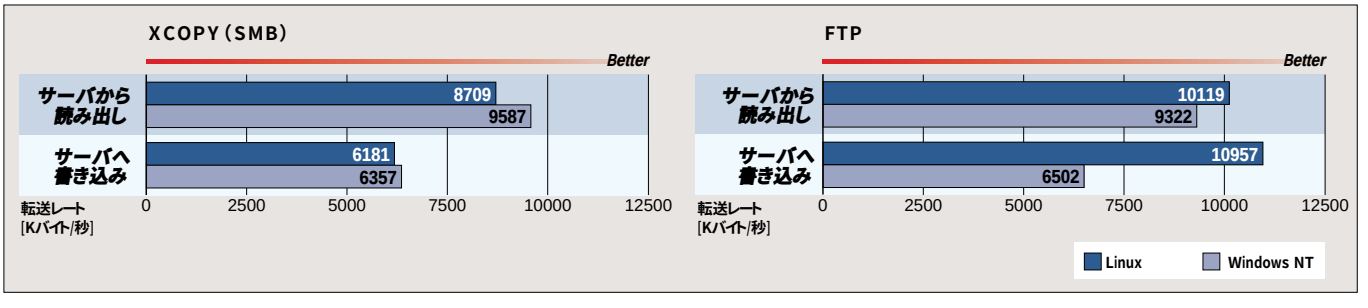
1台のクライアントPCでのテストでは、サーバ側でファイルはシーケンシャルにアクセスされた。しかしこの5台のテストでは、同時に5個以上のファイルがアクセスされるため、ハードディスクではランダムアクセスに近い激しいヘッドシークが生じる。また同時に5台が送受信するので、ネットワークにも大きな負荷がかかり、Ethernetのコリジョンも頻繁に生じる。しかしプロセッサへの負荷は大したことはない。このベンチマークテストはサーバのディスクサブシステムとネットワークの性能が表れやすいだろう。

■クライアント5台のテスト結果

サーバ側での平均転送レートをグラフ2に表した。これによると、ファイルサイズが1Mバイトの場合はWindows NTが若干速いという、グラフ1のSMBの結果と同じ傾向が見られる（その差がそれほど大きくない点も同様だ）。一方、ファイルサイズが10Kバイトと小さい場合は、Linuxの落ち込みが目立つ。Sambaの設定を色々変更してみたが、変化はなかった。原因としては、ファイルのオープン／クローズに手間取っていることなどが推測されるが、確証は得られなかった。

今回のテストでは、5台のクライアントPCでしかテストできなかったが、これまでインターネットなどで公表されてきたベンチマークテストでは、さらにクライアントPCの台数が多いとLinux＋Sambaが優位という結果もあった。機会があれば、より規模の大きなテストをしてみたい。

グラフ1　クライアントPC×1台とのファイル転送



Column

テストに使用したPCサーバ

LinuxとWindows NTとの間で、サーバでのパフォーマンスを比較するうえで必要だったのは、どちらのOSでも動作するサーバマシンだった。今回、ターゲットとしているのはSOHOやワークグループ（大きくても部門クラス）という、比較的小規模なネットワークなので、サーバマシンもエントリーまたはワークグループクラスのPCサーバが対象となる。

こうした条件のもと、ベンチマークテストのために用意したのが、コンパックコンピュータ製のPCサーバであるProLiant 1600だ（写真1）。ProLiantシリーズのサポートOSはWindows NT、NetWare、UnixWareだけで、Linuxは含まれていない。しかし、Linuxでの動作確認は進められており、確認された機種はLinux-readyモデルとしてWebサイトに公開されている（http://www.compaq.co.jp/products/linux/linux-ready.html）。ProLiant 1600については、Pentium III-450MHz／500MHz／550MHz搭載モデルがLinux-readyモデルである（1999年8月下旬時点）。

写真1　ProLiant 1600

ProLiantシリーズの中でワークグループサーバとして位置づけられているProLiant 1600は、最大2基のプロセッサ、最大1Gバイトのメモリ、6本のPCIスロット、そして6スロットのホットプラグ対応ハードディスクベイなど高い拡張性を持つ。またオプションのRAIDコントローラであるSmartアレイ221／3200には、開発段階ではあるがLinux用ドライバが存在するので、そのうち耐障害性の高いLinuxファイルサーバを構築できそうだ。そのほか、Linux-readyモデルだけあって、標準装備のSCSIコントローラやEthernetコントローラには、Linuxで動作するドライバが存在する。今回はRed Hat Linux 6.0だけでドライバを別途追加することなく、Linuxを稼働させることができた。

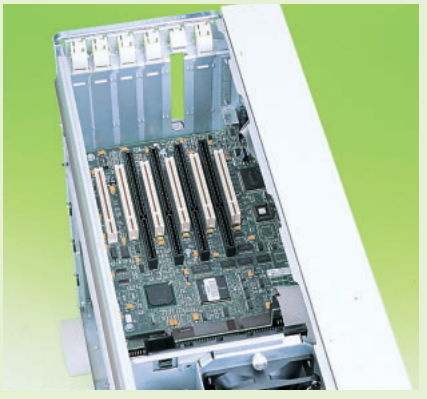


写真2　ProLiant 1600の拡張スロット部分
PCI-PCIブリッジを用いて6本のPCIスロットを実装する。この部分はケースからそっくり抜き出すことができる。メンテナンス性は高い。

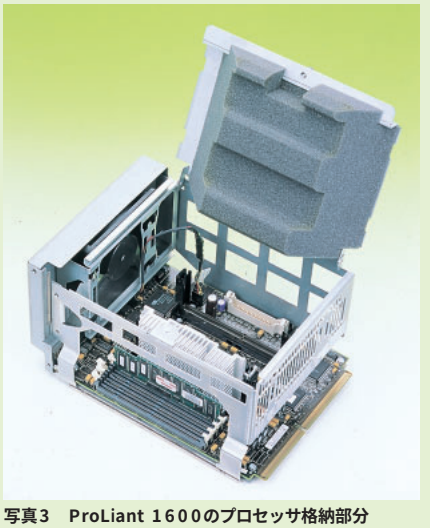
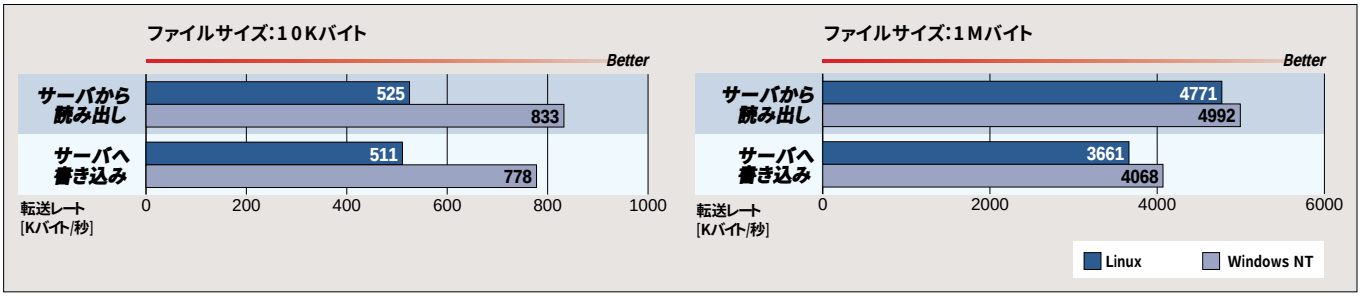


写真3　ProLiant 1600のプロセッサ格納部分
プロセッサのほか、メモリモジュールやグラフィックス／Ethernet／SCSIなどが集約されている。このブロックもケースから抜き出し可能だ。

ハードウェア	
製品名	コンパックコンピュータ ProLiant 1600 6/550
プロセッサ	Intel Pentium III-550MHz×1基
チップセット	Intel 440BX AGPset
2次キャッシュメモリ	512Kバイト（プロセッサ内蔵）
メインメモリ	SDRAM 128Mバイト（100MHz）
ハードディスク	9.1Gバイト Wide Ultra2 SCSIハードディスク×4台
SCSIホストアダプタ	Wide Ultra SCSI-3×2チャネル（Symbios 53C876）
Ethernet	10BASE-T／100BASE-TX（Netelligent 10/100TX PCI UTPコントローラ）
ソフトウェア	
Windows NT	Windows NT Server 4.0、Service Pack 4、Option Pack（IIS 4.0）、コンパック提供のデバイスドライバ
Linux	Red Hat Linux 6.0、Red Hat提供のパッチ、Samba 2.0.5a

表1　テスト時のサーバマシンの構成

グラフ2　クライアントPC×5台とのファイル転送



Webサービス

編集部

Web（http）サービスは、インターネット時代の今、最もホットな分野のひとつである。

Webサービスを提供するサーバソフトウェアとしては、Apache、Netscape、Zeus、IIS、Sun Web Serverが5大メジャーとして知られている。もちろん、これら以外にも非常に多くの実装があり、それぞれに特徴を持っている。

Linuxでは、商用のZeusなども利用できるが、やはりApacheという選択がほとんどを占めるであろう。多くのディストリビューションでも、標準としてインストールされ、利用できるようになっている。

Apacheは、パフォーマンスもさることながら、mod_perlをはじめするさまざまな機能モジュールをダイナミックに追加できる拡張性、そしてほとんどのOS（UNIX系だけでなくWin32用もある）にポーティングされていることもあり、インターネット上での稼動数は一番多い。このことは、実績のともなったWebサーバとしての地位を揺るぎないものにしている。

このフリーソフトウェアのApacheの存在ゆえ、それ以下のレベルの商用Webサーバというのは、事実上存在する余地がない。そういった意味で、上記の商用Webサーバでは、使いやすさの向上や、特にコマース関連の機能を充実させて、差別化を図っている。

一方Windows NTでは、ApacheやNetscapeなども利用できるが、ほぼOSの一部（Option Packということになっている）であるIISを利用することが多いだろう。

オペレーティングシステムにWebサービスのための特別な機能は必要ないが、そのネットワーク性能は大きく影響する。また、信頼性（安定性）も重要なポイントとなる。

つい先日もNT＋IISとLinux＋ApacheのWebベンチマークが物議をかもしていたが、条件次第で結果は変わるもので、Linux＋Apacheがよい結果を出すときもあるし、NT＋IISがよい結果を出すときもある。ただ結果からは、改善できる（すべき）点が見えてくる。

Linuxカーネルに関していえば、SMPへの対応はまだ不十分で、TCP/IPプロトコルスタックも複数コネクションの扱いの部分などで改善が必要であるということだ（これらのことは、開発に関わっている人は当然わかっているのだろうが、実際に実装するのは本当に難しい）。

NT＋IISにも同様のことがいえる。NTのネットワーク部分はSP4で大きく改善されたが、完全なスレッドセーフになっているわけではないし、ほかの商用OSと比べるとかなり見劣りする部分も多い。

▼リスト1　Apacheのシステムコールの流れ

```
fcntl(18, F_SETLKW, {type=F_WRLCK, whence=SEEK_SET, start=0, len=0}) = 0
accept(16, {sin_family=AF_INET, sin_port=htons(3896), sin_addr=inet_addr("192.168.1.3")},
[16]) = 3
fcntl(18, F_SETLKW, {type=F_UNLCK, whence=SEEK_SET, start=0, len=0}) = 0
rt_sigaction(SIGUSR1, {SIG_IGN}, {0x401ba45c, [], SA_INTERRUPT|0x4000000}, 8) = 0
getsockname(3, {sin_family=AF_INET, sin_port=htons(8080),
sin_addr=inet_addr("192.168.1.11")}, [16]) = 0
setsockopt(3, IPPROTO_TCP1, [1], 4)     = 0
read(3, "GET / HTTP/1.1\r\nAccept: applicat"..., 4096) = 335
rt_sigaction(SIGUSR1, {SIG_IGN}, {SIG_IGN}, 8) = 0
time(NULL)                              = 935595578
stat("/usr/htdocs", {st_mode=S_IFDIR|0755, st_size=2048, ...}) = 0
rt_sigaction(SIGALRM, NULL, {0x401ba45c, [], SA_INTERRUPT|0x4000000}, 8) = 0
stat("/usr/htdocs/index.html", {st_mode=S_IFREG|0644, st_size=1622, ...}) = 0
stat("/usr/htdocs/index.html", {st_mode=S_IFREG|0644, st_size=1622, ...}) = 0
open("/usr/htdocs/index.html", O_RDONLY) = 4
mmap(NULL, 1622, PROT_READ, MAP_PRIVATE, 4, 0) = 0x40015000
close(4)                                = 0
select(4, [3], NULL, NULL, {0, 0})      = 0 (Timeout)
write(3, "HTTP/1.1 200 OK\r\nDate: Wed, 25 A"..., 1921) = 1921
time(NULL)                              = 935595578
write(17, "192.168.1.3 - - [26/Aug/1999:00:"..., 71) = 71
munmap(0x40015000, 1622)                = 0
rt_sigaction(SIGUSR1, {0x401ba45c, [], SA_INTERRUPT|0x4000000}, {SIG_IGN}, 8) = 0
read(3, "GET /apache_pb.gif HTTP/1.1\r\nAcc"..., 4096) = 249
rt_sigaction(SIGUSR1, {SIG_IGN}, {0x401ba45c, [], SA_INTERRUPT|0x4000000}, 8) = 0
time(NULL)                              = 935595578
stat("/usr/htdocs/apache_pb.gif", {st_mode=S_IFREG|0644, st_size=2326, ...}) = 0
open("/usr/htdocs/apache_pb.gif", O_RDONLY) = 4
mmap(NULL, 2326, PROT_READ, MAP_PRIVATE, 4, 0) = 0x40015000
close(4)                                = 0
select(4, [3], NULL, NULL, {0, 0})      = 0 (Timeout)
write(3, "HTTP/1.1 200 OK\r\nDate: Wed, 25 A"..., 2624) = 2624
time(NULL)                              = 935595578
write(17, "192.168.1.3 - - [26/Aug/1999:00:"..., 84) = 84
munmap(0x40015000, 2326)                = 0
rt_sigaction(SIGUSR1, {0x401ba45c, [], SA_INTERRUPT|0x4000000}, {SIG_IGN}, 8) = 0
read(3,  <unfinished ...>
```

今回ベンチマークは実施しなかったが、少し考察してみたのでそれを紹介しておこう。基本的にはWebサーバソフトウェア側の考察だが、間接的にオペレーティングシステムが見えてくると思う。

Webサーバの考察

Webサーバの基本的な仕事は、ネットワーク先のクライアントからリクエストを受けて、そのリクエストを解析し、必要なデータをそのクライアントに送信することである。データはスクリプトなどでダイナミックに生成される場合もあるが、主はスタティックなファイルである。

Linux用にコンパイルされたApache（Ver.1.3.9）の基本コンセプトは、複数のコネクションを処理するのに複数の子プロセスを生成して、それをシグナルでコントロールしながら使いまわすことだ。実際いくつの子プロセスで対応するかは、httpd.confのMinSpareServersとMaxSpareServersで設定できる（デフォルトは、それぞれ5と10）。またこれとは別に最初の初期化時にスタートさせるプロセスも設定できる（デフォルトは5。バージョンにより異なる）。このように複数のコネクションを複数の子プロセスで対応するのは、UNIX系OSではごくごく一般的なことだ。

では、実際にApacheの子プロセスがクライアントからのリクエストを処理する際のシステムコールの流れを見てみよう（**リスト1**）。

accept()からsetsockopt()までは、コネクションの処理で、その後のread()でそのコネクションからのリクエストデータを取得している。リクエスト（http://linux01/）は、ファイル名を直接指定していなかったので、いくつかのstat()が実行されている。これはファイルの情報を得るためだ。その後クライアントに送るべきデータ（/usr/htdocs/index.html）をopen()し、mmap()を用いて、プロセスのメモリ空間にマップしている。そしてプレフィックスデータを付加したものを最初のwrite()でソケットに書き込んでいる。その次のwrite()はログファイルへの書き込みだ。続いてクライアントは、取得したindex.htmlを解析して、apache_pb.gifを要求している。こちらもstat()、open()、mmap()、write()とindex.htmlと同様に処理している。なおリスト1では、write()しか使われていないが、書き込むデータが4096バイト以上だとwritev()が使用される。書き込むデータには、プレフィックスが含まれるので、実質3797バイト程度が

▼リスト2　Zeusのシステムコールの流れ

```
poll([{fd=7, events=POLLIN}, {fd=9, events=POLLIN}, {fd=10, events=POLLIN}], 3, 7000) = 0
poll([{fd=7, events=POLLIN}, {fd=9, events=POLLIN}, {fd=10, events=POLLIN, revents=POLLIN}],
3, 10000) = 1
accept(10, {sin_family=AF_INET, sin_port=htons(3975), sin_addr=inet_addr("192.168.1.3")},
[16]) = 4
ioctl(4, FIONBIO, [1])                  = 0
getsockname(4, {sin_family=AF_INET, sin_port=htons(8000),
sin_addr=inet_addr("192.168.1.11")}, [16]) = 0
brk(0x81ab000)                          = 0x81ab000
socket(PF_INET, SOCK_DGRAM, IPPROTO_IP) = 6
connect(6, {sin_family=AF_INET, sin_port=htons(53), sin_addr=inet_addr("192.168.1.2")}, 16) =
0
write(6, "\0\1\1\0\0\1\0\0\0\0\0\0\0013\0011\003168\003192\7in-a"..., 42) = 42
accept(10, 0xbffff78c, [16])            = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=6, events=POLLIN, revents=POLLIN}, {fd=7, events=POLLIN}, {fd=9, events=POLLIN},
{fd=10, events=POLLIN}], 4, 3000) = 1
read(6, "\0\1\205\200\0\1\0\1\0\0\0\0\0013\0011\003168\003192\7"..., 1024) = 70
close(6)                                = 0
read(4, "GET / HTTP/1.1\r\nAccept: applicat"..., 511) = 335
stat("/usr/htdocs/", {st_mode=S_IFDIR|0755, st_size=2048, ...}) = 0
stat("/usr/htdocs//index.html", {st_mode=S_IFREG|0644, st_size=1622, ...}) = 0
open("/usr/htdocs//index.html", O_RDONLY) = 6
read(6, "<!DOCTYPE HTML PUBLIC \"-//W3C//D"..., 1622) = 1622
close(6)                                = 0
writev(4, [{"HTTP/1.1 200 OK\r\nServer: Zeus/3."..., 191}, {"<!DOCTYPE HTML PUBLIC \"-
//W3C//D"..., 1622}], 2) = 1813
poll([{fd=4, events=POLLIN, revents=POLLIN}, {fd=7, events=POLLIN}, {fd=9, events=POLLIN},
{fd=10, events=POLLIN}], 4, 3000) = 1
read(4, "GET /apache_pb.gif HTTP/1.1\r\nAcc"..., 511) = 249
stat("/usr/htdocs//apache_pb.gif", {st_mode=S_IFREG|0644, st_size=2326, ...}) = 0
open("/usr/htdocs//apache_pb.gif", O_RDONLY) = 6
read(6, "GIF89a\3\1 \0\367\0\0\377\377\377\316\316\316\245\245\245"..., 2326) = 2326
close(6)                                = 0
writev(4, [{"HTTP/1.1 200 OK\r\nServer: Zeus/3."..., 191}, {"GIF89a\3\1
\0\367\0\0\377\377\377\316\316\316\245\245\245"..., 2326}], 2) = 2517
poll([{fd=4, events=POLLIN}, {fd=7, events=POLLIN}, {fd=9, events=POLLIN}, {fd=10,
events=POLLIN}], 4, 3000) = 0
poll( <unfinished ...>
poll([{fd=7, events=POLLIN}, {fd=9, events=POLLIN}, {fd=10, events=POLLIN}], 3, 10000) = 0
poll([{fd=7, events=POLLIN}, {fd=9, events=POLLIN}, {fd=10, events=POLLIN, revents=POLLIN}],
3, 10000) = 1
accept(10, {sin_family=AF_INET, sin_port=htons(3988), sin_addr=inet_addr("192.168.1.3")},
[16]) = 4
ioctl(4, FIONBIO, [1])                  = 0
getsockname(4, {sin_family=AF_INET, sin_port=htons(8000),
sin_addr=inet_addr("192.168.1.11")}, [16]) = 0
read(4, 0x81aad80, 511)                 = -1 EAGAIN (Resource temporarily unavailable)
accept(10, 0xbffff78c, [16])            = -1 EAGAIN (Resource temporarily unavailable)
poll([{fd=4, events=POLLIN, revents=POLLIN}, {fd=7, events=POLLIN}, {fd=9, events=POLLIN},
{fd=10, events=POLLIN}], 4, 9000) = 1
read(4, "GET / HTTP/1.1\r\nAccept: */*\r\nAcc"..., 511) = 205
stat("/usr/htdocs/", {st_mode=S_IFDIR|0755, st_size=2048, ...}) = 0
stat("/usr/htdocs//index.html", {st_mode=S_IFREG|0644, st_size=1622, ...}) = 0
writev(4, [{"HTTP/1.1 200 OK\r\nServer: Zeus/3."..., 191}, {"<!DOCTYPE HTML PUBLIC \"-
//W3C//D"..., 1622}], 2) = 1813
poll([{fd=4, events=POLLIN, revents=POLLIN}, {fd=7, events=POLLIN}, {fd=9, events=POLLIN},
{fd=10, events=POLLIN}], 4, 9000) = 1
read(4, "GET /apache_pb.gif HTTP/1.1\r\nAcc"..., 511) = 249
stat("/usr/htdocs//apache_pb.gif", {st_mode=S_IFREG|0644, st_size=2326, ...}) = 0
writev(4, [{"HTTP/1.1 200 OK\r\nServer: Zeus/3."..., 191}, {"GIF89a\3\1
\0\367\0\0\377\377\377\316\316\316\245\245\245"..., 2326}], 2) = 2517
poll( <unfinished ...>
```

しきい値となっている。writev()は簡単にいうと、いくつかのメモリ領域をまとめて書き込むために利用する。ここでは、別々の場所に保持されたプレフィックスとmmap()で読み込んだデータをまとめて書き込むのに使用されている。

ここでは、ファイルをメモリへ読み込むのに、mmap()を使用しているが、Linux用のApacheでは、サイズにかかわらず（といっても1バイト以上）mmap()を使用するようになっている。そのほかtime()は、ファイルが変更されていないかどうかのチェックとログに時間を記録するために使用している。あとrt_sigaction()は、親プロセスなどとのシグナルをハンドリングするために実行している。

比較対象がないと今ひとつどうなのか分からないので、Zeos Web Server（http://www.zeustechnology.com/）の状態もチェックしてみよう（**リスト2**）。

Zeusは、Apacheよりもパフォーマンス的に優れているという評判もあるWebサーバだ。なお、以下はシステムコールを眺めただけで、Zeusのソースコードを見たわけではないので、基本的に予想でしかない。

Zeusは、Apacheと異なり、複数コネクションを処理するのに複数の子プロセスを生成しないで、１つのプロセスで対応している。

まずpoll()だが、これで待ち受け状態のソケット状態をチェックしている。デフォルトでオープンしているソケットは3つのようだ。このソケットは当然要求に応じて増加する（上限まではチェックしていない）。3行目で、待ち受けていたイベントが10番ファイルディスクリプタにマップしているソケットに発生している。続いて、そのソケットからデータを読み込むための処理を行っている。次に通常はread()でデータを読み込むのだが、Zeusでは、デフォルトでPerform DNS lookupの設定が有効（無効にすることも可能）になっているので、DNSに対して、その要求を行っている。これはコネクションのたびに発生するものではない。

DNSへの処理が終わると、read()で返答結果を読み込んでいる。そしてstat()でファイルの状態を確認した後、open()、read()している。そしてwritev()でプレフィックスとともにソケットに書き込んでいる。同様にapache_pb.gifの処理もこなしている。

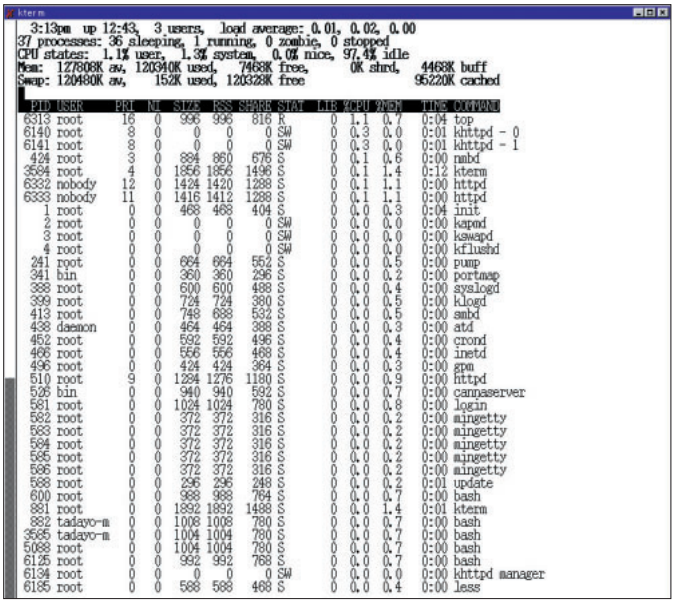
それに続くリストはクライアントから別コネクションで接続したものだ。違いに気がついただろうか？ この要求では、read()でクライアントからのリクエストを読み込んだ後、必要なファイルのstat()をチェックしただけで、writev()を行っている。Zeus自身がデータのキャッシュを行っているためだ。そのためにstat()で更新されていないことを確認しただけで、データを送信することができるのだ。

また例では、read()でデータを読み込んでいたが、これは比較的小さなファイルの場合に使用され、大きなファイルでは、mmap()が利用されている。ただし、Apacheと異なり、mmap()でファイルをハンドリングし、データを処理してからも、しばらくの間munmap()は実行されない。

つまりZeusでは、複数のコネクションを処理するのに複数の子プロセスを走らせたりせず、処理を1プロセスで行う代わりに、そのプロセスでデータをキャッシュしておき、システムコールのオーバーヘッドを軽減しているようだ。複数のコネクションを処理する部分では、内部的にユーザーレベルのスレッドが利用されているかもしれない。

一方Apacheは、複数の子プロセスで対応しているため、それぞれのプロセスでデータをキャッシュしてしまうと、それこそメモリの無駄遣いになってしまう。また親プロセス、もしくは専用プロセスでキャッシュして、それをプロセス間通信でやり取りするのもオーバーヘッドが大きい。そこで、OSのファイルキャッシュ機構とmmap()の効率に頼っているのだ。

ApacheとZeusをシステムコールレ



画面　kHTTPdを実行しているところ
プロセスIDの6140と6141に処理スレッドが、6134にマネージャが動作しているのがわかる。

ベルで見た場合、発行数のうえでは、Apacheは少し効率が悪い感じがする。ちなみに、システムコールのオーバーヘッドは筆者のテストマシン（Cyrix 6x86MX 233）で大体1.6usであった（単にsched_yield()を1000回呼んでかかった時間を1000で割っただけなので、ひとつの目安として考えてほしい）。

それぞれのプログラムが呼び出すシステムコールからすべてが分かるわけではないが、Webサーバの主要な仕事であるリクエストされたデータを単に送り出すというのは、非常に単純な作業なので、上記の影響が大きいということはいえる。

なおIISでは、基本的に各コネクションに対し、スレッドで対応する実装だ。またプロセス内でデータのキャッシュも行っている。そういった意味では、Zeusと似ている。しかし、NTのユーザースレッドはカーネルに実装されたもので、スケジュール、特に同じプロセスに属するスレッドを効率よく切り替えることができる。通常、プロセスの異なるスレッドであれば、切り替えに際して、そのメタデータすべてを保存し、セットし直す必要があるが、カーネルが一般に兄弟スレッドと呼ばれる同じプロセスに属するスレッドをきちんと把握していれば、コンテキストスイッチでは、スレッド特有のメタデータを操作するだけでよくなる。中でも各プロセスのメモリ空間関連は、仮想記憶機構のために複雑なものとなっているので、トータルとしては、かなりの効果をもたらす。

またApacheのWin32用にポーティングされた実装では、_begin_thread()を利用したマルチスレッドで各コネクションに対応するようになっている。Win32にはfork()がないのと、Win32ではスレッドが効果的だからだろう。

しかし、Webサーバの仕事はこれだけではない。もっと複雑なスクリプトの処理や、バックエンドのデータベースなどとの連携も求められている。そのため1プロセスで対応する場合、処理が複雑になったときのペナルティに不安があるのも確かだ。

Linuxでの新しい試み「kHTTPd」

Linuxでは、ベンチマーク対策かどうかは分からないが、Webサーバに関して、いくつかの面白い試みが行われている。そのひとつが、kHTTPd（http://www.fenrus.demon.nl/）だ。kHTTPdは、Webサーバの基本部分、つまり要求されたファイルを送信する部分をカーネルモードで動作するモジュールとしてしまうというものだ。現在では、開発版の最新カーネルkernel-2.3.14にも組み入れられている。Linuxでは、すでにNFSサーバがカーネルモードで動作している（そのうち、smbdもカーネルモードでということになるかもしれないが）。

kHTTPdは、Kernel httpd acceleratorと呼ばれ、ローダブルカーネルモジュールとして実装されている。カーネルスレッドが利用されており、処理を振り分けるマネージャと、実際に処理を行ういくつかのスレッドによって構成されている。

今のところスタティックなWebページ、つまり通常のファイルベースのリクエストのみを処理し、それ以外のリクエストは、ユーザーモードで動作している通常のWebサーバ（何でもよい）に渡すようになっている。カーネルモードで動作することに不安を覚える読者も多いと思うが、処理内容が実にシンプル（リクエストされたデータをソケットに書き込むだけ）なので、それほど危惧する必要はないと思われる。

コンパイルとインストールは通常のローダブルカーネルモジュールと同じ要領で行えばよい。

モジュールをロードすると、/proc/sys/net/khttpd/が構成される。khttpdの設定はここで行う。kHTTPd用のスタート、ストップスクリプトのサンプルは、**リスト3**のようになる。

モジュールをアンロードする場合は、stop_khttpdを実行した後、次のコマンドを実行し、rmmodなどを実行する。

```
echo 1 > /proc/sys/net/khttpd/unload
```

少し試してみただけだが、kHTTPdは、さすがにクライアントからのコネクションを受けてから、ファイルを送信するまでのすべての処理がカーネルモードで完結されるため、CPU使用率が低く抑えられている。ただ、もっと高い負荷をかけた場合にどうなるかは、試してないのでわからない。またカーネルモードで動作させるには、バッファの扱いを含めたセキュリティ面で、もう少し配慮する必要がありそうだ。しかし、今後大いに期待が持てる機能となるかもしれない。

▼リスト3　kHTTPd用のスクリプト

スタートスクリプト：start_khttpd

```
#!/bin/sh
modprobe khttpd
echo 8080 > /proc/sys/net/khttpd/serverport
echo 80 > /proc/sys/net/khttpd/clientport
echo /var/www >
/proc/sys/net/khttpd/documentroot
echo php3 > /proc/sys/net/khttpd/dynamic
echo shtml > /proc/sys/net/khttpd/dynamic
echo 1 > /proc/sys/net/khttpd/start
```

ストップスクリプト：stop_khttpd

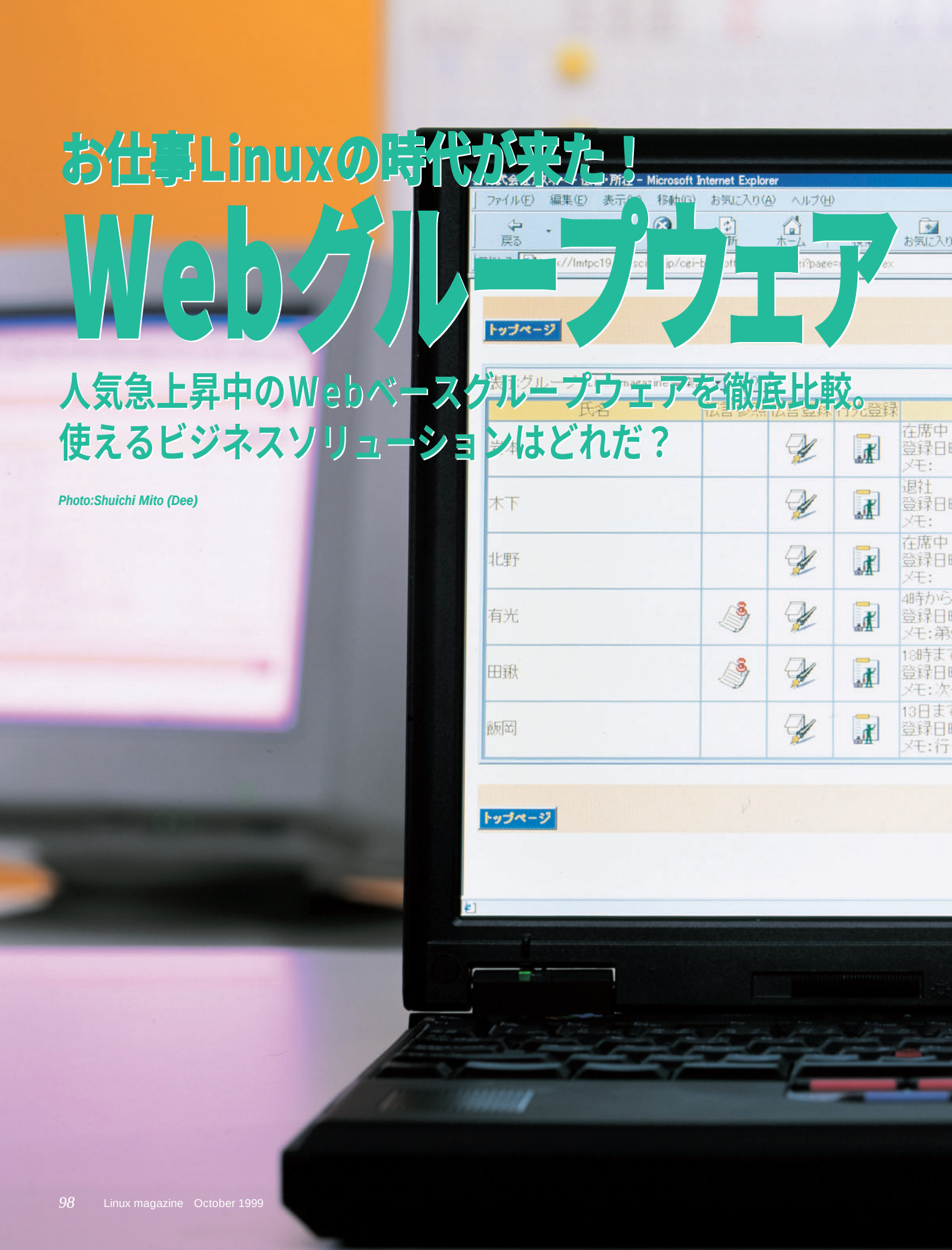
```
#!/bin/sh
echo 1 > /proc/sys/net/khttpd/stop
```

お仕事Linuxの時代が来た！

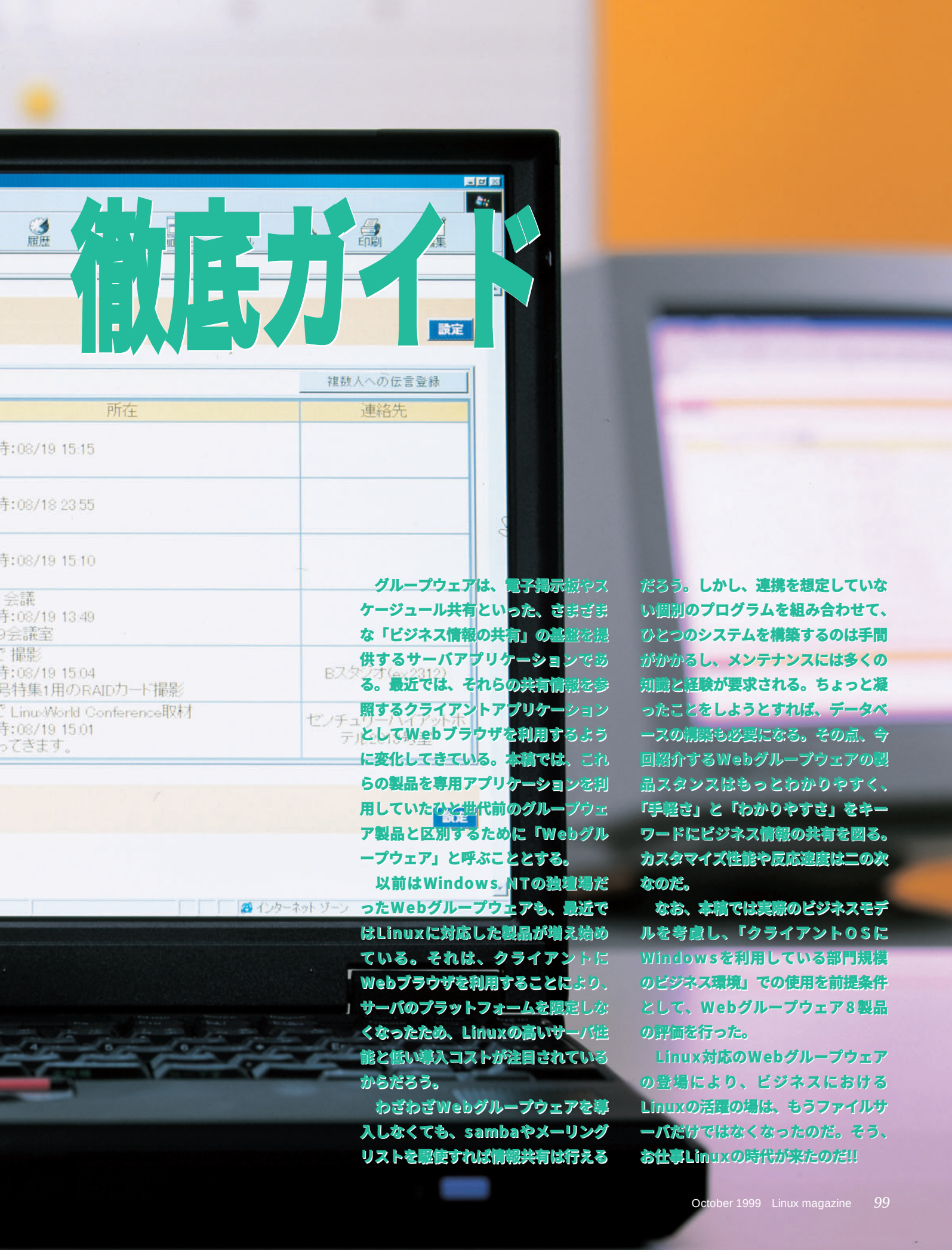
Webグループウェア

人気急上昇中のWebベースグループウェアを徹底比較。使えるビジネスソリューションはどれだ？

Photo:Shuichi Mito (Dee)



徹底ガイド



グループウェアは、電子掲示板やスケージュール共有といった、さまざまな「ビジネス情報の共有」の基盤を提供するサーバアプリケーションである。最近では、それらの共有情報を参照するクライアントアプリケーションとしてWebブラウザを利用するように変化してきている。本稿では、これらの製品を専用アプリケーションを利用していたひと世代前のグループウェア製品と区別するために「Webグループウェア」と呼ぶこととする。

以前はWindows NTの独壇場だったWebグループウェアも、最近ではLinuxに対応した製品が増え始めている。それは、クライアントにWebブラウザを利用することにより、サーバのプラットフォームを限定しなくなったため、Linuxの高いサーバ性能と低い導入コストが注目されているからだろう。

わざわざWebグループウェアを導入しなくても、sambaやメーリングリストを駆使すれば情報共有は行えるだろう。しかし、連携を想定していない個別のプログラムを組み合わせて、ひとつのシステムを構築するのは手間がかかるし、メンテナンスには多くの知識と経験が要求される。ちょっと凝ったことをしようとすれば、データベースの構築も必要になる。その点、今回紹介するWebグループウェアの製品スタンスはもっとわかりやすく、「手軽さ」と「わかりやすさ」をキーワードにビジネス情報の共有を図る。カスタマイズ性能や反応速度は二の次なのだ。

なお、本稿では実際のビジネスモデルを考慮し、「クライアントOSにWindowsを利用している部門規模のビジネス環境」での使用を前提条件として、Webグループウェア8製品の評価を行った。

Linux対応のWebグループウェアの登場により、ビジネスにおけるLinuxの活躍の場は、もうファイルサーバだけではなくなったのだ。そう、お仕事Linuxの時代が来たのだ!!

iOffice2000

ネオジャパン

問合わせ先　ネオジャパン
TEL：045-912-2145
FAX：045-912-5975
e-mail：ioffice@neo.co.jp
URL：http://www.neo.co.jp/ioffice/

Point

- Cobalt Qubeなどにも対応した幅広いプラットフォームサポート
- グループウェアに求められるものはほとんど持つ多機能
- 導入／管理もわかりやすく、簡単
- 60日間全機能を試せる試用版あり

「iOffice2000」は、ネオジャパンが開発し、Webなどを通じて販売されているWebグループウェアである（画面1）。LinuxやCobalt Qubeを始めとする多くのプラットフォームに対応し、12種類もの多くの機能を持った製品である。

製品を購入する前に50ユーザー版を60日間無償で試用できる。購入するとライセンスキーが発行され、利用期間の制限が解除される。

■インストール

CGI関連のファイルを適当なディレクトリにコピーするだけなので、インストールそのものも非常に簡単である。tar.gz形式のファイルならtarコマンドで展開したあと、mvコマンドでディクレトリを1つ移動させるだけである。rpm形式のファイルなら、“ rpm -i xxxx.rpm ”を実行するだけで終了である。

■管理機能

管理画面はかなりシンプルな構成となっている（画面2）。iOffice2000では管理者と一般ユーザーという区分はなく、初期設定では、すべてのユーザーが設定などを変更できるようになっている。もしも、ある特定のユーザーだけに管理をさせたい場合は、パスワード設定をしておき、パスワードの入力で管理権限の有無を判別するようにできる。

ユーザー登録にはCSV形式のデータからインポートができるので、大量にユーザー登録が必要な場合は便利だろう。逆にユーザーデータを出力するエクスポート機能もついている。

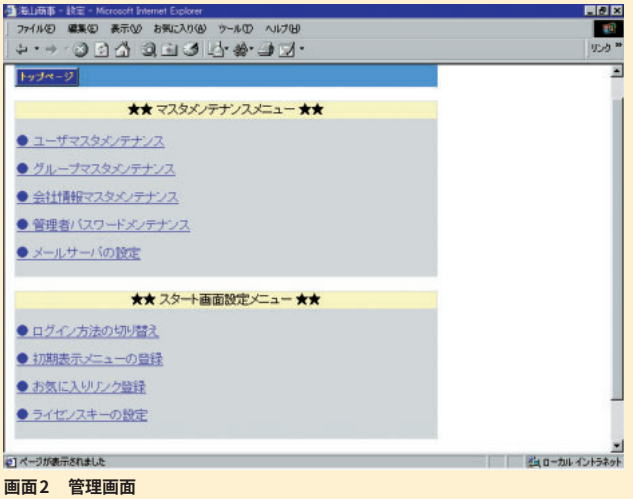
■標準機能

標準で実装されている機能は、スケジュール共有、仕事リスト、WebMail、伝言・所在、タイムカード、設備予約、ワークフロー、回覧板、掲示板、共有アドレス、電子会議室、文書管理の12個で、今回評価したWebグループウェア製品の中でも最も数が多い。紙面の都合で、これらすべてを紹介することができないので、主要な機能についてのみ紹介する。

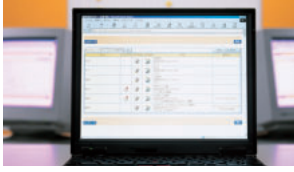
・スケジュール

各個人のスケジュールを公開して、グループメンバー全員のスケジュールを参照する機能である（画面3）。表示形式は、個人は1日／週間／月間の3モードとグループは1日／週間の2モードで、計5つのモードがあり、ボタンで簡単に切り替えが可能である。なお、他のメンバーに公開したくないプライベートなスケジュールは、設定時に［情報公開レベル］で非公開にしておけば公開されない。ただ、スケジュール追加画面を呼び出すための［Add］というリンクがすべての日に表示されるので、画面が煩雑な印象を受ける。

画面1　iOffice2000のスタートアップ画面



画面2　管理画面



・仕事リスト

仕事の内容を「件名」、「期限」、「優先度」、「進捗状況」、「達成率」でカテゴリ分類し、「件名」、「期限」、「優先度」においてはそれぞれの検索条件で並べ替えが可能である。

・WebMail

WebMailは、Webブラウザを利用して、メールサーバ上にあるメールを参照する機能である（画面4）。そのため、Webブラウザさえあれば、専用メーラがない環境（外出先など）からでも社内の自分のメールの送受信が可能となる。さらに、送受信したメールは使用後もクライアントには一切残らないので、外出先で他人のパソコンを借用したとしても、メールを読まれる心配はない。初期設定では、他のメーラと併用してメールを読むことを想定して、サーバからメールを削除しない設定になっている（削除するモードもある）。

・伝言・所在

行き先を登録しておき、グループメンバーの動静を一覧することができる。後述のタイムカード機能と連携しているので、出社の打刻を行うと自動的にこちらにも「在席中」と表示される。

・タイムカード

スタートアップ画面にある「出社」「退社」「外出」「戻り」の打刻データを管理する機能。CSV形式のデータとしてエクスポートが可能であるため、他システム（勤怠管理など）と連動させることが可能である。通常勤務、深夜勤務など複数の勤務体系が登録でき、複数の勤務時間の管理が可能となっている。

・ワークフロー

数種類の「届け出」「申請」を登録してある雛形フォーマットで承認ルーチンに流す機能。Web Mailとの連携機能があり、事前通知がメールでなされるので、承認が滞ることが減り、申請者も現状を確認できる。

・そのほか

上記の機能以外にも、全社員への通知、グループ内でのお知らせを回覧する［回覧板］、メンバーへの告知システムである［掲示板］、個人で保有しているアドレス帳を全社で共有する［共有アドレス］、オンライン上で仮想的な会議システムを実現する［電子会議室］、文書を共有化して、サーバ上で一元管理する［文書管理］などがある。

■多機能が魅力

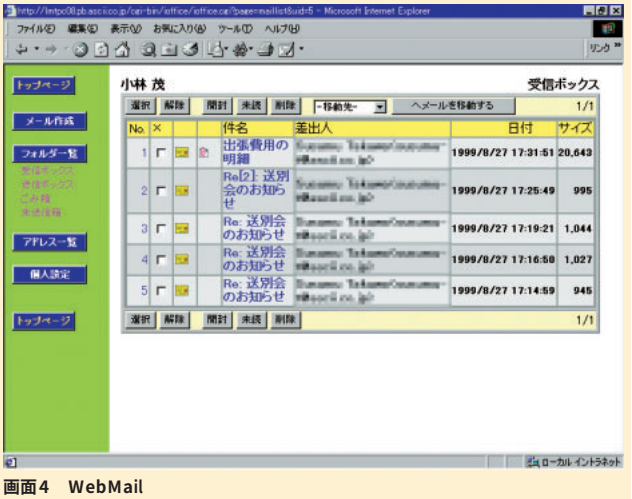
この機能の多さは魅力的である。さらに、ワークフローのような中規模環境を意識した機能も実装されており、それぞれの機能の水準も高い。管理しやすさを優先し、アクセス制御はあまり厳密に行っていないため、大規模環境の導入は難しいと思うが、それ以外の環境ならどこでも対応できる機能を持った製品である。

導入のしやすさ	★★★★★
管理のしやすさ	★★★★☆
サービスの使いやすさ	★★★★☆
価格	★★★★☆

ユーザー数	価格
5ユーザー	3万9800円
10ユーザー	6万4300円
20ユーザー	9万8000円
30ユーザー	13万7600円
50ユーザー	19万8000円
100ユーザー	37万8000円
150ユーザー	54万8000円
200ユーザー	69万8000円
250ユーザー	84万8000円
無制限	99万8000円

価格表

画面3　スケジュール



画面4　WebMail

イントラネット for Linux Ver.2.0 Standard版

開発　ギデオン
販売　富士マグネディスク

問合わせ先　富士マグネディスク
TEL：0424-81-8222
FAX：0424-89-8610
e-mail：fmdinfo@fmd.fujifilm.co.jp
URL：http://www.fujifilm.co.jp/fmd/lintop.html

Point

- 充実したWebメール機能
- ユーザーごとにデータを保存、管理できるマイボックス機能
- メールと親和性の高い電子会議機能
- クライアント数無制限のライセンス体系

「イントラネット for Linux Ver.2.0 Standard版」は、ギデオンが開発し、富士マグネディスクが販売しているWebグループウェアパッケージだ。Web上でのメール送受信、掲示板、電子会議、ホワイトボード（予定表）などの機能を実現している。

高機能なWebメール機能と、ユーザーごとにデータを保存、分類できる［マイボックス］機能が特徴的だ。また、サーバライセンス方式を採用し、クライアント数に制限がないこともポイントと言える。

■インストール

動作環境としては、姉妹製品の「プロサーバ for Linux Ver.2.0」以降に対応となっているが、動作保証はないものの、国内で流通しているほとんどのディストリビューションで動作実績がある。詳細は、製品のWebページで確認して欲しい。今回は、Vine Linux 1.1CRの動作するマシンにインストールしたが、問題なく動作した。

インストール作業は、コンソール画面から専用インストーラを起動し、質問に答える形で行う。「プロサーバ for Linux Ver.2.0」以外のディストリビューションを利用している場合は、インストーラが検出するSendmailや、Webサーバの設定ファイルのパス名などを確認する必要がある。正しく検出されなかった場合は、環境に合わせ正しいパス名を指示する。

■初期設定

初期設定などの作業はすべてWebブラウザから行う。

最初に管理者のユーザー登録を行う。次に［管理者メニュー］（画面1）からユーザー登録や電子会議、掲示板などの設定を行う。

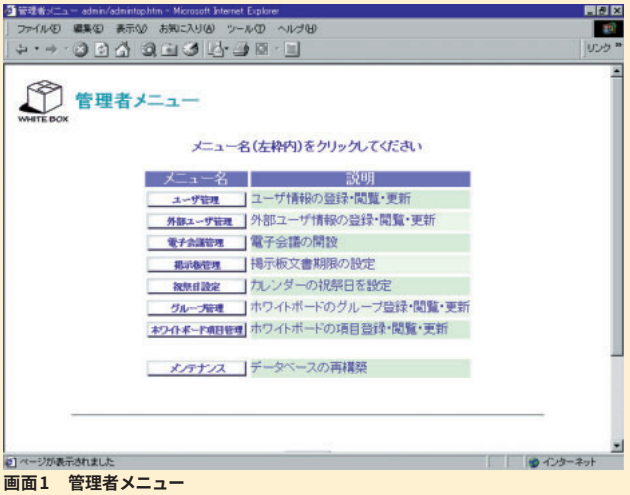
各ユーザーには、ログイン名などのほかに、0〜99の間の数値でアクセス権限を設定する。これは掲示板と電子会議の利用をユーザーごとに制限するためのものだ。

一般のユーザーのほかに、取引先などイントラネットの外にメールアドレスを持つユーザーを「外部ユーザ」として登録することができる。「外部ユーザ」には、指定した電子会議や掲示板のメッセージをメールで配信できる。

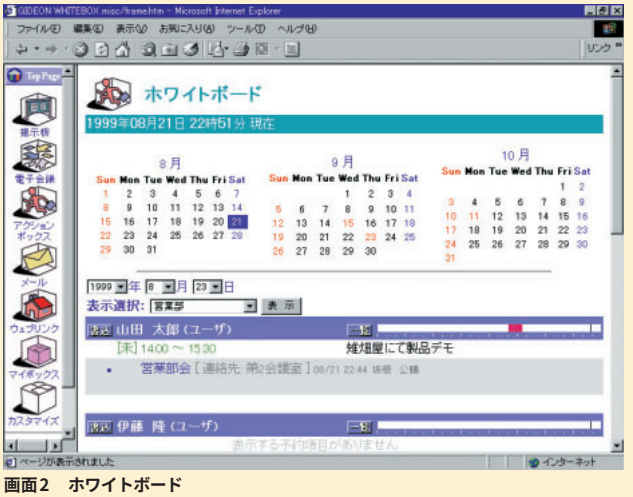
［電子会議］や［掲示板］は、それぞれ複数開設でき、個別にアクセス権を設定する。

ホワイトボードから施設予約する会議室などは、［ホワイトボード項目管理］で登録する。

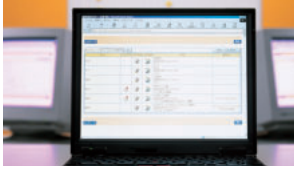
部やプロジェクト単位で「グループ」を作り、そこにユーザーや施設を登録する。「グループ」は［ホワイトボード］で予定を表示、設定する際に利用される。



画面1　管理者メニュー



画面2　ホワイトボード



■ホワイトボード

ユーザーがログインすると、3カ月分のカレンダーとその日の予定を表示するホワイトボード画面になる。ここで予定を追加したり、表示する（画面2）。［一覧］アイコンを押すと、1週間から最長1年間分の予定を一度に確認できる。また、ほかのユーザーの予定や施設の予約状況も、グループごとに表示、追加可能だ。

■掲示板

掲示板は告知したい情報を公開する機能だ。掲示板は管理者が話題ごとに設置するが、個々の掲示板には誰でもメッセージが書ける。この製品では、掲示板のほかに［電子会議］が用意されており、告知には掲示板を、意見交換には［電子会議］を利用する。管理者は掲示板への掲載期間を定めることもでき、これを過ぎたメッセージは自動的に削除される。

■電子会議

電子会議は、特定の話題について意見交換を行うための機能だ。各会議に参加する際に、会議のやり取りをメールでも受け取るかどうか設定する。

会議の内容は、サブジェクトごとにメッセージとコメントが一括表示される（画面3）。参加者はコメントを書いたり、新規サブジェクトで新たなスレッドを作ることが可能だ。

■メール

Webメール機能によってメールの送受信を行う（画面4）。受信メールだけでなく、送信したメールも保存されるほか、ファイル添付にも対応するなど、Webメールとしては高機能だ。あらかじめ設定しておいた定型文や署名を挿入する機能も備えている。また、メールの送信先に電子会議を選べば、選んだ会議にメッセージが書き込まれる。

■マイボックス

文書を保存しておく機能で、それぞれのユーザーごとに複数のマイボックスを作成できる。マイボックスにはその場で作成した文書だけでなく、掲示板、電子会議、メールの文書を保存可能だ。文書は、自分で削除しない限り消えないので、重要なメッセージを保存するのにも向いている。さらに、保存してある文書をメールで送信する機能をもつ。

■情報共有とデータ管理に長けた秀逸な製品

製品パッケージには、製本されたマニュアルが付属し、インストール方法、システム管理、各サービスの利用方法が詳しく説明されているので、導入や運用で戸惑うことはないだろう。

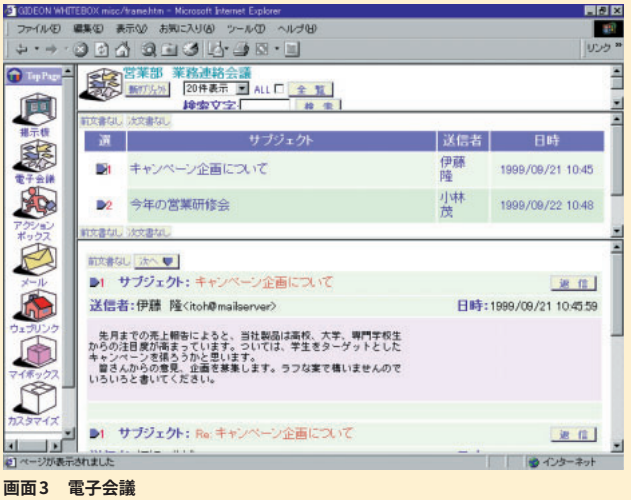
実用的なWebメールは、パソコンに慣れていないユーザーにも利用できる。また、すでにメールを利用しているユーザーにもメーリングリストとして利用できる電子会議は有用だろう。ただ、数値のみで設定するアクセス権は少々使いづらいので改善を望みたい。

マイボックス機能でユーザーごとに文書を管理でき、掲示板、電子会議、メール、マイボックスそれぞれが検索機能をもつなど、情報共有のみならず、個人でのデータ管理機能が充実した製品だ。

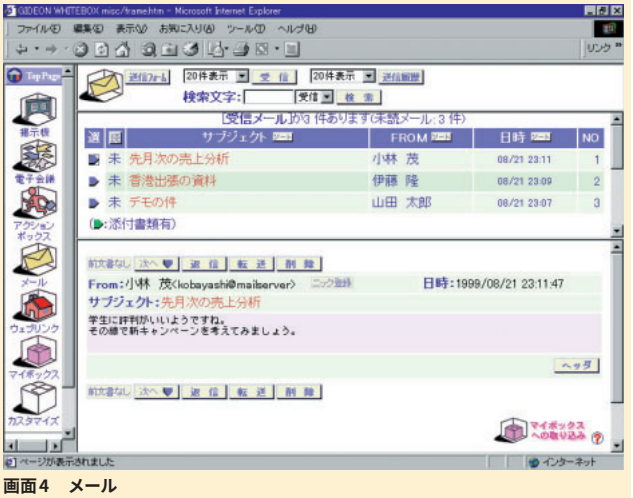
導入のしやすさ	★★★★☆
管理のしやすさ	★★★★☆
サービスの使いやすさ	★★★★☆
価格	★★★★★

ユーザー数	価格
1サーバ／クライアント無制限	5万9800円

価格表



画面3　電子会議



画面4　メール

オフィスケープ

NECソフトウェア四国
問合わせ先　NECソフトウェア四国
TEL：089-947-7222
FAX：089-947-3975
e-mail：office_support@d3.sysd.snes.nec.co.jp

URL：http://www.mesh.ne.jp/snes/seihin/office_htm/office_main.htm

Point

- 日常業務に有効な行き先案内と伝言メモ
- ユーザーが自由に開設できる掲示板
- スケジュールに定期イベントを登録可能
- 施設を定期的に予約可能
- ログインせずに閲覧できる公開情報

「オフィスケープ」は、NECソフトウェア四国がWebでオンライン販売しているWebグループウェアだ。

手軽に利用できる行き先案内と伝言機能、ツリー表示も可能な掲示板機能が特色だ。また、スケジュールや行き先案内などはログインせずに閲覧できる設計になっている。

製品を購入する前に、10ユーザー版を60日間無償で試用できる。購入するとライセンスキーが発行され、利用期間の制限が解除される。

ディストリビューションは、TurboLinux 3.0／4.0、および日本語redhat 5.2に対応する。今回は、最新のVer.1.04をWebでダウンロードし、TurboLinux 3.0にインストールした。また、動作保証外だが、Vine Linux 1.1CRにインストールしても問題なく動作した。

■インストール

ダウンロードしたファイルには、インストール方法を詳しく説明したドキュメントが含まれている。

まず、アーカイブから抽出されたMakefileを環境に合わせて編集する。TurboLinux 3.0、Vine Linux 1.1CRの場合、CGIプログラムとHTMLファイルの格納ディレクトリをそれぞれ修正する必要があった。

suコマンドでrootになり、make installを実行すると、設定したディレクトリにファイルがコピーされる。環境によっては、cpやlsなどのパスを設定したファイルを変更する必要がある。最後に、期限が過ぎたデータを定期的に削除するようにcrontabを設定してインストールが終了する。

■初期設定

最初にアクセスしたときには［オフィス設定メニュー］画面になる。まず、オフィス名、背景イメージ、管理モードに入るためのパスワードを設定する。

次にグループを設定する。グループは掲示板や、グループごとのスケジュールへのアクセスを制限したり、行き先案内などで最初に表示されるメンバーを指定するのに使われる。ユーザーを複数のグループに登録することもできる。

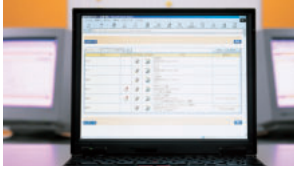
グループを作ったら、ユーザーを登録する。ユーザーを登録する際には、表示順を決める「メンバID」、「メンバ名」、「パスワード」などのほか、所属するグループを指定する。ユーザーを複数のグループに所属させる場合は、あとから所属グループを追加する。

■行き先案内

行き先案内は、同じグループに属するユーザーの行き先、連絡先、メッセージなどが一覧表示される機能だ（画面1）。ド

画面1　行き先案内

画面2　1カ月の予定



ロップダウンリストによって、別のグループや全ユーザーの行き先を表示することもできる。ここでは、自分の行き先を記入することはもちろん、自分以外のユーザーの行き先を代理入力できるようになっている。予定変更の電話を受けた場合などに使うと便利だろう。

この画面からユーザーを選び、伝言メモを送付することができる。不在の人宛に電話や来客があったときなど、日付、時刻入りのメモをすぐに用意できる。

■スケジュール

スケジュール画面では、所属するグループ全体や、各メンバーの予定が表示される。スケジュールは、自分の予定、グループの予定のどちらにも記入可能だ。初期状態では1週間表示だが、1カ月表示にすることもできる（画面2）。

また、定例会議のような定期イベントを設定できるのはありがたい。さらに、ほかのメンバーからは見えない予定を記入する機能もある。

■掲示板

掲示板には、新規メッセージとコメントを書き込むことができ、機能的には電子会議と言った方がよいかもしれない。各ユーザーが自由に新しい掲示板を作成できるのはありがたい。新規に作成する時に、アクセスできるグループを限定することも可能だ。掲示板全体からキーワードで検索したり、表題をツリー表示できるなど、使い勝手を重視したものといえよう（画面3）。

■施設予約

会議室や社用車などの施設を予約する機能も備えている（画面4）。予約した内容を、自分宛にメールすることもできるので安心だ。さらに、定期予約を行うこともできるので、定期的な会議を予約する場合などに威力を発揮する。

■アドレス帳

アドレス帳は、メールアドレス、電話番号、主となるグループを一覧表示する。メールアドレスはリンクとして表示されるので、クリックしてメールクライアントを起動できる。

■一般のオフィス向け機能が充実した実力派

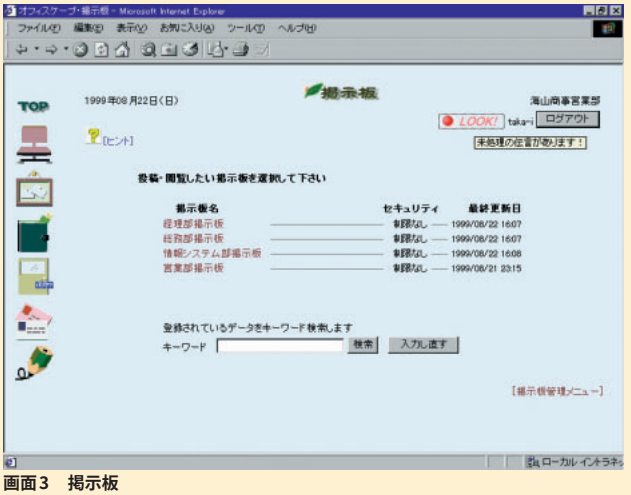
各サービスの画面には、［ヒント］のリンクが用意されており、ここをクリックするとそのサービスの利用法が表示される。ユーザーはここを見ながら使い方を学べるので、ユーザー教育のための管理者への負担も減る。ただ、ログインせずにスケジュールや行き先案内を見られるのは便利だが、セキュリティには気を付けたい。

「オフィスケープ」は、オフィスでの実用性に的を絞ったグループウェアだ。気軽に利用できる行き先案内と伝言メモ、ユーザーが自由に設置できる掲示板機能など、日常的な業務の効率アップに貢献するだろう。

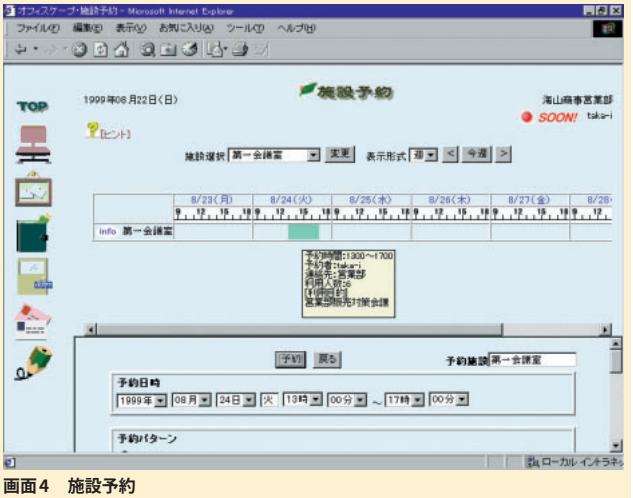
導入のしやすさ	★★★☆☆
管理のしやすさ	★★★☆☆
サービスの使いやすさ	★★★★☆
価格	★★★★★

ユーザー数	価格
10ユーザー	2万5800円
20ユーザー	4万8000円
50ユーザー	8万8000円
100ユーザー	16万6000円
※機能別販売あり	

価格表



画面3　掲示板



画面4　施設予約

CommunityBase

NTTビジュアル通信

問合わせ先　NTTビジュアル通信
TEL：03-3589-1213
FAX：03-3589-1748
e-mail：combase@nttvics.co.jp
URL：http://www.nttvics.co.jp/COMBASE/index.html

Point
- ●OSごとのインストールで専門知識不要
- ●メーリングリストと親和性の高い会議室システム
- ●説明付きでデータを公開できるライブラリ機能

「CommunityBase」は、NTTビジュアル通信が販売しているコミュニケーションソフトウェアだ。

電子会議、Webを通じてのファイル交換、チャットなどの機能を持つ情報交換サーバを手軽に構築できるのが特徴だ。

今回は、同社より提供された「CommunityBase Ver.1.0（お試し版）」を試用した。お試し版には、ユーザー数5名、設置できる会議室が1つという制限がある。

■コミュニティ

「CommunityBase」では、コミュニティという単位で情報交換用のWebページが作られる。たとえば、「営業部」というコミュニティを作ると、「営業部」コミュニティのページが用意される。ここには［電子会議室］、［ライブラリ］（ファイル交換）、［チャット］、［コミュニティ主催者からのお知らせ］のうち、好きなサービスを選んで設置できる。

コミュニティは一般のユーザーが開設し、そのユーザーが主催者として管理する。コミュニティへのアクセス認証の有無、自動入会処理をするかどうかなどは主催者が設定する。

ユーザーは、利用したいコミュニティのURLを直接指定するか、トップページでコミュニティを選んでサービスを利用することになる。複数のコミュニティを作ることができ、それぞれのコミュニティは独立しているので、複数の部署での利用にも対応できる。

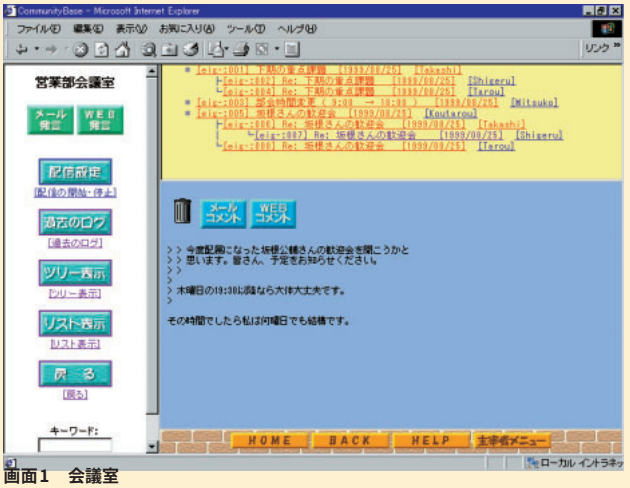
■インストール

「CommunityBase」は、SlackwareベースのLinuxとともにインストールされる。このため、そのマシンはほとんど「CommunityBase」専用機となるわけだが、かえってLinuxの知識や、ディストリビューションごとの差異に気を使う必要がないともいえる。インストール時にハードディスクが初期化され、既存のデータはすべて消却されるので注意が必要だ。

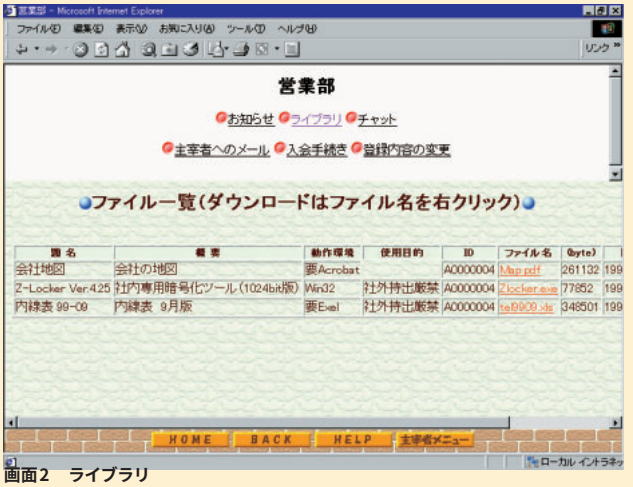
お試し版のパッケージには、ワープロで作られたインストールマニュアルが付属する。まず、2枚の起動フロッピーディスクからLinuxを起動し、rootでログインしたら、インストーラを実行する。すると、ハードディスクにパーティションが作成され、CD-ROMからLinuxのファイルがコピーされる。次に、ホスト名などのネットワーク設定と、メモリ搭載量を設定し、画面の指示にしたがってマシンを再起動する。最後に「CommunityBase」のインストーラを実行し、システム管理者のユーザー名とパスワードなどを設定すれば作業は終了だ。

■初期設定

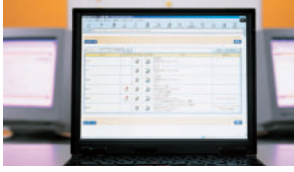
Webブラウザで管理用画面のURLにアクセスすると、ユーザー名とパスワードを入力するダイアログボックスが開くので、



画面1　会議室



画面2　ライブラリ



インストール時に設定したシステム管理者のユーザー名とパスワードを入力する。管理用画面からシステムの運用状態を細かく表示することもできるが、システムに慣れてから見れば十分だろう。システム管理の詳細は、添付のシステム管理マニュアルが参考になる。

初期状態では、ユーザー登録、コミュニティの開設はユーザーからの申請に対し自動承認されるようになっている。システム管理者が行うのは、申請受付時に自動返信されるメールの内容を設定することだ。もちろん、システム管理者が申請を受け取り、承認／却下するようにもできる。

■会議室

会議室では、サブジェクト一覧から読みたいものをクリックすると本文が表示される。一覧は、時系列のほか、スレッド表示させることも可能だ（画面1）。また、サブジェクトと本文のキーワード検索機能も備えている。

この会議室は、メーリングリストをベースに作られているので、会議室のやり取りをメールで行うことも可能だ。もちろん、メールでの配信を行うかどうかは、個々のユーザー自身で設定できる。ブラウザで設定可能なメーリングリストサーバとして利用するという使い方もいいかもしれない。

■ライブラリ

Webベースでのファイル交換機能だ。コミュニティのメンバーは、ファイルの登録、ダウンロードができる。登録したファイルには、題名や概要、動作環境などの情報を付加できるので、目的のファイルを探すのに役立つ（画面2）。グループのメンバーで共有するドキュメントやツールなどを登録しておくと便利だろう。

システム管理者は、コミュニティごとにファイルの総容量を制限できる。

■チャット

コミュニティにアクセスしているメンバー同士でリアルタイムにメッセージを交換できる（画面3）。メッセージ画面を更新するためのリロードを自動で行うことも可能だ。自動リロードの間隔は、15秒から120秒までの5段階から選択する。

■お知らせ

コミュニティ主催者からメンバーに向けてのメッセージを掲示する。掲示できるのは主催者のみなので、サービスにかかわる告知などを掲示するのに利用する（画面4）。

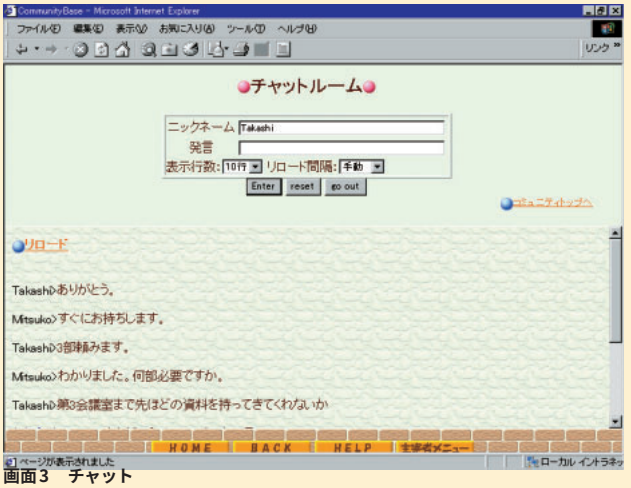
■グループウェアの枠にとらわれないサーバパッケージ

「CommunityBase」は、グループウェアというよりも、Webをベースにした情報交換サーバパッケージと言えよう。イントラネット内にサーバを設置すれば、情報共有を目的としたグループウェア的に利用することも可能だ。また、セキュリティのスキルがあればインターネットでのサービス提供にも応用できるだろう。動作に必要なツール類が組み込まれたLinuxをセットにすることで、Linuxの知識がほとんどなくてもこれらのWebサービスを提供できる。しかし、Linuxとセットになっていることで、事実上マシンを占有してしまうなど、柔軟性に欠ける面があることも否めない。各種ディストリビューションに対応した単体パッケージの発売を期待したい。

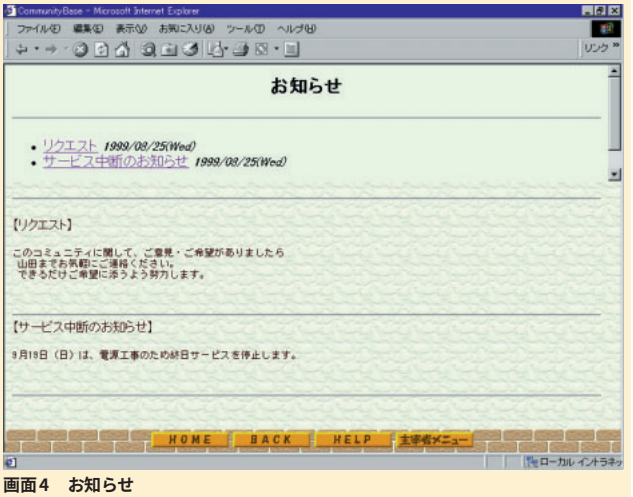
導入のしやすさ	★★★☆☆
管理のしやすさ	★★★☆☆
サービスの使いやすさ	★★★☆☆
価格	★★★★☆

ユーザー数	価格
100ユーザー	19万8000円
200ユーザー	34万8000円

価格表



画面3　チャット



画面4　お知らせ

サイボウズ Office 2（Linux版）

サイボウズ

問合わせ先　サイボウズ
TEL：06-6485-2220
FAX：06-4802-8487
e-mail:info@cybozu.co.jp
URL:http://cybozu.co.jp/

Point
- コンパクトでよくまとまった標準機能
- 簡潔でわかりやすい管理機能
- 他の製品にはないプロジェクト管理
- 60日間全機能を試せる試用版あり

「サイボウズ Office 2」は、サイボウズが開発し、Webなどを通じて販売されているWebグループウェアである（画面1）。また、各機能はコンポーネント化されているので、今回紹介するOffice製品以外にも、各機能ごとに購入することも可能である。

製品を購入する前に60日間無償で試用できる。購入するとライセンスキーが発行され、利用期間の制限が解除される。

■インストール

インストールは、まずtar.gz形式で配布されているファイルをCGIが実行可能なディレクトリ（TurboLinuxでは/home/httpd/cgi-bin/）に展開する。さらに、そこに作成されたcb2ディレクトリにあるMakefileを修正してデータディレクトリを指定し、“make install”でデータディレクトリを作成して、インストールは完了となる。あまり難しい作業ではないが、ファイルの修正やmakeなどは慣れないユーザーには不安が伴うものだと思う。インストーラの改善を望みたい。

■管理機能

管理できる項目は、iOffice2000とほぼ同様である（画面も似ている）。こちらも、管理者と一般メンバーという区分はなく、全員が環境設定を変更できるようになっている。変更権を一部の人に限定したい場合は、パスワード設定を行い、そのパスワードを通知することで実現する。設定で悩まされるようなところはなく、素直な作りとなっている（画面2）。

■標準機能

標準で実装されている機能は、スケジュール、行き先案内板、掲示板、施設予約、共有アドレス帳、ToDoリスト、プロジェクト管理、電子会議室の7つで、よく使われるような機能はほとんど網羅されているといっていいだろう。主な機能は、次のとおり。

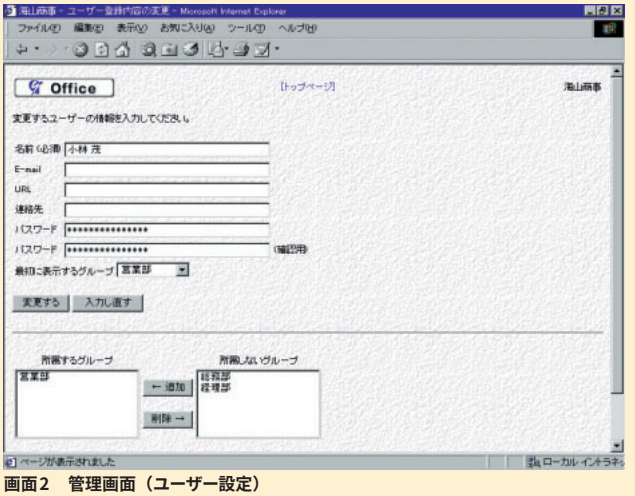
・スケジュール

メンバーのスケジュールを公開し、動静を把握できるようにする（画面3）。常にグループ表示されるようになっており、週間表示と1日表示をアイコンのクリックで切り替えられる。予定は、公開するか非公開にするかを選択できる。

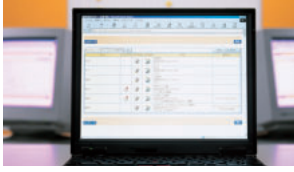
・行き先案内板

メンバーの行き先を公開する。他のメンバーはそれを参照し、不在中に連絡などがあれば伝言を残すことができる。プルダウンメニューで表示される行き先はカスタマイズ可能なので、頻度の高い行き先を登録しておける。

画面1 サイボウズ Office 2(Linux版)のスタートアップ画面



画面2　管理画面（ユーザー設定）



・掲示板

あらかじめ設定しておいたテーマについて、メンバーが自由に書き込みができる機能。また、キーワード検索を用いて、読みたい記事を探し出すことが容易である。検索条件としては、投稿者の名前のほか、記事のタイトルや本文からも検索してくれる。また、書類などを添付して投稿することもできる。

・施設予約

会議室や備品など共有リソースの予約状況を一覧表示する。収容人数、設備などがわかる詳細表示がついており、予約の際便利である。

・共有アドレス帳

「五十音INDEX」と「キーワード検索」で、素早いデータ検索が可能となっている。アドレス項目もカスタマイズが可能。アドレスデータは、CSV形式のデータのインポート／エクスポートに対応しているので、他のアプリケーションとの連携が可能。

・プロジェクト管理

プロジェクトの進捗状況をガントチャートで表示する機能で、他の製品にないユニークな機能（画面4）。進捗状況を自動判別し、遅れているプロジェクトには警告を表示してくれる。基本的な機能に限定されるが、あまり大きな規模のプロジェクトでない限り、有効に使えるはずだ。

■サイボウズ Office 3

本稿執筆時点ではまだβテストの段階だが、「サイボウズOffice 3」が公開されている。ただし、Windows版のみの公開で、Linux版は公開されていない。製品リリースの時期についても未定のようだ。主な強化点は、

・パーソナライズ機能

トップページを個人ごとにカスタマイズ可能となる。

・既存の機能の改良

現バージョンでのユーザーの要望を元に、新機能を追加し、さらに使いやすくした。

・新コンポーネント追加

要望の多い「Webメール」「文書管理」「ワークフロー」という3つのアプリケーションを追加。

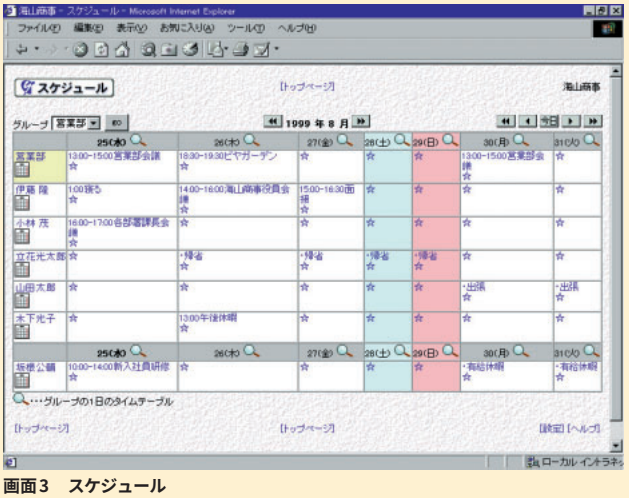
■まとまりのよさが身上

機能は多すぎず、少なすぎず、コンパクトによくまとまっている。管理項目も簡潔で設定しやすい。ただし、その分カスタマイズの柔軟さには欠ける。さらに、各機能へのアクセス権の設定（閲覧可能、変更可能）を、ユーザーごとに設定しなければならないため、大人数の使用には不向きかもしれない（標準では、すべてのユーザーがすべての情報を閲覧／変更可能である）。専任の管理者を置かないような小さい部門内での情報共有に向いているだろう。

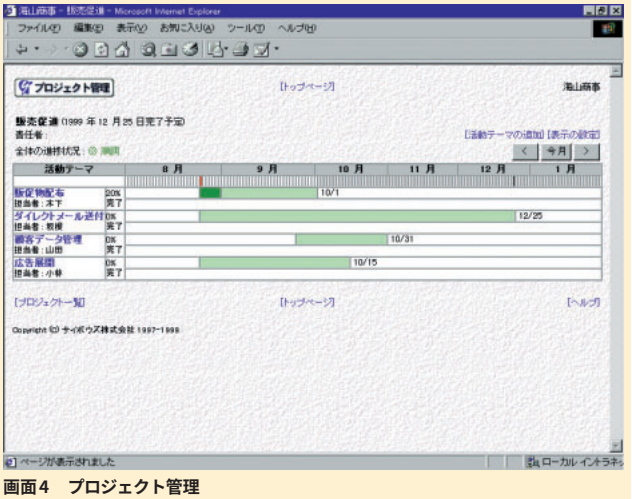
導入のしやすさ	★★★☆☆
管理のしやすさ	★★★★☆
サービスの使いやすさ	★★★★☆
価格	★★★☆☆

ユーザー数	価格
10ユーザー	―
50ユーザー	19万8000円
100ユーザー	38万円
無制限	88万円

価格表



画面3　スケジュール



画面4　プロジェクト管理

BiSESSiON Lite for Linux

B-Session Inc.

問合わせ先 B-Session Inc.
TEL：+1-408-577-1763（米国）
FAX：+1-408-435-1152（米国）
e-mail：webmaster@b-session.com
URL：http://www.b-session.com/products/bisession.shtml

Point

- 申請経路を自由に設定できるワークフロー機能
- 受領確認のできる回覧機能
- きわめて安価な価格設定
- 5ユーザーまで、全機能を試せるWeb配布版あり

「BiSESSiON Lite for Linux」は、B-Session Inc.がWebを通じて販売しているWebグループウェアだ。B-Session Inc.は米国にある会社だが、本製品は日本語化したものではなく、日本向けに作られたものだ。

ユーザーが個別に経路設定のできるワークフロー機能、受領を確認できる回覧機能が目玉だ。また、100ユーザーまで年間100USドルというきわめて安価なライセンス料も魅力だ。

今回は、Webサイトから無料でダウンロードできる5ユーザー版をVine Linux 1.1CR上で試用した。

■インストール

インストール方法は、Webサイトのダウンロードページで説明されているので、インストールする前に製品紹介URLからリンクをたどって欲しい。

Webで配布されているファイルを/usr/local/ディレクトリで展開し、ファイルのオーナー／グループをhttpデーモン（通常はApache）がアクセスできるように変更する。Vine Linux 1.1CRの場合、Apacheの実行ユーザー／グループはともにnobodyとなる。

次に、利用するためのURLをhttpデーモンが解釈し、CGIが実行できるようにhttpデーモンの設定ファイルを修正する。Vine Linux 1.1CRに付属のApacheの場合、srm.confとaccess.confにそれぞれ数行書き加えるだけだった。設定がすんだら変更内容を反映させるためにhttpデーモンを再起動する。

■初期設定

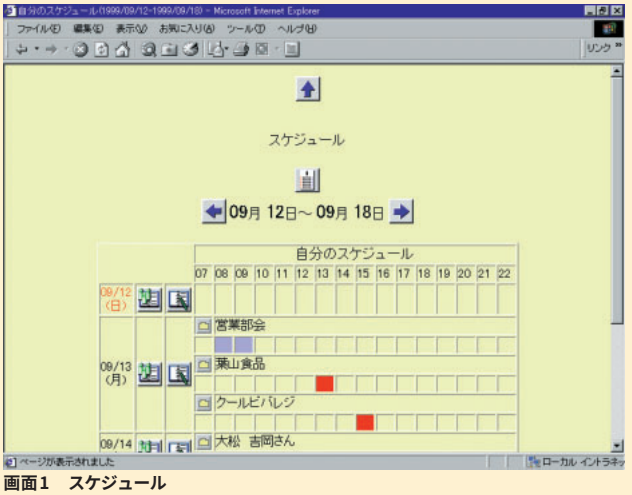
Webブラウザでアクセスし、システム管理者としてログインする。最初に行うのは、ユーザーの登録だ。ユーザーは、氏名、読み、ログインネーム、パスワードのほか、社員番号も登録できる。

次に、［掲示板］を設置する。話題ごとに複数作成すると便利だろう。もちろん、あとから追加／削除もできるので、難しく考える必要はない。

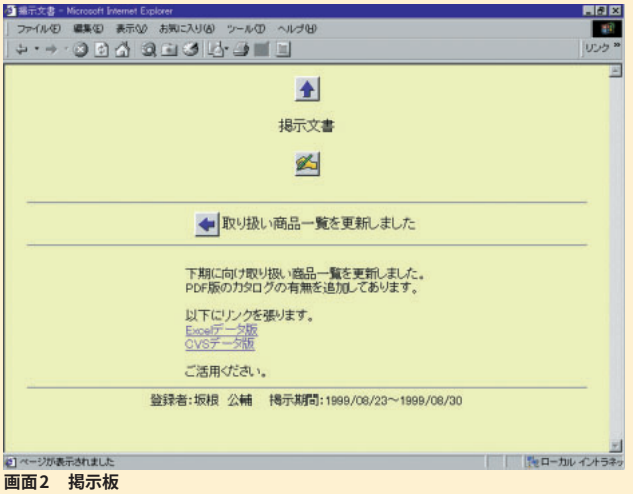
最後に、デフォルトで設定されているシステム管理者のパスワードを変更し、セキュリティを確保しよう。

■環境設定

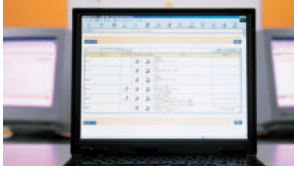
各ユーザーは、環境設定画面から、ほかのユーザーをまとめたグループを登録することができる。グループは、複数作ることができ、［スケジュール］でのグループ表示や［連絡箱］の回覧機能で利用される。ここで設定したグループは全員で共有するのではなく、ユーザーごとに管理される。また、パスワード変更もここで行う。



画面1　スケジュール



画面2　掲示板



■スケジュール

デフォルトでは、自分の予定が1週間分表示される（画面1）。カレンダーのアイコンを選んで、ほかの週のスケジュールを開くこともできる。

スケジュールの画面では、予定を追加したり、すでに記入してある予定の詳細を確認することができる。

さらに、利用者全員、あるいは指定したグループメンバーのスケジュールを確認したり、予定を記入することも可能だ。

スケジュールの記入は簡単にできるのだが、スケジュールの詳細を1件ずつしか確認できないのが少し面倒だ。

■掲示板

掲示板は、話題ごとにシステム管理者が用意する。掲示板という名前だが、むしろ電子会議に近い位置付けのサービスになっている。

掲示板に書き込むときに、テキスト形式のほかHTML形式を選べるので、HTMLタグが書ければ、文字装飾をしたり、リンクを張ることも可能だ（画面2）。

各掲示には、1日単位で掲示期間を設定するが、自分で掲載したものは、いつでも削除できる。また、システム管理者はどの掲示も削除できる。

■連絡箱

連絡箱には、回覧と申請の機能がある（画面3）。回覧は、利用者全員、またはグループメンバー宛にメッセージを同報する機能だ。回覧を送付したメンバーが読んだかどうか確認することもできるし、内容へのコメントを求めることもできる（画面4）。

申請は、いわゆるワークフロー機能で、文書を申請経路に流し、承認を求めることができる。申請経路は、あらかじめユーザーがそれぞれで設定しておく。たとえば、係長→課長→部長という申請経路を登録し、この経路で文書を送ると、まず係長に送信される。係長が承認／却下の判断をし、承認すれば承認済み文書として課長に送られる。フローの途中で却下されるとそれ以上は送信されない。承認／却下の際にはコメントを付けることもできる。申請者は、文書が経路のどこで承認／却下されているのか、またそのときに付けられたコメントをいつでも読むことができる。

■電話帳

電話帳は3種類に分かれており、ユーザーが各自でデータを書き込む形式だ。［社員アドレス帳］が登録ユーザーのアドレス帳で、それ以外の人は［一般アドレス帳］に登録する。これら2つは全ユーザーで共有される。また、この中から任意のユーザーを選んで、ユーザーが各自で管理するのが［個人アドレス帳］だ。

■安価ながらワークフロー機能を実装

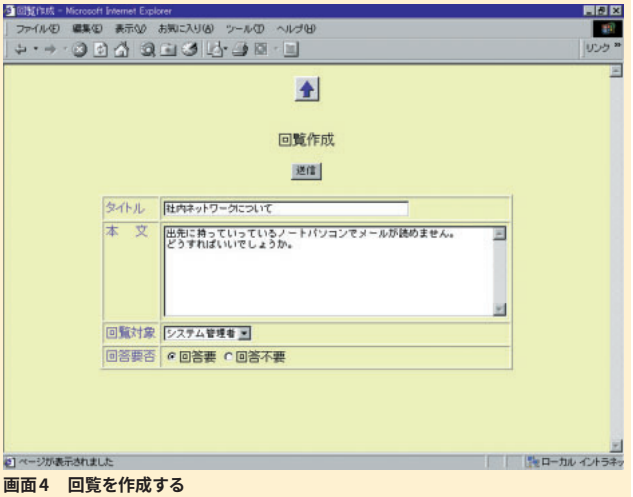
メーリングリストなどでは実現しにくい回覧の受領確認やワークフロー機能が便利だ。ユーザー向けのドキュメントが用意されていないのが残念だが、気軽に試せる無料ダウンロード版を使ってみるのもいいだろう。安価なライセンス料金設定と、動作の軽さは特筆すべきものだ。一般のメール利用環境を補完するビジネスツールとして活用してみてはどうだろうか。

導入のしやすさ	★★☆☆☆
管理のしやすさ	★★★★☆
サービスの使いやすさ	★★★☆☆
価格	★★★★★

ユーザー数		価格
期間契約の場合	100ユーザーまで／1年間	100USドル
数量契約の場合	1ユーザーごと	10USドル

価格表

画面3　連絡箱



画面4　回覧を作成する

L@Mail Wパック

アドバンスドソリューションズ

問合わせ先　アドバンスドソリューションズ
TEL：03-5543-6331
FAX：03-5543-6335
e-mail：sales@asi.co.jp
URL：http://www.asi.co.jp/

Point

- ●機能ごとにサーバマシンを分散可能
- ●60日間全機能を試せる試用版とデモサイトあり
- ●細かい管理設定が可能だが、わかりづらい

L@Mail（ラメール）は、アドバンスドソリューションズが販売しているWebグループウェアである（画面1）。それぞれのサービスごとに管理者やホスト指定が可能で、ユーザー登録もそれぞれごとに分けて登録できるため、細かい制御が可能である。ただし、現時点ではWebブラウザからのアクセスは完全なものではなく、専用クライアントのみ対応という機能もいくつかあった（近日中に対応がなされる予定）。

製品を購入する前に、20ユーザー版を60日間無償で試用できる。購入するとライセンスキーが発行され、利用期間の制限が解除される。また、アドバンスドソリューションズの公式Webサイトにはデモサイトも用意されている。

■インストール

サーバそのものの設定は、tar.gz形式のファイルを展開したあと、添付のインストールプログラム（シェルスクリプト）によってインストールを行うというオーソドックスなものである。

ただし、評価時点では、Webブラウザから環境設定を行うことができないため、「L@Mail登録プログラム」（画面2）という環境設定用の専用Windowsアプリケーションをインストールする。そのため、サーバマシン以外に環境設定を行うためのWindowsマシンが最低1台は必要になる。

■管理画面

前述のとおり、現段階では管理に関する設定は、Windows上で動作する専用クライアントから行わなければならない。さらに、細かいシステム管理ができるようになっている半面、設定しないといけない項目が多く、動き出すまでにかなり面倒な作業を強いられる。また、ユーザー登録は、ログイン名の設定場所が非常にわかりづらかった（システムが［メールアドレス］記入欄に書かれたメールアドレスから自動的にログイン名を特定する）。

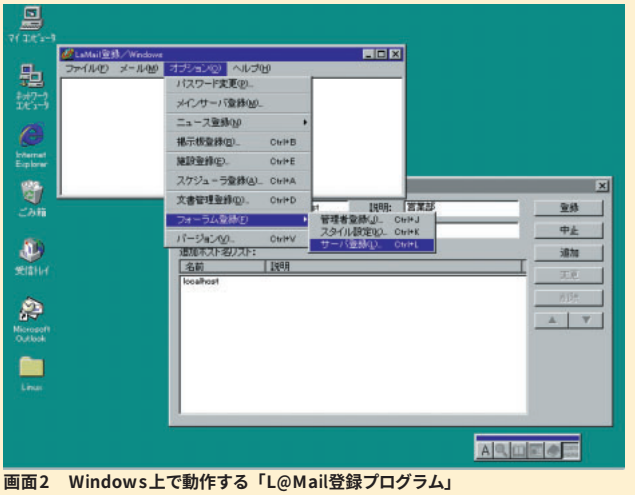
■標準機能

電子メール、スケジューラ、電子掲示板、共有アドレス、ToDoリスト、文書管理といった使用頻度の高い機能はほとんど備わっている。足りないと思われたのはメンバーの行き先を一覧する機能くらいである。

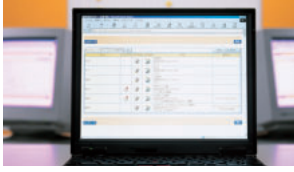
・電子メール

Webブラウザ経由で電子メールを読むことができる。通常のインターネットメールに加えて、「L@Mailメール」と呼ばれる、グループウェア内で使用するメールシステムも利用できる。このL@Mailメールを使うと、開封確認や送信メールに対しての回答要求も行える。添付ファイルも登録できるうえ、あらか

画面1　L@Mailのスタートアップ画面



画面2　Windows上で動作する「L@Mail登録プログラム」



じめ設定されたリストにしたがった回覧もできるので、簡易的なワークフローにも利用できる。

・ToDoリスト
　現在かかえている仕事や用事などを記録しておき、あとから未処理のタスクを確認できる機能（画面3）。期限設定や、優先度設定など基本的な機能は備わっている。

・スケジューラ
　スケジュールを登録しておき、グループ内で公開することでグループ内のスケジュールを一覧表示できる（画面4）。グループ内のメンバー全員の一括登録も可能である。後述の「施設予約」の機能とリンクしているため、スケジュールに沿った施設予約も行える。

・掲示板
　社内である特定のテーマを扱ったフォーラムを開設したり、回覧事項を掲示したりすることができる機能で、掲示した記事単位で、読んだユーザーやその時間を確認できる。

・文書管理
　ワープロ文書や表計算ファイルなどの文書を共有フォルダに保存し、全社レベルで公開したり、重要文書を解説つきで保存することができる機能。フォルダや文書ごとに個人、部署、役職別に細かいアクセス権を設定することができる。さらに、文書データを格納するフォルダも誰でも参照できる「公開フォルダ」と一部の人のみが参照できる「非公開フォルダ」に区分して保存ができる。ただ保存するだけでなく、次の3段階での版数管理も行え、簡単に呼び出せる。

・直近に更新された最新文書
・最新版を更新する直前の文書
・最初に文書登録されたオリジナル文書

　なお、登録された文書データは、文書名／登録者／登録日時／更新日時／文書種類／文書コメントなどの情報に基づき検索が可能である。ただし、現時点ではこの機能はWebブラウザから利用できないため、使用するには後日アップロードされる予定のモジュールを利用しなければならない。

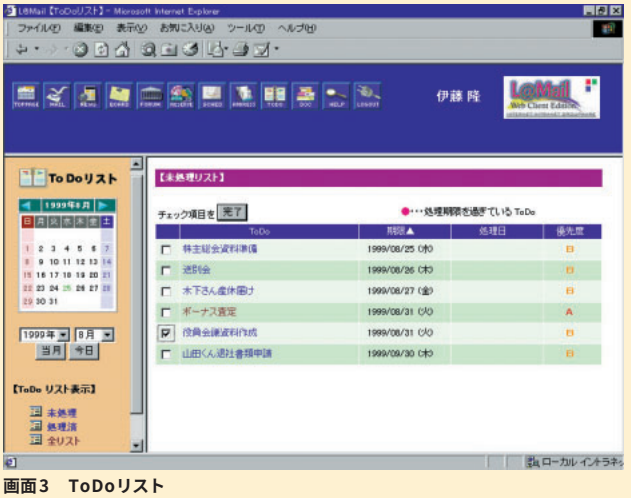
・施設予約
　会議室などの社内施設の予約／管理を行う機能。施設単位で公開／非公開の予約設定が行え、当日／毎週／連続といった予約が行える。また、予約情報を任意のユーザーへ通知することができる。

■中規模環境を考えた作り
　L@Mailは、機能ごとに管理者設定やユーザー登録できる管理の細かさから考えても、部門サーバーよりも大きな環境、たとえば全社規模のグループウェアとしての利用に向いているといえよう。ただし、そのような用途に使うにはCSV形式のユーザーデータのインポート／エクスポート機能がほしいところだ。

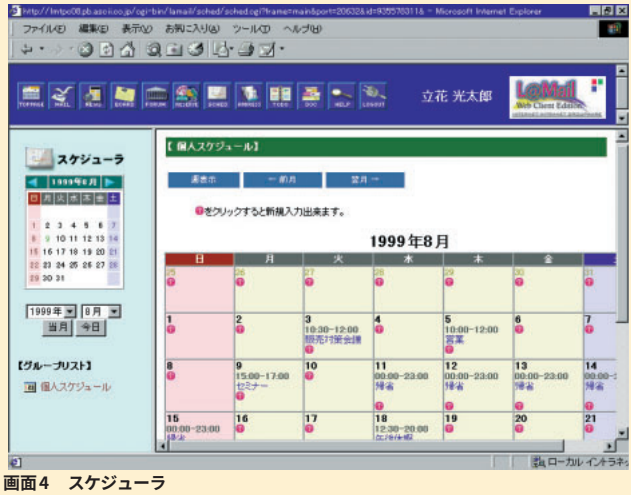
導入のしやすさ	★★★☆☆
管理のしやすさ	★★☆☆☆
サービスの使いやすさ	★★★☆☆
価格	★★★☆☆

ユーザー数	価格
20ユーザー	12万8000円
40ユーザー	19万8000円
100ユーザー	39万8000円
250ユーザー	79万8000円
250ユーザー以上	応相談

価格表



画面3　ToDoリスト



画面4　スケジューラ

ラ・クルールシリーズ

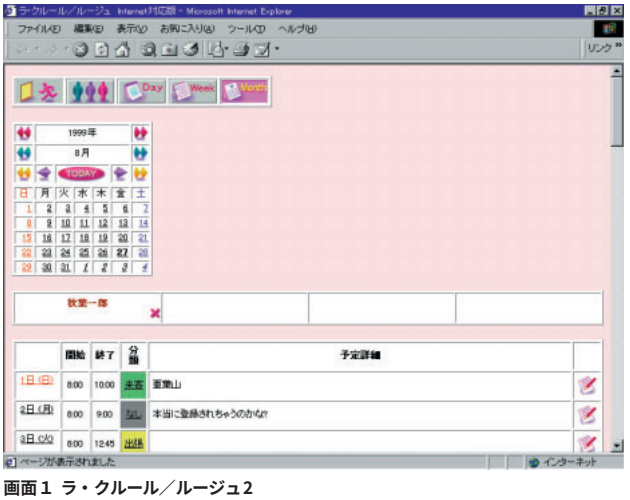
テンアートニ

問合わせ先　ネオジャパン
TEL：045-912-2145
FAX：045-912-5975
e-mail：ioffice@neo.co.jp
URL：http://www.neo.co.jp/ioffice/

Point

- 100%Pure Javaで開発されたユニークな製品
- 目的を絞って、高機能なソリューションを提供
- インストールが非常に難しい

ラ・クルールは、テンアートニが販売している製品である。Javaテクノロジーを利用しており、プラットフォームに依存しないWebグループウェア製品となっている。また、今回紹介している他の製品と異なり、機能ごとのコンポーネント単位のみという販売形態をとっている。



画面1 ラ・クルール／ルージュ2

製品名とそれぞれの機能は次のとおり。

ラ・クルール／ルージュ2（スケジュール管理）
ラ・クルール／ジョンヌ（勤務管理）
ラ・クルール／ヴェール（文書管理）

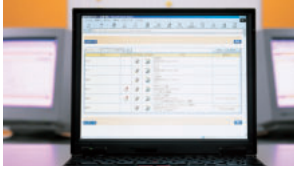
試用版は用意されていないが、「ラ・クルール／ルージュ2」は、テンアートニの公式Webサイトにデモサイトが用意されている。

■インストール

3製品とも「JRE」というJavaの実行環境を必要となるが、JREのインストールまではインストーラが面倒をみてくれないため、ユーザーが自力で行わなければならない。さらに、Javaサーブレットとして動作する「ラ・クルール/ルージュ2」にいたっては、サーブレットエンジンまでダウンロードしてインストールしなければならず、かなり負荷のかかる作業である。テクノロジー的な意味ではJava利用というスタンスは面白いと思うが、他の製品の導入の容易さと比べると、導入

機能	iOffi ce2000	イントラネット for Linux Standard版　Ver.2.0	オフィスケープ
スケジューラ	○（スケジュール）	○（ホワイトボード）	○（スケジュール）
掲示板	○（掲示板）	○（掲示板）	―
電子会議	○（電子会議室）	○（会議室）※1	○（掲示板）
施設予約	○（設備予約）	○（ホワイトボード）	○（施設予約）
Webメール	○（WebMail）	○（メール）	―
回覧	○（回覧版）	―	―
ワークフロー	○（ワークフロー）	―	―
アドレス帳	○（共有アドレス）	―	○（アドレス帳）
行き先掲示	○（伝言・所在）	―	○（行き先案内）
その他	仕事リスト タイムカード 文書管理	マイボックス	
備考		※1メーリングリスト可能	

各製品の機能一覧



にかかる手間は無視できないくらい大きい。

■ラ・クルール/ルージュ2

この製品は、スケジュール管理の機能を持つ（画面1）。前述のとおり、サーブレットやJavaアプレットとして動作する異色の製品である。スケージュール管理に特化した製品だけあって、予約時の通知メール自動送信や空き時間検索、さらに、iモード携帯電話やWorkPadからのアクセスサポートなど他の製品にはないさまざまな機能を持つ。

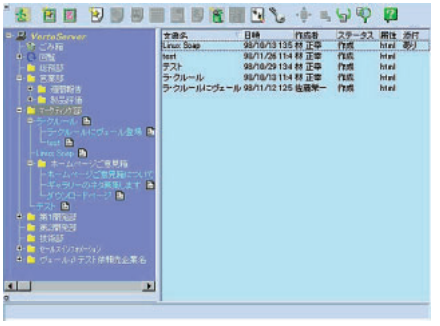
■ラ・クルール/ジョンヌ

この製品は、勤務管理の機能を持ち、Javaアプレットとして動作する（画面2）。ユーザーアイコンを用いた出社状況一覧やアイコンをクリックするだけの出退勤処理、出退勤データのプリントアウトなどの基本機能に加えて、誕生日をもとに計算されたバイオリズム表示機能などユニークな機能もある。

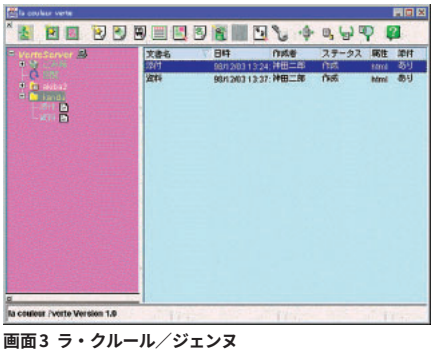
■ラ・クルール/ヴェール

この製品は、HTMLベースの文書管理の機能を持ち、Javaアプレットとして動作する（画面3）。文書をサーバ上で一元管理し、文書ごとにユーザーもしくはグループ単位のアクセス制御を行う。ツリー上にある文書の上にマウスポインタを置くと、文書内容が別ペインに表示される。更新通知も文書ごとに設定でき、文書情報の変化を確実に把握することができる。文書管理のデータベースにはOODB（オブジェクト指向データベース）を使用して、各文書の階層構造の表示にはXMLを用いるなど、将来を先取りした機能が用いられている。

どの製品も導入規模を問わず、非常に高いポテンシャルを持つだけに、インストールの敷居が高いのが残念である。インストーラを改良し、より導入しやすい製品となることに期待したい。



画面2 ラ・クルール／ヴェール



画面3 ラ・クルール／ジェンヌ

価格（1サーバ／クライアント無制）	
ラ・クルール/ルージュ2	9万8000円
ラ・クルール/ジョンヌ	4万9800円
ラ・クルール/ヴェール	19万8000円

価格表

CommunityBase	サイボウズ Office 2（Linux版）	BiSESSiON Lite for Linux	L@Mail Wパック
－	○ （スケジュール）	○ （スケジュール）	○ （スケジューラ）
○ （お知らせ）	○ （掲示板）	－	－
○ （会議室）※1	○ （電子会議室）	○ （掲示板）	○ （電子掲示板、フォーラム）
－	○ （施設予約）	－	○ （施設予約）
－	－	－	○ （電子メール）
－	－	○ （連絡箱）	－
－	－	○ （連絡箱）	－
－	○ （共有アドレス帳）	○ （電話帳）	－
－	○ （行き先案内板）	－	－
ライブラリ チャット	ToDoリスト プロジェクト管理		文書管理
※1メーリングリスト可能			※8月末時点でWebブラウザからのアクセスできる機能のみ

Webグループウェアの課題と展望

さて、Webグループウェアの製品概要はご理解いただけただろうか？　まだ製品としての歴史が浅いためか、Webグループウェアにはいくつかの課題がある。ここでは、それらを紹介し、対策を考えることで今後のWebグループウェア像を明らかにしていこう。

利用頻度の向上

ビジネス情報の共有の場であるWebグループウェアは、そこに利用すべき情報があって初めて真価を発揮できるものである。価値ある情報を蓄積していくためには、当然ながら参加メンバーが日常的に利用し、すべての情報をそこに保存するようにしなければならない。しかし、実際は導入した当初は珍しさも手伝って頻繁にアクセスされるが、そのうちに徐々にアクセスされなくなって、やがて使われなくなることも少なくないようである。その理由はいくつかある。

■情報ツールの一元化

メールのようなPush型の情報ツールと異なり、WebアクセスはPull型の情報ツールであるため、アクセスを忘れてしまったり、面倒になったりする。そのようなことを防ぐ意味でも、ツールは一元化されているほうが望ましい。たとえば、行き先連絡機能を利用していても、書かれている内容と現実が異なる（食事中と書いてあるのに、席にいる）と、情報の信頼性が薄れ、参照されなくなる。これを防ぐためには、必ずアクセスするツール、メーラとの連携を高めることである。メールを読まない人はおそらくいないと思うので、メールを読む目的でWebグループウェアにアクセスするようになれば、そのほかの機能にもアクセスする面倒は減り、利用頻度は高まるはずである。そういった意味では、現在のWebメールのさらなる強化が望まれる。

■トップダウン型の運用

今回紹介したWebグループウェアは導入が簡単であり、対象規模もあまり大規模なものではない部門レベルとしている製品が多い。しかも、機能的には製品版と変わらない試用版をダウンロードして、ある程度運用してみてから購入を検討できるので、小人数での同意を得るだけで、柔軟な導入が可能である。そういった意味では、これらの製品は「ボトムアップ」型の導入となるケースが多いだろう。この点は事前に入念な設計が必要な「トップダウン」型の導入に比べてメリットとも言えるが、こと運用に関しては、「トップダウン」型が必要になるケースもある。

つまり、メンバー任せの自由な書き込みだけではなく、仕事上の重要な情報をWebグループウェアを利用して告知する（たとえば、人事の回覧など）。そうすれば、否が応でもアクセスしなければなくなるし、メンバーのアクセス保証さえあれば、いろいろなサービスを提供できるはずである。

さらに、Webグループウェアを会社に合わせるべくカスタマイズをガリガリ行うのではなく、業務上における特殊性を洗い出し、可能であればそれらをWebグループウェアの形態に合うように変更していくような柔軟性が会社側にも必要になる。たとえば、スケジュール機能に書き込まれたものを正式な業務報告として認める、といったことである。いくらちゃんとスケジュール機能に自分のスケジュールを綿密に書き込んでいても、別に提出用の書類を手書きで作成しないといけないようでは、使う気が起こらないからである。

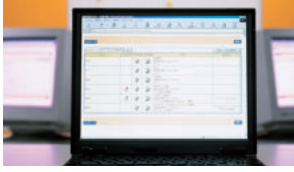
■ユーザーインターフェイスの向上

メンテナンスが面倒で、メンバーのプラットフォームごとに用意しておかなければならなかった専用クライアントと異なり、Webブラウザをクライアントとしているグループウェアにはそのような心配が一切ない。使い方の習得も簡単だ。さらに、セキュリティさえ配慮しておけば、外出先からのアクセスも難しくない。

しかし、Webブラウザを使うデメリットもある。ユーザーインターフェイスに難があるのだ。たとえば、スケジュールの日時指定をする際、いちいちプルダウンメニューから選ぶのではなく、カレンダーコントロールなどで選択できるほうが簡単だし、直観的でもある。今のインターフェイスではストレスがたまるし、数多くの登録をしないといけなくなると面倒になってくる。ユーザーインターフェイスが重要になる業務は専用クライアントを利用し、手軽なアクセスでいいものはWebブラウザからアクセスするというような使い分けが必要なのかもしれない。

より柔軟なクライアントへの対応

Webグループウェアの利用頻度が高くなってくると、参加メンバーの多様なニーズに対応する必要がある。今後、クライアントとして対応が求められるであろうものに次の2つがある。



■iモード携帯電話

Webブラウザによるアクセスがもたらしたメリットについては前述したとおりだが、Webグループウェアの使用頻度が高まってくると、外出先からもアクセスしたいという要求が出てくるだろう。しかし、いくらWebブラウザがどんなプラットフォームにも搭載されているといっても、ノートパソコンを持ち歩かないと、外出先ではWebブラウザを使うことができない。これはちょっと不便である。小さくなったといえ、やはりノートパソコンは軽くないし、大きい。バッテリ時間の問題もある。

そこで、外出先からのクライアントとして有力な候補といえるのが、「iモード携帯電話」である。iモード携帯電話なら持ち運びに不便はないし、HTMLの表示も可能である。もちろん、iモード携帯電話が表示できるHTMLはフルスペックのものではないため、有効に活用するには専用のページを作成しないといけないが、簡単なデータの参照程度なら十分実用的なはずである。今回紹介したWebグループウェア製品の中では、「ラ・クルール／ルージュ2」がiモード携帯電話からのアクセスに対応しており、「iOffice2000」もiモード対応版を公開している。

■PDA

ザウルスやWorkPadといったPDAで個人のスケジュール管理をしている読者も多いことだろう。ここに入力したデータをWebグループウェアと相互活用できないと、データの二重管理を強いられてしまうため、利用するのが面倒になってしまう。したがって、これらの携帯端末のサポートもこれから求められる機能といえよう。「ラ・クルール／ルージュ2」は、今回紹介した製品の中では唯一、WorkPadからのアクセスに対応しているが、スケジュールの同期機能までは実装されていない。より一歩進めて、このような機能の実装を期待したい。

より大規模な環境への対応

部門レベルで弾力的に導入／運用できることは、Webグループウェアの利点であるということは前述したとおりだが、会社によっては全社的な導入／運用が必要なケースもある。そのため、より大規模な環境へWebグループウェアを導入／運用していくには、次のような要件を満たす必要があるだろう。

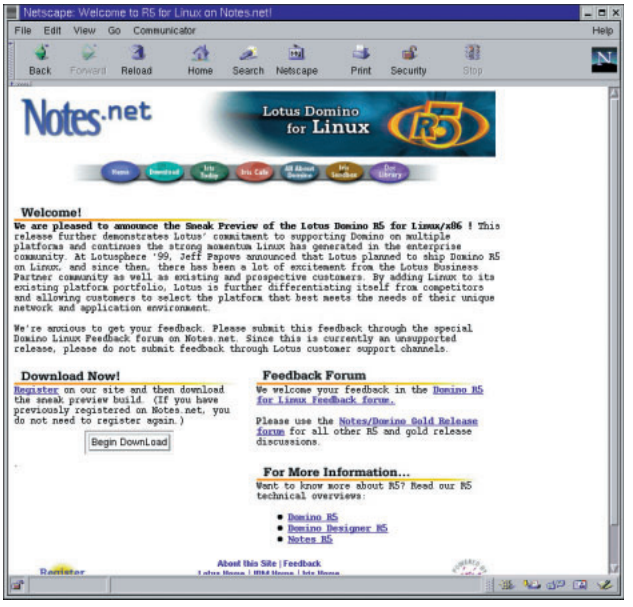
- セキュリティの強化
- 柔軟なカスタマイズ
- きめ細かいバックアップ機能のサポート
- クラスタリング対応
- SIベンダーの育成

そして、先日「Sneak Preview」版を公開したLotusの「Lotus Domino R5 for Linux」（http://www.notes.com/notes/）の一刻も早い製品化が望まれるところだ（画面1）。いうまでもなく、Lotus Domino/NotesはWindows市場で「グループウェア」というアプリケーションを定着させた製品であり、国内では高いシェアを持つ製品である。大規模な環境への新規導入はもとより、堅牢さやコストパフォーマンスを考え、既存のWindows NT上に構築されたシステムの置き換え需要も出てくることが予想される。このような製品が出てくれば、現状のLinuxのビジネスに対するかかわり方にも大きな影響を与えるだろう。

Linuxビジネスソリューションのトップバッター

Linuxは、なかなかビジネス市場での利用の具体的なプランが見えてこないため、Linuxの主な用途はインストールとさえ揶揄する人さえいる。そういう意味では、WebグループウェアはLinuxビジネスソリューションの到来を示すトップバッターと位置づけることができるだろう。

また、このWebグループウェアは、仕事環境を便利にするソリューションであるだけでなく、ビジネスロジックそのものに新しいパラダイムをもたらす可能性さえ秘めたものである。まさに、Linuxのビジネス市場での新境地を切り拓く製品でもある。これからの普及に期待したい。



画面1　「Lotus Domino R5 for Linux」のダウンロードサイト

Silicon Graphics 320でLinux

SGIのVisual Workstationで Red Hat Linux 6.0を動かしてみた！

文：田中一郎
Text:Ichiro Tanaka

8月9日から12日にかけて、カルフォルニアのサンノゼではLinux World Expoが開催されていた。SGIは展示会の初日にあたる8月10日に大幅な事業再編と、数日前に発表したインテルベースのサーバであるSGI 1400Lを軸にしたLinuxビジネスへの傾倒を明らかにした。

SGI 1400はサーバOSとしてSGI Linux Environment with Red Hat 6.0、またはマイクロソフトのWindows NT 4.0 Serverを採用した同社初の非MIPSサーバであり、これは今後のSGIの動向を示唆しているマシンでもある。

写真1　Visual Workstationは、Pentium IIIを搭載したWindows NTマシンだ。しかし、専用チップセットにより驚異的なグラフィックス性能を誇っている。

一方、LinuxWorld Expoの展示会場においてSGIは興味深い製品を参考出品した。OpenGL/Linuxと呼ばれるOpenGLのハードウェアサポートによるLinuxへの実装で、プラットフォームとしては同社のSilicon Graphics 320上で動作していた。320はいわずと知れた同社初のインテルベースのNTワークステーションであり、Linuxすら正式サポートされていない。そんなマシンにOpenGLを実装してみせるあたりに、SGIの今後のLinuxへの取り組みを感じさせるような展示であった。

SGI 320 Visual Workstation でLinuxを動かしてみる

前号ではSGIの少し古めのグラフィックスワークステーションのIndy（インディ）に、Red Hat系のLinuxであるHard Hat 5.1をインストールしてみた。

今回は、今年の1月にSGIが満を持して発表した同社初のインテルベースのNTマシンであるSilicon Graphics 320（以下SGI 320）に、Red Hat 6.0ベースのLinuxをインストールしてみたい。

LinuxをSGI 320で稼働させるためには、いくつかのハードルが存在する。まず、基本的にSGI 320はAT互換機ではないためIBM BIOSを持たない。また、SGIが開発したCobaltというチップセットを使用していて、メモリ、グラフィックスコントール、SMPなどのコントロールもAT互換機とは違う。

カーネルに関しては、SGI自身によってすでに開発され、コントリビュートされており、その他LILOに相当するブートローダなどのファームウェアもSGIのWebサイト（http://www.linux.sgi.com/intel/）で入手可能である。

CPU	Pentium III 450MHz～500MHz、2CPUモデル有
メモリ	128Mバイト～1Gバイト（ECC SDRAM）
ハードディスク	6.4Gバイト（UltraATA）～9.1Gバイト（Ultra2 SCSI）
CD-ROM	32倍速
フロッピードライブ	3.5インチ1.44Mバイト
キーボード	USB 日本語キーボード
マウス	USB 3ボタンマウス
OS	WindowsNT 4.0 Workstation日本語版
価格	81万6000円～162万6000円

表1　Silicon Graphics 320ハードウェアスペック

ただし、忘れてはならないのは、SGIのWebサイトから入手できるにしても、現在のバージョンはあくまで個人の責任において使用可能なソフトウェアあり、SGIから正式にリリースされている製品ではないという点である。ユーザーによる各種のバグレポートやコードコントリビューションなどは歓迎してくれるかもしれないが、クレームは受け付けられないということである。今回インストールしたところ、大きな問題点は見受けられなかったが、まだ完成度の高いベータ版だという認識は必要であろう。

事前に準備するもの Step 1

1. SGI 320 Visual Workstation

もちろん、今回の素材のメインである。現在ではSilicon Graphics 320に加えて540でも動作するようである。

SGI Linuxが動くために必要なシステム構成は、**表2**のようになる。インストールするハードディスクは、Linux専用に1台用意した方が安全だ。

2. ブランクのフロッピーディスク4枚

3. Linuxマシン

あらかじめ、フロッピーディスクにWebサイトからダウンロードしたデータを書き込んでおくために必要。

4. Red Hat Linux 6.0のバイナリCD-ROM（製品版、またはLinux magazine No.2付録CD-ROMでもよい）

ハードディスク	少なくとも1つのIDEディスクか純正のSCSIボードとSCSIディスク
CD-ROM	IDEのCD-ROMドライブ（通常はシステムに内蔵）
フロッピードライブ	本体内蔵のフロッピードライブを使用
キーボード、マウス	USBキーボードとUSBマウス（シリアルのPS/2キーボード/マウスは不可）
ディスプレイ	CRTまたはフラットパネルディスプレイ（SGI 1600SW）

表2　SGI 320でSGI Linuxが動くためのシステム構成

▼リスト1　用意するフロッピーディスクに書き込む内容

```
■FD-1 （FATフォーマット）

      2.2.10カーネル  Size: 1359183 bytes. Updated: July 28, 1999.
      http://www.linux.sgi.com/intel/visws/la2210.vw

      ARCローダー     Size: 46638 bytes. Updated: July 21, 1999
      http://www.linux.sgi.com/intel/visws/arclx.exe

■FD-2（rawフォーマット）

      インストーライメージ
      http://www.linux.sgi.com/intel/visws/rh60.initrd.img

■FD-3（rawフォーマット）

      レスキューディスクイメージ
      http://www.linux.sgi.com/intel/visws/rh60.rescue.img

■FD-4 （FATフォーマット）

      FATのブートパーティション用のARCローダーとカーネル、XFree86用のコンフィグレーションファイル
      http://www.linux.sgi.com/intel/visws/f4jul28.tgz
      （ダウンロードしたデータを解凍しないでそのままコピーする）

      このtar イメージには次の5ファイルが含まれている。

      $ tar zvtf f4jul28.tgz
      drwxrwxr-x bh/bh         0 1999-07-28 09:31 f4/
      -rw-r--r-- bh/bh      1201 1999-06-07 10:51 f4/XF86Config.crt
      -rwxr-xr-x bh/bh      1204 1999-06-07 10:51 f4/XF86Config.fp
      -rwxr-xr-x bh/bh     46638 1999-07-21 08:17 f4/arclx.exe
      -rw-r--r-- bh/bh       353 1999-06-07 10:51 f4/fb.modes
      -rwxr-xr-x bh/bh   1359183 1999-07-28 09:31 f4/la2210.vw
```

■インストール前の作業

フロッピーディスクに書き込むデータの入手先は**リスト1**にまとめてあるので、Webサイトからダウンロードしておく。

4枚のフロッピーディスクを、Linuxのfdformatでフォーマットし、1枚目（FD-1）と4枚目（FD-4)は、mformatでDOSのFATフォーマットを行う。

1枚目（FD-1）と4枚目（FD-4）は、下記のようにマウントしてから書き込む。コピーが終了したらアンマウントする。

```
# mount -t msdos /dev/fd0 /mnt/floppy
# cp <ファイル名> /mnt/floppy
# umount /mnt/floppy
```

2枚目（FD-2）と3枚目（FD-3）のフロッピーディスクには、

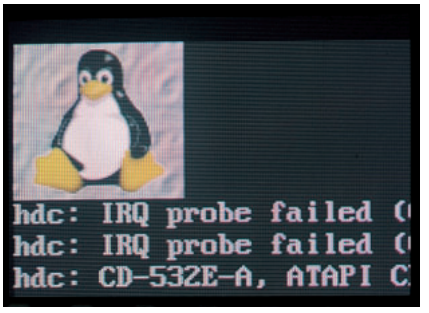
```
# cp rh60.initrd.img /dev/fd0
```

または、

```
# dd if=rh60.initrd.img of=/dev/fd0
```

画面1　Silicon GraphicsのPROMモニタ画面。AT互換機でいうBIOSのようなものだ。

画面2　Start Settingsメニューでは、ブートセッティングを細かく指定できる。



画面3　かわいいペンギン（Tux）のイメージを表示しながらLinuxをブートする。

のように行う。3枚目はファイル名rh60.initrd.imgをrh60.rescue.imgに置き換える。

Webサイトには、さまざまなバージョンが置いてあるが、必ずしも最新のものが良いとは限らないので注意が必要である。今回は、ブートローダはにarclx.exe（Jul.28）ではなくlarc.exe（Jul.9）を使用した。SGI 320用のカーネルla2210.exeとARCローダの2つのファイルを、DOSまたはWindowsでFATフォーマットしたブートフロッピー（FD-1）にコピーしておく。

なお、ここで紹介しているのはシングルプロセッサ仕様のカーネルである。これを使うとマルチプロセッサマシンでも1つのCPUしか稼働しないことに注意していただきたい。

最後に、これらのフロッピーディスクのライトプロテクトを忘れずに行うこと。さらに、インストールする前に、ターゲットのハードディスクのフォーマットを、他のLinuxやWindowsマシンで行っておくことをお勧めする。詳しくは後述するが、現在SGI 320/540でLinuxを起動する場合、最低10Mバイトほどの FATパーティションが必要になる。実際にインストールが始まってからでは面倒な場合もあるので、あらかじめFAT16のパーティションを用意しておくのが賢明である。

Red Hat 6.0のインストール Step 2

最初に、SGI 320をPROMモニタ画面にする。SGI 320では、マルチOSのブートセレクト画面がGUI化されており、各項目に記述するだけで起動OSを選ぶことが可能である。PROMモニタ画面にはリセットキーを押すか電源投入後、ESCキーを押すといくつかのメニューを表示する（**画面1**）。

そこで、Startup Settingsメニューを選び、フロッピーブートに設定する。GUI化された設定項目をそれぞれ**リスト2**のように指定する。まず“New”を選ぶ。ここで他の環境を“Copy”して編集してはならない。設定が終わ

パーティション	容量	内容
/dev/hda1	500Mバイト	“Linux native”ルートパーティション
/dev/hda2	128Mバイト	“Linux swap”パーティション
/dev/hda3	10Mバイト	FAT フォーマットのブートパーティション
/dev/hda4	残った領域	“Linux native”ユーザーパーティション（/usr）

表3　Linuxパーティションの分け方

接続先	SGIのデバイス名	Linuxのデバイス名
IDE bus 0 master	multi(0)disk(0)	/dev/hda
IDE bus 0 slave	multi(0)disk(1)	/dev/hdb
IDE bus 1 master	multi(0)disk(2)	/dev/hdc
IDE bus 1 slave	multi(0)disk(3)	/dev/hdd

表4　デバイスの対応表

▼リスト2　インストールフロッピーからの起動設定（Startup Settings）

```
Load Identifier          Linux RX                  この項目は任意の名称でも可
OSLoader                 larc.exe                  ARCローダーのファイル名
OSLoadFilename           /la2210.vw                ブートカーネルのファイル名
OSLoadPartition          Floppy1                   ブートするデバイス名
System Partition                                   空白でよい
OSLoadOptions            root=/dev/fd0 load_ramdisk=1   ブートローダに渡すメッセージ
```

画面4　1枚目のインストール用フロッピーディスクでブートしたところ。

画面5　SGI 320でRed Hat Linux 6.0のインストール画面が表示された。

ったらBoot Selectionを“Default”として選ぶ。これを忘れるとこの設定で起動しないので注意する。最後に“Hit Save&Exit”ボタンをマウスでクリックし環境を保存する。

■ フロッピーブートでインストール開始

まずFD-1をフロッピードライブに入れる。PROMモニタよりStart Systemを選ぶ。しばらくするとIndyなどで見慣れたコンソール画面が現れ、ARC Linux loaderが起動する。

数十秒後、**画面3**のように、左上に小さなペンギン（Tux）のイメージを表示しながらカーネルのブートが行われる。最後のメッセージが“USB Keyboard found.”と表示されて止まると1枚目は終了である（**画面4**）。

次に、FD-2をフロッピードライブに入れ、Red Hatインストーラを起動する。

しばらく立つと見慣れた“Welcome to Red Hat Linux”のモノクロのインストール画面が現れ、この時にRed Hat 6.0 CDを入れるか、すでにドライブに入れてある場合はブルーの画面と、Red Hat Linuxのインストール初期画面を表示する。これで通常のRed Hat Linuxインストールが開始される（**画面5**）。

SGI 320にはBIOSがないのでインストール中にLIIOについての問い合わせはすべて“No”と答えること。LILOそのものはインストールすることは可能だが、まったく使い道がないのである。

■ パーティションのセットアップ

SGI320の出荷時にはすでにWindows NTがインストールされている。したがってパーティションを切り直すか新しいディスクを用意してインストールするとよいだろう。今回は、新しいディスクを用意し、パーティションの分け方は**表3**のように行った。

出荷時の構成の場合、システムドライブはhdaでありCD-ROMはhdcとなる（**表4**）。ディスクのパーティションを作成するにはDisk DruidまたはfdiskをLinux使用する。パーティション作成が終わればルートパーティションを指定し、スワップ領域を初期化する。

その後、インストールするパッケージを選びインストールを開始するのであるが、この時、選択するパッケージは極力、最小構成にする方がよい。一度システムが起動してしまえば残りは追加インストールすればよいからである。ただし、X関係は最初の時に選んでおいた方がいい。

パッケージのインストールが終了した時点では、まだUSBマウスは認識されない。しかし、ここで決してシリアルマウスを選んではいけない。マウス関係は後のXの修正時に設定することになる。

インストールでは、いろいろな設定を行うが、下記の部分はSGI 320固有のものではないので割愛する。

- **Network Configuration**
- **Configure Timezones**
- **Services**
- **Configure Printer**
- **Root Password**
- **Authentication Configuration**

ここからは**リスト3**のように選択する。これでインストールの設定は終了である。

▼リスト3　Red Hat Linux 6.0インストール時に選択する項目

```
Bootdisk                  “No”
Lilo Installation         “Skip”
Choose a Card             “Unlisted Card”      ← 最後尾にある
Pick a Server             “Mono”
Monitor Setup             “Sony GDM-17SE2T(NEW)”
Screen Configuration      “Don't Probe”
Video Memory              “256kb”              ← もともと存在しないので適当に答える
Clockchip Configuration   “No Clockchip Setting”
Probe for Clocks          “Skip”
Starting X                “Skip”
```

インストール後の設定 Step 3

基本的なインストールを終えただけでは起動しないので、いくつか追加設定を行う。

■ Linux RX kernelのブート設定

最初に作成したブートフロッピー（FD-1）をフロッピードライブに入れ、Start Systemで起動する。次に、FD-

▼リスト2　ブートローダをFATファイルパーティションに書き込む。

```
bash# /sbin/mount /dev/cdrom /mnt/cdrom
bash# rpm -i /mnt/cdrom/RedHat/RPMS/mkdosfs*.rpm
bash# /sbin/umount /dev/cdrom
bash# /sbin/mkfs -t msdos /dev/hda3
bash# mkdir /b
bash# /sbin/mount -t msdos /dev/hda3 /b
bash# cp larc.exe /b
bash# /sbin/umount /b
```

3に入れ換えてEnterキーを押すと、**RAMDISK: Compressed image found at block 0**

というメッセージとともに、約30秒でRAMDISKベースのカーネルが起動する。

■ rootドライブをマウント

次の手順で、現在のRAMDISK rootから本来のrootドライブに切り換える。インストール時にパーティションを設定したとき、'/'に設定したディスクを指定すること。

```
# mkdir /r
# mount /dev/hda3 /r
# chroot /r
```

この時点で本来のrootファイルシステムが立ち上がり、プロンプトが“ bash# ”に変わる。

■ システムの最終設定

FD-4をドライブに入れ、フロッピーディスクをシステムにマウントし、フロッピーディスクの中のファイルをtarで展開する。

```
bash# mount -t msdos /dev/fd0 /mnt/floppy
bash# ls /mnt/floppy
bash# cd /tmp
bash# tar zxvf/mnt/floppy/f4jul28.tgz
```

ここにはFATのブートパーティション用のARCローダとカーネル、さらにXFree86用のコンフィグレーションファイルが入っている。

■ ブートローダのインストール

FATフォーマットのブートローダを格納するパーティションに、ブートシステム（larc.exe）を格納する作業を行うが、そのマウントのファイル名を/boot という名前にしてはいけない。Red Hatがすでに/bootディレクトリを使用しているからである。ここでは“ /b ”という名称にする。

まず、FATのパーティションを作成するために、Red Hat 6.0のバイナリCD-ROMからmkfsをインストールする。次に、FATのブート用パーティションの中にブート用のファイルを格納するディレクトリを作成する（**リスト4**）。

■ SGI 320用カーネルのインストール

generic kernel（/tmp/f4 に展開したもの）を、ルートディレクトリにコピーする。カーネルのアップデートがあった場合には、このファイル（/la2210.vw）を入れ換えればよい。さらに、フレームバッファの設定ファイルを、/etcにコピーする。

```
bash# cp la2210.vw /
bash# cp fb.modes /etc/fb.modes
```

■ Xサーバの設定

XFree86のコンフィグレーションファイル（XF86Config）は、CRTとフラットパネル用の2つが用意されているので、使用するディスプレイに応じてコピーする。

```
bash# cp XF86Config.crt /etc/X11/XF86Config.vw （CRTの場合）
bash# cp XF86Config.fp /etc/X11/XF86Config.vw （SGI 1600SWの場合）
```

USBキーボードとマウスを使用す

▼リスト5　インストールしたLinuxの起動設定（Startup Settings）

Load Identifier	Red Hat 6.0+Linux 2.2.10	この項目は任意の名称でも可
OSLoader	larc.exe	ARCローダーのファイル名
OSLoadFilename	/la2210.vw	ブートカーネルのファイル名
OSLoadPartition	IDE 0 Disk 0 Partition 3	ブートするデバイス名
System Partition		空白でよい
OSLoadOptions	root=/dev/hda1	ブートローダに渡すメッセージ

▼リスト6　X Windows Systemのセットアップ

```
# mount /dev/cdrom /mnt/cdrom
# rpm -e XFree86-Mono
# rm /etc/X11/X
# rpm -i /mnt/cdrom/RedHat/RPMS/XFree86-FBDev-3.3.3.1-49.i386.rpm
# ln -s ../../usr/X11R6/bin/XF86_FBDev /etc/X11/X
# cp /etc/X11/XF86Config.vw /etc/X11/XF86Config
```

るので、USBデバイス（HID - Human Interface Device）を設定する。

```
bash# mknod /dev/hidbp-mse-0 c 10 32
```

以上で、Linuxの設定は終了した。フロッピーディスク、ハードディスクをアンマウントする。

```
bash# cd /
bash# umount /dev/fd0
bash# exit
# umount /r
```

この後、リセットし、リブート後にESCキーを押しPROMモニタにする。

Linuxのブート設定 Step 4

PROMモニタで、Startup Settingsメニューを選び、ハードディスクからブートするように設定する。設定項目は**リスト5**のように指定する。

OSLoadPartition項目で、起動ドライブがIDEでパーティション3なら、“IDE 0 Disk 0 Partition 3”、SCSIなら“SCSI 2 Disk 0 Partition 3”）を選択する。

OSLoadOptions 項目では、実際のブート時に使用されるルートパーティションディスクを記述する。

このカーネルはメモリの割り当てを自動的には行わない。デフォルト指定では“128MB”に設定されている。128Mバイトを越えるメモリを実装している場合、メモリの容量を“mem=xxxM”と明記しなくてはならない。もし512Mバイト搭載している場合、OSLoadOptions項目に“root=/dev/hda1 mem=512M”と記述する。

画面6　インストール完了! X Window SystemでGNOMEを起動したところ。

■ X Window Systemのセットアップ

PROMモニタからStart Systemを選ぶとLinuxが起動するが、まだXの設定は行われていない。インストール時に“Mono”Xサーバを選んだのを覚えているだろうか。あれは後で入れ換えるために仮に入れたものなので、ここで削除し、Red Hat Linux 6.0 CD-ROMから正しいXの環境をインストールする**（リスト6）**。

startxコマンドで、X Window Systemが起動すれば成功である。

今後の各種アップデートによってシステムをさらに強化することを期待している!

Column

Linux/OpenGL

Linux World ExpoがSan Joseで開催されていたころ、まったく同じ日程でコンピュータグラフィックスの夏の恒例行事とも言うべきACM/SIGGRAPH'99がLos Angelsのコンベンションセンターで開かれていた。そこでも（実は、SGIはこちらにグラフィックスの主力部隊を送り込んでいた）SGI 320上で稼動するLinux/OpenGLが展示されていて、さらに突貫作業でとりあえず動かしたというIRIS Performerが参考出品され、大きな注目を集めていた。

IRIS Performerとは軍事用や航空機のフライトシミレータなどの、リアルタイムのグラフィックス分野で標準的に使用されている、いわばSGIの虎の子ともいえるグラフィックスライブラリである。

また、最終日の12日にはウエスティン・ボナベンチャーホテルにてLinux/OpenGLのBOF（Birds of a Feather: 類は友を呼ぶ）が開かれた。SGIの創業者の一人であり、GLおよびOpenGLの設計者でもあるKart Akeleyが、SGIによる今後のLinuxへの強力なコミットを行うとコメントした。XFree86の開発を行っているPrecision Insight社のDaryll Straussは、ハードウェアサポートによる完全なOpenGL環境の構築について、既存の3DfxやnVIDIA、3Dlabsといった主要な3Dグラフィックスボードをサポートしていくことを強調していた。

SGIは長らくグラフィックスのトップベンダーとして独自のIRIXというUNIXをメインにしてきた会社であるが、この半年間で突如、UNIX系のベンダーとしてはもっとも積極的にLinuxに取り組みだした。

前述のOpenGLのLinuxへの移植以外にも、自社のIRIXのファイルシステムであるXFSをはじめ多くのソフトウェアを精力的に移植中のようである。SGIのような数多くのUNIX資産のある企業が、本格的にLinuxに取り組み出したということは、Linuxも新しいステージに突入したと考えてもよいのではないだろうか。

5万円で作るLinuxマシン

ふだん使っているWindowsマシンにLinuxをインストールし、デュアルブートでOSを切り換えるのはいろいろ不便だ。できればLinux専用マシンがほしい。でもこのご時世、できるだけお金はかけたくない。出しても5万円まで。そこで、セカンドマシンを5万円で作れるか、秋葉原へやってきた。

文：上間俊雄
Text：Toshio Uema

5万円でマシンは作れるか

現在では、無料のPCまで登場しているが、つい昨年まで、アメリカでは「1000ドルPC」という言葉が騒がれていた。これは、その名の通り、1000ドルで購入できるPCのことである。最近ではホワイトボックスと呼ばれる、PCショップが組み立てた自作AT互換機が500ドルPCや300ドルPCとして販売されている。日本でも、Windows 95が発売される前後から、ショップブランドの10万円を切るマシンが登場している。しかし、その頃の10万円マシンは、メモリやハードディスクの容量が少なく、かなり割り切ったマシンだった。最近では、Windows 98プレインストール、15インチディスプレイ込みで、10万円を切るマシンがかなり売れているようだ。

もう、耳にタコができるほど聞いただろうが、技術革新と価格競争でマシンもパーツ単位の価格がかなり下落している。たとえば、300MHzを超えるクロックで動作するCPUであるCeleronやK6-2、128Mバイトのメモリ、4Gバイトのハードディスクなど、少し前なら想像もできないほどの性能を持ったパーツが、それぞれたった1万円ぐらいで買えるようになった。

そんな7月末のある日、Linux magazine編集部から「5万円でマシン組み立てられるか?」と持ちかけられた。そのときは、気軽に「割り切ればできますよ」と、以前あった10万円PCのような気持ちで答えたのだが、これが運のつき。「それでは週末に秋葉原で待ち合わせね」と言われたので、しぶしぶ秋葉原に出頭した。秋葉原に着くと、その場で5万円手渡され、「TurboLinux 日本語版4.0が動くマシンを買ってね」と頼まれた。

パーツ	金額（円）
CPU	9000
マザーボード	1万1000
グラフィックスカード	オンボードVGA
メモリ	6000
ハードディスク	1万1000
CD-ROMドライブ	4000
ネットワークカード	1000
キーボード	2000
マウス	1000
ケース	5000
合計	5万

表1：購入するパーツ予算

Linuxサーバにフロッピードライブはもう要らない

5万円のマシンであっても、最低限WindowsマシンからLANでつながるファイルサーバを構築しよう、できればX Windows Systemがきちんと動くクライアントとしても使いたいというのが目標だ。そのためにパーツ単位で何が必要かと考えるのだが、5万円という予算は結構きつい。秋葉原に行く前に、Webで価格を調べて立てた予算の配分は**表1**にまとめたので、そちらをご覧いただきたい。

このリストを見てお気づきのかたもいらっしゃるだろうが、フロッピードライブは項目に入れていない。なぜなら、もうフロッピーディスクを使用する機会はないと考えたからである。あったとしても、最初のインストールのときに使用するくらいである。それで

は、そのインストールのときにはどうするのかという疑問があるだろう。これは、CD-ROMドライブからブートして、インストーラを起動すれば解決できるのだ。先月号に付属していたTurboLinux 日本語版4.0もCD-ROMドライブからブートできるので、フロッピードライブは思い切って外してしまった。iMacでもフロッピードライブは搭載されていないし、B5サイズのノートPCに、通常フロッピードライブを付けないで利用していることを考えれば、セカンドマシンとしてはフロッピードライブは必要ないだろう。フロッピーディスクを利用するなら、Linuxマシンで読み書きせずにメインのWindows 98のマシンで利用するはずである。データのやり取りにはネットワークを利用したい。もし、Linuxマシンの調子が悪くなって、フロッピーからの起動が必要となったら、そのときにフロッピードライブを購入するか、一時的にWindowsマシンから借用すればよい。

ほかにもディスプレイなども必要なのだが、Linuxでは主にサーバとして利用する理由からリストには入っていない。

動けば何でもいいパーツ

さて、5万円でマシンを組み立てるには、何かを割り切らなければならない。はっきり言えば、動けば何でもいいというパーツを購入しなければならないのである。今回は、動けば何でもいいパーツとして、キーボードとマウスを挙げた。ほかにもインストール時にしか使用しないCD-ROMドライブも動けばいいことにした。たぶん、何倍速のを買っても値段はそれほどかわらないだろうが、少しでも安く済ませるためだ。ここでは「せっかくだから」は禁物である。また中古品やジャンクを購入するという手もあるのだが、新品でもLinuxの動作保証をしているものがほとんどないという状況の中で、そういった品物が動作しなかった時に途方にくれてしまうので、安全を考えてすべて新品を購入することにした。それでも秋葉原で、土曜から日曜によく出る特価品を見つけ出さないと5万円の予算では収まらない。

CPUとマザーボード

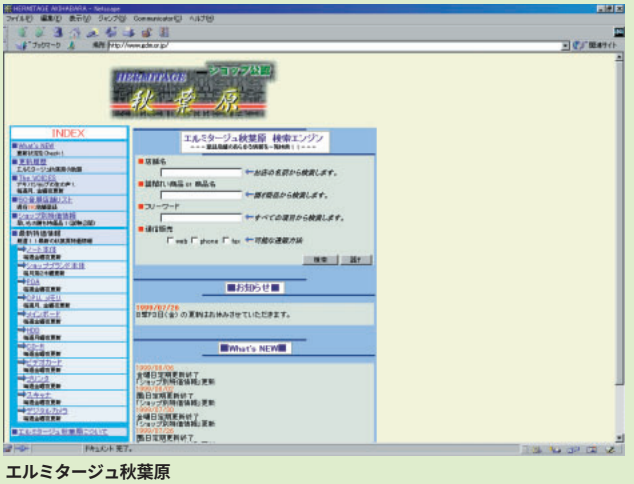
今回は、5万円の制約から必然的に低価格のCPUを選択することになるので、入手のしやすさや、コストパフォーマンスからCeleronを購入することにした。秋葉原に買い出しにきたときは、ちょうどCeleron 500MHzの発売の時期であった。しかし、計算では、

Column

手早く買うための秘訣

秋葉原は広い。100店はあろうかというPCパーツのショップを回るだけで、ゆうに1日はかかってしまう。ましてや、5万円でマシンを作るとなると、秋葉原で1番安いパーツをそろえなければならない。これを秋葉原に行ってから見つけ出すことははっきり言って難しい。そこで、秋葉原を効率よく回るために、Webを利用して事前に購入するものを探しておく。DOS/V ISSUEでもおなじみの「サハロフの秋葉原レポート」や、「エルミタージュ秋葉原」などがパーツ購入の参考になるはずだ。また、ハードディスクひとつ取っても、4Gバイトのハードディスクでも何種類とあるので、できればこれらのページで具体的な品名と型番を決めておけば購入しやすい。

サハロフの秋葉原レポート
http://www2s.biglobe.ne.jp/~sakharov/



エルミタージュ秋葉原
http://www.gdm.or.jp/

CPUにかけられる価格はどう見積もっても1万円以下で、最新のCPUは購入できない。さて、1万円以下で入手しやすいCeleronは、Celeron 300A、333、366の3種類だ。おかしなことに、この3種類のCPUのどれを購入しても、値段が同じという現象がどこのショップでも見られた。さらには、クロックアップがしやすいCeleron 300Aのほうが若干割高というショップも多くあった。筆者は、リスクのある自己満足程度のクロックアップには反対なので、当然ながらPPGA（Socket370タイプ）のCeleron 366を選択した。

PPGAのCeleronには、Intelの箱に入ったリテール版と、バルクと呼ばれる簡易包装のものの2種類がある。リテール版とバルクの違いは、リテール版にはCPUクーラーが同梱されており、バルクには付属していないという点にある。保証に関してはたいていどこも同じである。リテール版とバルクの価格差は1000円にも満たないので、CPUクーラーの付いているリテール版を選択した。価格は8900円。今回購入したリテール版には、Nidecブランド（日本電産）のCPUクーラーが付属されていた。

この時点での残金：4万655円

さて、CPUを購入する上で、密接にかかわりあってくるものがある。それは、マザーボードだ。CPUがSocket7のK6-2やM IIにすると、Socket7用のマザーボードを選ばなくてはいけない。また、Pentium IIならSlot1用のものを、Socket370のCeleronならSocket370か、Slot1用のマザーボードとSocket370→Slot1変換アダプタを用意しなくてはならない。そんなことは当たり前のことだが、最近のチップセットはどうも覚えにくく、Socket370とSocket7の違いもよく見ないとわかりくい。今回は、Socket370タイプのCPUを購入しているので、マザーボードもSocket370タイプのものを購入する。それも、今回は予算の都合上、グラフィックス機能を搭載したオンボードVGAタイプのものを探すことにした。

Celeron対応のマザーボードのうち秋葉原で多く見つけることのできたのは、チップセットに、SiS620、Intel810、Intel440BX+VGA、Intel 440ZX+VGAを使用したマザーボードだ。オンボードVGAでもチップセットに統合されていないものは割高になるので、SiS620とIntel810の2つのどちらかを選ぶことになる。ここで問題になるのが、Linux上でX Window Systemが動作するかである。もし、動作しない場合は別のグラフィックスカードを別途購入しなくてはならなくなる。幸いにも、7月にSiSのチップセットに対応したXFree86のドライバがようやくリリースされたので、SiS620のマザーボードを選択した。

購入したマザーボードは、ELITEの「P6SEP-ME」、価格は1万800円。Micro ATXにオンボードで、Sound Pro HT8738AM/PCIというサウンドチップが搭載されていた。このサウンドチップは、CMEDIAのCMI8738と同じチップのようだ。なお、CMEDIAのWebサイトでは、Linux用のドライバが提供されている。また、4Front TechnologiesのOSSドライバからも利用できる。

残金：2万9315円　（まだまだ余裕）

CMEDIA
http://www.cmedia.com.tw/
4Front Technologies
http://www.4front-tech.com/

オンボードVGAでないIntel440ZXやVIA Apollo Proのマザーボードでも、

写真1
CPUはCeleron366のSoket370タイプ、リテール版を購入。クロックは66MHz×5.5倍で動作する。

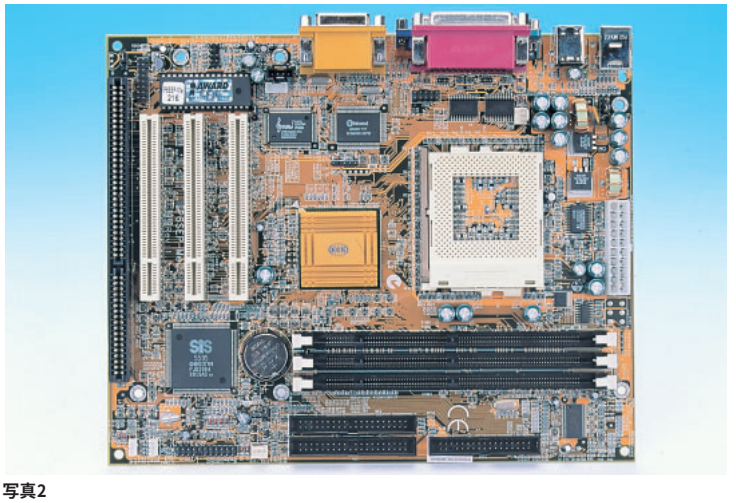


写真2
マザーボードチップセットにSiS620を使用したELITEのP6SEP-MEを購入。サウスブリッジはSiS5595。統合型のマザーボードはMicroATXが多い。

8000円程度で見つけられる。これに、SiS6326やIntel740を搭載した安価なグラフィックスカードを3～4000円で購入できれば、上記の組み合わせと変わらないので、注意深く探してほしい。

メモリ

CPUとマザーボードが決まったので、メモリを探す。メモリといえば、7月初めには、PC/100対応SDRAMの128Mバイトの価格が9000円を割り、下げ止まらない感じであったが、アメリカの大手メモリメーカーのトラブルの噂で、7月末の時点では1万1000円台に逆に値上がりしてしまった。今回は128Mバイトのメモリでは予算に足がついてしまう。逆にGUI環境では32Mバイトではかなり辛いので、64MバイトのPC/100対応SDRAMを6000円で購入した。

残金：2万3015円　（結構減ったなぁ）

ハードディスクとCD-ROMドライブ

ハードディスクは、価格面やデバイスなしに使える点でIDEのドライブを選択する。ハードディスクの大容量化には、この1～2年、目を見張るものがあり、現在では2万円で購入できるハードディスクは軽く10Gバイトをオーバーしている。また、Ultra-DMA/33やUltra-DMA/66の技術で、SCSIのドライブ並にデータアクセスの高速化も進んでいる。ここでハードディスクにかけられる予算は1万1000円までだ。今回は、Seagateの「ST34312A」という4.3Gバイトのハードディスクを9980円で購入した。それにしても、秋葉原のショップでは、あと3～4000円出すと6～8Gバイト、6000円出すと10Gバイトの容量のものが買えてしまうのだ。「ハードディスクの価格っていったい」と考えてしまう。

残金：1万2536円　（あれ？あとこれだけ？）

CD-ROMドライブは、たぶんインストールのときしか使用しないと考え、ひたすら安いドライブを探すことにした。もちろん、ATAPIのドライブだ。しかし、安いCD-ROMドライブはなかなか見つからない。6000円ほど出せば、30倍速でも40倍速でも買えてしまうのだが、今回予定している金額は4000円

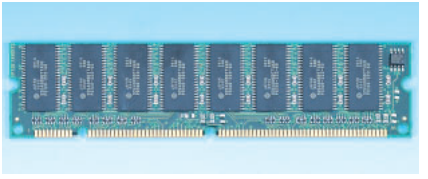


写真3
メモリは、日立製作所のチップが搭載されている。今回は入手しやすさから、あえてPC/100対応メモリを選択した。

写真4
ハードディスクはSeagateのMedalist 4312（ST34312A）という4.3GバイトでUltra-DMA/33対応のドライブを購入。回転数は5400rpm、512Kバイトのバッファが搭載されている。

写真5
CD-ROMは、24倍速ドライブを購入。MITSUMI製のATAPIのCD-ROMドライブはよく見かける。DOSでも汎用のATAPIドライバですんなり認識してくれるので、自作ユーザーに人気が高い。

Column

ハードディスクのスペックの見方

最近、ハードディスクのモデルチェンジが早い。新しく発売されるハードディスクは大容量、低価格、そして高速化が進んでいる。ハードディスクの性能の違いを判断するために、回転数、シークタイム、ディスクバッファ容量を説明しておこう。

まず、回転数は、「rpm」という車のエンジンの回転数と同じ単位で表している。ハードディスクでのrpmは、ドライブ内の円盤（プラッタ）が1分間に何回転するかで表し、数字の大きいほうがより多くのデータを短時間に読み書きできるというわけだ。SCSIのハードディスクでは10000rpmを超えたものもあるが、IDEのドライブでは、5400rpmと7200rpmの速度が主流だ。しかし、現状では5400rpmと7200rpmの体感的な速度の違いはほとんどない。ハードディスクの回転数が上がると、当然それだけ発熱も高くなるので、回転数の高いドライブを購入するときにはケース内の熱対策にも注意しなければならない。

シークタイムはヘッド（データを読み書きするピックアップ）の移動速度のことで、単位はms（ミリ秒）が使われる。今回購入したST34312Aの場合、平均シークタイムは9msとなっている。この数字が小さい方が目的のデータのある場所にヘッドが到達するのが速い。

ディスクバッファ（キャッシュ）は、最近アクセスされたデータを一時的に取っておき、次のアクセス時にバッファ内にデータがあったなら、実際にはディスクにアクセスせずに以前使われたデータを利用することで高速化を計るもの。連続するデータをバッファに先読みしておくことで、何度もハードディスクにアクセスしないで済むようにも使われる。

ディスクバッファ容量は大きいほど性能が良い。現在発売されている主なハードディスクには、256Kバイト～2Mバイトという容量のメモリが搭載されている。

以下なので、16倍速から24倍速のドライブを探すことになる。5店舗に1台は、安いCD-ROMドライブが見つかるであろうとたかをくくっていたのだが、見とおしはかなり甘かった。CD-ROMドライブを探すために、もう1周秋葉原を回る羽目になる。結局、MITSUMIの24倍速CD-ROMドライブを3980円で見つけたので、有無を言わずに購入した。

残金：8357円　（うっ、1万円札がない）

Ethernetカード

LinuxではDEC（COMPAQ）の通称「Tulip（DEC2114x）」搭載のEthernetカードが10Base-T/100Base-TXで安定して使いやすいため人気である。しかし、これでも十分安いのだが、値段が4000円台からしかないため、今回は手が出せない。

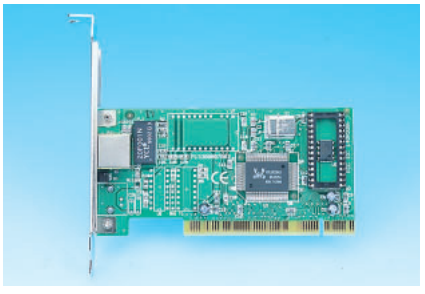


写真6
今回、購入した無印のPCIバス対応Ethernetカード。10Base-T対応にしか対応していないが、カード自体の設定なしにNE2000互換として動作する。

写真7
ケース内部もきれいに作られていて、特価品とは思えないATXケースを購入した。

さて、秋葉原に来るまではISAバスの安価なNE2000互換Ethernetカードを予定していた。Realtekの「RTL8019」のチップを搭載したISAバスのNE2000互換カードが、高くても1200円くらいで購入できるからだ。最近のNE2000互換のチップを搭載したISAバスEthernetカードは、たいていDOSから起動して、IOポートアドレスとIRQを指定しなくてはならないので、非常に面倒ではあるが、これさえ指定すれば、それほど相性問題もなくすんなり動いてくれることが多い。

しかし、今回は、PCIバス対応のNE2000互換として利用できる、Realtekの「RTL8029」を搭載した無印Ethernetカードが990円で販売されていたため、迷わずこちらを購入した。なお、インストールのところでも述べるが、このカードをLinuxで利用するには、Linux上での設定が必要だ。

残金：7318円　（少しでも節約せねば）

それにしても、今回、秋葉原を回ってみて気づいたのだが、10Base-T/100Base-TXに対応したPCIバスのEthernetカードが2000円台で購入できるようになってきている。特に、VIAの「Rhine（VT86C100A）」というチップを搭載したPCIバス対応のEthernetカードがあちらこちらで見られた。メルコの「LGY-PCI-TXR」や、プラネックスコミュニケーションズ「FNW-9700-T」、コレガの「FastEtherII PCI-TX」などだ。本当はこちらを選んだほうが、100Base-TXへの移行がスムーズに行くだろう。

写真8
動けば何でもいいと思って購入したキーボードとマウスだが、思ったより使いやすかった。

その他のデバイス

さて、これまで、CPU、オンボードVGAマザーボード、メモリ、ハードディスク、CD-ROMドライブ、Ethernetカードを取り上げてきたが、その他に購入したデバイスを紹介しよう。

本来は、ケースの良し悪しにも目を向けるべきだが、予算を考え、安さを第一条件にケースを選んだ。今回は、ATXケースを4980円で購入した。通常、ATXケースは安くても7000円以上であり、この価格ではめったに手に入らないが、パーツショップの店頭で数量限定土日特価で販売されていたのを購入。安物だと馬鹿にしていたのだが、持ち帰ってから中を開けてびっくり。内部の作りもしっかりとしていて、おかげで組み立て中にケースを血に染めることなく作ることができた。電源はENARMAXの250Wが載っている。

秋葉原ではよくケースが安値で販売されているが、特価品の中には、組み立てるときに手を切りそうなものや、電源ファンがうるさいものが多い。マザーボードがMicro ATXならば、ケースも小さめのMicro ATXにしてもよいだろう。Micro ATXのケースは、電源の電力が少ないものが多いが、拡張性を考えなければ、省スペース、省電力で使えるのは魅力である。

残金：2089円（安いのが見つかってよかった）

残るは、キーボードとマウスだが、これに関しては、セカンドマシンとして利用するので、本当に動けばよいというものを選択した。そこで、日本語

109のPS/2キーボードに「Power off、Sleep、Wake up」の3つのキーが付いた、ACPIから電源を管理できるキーボードを1680円で購入した。

残金：325円　（あれ？足りないかな）

マウスは、2ボタンのPS/2マウスが500円くらいから見つけられたのだが、LinuxのX Window System環境で利用するということで、所持金不足だが、3つボタンのPS/2マウスを980円で購入した。ちなみに、2ボタンのPS/2マウスでも現在のX Window Systemを利用する上ではまったく問題ない。マウスとキーボードは、USB接続のタイプがあるが、LinuxではUSBのサポートは最近始まったばかりであるため、はたして動作するか不安だ。もちろん、マザーボードにUSB機能が付いていなければ、動作以前の問題で接続することもできない。キーボードの中には一般的なPS/2ではなくAT用のコネクタを持つものもあるので、この辺も間違えないようにしよう。もし不安ならば、ショップの店員に確認すればよい。

AT用のコネクタのキーボードを購入した場合には、ATからPS/2に変換するコネクタを500～1000円で購入できる。マウスもシリアルポートに接続するタイプのものがあるが、こちらはLinuxでも問題なく利用できるので、シリアルポートを2つ使う場合を除いてはそれほど心配する必要はない。

残金：▲704円　（赤字だー）

買ってきたパーツを組み立てる

さて、パーツを購入したら、いよいよ組み立てだ。たいていのPCのパーツは保証期間が1週間から1カ月と短いので、できるだけ早めに組み立てるようにしよう。静電気や配線ミスのショートがあると、パーツはすぐに壊れてしまうので注意が必要だ。

まず、ケースの内部をばらして、マザーボードを取り付けられるようにする。ジャンパーピンの設定は、このマザーボードでは必要なく、CPUのクロック設定などはBIOS側で行う。マザーボードを固定するスペーサは、6カ所固定するのが一般的だ。このスペーサを穴以外のところに付けたまま電源を入れると、ここでショートしてしまうこともあるので正確に取り付けよう。

次はケースとマザーボードのケーブルの取り付けだが、その前にこのケースでは、電源が邪魔して、あとからパーツが取り付けにくくなっているため、先にCPUとメモリ、電源ケーブルをマザーボードにセットする。マザーボードへのケーブルの取り付けは、マザーボードのマニュアルをよく確認しながら行う。通常、これらのケーブルは、黒または白い線がマイナスを示してい

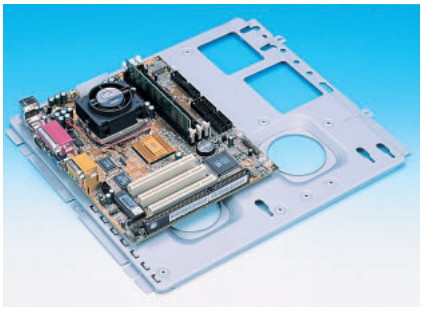


写真9
ケースから底板を外し、マザーボードを取り付ける。CPUとメモリ、CPUファンを装着する。

写真10
マザーボード、CD-ROM、ハードディスクをケースに取り付ける。コネクタ類やCPUは、電源部分の奥になる。

Column

Linuxで買ってはいけないパーツ

Linuxで買ってはいけないPCパーツといえば、まず、最新のチップの搭載されたマザーボードや、グラフィックスカード、Ethernetカードが挙げられるが、特にグラフィックスカードには注意したい。

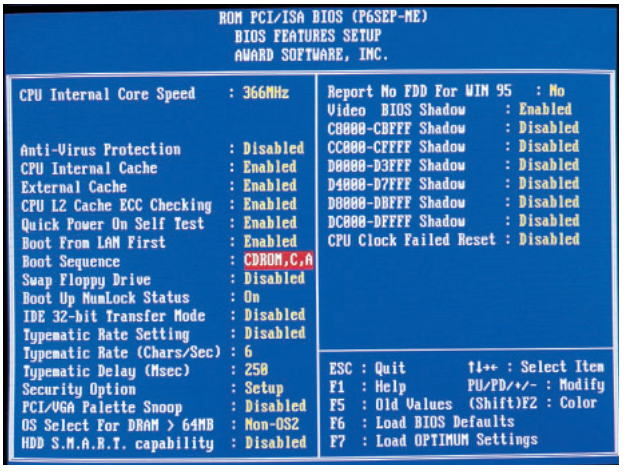
LinuxでX Window Systemを動かすときにはグラフィックスカードのチップに対応したドライバが必要だ。通常、グラフィックスボード付属のドライバは、Windows用しか提供されていないところがほとんどで、あってもOS/2のものくらいだ。一般にLinux用のドライバは他の開発者の手によって、ボランティアで作られているのだが、グラフィックスチップの仕様が公開されていなければ、そのチップに対応したLinux用のドライバは作れない。

現在でこそLinuxも市民権を得るまでに至ったため、ほとんどのチップベンダーが仕様を公開しているが、以前は仕様の公開されていないチップベンダーの製品に対して、Linuxのコミュニティで不買運動が起こったりしたものだ。昔からS3製のグラフィックスチップを搭載した製品がLinuxのコミュニティで人気があったのは、早くから仕様を公開していたからである。Linuxで安定したものを利用したければ、少し型落ちした製品が購入することをお勧めする。

モデムについても触れておこう。内蔵モデムの中には、コストを下げるためにモデムチップを使用せず、DOSやWindowsのソフトウェアでモデムの信号を作り出すDSPタイプのものがあるが、そのようなモデムはLinuxでは使えない。このタイプは最近のPCIバスの56kbpsモデムや、サウンド機能と一体型のモデムによく見られる。できれば外付けモデムを利用しよう。

写真11
完成した5万円Linuxマシン。

画面3
Boot Sequenceでブートしたいデバイスの順番が変えられる。「C」はハードディスク、「A」はフロッピーディスク。



る。LEDだけは、プラスとマイナスの極性に気をつけよう。

ケースにCD-ROMドライブとハードディスクを取り付ける。まず、IDEのマスタとスレーブのジャンパ設定を先に行う。通常は、ハードディスクがマスタ、CD-ROMドライブがセカンダリに設定されている場合がほとんどだ。ケースにネジ止めし、IDEケーブルと電源コネクタを付ける。このとき、マザーボードに付けるケーブルの向きは、IDEポートの1番ピンがある方向にIDEケーブルの赤い線があるようにして付ける。ハードディスクとCD-ROMドライブ側は、電源に近いほうから1番ピンとなっている。

あとは、EthernetカードをPCIスロットに取り付ければ、ほぼ完成だ。このときケースを閉めてしまわずに、一通り動作を確認してからケースを閉めるほうが、問題が起きたときに対処しやすい。

最後にキーボード、マウス、電源のコード、ディスプレイケーブルを取り付ければひとまず完成だ。マシンが起動したら、まずBIOSを立ち上げよう。このマザーボードでは、AWARDのBIOSを使用しているので、起動したらすかさずDELETEキーを押す。

BIOSでの設定は、まず、[STANDARD CMOS SETUP]で時計の調整、IDEのハードディスクをすべて「AUTO」に、Drive A:を「Disable」に変更した。

[BIOS FEATURES SETUP]では、1番上のCPUクロック設定、Boot Sequenceを「CDROM,C,A」として、CD-ROMブートできるように変更。[CHIPSET FEATURES SETUP]のVGA SHARED MEMORYを「8MB」に設定した。あとは、セーブして再起動すればマシンの組み立ては完了だ。(**画面3**)

TurboLinux 日本語版4.0のインストール

TurboLinux 日本語版4.0をインストールしてみよう。前号の付録CD-ROMに収録されていたTurboLinux日本語版4.0は、CD-ROMブートに対応している。先ほどBIOSでCD-ROMブートに設定し、起動前にCD-ROMにセットしておくことで、自動的にCD-ROMからインストーラが起動されるようになっている。CD-ROMから起動しても、インストール方法は変わらないので、通常通りインストールを行う。

インストール方法は、ほかに任せるとして、ここでは今回組み立てたマシンで、インストールの注意点を2点述べたい。まず、インストールデバイスの検出のところでは、ネットワークデバイス(Ethernetカード)が検出される。検出されたデバイスは、「ne(VIA Rhine PCI First Ethernet)」であった。しかし、このデバイスドライバは、購入記のところで述べたVIAのRhineというチップを搭載したEthernetカード用のものであり、今回のマシンのEthernetカード(RTL8029使用)を自動認識することができなかったようだ。これは、インストール終了後にあらためて設定することで回避できるので、そのままインストールを続行しよう。

次は、インストール途中のグラフィックスカードの検出だ。今回使用したマザーボードのグラフィックス機能は、SiS620というチップセットに統合されている。通常の自動インストールではビデオメモリが0バイトのSVGAカードと検出されてしまう。ここでは「XF86_SVGA」がインストールされるが、そのままインストールを続行しよう。あとで、この「XF86_SVGA」が必要になるので、別のグラフィック

スカードに設定し直さないほうがよい。

インストールが終了するとPCを再起動する。そのとき、忘れずにCD-ROMを取り出そう。ここでCD-ROMを取り出さないと、再起動したときにまたCD-ROMからブートして、またまたインストーラが起動してしまう。もちろん、再起動時にBIOSでCD-ROMよりもCドライブを先に起動するように設定し直せば問題ない。

インストール後にネットワークカードを設定

さて、インストールが終了したら、インストールのときに検出に失敗したEthernetカードと、グラフィックスカードの設定をしよう。まず、ネットワークが使えない状態だと、グラフィックスカードの修正パッチをコピーできないので、ネットワークの設定から先に行う。先ほど述べた通り、Ethernetカードは、VIAの「Rhine」というチップとして認識されている。しかし、今回はRealtekの「RTL8029AS」というチップを用いたEthernetカードなので、rootでログインするか、suコマンドでスーパーユーザーになってから/etc/conf.modulesをviなどのエディタを使って修正しよう。

```
$ su
# vi /etc/conf.modules
```
(viを使って/etc/conf.modulesを編集)

ここで、「alias eth0 via-rhine」という行を「alias eth0 ne2k-pci」と書き換える。もし、NE2000互換のチップを採用したものの場合、ほかに「ne」、「rtl8139」という設定に書き換えれば使えるようになることがある。あとは、マシンを再起動すればEthernetが利用できるようになっている。これを確認するには/sbin/ifconfigで確認するか、Windowsマシンにpingを打ってみるとよいだろう。

XFree86のドライバを手に入れる

さて、ネットワークが使えるようになったら、続いてGUI環境を整えよう。まず、インターネットに接続できるWindowsマシンなどから、このマシンで用いているSiS620に対応したドライバをダウンロードする。チップベンダーのSiSのWebページにLinux用のドライバが提供してされている。

SiS620とSiS530用のドライバのURL
http://www.sis.com.tw/driver/mb_620.htm

XF86_SVGA.gzファイルのサイズは1Mバイト弱だ。一緒にReadme.txtとXF86Config.gzもダウンロードする。Readme.txtは、このドライバのインストール方法が書かれているドキュメントである。

XF86_SVGA.gzを手に入れたら、LinuxマシンにFTPなどを利用して、アップロードする。以降の作業はrootで行い、/rootにXF86_SVGA.gzがあるものとする。まず圧縮されているファイルを元に戻す。

```
# gunzip XF86_SVGA.gz
```

これでXF86_SVGAが得られる。このファイルには実行属性がついていないので、修正する。

```
# chmod 755 XF86_SVGA
```

これを/usr/X11R6/binにコピーするのだが、同名のファイルがすでに存在するので、古いファイルの名前を変更しておく。

```
# cd /usr/X11R6/bin ——/usr/X11R6/binに移動
# mv XF86_SVGA XF86_SVGA.orig——古いファイルの名前を変更
```

Column

XFree86 3.3.4について

今回用いたTurboLinux 日本語版4.0には、XFree86 3.3.3.1が含まれているが、8/18現在、より新しいXFree86 3.3.4(以下3.3.4)がリリースされている。3.3.4から、新たに多くのグラフィックスチップがサポートされており、その中にはSiS620/530も含まれている。当初は3.3.4を用いてX Window Systemの設定をしようとした。ところが、turboxcfgのリストにSiS620は出てこなかったため、他のSiSチップを指定したり、「Unlisted Card」を選択してみたが、正しく動作させることができなかった。本文中でSiSの提供するドライバを使用したのは、このためである。

SiS620/530以外に、3.3.4から新たにサポートされたグラフィックスチップは、Intel i740、3dfx Voodoo Banshee、Voodoo3、Matrox G400、nVIDIA RIVA TNT2などである。

Windows環境で非常に人気の高いG400、TNT2、Voodoo3を使ったマシンでもX Window Systemが使えるようになったのは、とても喜ばしいことだ。「超爆速3Dゲーム環境とX Window System環境を1台のマシンで実現したい」という欲張りな願いがかなえられるようになったのだ。

TurboLinux用XFree86 3.3.4が提供されているURL
ftp://ftp.turbolinux.co.jp/pub/turbolinux/snapshot/i386/TurboLinux/RPMS/XFree86-3.3.4-1.i386.rpm

▼リスト1　XF86Configファイルをviで修正する。

```
Section "Device"
    Identifier  "Unlisted Card"
    VendorName  "Unlisted Card"
    BoardName   "unconfigured"
    VideoRam    8192
    Option      "hw_cursor"
    Option      "noaccel"
    Option      "no_bitblt"
    Option      "linear"
EndSection
```

追加

画面4
SiS620のドライバを入手して、X Window Systemを起動した。

```
# cd /root ―――――――――/rootに移動
# cp XF86_SVGA /usr/X11R6/bin―新しいXF86_SVGAをコピー
```

インストールが終了したら、日本語モードのkonからturboxcfgを実行する。ここで、X Window Systemの設定をもう一度やり直そう。

```
# kon ―――――日本語モードのkonに変更
# turboxcfg ――――― turboxcfgを起動
```

「Configure Keyboard」、「Configure Mouse」を設定後、「Probe Video Card」を設定しないか、ビデオカードの自動認識をしないようする。そして、「Configure Video Card」で、手動で設定するようにして、カードリストの1番上にある「Unlisted Card」を選択。サポートしているXサーバは「XF86_SVGA」を選択する。カードのクロックを自動設定した後、ビデオメモリを指定する。単位は「Kバイト」だ。ここでは、BIOSで8Mバイト分確保しているので、「8192」とした。「Configure Monitor」ではモニタを設定し、「Configure Video Modes」では、X Window System起動時の画面領域を設定する。「Configure Font Path」では、手動設定で、「/usr/X11R6/lib/X11/fonts/tt/」という行を削除する。「Test X Configuration」は実行しないで、「Change Login Method」でログイン時の設定を済ませたら、「Write Configuration」で設定を書き込んで終了させる。

さらに、turboxcfgによって作られた設定ファイルに修正を加えよう。「/etc/X11/XF86Config」をエディタで修正する。

```
# vi /etc/X11/XF86Config
(viで/etc/X11/XF86Configを編集)
```

XF86Configは、Section～EndSectionで囲まれたいくつかのセクションに分かれている。そのうちの"Device"セクションに4行追加する。（**リスト1**）

これで、XFree86の設定は完了だ。あとは、通常通りX Window Systemを起動させよう。

```
$ startx
```

これで念願のX Window Systemが起動した（**画面4**）。

消費税込み？

今回購入してきたLinuxマシン用のパーツは**表2**にまとめた。消費税抜き

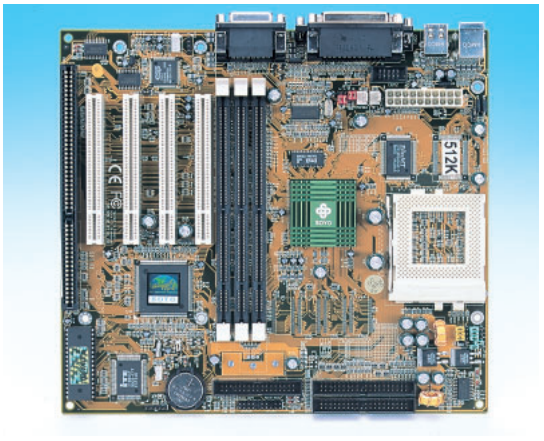


写真12
消費税込みで、5万円に収めるためにSocket 7に挑戦。マザーボードはSOYOのSY-5SSM/5を購入。

写真13
AMDのK6-2 350MHzも箱に入ったリテール版を購入。クロックは100MHz×3.5倍で動作する。

なら4万8290円と、どうにかぎりぎり5万円以内で購入できた。ふだん、秋葉原やWebでパーツの価格を見ているので、最初に立てた予算で楽勝と思っていたが、頭の中では消費税抜きで計算してたりするので、5万円の消費税分5%＝2500円がこんなに大きいとは思わなかった。ところで消費税込みで5万円を超えてしまうのでは、どこが“5万円で作るLinuxマシン”だと、ご指摘される方もいらっしゃるかもしれないので、ほとんど同機能、同スペックになるように、別途、パーツを購入してみたので紹介しよう。

まず、CPUをSocket 7のK6-2に変更して、次にそれに合うマザーボードを探した。残りのパーツは、すでに購入したものと組み合わせる。CPUは、Celeron 366とクロックの近い、K6-2の350MHzを6580円で購入した。CPUクーラーが付属したリテール版である。K6-2の350MHzは、ベースクロックが100MHzで動作するので、ベースクロック66MHzのCeleron 366よりも高速に動作する場合もある。次に、このCPUに合うマザーボードだが、SOYOの「SY-5SSM/5」というSocket 7用のオンボードVGAタイプを1万800円で購入した。セカンドキャッシュが512Kバイトで、オンボードVGA機能つきの「SiS530」というチップセットが搭載されている。このVGA機能も、Celeronで構成したときの「SiS620」のX Window Systemのインストールと、まったく同じ方法でX環境を作ることができる。サウンド機能はESSのSolo1というチップが搭載されていた。

気になる合計金額であるが、**表3**の通り、税込みで4万8269円と、なんとか5万円を切ることができた。

今回5万円でLinuxマシンが作れるのか検証してみたわけだが、思ったよりも高スペックなマシンを組み立てることができた。筆者の自宅のマシンよりも高性能なマシンが5万円で作れてしまったことにはいささかショックを隠すことができないが、このマシンを生かすも殺すもそのあとの使い方次第。さて、このマシンをいかに使うかはまたの機会にご紹介しよう。

パーツ	商品名	税抜き金額（円）	税込み金額（円）
CPU	Celeron 366MHz	8900	9345
マザーボード	ELITE P6SEP-ME	1万800	1万1340
メモリ	PC/100 SDRAM 64Mバイト	6000	6300
ハードディスク	Seagate ST34312A (4.3Gバイト)	9980	1万479
CD-ROMドライブ	MITSUMI FX240 (24倍速)	3980	4179
ネットワークカード	無印 10BASE-T/PCI (Realtek RTL8029AS)	990	1040
キーボード	日本語112キーボード	1680	1764
マウス	3ボタンマウス	980	1029
ケース	ATXケース 250W	4980	5229
合計		4万8290	5万705

表2：購入したパーツ

パーツ	商品名	税抜き金額（円）	税込み金額（円）
CPU	K6-2 350MHz	6580	6,909
マザーボード	SOYO SY-5SSM/5	1万800	1万1340
メモリ	PC/100 SDRAM 64Mバイト	6000	6300
ハードディスク	Seagate ST34312A (4.3Gバイト)	9980	1万479
CD-ROMドライブ	MITSUMI FX240 (24倍速)	3980	4179
ネットワークカード	無印 10BASE-T/PCI (Realtek RTL8029AS)	990	1040
キーボード	日本語112キーボード	1680	1764
マウス	3ボタンマウス	980	1029
ケース	ATXケース 250W	4980	5229
合計		4万5970	4万8269

表3：CPUをK6-2に変更して税込み5万円に収めた場合

Column

HDBENCH cloneでベンチマーク

5万円PCの性能がどれくらいなのか、Linux上でベンチマークテストを行ってみた。

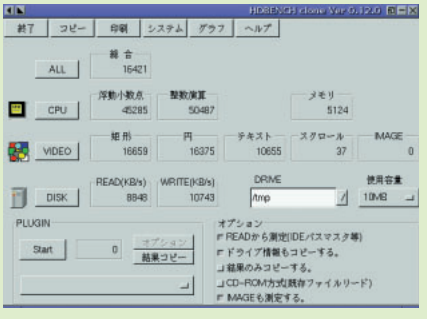
今回試したのは、CPU、グラフィックス、ハードディスクの基本性能を計測するHDBENCH cloneだ。Windowsのベンチマークテストで有名なHDBENCHと、外観はまったく一緒だが、移植ではなくLinux用に新たに作成されたもので、X Window System上で動作する。

原稿執筆時点の最新バージョンであるHDBENCH clone Ver.0.12.0で計測した。本家のHDBENCHとHDBENCH cloneは、計測のアルゴリズムは同じだそうだ。

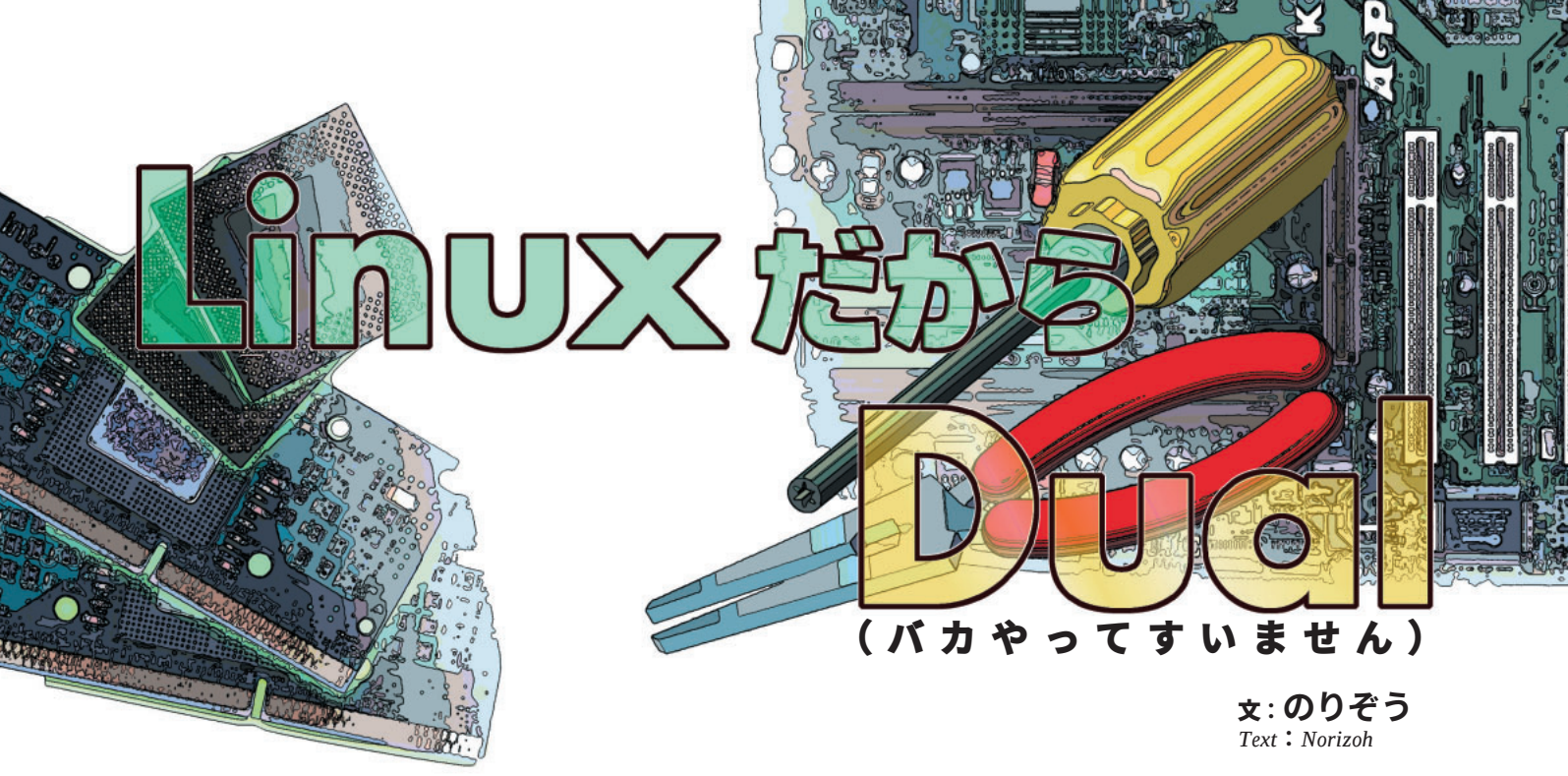
あたりまえだが、OSの構造やドライバの構造、アクセス方法などがすべて違うため、WindowsのHDBENCHと、LinuxのHDBENCH cloneの数字を比較することは意味がない。

HDBENCH cloneは、現在、頻繁にバージョンアップされていて、本家HDBENCHのすべての機能を実装するべく開発が進んでいるようで、Linuxでの手軽なベンチマークソフトとして今後が楽しみである。

HDBENCH cloneのWebページ
http://www.enjoy.ne.jp/~gm/program/ihdbench.html



HDBENCH cloneの画面は、本家HDBENCHとまったく同じといっていいほどそっくりだ。



Linuxだから Dual!

（バカやってすいません）

文：のりぞう
Text：Norizoh

■デュアルCeleronマシンのゆくえ

前号で、SMP Linuxマシンを作るべく、マザーボードやらCPUやらを買い込んできた「のむぞう」氏だが、買い物をすませ、デュアルCeleron 450MHzマシンが無事に動作したところで、ふと我に返ってしまった。「いかん、仕事がたまっている。おい、のりぞう、SMPマシンのパフォーマンスを調べておけ。がっはっは」こういい残すとのむぞう氏は新宿の街へと消えていったのだった。

というわけで、今回は不肖のりぞうがSMPマシンのパフォーマンスを試すこととなった。バカをやらせていただくが、どうかお付き合いをお願いしたい。

■SMPってなんだ？

そもそもSMP（Symmetric MultiProcessing：対称型マルチプロセッシング）とはどのようなものなのだろうか。SMPとは、複数のCPUが同じメモリ空間を共有しながら、互いに同等のものとして並行動作する仕組みだ。複数のCPUが同時に働くことで、システムの高速化を図ることができる。SMPでは、複数のCPUが同時に動作するため、それぞれのCPUが持つメモリキャッシュの整合性をとるなど、OSがSMPに対応した処理を行う必要がある。Linuxは、カーネル2.0のころからSMPをサポートしていたが、カーネル2.2になってSMPへの対応が強化され、カーネル全体をロックしなくなったため、動作がより高速になった。

今回は、デュアルCeleronを搭載したSMPマシンは本当に速くなるのかどうか検証する。

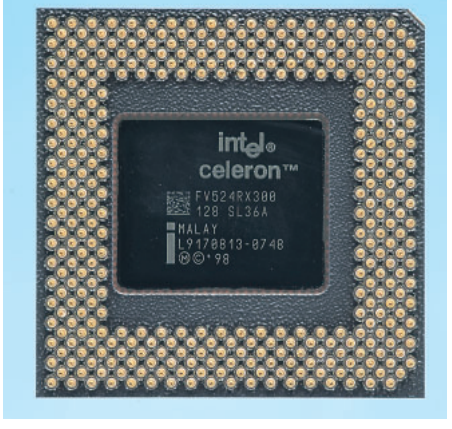
■今回テストしたデュアルマシン

前回スクラッチから組み上げたデュアルCeleronマシンに、英語版Red Hat 6.0をインストールした。このディストリビューションは、Linuxカーネル2.2.5、glibc 2.1を採用している。

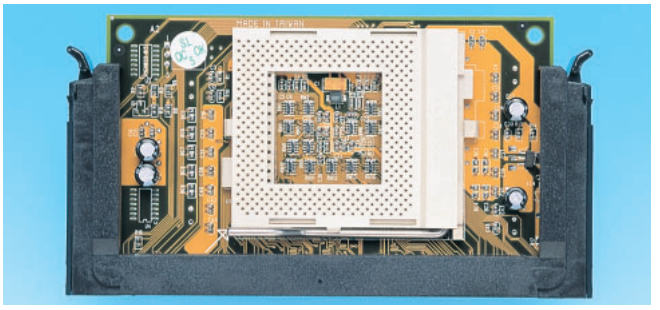
マシンのスペックは**表1**のとおりだ。FSBが103MHzの464MHzでも動作したが、今回は清く正しく定格どおりFSB 66MHzの300MHz動作で実験し

CPU	Intel Celeron 300A（PPGA）×2
マザーボード	Epox KP6-BS
Slot1→Socket370アダプタ	SOLTEK SL02
メモリ	128Mバイト SDRAM（PC100）
ハードディスク	IBM DTTA-35010（10.1Gバイト）
ビデオカード	Matrox Millennium G200 8Mバイト

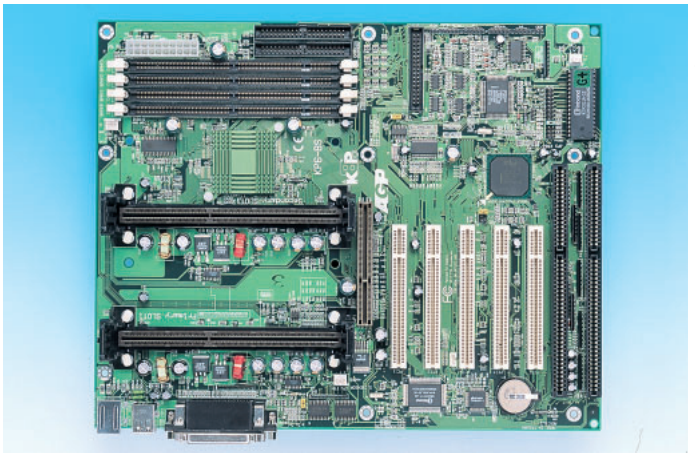
表1　テストに使用したマシンのスペック



PPGA（Plastic Pin Grid Array）のCeleron 300A。オーバークロック耐性が高いのでマニアに人気が高い。今回はおとなしく300MHzで使用した。



SOLTEK社のSlot1→Socket370変換アダプタSL02。基板の改造なしでデュアル化できるバージョンだ。



Epox社のデュアルマザーボードKP6-BS。440BX AGPsetを搭載したごく普通のデュアルマザーボードだ。

画面2
xosviewはマシンのリソースをリアルタイム表示するXアプリケーション。最新版はSMPに対応し、CPUごとの負荷を表示する。

た（Celeronをデュアルで使っている時点で動作保証外なのだが）。

■本当にデュアルで動いてるのか？

マシンを観察すると、スロットに2つのCPUが刺さり、CPUファンが回っているのが見えるのだが、Linuxカーネルは両方のCPUを認識しているのだろうか。SMP対応カーネルで起動してlessで/proc/cpuinfoをのぞくと、processor：0とProcessor：1が記述されている。SMP非対応カーネルで起動すると1つしか記述されない。これで正しく認識しているはずだ。

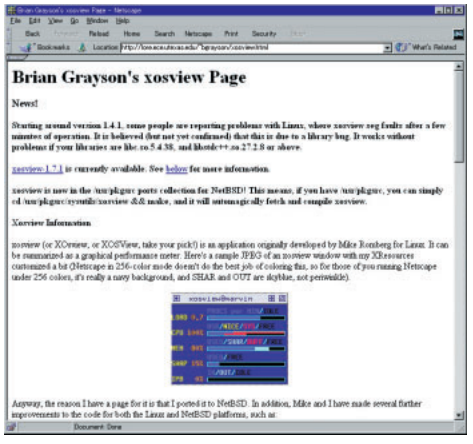
だが、これだけでは不安だ（私は疑り深いのだ）。Red Hat 6.0のCD-ROMを調べると、システムのリソースをグラフィカルに表示するXアプリケーションxosviewがあったのでインストールしてみた。これはSMPに対応した最新バージョンだ（**画面1**）。

X上でxosviewを起動すると**画面2**のような窓が開く。CPUへの負荷、メモリの利用状況やディスクへのアクセスがグラフのように表示される。SMP対応カーネルではCPUメータも2個表示される。ちょっと感動する瞬間だ。このCPUメータは、右から左へと表示が流れており、過去数秒間のCPU負荷が連続的に見えるようになっている。

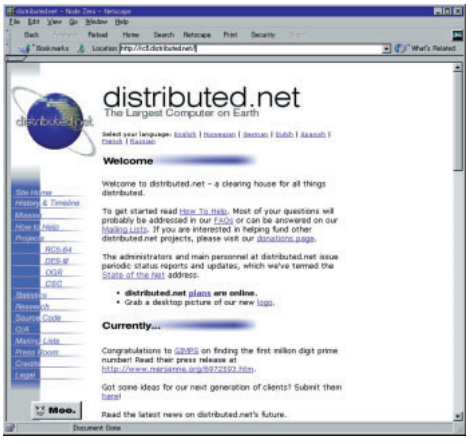
■よし、動かそう（失敗への道）

CPUが2つもあるんだから計算が速いはずだと単純に考え、ひたすら計算をさせてみることにした。サンプルとして、RC5-64／DES解読クライアントソフトを使おう。このソフトは、解読が困難なRC5暗号やDES暗号をインターネットにつながった数多くのコンピュータでよってたかって計算し、総当りで試して解読しようというプロジェクトで使われるものだ（**画面3**）。通常は、他のプログラムの邪魔にならないよう、CPUが暇なときに計算をするようになっている。プロジェクトには、個人やチームで参加でき、それぞれの解読量ランキングが毎日発表される。ランキングが競争心をあおるのか、世界中の人が手近なコンピュータ（職場のマシンも含む）を動員している。RC5-64は、プロジェクト開始から2年近くたった今でも正解が見つかっていない（筆者も自宅のマシンを600日以上計算させっぱなしだ）。

今回使ったクライアントソフトは、ネットワークでデータをやり取りする機能を省いたテスト用のものをhttp://



画面1
xosviewのWebページ（http://lore.ece.utexas.edu/~bgrayson/xosview.html）。



画面3
RC5-64クラッキングコンテストのWebページ（http://rc5.distributed.net/）。いろいろなプラットフォームに対応したクライアントソフトが用意されている。

www.distributed.net/source/から入手し、ソースコードからビルドした。

プログラム本体は「rc5des」で、実行時にオプションとして－benchmarkを与えると一定数のRC5-64とDESの解読計算を行う。このベンチマークモードを使って、シングルCPUとデュアルCPUそれぞれの場合の速度を比べてみよう。シングルCPUとSMPの実行環境は、Linux起動時にカーネルを選択することによって切り替える。

測定は、Xではなくコンソール画面から次のように行った。

```
% time ./rc5des -benchmark
```

bash組み込みのtimeコマンドで実行時間を計っている。さて、その結果は、シングルCPUで19.8秒、SMPで19.7秒。「何でやねんっ！」関西弁で突っ込みたくなる結果だ。

■失敗から学ぶ「マルチタスク」

突っ込んでも問題は解決しないので、編集部の人におとなしく相談する。「こういう結果になったんだけど、SMPって単なるインチキじゃないか」「インチキはあなたです。マルチプロセスでも、マルチスレッドでもないプログラム単発じゃSMPの効果はほとんど出ませんよ。」「マルチ…？ ああ、マルチね、うんそうだね。」なんだそれは。しかたがない、調べてみよう。ごそごそ。

なるほど、わかった気になったぞ。「プロセス」って言うのはプログラムの実行単位のことで、Linuxなどのマルチタスク OSでは、複数のプロセス（プログラム）を同時に動かせる。また、プロセスが自分の複製を作って働かせることもできる。この複製を子プロセスという。DOSでも子プロセスってあったな。でも、DOSはシングルタスクOSだから、子プロセスの実行中は親プロセスは動かなかったっけ。

マルチタスクOSが動作するSMPマシンで複数のプロセスが動いていれば、それぞれの仕事を別々のCPUに割り振ることができるわけだ（**図1**）。

じゃあ、「マルチスレッド」とは何だろう。なになに、「スレッド」もプログラムの実行単位で、マルチスレッドOSでは、ひとつのプロセス内で複数のスレッドを作り、同時に実行できる。ふむ、プロセスよりも小さい単位なんだな。でも、マルチプロセスの仕組みがあればマルチスレッドっていらないんじゃないか。

■マルチスレッドのいいところ

そうだ、編集部の人に訊いてみよう。「マルチプロセスがある以上、マルチスレッドというのは無用の長物だね」「また、インチキなことを。マルチスレッドの方がリソースを食わないし、軽いんですよ。いいですか、マルチタスクシステムは複数のプロセスやスレッドが同時に動いているように見えるけど、厳密にはそれぞれを高速に切り替えてちょっとずつ動かしています」「うん、それは知っている」「個々のプロセスは、独立したメモリ空間やI/O空間を割り当てられています。リソースを個別に割り当てることで他のプロセスとの相互作用を意識しないで動作できるわけです」「マルチプロセスだな。それでいいじゃないか」「マルチプロセスは独立している分リソースを使いますし、OSがプロセスを切り替える際に、現在のCPUレジスタを退避して、次に実行するプロセスのためのレジスタ値を読み込むなど、重い処理を行う必要があります」「忙しそうだな」「一方、スレッドは同一のプロセス内部で動きますから、スレッド間の切り替えも速いしリソースも共有できるんです。SMPシステムではスレッド単位でもCPUを割り振るのでSMPの効果もあがります」「プロセス中に計算のスレッドや描画のスレッドを作って分業させたりするのかな」「そうですね。ユーザーインターフェイスも別スレッドにされることが多いようですね（**図2**）。

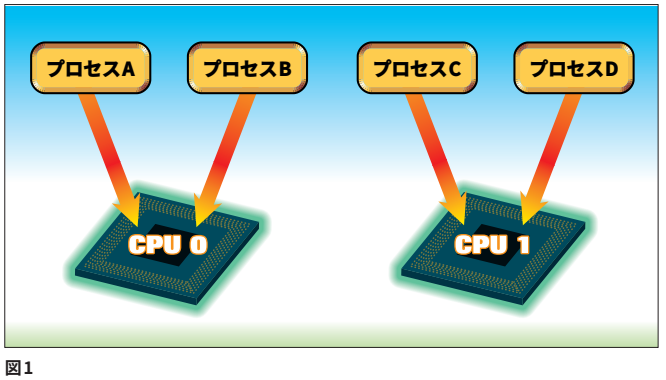


図1
SMPシステムでは、プロセスはそれぞれのCPUに分散して実行される。デーモンやユーザーのプログラムを多数実行する時などに効果が発揮される。

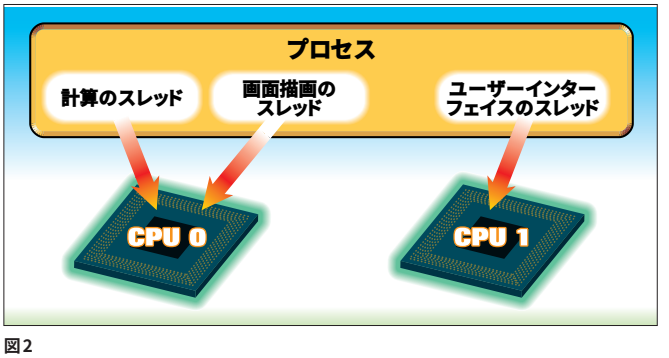


図2
SMPに対応したマルチスレッドOSでは、スレッド単位でも各CPUに仕事を割り振る。たとえば、計算と画面描画、ユーザーインターフェイスの処理を別スレッドにすれば、計算と描画を同時に行いつつ、ユーザーからの入力を受けることができる。

■新たなる試行

先ほど試したrc5desは単一プロセスで、マルチスレッドでないから2つのCPUに仕事が割り振られないんだな。それじゃあ速度差が出るはずもない。

同時に複数のプログラムを起動すればSMPの方が速いはずだ。そこで、シェルスクリプトrc5test.sh（**リスト1**）を作り、次のようにrc5desを2つ同時に走らせ実行時間を測定した。

```
$ time ./rc5test.sh
```

スクリプト中、rc5desの行末に「&」を付けてrc5desをバックグラウンドで動作させる。また、最終行の「wait」は、バックグラウンドで実行しているプログラムの終了を待つコマンドだ。さらに調子にのり、3つ、4つ同時に実行した結果をグラフ1にまとめた。

先ほどと同じように、rc5desを1つだけ実行したときは実行時間に差は出なかった。しかし、複数のrc5desを同時に起動すると、SMPではシングルCPUの時に比べほぼ半分の時間で計算が終わっている。SMPの威力炸裂だ。

余談だが、ネットワーク機能を持った本物のLinux用rc5desクライアントには、マルチスレッドに対応したものもあり、これを使えばrc5desを1つ実行するだけで2つのCPUをぶん回してくれる。実際に試したところ、デュアルCPUでの速度は、シングルCPUのほぼ2倍になった。ただし、ベンチマークモードはシングルスレッドで実行されるようだ。

■今度はMP3に挑戦

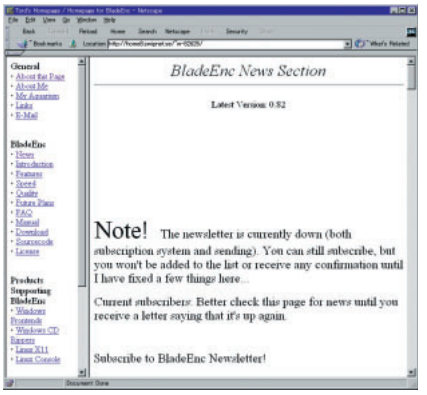
rc5desでの結果は素晴らしいものだった。続いては、MP3のエンコーディングにチャレンジしよう。MP3は、PCMの音声データをおよそ10分の1に圧縮するMPEGの技術だ。1曲で数十Mバイトもある音楽データが、数Mバイトに圧縮できるので、1枚のCD-Rに100曲以上を焼くことができ、重宝するのだが、エンコーディングには膨大な計算が必要で、時間がかかるのが難点だ。

今回は、フリーで公開されているMP3エンコーダBladeEnc Ver.0.82（**画面4**）を使い、41.1Mバイト（4分4秒）の曲をエンコードした。ちなみに、これはある編集部員が持っていた森高千里の曲だ。彼は「結婚しちゃったからもういいんです。さよなら森高」と言いながらCDを差し出した。悲しいお話だ。

PCMデータ（test.wav）は44.1kHz、16ビット、ステレオで、これをビットレート128kビット、ステレオのMP3データに変換する。今回も、シェルスクリプトmp3test1.sh（**リスト2**）、mp3test2.sh（**リスト3**）を作り、BladeEncを単独で実行したときと、同時に2つ実行したときの時間を計った。

計測結果は**表2**のようになった。今度もSMPでは2倍の速度が得られた。同時に複数の曲をエンコードする機会が多い人には朗報だ（そんな奴ぁいないって？）。

しかし、CPUが2つになるとこんなに性能が上がるとは思わなかった。さっそく編集部の人に自慢しよう。「SMPってすごいじゃないか。計算速度が倍だよ、君」「この間はSMPはインチキだって言ってたじゃないですか。



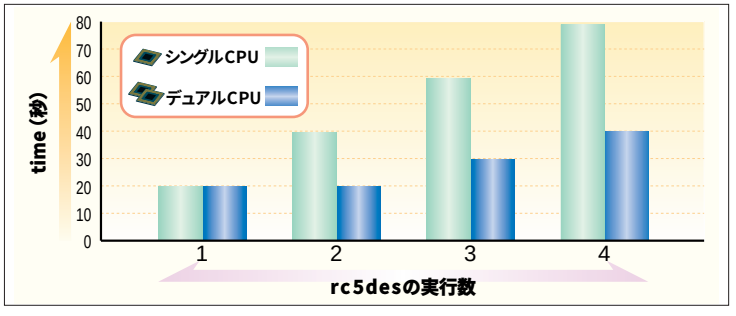
画面4
フリーのMP3エンコーダBladeEncのWebページ（http://home8.swipnet.se/~w-82625/）。BladeEncは音質の良い高ビットレートのMP3ファイルを作成するのに向いている。

▼リスト1　rc5test.sh

```
#!/bin/sh
./rc5des -benchmark&
./rc5des -benchmark&
wait
```

	BladeEncの実行数	
	1	2
シングルCPU	352	702
デュアルCPU	351	351
倍率	1.00	2.00

表2　BladeEncを実行した結果（秒）



グラフ1
rc5desを実行した結果（秒）

▼リスト2　mp3test1.sh

```
#!/bin/sh
./bladeenc -q -quiet \
-outdir=./1 test.wav&
wait
```

▼リスト3　mp3test2.sh

```
#!/bin/sh
./bladeenc -q -quiet \
-outdir=./1 test.wav&
./bladeenc -q -quiet \
-outdir=./2 test.wav&
wait
```

それはそうと、計算量の多いプログラムはCPUばかり使うからSMPは理想的な結果を出しますよ。でも、普通にコンピュータを使うときにはディスクI/Oなども発生しますから、デュアルCPUにしても、全体でのパフォーマンスは2倍にはならないんですよ」「なんだ、わざわざSMP向きのテストを考えたのに」「それだけじゃダメです。定番ともいえるカーネルビルドの時間計測くらいはしてくださいね」

■定番　カーネルの再構築

Linuxカーネルのソースコードは機能の追加とともに、そのファイルサイズも増加の一途をたどっている。現在最新のLinux 2.2.11では、圧縮状態で13.8Mバイト、展開した状態では60Mバイト近い巨大さだ。

カーネルを再構築する際には、ソースファイルがディスクから読み込まれ、コンパイラがコンパイル作業を行ったあと、オブジェクトファイルをディスクに書き出す。この作業を繰り返し、必要なオブジェクトファイルがすべて作られると、リンカがオブジェクトファイルを結合して実行ファイルの完成となる。

もちろん、60Mバイト分すべてのソースコードを使うわけではないが、大量のファイルを処理することになる。コンパイラがCPUパワーを消費し、大量のディスクI/Oをともなうことから、カーネルビルドにかかる時間は、マシンのパフォーマンス指標としてよく使われているのだ。

ここでは、Linux 2.2.11のソースコードからバイナリイメージをビルドする時間を計測した。まず、make mrproperで初期状態にもどす。次に、make menuconfigを実行する。これは、カーネル設定を行うためのものだが、テストのため設定変更はせずにそのまま［EXIT］を選び終了する。ここまでを手動で行い、カーネルをビルドするkerneltest.sh（**リスト4**）の実行時間を計測した。

スクリプトを実行すると、激しくディスクをアクセスしながら、目にもとまらぬ速さでメッセージが流れていく。シングルCPUでの所要時間は380秒。続いてデュアルCPUで計測する。シングルのときと同様、やはりメッセージが飛んでいく。「ん？同様？！」いやな予感がする。ビルド終了、所要時間は375秒。5秒の差は、1.01倍の速度差に過ぎない。

■CPUは働いているか

カーネルのビルドでは2つのCPUが効率的に働いていないのだろうか。ここでXを起動し、先ほども登場したxosviewでCPU負荷をチェックしてみよう。xtermからカーネルビルドスクリプトを動かしたときのCPUメータは**画面5**のようになった。やっぱりだ、片方のCPUが働いているときは、もう1つのCPUはのんびりと休んでいる。

考えてみれば、makeはコンパイルすべきソースファイルを指定してコンパイラを起動し、コンパイルが終わると次のファイルを指定してコンパイラを起動するという動作を繰り返しているのだ。まったく並列化が行われていない以上、2つのCPUは効率的に働かない。

なんだ、もったいない。依存関係のないソースファイルを同時にコンパイルできればもっと効率が上がるはずだ。ところが、私が思いつく程度のことはとっくに実現されているのだ。デフォルトでは、makeは一度に複数のジョブを実行しないが、オプションで実行するジョブの数を指定できる。一般に、ジョブの数はCPUの数＋1にするのがよいとされている。そこで、Makefile中のMAKE＝makeをMAKE＝make －j 3に書き換えて、先ほどのスクリプトを再度実行する。xosviewでCPUの働き具合をチェックすると両方のCPUでバリバリと仕事をこなしていた（**画面6**）。

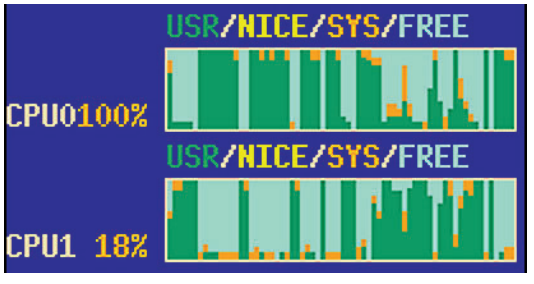
Xを終了して、あらためてカーネルのビルド時間を計った結果が**表3**だ。SMPは、シングルCPUの1.8倍の速度になる。ハードディスクは1台だけなので、ディスクアクセスは一度に1つのCPUしかできないことを考えるとかなり良い結果だと思う。複数のハードディスクを接続して、ドライブの負荷分

▼リスト4　kerneltest.sh

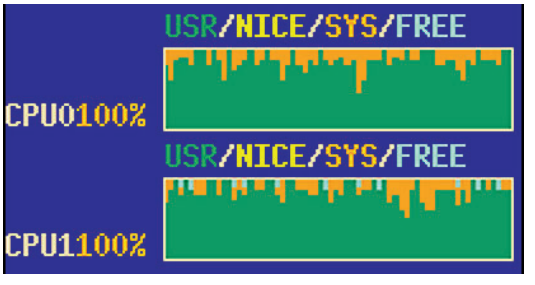
```
#!/bin/sh
make dep;make clean;make \
bzImage
```

シングルCPU	384
デュアルCPU	213
倍率	1.80

表3　make －j 3でのカーネルビルド（秒）



画面5
makeにオプションをつけずにカーネルビルドをすると、CPUは交互に動き、いづれか一方は休んでいる。



画面6
makeの－jオプションで2つのCPUをこき使う。デュアルCPUの醍醐味だ。

散を図ったり、RAIDを導入すれば、SMPの威力がさらにアップするだろう。

■Webサーバのパフォーマンス

Linuxマシンは、ネットワークにつなぎ、サーバとして利用することが多いらしい。そこで、WebサーバをSMPにすることがパフォーマンス向上に寄与するのかを調べよう。

デュアルCeleronマシンにWebサーバプログラムApache 1.3.6をインストールし、ptester（ftp://ftp.lysator.liu.se/pub/unix/ptester/ptester-1.2.tar.gz）というWebサーバパフォーマンステストプログラムを使って測定をした。ptesterは、ソースで提供されているので、次のようにして実行ファイルを作成する。

```
$ make gcc-linux
```

初めに、100BASE-TXでつないだ他のマシンでptesterを実行し、デュアルCeleronマシンのパフォーマンスを調べたのだが、通信速度がボトルネックになるようで、シングルCPUとSMPの性能に差が出なかった。そこで、デュアルCeleronマシン上でptesterを実行し、ローカルループバックでApacheに接続した。

テストは、コマンドラインから次のように入力して行った。

```
$ ./ptester -hlocalhost -k10 -r1
 -t10 /index.html
```

－hでWebサーバを、－kでkeep aliveの回数を、－rで同時にリクエストする数をそれぞれ指定する。また、－tで測定時間を指定し、最後に転送するファイルを指定する。

今回は、2242バイトのindex.htmlファイルを1秒間に何回転送できたかを測定した。テスト時間は10秒間とし、同時リクエスト数1、5、10、20、30の場合の結果を**グラフ2**にまとめた。

通常、Apacheは複数のコネクションに対し、素早いレスポンスを返せるように複数のhttpデーモンを起動する。今回はhttpデーモンが30個起動した状態でテストしている。コネクションが1つしかない場合は、1つのデーモンしか働かないために、SMPにしてもパフォーマンスは変わらない。一方、同時リクエスト数が複数になるとおよそ1.5倍のパフォーマンスの向上が見られる。Apacheは、リクエスト数に応じてデーモンを追加起動するのだが、httpデーモンが初期状態で10個起動した状態で測定すると、同時リクエスト数を30にしたときにパフォーマンスがわずかに低下した。これは、デーモンを起動するオーバーヘッドが原因だと思われる。

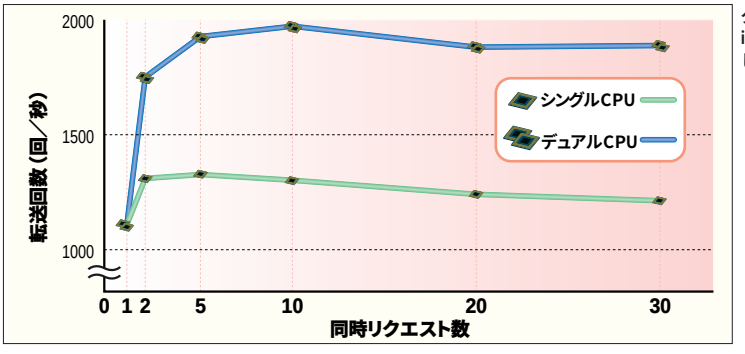
このテストでは、小さなファイル1つを繰り返し転送しただけなので、CPUへの負荷が小さく、100BASE-TXくらいの回線速度ではSMPの効果はほとんど得られないようだが、実際の運用を考えると、CGIやSSIを多用するなど、CPUへの負荷が大きい場合はSMPにする意義があるだろう。また、数多くのファイルをアクセスする場合にはディスクの高速化もあわせて行いたい。システム全体での性能向上を図るためには、一点豪華主義ではダメなのだ。

■SMP化する？しない？

非常に大まかなテストだったが、以上の結果を踏まえると、SMPによる恩恵は、マルチタスク環境下において、CPU負荷に対する耐性が強化されることだと言える。ケースバイケースなので一概には言えないのだが、マシンの利用方法によっては高価な高速CPUに換装するよりも安価なCPUを使ってSMPにした方がコストパフォーマンスが良いかもしれない。用途に合わせ、SMPにするのか、それともCPUを高速なものにするのかを決めよう。もちろん、高速なCPUでSMPにするのがベストなのは言うまでもない。

また、趣味で使うのなら費用対効果などというヤボなことは考えずに「個人的な興味」あるいは「自己満足」のためにSMPマシンを組むのもいい。それでこそ趣味の王道だ。食費は削っても趣味にかける金は惜しまないくらいの気概が欲しい。「そんなに力んでバカなこと書くなよ。見ているこっちが恥ずかしいぜ」「げ、のむぞうさん、いたんですか。そんなこと言ったって、このコーナーはサブタイトルが（バカやってすいません）ですからね。」「そういえば、そのタイトルは俺が付けたんだっけ。」

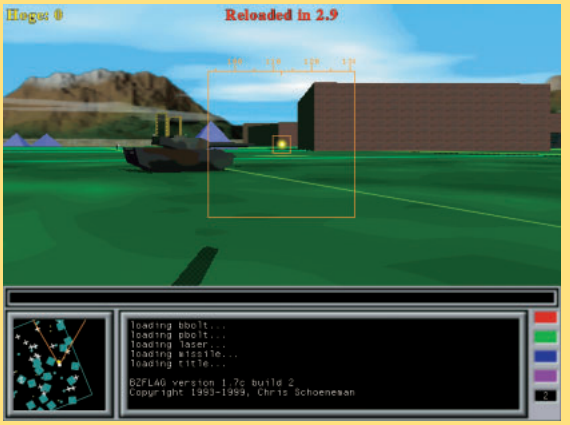
おあとがよろしいようで。



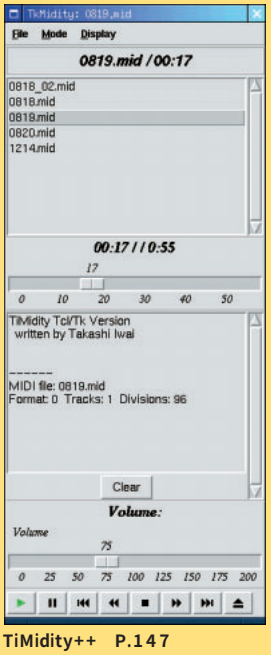
グラフ2　1秒間にindex.htmlを転送した回数（回／秒）

Free Application Showcase

文：出井 一



BZFlag　P.144



TiMidity++　P.147

※紹介したソフトはすべて付録CD-ROMに収録されています。

サウンドアナライザアプレット

Visual Sound Analyzer

バージョン：0.9　　種別：GPL

http://vsa.linuxcore.com/

Visual Sound Analyzer（以下VSA）は、サウンドをビジュアル化するGNOME用のアプレットだ。再生中のサウンドを高速フーリエ変換（FFT）し、スペクトラムアナライザ風のグラフィックをリアルタイムに表示する。背景やグラフの表示がプラグイン形式になっており、好みに合わせて自由に組み合わせられるのが特徴だ。実行にはGNOMEとサウンドデーモンEsounD、高速フーリエ変換ライブラリFFTW（http://www.fftw.org/）が別途必要だ。

●tarボールからビルド

VSAはファイル一式をtar＋gzipしたtarボールとRPM形式の両方で配布されている。ただし、バイナリRPMはRedHat6.0用しか用意されていないので、他のディストリビューションではソースRPMをリビルドするか、tarボールから作成する必要がある。

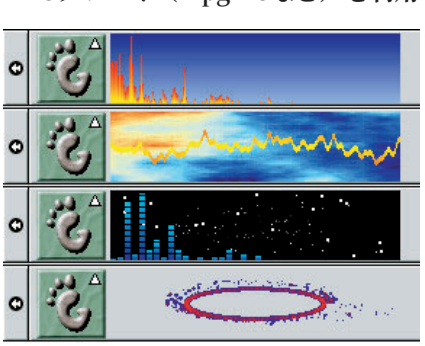
tarボールから作成する場合、「./configure」→「make」→「make install」という一般的な手順をとる。ただし、著者の環境（Vine Linux 1.1）ではEsounDのライブラリがビルド時に読み込まれず、makeに失敗した。この場合は、Makefile.inの17行目を、「LIBS = $(GNOME_LIB) `esd-config --libs` @LIBS@」と修正した後でconfigureから作業をやり直せば正常にビルドできるはずだ。

●好きなプラグインを組み合わせる

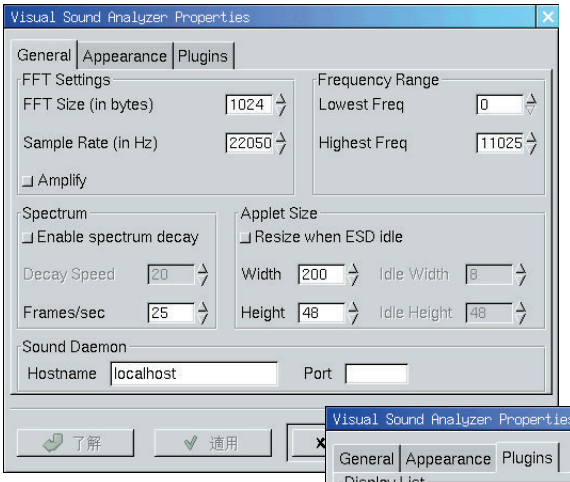
EsounDが動作している状態で、「vsa_applet&」としてVSAを起動すると、GNOMEのパネルに表示領域が確保され、EsounD付属のWAVEプレーヤ（esdplay）や、EsounD対応のMP3プレーヤ（mpg123など）を利用する際に、スペアナ風のグラフィックが表示されるようになる（**画面1**）。

ところで、初期設定ではディケイ（減衰）が有効になっているのだが、この機能は正常に動作していないようだ。パネル上のVSAを右クリックし、メニューから[Properties]を選択してダイアログを開き（**画面2**）、[General]ページの[Enable Spectrum decay]のチェックを外しておこう。

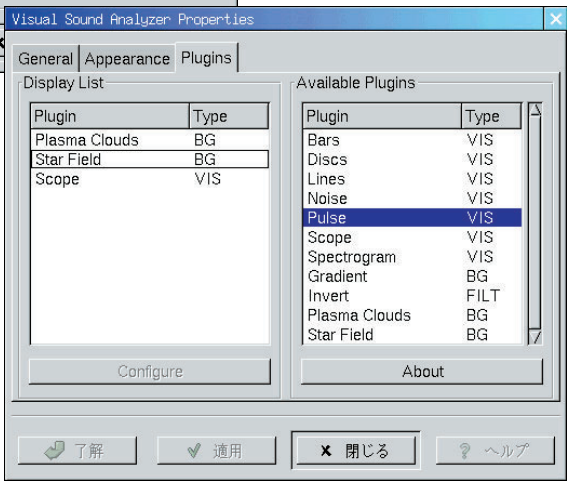
一方、ダイアログの[Plugin]ページ（**画面3**）ではグラフや背景の表示に使われるプラグインを変更できる。グラフ（VIS）は7個、背景（BG）は3個、フィルタ（FILT）が1個と計11個のプラグインが用意されており、自由に組み合わせて使える。右側の[Available Plugins]の中から使いたいプラグインをドラッグし、左側の[Display List]にドロップすればいい。同種のプラグインを複数同時に使うことも可能だ。[Diaplay List]内では、ドラッグ＆ドロップによりパネルでの表示順を変更できる（リスト末尾が手前に相当）。また、[Configure]ボタンで各プラグインの設定を変更可能だ。



画面1
プラグインの組み合わせによりさまざまな表示が可能だ。



画面2
プロパティダイアログで各種設定を変更。



画面3
グラフや背景のプラグインを複数組み合わせられる。

スケジュール管理が可能な日本語カレンダー

nittei

バージョン：1.38　　種別：GPL

http://member.nifty.ne.jp/seto-yoneji/nittei.html

X用スケジュール管理＆カレンダー（以下nittei）は、スケジュール管理を行えるカレンダーソフトだ。国産アプリなので最初から日本語が使える点がうれしい。スケジュールは分単位で設定でき、予定時刻やその何分前かにユーザーに通知してくれる。指定時刻にプログラムを自動実行したり、1カ月分の予定表をまとめて表示することも可能だ。また、休日や記念日の設定はユーザーが自由に追加できる。

●tarボールからビルド

nitteiはファイル一式をtar＋gzipしたtarボールで配布されている。Linux用のMakefileが付属するので、適当なディレクトリに展開後、「make」とするだけでビルド可能だ。

このMakefileにはインストール処理は含まれていないので、ファイルのコピーなどは手動で行う。まず、ビルドされた実行ファイル（nittei）を、環境変数PATHに設定されたディレクトリ（/usr/local/binや~/binなど）にコピーする。

このほかに、tarボールに含まれるリソースファイル（Nittei）と休日ファイル（nit_holiday）の2つが実行に必要だ。リソースファイルは/usr/X11R6/lib/X11/app-defaulsに、休日ファイルは各ユーザのホームディレクトリ（~）にそれぞれコピーしておこう。

●スケジュールの入力

コマンドラインで「nittei」とすると、今月のカレンダーが表示される（**画面1**）。

カレンダーの日付をクリックすると、スケジュール入力用のウィンドウが開く（**画面2**）。スケジュール1件につき1行を使用し、複数のスケジュールを入力可能だ。もちろん、Shift＋Spaceキーや Ctrl＋oキーで日本語入力に切り替えて、日本語でスケジュールを入力できる。

予定時刻に通知してほしい場合は、「(時:分）用件」のように、スケジュールの先頭に予定時刻をカッコで囲んで指定する。たとえば、「(15:30)お茶の時間」とすると、その日の午後3時半になった時点でベルとともに通知用のウィンドウが開く。

予定時刻より前に通知してほしいときには、「(時:分)(分)用件」のように、通知をさかのぼる時間（分単位）を2番目のカッコ内に指定する。たとえば、上の例を「(15:30)(15)お茶の時間」と変更すると、今度は予定時刻の15分前、つまり午後3時15分に通知用のウィンドウが開く。

また、ちょっと変わった機能として、「(時:分)exec=プログラム名」とすることで、予定時刻に指定したプログラムを実行可能だ。たとえば、「(15:30)exec=mpg123 hoge.mp3」とすると、その日の午後3時半になると自動的にmpg123が起動して、hoge.mp3を再生してくれる。

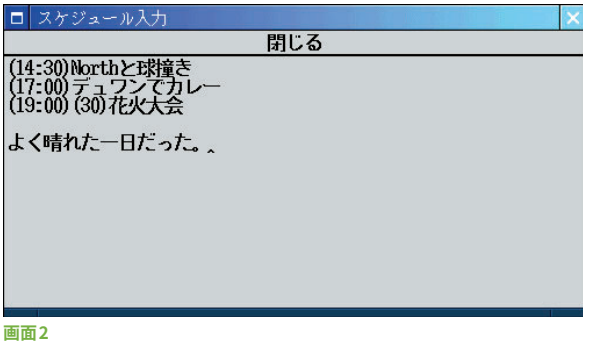
なお、予定時刻を記述しなかった場合、入力した内容はそのまま保存されるが、通知などのスケジュール管理からは外れる。つまり、簡単な日記代わりに使えるわけだ。

●繰り返しの予定を入力する

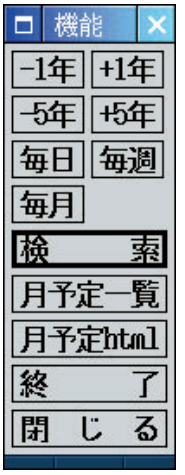
習い事や雑誌発行日のような繰り返しスケジュールを簡単に入力する機能も用意されている。まずは、ウィンドウ下の[機能]ボタンを押して機能ウィンドウを開こう（**画面3**）。

繰り返しの予定を入力するには、こ

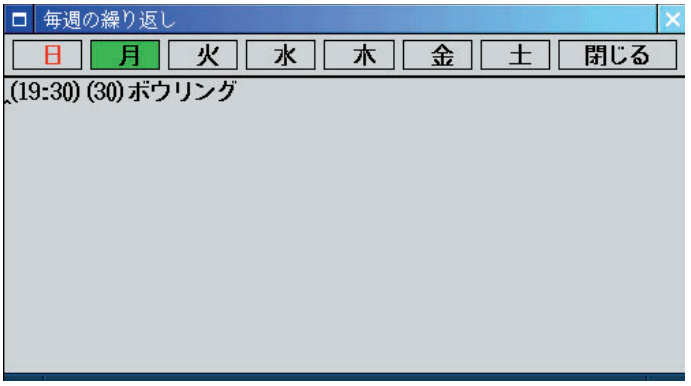
画面1
nitteiのカレンダー。もちろん日本語で表示される。



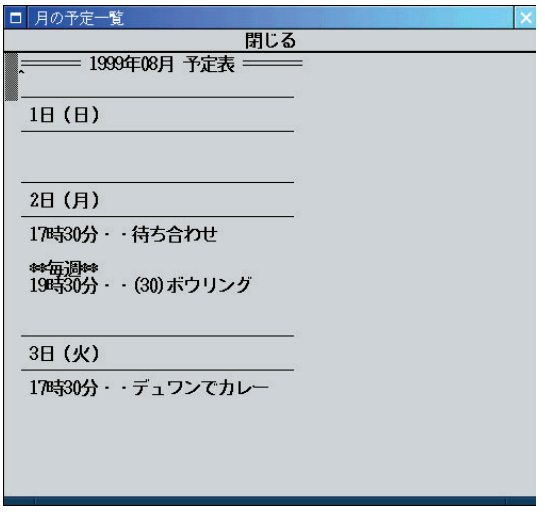
画面2
スケジュールはこのウィンドウで設定する。



画面3
より複雑な設定などは機能ウィンドウで行う。



画面4
毎週決まった曜日に繰り返されるスケジュールを設定。



画面5
1カ月分の予定をまとめて表示することも可能。

のウィンドウの[毎日][毎週][毎月]ボタンを利用する。たとえば、[毎週]ボタンを押すと、毎週の繰り返しウィンドウが開くので、曜日を指定した後、通常のスケジュールと同様に入力すればいい（**画面4**）。

一方、毎年繰り返される誕生日や記念日については、スケジュールではなく休日（のようなもの）として、ホームディレクトリの休日ファイル（nit_holiday）に記述を追加する。このファイルはテキストなので、Muleなど適当なエディタで直接編集可能だ。データは1行1件で、「年,月,日,名前,振替」というカンマ区切り形式だ。

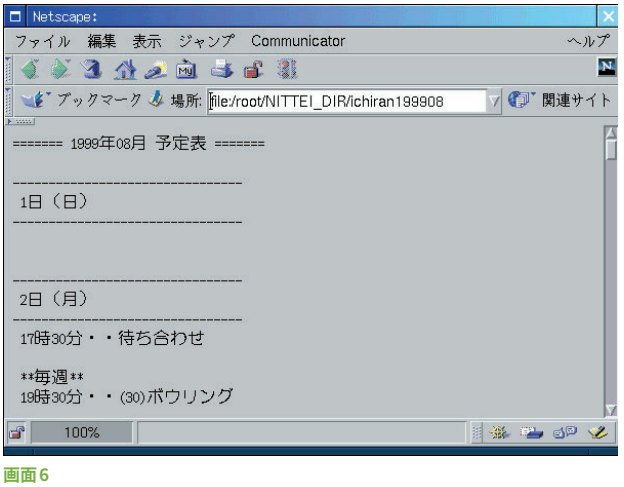
たとえば、3月6日があなたの誕生日ならば、「0,3,6,私の誕生日,n」とする。ここで、年が「0」なのはすべての年に適用すること、振り替えが「n」なのは祝日の振り替え休日の処理をしないことを意味する。

●月予定一覧と検索

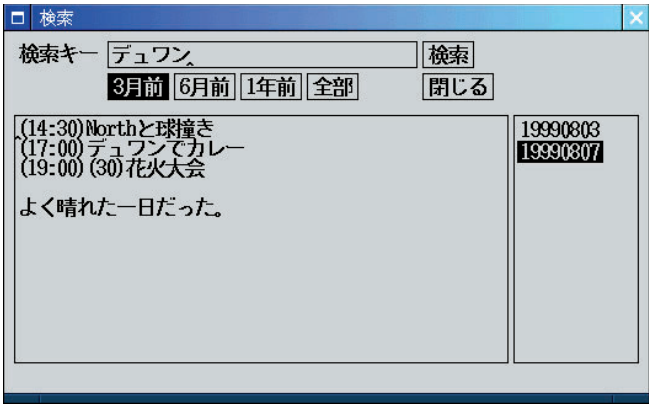
カレンダーには、現在注目している日付を表すカーソルが緑色で反転表示されており、日付のクリックで移動するほか、ウィンドウ下部のボタンで日・月単位で切り替えられる。スケジュール入力ウィンドウを開いた状態であれば、カーソル移動と同時にスケジュールの表示も切り替わる。

しかし、1日ずつスケジュールを眺めるのは大変だ。機能ウィンドウの[月予定一覧]ボタンを押すと、その月のスケジュールがウィンドウにまとめて表示される（**画面5**）。複数月の予定をそれぞれ独立したウィンドウに表示することも可能だ。一方、[月予定html]ボタンを押すと、~/NITTEI_DIR/HTML以下に月予定がHTMLファイルとして保存され、Webブラウザで閲覧や印刷を行える（**画面6**）。

このほか、指定したキーワードを含む日の予定を表示する検索機能も用意されている。機能ウィンドウの[検索]ボタンで検索ウィンドウを開き、検索キーを指定して[検索]ボタンを押せばいい（**画面7**）。



画面6
月予定をHTMLファイルにしてブラウザで表示。印刷もここから。



画面7
検索キーを含む日を検索し、選択した日の予定を表示できる。

ネットワーク3Dタンクバトルゲーム

BZFlag

バージョン：1.7c　　種別：フリー

http://www.bigfoot.com/~bzflag

BZFlagは、サーバ・クライアント方式のマルチプレイヤー3Dタンクバトルゲームだ。TCP/IP接続された各マシンに1人ずつプレイヤーがエントリーし、障害物の置かれたフィールド内で戦闘を繰り広げる。画面の品質はオプション設定で調整できるので、3Dアクセラレーションなしの描画でも十分プレイ可能だ。ここで紹介するLinux版のほか、Windows 9x／NT版やIRIX版も配布されており、異なるOSのマシン同士で対戦することもできる。

●インストール

BZFlagの実行にはOpenGL互換の3Dライブラリ Mesa 3.0（http://www.tempusmud.com/mesa/）が必要。さらにVoodoo 2など3Dfxの3Dカードを利用する場合はGlide（http://glide.xxedgexx.com/3DfxRPMS.html）も必要だ。いずれもRPM形式で配布されているものを利用するといいだろう。

BZFlagはソースファイル一式をtar＋gzipしたtarボールとバイナリのRPM形式の両方で配布されている。どちらもサーバとクライアント両方のソフトが含まれている。通常はRPM形式を利用するのが簡単だ。Mesa 3.0とGlideをRPM形式でインストールした後、「rpm -Uvh bzflag-1.7c-2.i386.rpm」とすればいい。なお、Mesa 3.0やGlideをソースからビルドしたり、3Dカードを使わないのでGlideをインストールしていない場合は、--nodepsオプションをつけてRPMの依存チェックを回避する必要がある。

●クライアントとサーバの起動

BZFlagを実行するには、コマンドラインで「bzflag」とすればいい。画面左上にクライアントソフトのウィンドウが開いて、タイトルとメニューが表示される（**画面1**）。メニューの移動は↑↓キー、選択はEnterキー、キャンセルはEscキーで行う。

まずは、ゲームに参加するマシンのどれか1台でサーバソフト（bzfs）を起動しなければならない。[Join Game]を選択して画面を切り替え、[Start Server]を選択しよう。この画面では、ゲームの種類などさまざまな条件を設定できる（**画面2**）。

ゲームの種類は、参加人数が少ない場合は「Free for all」（自分以外はすべて敵）、ある程度人数がいる場合は「Capture the Flag」（チーム戦で他チームのフラッグを取り合う）を選択するとよいだろう。また、最大プレイヤー数、連続発射可能弾数を制限できるほか、別の場所に瞬時に移動するゲート「テレポータ」やTabキーでの「ジャンプ」、各種の効果をもたらす「スペシャルフラッグ」などの有無も設定できる。

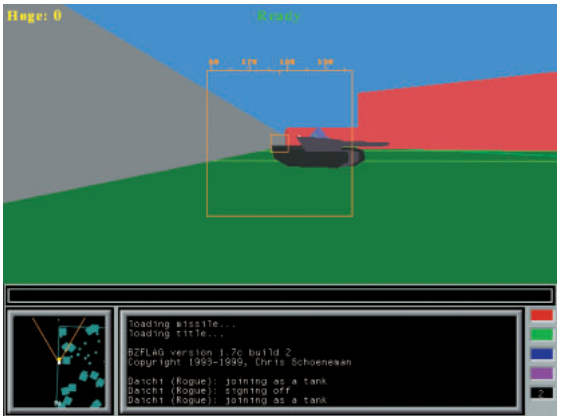
最後に、最下行の[Start]を選択するとサーバソフトが起動する。Escキーでこの画面から抜け、[Find Server]を選択すると、サーバを起動したマシン名が画面下に表示されているはずだ。他のマシンでも同様に、[Find Server]でこのマシンを選択する。あとは、[Join Game]の画面で[Connect]を選択すればいい。コールサインなどを設定したらゲーム開始だ。

●ゲーム中の操作

マウスの右ボタンをクリックするとあなたのタンクがフィールドに現れる。画面にはフィールド内の様子が3D表示

画面1
クライアントを起動すると、最初にタイトルが表示される。

画面2
サーバを起動する。ゲーム全体の各種条件はここで設定。



画面3
プレイ開始。ソフトウェアのみの描画でもこの程度ならスムーズ。

画面4
オプション設定で、プレイしやすい範囲で綺麗な画面を探す。

され、左下にはレーダー、その右にはシステムメッセージや会話のログ、右下には各チーム（色別）のプレイヤー数が表示される（**画面3**）。

タンクの操作には主にマウスを使用する。カーソルを画面の上下に動かすと前進・後退、左右に動かすと旋回だ。移動速度はカーソルが画面中央からどれだけ離れているかに比例する。停止するには中央部の小さな矩形にカーソルを置けばいい。

まずはレーダーを使って敵を見つけよう。レーダーのレンジは1、2、3キーで変更可能だ。敵を視界に捕らえたら、画面中央に来るように移動しつつ左ボタンで弾を発射する。なお、一度に連射できる弾数（初期設定では1発）を超えると、リロードに3秒を要するので注意。このほか、プレイヤー全員（あるいは仲間だけ）にメッセージを送ることもできる。

●画面品質は柔軟に設定可能

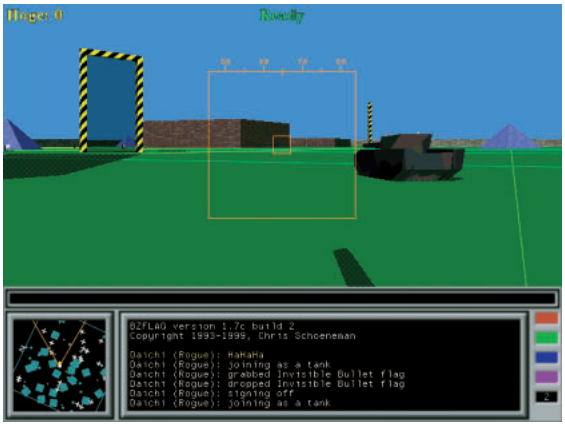
初期設定の画面表示では貧弱すぎて燃えないという人は、Escキーでいったんゲームから抜け、[Options]を選択してオプション設定を行おう。ここでは、画面表示やサウンド、キー割り当てなどの設定を変更できる（**画面4**）。

最初の8項目が画面表示に関する設定で、初期設定では「On/Off/On/Off/Off/Low/Off/Off」とかなり抑え目の設定になっている。CPUがPentiumIIやCeleronならば、ソフトウェアによる描画でも「On/On/On/On/Linear/Low/On/On」程度は軽くこなせるはずだ（**画面5**）。さらに、Voodoo2などの3DカードとGlideライブラリを組み合わせれば、最高品質の画面（**画面6**）で快適にプレイできる。

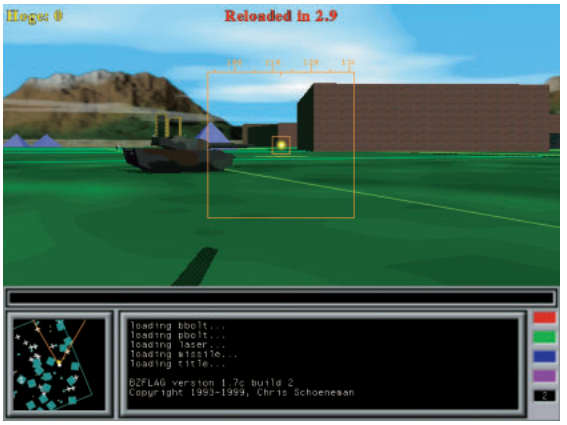
●インターネットでも対戦可能

BZFlagのサーバを起動する際、下のほうに[Server Visibility]という項目があることにお気づきだろうか。この部分を変更することで、ローカルなネットワーク内だけでなく、インターネットに接続している他のプレイヤーとの対戦も可能になる。[Join Game]の画面で、[Server]に使用するサーバ名を入力すればいい。

いちいち対戦相手を見つけるのが面倒なら、すでにインターネット上で運用されているサーバに接続しよう。BZFlagのWebページには、誰でも接続できるサーバのリストが公開されている（http://groundhog.pair.com/bzflag/servers.html）。なお、これらの中には運用時間が限られているサーバもあるので注意されたい。



画面5
テクスチャや影などを有効にする。CPUが速ければまだ余裕。



画面6
品質を最高にする。さすがに3Dカードなしではゲームにならない。

著名ソフトシンセソフトがバージョンアップ

TiMidity++

バージョン：2.3.0　　種別：GPL

http://www.goice.co.jp/member/mo/timidity/index-jp.html

TiMidity++は、外部音源を持たないマシンでも、MIDIデータの再生が可能なソフトウェアシンセサイザだ。音源データ（パッチファイル）とMIDIデータから実際の音のWAVEデータを生成し、リアルタイムに音楽を再生してくれる。WAVEデータをファイルに保存することも可能だ。コンソールベースで使うだけでなく、Tcl/TkやGTK+などを利用した各種インターフェイスも用意されている。

●インストール

TiMidity++はソースファイル一式をtar＋gzipしたtarボールで配布されている。「./configure」→「make」→「make install」という一般的な手順でビルドすればよいのだが、初期設定ではdumbインターフェイスしか組み込まれない。Tcl/TkやGTK+インターフェイスを組み込むには、「./configure --enable-tcltk --enable-gtk」のように、configureスクリプトにオプションを指定する必要がある。

なお、このtarボールには音源データ（パッチファイル）は含まれていないので、リンクページで紹介されている配布サイトから適当なパッチファイルを取得する必要がある。たとえば、作者の出雲正尚さんのページには33Mバイトの高品質なパッチファイルが用意されている。

これではちょっと大きすぎるという場合は、松本庄司さんのページ（http://www.i.h.kyoto-u.ac.jp/~shom/timidity/）にある10Mバイトや4Mバイトのパッチファイルを利用するといい。

●インターフェイスを選択可能

TiMidity++を使ってMIDIデータを再生するには、

```
$ timidity ファイル名
```

とすればいい。ファイル名は複数指定可能だ。ファイル形式は、一般的な標準MIDI形式のほかレコンポーザ形式にも対応している。

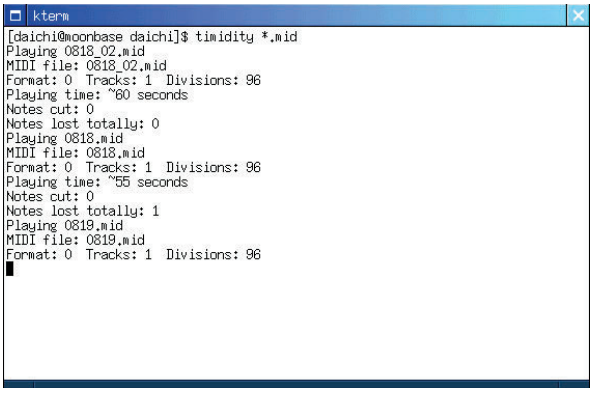
また、これらのファイルをtar＋gzipした状態のまま再生することも可能だ。具体的には、

```
$ timidity hoge.tar.gz#hoge.mid
```

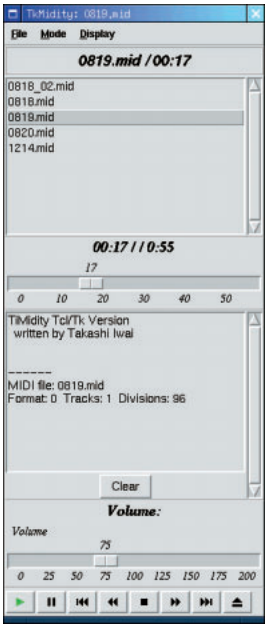
のように、「アーカイブファイル名＋#＋ファイル名」の形で指定する。「#ファイル名」を省略すると、アーカイブ内の全ファイルを再生する。このほか、HTTPプロトコルなどを利用してネットワーク上のファイルを再生する機能も用意されている。

Tcl/TkやGTK+などのインターフェイスを利用するには、-iオプションに続けてインターフェイスの種類を指定する。

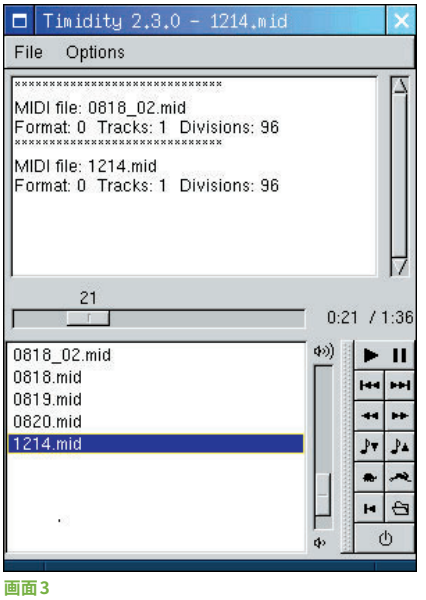
たとえば-ikでTcl/Tkインターフェイス（**画面2**）、-igでGTK+インターフェイス（**画面3**）が使われる。



画面1
コンソールベースでの使用が基本。



画面2
Tcl/Tkインターフェイス「TkMidity」による表示（-ikオプション）。



画面3
GTKインターフェイスによる表示（-igオプション）。

ファイルを自動ダウンロード／アップロード

cURL

バージョン：5.9.1　　種別：MPL（Mozilla Public License）

http://www.fts.frontec.se/~dast/curl/

cURLは、指定したURLのファイルを自動的にダウンロードするコンソールベースのツールだ。この種のソフトとしてはwgetが有名だが、cURLはHTTP、FTPプロトコルのほか、Gopher、DICT、HTTPS（SSL）プロトコルにも対応しており、ダウンロードだけでなくアップロードも可能なのが特徴。反面、単独ではリンクをたどって複数ページをダウンロードできないので、自動巡回などには向いていない。

●インストール

cURLはtarボールとRPM形式の両方で配布されている。ソースからのビルドも難しくないので、どちらを利用してもよいだろう。

なお、SSL対応のcURLをビルドする場合は、あらかじめOpenSSL（http://www.openssl.org/）をインストールしてから、「./configure --with-ssl」のように、configureスクリプトにオプションを指定する必要がある。

●使い方は簡単

使い方は簡単で、コマンドラインにそのままURLを記述すればいい。たとえばFTPの場合、

```
$curl ftp://www.ring.gr.jp/pub/linux/Vine/
```

とすると、FTPサーバ（www.ring.gr.jp）の/pub/linux/Vineディレクトリのファイル一覧が端末画面に出力される。

```
$curl ftp://www.ring.gr.jp/pub/linux/Vine/VinePlus/README.EUC
```

のようにファイル名を指定すると、その内容が端末画面に表示される。ファイルをダウンロードするには、-Oオプションを指定して、

```
$curl -O ftp://www.ring.gr.jp/pub/linux/Vine/VinePlus/README.EUC
```

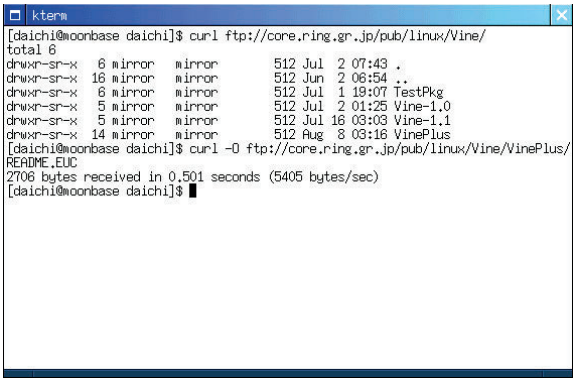
とすればいい（**画面1**）。anonymous FTPサーバでない場合は、「-u name:passwd」という形式でユーザー名とパスワードを指定する。

●各種フロントエンドもあり

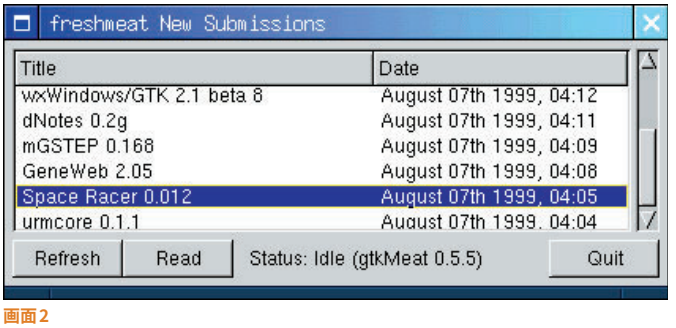
cURLをGUIで使えるようにするフロントエンドソフトもいろいろ開発されている。

cURL用のフロントエンドの代表格は、複数ページのダウンロードが可能なGetleftだが、Tcl/Tk8.1が必須なので日本語環境で使うのは難しい。

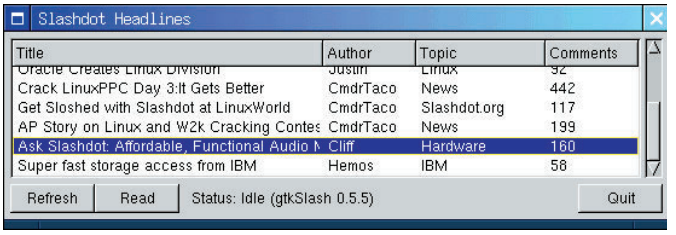
また、gtkMeatとgtkSlash（http://www.sonic.net/~tengel/）は、Freashmeat（ソフト情報サイト）やSlashdot（ニュースサイト）にcURLを使ってアクセスし、得られた情報をリスト表示してくれるというもの（**画面2、3**）。[Read]ボタンを押すとWebブラウザが起動して詳細が表示される。



画面1
ファイルのダウンロード中には受信済みサイズなどが表示される。



画面2
cURLを使ってFreashmeatからソフト更新情報を取得するgtkMeat。



画面3
Slashdotのニュースを取得するgtkSlash。詳細はブラウザに表示。

プロセス情報をグラフィカルに表示

gPS

バージョン：0.2.1　　種別：GPL

http://www.dcc.unicamp.br/~guazzibe/gPS/

gPSは従来psやtopコマンドを使って表示していたプロセス情報を、GTK+を使ってグラフィカルに表示するソフトだ。topコマンドと同じ形式でプロセス情報が表示され、項目名の内容によってソートしたり、特定のプロセスだけに表示を絞り込んだりできる。また、各プロセスを選択後、メニューやボタン操作によりKillなどの各種シグナルを送ることも可能だ。

●プロセス情報を一覧表示

gPSは、ファイル一式をtar+gzipしたtarボールで配布されている。「./configure」→「make」→「make install」という一般的な手順でビルドすればいい。

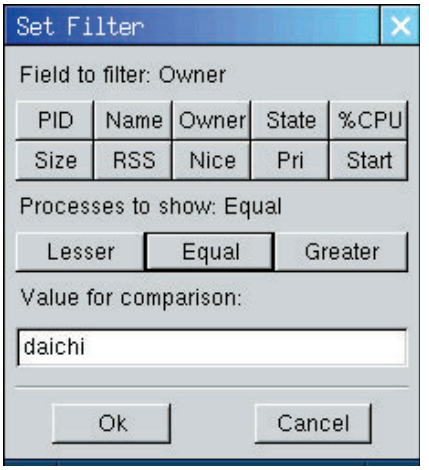
ktermなどのコマンドラインで、「gps」とすると、ウィンドウが開いて現在のプロセス情報が表示される（**画面1**）。これらは、左から順にプロセスID・プロセス名・オーナー名・状態・CPU占有率・メモリ使用量・実メモリ使用量・nice値・優先度・開始時間を表しており、項目名をクリックすると昇順・降順でソートされる。

なお、そのままでは[Refresh]ボタンを押さない限りプロセス情報は更新されない。一定時間ごとに自動的にプロセス情報を更新するには、[View]-[Continuous Refresh]を有効にする必要がある。

●フィルタ設定とシグナル送出

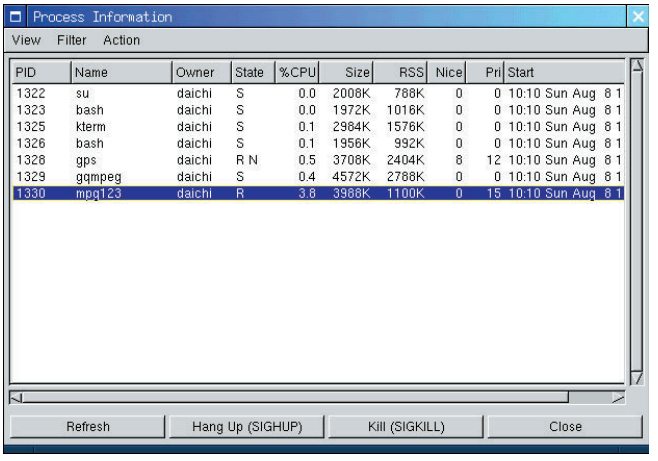
表示されるプロセスが多すぎて、目的の情報が見つけにくい場合は、フィルタを使って表示を絞り込もう。[Filter]-[Set Filter]でフィルタ設定用ダイアログが開く（**画面2**）。対象項目、比較用の値、表示条件（比較用の値より大きい・等しい・小さい）を設定して[OK]ボタンを押すと、表示条件にマッチするプロセスだけが表示されるようになる（**画面3**）。元の表示に戻すには、[Filter]-[Clear Filter]を選択すればいい。

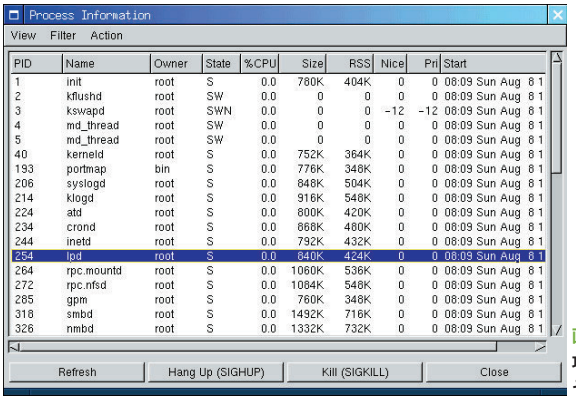
また、プロセスをクリックして選択すると、[Hang Up]／[Kill]ボタンが利用可能になる。これらはそれぞれSIGHUP／SIGKILLシグナルを選択したプロセスに送るので、プロセスの終了／強制終了処理が可能だ。このほか、[Action]-[Renice]では、各プロセスのnice値の変更も行える（**画面4**）。



画面2
フィルタ機能を使うと、表示するプロセスを絞り込める。

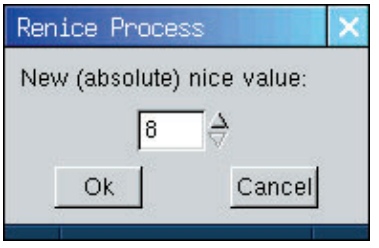
画面3
オーナーが「daichi」であるプロセスだけを表示。





画面1
項目名をクリックすると情報がその項目内容でソートされる。

画面4
各プロセスのnice値を後から変更することも可能。



Tcl/Tkで書かれたHTMLエディタ

August

バージョン：0.37　　種別：GPL

http://www.lls.se/~johanb/august/

Augustは、Tcl/Tkで書かれたタグ挿入型のシンプルなHTMLエディタだ。カラー構文表示などは備わっていないものの、ツールバーのボタンでタグを効率よく入力でき、ブラウザを起動して実際の表示を確認できる。また、日本語化されたTcl/Tk8.0jpを使用すれば、ソースを変更することなく日本語の入力・編集が可能だ。

●インストールと内容の修正

Augustは、ファイル一式をtar＋gzipしたtarボールで配布されている。このうち、実行に必要なのはTcl/Tkのスクリプト（august）のみだ。

このままでは表示フォントが小さいなどの問題があるので、テキストエディタを使って以下のように修正する。

まず、966行目の「font create Courier…」の末尾「14」を「10」に変更し、1338行目の「-foreground black…」の末尾に「-font Courier」を追加する。また、1124行目の「ｧ」を「(C)」に、その下の「ｮ」を「(R)」に変更しておこう。

修正後のファイルは、cpコマンドなどを使って環境変数PATHに設定されたディレクトリ（/usr/local/binや~/binなど）にコピーすればいい。

●ボタンやメニューでタグを挿入

コマンドラインで「august」とすると、ボタンが2列に並んだウィンドウが開く（**画面1**）。使い方は一般的なタグ挿入型のHTMLエディタと同じで、ツールバーのボタンを押したり、[Tags]メニューの項目を選択すると、対応するタグがカーソル位置に挿入される。

なお、ボタンの中には、タグで囲むテキストを選択した状態で押さないと機能しないものもある。

画像やテーブルなど、属性の指定が複雑なタグについては、専用のダイアログが開く。

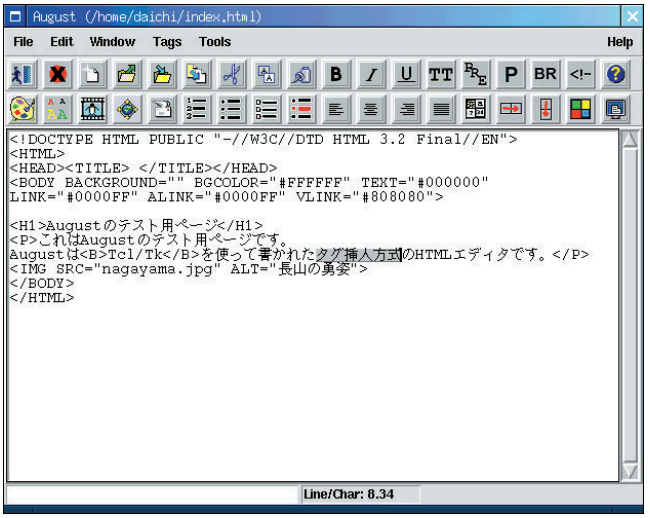
たとえば、画像の場合はファイル名やサイズなどをダイアログで設定する（**画面2**）。

このほか、定型的な部分をテンプレートとして登録しておき、リストから選択するだけで一気に挿入することも可能だ（**画面3**）。初期設定では、HTMLファイルの基本構造のテンプレートが用意されている。

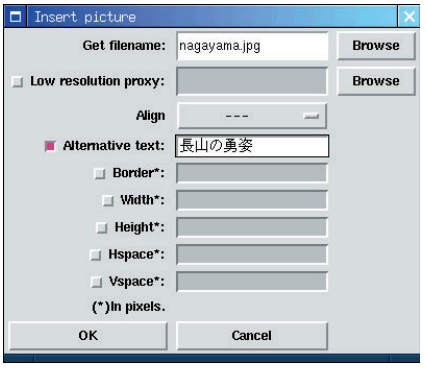
●日本語の入力・編集もばっちり

日本語化されたTcl/Tkでは、文字コードは自動的に日本語EUCに変換されるため、JISやシフトJISで書かれたHTMLファイルをそのまま読み込んでも正常に表示され、そのまま編集可能だ。ただし、保存時の文字コードは日本語EUCになる。

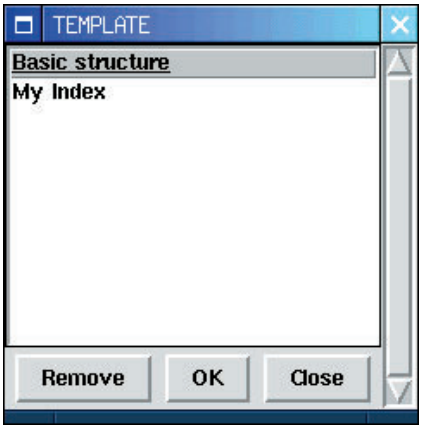
なお、日本語の入力切り替えは、一般的なCtrl＋oキーではなく、Muleと同じCtrl＋\キーで行う。元の状態に戻すにはCtrl＋oキーを使う。



画面1
2列のツールバーに並んだボタンを押すとタグが挿入される。



画面2
画像用のタグ（<IMG>）の属性はこのダイアログで設定。



画面3
定型的な部分はテンプレートとして一気に挿入してしまおう。

オープンソース
・
アクティビズム

ジャカルタでインド人とバラデロの夢を見た

文：安田幸弘
Text : Yukihiro Yasuda

新しいビジネスの誕生

先日、ジャカルタのホテルに泊まってテレビを見ていたときのこと。確かCNNだったと思う。アメリカ製と思われるニュース番組で、キューバの美しい海岸にゴージャスなリゾートホテルの建設を計画しているアメリカ人ビジネスマンが、自由化の進むキューバへの投資がいかに有望なビジネスであるかを自信満々に語る映像が流れた。

「こんな美しい海岸を壊して、醜悪なホテルを建てるなんて」と、一緒にテレビを見ていたインド人ともども、大いに憤慨していたのだが、それにしてもアメリカという国のビジネス好きは大変なものである。一見、無価値なように見えるものに価値を付け、それを売る。そもそも、ソフトウェア産業というビジネスだってそうだ。ビル・ゲイツというアメリカ人の天才的なビジネス感覚は、それまでハードウェアのオマケぐらいにしか思われていなかったソフトウェアをビジネスの種に仕立て上げたわけだ。

しかし、キューバの美しい海岸を破壊して醜悪なホテルを建設することに憤慨する人がいるように、多くの人々の手によるソフトウェアという創作物を独占して売ることに憤慨する人もいる。そんな人々の筆頭格が、GNUのリチャード・ストールマンである。

ストールマンは、決してビジネスを否定はせず、フリーソフトウェアを土台とするビジネスモデルを提示してはいるものの、ビジネスセンスという点では決して才能豊かというわけではなかったようだ。彼の思想に共感する世界中のプログラマーの支持を得ながらも、フリーソフトウェアでは食えないというのがプログラマーたちの本音だったというべきだろう。

ところが、である。ビジネス好きなアメリカ人の中から、食えないはずのフリーソフトウェアをビジネスにする方法を考えついた連中が出てきたのだ。

もちろん、これまでにもフリーソフトウェアをビジネスにしてきた人々はいた。フリーソフトウェアのCD-ROMを売ったり、フリーソフトウェアの解説書を作って売るビジネスは、それなりに成立していたのも事実である。しかし、それらのビジネスは一部の好事家を対象としたビジネスであり、決して大きな市場を見込めるものではなかった。

そんな事情を変えたのが、Linuxの成功だった。

しばしば、なぜBSD 4.4Lite系のFreeBSDではなくLinuxが普及したのかが論じられるが、ぼくはLinuxがカーネルだけの存在だったことが大きいと考えている。つまり、システムができあがっていた4.4 Lite系には手を入れる余地が少なかったのに対し、Linuxには誰もがさまざまなディストリビューションを工夫する余地があった。その工夫がビジネスチャンスだったのではないだろうか。

新しいビジネスが意味するもの

現在、先進国からの観光客は発展途上国にとって大きな収益源になっている。しかし、観光ビジネスによる、美しい自然の破壊や私企業による独占といったさまざまな弊害は深刻である。それに対し、美しい自然そのものを楽しむことを目的とするエコツーリズムといった提案がなされているという。

そこにある美しい自然をそのままに楽しむというエコツーリズムは、ぼくはどこかオープンソースというビジネスに似た発想を感じる。

知識や情報は、本来、他者に伝えられ、コピーされ、少しずつ形を変えながら伝播されていくものだ。ソフトウェアにしても同じだろう。

ソースコードが隠され、パッケージ化されたソフトウェアは、自由なコピーも変更も許されないと言う意味で死んだソフトウェアだ。だが

オープンソースで配布されるソフトウェアは、人から人へと伝えられ、使われていくうちに少しずつ進化を続けていく、生きたソフトウェアである。

オープンソースソフトウェアのユーザーは、そんな生きたソフトウェアのダイナミズムを楽しむ人々であり、それをめぐるビジネスはソフトウェア本来の楽しみを手助けする水先案内のようなものなのではないだろうか。

数カ月前、プラハで行われたNGOの活動を支援する国際ネットワークのワークショップに参加した。そこでの議論はオープンソース一色だったと言っていいだろう。国際的なNGOの活動を支援するために、どのようなグループウェアが必要か、ということが議論されていたのだが、執行部からは当初、完成度の高い市販のソフトの採用が提案された。それに対して、技術者の間からオープンソースによるソリューションの対案が提出され、結局、オープンソースをベースとした開発が進められることになった。

この間、パッケージ製品とオープンソースの特質がさまざまな観点から議論された。ユーザー側からは、市販製品を支持する声もあった。また国際NGOを支援するには特定のメーカーの製品に依存するべきではない、知識は共有すべき、というポリシー的な議論もあったのだが、結局、「どんなに優れた製品でも、他人の作った世界の中で他人の技術のお守りをするだけの仕事なんてしたくない。自分の技術と独創を生かせるオープンソースをベースに仕事をしたい」という技術者の本音がオープンソースの支持につながったようだ。

こうした技術者の本音は、企業では通りにくいかもしれない。オープンソースがビジネスモデルとして提示されたのはつい最近のことだ。しかし、このような技術者たちの声が実績を積み上げ、オープンソースのビジネスモデルがじわじわと普及していく可能性は十分にあるのではないだろうか。

そして、冒頭で触れたジャカルタの話だが、実はこれも国際NGOがらみのワークショップだった。ここではより政治的な観点からコンピュータ、ネットワークという技術が語られたのだが、論点のひとつが開発途上国でのソフトウェアの価格だった。

現在、ソフトウェアは、世界中でほぼ同一価格で販売されるようになっている。一見、フェアに思われるかもしれないが、所得水準を無視した価格は決してフェアではない。実際、インドネシアの労働者は、OSを買うために1か月分の給料をはたかなければならない。ハードやアプリケーションを含めると数年分の給与に匹敵する。

ワークショップでは、ライセンス協定による格差の解消が提案されていたが、もうひとつの可能性として提示されたのが、やはりオープンソースだった。一般のユーザーが利用する場合、オープンソースもサポートにコストがかかる。だが、オープンソースなら現地のスタッフが現地の価格水準でサポートできるので、各国の経済事情に合った価格でシステムを提供できる。メキシコでは、すべての学校でのコンピュータ教育はLinuxで行うことにしたというが、今後、発展途上国でオープンソースがインフラとして普及する可能性は十分にあり得る。

もちろん、既存のソフトウェアビジネスの反撃は大いに予想されるが、その反撃がフェアなものである限り、かえって歓迎されるだろう。

次号から何回かに分けて、オープンソースをめぐる知的所有権をはじめとする、個別の問題について考えていきたい。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ぬくぬくLinux

アミーガ、Linux、QNXのあやしい関係。

文：山形浩生
Text:Hiroo Yamagata

Amiga comes back!

アミーガ（http://www.amiga.com/）を知っている人はかなり古参のマニアで、持ってるヤツは、こりゃもうかなりビョーキ入ったマニアだろう。そうだなあ、いまでも「あのウゴウゴルーガのCGマシン」とかいう紹介をするのがいいのかなあ。とにかくもう、変なことしかできない。画像やサウンド処理に専用のカスタムチップを使い、'80年代からバリバリのフルカラーグラフィクスやステレオサウンドが使えた。変なゲームとか、変なCGとか、あるいは音楽関係とか、マルチメディア系がむちゃくちゃ強くて、さらにお値段の安さから、貧乏人のたむろするヨーロッパでは絶大な人気。さらにMacintoshのエミュレータとか、異常な発想の機器やソフトがテンコ盛り。最近、PC/AT互換機をMac化する、FUSIONというソフトが話題になったけれど、あれをつくったジム・ドリューも、BeOSやLinux用のMac環境Sheepshaverを作っているクリスチャン・バウアーも、アミーガ出身の変なやつら。CGソフトのLightWave3Dも、もとはアミーガ用の化け物ビデオ編集システムVideoToasterの付属ソフトだったんだ。さらにアミーガOSは、初めからマルチタスクを平気でこなす先進ぶり。

そんなわけで、このシステムには貧乏であやしいファンが群がった。Linuxのファンは狂信的でヤな連中が多いと言われるけれど、アミーガのファンはそれを蹴倒すキXXイぞろい。製造元のコモドールが10年以上前に倒産したのに、このマシンはまだ生き残っているのだ。まずドイツの数社が、アミーガのCPUボードを出している。しかも、もともと68Kマシンだったのが、今ではPowerPCアップグレードボードまである。そしてそれをベースに、クローンシステムみたいなものも出ている。ソフトもまだちまちま出ている。

だから製造元の倒産後も、買収話はたくさんあった。けれど、まったく腰が落ち着かない。たらいまわし状態で、どこもきちんと開発に資源を割かない。一方でかつての優位性がどんどん薄れる。もうアミーガくらいのマルチメディアなら、MacやDOS/Vでもこなせるし、マルチタスクもそんな先進的なものではなくなってきた。このままではじり貧、存在意義すらなくなってしまう！

それがようやく、去年に落ち着き先が決まって、開発も本気で再開されようとはしていた。そして次期OSの開発パートナーもほぼ決まりかけていた。あのQNX（http://www.qnx.com/）だ。

驚異のOS「QNX」

QNX。リアルタイムOSの雄。マイクロカーネルで、ほとんど極限までコンポーネント化が進んでいる。Linuxも安定性では定評があるけれど、QNXの敵ではないね。Linuxでも、カーネル更新にはシステムを止めるしかないけれど、QNXはシステムを止めずにリアルタイムで更新できる。常に最新システムを使いつつマシンを一度も止めない、などという芸当ができてしまう。

リーヌス・トーヴァルズは、マイクロカーネルはオーバーヘッドが増えて、かえってでかくなっていやだ、という。でもQNXの場合、本当に必要なコンポーネントだけを使えばすさまじくコンパクトなシステムが構築できる。いまでもQNXのホームページでデモシステムがダウンロードできる。これは、フロッピーディスク1枚に、システムとWebブラウザ（それもLynxみたいなシケた代物じゃねーぞ、ちゃんと画像も音声も使える）が収まる驚異的な代物。

アミーガファンは狂喜した。こ、これはすごい。こいつをベースに新アミーガOSが出たら、かつての栄光も夢ではない。考えただけでもあやしい可能性がうじゃうじゃわいてくるではないの。フロッピー1枚でそこらのマシンがアミ

マニアは健全性なんかで動いちゃいない。健全だけなら、そもそもだれもLinuxなんかに手を出さなかったろう

アミーガのA2000

ーガ化するとかできちゃわないかな。うー、もう二度とよみがえることはないと思っていた、あのアミーガの先進性とあやしさを、ふたたびうち立てられるかもしれない！

ところが。ここで（やっと）Linuxが出てくるのだ。この夏、正式アナウンスが出た。QNXとの話はご破算。次期OSは、Linuxをベースに開発する、という。

えええええええっ！　なにそれぇ？

何を考えておるのじゃ。アミーガのファンたちは激怒した。りぬっくすぅぅぅ？なんだぁ？あの古くさい、手垢にまみれた、既存の標準を実装しただけの新規性まったくなしの、モノリシックの、すでにミーちゃんハーちゃんにIBMだのオラクルだのまで手を出している、とりあえず動くだけが取り柄の、これ以上の発展も成長もない、あのどんくさいただのUNIXもどきぃ？　いまさら、いまさらだよ、いちばん最後になってノコノコやってきて、何をしようってんだ！　だいたいLinuxをベースにってどういうことだ、オープンソースにでもしようってのか！

不健全ゆえに我愛す

確かに、ここで出てきたLinux罵倒というのは全部本当だ。リーヌスだって、もうLinuxカーネルはピークにきちゃったよ、と発言しているし、Linuxの売りが先鋭性ではないことも、知っている人はちゃんと知っている。Linuxって、確かにシステム自体は動くだけが取り柄だよな。どうせ、いま話題のLinuxに便乗するとPR効果もあるだろう、というような腹はあるんだろう。なんか、そんなことでLinuxが利用されちゃうってのが、ずいぶんつまらないのだ。その後、話が進んで、コーレルとツルんでアプリケーションをつくってもらうとか。そうか、そういう話か。アプリケーション目当てね。でもいまさらアミーガでそんなオフィス系のアプリを使おうなんてヤツがいるのかしら。

でもそれはさておき、この一件があって、なんとなく最近ちょっとLinuxも飽きたなー、という一部の雰囲気がわかるような気がする。Linuxの最初のおもしろさってのは、こんなの動くの？　おおおっ、動いたぁ！　すげぇ！　しかもまともに！　というあやしさと驚きとうれしさの混じった感動だったんだよね。それがもう、動くのが当然になって、もっとあやしい世界が見たいよー、という感じなのだ。かつてLinuxそのものがあやしかった時代が終わって、あとは使うだけ。それはもちろん健全で望ましいことなんだけれど、マニアは健全性なんかで動いちゃいない。健全だけなら、そもそもだれもLinuxなんかに手を出さなかったろう。かつての魅力だった不健全さが、確かにもうなくなってきているなあ、それが寂しいというかつまらんというか。

もちろん、アミーガは存在自体が不健全といっていいような代物。今回も、事態はさらにわけのわからん展開を見せている。資源をつっこんだあげくにあっさり蹴られたQNXはカンカン。で、黙って引き下がるどころか、なんと今度は、ドイツのアミーガ用CPUボードのメーカーPhase5とつるんで、独自システムの開発をアナウンス！　おおおおっ！　Linux VS QNXだぁっ！　いいぞういいぞう！　Linuxめ、目にもの見せてくれるわ！　アミーガ・コミュニティは今日もあやしさをみなぎらせているが、さてどうなりますやら。

Profile

やまがた　ひろお

1964年東京生まれ。経済や文化面でさまざまな雑文や翻訳をものしており、コンピュータ分野ではLinuxやオープンソースがらみの翻訳で知られる。『伽藍とバザール』『ハロウィーン文書』訳者。マイナーだったはずのLinuxがいつのまにやらメジャーになって多少困惑気味。

隠喩としてのコンピュータ

「Perlとポストモダニズム」

文：豊福 剛
Text：Tsuyoshi Toyofuku

思わぬところで、「ポストモダン」という言葉に遭遇してしまった。Perlの作者ラリー・ウォールがポストモダン言語としてのPerlを語った、*Perl, the first postmodern computer language*と題されたテキストだ。これは、'99年3月にSan Jose Convention Centerで開催されたLinuxWorld Conference & Expoでの講演の記録である（http://www.perl.com/pace/pub/perldocs/1999/03/pm.html）。

'98年以降、Linuxはオープンソースを代表するものとして語られてきた。ラリー・ウォールがLinuxWorldで講演するのは、Perlもまたオープンソースを代表するソフトのひとつであるからだ。とはいえ、講演のすべてをPerlの布教活動に費やすのは、さすがに躊躇したのだろうか。ポストモダニズムとPerlを語ったうえで、Linuxおよびオープンソースについて語るという構成で展開されている。

それにしても、なぜ、いまどきポストモダンなのだろうか。日本では'80年代にポストモダン思想が流行し、消費されたものの、いまではポストモダンが使われるときのニュアンスは、かなり否定的なものになっている。ところが、アメリカではポストモダンは人文系の学問におけるひとつのジャンルとして確立しているようで、それこそインターネット文化論まで含むような裾野の広がりを持っているようだ。

様式としてのポストモダニズム

もともとポストモダンという言葉は、建築の世界で使われ始めたものである。建築批評家チャールズ・ジェンクスが『ポストモダニズムの建築言語』を発表したのは'77年なので、その歴史もけっして短くない。また、'80年代と'90年代とでは、ポストモダンを語るときのモチーフや背景が大きく変化してきている。それは、モダニズムをどのように規定するかによってポストモダニズムの意味が変わってくるからである。

建築の世界でポストモダニズムが主張された背景には、モダニズム建築に対する反発があった。モダニズム建築は、機能主義に特徴づけられる建築様式で、装飾をはぎ取り、抽象化された均質空間は、まさに近代工業化社会が必要とするものであった。そして、近代工業化社会から脱工業化社会への移行に対応して、ポストモダニズム建築が登場したのは歴史的必然とも言えよう。

モダニズムは進歩主義と同義であり、過去は否定すべきものとしているのに対して、ポストモダニズムは進歩主義への批判として、モダニズムが否定した過去や伝統への回帰を示す。ただし、単に過去の様式を再現するのではなく、モダニズム建築空間に、過去の建築様式を折衷的、コラージュ的に引用する点に手法上の特徴がみられる。モダニズムが、単一の原理に事象を還元し、一元化された体系や価値観を強制するのに対して、ポストモダニズムは、意味や価値の多元性を強調する。

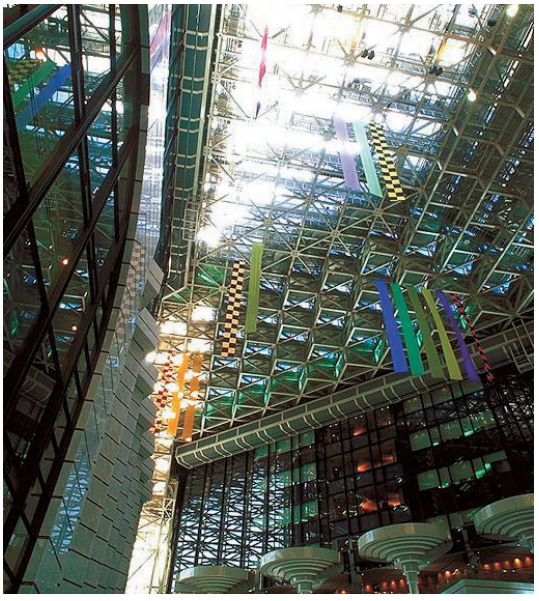
そういった意味では、「実用データ取得レポート作成言語」（Practical Extraction and Report Language）であると同時に「病的折衷主義のがらくた出力機」（Pathologically Eclectic Rubbish Lister）でもあるPerlに、建築やデザインにおけるポストモダニズム的傾向と共通する要素を見いだすことは難しくない。チャールズ・ジェンクスは、建築言語において、建築上の意味と生活上の意味の多重性を強調した。それと似て、ラリー・ウォールもプログラミング言語における形式上の一貫性と完全性よりも、プログラマーの思考のスタイルを重視している。モダニズムのプログラミング言語がプログラマーに思考のスタイルを強制するのに対して、ポストモダニズムのプログラミング言語であるPerlは、プログラマーの思考のスタイルに応じたプログラミングを許容する多元性がある、というわけだ。

モダニズムを特徴づける4つの教義

ポストモダニズムがモダニズムとの対比で語られる以上、プログラミング言語におけるモダニズムをどのように規定するのかが重要になる。ラリー・ウォールは、モダニズムを特徴づける4つの教義（カルト）を列挙し、それぞれの教義ごとにモダニズムとポストモダニズムを対比させている。その4つの教義とは、簡潔、オリジナリティ、生真面目、客観性である。

――簡潔の教義。モダニズムにおけるプログラミング言語は、何よりもミニマリズムの貫徹を目指していた。余計な要素を削ぎ落とし、必要最小限の要素で構成される簡潔さが何よりも優先される。しかし、こうした言語は、簡単な問題は簡潔に記述できるが、現実的な問題における複雑さを簡潔に記述することには失敗してきた。

プログラミング言語におけるミニマリズムは、還元主義と関係している。還元主義は問題を簡潔にする代償として、問題のコンテキストそのものを抹消してしまう。これに対して、UNIXおよびそのサブセットであるPerlは脱構築的である。 UNIX



（Linuxも含む）のコマンドはひとつの仕事しか実行しない（還元されている）が、パイプを使ってそれらを接着することができる。つまり、各要素の相互関連、コンテキストを示すことができる。さらに、Perlは複雑系の科学が木の葉の複雑な形の背後に単純な秩序を見るように、複雑さの中にある単純なものを記述できるというのだ（ほんとうかね？）。

ここでラリー・ウォールは、脱構築をモダニズムとポストモダニズムの架け橋として説明しているのだが、かなり違和を感じる。もともと脱構築（デコンストラクション）は、フランスの思想家ジャック・デリダの用語だが、それは哲学の歴史的、伝統的概念に忠実に内在的に仕事をしながら、他方で、哲学の概念体系が成立したときに隠蔽し、排除してきたものを明るみに出すことであった。UNIXやPerlを脱構築的と形容するのであれば、モダニズムが抑圧してきたものをモダニズムに内在しつつ示さなければならないだろう。単なる再構築は脱構築ではなく、それ自体還元主義の枠内にとどまるのではないだろうか。

――オリジナリティの教義。モダニズムにとって、盗作や模倣は罪である。 Perlはオリジナリティには囚われない。他の言語にある気に入った機能を、ためらわず、どんどん取り入れる。

――生真面目の教義。モダニズムはあまりにくそ真面目なので、自分を笑い飛ばすことができない。下手なコーディングで笑われることを恐れ、ソースコードを隠したままにするし、楽しみのためのプログラミングという発想がない。ポストモダニズムは、ユーモアが好きで、自分を笑うことを恐れない。キュート、ファンキー、キッチュであろうする。懐古趣味や感傷すら拒絶しない。

――客観性の教義。モダニズムのプログラミング言語は、できるだけ慣習を廃止しようとしたが、それは慣習を不可視なものにしただけだ。Perlは慣習や文化があることを恥じない。Perlのプログラムは、汚く書くことも、きれいに書くことも、どちらもできる。Perlそのものは、特定のプログラミングスタイルを強制することはない。

ラリー・ウォールによるポストモダン的パフォーマンス

「モダニズムは、解決すべき問題に焦点を合わせるのに対して、ポストモダニズムは、それよりもむしろ問題を解決する人物に焦点を合わせる」のであれば、ポストモダンをテーマとするスピーチそれ自体が、ポストモダンなパフォーマンスとして試みら

れなければならない。そして、この講演の詳細を追ってみると、笑える仕掛けがあちこちに施されている。ポストモダンなテレビ番組といえば「料理の鉄人」と「少女革命ウテナ」なのだし、モダニズムにおけるコンピュータの振る舞いとして示されるフレーズ、*I'm sorry Dave. I can't allow you to do that.*は「2001年宇宙の旅」のHALだし、キリストや12使徒をもじったくだりもある。最後では、C言語の後継はC++ではなくPerlである（BCPL→B→C→PLだから）と、こじつけたりしている。

このような語り口の面白さは十分にたんのうできるものの、残念ながら、Perlがポストモダン的言語であるという主張に比べると、Linuxおよびオープンソースについての主張は、説得力に欠けていることは否定できない。ただし、Linuxおよびオープンソースをポストモダニズムと関連づける着想そのものは興味深い。おそらくそれは、別の視点から検討されなければならないだろう。OSについては、GUIとポストモダニズムの関係が重要であると思われるし、またオープンソースについては、ポストモダンにおける知のあり方と関連づけて考察されるテーマであるはずだ。

オープンソースはモダニズムなのではないか

建築や美術、文学の世界で展開されたモダニズム批判は、さらに社会思想、政治思想の世界にまで、その射程を拡大していった。このようなポストモダニズムの台頭を、モダニストはどのように受け止めたのだろうか。ドイツの思想家ユルゲン・ハーバーマスは、ポストモダニズムを高度資本主義社会における諸矛盾を隠蔽する新保守主義の言説であると批判し、近代の基本原理である啓蒙の有効性が失われたわけではないと主張した。

一方、啓蒙の有効性を説くハーバーマスに対して、フランスの思想家ジャン・フランソワ・リオタールは、啓蒙が未来における人間の解放、救済という「大きな物語」に依拠しているとしたうえで、そのような「大きな物語」が無効となった危機こそ、ポストモダンであると反論した。

'80年代におけるこのポストモダン論争は、インターネット以後の現在のオープンソース運動の意味を考えるうえで、非常に重要な視点を提供している。パソコン、インターネット、そして現在のオープンソースが「大きな物語」に依拠しているとすれば、そこで語られる多くのモダニズム的言説は、ポストモダンにおけるモダニズムの亡霊に過ぎないのだろうか。

Profile

とよふく つよし

1962年生まれ。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。

ドクターShiotaの

ギョーカイSnapShot

PC業界今月の話題

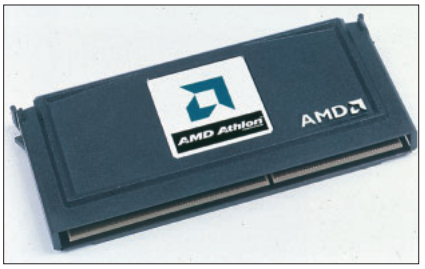
文：塩田紳二
Text : Shinji Shiota

7・13　Intel Mercedの設計を完了
7・14　IDT　x86互換チップビジネスから撤退
7・21　Apple iBookを発表
7・23　AOL vs. MS インスタントメッセージング対決始まる
7・23　Microsoft Millenniumのプレビュー版を一部テスターに公開
7・27　IBM Mylexを買収
8・ 9　AMD Athlonを正式発表

CPUビジネスの行方

Linuxコミュニティでも、多くのマシンに使われているインテルのCPU、我々はこれを俗に86系と呼んでいたが、インテルはIA-32と呼ぶ。どうしてそうなのかというと、今年末に登場するといわれているインテルの新しいアーキテクチャをIA-64と呼ぶからだ。

そのIA-64の第1号がコードネームMerced（マーセッドと読む）と呼ばれるCPUである。基本的にはこのCPUは、従来のIA-32とは全く違うもの（ただし、IA-32コードの実行機能は持っている）で、VLIW（Very Long Instruction Word）というアーキテクチャを採用する（インテルはこれをEPIC：Explicitly Parallel Instruction Computing・明示的並列命令コンピューティングと呼んでいる）。簡単にいえば、ガバッっと大きな命令を読み込んで、チョコチョコと小さな命令に分割して並行に実行するわけだ。



AMDが発表した新CPU Athlon

大方の予想としては、このMercedは普及するだろうといわれている。というのは、すでにいくつもの企業が採用を予定しているし、インテル自身もLinuxや他のOSなどに積極的に投資しているからだ。一応対抗馬としては、旧DEC（現Compaq）のAlphaと、IBM／MotorolaのPowerPCがあるが、どちらもVLIWは採用していない。

このうち、最近大きな動きが見えるのがPowerPCである。このCPUは、ご存じの通りMacintoshに採用されており、0.18やそれ以下のプロセスルールで有利だといわれる銅配線をすでに実用化している。最近登場したAppleのiBookにもこの銅配線を使ったPowePCが採用されている。

そういえば、このiBook、「男らしくない」と言われているが、Linux用ノートにすると結構いいんじゃないかって気がするんだけど、日本人には大きすぎるだろうなぁ。

そういえば、Compaqは銅配線技術に興味があるようで、Alphaチップの製造についてIBMと交渉中というウワサがある。どっちにしても、対IA-64ということではがんばってほしいものである。なにせ1社独占というのは、ユーザーにとって何もいいことがないからねぇ。

IA-32のほうでは、CyrixがVIAに売られ、IDTが撤退という状況だが、その中でAMDはようやく新アーキテクチャAthlonの発表にこぎ着けた。これは、ソフト的にはIA-32互換であるものの、バス部分は、Alpha用に開発されたバスアーキテクチャ（EV-6と呼ばれる）を採用しており、回路的にはインテルのものと互換性がない。チップセットなどもまったく別のものになるため、いままでのSocket-7互換のようなインテルの資産を使うものとはちょっと違う。

その意味では、互換という枠からは多少はみ出したもので「成り行きが注目される」製品である。コンパックなんかが採用して、ついでに同じCPUソケット（Athlonのソケットは、Pentium IIと同じSLOT1コネクタを使用する）に刺さるAlpha 21264チップ（ボード？）なんか作ってくれると面白いのだが……。

MSNはApacheで動いている？

Office2000を出荷して、あとはWindows2000の完成を待つばかりとなったMicrosoftだが、そのため大きなニュースもなく、ウワサや本業とは関係ないニュースが多い。

ちょっと面白いネタが、MSNもApacheを使っていたという話（リクエストしたページがない場合のエラーにサーバソフト名が表示されるので判明したという）。Microsoftは、自社のWebサーバであるIISやWindows NTなどについて、いかにも世界一というような話をしているが、そのMicrosoftの運営するMSNでApacheが動いていたというので、ちょっとしたニュースになった。

実際には、サイト運営の一部を外部の業者に委託しており、そこがNTを使っていなかっただけのこと。自社製品に関わる分野なので、もうちょっと気を使うべきだったかも。

これとは別にMicrosoftがWindows2000サーバをファイヤウォールの外側に置き、インターネット側からセキュリティを検証できるサービスを始めたところ、どうも、システムがおかしいらしく、ハッキングどころか誰もアクセスができなかったとか。この話、開始前からちょっと話題になっていて、クラッカー諸君が手ぐすね引いて待っていたらしいが、肝心のマシンが動いてないなら、誰も侵入できないよねぇ。

Microsoftの話によると「落雷」の影響とのことだが、アクセスが集中して過負荷状態であったことも事実らしい。まさか、動かさないのが最大のセキュリティっていうジョークじゃないだろうねぇ。

インスタントメッセージングソフト戦争

そのMicrosoftだが、この夏一番の話題は、AOLとのインスタントメッセージングソフト戦争である。ご存じの通り、AOLはNetscapeを買収したため、いまやブラウザ戦争はMS対AOLってことになっているけど、最近はこの話題も下火である（というのは、Netscapeの新しいバージョンが出てないからだ）。

その代わりに「熱く」なってきたのが、インスタントメッセージソフトである。

これは、サーバを介して、ユーザーがメッセージを送り合うためのソフトで、まあ、チャットソフトの一種。ICQがその火付け役だったが、この会社は、昨年AOLに買収されている。AOLはこれとは別にAOL Instant Message（AIM：Netscape Communicatorなどに付属している）というソフトを持っており、両者を合わせるといまやインスタントメッセージソフトの最大シェアを持つ（4000万とも7000万ともいわれる）。

これに対抗して、Microsoftが出したMSN Instant Message（MIM）はこのAIMと互換性があり、相互にメッセージのやりとりが可能なもの。ただし、この手のソフトは相手との接続にサーバを介する必要があるため、AOLのサーバを使う必要がある。AOLはこのサーバに手を入れ、MIMからは接続ができないようにしてしまった。それに対して、Microsoftはこのブロックを回避するバージョンを即座に公開した。

ブロック、回避バージョンの公開というサイクルがもう1回繰り返されたあと、AOLは別の手を打った。AppleやSunなどを味方に引き入れ、Earth Linkなどの大手プロバイダにAIMをライセンスし、グループを結成したのである。形の上では、このグループでインスタントメッセージの今後を検討しようというものだが、実際にはアンチMicrosoftの連合軍である。

これに対するMicrosoftの動きも速い。AT&Tや同じくAOLと互換性のあるインスタントメッセージソフトを開発したYahoo!、Prodegyと組んで、AOLに公開質問状（http://www.microsoft.com/presspass/press/1999/Jul99/AOLletter.htm）をたたきつけた。

というわけで、この争い、ブラウザに代わって、またPC業界を２分した争いになりそうな雰囲気である。しかも、MicrosoftはMIM発表前にLotusなどとインスタントメッセージに関する標準案をIETFに提出済み（6月25日）なのである。つまり、ユーザー数で押してくるだろうAOLに対して、「標準化」という方法で、プロトコルの独自性を崩そうという作戦である。

つまり、この騒ぎは、予め計画されたものだったのである。当然、Microsoftとしても、ある程度展開を読んでいたはずである。さて、これに対してAOL側が次にどう出るのか？　争いに巻き込まれたインターネットの標準化は？　しばらくのあいだは、これがブラウザ戦争に代わる大きな話題ではなかろうか。

だが、個人的には、どっちでもいいような気がする。だって、インスタントメッセージソフトとか、チャットとかって、いろいろやってみたけど、なんか生理的に受け付けないんだよね。なんか、ああゆう、バーチャルなつき合いって苦手。

では、今月はこのへんで……。

正しきタコへの道

初級Linuxer養成講座

コンピュータと人間の悲しい関係

Linuxをインストールしてみたけど、何に使えばよいのか分からない。X Window Systemは起動したものの、次に何をすればいいのかわからなくて途方に暮れてしまった。

ここではそんなユーザーにもLinuxが楽しく便利に使えるよう、最低限覚えておかなければならないもろもろを独断と偏見を交えつつお伝えしよう。ご心配なく。誰でも最初はタコ以下だったのだ。

文：竹田善太郎
Text : Zentaro Takeda

「相手の立場を思いやる」というのは、人間関係を円滑にするために最低限必要な要件だが、これと同じことが「人間とコンピュータの関係についてもあてはまる」といったら論理の飛躍のしすぎだと言われるかもしれない。ちなみにこれは、ありがちなネット上のエチケットの話などではなく、あくまで**コンピュータ対人間**の間でのやりとりの話である。

「ユーザーフレンドリー」とか「人に優しい」とかいうキャッチフレーズが、PC関連製品に限らず、家庭用のあらゆる機械製品に対して使われているが、これは「機械が人間の立場を思いやる」ことを意味しているわけだ。しかし最近、周囲を見ていると機械が人間を思いやることに**甘え**すぎて、人間が機械に対して配慮するということが、あまりに少なくなっているような気がする。たとえば、現在の家電製品などは滅多なことでは壊れないようにできているらしいが、マニュアルをろくに読まずに操作して、期待通りの動きをしないと「壊れている」とメーカーに文句をつけたり、機械本来の性能を無視する使い方を続けて、寿命を縮めるようなことをする例をよく見かける。

私はべつにメーカーを擁護しようとしているわけではない。逆に、最近の日本の工業製品は、一時の「過剰品質」への反動からか、目立って**品質が悪くなっている**ような気がする。数年前まではそんなことはなかったのだが、ここ2、3年、家電製品などで「初期不良」の製品をつかまされることが多くなった。モニターの映らないデジカメ、リモコンが効かないMDプレーヤー、音の出ないパソコン用スピーカ、塗装がべりべりとはがれるノートPC、ダイヤルボタンの一部が使えないPHS、データ通信コネクタが不良の携帯電話、カレンダーが止まったままのPDAなど、さまざまである。ほとんどは販売店での交換などで解消したが、面倒なのでそのまま使い続けているものもある。

さて、話が脱線してしまったが、ともかく人間が作った「機械」というものは、人間自身がそうであるように**完全無欠**ではなく、なんらかの欠点は備えているものだし、設計者が想定していないような使い方はできないようになっている。

コンピュータのハードウェアやソフトウェアもそうであって、一般の「機械」よりは柔軟性があるように見えても、やはり設計段階で組み込まれて（プログラムされて）いないことは、**逆立ちしても**できないのである。Linuxだって例外ではなく、システムにインストールされていないコマンドは初めっから使えないわけだし、コマンドが理解できないようなオプションや命令を与えても、**エラー**としてつき返されるのは当然である。これをもって**Linuxは使えない**とか**Linuxはユーザーフレンドリーではない**などと言うのは、はなはだ筋違いである。

ずいぶん昔に、「パソコンを買ったのだが、うまく動かない」と、友人から相談を受けたことがある。アパートを訪ねて、その状況を見せてもらったのだが、次のようなBASICプログラムが思ったように動かないという。

```
10 LET Y = X*2 + 2*X + 1
20 PRINT X
```

どうやら、彼はパソコンのBASICで二次方程式の一般解が簡単に得られるものと勘違いしていたらしい。プログ

ラミングの知識が多少でもあれば、このプログラムが常識はずれなことはすぐ分かるのだが、**コンピュータは万能と信じている人にとっては、なんでこんなこともできないの？**ということになるらしい。件の友人には、「BASICでは数値計算と簡単な文字列処理しかできない」ということをなんとか納得してもらった。しかし、これと同じような勘違いは、だれでもやってしまう可能性がある。Linuxだって、何ができて何ができないのかをよく理解していないと、とんだ恥をかくことになるのだ（もちろん、恥をかくことは悪いことではなくて、「恥を知らない」ことのほうが問題なのだが）。

コマンドの使い方を知りたい

さて、なんとも説教くさい始まりになってしまって、これで半分くらいの読者が離れてしまったかもしれないが、ここからが本題である。Linuxを使ううえで「何ができるか」「何ができないか」を知ることが大切なのだが、それにはいったいどうすればよいのか。

乱暴なアドバイスをすれば、**試行錯誤**するのが一番ということになるのだが、個人の使える時間は有限だし、かといって、分からないことがあるたびに、側にいる人間に質問するのも面倒だ。インターネットのニュースシステムやメーリングリスト、各種掲示板で質問するという手もあるが、**マニュアルを読めとか、マシン環境と使っているソフトのバージョンをすべて述べよ**といったような官僚的な答えしか得られないこともあるだろう。もちろん、このような仕打ちを受けるのは、質問の仕方が悪いからだが、初心者にとってこれらのメディアはあまり優しくないのが常である。きちんと下調べをしたうえで、必要な情報をつけて質問すれば解決することもあるので、**役に立たない**というつもりはないのだが、答えてくれる相手もそれを仕事にしているわけではないのだから、**頼ることはできない**と考えておいたほうがよい。

これより少しはましなのは、Web上の検索エンジンを利用する方法だ。「Yahoo!」などのインデックス型のサーチエンジンより、「goo」や「Altavista」といった全文検索型の検索エンジンが便利である。

たとえば、あるコマンドの使い方を詳しく知りたいとする。調べたいコマンドの名前や機能の名称などをキーワードにして検索すれば、なにかしら関連する情報が得られる。もちろん、常に目的の情報が得られるわけではなく、見当違いのWebサイトしか見つからないことの方が多いが、他人に訊くより簡単だし、キーワードを変えて何度でもトライできる。運良くそれらしいページが見つかっても、情報が古かったり、間違っていたりすることもあるので注意が必要だが、そのページ内の情報から別の情報源を探すヒントを見つけたり、うまくすれば、そのコマンドの作成者自身が情報を提供しているページが見つかることもある。

ところで、**Linuxはドキュメントが充実しているので、初心者向きだ**と言われることが多いが、これは鵜呑みにはできない。いわゆる「Linux」と呼ばれているものは、Linuxカーネルを中心に、無数のフリーソフトウェアが組み合わされてできているものだ。それぞれのソフトウェアごとに作者も違うし、当然ドキュメントの善し悪しも違ってくる。必要とするコマンドのドキュメントが、ディストリビューションに含まれていなかったり、あっても不親切なことは多々ある。そのような場合は、だれかが作成してWebなどで公開しているドキュメントや、市販の書籍などの資料を自分で探さなければならないわけだ。

とはいえ、たいていのコマンドには「オンラインマニュアル」が付属していて、Linuxをインストールすればこれらのマニュアルも一緒にハードディスク上にインストールされているはずである。Linuxマシンにログインして、コンソール画面から、あるいはX Window Systemを使っているのなら、ktermやxtermなどのターミナルエミュレータのシェルプロンプトから、manコマンドを使えば、これらのオンラインマニュアルを参照できるはずである。

たとえば、「ls」というコマンドのマニュアルを参照したい場合は、

```
$ man ls
```

と入力するだけでよい。**画面1**のようにlsコマンドの使い方などの文書が日本語、または英語で表示されるはずだ（日本語のマニュアルが付属しているかどうかは、Linuxのディストリビューションによって異なる。また、設定によっては英語のマニュアルしか表示されない場合もある）。

オンラインマニュアルが付属していないコマンドもある。しかし、そのような場合でも、コマンド自身にヘルプ機能が備わっていることがある。

たとえば、コマンドラインから次のように入力してみる。

```
$ ls --help
```

すると、lsコマンドのオプション一覧

などが表示されるはずだ（**画面2**）。Linuxで使えるコマンドのほとんどは、-hか、--helpのどちらか一方、あるいは両方のオプションに対応していて、コマンドの簡単な使い方を表示してくれるようになっている。使い方が分からないコマンドがあったら、とりあえずこの2つのオプションのいずれかを付けて起動してみるとよい。ただし、オンラインマニュアルの場合と同様に、すべてのコマンドがこの簡易ヘルプに対応しているわけではない。-hも--helpも受け付けないコマンドがあるし、受け付けても、1行ほどの簡単なメッセージしか表示されないこともある。

なお、「-h」が別の意味を持つコマンドもある。UNIX系の古いOSでは、コマンドに与えるコマンドラインオプションは、「-」（ハイフン）で始まるものがほとんどだったが、必ずしもすべてのコマンドで統一されていたわけではない。しかし、それでは使いづらいということで、UNIX系OSのインターフェイスを統一するための規格「POSIX」が制定され、「--」（ハイフンが2つ）で始まる統一されたコマンドラインオプションが決められた。「--help」というのはこのような「POSIX形式」のオプションなのだ。POSIXのオプションは、--helpや--versionなど、従来の「1文字オプション」よりも意味が分かりやすくなっているが、その分コマンドラインが長くなって入力が面倒になるという欠点もある。このため、POSIX形式のオプションと従来形式のオプションの両方をサポートするコマンドがほとんどで、POSIX形式については一部のオプションでだけ使えるようになっている場合もある。

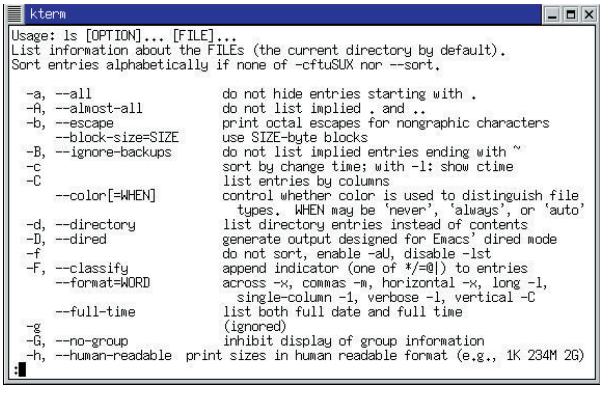
とはいえ、--helpやコマンドのバージョンを表示する「--version」オプションなどは、ほとんどのコマンドが対応しているようなので、あるコマンドについて調べたい場合は、まずこれらのオプションを試してみるべきだろう（もちろん、manコマンドも）。

どんなコマンドがあるのか？

さて、コマンドのドキュメントの探し方に、なぜこれほどこだわるのかといえば、自分のマシンのLinuxでどんなコマンドを使えるかを知っていることこそ、**Linuxを自在に使える**ようになる近道だからだ。「Linux」とひとくちにいっても、さまざまなディストリビューションが存在するし、たとえ同じディストリビューションでも、インストールの際にどのオプションやパッケージを選択するかによって、使用できるコマンドやアプリケーションはまったく異なってくるからだ。

同じLinuxをインストールしているはずなのに、自分のマシンだけうまく動かないといったトラブルに遭遇した場合、設定ファイルの記述ミス以前に、**必要なコマンドがインストールされていない**ことが原因である場合が、意外と多いものだ。Linuxのインストールの際に「カスタムインストール」など、自分でインストールするパッケージを選択するようなオプションを選択すると、膨大な数のパッケージの一覧が表示されて、面食らうと思う。このとき、**わけがわかんないから、入れなくてもいいや**といって、あるパッケージのインストールをやめてしまうと、あとから別のアプリケーションなどをインストールしようとしたときに、原因不明のエラーが出てうまく動作しないことがある。もちろん、最近のLinuxディストリビューションでは、パッケージ相互の依

画面1 「man ls」の実行結果
Red Hat LinuxやTurbo Linuxなどの日本語版のディストリビューションでは、一部のコマンドについては日本語のオンラインマニュアルが表示される。標準の設定ではオンラインマニュアルの内容は「less」コマンドを使って表示されているので、スペースキーと「b」キーなどを押せば、画面を上下にスクロールすることができる。



画面2 「ls --help」の実行結果
lsコマンドは「-h」オプションには対応していないので、簡易ヘルプを表示する場合は「--help」オプションを使う。--helpオプションで表示される簡易ヘルプはかなりの分量になり、1画面では表示しきれないので、実際には「ls --help | less」を実行して、表示内容をスクロール表示できるようにするとよい。

存関係を管理して、必要なパッケージがインストールから漏れることのないように配慮されているのだが、どうやらこれも完璧とは言えないようだ。

少し前のLinuxディストリビューションでは、すべてのパッケージをインストールすると、パッケージ同士がぶつかって不具合が生じることもあったが、最近のディストリビューションではそのようなことはなくなっているようだ。だから、Linuxをインストールするときは、後々のトラブルを避けるためにも、**すべてのパッケージをインストール**することをお勧めしたい。「ハードディスクがもったいない」という人がいるかもしれないが、数Gバイトのハードディスクが2万円以下で買える時代である。Linuxのディストリビューションをまるまるインストールしても、2Gバイト以下のディスクスペースしか消費しないのだから、ここはけちけちしないで全部インストールしてしまったほうがよい。そのうえで、自分のシステムにどんなコマンドがあり、どんなコマンドがないのかを頭の片隅で意識するようにすればよい。

コマンドの種類

Linuxのコマンドラインで使えるコマンドには、**シェルの内部コマンド**と**外部コマンド**の2種類がある。どちらも使い方に変わりはなく、コマンドラインでコマンド名とオプションを続けて入力するだけで実行できる。ユーザーがこれらの違いを意識することはない。ただし、使っているシェルが違うと、利用できる内部コマンドも変わってくることには注意が必要だ。シェルそのものの詳細については、おいおい解説していくことにする。

外部コマンドの実体は「コマンドファイル」の形で、ハードディスク上に存在している。コマンドファイルもスクリプトファイルからバイナリ形式の実行ファイルまで、さまざまな種類があるが、この違いについてもユーザーが意識する必要はない。コマンドファイルは、その用途などに応じて、ハードディスク上のいくつかのディレクトリ上にまとめて配置されている。コマンドファイルが置かれているディレクトリは、

```
$ printenv PATH
```

や、

```
$ echo $PATH
```

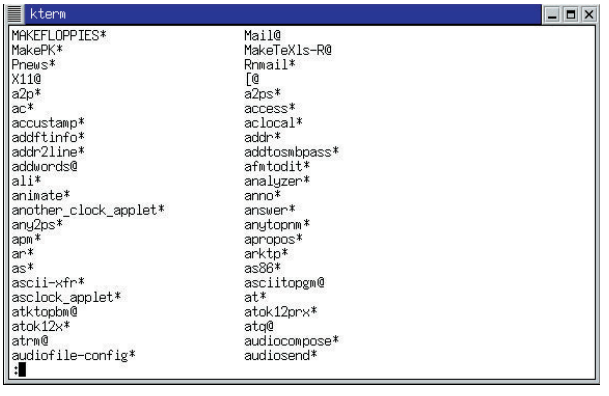
などのコマンドを実行すると表示される（**画面3**）。実際には、ここで表示される以外のディレクトリにも、さまざまなコマンドファイルが存在するのだが、**細かいことは気にしないで**今のところは「ここに置かれているコマンドが使える」という程度に考えておけばよい。

```
$ ls <ディレクトリ名> | less
```

というコマンドラインを実行すれば、指定したディレクトリ中のコマンドファイルの一覧を表示させることができる（**画面4**）。

Linuxをインストールしたら、まずはこのようなコマンドを使って、自分のマシンにどんなコマンドがあるのかを眺めてみるのがよいだろう。気になるコマンドがあったら、先に指南したような方法で、使用方法の資料を集めてみればよい。

画面3 「コマンドパス」の確認方法
Linuxではコマンドファイルの場所を「コマンドパス」で指定する。コマンドパスはPATHという名前の「環境変数」で指示することになっているが、これをコマンドラインで確認するには「echo $PATH」と入力すればよい。ちなみに、この画面はRed Hat Linux 5.2日本語版の標準設定時のコマンドパスなので、ディストリビューションやバージョンが違えば別の結果が表示される。



画面4 /usr/binの内容を確認したところ
コマンドパスの中に含まれていた「/usr/bin」ディレクトリの内容を表示させたところ。普通に「ls /usr/bin」とすると、大量のファイル名が表示されて確認しきれないので、ここでは「ls -x /usr/bin ¦ less」と入力してみた。ちなみに「-x」オプションは「表示結果を複数の列に並べて表示せよ」という意味のlsのオプションである。

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台
～Web制作の現場から～

当コーナーは、日刊アスキー Linux(http://www.linux24.com/)のLinux magazine出張店舗として今月から登場しました。今回はWebサイトの紹介を含めつつ、運営の裏話をいたします。本家Webサイトともどもよろしくお願いします。

「全ての人」を対象に

日刊アスキー Linuxは、Linuxに関する総合情報の提供を目的として創刊された情報Webサイトで、Linux magazineの、もうひとつのホームページとしての役割も担っている。キャッチフレーズは「ここに来れば、目的の情報が見つかる」。そのため、ニュースはもちろんのこと、他のWebサイトや書籍といった各種情報源へのリンク、イベントやセミナー情報などを日々豊富に提供している。

対象読者は「すべての人」。Linuxに古くから関わっているユーザーや開発者はもちろんのこと、ビジネスでLinuxを扱うことになって勉強している人、入門者、果てはLinuxにまったく興味のない人でも、読んで役に立ち、楽しんでいただければと思って作っている。こうしてでできあがったのが次のようなコンテンツだ。

ニュース

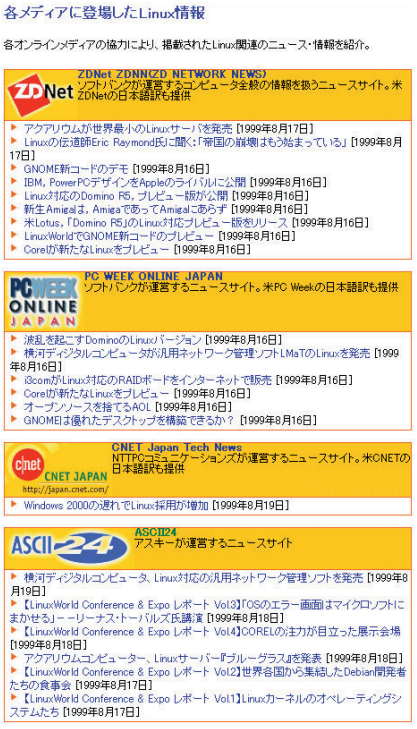
「Linuxに関するニュースをすべて網羅する」を方針としている（**画面1**）。とはいえ、ただニュースを発信していくだけでは面白くない。特に、取材ではなくプレスリリースを元にしてニュースを作成する場合、ただ発表をまとめただけでは、リリース発信元の企業のWebサイトにアクセスしてもらえばいい、ということになってしまう。したがって、可能な限り付加価値を付け、わかりやすくしてニュースを作成していくことにしている。

また、「網羅する」意味で、さらにひと工夫加え、「各メディアに登場したLinux」と題し、ZDNet、PCWEEK ONLINE JAPAN、 CNET Japan Tech News、ASCII 24の各サイトでLinuxを扱ったニュースへのリンクも張っている（**画面2**）。このページを見れば、インターネット上のさまざまなメディアにアップロードされたLinux情報を総覧することが可能なはずだ。

また、記事を可能な限り迅速に仕上げることも方針のひとつ。最近では、日本語redhat Linux6.0ベータ版をはじめとした試用レポートや、Lotus Domino R5 for LinuxのSneak Preview版登場速報、Oracle 8i for Linux登場速報、ImpriseのVisiBroker登場速報など、各種のスクープ記事をお届けしている。

では、実際の記事作成過程を追ってみよう。

画面1
「ニュースと読み物」ページ。過去の記事を日付ごとに一覧できる。



画面2
「各メディアに登場したLinux情報」。各サイトごとに特徴が出ていて面白い。

画面3
Stormixの池田氏からのアスキーアートで描かれた図版。

●ドキュメント―ニュース制作

StormLinuxは、カナダのStormix.incという企業が、Debian GNU/Linuxベースで作成したディストリビューションで、インストールの簡便さにプラスして、簡単で使いやすいリモートサーバ管理機能を持ったものだ。

取材は、Stormix.incからのニュースリリースメールが編集部に届いたことから始まった。海外の企業なのに、日本語のリリース、さらに作成者も日本人らしい。編集部では記事を書きつつ質問メールをリリース作成者に送ってみところ、すぐにStormix.incから返事が返ってきた。そこには、アスキーアートを駆使して描かれた図解つきの、Storm Linuxの技術解説がなされている（**画面3**）。とりあえずその返信をもとに記事を改訂、Webにアップロードした。

ふとカナダとの時差を考えると、先方はちょうど昼間。メールに書いてあった電話番号にかけてみることにした。余談だが、カナダの国際電話の国番号は06なので、実は担当編集者がプレスリリースを受け取った当初、大阪と勘違いして「かからない！」と電話機に八つ当たりしていた。

こうしてつながったのが、現在コラム「Storm is comming. Let's close the Windows!!」を執筆していただいている池田篤司氏である。この電話でのやり取りは、朝までにはStorm Linuxのニュース記事第2報として、電話インタビューの形でアップロードされた（**画面4**）。

さらに、池田氏のお話を聞くにつれ、コラムを書いていただいたら面白そうだとひらめき、こちらもお願いした。最初のプレスリリースを元にした記事作成から12時間あまりで、記事2本とコラムの執筆依頼まで話が進んでしまったわけだ。

こうした出来事は、Webならではのスピードだろう。

●ドキュメント―路上発見レポート

また、こんなこともあった。たまには編集部員も自宅へ帰るのだが、帰宅途中、たまたま渋谷の街で「LinuxPlaza」という看板を見つけたのである。「Linuxに関することなら何でも取材しよう」の方針どおり、まずは飛び込みで取材させてもらおうと、看板のかかっているビルに入ってみたのだが、それらしい入り口はない。早朝でもあったため、あまり期待はしていなかったのだが、それでも店舗の入り口だけは見つけようと、ビル内を怪しまれない程度にうろうろとしてみた。

周囲にいるのはカラスと掃除のおばさんだけだったので、おばさんに「あのLinuxと書いてある看板のお店はどこですか？」と聞いてみたところ、「隣は薬屋さんだけどねぇ」とちっとも要点を得ない。仕方なく帰途につき、そのまま検索エンジンで情報をあさり、ようやく連絡先を突き止めたのであった。

その看板の素性は、後日「渋谷の街でLinux！？（**画面5**）」という題名で掲載された。結局、実際の店舗は看板に書いてある場所から違うところに移動したとのこと。どうりで見つからなかったわけである。

画面4
Storm Linuxの電話インタビュー。画面3の図をもとに、技術担当加賀がその場で図を作成した。

実はこの原稿も、盛岡のホテルの一室で書いている。盛岡のISP「エクナグループ」の取材を行なった直後なのだ。たまたま編集部員が目を通した新聞記事に、日本アイ・ビー・エムのNetfinityにLinuxをインストールして使っているプロバイダが紹介されており、その場でWebページを検索、メールでアポをとったのである。

こうして、日刊アスキー Linuxのニュースコーナーは、時間、場所、状況を問わずアップロードし続けている。情報的な価値はもちろんのこと、こうしたライブ感覚のノリを随時提供していきたいと考えている。

リンク／データベース

ニュースで日々の動きを伝えるだけではなく、Linuxのユーザーが日々利用できるようなコンテンツも必要だろう。それがリンク／データベースである。早い話がリンク集なのだが、Linux magazine編集部にも協力いただき、かなりの手間をかけて作成されたものである。このコーナーも、ニュースと同様に「網羅する」ことを第一に作成している。Linux関連サイトリンク集や、ソフトウェアダウンロードリンク集、ディストリビューション情報など、Linuxを活用するうえで、欠かせないリンクが数多くある。

また、アスキーのオンラインコンピュータ用語集「Glossary Help」も、日刊アスキー Linux公開とともに語彙を拡充した。Linuxについてなんらかの情報を知りたい場合、このリンク／データベースコーナーにアクセスしていただければいいようになっている。日刊アスキー Linuxの中にあるコンテンツだけで、読者がすべてのLinux情報を知ることは不可能である。よって、こうしたリンク集を用意し、情報収集のナビゲートを行っていこうという主旨である。

ブックストア

Linuxと名の付く書籍を、とにかく集めてデータを提供しようという話から出発したのがブックストア。検索エンジンやその他さまざまなサービスを使って本の情報を集めた。出版社に連絡し、画像をいただき、目次原稿も送ってもらった。現在102冊が登録されている。

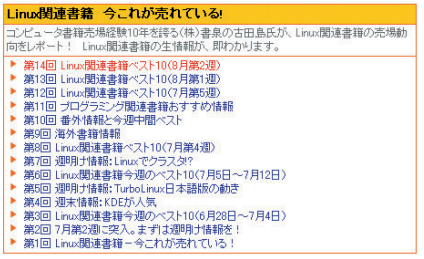
連絡作業中「いやあ、ウチはWebはまだまだで」というところから、「データはすべてWeb上で公開しています」というところまで、出版社によってインターネットへの対応は多種多様だと感じたのだが、さすがにLinux関連の書籍を扱っている出版社ばかりなので、Webページをまったく作っていない会社はなかった。総じて目次データまで公開している出版社が多く、カタログ的な情報収集ならば十二分な情報量が得られる。読者の方も、本をじっくり探したいのなら、ぜひ各出版社のWebページへアクセスすることをお勧めしたい。日刊アスキー Linuxの書籍紹介コーナーは、それら出版社ごとのLinux関連書籍を横並びにして、ほしい本が簡単に見つかるような仕掛けになっている。

さて、「ブックストア」というコーナー名は、インターネット上にカタログ的なページがある限り、「実際に本が買えるようになるのは当然」という思いから付けられたものだ。すでに日刊アスキー Linuxでは、アスキー刊行のLinux書籍を購入することができるようになっているが、これに加えて近日オープン予定のeコマースサイト「e-sekai」(http://www.e-sekai.com/)と提携してアスキー以外の出版社のLinux関連書籍も購入できるようになる予定だ。

さらに、「ブックストア」ではLinux関連書籍紹介に加えて、コンピュータ書籍業界では有名な、株式会社書泉の古田島氏による売れ筋情報も提供している。氏は10年以上にわたり、コンピュータ書籍売り場と関わってこられた方であり、秋葉原にほど近い書泉グランデのコンピュータコーナーを担当されていた方でもある。現在は埼玉県川口の「書泉ブックドーム」に勤務しておられ、週間ベストだけではなく氏がウォッチしている洋書などの情報もトピックとして送っていただいている。

こうしてでできあがったのが「Linux

画面5
「渋谷の街でLinux!?」。こんな街並みに、突如として「Linux」の文字が出現していたのだ。



画面6
古田島氏の精力的なレポートのおかげで、すでに14回を重ねる「Linux関連書籍　今これが売れている!」。

関連書籍 今これが売れている!」コーナーである（**画面6**）。UNIX系書籍で有益な本を探している方、ぜひコンピュータ売り場からのナマの声をお聞きいただきたい。

コラム

役に立つ情報だけではなく、読んで楽しいコンテンツも用意したい。しかも、ただ単に週に何回か掲載するのではなく、日替わりでいろいろな方に書いていただけたら面白いだろう。そう考えて「今日のコラム」はできあがった。Linux界、広くはコンピュータ界で活躍されるさまざまな方にご登場いただいている。

また、定例のコラムだけではなく、何か話題になることが起こったときは、話題の中心人物や詳しい方にお願いして、単発コラム「Follow-Ups」にて公開している。これまで、AMIGAが次期OSにLinux採用を決めた際には月刊インターネットアスキーの根岸智幸編集長に、Qt Version 2.0リリースの際にはQt/KDEの国際化や日本語化に取り組んでおられる高木淳司氏に、LinuxWorldが開催された時は、出展者側として池田篤司氏にご登場願った。

ALL ASCII Linux ISSUE

アスキーから発行されている雑誌のLinux関連記事を、HTML化して公開している（**画面7**）。いくらニュースやリンク集があるとはいっても、それだけでは物足りない。ぜひとも、もっと深い情報もお届けしたい。そこで、アスキーの各編集部の協力のもと、雑誌に掲載されたLinux関連記事を掲載することになった。現在アップロードされているのは、Linux magazine No.1全記事のほか、月刊アスキー、月刊アスキーNTなどの記事で、今後もコンテンツを随時追加予定だ。

特に、Linux magazineは丸ごと1冊PDFファイルで掲載するという大胆な試みであったが、結局、蓋を開けてみると、ALL ASCII Linux ISSUEに掲載されたコンテンツは、軒並み人気コンテンツとなっている。

ただし、人気なのはありがたいが、雑誌の誌面をそのままWebにするのは、きわめて手のかかる作業だ。雑誌のテキストをただ単純にHTMLにすればよいというわけではない。この誌面を見ていただいてもわかるが、雑誌の誌面はさまざまなパーツで成り立っている。たとえば、タイトル、見出し、本文、注釈、コラム、写真、図版、イラスト、キャプション、表などである。それらはデザイナーによって見やすい色や配置でレイアウトされる。これをどうやってHTMLに移植するか。

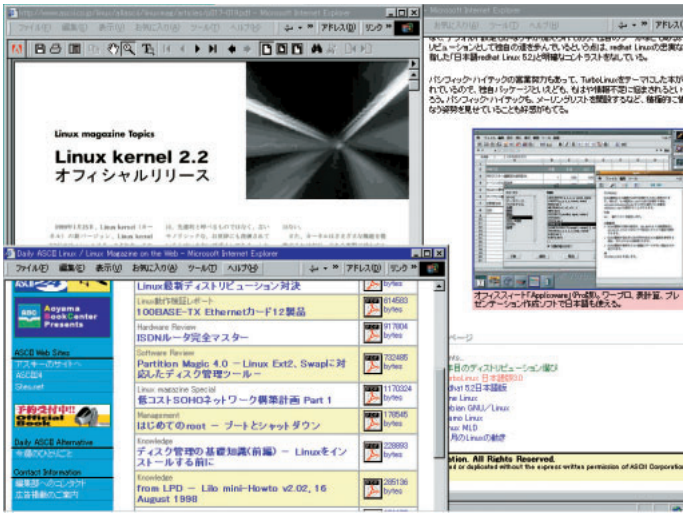
まず、表やコラムなどは、各雑誌、各コーナーごとに違うものを、HTML的に統一しなければならない。さらに、誌面の区切りとHTMLのページの区切りは、頭が痛い問題だ。たとえば、製品紹介ページなどは、そのまま雑誌の1ページ単位＝HTML1ページといったように置き換えることはできない。見開きで2ページにわたっている表はどうすればいいのだろう。そのままHTMLにすれば、見るに耐えない横長のWebページができあがってしまう。

現段階で、上記のような極端な状況には出くわしていないが、Webと紙の違いには、今後もその都度悩みながら解決していくしかなさそうだ。実際、ALL ASCII Linux ISSUEでは、まずは情報最優先ということと、アクセス速度の問題もあり、説明図版以外のデザインイラストは掲載していない。

雑誌と同じ原稿を使いつつも、雑誌とはまた違った見せ方や利点を追求していかねばならないのは、たいへんだがメディアの違いを乗り越える新たな試みとして楽しみでもある。

●

以上、代表的な日刊アスキー Linuxのコンテンツと、その裏舞台を紹介した。オープンして2カ月ほど、まだまだ改善すべき点は多々あるが、アクセスしていただければ、確実に面白く役に立つ情報がそろっていると思っている。今後とも、当サイトをよろしくお願いいたします。



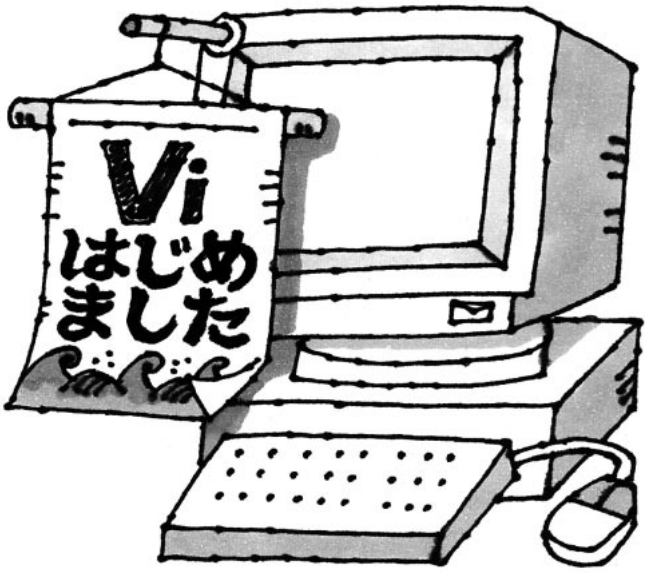
画面7
「ALL ASCII Linux ISSUE」。圧巻のLinux magazine全ページPDF一覧。ほかの雑誌はHTMLで提供している。

viバージンに贈る超入門講座

viはじめました

第2回　中級編
おサルなvi使いにならない法

Linuxユーザーアンケートによれば、よく使うエディタはviが70%でトップという驚きの結果。みなさん、さぞや最初は苦労したんじゃないでしょうか。vi入門講座第2回目の今回は、カット、コピー&ペーストと検索を中心に解説します。



illustration;Manami Kato

文：佐々木太良
Text:Taroh Sasaki

●習うより慣れてね

皆さん、残暑に負けずにLINUX／BSDしてますかー？ 前回（第1回目）はviのごくごく基礎のコマンドを紹介しました。第1回目でご紹介したようなコマンドは、手癖、いや脊髄反射になっていないと困るくらいのコマンドばかりですから、考えなくても出てくるまで練習を繰り返しましょう。

どんなエディタでもワープロでも（あるいはフォトレタッチソフトなどについてもいえることですが）、道具の中のコマンド（操作）は3タイプに分けられると思います。

①考えなくても反射的にできる操作
②ちょっと考えれば思い出す操作
③マニュアルやヘルプを見ないとわからないような操作。しかし、その操作一発で効果は絶大

巻末の表では①を『★★★』『☆☆☆』、②を『☆☆』、③を『☆』または星なしに分類しています。第2回目の今回も、中級とはいえ①の操作を主に紹介します（といいつつ☆な操作を紹介してウンチクをたれるのはやめられません）。

ある操作を手癖にするにはどうしたらよいのでしょうか。たとえばviなら、何らかのファイルを編集するとき、emacs(mule)ではなくviを立ち上げてみる、この心掛けが大事です。おっと、使い方が未熟で大事なファイルを壊しても、筆者および編集部は一切関知しませんよ〜。

●copy、cut&paste

いにしえのフォークグループに『Peter, Paul&Marry』ってのがありましたが、関係ありません。

エディタならこの**コピー／カット&ペースト**があるはずじゃないか、と思ったあなた、お待たせしました。思ってない人……コピー／カット&ペーストって何？ っていう人は、別途勉強してください。というのも、この概念はUNIX系のエディタやXウィンドウシステム全体にとどまらず、Windows系やMacOSのウィンドウシステムや、ありとあらゆるアプリケーションで使用できる（使用できないと不便だ）という概念だからです。

カットのコマンドは、実は前回紹介した[x]や[d][d]など、削除系のコマンドがすべてあてはまります。他のユーティリティ・OSなどが提供する類似の機能では、『カット』『削除』が明確に分類されていることがあります（何のためでしょうねえ）が、viでは削除したものはすべて**カットバッファ**に入ると思ってください。ただし挿入モード中に<BS>（端末の設定によっては<DEL>）で削除したものはこれに該当しません。また後の回で登場しますが、exモード・exコマンドで削除したものも該当しません。

ではカットしたものをペーストしてみましょう。viを起動して、適当な文字を打って、どこかで[x]と操作してください。また別の場所に移動して[p]を

打ってください（**図1**）。削除した文字が貼り付けられました。

また別の例として、どこかの行で[d][d]と操作し、移動して、別の場所で[P]と操作してください（**図2**）。削除した行が貼り付けられたはずです。**行単位で**カットされたので、カーソルが行の途中にあっても関係なく**行単位で**挿入されたのがおわかりでしょうか。また、[p]と[P]の違いについて理解してもらえたでしょうか。

あれ？ 賢明なあなたなら気付きましたね。

「viのカットバッファって、文字が入るものなの？ 行が入るものなの？」

答えは**『どちらも入る』**です。カットバッファに『文字が入る』場合は簡単です。emacs（mule）やWindows系のメモ帳など、世の中の大半のエディタのカット＆ペースト過程はこれだからです。カットバッファに『行が入っている』場合については注意してください。たとえば**図3**のようなケースで、カットバッファの内容が『文字』である場合は（a）のようになりますが、『行』である場合は（b）のようになります。

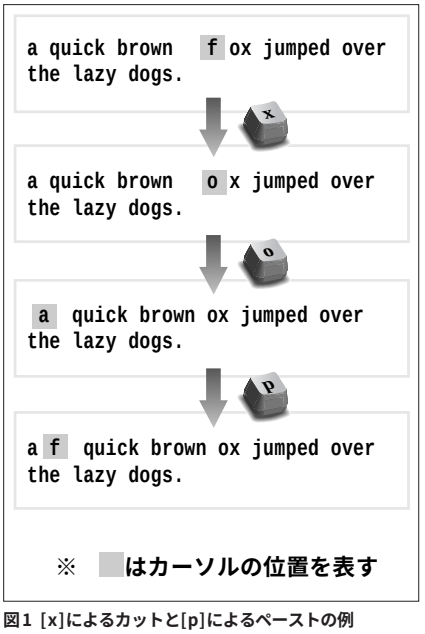


図1 [x]によるカットと[p]によるペーストの例

だからといって、カットバッファの内容に常に気を使っていないとviが使えない、というものではありません。むしろ普通のエディタのカット＆ペーストに加えて『行単位でのペースト』ができるからviは便利なんだ、といえましょう。

●数字でポン

「でも1行とか1文字とかのカットを覚えたって、役に立たないよ」。あ、そうですね、すみません。ですが、viコマンドの前に数字を付けると、その回数だけ繰り返すことができるのです。

また何かテキストファイルを作って（ちょっと長いほうがいいですね）、適当な場所で[5][x]とか[3][d][d]とか操作してみてください。また別の場所に移動して、カットバッファの内容を[p]してみてください。実にわかりやすいですね。

「あのさぁ、範囲を削除したいときとか、数えないといけないの？」あ、あはは。その通りです。以後の回でご紹介する組み合わせ操作を利用すると、『（行内で）Aという文字までを[x]する』（[d][t][A]）とか、『hogeで始まる行までを[d][d]する』（[d][/]^hoge<Enter>）とか、あるいは『カッ

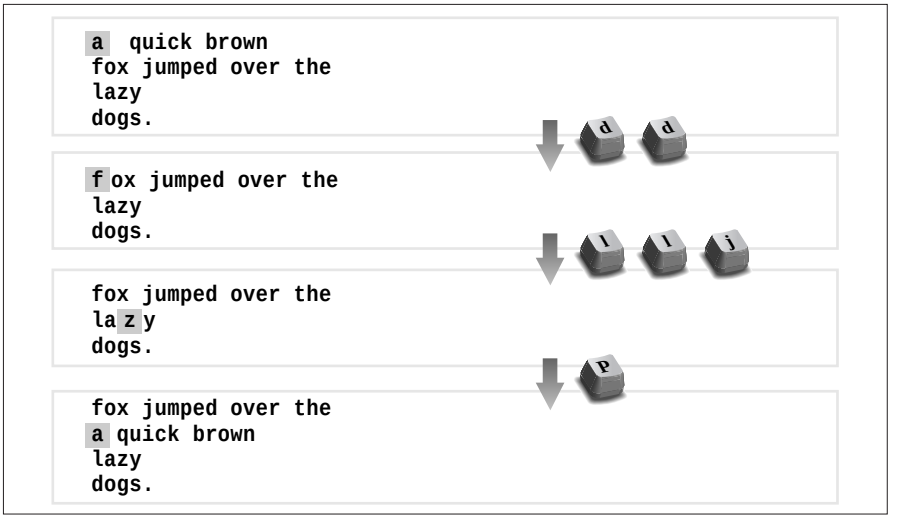


図2 [d][d]によるカットと[P]によるペーストの例

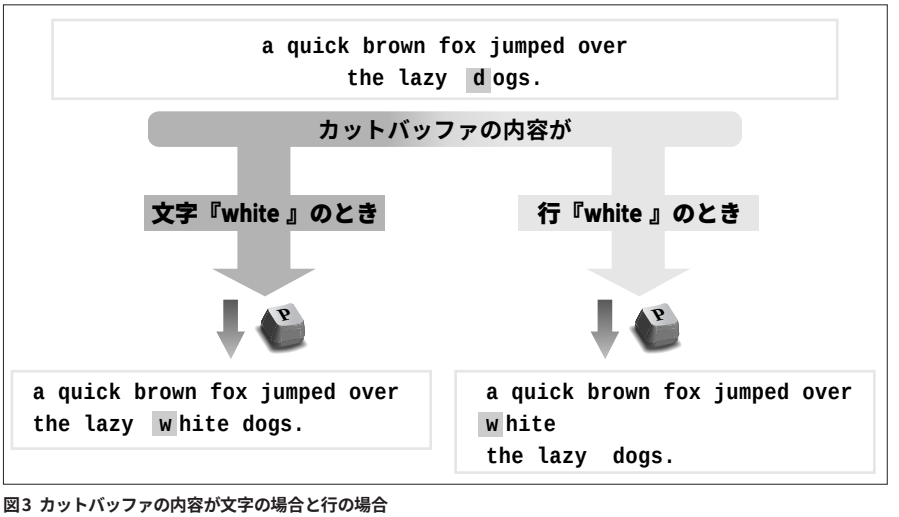


図3 カットバッファの内容が文字の場合と行の場合

コに対応するカッコ閉じまで削除する』([d][%])とかいう操作ができるのですが(**図4**)……。実際上は、私なんか(vi歴15年。しつこいですね)も画面上で目で数えながら[7][d][d]などとすることも多いです。

さて、繰り返しはさまざまな局面で使えます。たとえば[7][2][i]-<ESC>と操作してみてください(びっくりしました?)。私はメールや文章中で罫線もどきを引くときによく利用しています。でも逆にいうと、数字のキーにうっかり触れると、とんでもないことが起きるということです。たいていのviコマンドは数字と組み合わせられるということと併せて、覚えておいてください。

●コピーはどこへ行った?

そうですね、カットをご紹介しておいてコピーは紹介していませんでした。まあカットしたものをその場でほいっと貼り付けると元通り、なおかつペーストしたものは依然カットバッファに入っています(何回でもペーストできます)ので、**図5**のような操作でコピーはできるんですが…。

意地悪しないで書きますと(笑)、[y][y](yankの略だそうです)で[d][d]と同様に、しかしカットはしないでカットバッファにコピーすることができます。もちろん数字コマンドも使えるばかりではなく、(viのバージョンによって違うようですが)ほぼ[d]と操作するところはすべて[y]で置き換えることができるようです。この応用性は、後に紹介する組み合わせコマンドで活きてくるでしょうが、特別な場合を除きこの後は[d]はすべて[y]で置き換えることができる、というのは紹介しませんので覚えておいてください。

ところでカット&ペーストを利用すると、ある範囲の移動ができますね。viモードのコマンドでは特別『ある範囲を移動』というのは用意されていません(exコマンド・exモードにはあります)。**図6**のような移動はどう操作したらよいでしょうか?(答えは174ページ)。

●検索あってこそ

エディタ(ワープロでもそうです)の2大機能、というか、覚えているかどうかで初級者か中級者かの区別ができる2大要素、それはコピー、カット&ペーストともう1つ、検索&置換というのがあります。

昔話になりますが(音響さん、エコーお願いします)、MS-DOS時代のPCのベンチマークテストで『一太郎&ロータス スクロールテスト』というのを見たことがあります。最近は専用のベンチマークプログラムも発達して、ベンチマークテストで良い値をたたき出すが実用的には遅い、ということはあまりないように思いますが、当時としては『最も現実的なマシンの使われ方に近い』ベンチマークとしてそれが行われていました。

で、私があきれたこととは、くだんのベンチマークテストとは『一太郎

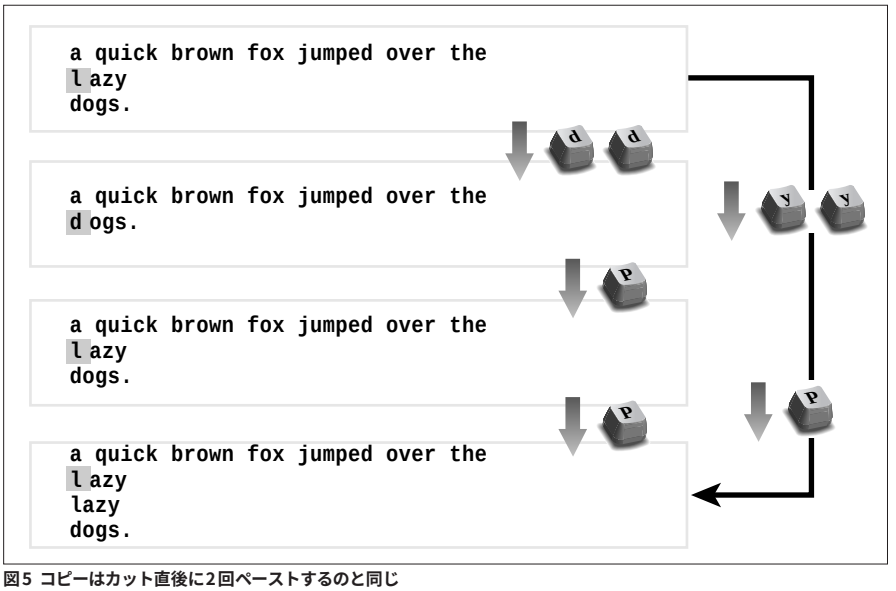


図5 コピーはカット直後に2回ペーストするのと同じ

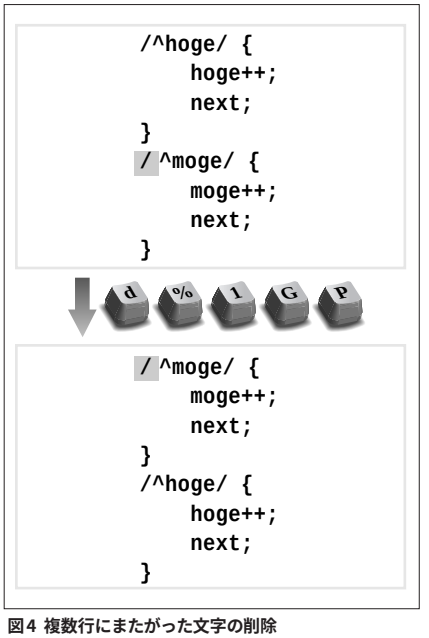


図4 複数行にまたがった文字の削除
※[d][%]は対応するカッコ閉じまで削除

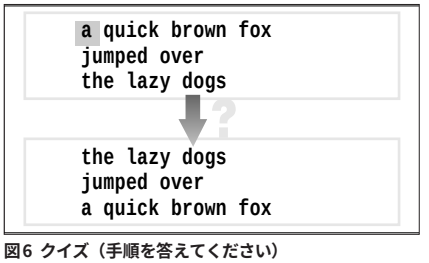


図6 クイズ(手順を答えてください)
※答えは174ページ

（など）で数百〜数千行ある文書を頭からお尻までスクロールさせ、それで何秒かかるか、というものだったことです。つまり、文書の目的の場所に到達するのに、『大多数のユーザー』にとって『最も現実的』な移動方法はスクロールだ、ということに、私は膝の力が抜ける思いがしたものです(うそ)。

viでこんなことをしたらどうでしょうか。たとえば、

```
% man csh > csh.man

% vi csh.man
```

として表示された文書を[j]（押しっ放し）でスクロールしてください。どんなもんです、遅いでしょう(笑)。

現在はXなどのウィンドウシステム上で高速に表示(画面制御)できているから、これでもまだいいほうなのです。かつてのTTY端末（つまりキーボードと画面です）は、CPU本体と300bpsなり1200bpsのモデム、あるいはせいぜい9600bpsなりのシリアル回線で接続され（56kbpsモデムの数十分の1ですね)、その速度で文字が書かれていたわけです。普通に1画面を文字で埋めるだけでもかなり時間がかかるわけですから、たとえ[^F]（後に触れます）で1画面ずつ移動したとしても、スクロールというのはえらく時間がかかる操作なわけです。

ましてや、目的とする文字列を目で追いながらその場所を探す、などというのは、『本来人間がやるべきではない』操作を人間がやっているわけです。

現在のPCは画面表示デバイス（VRAM）は100Mバイト／秒などというバスでCPUに直結され、巨大なCPUパワーがあり余っているのをよいことに、重いワープロソフトでスクロールが速い速い速いなどと言って悦び、あげくの果てに機械のシモベとなって文字列検索を人間が行い、OA病だドライアイだとは何事かっ、くわっ（©スタパ斉藤）……とかいうつもりはありませんが、ちょっとした操作を覚える（ワープロソフトでももちろん検索とか指定行にジャンプなんて機能がないわけないですね）ことをサボったばかりに、おサルなままでいるのは情けないではないですか。

というわけで、viはそもそもおサルな使い方ができないというハンディを背負っているがために、viユーザーにおサルは少ないといえるかもしれません。

おっと、前振りが長くなりましたが検索です（笑)。

後方検索は、[/]の後に検索したい文字列を入力し、<Enter>（または<ESC>）です（**図7**)。なお<Enter>の代わりに<ESC>が使えるというのはexコマンド同様なので以下省略します。最下行に検索する文字列がエコーバックされます。exコマンドモードや挿入コマンドと似ていますね。

逆に前方に検索するのは『[?]文字列<Enter>』です。

そして、前回と同じ方向に検索するのは[n]、逆方向に再度検索するのは[N]です（**図8**)。

前回と同じ文字列で後方検索するのは[/]<Enter>、前方検索するのは[?]<Enter>です。この2つは[n]や[N]と似ていますが、前者が方向を指定しているのに対して、後者は『前回の検索方向(ただし[n]と[N]以外)と逆かどうか』を問題としています。ややこしいですが、自分でもやってみて理解してください。通常は同じ方向の検索を次々と連続して使うことが多いため、[n]や[N]での検索の向きが一定していないというのは、ややこしいようで実は非常に直感的なのです。

●正規表現?

検索するときに入力するのは『文字列』と書きましたが、実は**正規表現**というものなのです。正規表現は、ある（有限の）表現で無限の文字列の集合を表現する……あー小難しいな（笑)、とにかくそんなものなので、『これこれのパターンをもった文字列を検索する』場合にとても役立つものなのです。

といいつつ、この正規表現のすべてを解説するには、本1冊の分量の説明が必要でしょう（うそです)。いずれにしてもここで正規表現を解説するわけにはいかないので、自分の思った通りのパターンを検索する（この後の回で登場する置換にも必要です）ためには関連書籍などを参照していただくとして（本誌前号No.3の81ページ『賢く使うUNIX』にも解説があります)、ここでは代表的な検索パターンの例を表1にまとめてみました。

注意していただきたいのは、正規表現にもバージョンがあって、awk（gawk）・sed・perlやvi・emacs（mule）の各バージョンによって細部に差異があるということです。さらにviによっては、正規表現のバージョンを切り替えられるものまであります。

重要かつ広範囲な正規表現による置き換えなどを行う前には、viのmanページ（マニュアル）を見るのももちろ

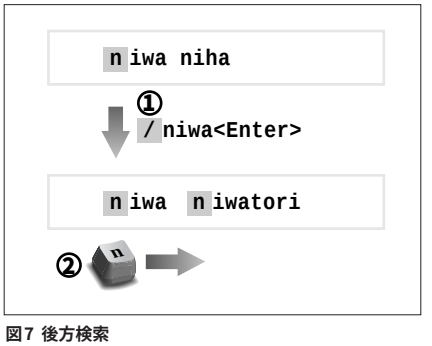
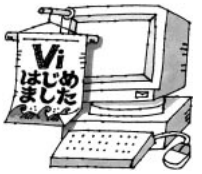


図7 後方検索



ん大事ですが、あるいはもう1つviを立ち上げて、そちらで簡単なパターンでテストしてみてもよいでしょう。

●カーソルを遠くへ移動

検索ネタのついでに、カーソルを一気に移動させるもう1つの方法、行番号指定ジャンプをお教えしましょう。C言語のプログラミングやPerlでCGIを書いているときなど、コンパイラやインタプリタが『何行目にエラーがあるよ』と教えてくれることがあります。そのときに一発でそこにジャンプできなくてはエディタとして失格です。

これには2つの方法があります。

- **[:]数字<Enter>**
- **数字[G]**

前者はexコマンドで数字のみをタイプするとその行にジャンプするというものです。数字の代わりに『$』を指定するとファイルの最終行にジャンプします。ここで『$−2』などと指定すると、ファイルの最後から2行目にジャンプできます（あまり使い道はないかもしれませんが、[G]コマンドでは同様のことはできません）。

後者は[G]コマンドを数字で繰り返している、とみなすことができます。さて、では[G]のみではどうなるでしょうか？（やってみてください）

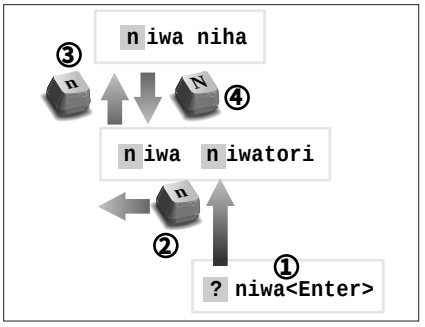


図8 検索方向の変更

[G]から連想されるからでしょうか、『今どこの行番号にいるか』を表示させるのは、[^G]（Ctrl＋G）というコマンドです。このとき最下行には行番号だけではなく、編集中のファイル名やファイルが前回セーブして以降、変更されている（ダーティ）か（modified）、などの情報も表示されるのでなかなか有用です。

さて、プログラマーの中には、常に行番号が出ていないと気持ち悪い、という人がいます。この場合は、exコマンドで[:]set nu<Enter>とすることで行番号表示モードになります。解除は[:]set nonu<Enter>です。このモード設定コマンドは大量にあるので、後の回で有用なものをまとめて解説しましょう。私は個人的には、行番号が表示されていると行番号と正味のファイルの中身（行番号そのものはセーブされません）の区別がつきにくくなること、C言語などは（昔のBASICなどとは異なり）せっかく自由構文規則で、行番号という『場所』に縛られていない文法になっているのに失礼だ（笑）という理由で、行番号表示モードは嫌いです。まあこれは宗教の問題だと思いますが。

●おまけ：複数ファイルの編集

前回の最後に、viの編集ファイル名指定規則について学びました。ところでmuleなどでは複数のファイルをじゃんじゃん開いて次々編集して、場合によっては並べて編集して見比べたり引用したり、ということができるのに、viはいささか原始的に感じられます。

というわけで最近の拡張バージョンのviでは、さまざまな複数ファイルの取り扱い方法が拡張されてきてはいるようです。ただし古典的な方法（つまりどんなバージョンのviでも必ず使える）でも、複数ファイルを渡り歩いての編集というのは十分できます。

○初めから複数ファイルを編集するつもりの場合

```
% vi hoge1 hoge2 hoge3
```

あるいは、

```
% vi hoge*
```

などとして複数ファイルを編集対象として指定しておくことができます。1つのファイルの編集が終わったら、（ダーティーであれば[:]w<Enter> でセーブして）exコマンド[:]ne<Enter>で次のファイルに移ることができます。

○別のファイルを編集したい場合

```
% vi hoge1
```

としてhoge1を編集中に、[:]ne hoge2<Enter>とするとファイルhoge2を編集することができます。この場合もあくまでviを終了して、

```
% vi hoge2
```

『hoge』で始まる行	^hoge
『moge』で終わる行	moge$
『hoge』『moge』『poge』にマッチ	[hmp]oge
『hoge』『moge』『poge』…など	.oge
『h』『hm』『hmmm』などにマッチ	hm*
『hm』『hmmm』などにマッチ	hmm*
『hoge』『hoe』『hige』などにマッチ	h.*e

表1 正規表現のいろいろ

したのと同じようなものなので、ダーティーであれば保存してから上記を実行しないといけません。

なお、それだけならviを終了して、シェルからまたviを起動したほうがよさそうですが（ワイルドカードによるファイル名の指定など、便利なことが多いですからね）、シェルとviのバージョンによってはvi上でワイルドカードを使ったファイル名の指定ができることがあります。また、ここから複数ファイルの編集（前項）に移ることもできます。

○交互に2つのファイルを編集したい場合

今、1回viを終了して別ファイルを編集しているのと同じようなもの、と書きましたが、実は直前に編集していたファイルに移りたいときは[:]ne#<Enter>と一発で戻ることができます。これを繰り返すと、2つのファイルを交互に編集することができます。なお、これもダーティであればセーブしてからでないと直前ファイルの編集には移れません。

ところでおまけの最初に『どんなviでも使える複数ファイルの編集』と書きましたが、ここまでの挙動もviのさまざまなバージョンによって細部が異なっています（たとえばカットバッファに入ったものが別ファイルの編集でもペーストできるか、[.]による繰り返しコマンドは別ファイルの編集でも有効か、など）。ですから1回は自分でviを立ち上げて、複数ファイルの編集のいろいろなケースを実行してみてください。

○おまけのおまけ：自動セーブ

いささか危険な場合もあるのでご紹介するのはためらわれるのですが……ここまでダーティな場合はちゃんとセーブしてね、と書いてきましたが、ダーティであれば強制的に保存してから終了や次のファイルの編集に移るようにviのモードを設定することができます。

exモードで[:]set aw<Enter>（解除は[:]set noaw<Enter>）です。

なお強制的に**編集を放棄**して次のファイルの編集に移る場合は、ご想像どおり[:]ne! ファイル名<Enter>（など）です。

●おわりに

だいぶエディタらしいことができるようになってきたと思いませんか。しかしここまででではまだ、「Windows系のメモ帳なんかのほうが便利だよ」という声も聞こえてきそうです。次回は中級編の続きとして、今回取り上げなかった移動系コマンド、そしてちらほら顔を見せている組み合わせコマンドについて見ていきましょう。それまでに十分、今回のコマンドを練習しておいてくださいね。ではhappy viing!

※図6の答え
[d] [d] [p] [j] [d] [d] [k] [P]
（最短手順は [2] [d] [d] [p] [d] [d] [p]）

コマンド一覧 （今月取り上げたもの）

重要度:
★★★ = 覚えるまではviを起動しない方がよい
☆☆☆ = 知らなきゃ死ぬでしょ。
☆☆ = 知っていると編集が速くなる。
☆ = 知っていると自慢できる。
(なし) = 知ってもあまり自慢できない。

※ 重要度は筆者 (ちなみにvi歴15年)の主観です。人によっては操作の宗派、いや流儀が異なる場合もあります。

● 今月取り上げたvi コマンド

コマンド	重要度	動作
p	☆☆☆	カーソルの後ろにペースト
yy	☆☆	1行コピー
P	☆☆☆	カーソルの前にペースト
数字	☆☆☆	直後のコマンドを数字の回数だけ繰り返す
/文字列<Enter>	☆☆☆	後方検索
/<Enter>	☆	後方に再検索
?文字列<Enter>	☆☆☆	前方検索
?<Enter>	☆	前方に再検索
n	☆☆☆	前回と同じ方向に再検索
N	☆☆☆	逆方向に再検索
数字G	☆☆	指定行へ
^G	☆☆	行番号その他の情報表示
G	☆	最終行へ

● 今月取り上げたexコマンド

コマンド	重要度	動作
数字	☆☆	指定行にジャンプする
$	☆☆	最終行にジャンプする
$-2		最後から2行目にジャンプする
ne(xt)	☆☆	次のファイルを編集
ne(xt) ファイル名	☆	『ファイル名』を編集
ne(xt)#	☆	直前のファイルを編集
ne(xt)! (同上)	—	現在の編集を放棄して（同上）
set nu(mber)	☆	行番号表示モード
set nonu(mber)	☆	行番号非表示モード
set a(uto)w(rite)	☆	自動セーブモード
set noa(uto)w(rite)	☆	非自動セーブモード

※ exコマンドに入る<:>と最後の<Enter>（または<ESC>）は省略してあります
※ ( )内は省略可能です
※ nのみでneと同じ動作をするviもあります

賢く使うUNIX

これであなたもスマートなUNIX使い！

Illustration:kame

ワイルドカード

Linuxの標準シェルであるbashのコマンドラインを中心として、Linuxの便利な使い方について紹介していく連載。今回は、コマンドラインでファイル名を指定するときに使う「ワイルドカード」と、シェル変数の内容を利用する「変数展開」の使い方について紹介する。

文：大池 浩一
Text : Kouichi Ooike

今回のお題

複数のファイル名をまとめてリネームする

ワイルドカードは、複数のファイル名をまとめて指定するための特殊文字（「＊」と「?」）のこと。MS-DOSのコマンドラインでも使えるのでご存じの方も多いだろう。しかし、MS-DOSとLinuxではワイルドカードの実現方法に違いがあるため、いくつか注意しなければならない点がある。また、文字のリストや範囲を指定する「セット構造」のように、MS-DOSのコマンドラインでは使えない便利な記法も用意されている。

また、シェルをプログラミング言語的に使ううえで外せない機能のひとつにシェル変数がある。これは、シェルで利用するさまざまなデータを保管する変数で、ユーザーが自由に値を設定できる。こうしたシェル変数の内容をコマンドラインやシェルスクリプトで参照する「変数展開」という仕組みが用意されており、条件判断（if／else）や繰り返し（for、while）といったシェルのフロー制御構造と組み合わせることで、より複雑な処理が可能だ。

シェル変数は、UNIX系OSのシェルに共通の仕組みだが、Linuxの標準シェルであるbashでは、変数展開の際に値を加工する手段が豊富に用意されている。このため、Perlなどのツールの助けを借りることなく、シェル単独で文字列の置換やパターン照合を行える。なお、これらの処理は、bashのバージョン2.0以降でないと利用できないので注意されたい。

MS-DOSのそれとは一味違うワイルドカード

ワイルドカードは、ファイル名の一部しかわからない場合や、複数のファイル名を一度に指定したい場合に使用される特殊文字のことだ。基本的なワイルドカードには、任意の1文字にマッチする「？」と、0文字以上の任意の文字列にマッチする「＊」がある。UNIX系OSのシェルの場合、「？」の使い方はMS-DOSと同じだが、「＊」はより柔軟性に富んでいるので、最初にそのあたりを説明しよう。

まず、「＊」だけで拡張子（ファイル名末尾のピリオドに続く文字列）の部分までマッチする。UNIX系OSでも拡張子という概念はある程度有効だが、シェルのワイルドカードは拡張子を特別扱いしないのだ。たとえば、README とREADME.txtという2つのファイルがある場合、

```
$ ls README＊
```

とすると両方とも表示される。なお、ファイル名の大文字と小文字は厳密に区別されるので、この例の場合readmeというファイルは表示されない。

もうひとつの違いは、「＊」の後ろに続けて書いた文字列がちゃんと考慮されること。「＊」以降の文字は一切無視されるMS-DOSに比べ、こちらの仕様のほうが合理的であることはいうまでもないだろう。たとえば、

```
$ ls RE＊ME
```

とすると、「RE」で始まって「ME」で終わるファイル（READMEやRENKONtoEDAMAME、REMEなど）が表示される。このため、MS-DOSではすべてのファイルにマッチする「＊.＊」という表現は、UNIX系OSではピリオドを含むファイル名にしかマッチしないので注意されたい。

●文字のリストや範囲を指定するセット構造

Linuxの標準シェルであるbashには、MS-DOSのコマンドラインでは利用できない便利な「セット構造」も用意されている。これは、文字列のリストや範囲を大カッコ「[」と「]」で囲んで記述すると、それらのどれか1文字にマッチするというもの。

文字列のリストは、そのまま文字を列挙すればいい。たとえば、

```
$ ls [abcdefg]＊[02468]
```

とすると、「a」から「g」のいずれかで始まり偶数で終わるファイル名が表示される。簡略化のため、文字コードが連続している部分は「-」で範囲指定できる。つまり、先頭部分のセット構造を「[a-g]」として、

```
$ ls [a-g]＊[02468]
```

としても同じ結果が得られる。末尾の「[02468]」のほうは文字コードが連続していないので、「[0-8]」と書いてはいけない（1、3、5、7にもマッチするため）。

ちなみに、「[a-z]」ですべての英小文字、「[a-zA-Z]」ですべての英字をカバーする。こうした文字範囲はよく使われるので、特別に「[:文字クラス名:]」という表現方法が用意されている。たとえば英小文字は「[:lower:]」、英字は「[:alpha:]」となる。実際には、これを文字セットの大カッコの中に記述するので、

```
$ ls [[:alpha:]]＊
```

のように大カッコが2重になる。もし、「[:alpha:]＊」としてしまうと、「:」「a」「l」「p」「h」「a」で始まるファイルにしかマッチしない。

場合によっては、含まれる文字を直接記述するよりも、含まれない文字を記述したほうが簡単なこともある。そこで、セット構造の「[」のあとに「^」か「!」をつけると、それ以降に指定されている文字のリストや範囲を除く1文字にマッチするようになっている。たとえば、英字以外（数字や記号、日本語など）で始まるファイル名を表示するには、

```
$ ls [^[:alpha:]]＊
```

とする。あるいは、文字クラスを使わずに「[^a-zA-Z]＊」としてもいい。

「＊」と「?」だけしか使えないMS-DOSのコマンドラインに比べ、Linuxの標準シェルであるbashではワイルドカードの表現能力が大幅にアップしていることがおわかりい

?	任意の1文字にマッチ	
＊	（0文字以上の）任意の文字列にマッチ	
[…]	カッコ内の文字列リストの1文字にマッチ	
[^…]	カッコ内の文字列リスト以外の1文字にマッチ	
カッコ内の文字リスト	「-」で文字の範囲を指定可能	
	[:クラス名:]で以下の文字クラスを指定可能	
	alnum	英字と数字
	alpha	英字
	blank	スペースとタブ
	cntrl	制御文字
	digit	数字
	graph	非空白文字（スペース、制御文字以外）
	lower	英小文字
	print	印刷可能文字（制御文字以外）
	punct	句読点文字（「.」「-」「_」など）
	space	空白文字（スペース、タブ、改行など）
	upper	英大文字
	xdigit	16進数字

表1 bashで使えるワイルドカードと文字クラス

ただけただろうか。最後に、bashで使えるワイルドカードと、セット構造で指定できる文字クラスの一覧を表1にまとめておく。

●ワイルドカードはシェルが展開する

ところで、MS-DOSとUNIX系OSでは、ワイルドカードの扱いに大きな違いがある。それは、MS-DOSではワイルドカードが実行するコマンドにそのまま渡されるのに対し、UNIX系OSではコマンドの実行前にシェルによって展開されるという点だ（**図1**）。

MS-DOSのコマンドは、それぞれ自力でワイルドカードの展開を行わなくてはならない。このため、MS-DOS標準のワイルドカードが使えるコマンドだけでなく、ワイルドカードに対応していないコマンドや、自力でUNIX系OS風の展開を行うコマンドなどもあり、ワイルドカードの仕様に統一性がない。

これに対し、シェルが展開を受け持つUNIX系OSでは、ワイルドカードの仕様は使用するシェルによって一意に決まり、コマンドはワイルドカードを含まない通常のファイル名を扱うだけですむ。ワイルドカードを含むファイル名がそのままコマンドに渡されるのは、ワイルドカードがファイル名にマッチしない場合だけだ。

シェルによるワイルドカードの展開は便利だが、それが災いするケースもある。「＊」や「?」をワイルドカード以外の意味（前回登場した正規表現など）で使う場合や、ワイルドカードを含むファイル名を展開しないでコマンドに渡したい場合などだ。

たとえば、与えられた文字列を表示するechoを使って「＊L＊」という顔文字（？）を出力したいとする。何も考えずにそのまま、

```
$ echo ＊L＊
```

とすると、シェルが「＊」をワイルドカードとして展開し、その結果がechoによって表示される。つまり、カレントディレクトリに「L」を含むファイル名があると、それらがすべて表示されてしまうのだ。

また、ファイル検索を行うfindで、/usr/doc以下のディレクトリを対象に「＊.txt」にマッチするファイルを検索しようとして、

```
$ find /usr/doc -name ＊.txt -print
```

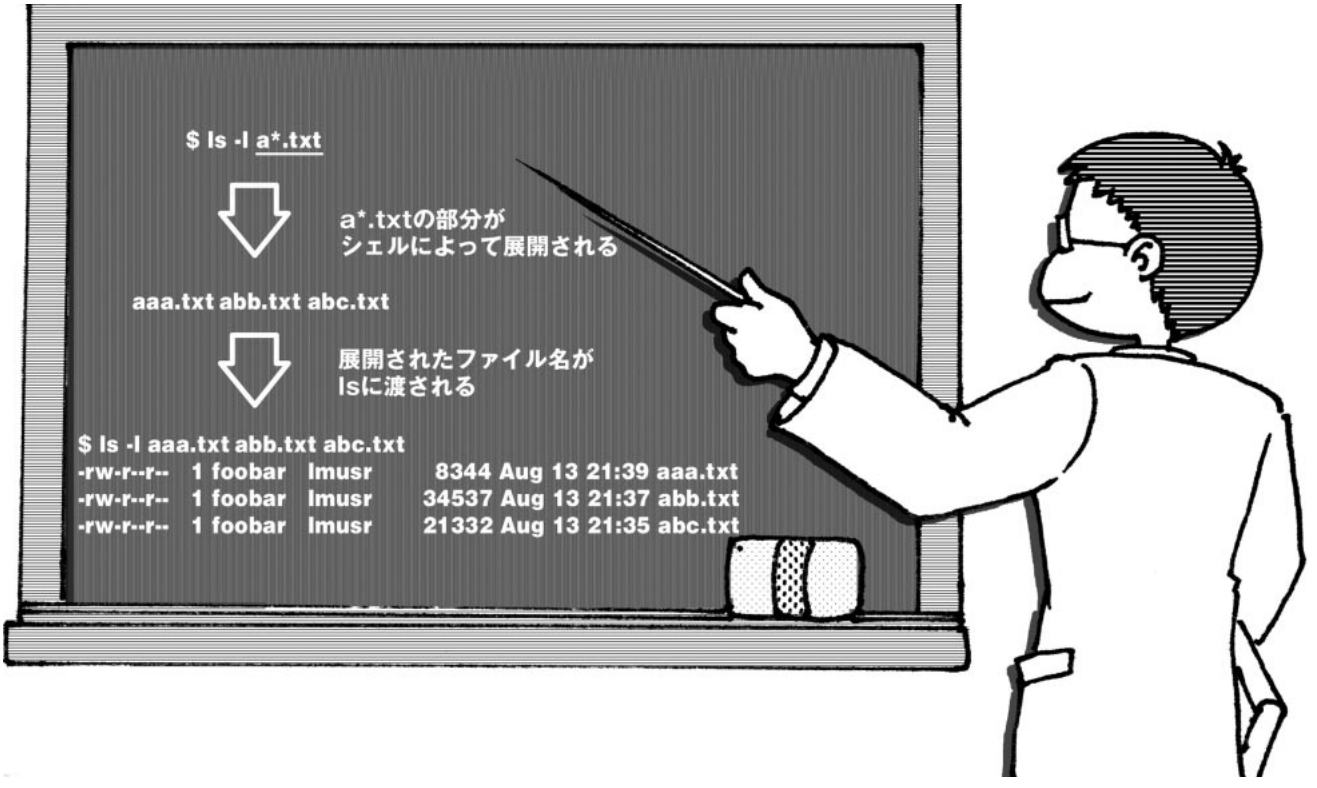


図1 Linuxのワイルドカードはシェルが展開する

とすると、シェルによって「＊.txt」の部分がカレントディレクトリの「＊.txt」にマッチするファイル名に展開された後でfindが実行される。

こうした場合には、ワイルドカードを含む文字列を「'」や「"」で囲んで（クォーティングし）、展開の対象外にしなければならない。

シェル変数を活用しよう

シェル変数は、シェルで利用するデータを保管するための名前付きの格納場所で、シェル自身が値を参照・設定する組み込み変数のほか、ユーザーが独自の変数を定義することもできる。これらはシェルスクリプトで使われることが多いが、今回のお題のようにコマンドラインで使っても便利だ。

ユーザーが新たなシェル変数を定義したり、既存のシェル変数の値を変更するには、コマンドラインで「変数名=値」とすればいい。変数名には英数字とアンダーバーが使える。たとえば、

```
$ foo=bar
```

とすると、fooという名前のシェル変数が定義され、値として「bar」が設定される（「=」の左右にはスペースを入れないこと）。なお、設定値に空白文字（スペース、タブ、改行）が含まれる場合には、値の両端を「'」か「"」で囲む（クォーティングする）必要がある。

コマンドラインやシェルスクリプトでシェル変数の値を参照するには、変数名の前に「$」をつけて記述する。たとえば、

```
$ echo $foo
```

とすると、シェルによって「$foo」が設定値「bar」に展開され（これを「変数展開」と呼ぶ）、その結果がechoによって画面に出力される。

なお、「$変数名」という表記は簡略形で、正式には「${変数名}」のように変数名を中カッコ「{」と「}」で囲む。変数名の後に英数字などが続いて、どこまでが変数名か判別できない場合には、こちらの正式形を使用する必要がある。たとえば、

```
$ echo ${foo}bar
```

とすると、変数fooの値「bar」と文字列「bar」が接続された「barbar」がechoに渡される。一方、中カッコを使わずに「$foobar」と書くと、シェル変数foobarの値が渡されてしまう（未定義なら空文字列）。

●クォーティングした文字列内での変数展開

変数展開に関しては、クォーティングした文字列内での扱いが重要だ。クォーティングとは、これまでの説明にも出てきたように、文字列を「'」や「"」で囲むこと。ワイルドカードを含む文字列をクォーティングするとシェルによるファイル名展開の対象外となり、「＊」や「?」をそのままコマンドに渡すことができる。

それでは、「$変数名」を含む文字列をクォーティングした場合はどうなるだろうか。実は、「'」と「"」という2つの記号がクォーティングに使われている理由がここにある。変数展開は、「'」では行われず、「"」では行われるからだ。たとえば、シェル変数fooの値に「bar」が設定されている状態で、

```
$ echo '$foo' = "$foo"
```

とすると、出力は「$foo = bar」となる。最初の「'$foo'」がそのまま「$foo」としてechoに渡されるのに対し、次の「"$foo"」は設定値「bar」に展開されてからechoに渡されるからだ。

もうひとつ注意すべき点は、シェル変数の値に連続した空白文字が含まれている場合、そのまま「$変数名」と書くのと、「"$変数名"」とクォーティングするのとで、得られる結果が異なることだ。たとえば、シェル変数fooの値に「bar□□baz」（スペース2つ）が設定されている状態で、

```
$ echo $foo
```

とすると、「bar□baz」と、間のスペースが1文字の出力にまとめられる。シェルが「$foo」を値「bar□□baz」に展開した後、空白文字を区切りとしてワードに分割してからechoに渡し、echoが受け取ったワード「bar」と「baz」をスペース1文字で区切って表示するからだ。

一方、シェル変数名をクォーティングして、

```
$ echo "$foo"
```

とすると、「bar□□baz」と、間のスペース2つがそのまま出力される。これは、クォーティングされた文字列は分割の際にひとつのワードとして扱われるからだ。そこでechoは、「bar□□baz」という空白文字を含んだひとつのワードを受け取ってそのまま表示する。

●シェルの組み込み変数と環境変数

シェル変数の中には、シェル自身が値を参照したり、値を自動的に変更したりする組み込み変数が含まれている。なお、組み込み変数は一般に英大文字（と数字、アンダーバー）で構成されるので、ユーザーが独自のシェル変数を定義する場合には、組み込み変数と名前が衝突しないように英小文字を使うといい。

組み込み変数の例として、bashが表示するプロンプト（この記事では左端の「$ 」で表現）の制御文字列を格納するPS1を取り上げよう。PS1の現在の値はユーザーごとに異なるだろうが、たとえば、

```
$ PS1='\w$□'
```

と設定すると、プロンプトの先頭にカレントディレクトリを含む表示（たとえば「/usr/src/linux$□」）に変化する。設定値の中の「\w」は、プロンプト定義ではカレントディレクトリを表す特殊文字だ。

ところで、X上でktermなどの端末ウィンドウを複数開いていると、片方でPS1の値を変更しても、もう一方の端末ウィンドウのプロンプトには影響しないことが確かめられる。シェル変数の定義や値はシェルごとに独立している。つまり、あるシェルで設定した変数はそのシェル内でのみ有効で、他のシェルからは参照できないのだ。

さらに、通常のシェル変数は、そのシェルから起動したプログラム（サブプロセス）からも参照できない。これでは不便な場合があるので、サブプロセスからも参照できる特殊なシェル変数「環境変数」が用意されている。HOME（ユーザーのホームディレクトリ）やPATH（実行可能ファイルの検索パス）など、重要な組み込みシェル変数の多くはこの環境変数だ。

シェル変数を環境変数にするには、シェル変数の値を設定した後で「export 変数名」とする（変数名は複数指定可能）。あるいは「export 変数名＝値」のように、値の設定とexportを同時に行ってもいい。exportを引数なしで実行すると、現在の環境変数の一覧が表示される。なお、シェルの変数展開では、通常のシェル変数と環境変数は区別されないので、「$変数名」や「${変数名}」というおなじみの形で値を参照できる。

●変数展開時に値を加工する

bashのバージョン2.0以降の特徴のひとつに、変数展開時に強力な文字列操作を行える点があげられる。単に「$変数名」でシェル変数の値を展開するだけでなく、文字列の置換やパターン照合といった操作がbash自身で行えるのだ（表2）。なお、こうした操作を行う場合、「${変数名}」のように変数名を「{」と「}」で囲む必要がある。以下では、代表的な文字列操作とその使用例をいくつか紹介しよう。

（1）パターンにマッチした部分を指定文字列で置換する

「${変数名//パターン/文字列}」というパターン照合演算子を使うと、変数の値のうちパターンと一致した部分をすべて指定文字列で置き換えたものが展開される（変数の設定値は変更されない）。パターンにはワイルドカードを使用できる。

たとえば、環境変数PATHには、実行可能ファイルが置かれたディレクトリが「:」区切りで列挙されている。現在のPATHの値は、次のようにechoで確認できる。

```
$ echo $PATH
/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:…
```

それでは、本連載ではおなじみとなったfindを使って、PATHに設定されたディレクトリを対象にファイル検索を行うことを考えてみよう。findは、「find ディレクトリ 判別式 アクション」という形で、検索開始ディレクトリ、検索条件（判別式）、検索ファイルの処理（アクション）を指定する。検索開始ディレクトリは複数指定可能だが、通常は空白文字（スペース、タブ、改行）を区切りとするので、PATHのように「:」区切りだと正しく扱えない。たとえば、PATHに設定されたディレクトリに含まれている「hoge」というファイルを検索しようとして、そのまま、

```
$ find $PATH -maxdepth 1 -name hoge
```

としてもエラーになる。なお、「-maxdepth 1」はサブディ

レクトリを検索しないという指定だ。

findを正常に動かすには、PATHの区切りに使われている「:」をすべてスペースに変換すればいい。具体的には、パターン照合演算子を使って、

```
$ find ${PATH//:/ } -maxdepth 1 -name hoge
```

とする。変数展開の結果は、PATHに含まれる「:」がすべてスペースに置換されたものになる（PATH自体の値は変更されない）。なお、置換結果の確認だけ行いたい場合は、findの前にechoを挿入する。

（2）パターンにマッチする先頭（末尾）部分を取り除く

「${変数名##パターン}」というパターン照合演算子を使うと、変数の値のうち、パターンと一致する最長の先頭部分を取り除いた文字列が展開される。一方、「${変数名%パターン}」では、変数の値のうち、パターンと一致する最短の末尾部分を取り除いた文字列が展開される。いずれも変数の設定値は変更されない。また、パターンにはワイルドカードを使用可能だ。

これら2つのパターン照合演算子は、ディレクトリ付きの長いファイル名から、ファイル名部分やディレクトリ部分だけを取り出す場合に重宝する。具体的には、ディレクトリの区切り「/」とワイルドカード「＊」を組み合わせてパターンを構成し、それぞれ「${変数名##＊/}」「${変数名%/＊}」とすればいい。ファイル名の場合は先頭からの最長マッチ、ディレクトリ名の場合は末尾からの最短マッチを使うことに注意されたい。

たとえば、シェル変数fooの値に「/usr/local/bin/hoge」が設定されている状態では、それぞれ、

```
$ echo ${foo##＊/}
hoge

$ echo ${foo%/＊}
/usr/local/bin
```

となる。

（3）指定位置から部分文字列を取り出す

bashのバージョン2.x（TurboLinux日本語版4.0、Debian2.1など）では「指定位置から部分文字列を取り出す」ことができる。「${変数名:位置:長さ}」や「${変数名:位置}」という置換演算子を使うと、変数の値全体ではなく、指定した位置から指定した長さの部分文字列だけが展開される（変数の設定値は変更されない）。位置が0以上の場合は文字列先頭から、負の場合は文字列末尾から数える（先頭は0、末尾は-1）。長さには切り出す文字数を指定し、省略すると指定位置から文字列末尾までが展開される。

これらは、変数の値を、先頭や末尾から決まった文字数だけ切り出す場合に重宝する。たとえば、シェル変数fooの値が「hogeHOGE」の場合、先頭と末尾から4文字ずつ切り出すには、それぞれ、

```
$ echo ${foo:0:4}
hoge

$ echo ${foo:(-4)}
HOGE
```

とする。

なお、「${変数名:-文字列}」という置換演算子と区別するため、位置に負数を指定する場合は上の例のようにカッコで囲む必要がある。

${変数名:-文字列}	変数が存在し空でない場合はその値、それ以外なら文字列を返す
${変数名:=文字列}	変数が存在し空でない場合はその値、それ以外なら変数に文字列を設定して返す
${変数名:?文字列}	変数が存在し空でない場合はその値、それ以外なら文字列を出力して終了する
${変数名:+文字列}	変数が存在し空でない場合は文字列、それ以外なら空を返す
${変数名:位置}	変数の値のうち、指定位置から末尾までの部分文字列を返す
${変数名:位置:長さ}	変数の値のうち、指定位置から指定した長さの部分文字列を返す
${変数名#パターン}	変数の値の先頭部分がパターンとマッチした場合、最短マッチ部分を除いて返す
${変数名##パターン}	変数の値の先頭部分がパターンとマッチした場合、最長マッチ部分を除いて返す
${変数名%パターン}	変数の値の末尾部分がパターンとマッチした場合、最短マッチ部分を除いて返す
${変数名%%パターン}	変数の値の末尾部分がパターンとマッチした場合、最長マッチ部分を除いて返す
${変数名/パターン/文字列}	変数の値のうち、パターンに最初にマッチした部分を文字列で置換する
${変数名//パターン/文字列}	変数の値のうち、パターンにマッチしたすべての部分を文字列で置換する

表2 変数展開時に使える置換・パターン照合演算子

今回のお題

☑ 複数のファイル名をまとめてリネームする

後半は毎回1つのテーマに絞り、コマンドを組み合わせて実現する方法を説明する。

今回のお題は、ファイル名を変更するmvを、シェルの繰り返し制御構文forと組み合わせて、**複数のファイル名のリネームをまとめて行う**というもの。ファイル名のパターンによって3つのケースをあげ、それぞれのコマンドラインを考える。

mvには、「ファイルを他のディレクトリへ移動する」「ファイル名を変更する」という2つの使用法がある。後者の使い方では、一度に1つのファイル名しか変更できず、3つ以上のファイルを指定するとエラーになる。

たとえば、画像ファイルの拡張子「.jpg」を「.jpeg」に変更する場合、MS-DOSのrenコマンドのように、

```
$ mv ＊.jpg ＊.jpeg
```

としてはいけない。mvが実行される前にシェルによってワイルドカードが展開され、以下のような困った事態を引き起こすからだ。

・「＊.jpg」と「＊.jpeg」の展開結果が3つ以上のファイルになる場合→mvの文法エラーになる
・「＊.jpg」にマッチするファイルがない場合→「＊.jpg」がそのままmvに渡されてエラーになる
・「＊.jpg」に1つのファイルがマッチし、「＊.jpeg」にマッチするファイルがない場合→「＊.jpg」にマッチするファイル名が「＊.jpeg」そのものに変更される
・「＊.jpg」と「＊.jpeg」がぞれぞれ1つずつファイルにマッチする場合→mvによるリネームが行われ、「＊.jpeg」にマッチしたファイルの内容が失われる

要するに、mv単独では複数ファイルの一括リネームは不可能ということだ。そこで、「＊.jpg」にマッチするファイルを1つずつ取り出し、mvを使って拡張子を「.jpeg」に変更する。そのためには、シェルの繰り返し制御構文であるforの使い方を覚える必要がある。

●forを使って複数ファイルをリネーム

forは、繰り返し処理を行うために用意されたシェルの制御構文で、コマンドラインでは「for ループ変数名 in リスト; do 繰り返す内容; done」という形で使われる。ループ変数と呼ばれる特殊なシェル変数に、リストに列挙された文字列が1つずつ設定され、「do」と「done」で囲まれた内容（複数のコマンドがあってもかまわない）が繰り返し実行される。ループ変数の更新は自動的に行われるので、ユーザーは気にしなくていい。たとえば、

```
$ for f in ＊.jpg; do echo $f; done
```

とすると、リスト（ここでは「＊.jpg」の展開結果）の内容が1つずつループ変数fに設定され、リストの内容の個数分だけ「echo $f」が繰り返し実行される。結果として、「＊.jpg」にマッチするファイルが1行に1つずつ表示されるわけだ。もっとも、これならば「ls -1 ＊.jpg」（「-1」は1行に1つずつファイル表示するオプション）とすればまったく同じ結果が得られるので、わざわざforを使う意味はない。そこで、mv単独では不可能な複数ファイルの一括リネームをforを使って行ってみよう。

まずは、拡張子「.jpg」を「.jpeg」に変更するケースをとりあげよう。mvを使ってリネームする部分以外は、上のechoを使った例と同じなので、リネーム部分を仮に「★」で表すことにすると次のように書ける

```
$ for f in ＊.jpg; do ★; done
```

★では、ループ変数fに設定されたファイル名の拡張子「.jpg」を「.jpeg」に変更する。リネーム後のファイル名の生成には、前節で説明した「${変数名％パターン}」というパターン照合演算子を利用する。末尾の「.jpg」を削除して「.jpeg」を付けるのだから、「${f%.jpg}.jpeg」とすればいい。これで★が「mv $f ${f%.jpg}.jpeg」と書けるので、先ほどのコマンドラインの中に入れると、解答は次のようになる。なお、置換結果の確認だけしたい場合は、mvの前に

echoを挿入すればいい。

```
$ for f in *.jpg; do mv $f ${f%.jpg}.jpeg; done
```

●数字を含んだファイル名を0でパディング

今度は、1.jpg、2.jpg、…、10.jpg、…のように、1から始まる数字を含むファイル名の数字部分を、最大桁（ここでは3桁）にそろえるように「0」を付加して001.jpg、002.jpg、…、010.jpg、…とリネームすることを考えよう。なお、途中の数字に抜けがあっても詰めは行わず、元の数字をそのまま利用してリネーム後のファイル名を生成するものとする。基本構造は、先ほどの拡張子の場合と同じで、forを使って繰り返し処理を行えばいい。リネーム部分を仮に「★」で表すと、

```
$ for f in [1-9].jpg [1-9][0-9].jpg; do ★; done
```

となる。ここではセット構造を使って数字のファイル名をリストに記述している。なお、「0」を付加する必要があるのは数字部分が1桁と2桁のファイル名だけなので、3桁のファイル名は含めなくてよい。

★では、ループ変数fに設定されたファイル名の数字の前に「0」を加えて3桁に変更する。そのためには、元のファイル名の前に「00」を付加し、拡張子部分も含めて末尾から7文字を切り出せばいい。

具体的には、「g=00$f」として、新たなシェル変数gに「00」とfの値を続けて設定する。続いて、前節で説明した「${変数名:オフセット}」という置換演算子を用いて、この変数の値の末尾から7文字を切り出す。これは「${g:(-7)}」と書ける。位置に負数を指定する場合はカッコで囲む必要があることに注意。

これで★が「g=00$f; mv $f ${g:(-7)}」と書けるので、先ほどのコマンドラインの中に入れると、解答は次のようになる。

```
$ for f in [1-9].jpg [1-9][0-9].jpg; do g=00$f; mv $f ${g:(-7)}; done
```

●パディングと同時に抜けている数字を詰める

前の例と同様の状況で、数字部分を「0」を付加してリネームする際、抜けている数字を詰めることを考えてみよう。つまり、1.jpgの次が3.jpgだったら、これを002.jpgにリネームするわけだ。リネーム後のファイル名に使用する数値は、新たなシェル変数iに格納することにする。このシェル変数iの初期値は1で、繰り返しのたびに1ずつ増加させる必要がある。

基本構造は今までと同じで、forを使って繰り返し処理を行えばいい。ただし、forの前でシェル変数iの初期値を設定する。リネーム部分を仮に「★」で表すと、

```
$ i=1; for f in [1-9].jpg [1-9][0-9].jpg [1-9][0-9][0-9].jpg; do ★; done
```

となる。先ほどとは違って、数字部分が3桁のファイルも数字を詰めるためにリネームするのでリストに含まれること、正しい順番で数字詰めを行うため、リストは1桁→2桁→3桁の順に指定していることに注意されたい。

★では、繰り返しのたびに1ずつ増やすシェル変数iの値に基づいてリネーム後の数字を決定し、「0」を付加して3桁に変更する。シェル変数iは数字だけで拡張子は含まれないことに注意しよう。具体的には、まず「g=00$i.jpg」として、「00」とiの値、「.jpg」という拡張子をシェル変数gに設定する。続いて、置換演算子を用いて「${g:(-7)}」としてシェル変数gの値の末尾から7文字を切り出す。ここまではいいだろう。

問題は、シェル変数iの値を1ずつ増やす部分だ。シェル変数は基本的に文字列を格納するためのものなので、単純に「i=$i+1」としてはいけない。これだと、iの値に文字列「+1」を付加した文字列がiに設定される。iの初期値は1なので、繰り返すたびに「1+1」「1+1+1」…となってしまうわけだ。

そこで、bashに用意されている数値演算構文を利用する必要がある。これは、「$((」と「))」で囲まれた部分を数値演算式として解釈するというもの。具体的には、「i=$(($i+1))」とすると、「$i+1」が「iの数値に1を加える」という数値演算だとみなされ、演算結果がiに格納される。なお、数値演算構文では展開するシェル変数名に「$」をつけなくてもよいので、この場合は「i=$((i+1))」としても同じだ。

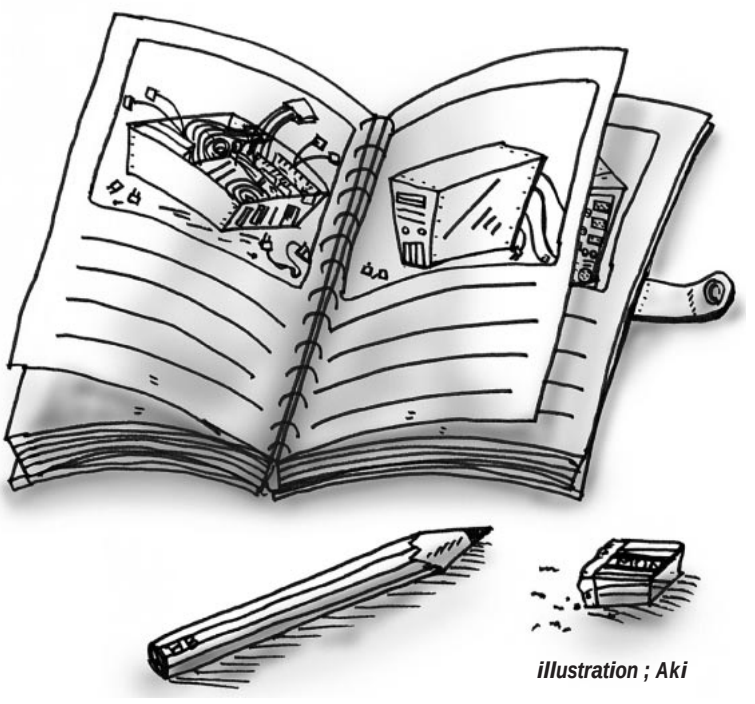
これで★が「g=00$i.jpg; mv $f ${g:(-7)}; i=$((i+1))」と書けるので、先ほどのコマンドラインの中に入れると、解答は次のようになる。

```
$ i=0; for f in [1-9].jpg [1-9][0-9].jpg [1-9][0-9][0-9].jpg; do g=00$i.jpg; mv $f ${g:(-7)}; i=$((i+1)); done
```

Linux日記

第1回　インストール前夜

Linuxを利用する上で、避けて通れないのがシステム管理です。ひとくちにシステム管理といっても、その内容は多岐にわたり、初級者にとっては荷が重い作業です。この連載では、新米管理者を対象に、システム管理についてお話していきます。ただし、行き当たりばったりに…。



文：榊 正憲
Text : Msanori Sakaki

illustration ; Aki

■ことの起こり

ある仕事の打ち合わせで書籍系の編集部に行った後、古くからの知り合いであるT氏の机に寄ってヨタ話を始めたのがすべての始まりだった。この時、なんのはずみか、T氏が「なぁお前、Linuxのシステム管理の連載の原稿書かない？」と言ってきたのである。「でもさぁ、そんなことしたら、あの本の原稿も遅れるし、こっちの厚い本の監訳も遅くなっちゃうよ」と抵抗したのだったが、いつの間にか仕事を請けることになっており、しかもそのために新しいマシンまで（自腹で）仕入れることになっていたのであった。「Linuxのディストリビューションはこれに入ってるよ」とLinux magazineの2号を渡され、とぼとぼ家に帰ったのである。

数日後、筆者は秋葉で新マシンを物色していた。最初は安いCeleronマシンでもと考えていたのだが、値段を見ているうちにだんだん欲が出てきてしまった。結局、仕入れたのはPentium III 500MHzという高速マシンである（ちなみに、いつも仕事で使っているWindowsマシンは Pentium MMXの200MHzである）。「これでやっと『電車でGo!』を仕事場でやれるぞ！」

そう、この日買ったマシンは、Linuxマシンではなく、新しいWindowsマシンなのである。だからCPUパワーや3Dアクセラレータにこだわったのだ。Linuxはどうするかって？そこらにある遅いマシンで十分でしょ。

■我が家の環境

Linuxの専門誌の中で、「Linuxは遅いマシンで十分」などと言い切ってしまうのには、れっきとした理由がある。LinuxではなくBSD/OSであるが、すでに家で使っており、その使用感からいって、速いCPUの必要性をまるで感じないからだ。ちなみにそのマシンの

Column

T氏

アスキー出版局の偉い人である。話によると、Linux magazineの創刊もこの人が仕組んだ（正確には業界でさらに偉いF氏がT氏に命令した？）らしい。筆者は昔、同じタコ部屋で仕事をして、ご飯を食べさせてもらった恩があるので、あまり強いことが言えないのである。

※ここで言及されている「タコ部屋」とは、かつて「メゾン」と呼ばれていた場所であり、「MS-DOSの黒本」や256倍シリーズの発祥の地でもある。

電車でGo!

タイトーから発売されている電車の運転シミュレーションゲーム。現在、「電車でGo!2　高速編」、「電車でGo!」、専用コントローラが販売されている。3Dアクセラレータが必須なので、家の古いPentiumマシンでは動かせなかったのである。PlayStation版は持っているのだが、画面のクオリティはやはりWindows版が上である。ただ、PlayStation版のほうがおまけが多い。Linux版なんて出ないだろうなぁ…。

CPUは、i486DX2　66MHzという骨董品である。それで困らないんだから、Pentiumの200MHzでも奢ってやればきっと死ぬほど速いに違いないと思っているのだ。どうしても遅いと思ったら、その時考えることにしよう。遠い先のCPUパワーより、目先の「電車でGo!」である。

すでに自宅でBSD/OSが稼働しており、Windowsマシンもあるとなると、当然ネットワークも引いている。筆者は自宅で原稿書きなどの仕事をしているので、実験などのために、普通のユーザーよりはたくさんコンピュータを持っている。そして、稼働可能な全マシンをスイッチングハブ（一部はダムハブ）を介して10Base-T／100Base-TXでつないでおり、このLANはさらにISDNダイヤルアップルータでプロバイダにつながっている。運用しているOSは、BSD/OS 2.0（面倒くさくてバージョンアップしてない）、Windows 95、Windows 98、 Windows NT Workstation 4.0、 Windows NT Server 4.0、MacOS 7.5、MacOS 8.1といったところだ。基本的にマルチブート環境は嫌いなので、マシン台数はこれ以上あることになる。よくも増えたものだと思うが、この中で最速なのがたったの200MHzだというのもけっこう情けない。しかし、ゲームやPhotoshopを動かさない限りこれで困らなかったのだ。仕事がらみや趣味でNetwareやSun 3がつながっていたこともあるが、今は動いていない。ここにLinuxが加わるのだから、ほとんどOSの展覧会状態である。

インターネットは、今では珍しくもなんともないダイヤルアップルータ接続だが、接続形態がたぶん世間と異なっている。普通は端末型接続とIPマスカレードを使うのだが、うちはネットワーク型接続サービスを使っている。これはプロバイダからグローバルIPアドレスの割り当てを受け、ネットワークそのものをインターネットに接続するというものだ。なので家のマシンは、子供用のPerformaも含めて、すべてグローバルIPアドレスを持っている（クラスCの4ビットサブネットに過ぎないのだけれど）。そして、プロバイダのサブドメインであるが、独自のドメイン名も持っている。

現状では、Windows NT Serverがファイル／プリント／ダイヤルインサーバ、BSD/OSがDNS／メールサーバという形で、後は適宜クライアントを使い分けている。システムに改善の余地は山ほどあるのだが、歳とともに生来の不精に磨きがかかり、「動くをもってよしとする」を実践している今日この頃である。本来、こういう性格の人間は、システム管理者になってはいけないのだが…。

■さて、Linuxで何をしよう？

この環境にLinuxを導入してどうするか？ひとつの使い方は、Xベースのワークステーションだろう。これでも筆者は、かつてUNIXのシステム管理とかソフトウェア開発をやっていたのである。その頃は、SunやSONYのNEWSを使っていた。しかしシステム管理はともかく、今はUNIXのプログラミングはやっていないので、この用途はペケである。まぁ、xtermをいくつか動かしておくというだけでもけっこう便利に使えるので、X環境を用意するというのは、用途によらず便利ではある。ちなみに、xtermを動かす程度なら、486マシンでも十分実用になる。

さて、ワークステーションでないとなると、サーバである。現在我が家では、Windows環境のサーバはNT Server、インターネット関係はBSD/OSが担当しているが、BSD/OSのほうは、パフォーマンスはともかく、そろそろ寿命なのではなかろうかと思われる。マシンをリプレイスすればいいだけの話なのだが、せっかくだからLinuxにこの辺の仕事をやらせてみようかと思っている。要するにメールとかDNSとか、インターネット用のサーバだ。また、マシンが増えてそろそろグローバルアドレスも足りなくなってきているので、プロキシサーバなんかを動かして、マシンをプライベートセグメントに移したり、安い専用線を使った常時接続にして、Webサーバなんかを動かしたりするかもしれない。

さらに、Windowsネットワークとの共存を実験してみるという手もある。別にNT Serverが動いているので、Linuxをその中に組み込むというのは必須項目ではないのだが、編集部としてはきっと書いてほしいに違いない。ここまでやれば、Macintoshもつなぎ込まないわけにはいかないだろう。幸か不幸か、我が家にはこれらを実践するために必要なリソースがすべて揃っているのだ。

まぁ、この連載でどんな話題を取り上げていくかは、神のみぞ知るといったところだろう。確かなことは、新しいPentium IIIマシンと「電車でGo!」の代金は回収しなければならないということだ。

■インストール

新しいPentium IIIマシンにアプリケーションやゲームを一式インストールし、各種データファイルを引っ越し、今までWindows 98だったマシンにLinuxをインストールする準備ができた。このマシンはのスペックはPentium MMX 200MHz、メモリ192Mバイト、

3Comのネットワークカード、2台のハードディスクとCD-ROMがSCSIでつながっている。調べてみたら、C:ドライブが4GバイトでWindows 98、D:ドライブが2Gバイトでデータのみだったので、今回はD:をつぶしてLinuxを入れることにした。一応、Windows 98とLinuxのデュアルブート環境にするわけだ。まぁ、Windows 98を起動する理由はほとんどないはずだが。

さて、インストールである。使ったのは、編集部からもらった雑誌の付録のRed Hat 6.0のバイナリディストリビューションである。まぁどうにかなるだろうと思って始めたら、あっさりどうにかなってしまった。まぁ、SunとかBSDの使用経験で、パーティションの切り方とか、サポートされるサービスなどを一通り知っていたというのは大きいだろう。

とりあえず、後からインストールする手間を省くため、明らかに不要なものも含めてすべてのパッケージをインストールした（/usrパーティションの使用量が1Gバイトあまりになったのはびっくりである）。また、動かしうるすべてのサービスを（矛盾するものも含めて）稼働させるというかなり無茶苦茶な設定を行った。これについては、後からいろいろといじっていこうと考えている。もちろんソースも入れるつもりだ。

■お断り

これから書いていく内容は、初心者が最初から順に読んでLinuxの管理を覚えていくという構成になっていない（そもそも、筆者だってLinuxについては素人なのだ）。筆者が気ままにいろいろなことを書いていくという構成になっている。では、囲み記事が内容を補足しているかというとそんなこともない。ほとんどはどうでもいいヨタ話である。

■ファイルシステムとパーティション

インストール時にまず頭を使うのが、パーティションをどのように分割するかということだ。

UNIXのファイルシステムは、当然ツリー構造なのであるが、ツリーをどのように構成するかという哲学がDOS／Windowsと異なっている。DOSは完全にドライブ指向である。ファイルシステムは階層化されているが、個々のドライブ（正確にはパーティシ

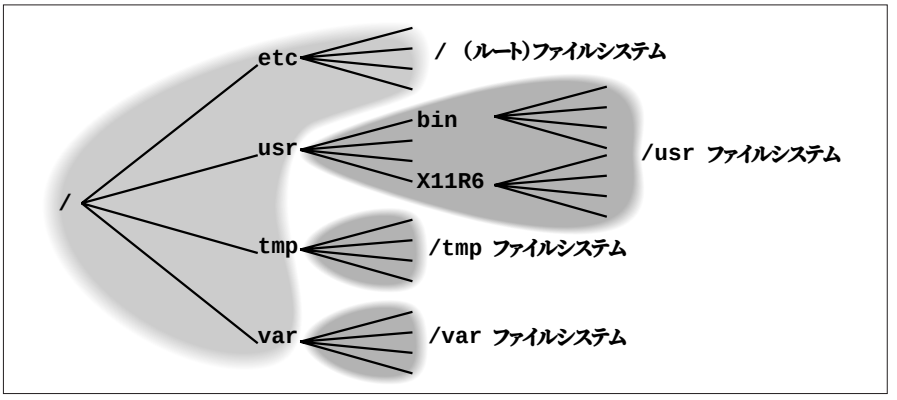


図 複数のサブツリーを収めた階層化ファイルシステム

Column

BSD/OS

BSD系のUNIX互換オペレーティングシステム。フリーUNIXではなく、Berkeley Software Design, Inc.（BSDI）という会社の製品である。現在では、製品名に「Internet Super Server」とかいう名称が入っているらしい。FreeBSDに比べ、足腰（I/Oまわり）がしっかりしており、大負荷に強いと知り合いの管理者がいっていた。現在のバージョンは4.Xだと思う。我が家の古い2.0はすでにuptimeが400日を超えている。停電とハードウェア障害さえなければ落ちない良いシステムだ。

i486

Intel社のプロセッサで、Pentiumの前の製品である。その前のi386をさらに高速化し、浮動小数点演算コプロセッサも内蔵した。買った時はとても速いプロセッサであったし、今でもうすら重たいソフト（Windows 95以降のOSはうすら重たいソフトである）を動かさなければ十分実用的な速度で動作する。

端末型接続とネットワーク型接続

プロバイダとの接続形態。端末型は、プロバイダに接続した時点でクライアント（ユーザーのコンピュータ）に、プロバイダがプールしているIPアドレスがひとつ割り当てられる。これは動的に割り当てられるので、接続ごとにアドレスが変わる。接続はユーザーからだけであり、IPアドレスも変わるので、当然ユーザーサイドにサーバを置くことはできない。IPマスカレードという機能を使えば、1つのIPアドレスで同時に複数の端末の接続をまかなえる。これが今、世間で流行っているISDNルータの使い方である。

ネットワーク型接続は、ユーザー側にグローバルIPアドレスとドメイン名を割り当てる形のサービスである。つまりIPアドレスは変化しない。また、プロバイダ側からの接続を許可すれば、ユーザーサイドにサーバを設置することも可能であるが、これをやると接続費用がいくらになるかわからないので現実的ではない。

そこまでやりたければ専用線接続にすべきだろう。

ョン）ごとにツリーが存在するという形だ。Windows 95以降では、デスクトップの下にドライブが配置され、複数のドライブ、あるいはネットワークドライブであっても、ひとつのツリーとして扱えるようになったものの、やっぱりC:とかD:といったドライブの識別が明確に存在している。

それに対してUNIXは、初めに1つのツリーありきという考え方だ。各パーティション上の階層化ファイルシステムは、全体のツリー構造の一部（サブツリー）となる。つまり、複数のサブツリーを組み合わせて、全体で1つのツリー構造を構成している。この中で特定のディレクトリやファイルを指定するのに、ユーザーはパーティション配置を意識する必要はない。これはすべてオペレーティングシステムによって管理され、あるディレクトリの下が別パーティションになっているとか、ほかのファイルサーバにぶらさがっているといったことを意識する必要はまったくないのだ。ただ、フロッピーディスクやCD-ROMなど、リムーバブルメディアを使う場合、ファイルシステムにアクセスするためにいちいちマウントし、取り出す前にアンマウントしなければならないという難点がある。

■パーティション分割をしないという選択

1台のドライブがギガバイトオーダーの容量を持つ現在では、わざわざ多数のパーティションに分割して、それを1つのツリーに組み合わせるという必然性はなくなった（スワップパーティションのような特殊用途は別である）。実際、WindowsやMacintoshでは、1台のドライブの1つのパーティションにシステムソフトウェアからアプリケーション、ユーザーデータまで、すべて収めているのが普通だ。

Linuxも含めて、UNIXもこのような構成にできる。大きなパーティションを1つ作り、そこにすべてのファイルとディレクトリを収めるのである。こうすることで、どのパーティションを何メガバイトにしようかと悩む必要はなくなるし、あるパーティションは余っているのに、別のパーティションの容量が足りなくなるといった事態もなくなる。パーティションサイズの制限内で、各ディレクトリの容量配分は自由なのだ。/homeの容量を稼ぐために、/usrの下の不要なソフトウェアを消去するといったことが可能になる。

複数のドライブを接続したサーバでもない限り、何もわざわざ1台のドライブを複数のパーティションに分割する必要などない、という考え方もある。特にパーティション割りなどやったことのない初心者には魅力的な選択であるし、筆者も初心者にはそう勧めるだろう。

■パーティション分割と耐障害性

では、パーティション分割が無意味かというと、そんなことはない。頻繁に読み書きされる部分と、ほとんど読み込みしか行わない部分を分けるとか、全体の一部だけをバックアップするといったことが可能になるからだ。

/や/usrといった部分は、オペレーティングシステムやプログラム、ドキュメントなどを収め、もっぱら読み込み専用だが、/home以下は頻繁に読み書きされる。/tmpや/varなどは、読み書き削除の頻度が非常に高くなる。一般に、書き込み頻度が高いファイルシステムほど、ソフトウェアのバグやハードウェア障害などで損傷を受ける可能性が高い。読み込み時にシステムが落ちても実害は少ないが、書き込み途中のクラッシュは、ファイルシステムに障害を与える可能性が高いのだ。データの書き込み中なら、ゴミが残るだけで済むこともあるが、管理情報の書き込みが失敗すると、いとも簡単に障害となる。また、障害が発生したファイルシステムは、修復しない限り使用できない。マウントするのが不可能なわけではないが、障害が増殖してしまう可能性が高いからだ。

ツリーを複数のファイルシステムに分けることで、このような障害が発生しても、システムをブートできる可能

Column

DOSの歴史的経緯

MS-DOSでファイルシステムが階層化されたのはバージョン2.Xの時である。それ以前はフラットなファイルシステムだった。また、この頃はフロッピーベースで使うのが普通だった（ハードディスクドライブが憧れのデバイスだった時代である）。このような環境では、ドライブ指向の構成のほうが明らかに使いやすかったのである。その後ハードディスクが普及し、フロッピーディスクなどめったに使わなくなったのだが、やはり互換性は重要である。結局、良くも悪くも未だにそれを引きずっているのだ。

Windows 95以降は、デスクトップの下にマイコンピュータ、ネットワークコンピュータがあり、その中にドライブがあるという名前空間が導入されたが、おかげで2種類の名前空間を使い分けるという状況が発生した。C:の下にデスクトップがあり、その下にC:があるというのも、考えてみればすごい話である。

結局、マイコンピュータ中の各ドライブのショートカットをデスクトップ上に作って、昔ながらのドライブ指向の使い方をしているユーザーも多いことだろう（筆者もそのひとりだ）。

性が高くなる。障害が発生する確率が高いのは、書き込みが頻繁に行われる/varや/tmp、/homeである。一方、システムを稼働させるのに必要な/や/usrは、書き込み頻度が低いので、致命的な障害を受ける可能性は低く、そして軽い障害なら、ブート時の自動チェックで回復可能なことが多い。つまり、一部のファイルシステムは使えないものの、システムは起動可能なのである。

これをさらに突き進めた選択が、/と/usrのリードオンリーマウントだ。ほとんど読み込みしか行わないファイルシステムでも、アクセス日時の更新は行われる。これにはファイルシステムへの書き込みが伴う。もちろん、わずかなデータなので、その書き込み中に障害が発生する可能性は低い。それに対して、リードオンリーマウントにしてしまえば、アクセス日時の更新も行わなくなる。つまり、ファイルシステムに対する一切の書き込みが行われなくなるのだ。耐障害性は一段と高まる。もちろん、この状態では新しいソフトウェアのインストールや設定変更なども行えないので、そのような作業を行う際は、一時的にリードオンリー属性を解除しなければならない。

■パーティション分割とバックアップ

一部に障害があっても、システムが動いてしまえば、ファイルシステムの修復やバックアップのリストアなどは比較的容易だ（もちろん、起動できない場合に比べての話だが）。システムがすべての1つのパーティションにインストールされている場合は、たとえば/varや/tmpで障害があった場合でも（経験的には、エラーが報告されるのは、たいていこの2つだ）、システムを安全に起動できなくなってしまい、フロッピーディスクやCD-ROMを使わない限り、修復が難しくなる。もっとも、フロッピーディスクやCD-ROMでブートできる現在の環境では、エラー時にはブート方法を変えるというのも、そんなに手間のかかるものではなくなったが。

バックアップにdumpコマンドを使う場合は、パーティション分割が重要になる。tarなどはディレクトリ単位でアーカイブするが、dumpコマンドはファイルシステム、つまりパーティション単位でアーカイブするからだ。/や/usrに比べ、/homeはバックアップの頻度が高くなる。逆に、/tmpはバックアップする必要はない。/varは微妙で、運用形態によってバックアップしたりしなかったりするだろう。いずれにせよ、パーティションを更新頻度やバックアップの必然性に応じて分割することで、効率的なバックアップが可能になる。/や/usrは、何らかの更新を行った時のみダンプすればよい。/homeは、定期的なダンプを行うことになるだろう。パーティション分割を行わなかった場合は、すべてまとめてダンプすることになる。実際問題としては、差分ダンプを行うだろうから、変更がなかった部分はダンプされないが、それでも定期的に行うフルダンプでは、すべてをまとめてバックアップメディアに書き込むことになる。パーティション分割を行えば、バックアップ処理が大幅に効率化される。また、クラッシュ後のリストア処理も容易になる。

バックアップメディアの容量も重要なファクターである。DATなどの大容量メディアであれば、たいていは1本のテープにダンプできるが、MOやCD-Rなどを使う場合は、容量がディスクドライブより小さくなる。このようなメディアにダンプする場合は、途中でメディアを交換する必要がある。やってみるとわかるが、これはえらく面倒な作業だ。パーティションサイズをバックアップメディア以下の容量にすれば、交換しないで済む。これにより、無人ダンプをcronで行うことができるようになる。

結局、筆者は/、/usr、/tmp、/var、/homeと5つのパーティションに分割した。上記のような理由もあるが、半分以上は体に染み付いた習慣という奴である。

Column

ディスクドライブの信頼性とソフトウェアの信頼性

今のディスクはなかなか壊れないから、ディスクのハードウェア障害に備えてパーティション分割をする意義は少なくなったというのは事実である。

実際、昔のドライブはよく壊れた。しかし、ファイルシステムが損傷する最大の理由は、ハードウェア障害ではなくソフトウェア障害である。プログラムのバグによりシステムがクラッシュし、ファイルシステムがダーティな状態のまま止まってしまうと、いとも簡単におかしくなる。本来ならば、ユーザープロセスの障害でシステムがクラッシュすることはないはずなのだが（あるいはそうであるべきなのだが）、実際に起こるものはしかたがない。

実際問題として捉えた場合、筆者の経験では、ファイルシステムの障害は、Windows系の場合、ソフトウェアの障害、停電、ハードウェア故障の順で起こっている。ぼろいマシンに載せたBSD/OSでは、停電、ハードウェア故障、ソフトウェア障害の順である。Linuxでどうなるかはまだ未知数だが、何が原因にせよ、備えておくことに越したことはない（といいつつ、筆者自身はきっと何もしないのだ）。

■複数のドライブのパーティション分割

ドライブを複数接続する場合は、好みに関らず複数のパーティションを組み合わせることになる。ファイルサーバやデータベースサーバなどで、後からデータ用の大容量ドライブを増設するといった場合は、追加したドライブ上にパーティションを1つ用意し、ツリー上の適当な位置にマウントするといったことになるだろう。

最初から複数のドライブを装備したシステムの場合は、また別の分割のしかたがある。複数のドライブに、/、/usr、/var、/tmpなどを分散させるというやり方だ。SCSIインターフェイスのように、複数のディスクI/Oリクエストを並行して処理できる場合は、頻繁にアクセスされるファイルシステムを別のドライブに配置することによって、同時に複数のI/Oリクエストを複数のドライブ上で行えるようになり、ファイルI/Oのパフォーマンスが向上するのである。頻繁にアクセスされるファイルシステムが1台のドライブに集中して置かれている場合は、いくら並行してコマンドを発行したところで、ドライブ本体は逐次処理しか行えないので、各プロセスがディスクI/O待ちに費やす時間が長くなる。

また、サーバマシンなどで、多数のディスクドライブを装備している場合は、ディスクコントローラ（つまりSCSIアダプタ）も複数にするとパフォーマンスが向上する。SCSIで複数のドライブを接続した場合、複数のドライブで同時にI/O処理を行うことができるが、ドライブの数が増えると、今度はSCSIのバスバンド幅がボトルネックになる。SCSIアダプタを複数用意し、多数のディスクを分散して接続することで、各ドライブあたりのバンド幅が増やせるのである。もっとも、次にPCIバスのバンド幅というボトルネックが待っているのだが。

まぁ、そこまで行き着く前に、マシンの増設を考えよう。サーバの場合は、ディスクインターフェイスがボトルネックになる前に、ネットワークのバンド幅がボトルネックになるからだ。ディスクインターフェイスにお金をかけるより、ネットワークの100Mbps化、スイッチの導入などの方がパフォーマンスの向上に寄与する可能性が高い。

■さて、次回は

Linux日記などという題が付いているにも関らず、今回はLinux固有の話を何ひとつしなかった。

実はすでに次回分の原稿にとりかかっている。パーティション絡みの話の続きとして、スワップパーティションの話を書き始めたのだが、スワップパーティションの予備知識として仮想記憶システムの話から始めたのが間違いだった。Linux絡みの話は、もしかすると最後にちょろっと出てくるだけかもしれない。とりあえず、仮想記憶のためのデーモンとか、vmstat、psなどの管理ツールの話くらいまでは持っていこうと思っているのだが、どうなることやら。

Column

CD-ROMやフロッピーディスクでブートできないシステムの場合

昔話になるが、フロッピーディスクやCD-ROMからブートできないマシンの場合、ディスク障害でブート不能なときにどうするかという話をしておこう。これを聞けば、今の環境がどれだけ便利か、そしてなぜ年寄りが複数パーティションを勧めるのかがわかってもらえるのではなかろうか。

筆者が昔使っていたSun 3というUNIXワークステーションは、ハードディスクのほかにはテープドライブしか付いていなかった。これでルートファイルシステムに障害が発生し、ブートできなくなった時はどうするか？そもそもどうやって最初にインストールするのか？

このマシンのROM BIOSは、ディスクのパーティション分割やフォーマット、ブートを行うことができた。興味深いのは、テープドライブからのブートもサポートしていたということである。まず、ディスク上にパーティションを設定する。そして、テープドライブからコピープログラムをブートするのである。このプログラムは、あるデバイスから読み込んだバイナリデータを別のデバイスに書き込むことができる（ddのようなものだ）。CD-ROM上のイメージファイルをフロッピーディスクにコピーして、ブートフロッピーを作ることを思い浮かべれば、これがどのようなプログラムかわかるだろう。このプログラムを使って、テープ上に用意されたルートファイルシステムのイメージをスワップパーティションにコピーするのである。これで、障害のないルートファイルシステムが準備できた。次に、マシンをリブートして、スワップパーティション上のファイルシステムを指定してブートする。このルートファイルシステムには、インストールプログラムのほか、診断／修復プログラム、リストアプログラムなどが含まれている。インストールプログラムを使えば、システムの最初のインストールをテープから行える。診断／修復プログラムとリストアプログラムを使えば、ファイルシステムを修復したり、ダンプからリストアすることができるのである。

問題は、これに要する時間だ。スワップパーティション上にルートファイルシステムを作り、それをブートするまでに優に1時間程度はかかる。修復やリストアはそれから始めるのだ。ちょっと手間取れば、あっという間に夜が明けてしまう。初めてフロッピーブートするPC UNIXを見た時、筆者は感動したものである。

Webサーバ構築術（第2回）

Linuxで使えるデータベースやイントラネットソフトが多くなってきた。クライアントにWebブラウザを使うようになってきているため、WebサーバとしてのApacheの必要性が高まっている。多機能で複雑な設定ファイルを理解してApacheを使いこなしてみよう！

自分だけのApacheコンフィグレーション

文：中島昌彦
Text : Masahiko Nakajima

ほかとはちがうApacheへ

前回はApacheの基本的な設定を解説したが、本当のApacheのおもしろさはこれからあとである。初回はApacheをとにかく動かすまで、今回は、自分ならではのApacheを動かしたい。

Apacheは多くのWebサイトで利用されているが、ISPでユーザーに解放しているWebサーバとして利用する場合、コンフィグレーションを書き換えて、Apacheでありながら、Apacheとは見えないような設定になっているところもある。

今回は、そんな設定部分を見ていく。なお、紹介例として、Red Hat Linux 6.0（英語版）でいっしょにインストールしたApache 1.3.6をベースとしている。ディレクトリ構造、コンフィグレーションファイルの場所など、すべてRed Hat Linux 6.0のApache設定ベースとなっている。

話がややこしくなりそうなので、あらかじめ書いておこう。ドキュメントディレクトリとディレクトリの関係だ。

Apacheでは、HTMLファイルなどのドキュメントファイルを置くディレクトリを、ドキュメントディレクトリという。Red Hat Linux 6.0ベースであれば、/home/httpd/である。

URLでサイトのトップを指定したとき、読み込まれるディレクトリがドキュメントディレクトリで、ルートディレクトリとは場所がちがう。ここを注意していてほしい。

なお、環境設定ファイルがある場所は、/etc/httpd/confで、ここはコンフィグレーションディレクトリである。

ファンシーインデックスのカスタマイズ

「http://www.ascii.co.jp/linux/」のように、ディレクトリ止めでURLを指定したとき、そこにindex.htmlなどのインデックスファイルがなかった場合、そのディレクトリ中のファイル一覧を表示させる機能がある。この機能は、access.confの40行目にある、

```
Options Indexes
```

である。これは前回もちょっと触れたとおりだ。この表示がファンシーインデックスで、Webらしくファイルのアイコン付きで表示される。これはこれで悪くないが、リスト一覧を出すたびに、むだな画像ファイルまで転送される。しかも、正しく画像ファイルを出力しようとすると、画像ファイルのディレクトリもちゃんと管理しなければいけない。実に手間な作業となる。

そこで、インデックス表示はしたいけれども、画像はいらないと割り切れるならば、/etc/httpd/conf/srm.confの31行目を、

```
# FancyIndexing off
```

として、ファンシー表示を設定しないようにする。これで、インデックス表

示時の画像ファイルの管理や、画像ファイル転送によるトラフィック問題に頭を悩まさなくてすむ（**画面1**）。

■アイコンファイルを書き換える

逆に、ファンシーインデックスで画像部分をもっとかっこよく、自分で定義した別のアイコンを表示したいこともある。そんなときは、srm.confの31行目以降にある部分を変更していく。

36行目以降、68行目までの記述が、インデックス表示時に組み合わせるアイコン指定だ。いくつか例を出すと、

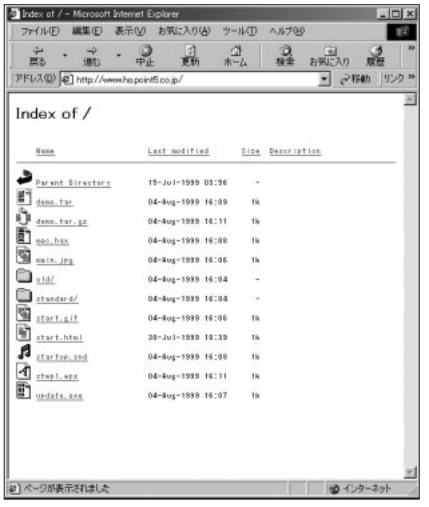
```
AddIconByEncoding (CMP,/icons
/compressed.gif) x-compress x-gzip
```

は、MIMEタイプで定義した形式に合わせてアイコンを表示させるオプションだ。この場合、x-compressもしくはx-gzipのMIMEエンコード形式のファイルには、/icons/compressed.gifの画像ファイルを表示する。画像ファイルの場所は、ドキュメントディレクトリから追っていく。Red Hat 6.0であれば、/home/httpd/iconsというディレクトリにある画像ファイルが該当する。

AddIconByEncoding以外にも、AddIconByType、AddIconというオプションがある。AddIconByTypeはMIMEタイプの拡張子に対して、AddIconはファイル名単位で画像を定義づけられる。これらの書式は、いずれも、

オプション名　画像ファイル名　指定形式

という形式になる。そして、いずれの設定にも合致しないファイルは、DefaultIconで指定した画像ファイルが適用される。デフォルトでは、/icons/unknown.gifが指定されている。



画面1　インデックス表示の2つの方法
右側がデフォルト設定になっているファンシーインデックス。左側がファンシーインデックスをオフにしたときのインデックス表示。ファンシーインデックスを利用することで、Webサーバのインデックス表示らしくなる。

コマンド名	動作概略	書式
AddIconByEncoding	MIMEエンコード形式に合わせた画像ファイル定義	オプション名 （ALT文字列, 画像ファイル名） 指定
AddIconByType	MIMEタイプに合わせた画像ファイル定義	
AddIcon	ファイル名単位での画像ファイル定義（正規表現で指定できる）	
DefaultIcon	いずれにも合致しない場合、デフォルトで表示する画像アイコン指定	DefaultIcon 画像ファイル
AddDescription	Description欄への記述	AddDescription "詳細内容" ファイル名
HeaderName	ヘッダ部分に表示する文字もしくはHTML	HeaderName ファイル名
ReadmeName	最後に付加する文字もしくはHTML	ReadmeName ファイル名
IndexIgnore	インデックス表示対象外のディレクトリ、ファイルを指定	IndexIgnore 名前

表1　ファンシーインデックス周りの設定コマンド

■アイコン画像にだってALTが必要

画像には、ALTを付けて、画像の中身を表示すべきというポリシーを持つ人が多い。FancyIndexingであろうとも、このポリシーは持つべきだ。

そこで、AddAlt、AddAltByEncoding、AddAltByTypeというオプションが用意されている。それぞれ、AddIcon、AddIconByEncoding、AddIconByTypeに合わせてALTを送り出すというものだ。しかし、AddIconで定義して、さらにAddAltで定義するという方法は実に面倒だ。しかも、定義忘れが出てくる。そこで、AddIconするときに、いっしょにALTも定義するという方式が使われる。

デフォルトのsrm.confでは、36行～41行がまさにそれで、

オプション名　（ALT文字列, 画像ファイル名） 指定形式

という書式になる。ただし、この書式をDefaultIconで指定しようとしてもできない。

■詳細表示欄にファイル内容を記載

ファンシーインデックスをよく見ると、右端にDescriptionという欄がある。ここに何も記載されていないと落ちつかないなら、AddDescriptionで指定していく。書式は、

```
AddDescription "詳細内容" ファイル名
```

となり、ファイル名部分には、拡張子のみ、ドキュメントディレクトリからのフルパス、ワイルドカード指定ができる。

```
AddDescription "HTMLファイル" .html
AddDescription "CGIファイル" .cgi
```

という指定をすれば、拡張子htmlとCGIに対しては、それぞれ「HTMLファイル」「CGIファイル」が表示される。

■先頭とお尻に定型文を表示したい

いよいよ設定も大詰めだ。個人だとそうでもないが、会社でWebサイトを運営するとなると、いろいろな制限が加わる。HTMLのヘッダ部分は統一とか、HTMLの最後にはCopyright表記を入れるといったように、いろいろな約束がある。インデックス表示ぐらいは例外であってもいいと思うが、これができてしまう以上、できないといって逃げるわけにはいかない。

デフォルトでは、srm.confの83行目と84行目に、

```
ReadmeName README
HeaderName HEADER
```

という定義がしてある。これは、指定したURLがインデックス表示設定になっていたとき、インデックスを表示するときにヘッダ部分をHEADER.htmlファイルもしくはHEADERファイルに置きかえ、最後にREADME.htmlファイルもしくはREADMEファイルを付け加えて表示するという設定だ。いずれも、指定したカレントディレクトリ上のREADME.htmlもしくはHEADERファイルを読み出す。このファイルは、インデックスを表示する全てのディレクトリにそれぞれ置いておく必要がある。

▼リスト1　srm.confのファンシーインデックス周りの設定例

```
AddIconByEncoding （圧縮ファイル,/icons/compressed.gif） x-compress x-gzip
AddIconByType （HTMLファイル,/icons/text.gif） text/*
AddIconByType （画像ファイル,/icons/image2.gif） image/*
AddIconByType （音声ファイル,/icons/sound2.gif） audio/*
AddIconByType （動画ファイル,/icons/movie.gif） video/*
AddIcon （バイナリファイル,/icons/binary.gif） .bin .exe
AddIcon （MacのBinHex,/icons/binhex.gif） .hqx
AddIcon （tarファイル,/icons/tar.gif） .tar
AddIcon （wrlファイル,/icons/world2.gif） .wrl .wrl.gz .vrml .vrm .iv
AddIcon （圧縮ファイル,/icons/compressed.gif） .Z .z .tgz .gz .zip
AddIcon （そのほかの画像,/icons/a.gif） .ps .ai .eps
AddIcon （HTMLファイルほか,/icons/layout.gif） .html .shtml .htm .pdf
AddIcon （テキストファイル,/icons/text.gif） .txt
AddIcon （Cのソース,/icons/c.gif） .c
AddIcon （スクリプトファイル,/icons/p.gif） .pl .py
AddIcon （forファイル,/icons/f.gif） .for
AddIcon （dviファイル,/icons/dvi.gif） .dvi
AddIcon （uuファイル,/icons/uuencoded.gif） .uu
AddIcon （シェルスクリプトファイル,/icons/script.gif） .conf .sh .shar .csh .ksh .tcl
AddIcon （texファイル,/icons/tex.gif） .tex
AddIcon （coreファイル,/icons/bomb.gif） core
AddIcon （1つ上へ,/icons/back.gif） ..
AddIcon （最初に読んで,/icons/hand.right.gif） README
AddIcon （ディレクトリ,/icons/folder.gif） ^^DIRECTORY^^
AddIcon （ブランク,/icons/blank.gif） ^^BLANKICON^^
DefaultIcon /icons/unknown.gif
AddDescription "HTMLファイル" .html
AddDescription "CGIファイル" .cgi
AddDescription "画像ファイル" .gif .jpg .eps
AddDescription "MacのBinHex" .hqx
AddDescription "ディレクトリ" ..
AddDescription "Windowsの実行ファイル" .exe
AddDescription "圧縮ファイル" .gz .zip .sit .sea
AddDescription "アーカイブファイル" .tar
AddDescription "音声ファイル" .snd
```

このように定義していくことで、ファンシーインデックスがよりいっそうファンシーになる。

ところで、この2つのファイルをディレクトリに入れてからインデックスを表示させると、あることに気がつくはずだ。入れたはずのREADME.html、HEADER.htmlが表示されないのだ。確かにファイルは存在するが、インデックス表示からは除外されている。こうした表示しなくてもいいファイルは、インデックス表示から排除する設定ができるようになっているのだ。

■インデックス表示しないファイルを設定

インデックス表示が危険な理由は、まさにここだ。ディレクトリによっては、隠しもっている設定ファイル、パスワードファイルの存在がインデックス表示でばれる危険性を持つ。

そこまでしてインデックス表示をすることはないと思うが、インデックス表示にこだわるならば、表示しないファイルの設定も必要となる。

これは、srm.confの89行目にあるIndexIgnoreで指定する。デフォルトでは、

```
IndexIgnore .??* *~ *# HEADER*
```

▼リスト2　HEADER.html、README.htmlの例

```
/のインデックス表示です<BR>

ここまで<BR>
<BR>
リンクは自由です。許諾願いはいただかなくて結構です
```

いずれも、カレントディレクトリにあるHEADER.html、README.htmlのみが有効。

画面2　HEADER、README、AddDescription処理を加えたファンシーインデックス表示
これぐらいまで手を加えると、独自Webサイトらしいイメージになる。

```
README* RCS
```

となっている。正規表現も含まれているので順次解説すると、ファイル名もしくはディレクトリ名が、

- 「.」で始まっている3文字以上のもの
- 「~」が含まれているもの
- 「#」が含まれているもの
- 「HEADER」で始まるもの
- 「README」で始まるもの
- 「RCS」

の6つがインデックス表示から除外される。これに合わせて見せたくないファイルを設定すれば、INDEX表示で見せてはいけないファイルを表示できなくできる。ちなみに、最初の指定である「.」で始まって3文字以上という指定は、カレントディレクトリ（.）親ディレクトリ（..）を除外するときによく使う指定方法。これを指定することで、.htaccessや.htpasswdの存在を公表しないですむ。

さらに、もうひとつ、HEADER.htmlもしくはHEADERファイル、README.html、READMEファイルのインデックス表示を抑制するために、HEADER*とREADME*というパターンが表示抑制されている。このため、HEADER1やREADME.txtというファイルやディレクトリも、すべてインデックスでは表示されない。これでは困るのなら、

```
IndexIgnore .??* *~ *# HEADER,html
HEADER README.html README RCS
```

という指定でIndexIgnoreを変更する。

独自のエラー画面でエラー内容をrootにメール送信

Apacheで見た目を大きく変える設定がもう1つある。エラー時の表示だ。エラー時の画面というと、多くのWebサイトが何も設定していないものだ。URLが間違っていたときは、ブラウザから質素なメッセージが返ってくる。このためか、IEでは独自にエラーページを作って返してくれる。

エラーページを独自のものにした場合、IEのこの設定は大きなお世話だ。IE 5であれば、[ツール]-[インターネットオプション]-[詳細設定]と選び、ブラウズの欄にあるHTTPエラーメッセージを簡易表示するというチェックを外しておく。これで、独自のエラー画面のデバッグができる。

■エラーコードに合わせたHTMLを作る

エラーページといっても、難しいことをするわけではない。エラーに合わせて、そのときに表示するページを作り、それを設定する。手間といえば、エラー表示のページを作ることぐらいで、ほかにこれといった面倒な作業は出てこない。それでもエラーページを独自のものにしない理由は、本来見せるべきコンテンツにパワーを割いていて、エラーページまでは手が回らないといったあたりだと信じている。

ところで、一部のISPは独自のエラーページを作っている。こうしたISPのWebサイトへアクセスすると、ユーザーの立場に立っているなと実感できるものである。一般のWebサイトでも、サーバのエラーコードを返すだけではなく、なにかしらの独自エラーページが出れば、わざわざ見に来てくれた人たちに対して、敬意を表せそうだ。

まず、コンフィグレーションファイルの設定だが、エラー時のリダイレクト処理設定は、srm.confの215行目から230行目となる。すべてがコメントアウトしてあるため、なかなか気がつかないが、ErrorDocumentというコマンドを使い、エラー番号が出たときには、それを特定のHTMLにリダイレクトするという設定がここである。リダイレクト先は、ファイルでもCGIでもSSIでもかまわない。

もっともよくある、ファイルが見つからない場合を例にとろう。エラー番号404のこのエラーは、srm.confの223行目のコメントを外し、さらに修正をかけて次のようにする。

```
ErrorDocument 404 /missing.cgi
```

書式は、ErrorDocumentというコマンドに続き、エラー番号、そして、リダイレクト先のURLである。この例では、404のエラーが出たときに、/missing.cgiにリダイレクトをする設定になる。

ここまで指定すれば、あとはmissing.cgiを作るだけ。これで、指定ファイルが見つからないときの対処ができる。

問題はCGIの中身だ。ユーザーを正しいURLに導くようなものもいいだろうし、エラー内容を記録しておき、エラーが出ないようにコンテンツ側を対処するという前向きな姿勢で使う方法もいい。

リスト3のサンプルでは、エラー画面を出すと共に、リクエストしたURLと飛んできた元のURLを控え、リンク元の修正に役立てようというコンセプトのものだ。わざわざリダイレクトしたファイルで記載せずに、アクセスログから確認するという方法もある。しかし、膨大な量のアクセスログからgrepするぐらいなら、特定ファイルに書き込んでしまうほうが楽だ。もしアクセス数が少ないWebサイトならば、エラーメッセージをすべてメールで送り付けるようにすれば、早く対策が取れる。そんなコンセプトである。

サンプルでは、リクエストのあったURLとリンク元の2つをメールとしてrootに送信する。

▼リスト3　missing.cgiの例

```
#!/usr/bin/perl
print<<END;
Content-type: text/html\n\n
<HTML>
<HEAD>
<TITLE>URLが間違っていませんか</TITLE>
</HEAD>
<BODY>
        <H1><FONT COLOR="red">$ENV{REQUEST_URI}</FONT>の指定は間違っているようです。</H1>
        もう一度入力したURLを確認してください。
</BODY>
</HTML>
END

open (MAIL,"|sendmail -t");
print MAIL<<MSG;
To: root
From: WEB MASTER<root>
Subject: WEB:NOT FOUND(404)

REQUEST URL:$ENV{REQUEST_URI}
REFERER URL:$ENV{HTTP_REFERER}

MSG
close MAIL;
exit;
```

■404、408、500のエラーに対処

Apacheの返すエラーコードは多数ある。これらすべてに独自のページを作っていてはあまりにも手間がかかる。最低限なにか1つというならば、404のファイルが見つからないときだけを押さえていればなんとかなる。

これに加えて、スクリプトが動いている環境下なら、エラーコード500のサーバエラーを、アクセス数が多くタイムアウトが続出するようなところであれば、408のリクエストのタイムアウトあたりもだ。このあたり、作れるならばいくらでも作ってオリジナルサ

Column

コンフィグレーションファイルを書き換えたらhttpdのrestartを忘れずに

いろいろとsrm.confの書き換えを紹介してきたが、srm.confに限らず、/etc/httpd/conf以下にあるコンフィグレーションファイルを書き換えたときには、Apache自体を再起動しないと書き換えた内容が反映されない。

再起動といっても、LinuxではOSの再起動をする必要はない。Red Hatでは、/etc/rc.d/init.d/以下に起動時のデーモンを動かすクーロンファイルが一式入っている。そこで、suでrootに入り直し、再起動する手順をとる。

コンフィグレーションを書き換えたあとに、

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

とすれば、Apacheのプロセスを停止し、再度スタートという形でリスタート作業をする。

もし書き換えたコンフィグレーションファイルが正しくなければ、エラーメッセージが出るので、それを元に問題点を解消する。

```
# /etc/rc.d/init.d/httpd restart
Shutting down http: httpd
Starting httpd: httpd Syntax error on line 14 of
/etc/httpd/conf/srm.conf:
Invalid command 'ERROR', perhaps mis-spelled or defined by a module
not included
 in the server configuration
```

画面　修正したコンフィグレーションファイルに問題があれば、ファイル名と問題のあった行番号、内容が表示される。

ーバを目指せばいい。

psをすると、httpdは10個動いている

デフォルトのまま利用していると、Apacheのサーバが10個動いていることに気がつく。10個も動かしたわけではないし、アクセスしたプロセスがそのまま生きているわけでもない。

サーバ周りでいちばん負荷のかかる部分は、コネクション時の処理である。気持ちよくレスポンスを返すには、ユーザーのコネクション要求に合わせてプロセスを立ち上げずに、待ち受けて見張るという方式だ。

こうした設定は、httpd.confの250行以下で設定する。とはいえども、ごく普通のWebサイトを立ち上げる限りでは、これらの設定を変更する必要はまずない。かなりのアクセス数が見込まれるWebサイトになった段階で、初めて意味をなす設定だ。また、実際に変更すると、どれだけレスポンスが変わるかという実証も、試せる環境がないので、ドキュメントの受け売りで触れておこう。

デフォルトの設定を見ていくと、**表**のようなコマンドが目につく。アクセス数の多いサイトではこの要素を書き換えることで大きく動作環境が変化していく。

Timeoutは、なにかしらの原因でクライアントが接続しっぱなしになったとき、サーバ側がタイムアウトとして接続を切断する時間だ。あまりにもトラフィックが高いサーバへのアクセス時や、接続経路のトラフィックが高いときに、この設定は有効になるはずである。既定値は5分になっているが、これ以前にクライアント側のブラウザがタイムアウトで接続を切るはずなので、もう少し小さな値でもかまわないはずだ。

KeepAliveは、一度コネクションを張ったポートに対して、連続した要求を受け付けるというものだ。Webブラウザは、HTMLの読み込み、画像の読み込みと、いくつものコネクションを連続して発生する。このとき、KeepAliveで接続すると、コネクションを張り直さないため、サーバとネットワークトラフィックの負荷が軽減する。最近のブラウザはKeepAliveできるため、通常どおり使っていれば何も考えずにKeepAliveしている。ただし、KeepAliveは、CGIには適用されない。あくまでもファイルサイズが決まっているもののみ適用される。

なお、ブラウザによって、KeepAliveを正しく処理できないどころか、フリーズするものがある。これを除外するために、BrowserMatchというコマンドを使い、特定ブラウザではKeepAliveしないという設定になっている。あえて修正する意味はないが、srm.confの243行目にある、

```
BrowserMatch          "Mozilla/2"
```

コマンド	既定値	説明
Timeout	300	クライアントの接続タイムアウト設定
KeepAlive	On	キープアライブ設定
KeepAliveTimeout	15	キープアライブのタイムアウト設定
MinSpareServers	8	最少のスペアサーバー数
MaxSpareServers	20	最大のスペアサーバー数
StartServers	10	起動時のサーバ数
MaxClients	150	最大クライアント数
MaxRequestPerChild	100	最大の子プロセス起動数

表2　Apacheの動作速度を改善する設定

```
[user]# ps ax | grep httpd
104  ?  S    0:00 httpd
106  ?  S    0:00 httpd
107  ?  S    0:00 httpd
108  ?  S    0:00 httpd
109  ?  S    0:00 httpd
110  ?  S    0:00 httpd
111  ?  S    0:00 httpd
112  ?  S    0:00 httpd
113  ?  S    0:00 httpd
114  ?  S    0:00 httpd
115  ?  S    0:00 httpd
122  p2 S    0:00 grep httpd
```

画面3　デフォルト状態でhttpdの動作プロセスを表示。10個のApacheが動いている。

```
nokeepalive
BrowserMatch    "MSIE    4\.0b2;"
nokeepalive downgrade-1.0 force-
response-1.0
```

はそのための設定である。

MinSpareServers、MaxSpareServers、StartServersが、いくつものhttpdプロセスを起動している張本人だ。書いてあるとおりで、起動時に10個のサーバプロセスを立ち上げる。

ちなみに、このStartServer値を8にしたところ、システムが使っているメモリ量が200Kバイトほど小さくなる。確かに有効な手段ではあるが、数を減らしたところでそれほど大きな節約にはならない。また、StartServer値を下げても、MinSpareServersの値も小さくしなければ、起動時はサーバプロセス数が少なくても、動かしている間にMinSpareServersの値までサーバプロセスが増える。プロセス数をコントロールするならば、StartServerとMinSpareServersはしっかり設定していかないといけない。

MaxClientsは、同時接続を許す最大のクライアント数を指定する。設定値は150で、KeepAliveが有効に動いていれば、150クライアントが接続できる。

Linuxの日本語環境

入力システム

文：富樫秀昭
TEXT：Hideaki Togashi

前々回に日本語の表示と入力を扱ったが、個々の入力環境に具体的に触れることまではできなかった。ちょうど、TurboLinux日本語版4.0にATOK12 SE for Linuxがバンドルされて出荷されたところなので、ここで日本語入力システムをざっと一望してみたい。個別のシステムを紹介する前に、UNIXの日本語入力システムについて、視点を変えてまとめ直しておくことにしよう。

●かな漢字変換サーバ

日本語入力では、かな漢字変換を欠かすことができない。UNIXの環境では、そのかな漢字変換を行う機能を文字入力から切り離している。つまり、入力そのものとは別に、かな漢字変換サーバを持つのである。標準的にネットワーク上で運営されるUNIX環境で好んで用いられるサーバクライアントモデルで実現されているのだ。変換機能をサーバとして別に持つことによって、

・変換サーバを別のホストに置くことによって、入力を行うマシンの負荷を減らすことができる。
・変換サーバという資源を共有することによって、資源を節約することができる。
・同じ変換サーバを使うクライアントすべてで同じ日本語環境を使え、辞書や、頻度情報などのかな漢字変換に必要な情報を共有することができる。
・各アプリケーションでも、同じ日本語入力環境が使える。

などのメリットが生まれたのだ（**図1**）。

●アプリケーションとかな漢字変換サーバ

かな漢字変換サーバとやり取りを行うクライアントには、いくつかの種類がある。

まず、ユーザーから入力を受け取ることが主な機能のひとつとなるエディタは、その機能を利用し、それを拡張して直接かな漢字変換サーバとやり取りすれば、かな漢字変換機能を実装したエディタとしてふるまうことができるようになる。これは日本語を扱うエディタとしては、非常に大きな魅力だ。Muleに代表されるEmacs系のエディタが、この機能を持つ。これらのエディタは、変換サーバさえ動いていれば、かな漢字変換を行うことができ、後述のフロントエンドプロセッサや、XのInput Methodなどは必要ない（**図2**）。必要なのは、かな漢字変換サーバを立ち上

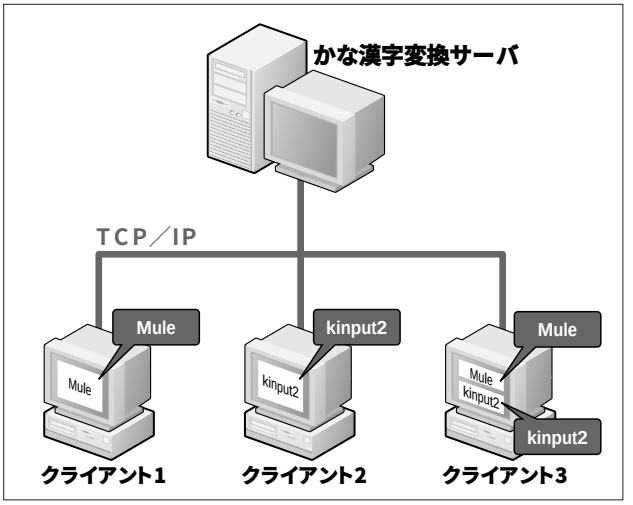


図1　サーバクライアントモデル

げておくことだけである。もちろん、Muleコンパイル時にかな漢字サーバとやり取りするために必要なコードを組み込んでおく必要がある。vi系のエディタには、かな漢字変換サーバとやり取りするためのライブラリとしてONEW（おニュー）があり、これを組み込んだjvim、jelvisなども、かな漢字変換サーバとやり取りすることができる。

●端末用日本語入力フロントエンドプロセッサ

しかしこれらのアプリケーションと直接やり取りする方法では、それを使っている間はよいのだが、使っていない時にはかな漢字変換を行うことができないことになってしまう。そこで、端末レベルでかな漢字変換機能を提供し、その端末上で動くアプリケーションならば何でも日本語入力できるようにと考えられたのが、端末用の日本語入力フロントエンドプロセッサである。具体的には、Wnn4のuum（うーむ）、Cannaのcanuum（きゃにゅうむ）などが、これに当たる。これらの日本語入力フロントエンドプロセッサは、端末とアプリケーションとの間に介在して、端末から受け取った入力をかな漢字変換サーバとのやり取りで変換して、変換後のデータをアプリケーションに渡すのである（**図3**）。

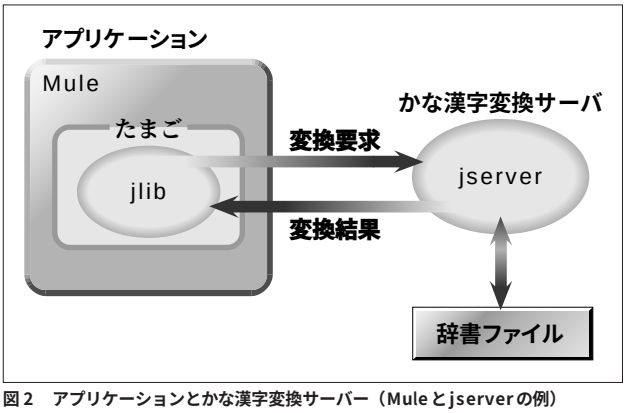


図2　アプリケーションとかな漢字変換サーバー（Muleとjserverの例）

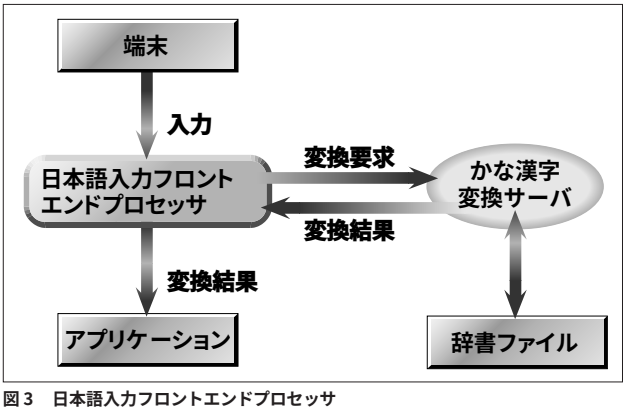


図3　日本語入力フロントエンドプロセッサ

この日本語入力フロントエンドプロセッサを使えば、端末上で動作するアプリケーションならば何でも日本語入力をすることができ、後述するXのInput Methodのように X Window Systemを必要とすることもない。かな漢字変換サーバと、この日本語入力フロントエンドプロセッサを立ち上げておきさえすればよいのである。一方、Xのアプリケーションのように端末で動作しないアプリケーションには無力である。

●XIM（X Input Method）

Xのアプリケーションがかな漢字変換サーバと個別にやり取りするのは非効率的である。そこで、X Window Systemにおいては、Xのアプリケーションとは別に日本語入力を行うプログラムを用意し、Xの各アプリケーションがそのプログラムとやり取りをして、入力を行うプログラムがかな漢字変換サーバとやり取りを行うという方式が取られている（**図4**）。その入力を行うプログラムは、かな漢字変換サーバに対してはクライアントとして動作するのだが、Xのアプリケーションに対してはサーバとして動作するので、入力サーバという言い方もされる。入力サーバとXのアプリケーションとの間のやり取りは、X11R6で定義されたXIM（X Input Method）というプロトコルで行われるので、この入力サーバはInput Method、もしくは単にIMと呼ばれることもある。IMの代表的なものとしてkinput2があげられる。

アプリケーションがこの入力サーバとやり取りを行うには、一定の手続きが必要となる。まず、当然のことながらかな漢字変換サーバを立ち上げておいてから、

1. 環境変数XMODIFIERSで入力サーバを指定する。

```
$ export XMODIFIERS="@im=kinput2"
```

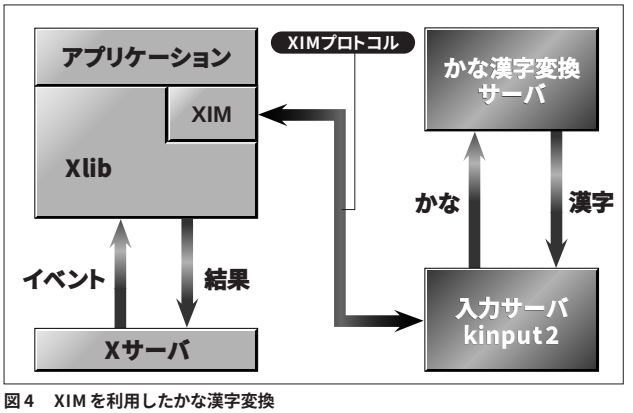


図4　XIMを利用したかな漢字変換

2. 入力サーバを利用するアプリケーションのリソースを指定する。たとえばktermなら、~/.Xresourcesなどに次のように指定する。

```
KTerm*VT100*translations:  #override \
  Shift<Key>space: \
      begin-conversion (_JAPANESE_CONVERSION)
KTerm*allowSendEvents:     true
```

1行目は「Shift＋スペースキー」で日本語変換を開始することを指定しており、2行目は入力サーバにイベントを送ることを許す指定である。

3. 入力サーバを立ち上げる。

以上が、入力サーバを利用するための一般的な手順だ。通常は、ドットファイルなどで自動的に設定されるようになっている。以上の手続きを済ませておけば、そこで指定したアプリケーションで、指定した入力サーバを利用できるようになるのである。

この方式の利点は、Xの各アプリケーションに統一的な日本語入力システムを提供できる点である。ktermはそれ自身端末であるが、同時にXのアプリケーションなので、同じIMが使えるのである。一方、X Window System以外では、この入力方式はもちろん利用することはできないし、Xの環境の中でもエディタなどのように入力作業が煩雑なアプリケーションでは、IMの操作性とエディタ自身の操作性に一貫性がないと使いにくい場合もあるだろう。

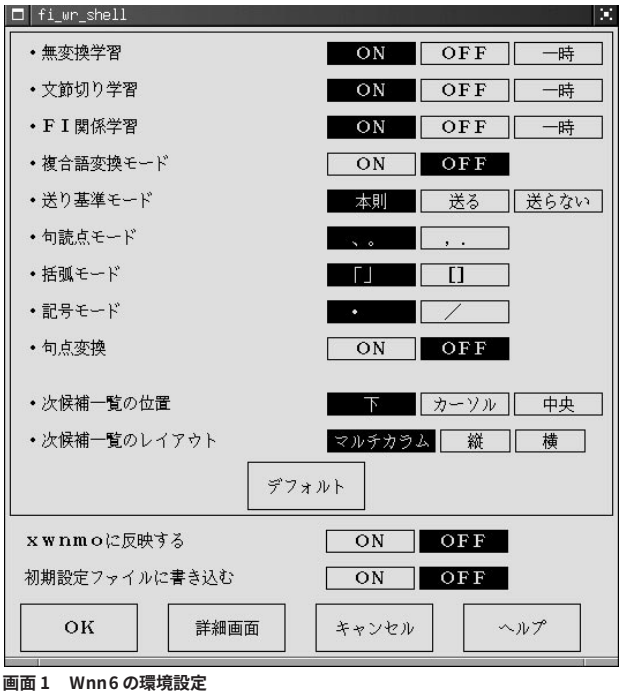
このように、UNIXの日本語入力システムの仕組みは、一様ではない。変換効率を支えるのがかな漢字変換サーバであることは間違いないが、入力時の操作性を支配するのは主にそのクライアントであり、動作する環境によって利用できるクライアントが異なるのだ。この点に留意しながら、以下各日本語入力システムを見ていくことにしよう。

Wnn6 for Linux／BSD Ver.3.0

Wnn（うんぬ）という名前は、「わたしの名前は中野です」（中野さんというオムロンの社員がいた）を一括変換できるように作られたシステムであることから、その各文節の頭文字を取ったものである。当時は一括変換できるということが画期的なことだったのである。開発開始が1985年、最初のリリースは1987年。Wnn4までは京都大学、オムロン、アステックによって共同開発されており、その後はWnnコンソーシアムとオムロンによってバージョンアップが継続して行われ、Wnn4.2でX11R6がサポートされている。商品としては、オムロンが共同著作権者の同意を得て、1995年にWnn6、1996年にWindows 95版のWnn95、1999年にWindows 98に対応したWnn98がオムロンソフトウェアから発売されている。一方で、Wnn4がフリーのソフトウェアでありながら制限もあり、商品のWnnとの混乱もあったため、Wnn4.2のライセンスをGPL2としたFreeWnn 1.0が1999年4月に公開されている（FreeWnnの公式サイトはhttp://www.freewnn.org/）。

ここで取り上げるのは、商品のWnnの最新バージョン3.0である（以下Wnn6と記述のあるものは、このバージョンのWnn6のこと）。単体でも発売されているが（9800円）、TurboLinux日本語版4.0にもバンドルされている。Wnn6の公式サイトはhttp://www.omronsoft.co.jp/SP/unix/wnn6/である。

Wnnの特徴を見ると、開発時の精神を垣間見ることができる。特に、変換効率、サーバクライアントモデル、ユーザーがなるべく自由に資源を利用できるようにすることという3点については、開発時から一貫して守られてきた特徴である。Wnn6は、商品としてそれをさらに強化したものだ。FI（Flexible Intelligence）機能によって変換効率を向上し、サーバクライアントモデルを一歩進めて活用



画面1　Wnn6の環境設定

画面2　xwnmoのルートウィンドウ

することによってオフライン学習機能、Windows版のWnnの提供などを行い、またユーザーの利便を考えて他の日本語入力システムと辞書のやり取りを行うツールを提供している。岩波書店の国語辞典をもとにした辞書が付属している。他の日本語入力システムにはないWnnの特徴として、韓国語、中国語を扱えることがあげられるが、Wnn6にはそれらの変換サーバは含まれていない（Wnn for Javaで多国語をサポートし、Windows版でそれぞれの入力システムが個別に販売されている）。

Wnn6の日本語変換サーバはjserverであるが、この他に変換サーバにライセンス供与を行うライセンスサーバdpkeyservが動いている必要があり、また変換サーバと独立に辞書引きサーバwnndsを稼動することもできる。クライアント（IM）ととなるxwnmoは、GUIで環境設定などを行うことができるものだが（**画面1**）、TurboLinux日本語版4.0でインストールを指定してもデフォルトで起動されることはなく、マニュアルの説明に従って起動する必要がある。基本的に、XMODIFIERSの指定とXアプリケーションのリソース指定だけだが、デフォルトで起動されるようになっているkinput2は起動しないようにし、さらに現在使っているマシンにtelnetでloginできるようにネットワーク設定を行う必要がある。詳細は、/usr/local/OMRONWnn6/doc/にあるマニュアルを参照していただきたい。xwnmoが正常に立ち上がれば、「Ctrl＋¥」でルートウィンドウが表示されるはずだ（**画面2**）。これで日本語入力が可能になる。

jserverのクライアントには、xwnmo以外にも、IMではkinput2、日本語入力フロントエンドではuum（うーむ）、そしてMuleのたまごなどが利用できる。ただし、Wnn6にはこれらは付属しておらず、FreeWnnに付属しているものではWnn6の実力を完全に引き出すこともできないようだ（uumはWnn6製品版には入っておらず、TurboLinux日本語版4.0にも入っていなかった）。xwnmoだけを利用するなら問題ないのだが、xwnmoのキーバインド（表1）はちょっと特殊であり、しかも（uum以外の）その他のクライアントとはキー操作が異なる。これは少し使いにくいだろう。たとえばMuleを多用するなら、たまごに合わせてegg風のキーバインドを選択するとよいだろう。これで、たまごのようにスペースキーで変換してくれるようになる。ちなみに、TurboLinux日本語版4.0でインストールされるMuleは、デフォルトでたまごを使うようになっている。「Ctrl＋¥」でフェンスモード（ローマ字変換のモード）に入ると、後はスペースキーで変換を行うほかは、ほとんどxwnmoと同じである。Muleの中でもxwnmoを使いたいなら、

機能	キー
変換開始	Ctrl＋¥
変換	Ctrl＋W
確定	Ctrl＋L／Enter
文節拡張	Ctrl＋O
文節縮小	Ctrl＋I
ひらがな変換	F6
カタカナ変換	F7
半角変換	F8
ローマ字変換	F9

表1 xwnmoの主なキーバインド

```
$ mule -nw
```

と、端末モードでMuleを使う。

かんな

かんなは、日本電気によって開発されたフリーソフトウェアの日本語入力システムだ。1989年に開発が始められ、1991年にバージョン1.2が公開されている。その後現在まで、UNIX上のフリーソフトウェアとして配布されている（Windows版は1998年からCanna for Windows 95が日本電気より販売されている）。かんなのホームページが、http://www.nec.co.jp/japanese/product/computer/soft/canna/に設けられている。

かんなの特徴は、開発時のコンセプトによく現われており、それが現在もそのまま継承されている。

- 複数のかな漢字変換の同時処理
- 簡単なライブラリの提供
- 豊富なカスタマイズ機能の提供
- カスタマイズファイルの一元化

ここに隠されているキーワードは、「統一的な入力インターフェイス」である。本稿の初めに解説したように、UNIX上の日本語入力環境は一様ではない。場面に応じて

さまざまな入力環境が提供され、そのどれひとつとしてすべての場面で使えるものはないのである。そのUNIXの日本語環境に統一的なユーザーインターフェイスを提供しようというのが、かんなのコンセプトなのだ。

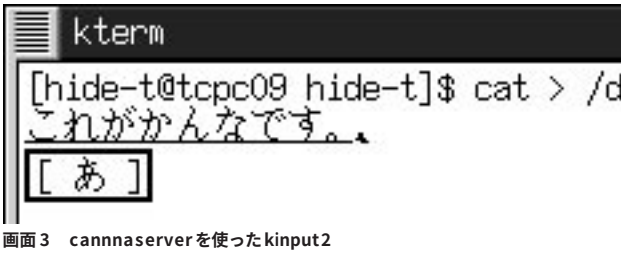
そのためのキーポイントとなるのが、2番目の「簡単なライブラリの提供」だ。ユーザーインターフェイスライブラリとそのライブラリが変換サーバとやり取りするための変換ライブラリを提供し、他のアプリケーションがそれを組み込むことによって容易にかんなを使えるようにするとともに、同じライブラリを使うことによって統一的なユーザーインターフェイスを確保できるのである。ユーザーインターフェイスライブラリが、1つのカスタマイズファイルを使うようにすれば、各アプリケーションでカスタマイズファイルを一元化できる。一方、いろいろな変換サーバとのやり取りを行う変換ライブラリを複数用意し、それらがユーザーインターフェイスライブラリとやり取りを行うAPIを統一すれば、複数のかな漢字変換を動的に切り替えて利用できることになる。実際には各日本語入力システムには特有のプロトコル、特有のインターフェイスがあり、もちろん各入力システム間で完全にシームレスな環境ができ上がっているわけではない。しかし、かんなに関する限り、この開発時のコンセプト、統一的な入力インターフェイスは、実現されていると言うことはできる。

かんなのかな漢字変換サーバーは逐次自動変換を実現したcannaserverだ。入力環境としては、日本語入力フロントエンドプロセッサのcanuum（きゃにゅうむ）、X上のIMであるkinput2（**画面3**）、Mule上の「かんな/emacs」が利用できる。そして前述のようにこれらの動作はすべて、同じカスタマイズファイル「~/.canna」でカスタマイズすることができる。

.cannaのサンプルは、かんなのインストールされたディレクトリの中のsampleというディレクトリに、いろいろな日本語入力システムに似せた設定とともに置いてあるので、それを自分のホームディレクトリに「.canna」という名前でコピーして使う。

かんなの操作体系はちょっと変わっているが（**図5**）、通常の入力はほとんど違和感なく行えるだろう（**表2**）。「Ctrl＋N」、「Ctrl＋P」で文字種を順に変更するようになっているところが、他の入力システムにはないところだ。

かんな/emacsの説明は、infoが提供されているので、MuleやXEmacsで読むことができる。また、詳しいかんなのドキュメントはTeXファイルで提供されているが、現在のLaTeXでタイプセットするには手直しが必要だった。



画面3　cannnaserverを使ったkinput2

sj3

sj3（エスジェースリー）は、ソニーが開発した日本語入力システムで、もともとNEWS-OSに付属していたsj2が改良され、フリーソフトウェアとして公開されたものだ。sj3の制作年は、コピーライトの表示によると1990年、付属のマニュアルNihongo.psのコピーライトは1994年からとなっているが、1990年以前はsj2がNEWS-OSに付属している。

sj3は、当時としては比較的変換効率が良いものだったようだ。キーバインドなどを非常にきめ細かく設定できるところも、sj3の特徴としてあげることができるだろう。

機能	キー
変換開始	Ctrl＋O／Shift＋スペース
変換・次候補	スペース
候補一覧	Ctrl＋W
次の文字種・次候補	Ctrl＋N
前の文字種・前候補	Ctrl＋P
文節伸長	Ctrl＋O
文節縮小	Ctrl＋I
中断	Ctrl＋G
確定	Enter
拡張メニュー表示	F10

表2　かんなの主なキーバインド
Mule（Emacs）のキーバインドを意識した作りになっている。

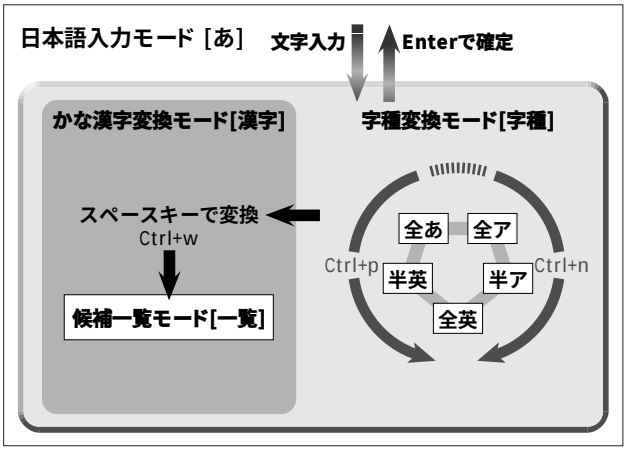


図5　かんなの操作体系

今ではたいていのかな漢字システムでかなりの設定ができるが、同じ設定ファイルの中で各設定項目を環境変数TERMの値ごとに設定ができるというものは、sj3くらいしかないだろう。

sj3が環境変数TERMの値ごとに違う設定ができるようになっているのは、sj3が端末で動くことを前提とした日本語入力フロントエンドプロセッサだからであろう（**画面4**）。日本語入力フロントエンドプロセッサなので、X Window Systemでなくても動くところが、大きな強みだろうか。うまく設定すれば、非X環境でも、Xのktermでも、設定ファイルを変更することなく、同じキー操作で日本語入力できるようになる。

設定ファイルは「~/.sjrc」である。デフォルトで使われる「sjrc」というファイルを変更して利用するのが普通だ。sj3デフォルトのキーバインドはファンクションキーを利用したもので（**表3**）、軽快なキー操作を行うにはカスタマイズは必須であろう。今となってはごく当たり前のことだが、ローマ字変換規則もカスタマイズ可能で、「~/.sjrk」というファイルで設定する。デフォルトで使われるファイル名は「sjrk」だ。sjrkもsjrcも、sj3のライブラリディレクトリに一緒にインストールされている。

sj3にはUnbufferモードとBufferモードがある（**図6**）。他のかな漢字変換システムでは起動している状態と起動していない状態に近いもので、普通に使う分にはそのような感じで動作するのだが、Unbufferモードはあくまでもバッファリングしない状態であるだけで、そこでかな文字を入れることももちろんできるが、その文字は変換できない。変換キーを押してBufferモードに入ると、入力した文字がバッファリングされるので変換可能となる。この際、入力された文字が半角カタカナでも、漢字に変換することが可能だ。整理されたシステムであるという見方もできる。

sj3そのものは、viを非常に意識した作りになっており、文節の編集モードのキー操作などほとんどviそのものであるし、viの中でsj3を使って日本語入力している際にESCでBufferモードを抜けると、そのままviのinsertモードを抜けるところなど、実に小気味がよい。viとの親和性は非常に高いといえるだろう。

sj3のかな漢字変換サーバは、sj3servである。Mule上でWnnの入力環境となっているたまごに、このsj3servを変換サーバとするsj3-eggがあり、sj3servと対話するライブラリを組み込んだMule上でこれを利用すると、sj3servを利用できる。IMとしてはsjxaがあり、これもしくはsj3servと対話するライブラリを組み込んだkinput2を使えば、Netscape CommunicatorなどのXIMで日本語入力を行うアプリケーションでも、sj3servを使うことができるだろう。

画面4　sj3での日本語入力

VJE-Delta Ver.2.5 for Linux／BSD

VJE（ぶいじぇーいー）は、PC-9801上のMS-DOSで単体のドライバとして動作する日本語入力フロントエンドプロセッサとして、かなり早い時期からバックスより販売されていたものだ。最初のVJE-αに続き、1985年には後

機能	キー
変換	F1
無変換	F2
その他	F3
決定	F4
コード	F5
半角アルファベット	F6
全角アルファベット	F7
半角カタカナ	F8
全角カタカナ	F9
ひらがな	F10
文節拡張	↑
文節縮小	↓

表3 sj3の主なキーバインド

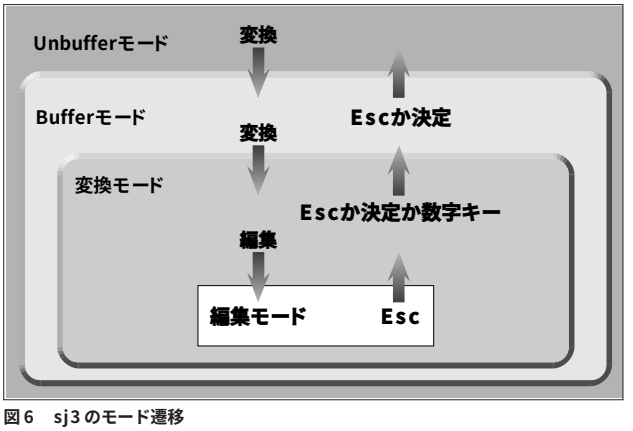


図6　sj3のモード遷移

継バージョンとなるVJE-α Ver.1.10とVJE-Σ Ver.1.00が発売されている。比較的早いうちからユーザーがキーアサインなどをカスタマイズできるようにしたシステムだったように記憶している。UNIX版のVJEは、SunOSの日本語ウィンドウシステムJSunView（1987年）にフロントエンドプロセッサのVJE-αが搭載されており、1991年からアスキーが発売していたMotif日本語版GUIであるOpenWare/Motifにはクライアントサーバモデルの VJE-βが搭載されている。Linux／BSD用として発売されているのは、VJE-Delta Ver.2.5である。価格は9800円。今回はこれを試用した。バックスのホームページ（http://www.vacs.co.jp/）にあるアップデートデータでアップデートを行っている。

付属のマニュアルで特徴としてあげられているのは、以下の5点である。

・AI変換アルゴリズムの採用
・XIMプロトコルに準拠
・快適な入力と変換
・豊富な辞書
・充実した辞書メンテナンス機能

格フレーム辞書を使い、離れた文節の関連性を解析するAI変換アルゴリズムを採用しているとのことだ。

XIMプロトコルに準拠とあるところからもわかる通り、用意されている入力環境は、IMだけである。変換サーバのvjedと、IMのvjeのワンセットである。XMODIFIERSに「@im＝vje」を指定してからvjeを起動する。.xinitrcに、

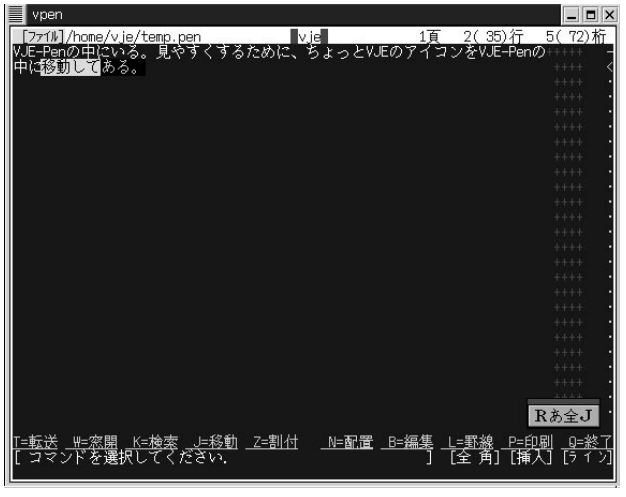
```
if [ -x /usr/local/vje/bin/vje ] ; then
   export XMODIFIERS="@im=vje"
   /usr/local/vje/bin/vje &
fi
```

と記述しておくとよいだろう。

vjeが起動していると、画面右下に「VJE-OFF」と表示したウィンドウが現われる。allowSendEventsリソースをtrueにしてあるIMに対応したXのアプリケーションで「Shift＋スペースキー」してかな漢字変換を始めると、このウィンドウの文字列が「Rあ全J」となって変換状態であることを示す（**画面5**）。面白いのはこのウィンドウの文字列をマウスでクリックするだけでモードを変更できることだ。ポインタで指示するだけだと、各文字の表わす現在の状態をバルーンヘルプで表示する。もちろん、キー入力の指示でこれらの状態を変更することもできる。そしてどの入力からも漢字変換を行うことができた。通常変換操作に必要となるキーバインドについては、**表4**をご覧いただきたい。

もちろん、カスタマイズも可能だ。「~/.vjeenv」でキーバインドなどの各種設定を、「~/vje.rom」でローマ字変換表の設定を行うことができる。設定方法などの詳細については、/usr/local/vje/doc/にインストールされるdelta.eucとdelta.sjisをご覧いただきたい。

おまけとしてワープロソフトのVJE-Penが付いており、しかもフルソース付きのフリーウェアとしてというのには少々驚いた。Xのアプリケーションになっている。VJE-Penのソースは、CD-ROMの/source/pen/ディレクトリにあるが、/source/delta/ディレクトリにはクライアント部である入力サーバのvjeと、vjeを終了させるvjekillのソースがすべて入っている。ライセンスについては、/source/delta/にあるvdl.eucを熟読されたい。



画面5　VJE-PenでVJE-Deltaを使う

機能	キー
変換開始	Shift＋スペース
変換	スペース
前候補	↑／Ctrl＋Z
ひらがな変換	F6／Ctrl＋J
カタカナ変換	F7／Ctrl＋K
全角英数変換	F8／Ctrl＋L
半角変換	F9／Ctrl＋O
確定	無変換
取消	Esc
文節区切り伸	Shift＋→／Ctrl＋W
文節区切り縮	Shift＋←／Ctrl+Q

表4 VJE-Deltaの主なキーバインド

画面6　Netscape CommunicatorでATOK12 SEを使う

機能	キー
変換開始	Shift＋スペース
変換	スペース／変換
ひらがな変換	F6／Ctrl＋U
カタカナ変換	F7／Ctrl＋I
半角変換	F8／Ctrl＋O
無変換	F9／Ctrl＋P
半角無変換	F10／Ctrl＋@
確定	Enter／Ctrl＋M
文節区切りを左へ	←／Ctrl＋K
文節区切りを右へ	→／Ctrl＋L
注目文節を左へ移動	Shift＋←
注目文節を右へ移動	Shift＋→

表5　ATOK12 SEの主なキーバインド

ATOK12 SE for Linux

ATOK（エイトック）は、ジャストシステムが開発した日本語入力システムで、当初はPC-9801シリーズ上で動作する人気のあるワープロソフト「一太郎」に付属する形で販売されていたものだ。一太郎の愛用者には馴染み深いものだろう。ATOKは、Advanced Technology Of Kana-Kanji Transferの頭文字を取って付けられた名前である。1985年に初代となるATOK3が「jX-WORD太郎」に添付する形で販売された。ATOKが単体で販売されるようになったのは、1992年のATOK7からである。1992年から発売されている日本語Solarisに、UNIX版のATOKが搭載されている。今回はATOK12のLinux版が、TurboLinux日本語版4.0に付属する形で提供された。OEM提供用のATOK12 SE（Standard Edition）for Linuxである。

本稿執筆時点では、このTurboLinuxに付属するATOK12 SE for Linuxだけが、入手可能な唯一のLinux上で動作するATOKなので、本稿もこのTurboLinux上のATOKをもとにしている。今年中にはLinux版のATOK単体でも販売されるとのことである。HTMLのドキュメントが付属しており、TurboLinuxでは/usr/doc/atok12/html/にある。

そのマニュアルでジャストシステムがATOKの特徴としてあげているのは、

・文脈解析変換
・数詞処理
・JUST MEDDLER2
・確定リピート
・半角全角変換

の5点だ。文脈解析変換は、CMで宣伝されているあれである。特徴的なのは、JUST MEDDLER2だろう。これは校正支援機能で、間違えやすい表現やかな遣いの誤り、一般名詞と間違えやすい商標名をそのつど指摘するというものである。

TurboLinuxインストール時にATOKを選択すると、デフォルトでATOKが使われるようになっており、OS立ち上げ時に変換サーバも自動的に起動する。変換サーバとしては、atok12xが/usr/bin/に置かれており、OSの立ち上げ時に/etc/rc.d/init.d/atok12seから起動されるようになっている。atok12xが利用する辞書類は、atok12xに-dオプションで辞書のあるディレクトリを指定するようになっており、辞書類は/var/dict/atok12/にまとめて置かれている。入力システムとしては、ATOKを利用するように修正されたIMのkinput2xが用意されている。これも自動で起動するようになっており、通常のkinput2と同様、ktermなどのallowSendEventsリソースをtrueにしてあるIMに対応したアプリケーションで「Shift＋スペースキー」を押すと起動する（**画面6**）。通常の変換に必要なキーは、**表5**の通りである。

atok12prxというGUIのカスタマイズプログラム（**画面7**）が付属しており、変換モードなどの基本的な設定を行えるようになっている。設定そのものは、ホームディレクトリの「.atok12.conf」というファイルに保存されるようだ。同じホームディレクトリに「.atok.sty」というファイルも作られており、ここにキーバインドやローマ字かな変換規則などが記述されているようだが、とても手作業で簡単に設定できるようなものではなかった。SEでは、これらのカスタマイズは未サポートなので、しかたがない。単

画面7　ATOK12 SEのカスタマイズプログラム atok12prx

体で発売される際には、このファイルを設定するツールが付くのだろう。

筆者の試した範囲だけでも、atok12prxでかな入力を指定してもかな入力状態にならないであるとか、マニュアルにあるキー操作を試してもマニュアルにある通りの動作をしないものがあるなどの、不完全な点がいくつかあった。単体で発売される際には、こういった不備のないことを祈っている。

ATOKのホームページはhttp://www.justsystem.co.jp/atok/。

SKK

SKKは、Simple Kana Kanji conversion programの頭文字を取って付けられた名前で、現京大教授の佐藤雅彦氏が1987年に開発した、Mule上で動作する非常に特異なかな漢字変換システムだ。GPLのフリーソフトウェアである。かな漢字変換システムとして特に特徴的なのが、一切文法解析を行わない点だろう。その代わりに、漢字変換部分をユーザーに明示的に指定させる方法を取っている。基本的な考え方からして180度異なるのである。

たとえば「漢字で入力。」と変換するには、

```
Kanji deNyuuryoku .
```

というように入力する。つまり漢字の始まり部分をShiftキーを押して大文字で明示的にユーザーが指定するのである。変換はスペースキーで行う。

「感じがいいやん。」と送りがな付きの入力を行う際には、

```
KanJigaiiyan.
```

と、送りがなが始まる部分でやはりShiftキーを押して大文字で送りがな部分をユーザーが明示的に指定する（**画面8**）。送りがな位置の解析も行わなくてもよいために、変換効率がさらに向上するという仕組みなのである。漢字変換部分の判断を完全にユーザーにゆだねているので、ユーザーは確実に意志を反映させることができるのだ。小指に負担のかかるシステムでもあるが、名詞がすべて大文字で始まるドイツ語を入力することを考えれば、大きな違いはないだろう。

辞書には送りがな情報が含まれており、フリーのソフトウェアとしては信じられないほど大きな辞書となっている。多くのSKKユーザーの貢献によるものだ。フリーだからこそという言い方もできるかもしれない。

これは、SKKの辞書管理の仕組みによるところも大きい。変換時に選択された候補を次々とホームディレクトリの個人辞書に送り込むことによって、各個人の変換効率を向上させる仕組みになっているのだが、そのホームディレクトリに置かれた個人辞書に、ユーザーが辞書登録した内容も書き込むようになっているのだ。辞書の差分を取るツールがあるので、それで共有辞書と個人辞書の差分を取れば、ユーザーが登録した辞書の内容を簡単に取り出すことができる。

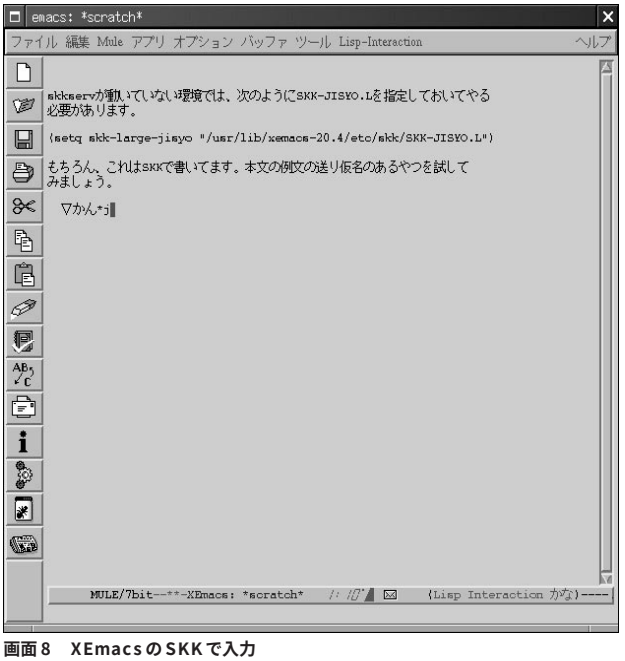
辞書登録の作業も、完全にシームレスになっており、他のシステムのようにわざわざ登録のために特別な作業を開始する必要がない。辞書にない変換を行うと、変換の継続動作としてそのまま辞書登録モードに入るのだ。そこで入力した内容が、変換結果としてそのまま使われ、同時に個人辞書にも記録される。いったん入力を行ってから、改めてそれを辞書に登録するという煩雑さが一切ないのである。しかも、辞書登録モードで行った変換がやはり辞書にない変換であった場合も、再帰的に新たな辞書登録モードに入る仕組みになっている。SKKの主なキーバインドは**表6**のとおりだ。

SKKは開発当初、Emacs Lispと共有辞書だけからなる非常に単純なシステムだった。しかしこれではMuleに負担がかかり過ぎるので、Muleの負担を軽減するために共有辞書にアクセスするサーバskkservが別に作られ、通常はこれが使われる。しかし、SKKはskkservとやり取りができないとその場で共有辞書を使うようになっている

ので、たとえskkservがダウンしてもSKKはそのまま使うことができる。管理者のいないマシンでも、安心して使うことができるのだ。

Emacs Lispのマイナーモードで実現されているので、Emacs／Muleとの親和性は極めて高い。日本語の表示できるEmacsならどんなEmacsでもSKKが使えると考えてよいだろう。それだけでなく、SKKを使っているとEmacs独自のインクリメンタルサーチを行う際にも日本語がそのまま使えるし、英字で1文字打ってからM－／（M－x dabbrev－expand） でバッファ内の英字を参照補完できるように、大文字で1文字入力してからTabキーを押すことによって、見出し語の補完を行うこともできる。これはタイプ数を減らすのに大きく貢献してくれる業だ。などなど、エディタを作っている言語で作られた日本語入力環境ならではの業が、これでもかというくらいてんこ盛りになっている。詳しくは、付属のマニュアルskk.psをご覧いただきたい。

では、SKKはMuleなどのEmacs系のエディタでしか使えないのか。いえいえ、日本語入力フロントエンドプロセッサとして動作するskkfep、IMとして動作するskkinputなど、次々と開発が進んでいる。これらの情報については、http://hiroshima.cool.ne.jp/deton/skkenv/にポインタがまとめられているのでこちらをご覧いただきたい。本家SKKのホームページは、http://skk.kuis.kyoto-u.ac.jp/skk/だ。



画面8　XEmacsのSKKで入力

では、どの入力システムを使うか…

この問題に結論を出せるとは考えていないし、結論を出すつもりも無い。ここでは、UNIX上の日本語入力システム特有の問題を、まとめておくにとどめておこう。

まず第1に、そもそもサーバクライアントモデルの日本語入力システムの場合、root権限の無いユーザーには希望するシステムが利用できるとは限らないという問題がある。PC UNIXのLinuxでは、ユーザーが同時に管理者でもあるケースも多いので、その場合はこの問題は回避可能だ。

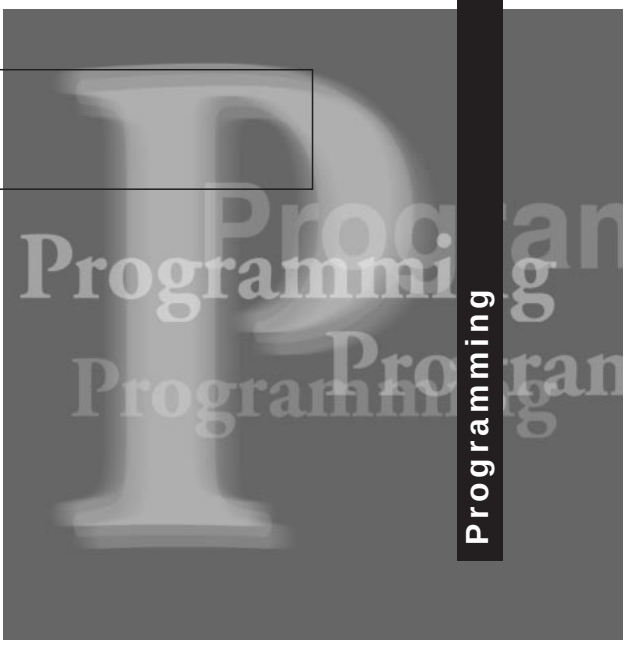
ユーザーが自由に環境を構築できる場合でも、UNIX上の日本語システム特有の問題がもう1つある。最初に見たように、利用環境によって日本語入力を行うシステムが異なるので、ユーザーがどの環境で主に日本語入力を行うかによって、選択肢が変わってくるのである。少なくともX Window System上では、現在はほとんどの入力システムのIMが利用できる状況となっているので、問題も少なく、選択肢ももっとも豊富である。一方、Xを導入していない環境では、フロントエンドプロセッサタイプのuum、canuum、sj3、skkfepのどれかを利用するか、Muleを起動して、Wnnのたまご、かんな/emacs、SKKを利用することになる。たとえX上で作業するにしても、筆者のように常にMuleないしEmacsで作業を行うような場合には、IMよりも、たまご、かんな/emacs、SKKなどの方が操作性の面で大きなメリットがある。いろいろな環境で利用することを考えた場合は総じてやはり、以前からUNIX上で使われ、こなれたシステムに一日の長のある感は否めない。

各自の利用環境に合わせて、使いやすいシステムを選択して欲しい。

機能	キー
かなモードとカナモードのトグル	q
かな／カナモードからアスキーモードへ	l
かな／カナモードから全英モードへ	L
アスキー／全英モードからかなモードへ	Ctrl＋J
変換	スペース
中止	Ctrl＋G
確定	Ctrl＋J
辞書モードでの確定	Enter

表6　SKKの主なキーバインド

Programming



プログラミングは難しい。せっかく入門書を読破しても、自分で何を作ればいいかわからない。これでは、インストールしたあと何をすればいいかわからないLinuxユーザーだ。そこで、本コーナーではUNIXプログラミングの王道、C言語を題材に実用的なプログラミングテクニックを解説していく。手始めに、簡単なネットワークプログラミングを紹介しよう。

「80番ポートからメールを読む」

文：藤沢敏喜
Text:Toshiki Fujisawa

プログラミングを始めるワケ

昨今、FreeBSDやLinuxが注目されるのは、自分で簡単に便利なプログラムが作れるという点も理由のひとつだと思う。プログラミングは、知的好奇心を満足させてくれるだけでなく、有用なプログラムを作成して公開すれば世界中の人に感謝される。Linuxの各種ディストリビューションインストールやウインドウマネージャ環境の評価に飽きたら、ぜひともプログラミングの世界にも興味を持ってほしいと思う。そこで、月刊化1号となる今回から、これまでと趣を変えUNIXプログラミングの世界を紹介していこう。

さて、ひと口にプログラミングと言っても、シェルスクリプトやAWK、Perlを使った簡単なものから、C言語を使った巨大なGUIアプリケーションまでさまざまである。GUIアプリケーションは見かけが派手なので、その開発をやってみたいと思う人も多いだろう。しかし、X Window SystemのプログラミングはGTK+などのツールキットの出現により以前より簡単になったとはいえ、初心者が取り組むにはまだまだ敷居が高い。かといってC言語の入門書にあるような課題をやるのはつまらないし、最終的になにか有用なことができなければ取り組む気も起きないだろう。

そこで、今回はネットワークプログラミングの初歩について解説する。インターネットのプロトコルは文字のやり取りが基本であるため、プログラミングは簡単であり、telnetなどを使ったテストができるし、デバッグもやりやすい。また、短いプログラムでもかなり面白いことができるので、初めてプログラミングをするという人でも興味を持って取り組めるのではないかと思う。

ネットワークプログラミングにPerlを利用しても、簡単にかなり高度なものも作成できるが、ここではより本格的にC言語を使ってみる。なぜなら、C言語に習熟しておけば、さまざまなアプリケーションの作成はもちろん、カーネルの理解やその改造も可能になるなど、Linuxを楽しむうえで世界が広がるからである。また、C言語はPerlと比べると（CPUタイムやメモリの消費などをとっても）非常に軽いプログラムが作成できる。したがって、Perlでは重くなりすぎるような処理でも、C言語なら快適なサービスを提供することも可能であろう。

モバイルのためのメールアクセス

さて、今回の第一目的は「C言語によるネットワークプログラミング」の紹介ではあるものの、最終的になにか有用なことができなければ面白くない。そこで、今回はWebブラウザさえあれば、全世界どこにいてもメールを読めるようなプログラムを作成してみることにしよう。

メールの発達にともない、いつでもどこでもメールを読みたいという人が多くなっているようだ。出張が多い人に

対しては一般プロバイダのアドレスを使うようにしている会社も多いし、自分で契約したプロバイダのプライベートなメールアドレスを持っている学生も相当数になるだろう。

こうなってくると、社内や学内にいても一般プロバイダあてのメールを読みたくなるのは当然の欲求なのだが、それは簡単なことではない。なぜなら、普通の実用環境にはセキュリティの観点からファイアウォールがあり、POP3プロトコルが使う110番ポートの穴が開いていないことがほとんどだからだ（**図1a**）。また、社内や学内から電話線による接続を禁止している組織も多い。しかし、どんなにセキュリティが厳しい会社や学校でもWebブラウザが使う80番ポートは確実に開いているはずだ。たとえ80番が直接アクセスできなくてもProxy経由でHTTPプロトコルを用いれば外部との通信が可能になる（**図1b**）。

そこで、今回は外部のレンタルWebサーバ上で、POP3プロトコルをHTTPプロトコルに変換するpop3gw.cgiという名前のプログラムを作成し、HTTPが使う80番ポートを通してメールデータにもアクセスできるようにする。つまり、Webブラウザさえあればメールを読むことができるようにするのである。

これを応用すると、たとえば海外出張で訪問した客先の会社のPCをちょっと借用してメールを読むこともできるだろう。また、Netscapeを利用してメールサーバから持ってきたメールデータをフロッピーディスクなどに保存しておけば、あとから使いなれたメールソフトで、フロッピー内に格納されたメールデータを自分のノートPCなどで読むといったことも可能である。

なお、今回はこの記事の検証のため、NTT PCコミュニケーションズが今年7月より開始したレンタルWebスペース、「WebARENA Suite」（http://www.arena.ne.jp）を借りて“fujisawa.gr.jp”という独自ドメインを取得した。このレンタルサーバは、OSにLinuxを採用しており、Linuxバイナリをサーバ上で動作させられるなど、いろいろな特徴があり興味深い。

HTTPプロトコルとPOP3プロトコル

プログラムの説明に入る前に、HTTPプロトコルとPOP3プロトコルについてちょっと説明しよう。もはや誰でもアクセスするようになったWebだが、実際にどのようなプロトコルを使っているかを知っている人は案外少ないようだ。Webは見かけが派手なため、非常に複雑な方法で通信していると思っている人も多いかもしれない。しかし、インターネットの通信プロトコルはシンプルなものが多く、例に漏れずこのHTTPプロトコルもとても簡単なプロトコルである。

たとえば、

```
$ telnet www.ascii.co.jp 80
```

としてwww.ascii.co.jpというマシンと80番ポートを利用して通信を開始すると、

```
Trying 210.140.231.23 port 80...
Connected to www.ascii.co.jp.
```

というメッセージが出てくるので、

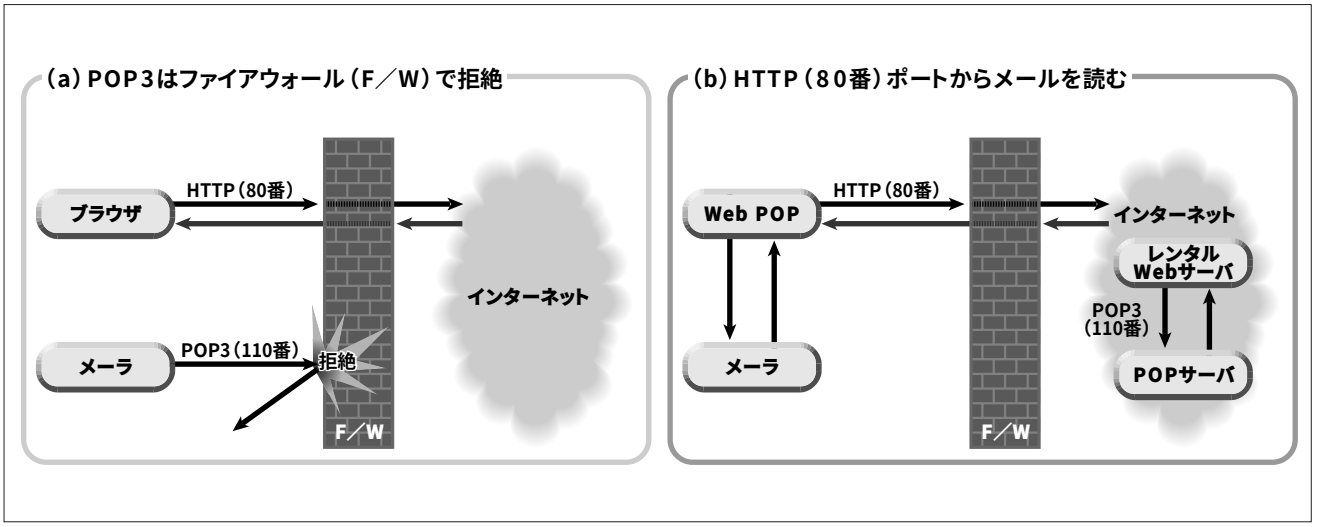


図1　ポート番号におけるアクセス制御

```
GET /linux/index.html HTTP/1.0
```
※ 空行（単にEnterを押す）

というように入力すると、そのサーバの/linuxディレクトリにあるindex.htmlというファイルの内容が転送され、画面に表示される。Webブラウザは単にこのデータを整形して表示しているだけなのである。ちなみに、“HTTP/1.0”という文字列はプロトコルのバージョン番号の指定である。

一方、ポート番号が8080のProxyを使っている場合は、

```
$ telnet プロキシサーバ名 8080
```

として、

```
GET http://www.ascii.co.jp/linux/index.html HTTP/1.0
```
※ 空行（単にEnterを押す）

というようにURLをタイプすれば、やはりlinuxというディレクトリにあるindex.htmlが表示される。

一方、フォームを使ってサーバへデータを送るような場合は、たとえば、フォームの名前がarg1、arg2、arg3でその内容がaaa、bbb、cccであるときに、

```
POST /cgi-bin/my-prog.cgi HTTP/1.0
Content-Length: 29
```
※ 空行（単にEnterを押す）
```
arg1=aaaa&arg2=bbbb&arg3=cccc
```

というようにしてWebサーバのmy-prog.cgiというプログラムへデータを渡すことができる。ここで、最後の1行にあるデータ行は29文字なので、“Content-Length: 29”を指定している。my-prog.cgiへはこのデータが標準入力として与えられることになる。

POP3プロトコル

POP3プロトコルについては、Linux magazine No.2の137ページからの**「GPG -自動暗号化システムをつくる-」**ですでに説明したので詳しくは触れないが、復習の意味を兼ねて、簡単に説明しておこう。

POP3プロトコルでは110番ポートを使うので、

```
$ telnet POP3サーバ名  110
```

として接続してから、

```
USER ユーザー名
PASS パスワード
```

を送るとログインできるので、あとはLISTコマンドで現在のメールのメッセージ番号とバイト数を調べ、RETRコマンドでメッセージ番号を指定してそのメッセージを読み、DELEコマンドでその番号のメッセージを消したりすることができる。

今回のシステムでは、まずクライアントからHTTPプロトコルのPOSTを使って、Webサーバ上のCGIプログラムへPOP3パスワードを送る。そして、そのCGIプログラムはコンパイル時にあらかじめ指定しておいたPOP3サーバの110番ポートに、USER、PASSコマンドでログインし、RETRコマンドを用いて得たメッセージを標準出力へ書き出す。このCGIプログラムの出力結果は最終的にクライアントマシンへ送られることになるので、あとはそれをメールクライアントに応じたメールボックス形式で保存すればよいというわけである。

今回のシステムで必要なプログラムの規模

さて、HTTPプロトコルとPOPプロトコルが理解できたところで、今回のプログラムの規模がどのくらいのものになるかを見てみよう。今回のシステムは、レンタルWebサーバ上に置く「pop3gw.cgi」という名前のプログラム（pop3gw.c）と、クライアントマシン上で、そのCGIプログラムと通信してメールメッセージを読み込み、読んだメールを1通ごとに切り分けて、メールボックスへ自動的に保存する、「webpop」（webpop.c）という2つのプログラムから構成される。

ソースファイルは、**表1**に示したとおり、上記2つのソースファイルと、その2つのファイルで共通に用いる関数を定義したwplib.cとそのヘッダファイルであるwplib.h、そしてコンパイル方法を記述したMakefileの全部で5つのファイルからなり、全部合わせても500行に満たない小さなプログラムである。

このくらいの量なら、わりと簡単に理解できるだろうし、自分で改造するのにも難しくないと思う。なお、これらのプログラムは今回の記事向けにスクラッチから書き起こし

たもので、掲載するスペースを考慮してあまり横幅が広がらないように書いてあるため、ちょっと読みづらいかもしれない。また、複雑になるのを避けるため、パスワード確認などエラー処理が十分でない部分があることをあらかじめご了承願いたい。ここで紹介した上記のプログラムは付録CD-ROMに収録してある。

Webサーバで動かすCGIプログラム（pop3gw.cgi）

さて、2つのプログラムのうち、今回はまずレンタルWebサーバで動かす「pop3gw.cgi」というプログラムのソースコードである「pop3gw.c」の詳細について解説する。**リスト1**を見てもらえばわかるとおり、C言語で書かれているとはいえ、やること自体が単純なため、メイン関数の中身は変数宣言を除けば、25行程度に過ぎない。それではリストのコメントに示した番号にしたがって解説してみよう。なお、各モジュールの呼び出し関係を**図2**に示すので参考にしてほしい。

ファイル名	ファイル内容	行数
pop3gw.c	Webサーバ上のCGIプログラム	139行
webpop.c	クライアント用受信プログラム	85行
wplib.c	上記2つで使う共通ライブラリ	214行
wplib.h	共通ライブラリのヘッダ	16行
Makefile	メイクファイル	37行
		合計491行

表1　今回の記事で使用するファイル一覧

（1）クライアントからのデータ（pass=xxxxx）を読む

クライアントから送られてくるPOP3パスワードは、CGIプログラムの標準入力に送られるため、ここではwplib.cというファイルで定義するread_line関数を使い、標準入力から“line”というバッファへデータを読み込む。そして、標準入力から読み込まれた値が“pass=*xxxxx*”という形式であるかをチェックしている。CGI_ARGというマクロには“pass=”という文字列が定義してあるので、strlen（string lengthの意）関数によってその長さ（つまり“5”）をlenに代入している。そして、C言語の標準ライブラリ関数strncmp（string *n*文字 compareの意）を用いてlineの最初の5文字が“pass=”と一致するかを調べている。

C言語の標準ライブラリ関数strncmpは、文字列が一致していると0を返すので、不一致の場合は致命的エラーとなり、wplib.cで定義しているfatal関数を呼び、プログラムを終了する。strncmpで文字列が一致した場合は、lineで示されるメモリからlen番目の番地から始まる文字列が、パスワードであるため、そのアドレスをpasswd変数に代

▼リスト1　pop3gw.cのmain関数

```
/*(1)クライアントからのデータ(pass=xxxxx)を読む */
read_line(stdin, line);
len = strlen(CGI_ARG);
if( strncmp(line, CGI_ARG, len ) != 0 ){
        fatal("passwd arg err");
}
passwd = line + len;
/*(2)送られてきたパスワードでログインする  */
fp = tcp_fopen(POP_SERV, 110);
check_ok(fp);
cmd_login(fp, POP_USER, passwd);
/*(3)メッセージ数を調べクライアントへ返送する */
cmd_list(fp, &msg);
printf("Content-type: text/plain\r\n");
printf("\r\n");
printf("%d messages\r\n", msg.max_num );
/*(4)すべてのメールを読み出しクライアントへ送る*/
for(i=0; i<msg.max_num; i++){
        id = msg.buf[i].id;
        cmd_retr(fp, id ); /* 標準出力へ書き込む */
        cmd_dele(fp, id ); /* 読んだメールをPOPサーバから削除 */
}
/*(5)POPサーバとの接続を切り終了              */
cmd_quit(fp);
exit(0);
```

標準入力

“pass=”という文字列定数

POPサーバ名。たとえば“pop3.xxxx.ne.jp”

POPサーバとの通信用ファイルポインタ

構造体msgの先頭アドレス

現在あるメール数

メールID

入している。

(2) 送られてきたパスワードでログインする

次に110番のPOP3ポートに、wplib.cファイルで定義しているtcp_fopen関数を用いて接続している。ここでPOP_SERVというマクロには、POPサーバ名の文字列が定義してある。正常に接続すると“+OK”という文字列が返ってくるので、それを**リスト2**に示すcheck_ok関数を呼び出して検査する。この関数へは先ほど110番ポートの通信路を開いたときに返ってきた“fp”という名前のファイルポインタを渡し、その通信路のデータを読み“+OK”であるかを検査する。

そして、今度は**リスト2**にあるcmd_login関数を用いて第2引数に「POPユーザー名」第3引数に「POPパスワード」を指定してログインを行う。

リスト2のcmd_login関数では、引数で渡されたユーザー名とパスワードを、C言語の標準ライブラリ関数fprintfを使って、POPサーバへ送り、check_ok関数を呼び出している。ここでcheck_okは**リスト2**で定義される関数で、最初にwplib.cで定義されるread_line関数でPOPサーバから1行読み出し、C言語の標準ライブラリ関数strncmpで“+OK”という文字列かどうかを検査している。

(3) メッセージ数を調べクライアントへ返送する

ログインが終わったら、**リスト3**の中で定義されているcmd_list関数を呼び出し、POPサーバへ“list”コマンドを送り、メールIDとそのメールのバイト数の組み合わせをメッセージの数だけ取得する。取得した結果は次のように定義される“msg”という構造体に保存されることになる。

```
typedef struct {
    int max_num;
    struct {
        int     id;
        int     bytes;
    } buf[MAX_MSG];
} msg_t;
msg_t   msg;
```

このmsg構造体は、“max_num”と“buf”というメンバから構成されるが、“buf”は構造体の配列で、“id”と“byte”という名前のメンバからなる構造体を、MAX_MSGの数だけ配列として用意している。この構造体は、構造体の中に構造体を含んでおり多少わかりづらいかもしれない。構造体についてあまり知識のない読者は、212ページにある特別講座「ポインタと構造体」を参照してほしい。

さて、ここでは*x*番目に登録するメールIDをmsg構造体の“msg.buf[x].id”というメンバに格納し、そのIDのファイルサイズを“msg.buf[x].bytes”という構造体メンバに格納する。登録時には“x”が1から順にインクリメントされながら格納されるが、現在登録されている数を

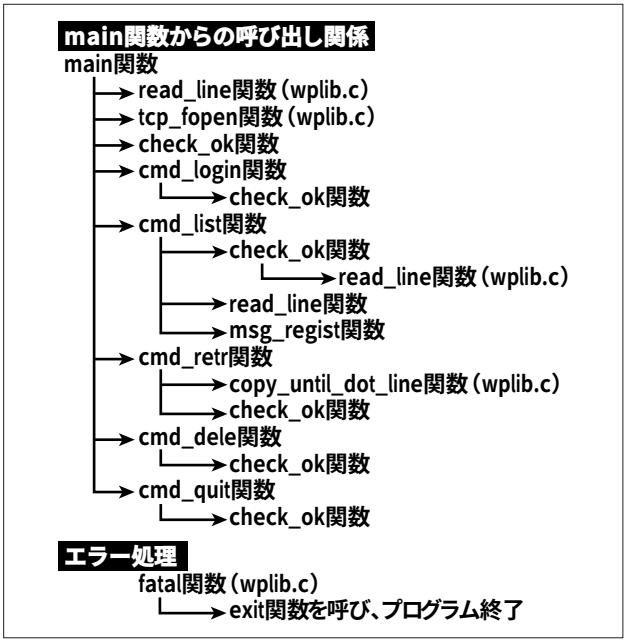


図2　各モジュールの関係図

▼リスト2　main.cのcheck_okとcmd_login

```
void check_ok(FILE *fp)
{
    char buf[MAX_STR];
    int  len = strlen(OK_STR);

    read_line(fp, buf);
    if( strncmp( buf, OK_STR, len ) != 0 ){
        fprintf(fp, "quit\r\n");
        fatal(PASSWD_INCORRECT);
    }
}
/*---------------------------------------------*/
void cmd_login(FILE *fp, char *user, char *pass)
{
    fprintf(fp, "user %s\r\n", user);
    check_ok(fp);

    fprintf(fp, "pass %s\r\n", pass);
    check_ok(fp);
}
```

“+OK”という文字列定数

POPサーバから1行bufに読む

エラー表示文字列

直後にある文字列userを代入

直後にある文字列passを代入

“msg.buf.max_num”という構造体メンバに記憶しておくことにする。この作業はwplib.cで定義されているmsg_regist関数で行われている

ここまででメールのメッセージ数がわかったので、クライアントへその数を返答することにする。CGIプログラムの場合、クライアントへの返答（すなわち、Webブラウザへの表示）とは単に標準出力へ表示することを意味するので、printf関数を使い、最初に“Content-type”というヘッダを送り、次にヘッダとデータ部を分けるために空行を送る。最後にデータとして

```
"xx messages"    ※ xxはメッセージ数
```

という文字列を送る。

(4) すべてのメールを読み出しクライアントへ送る

ここのfor文では、先ほど得たメッセージ数、つまり構造体に保存したmsg.max_numの回数分だけループする。ループ内では、前にLISTコマンドで取得したメールID、つまりmsg.buf[i].idという構造体の中のデータをidというローカル変数に保存し、関数cmd_retr(fp,id)でメールの本文を標準出力へ書き出し、そのid番号のメールを関数cmd_dele(id)で消す。この操作をforループでメッセージの数だけ繰り返すことになる。

▼リスト3　main.cのcmd_list

```
void cmd_list(FILE *fp, msg_t *msg)
{
    char line[MAX_STR];          ← 128バイト
    int id, bytes;
    int c;
    fprintf(fp, "list\r\n");     ← POPサーバへのファイルポインタ
    check_ok(fp);

    for(;;){
        read_line(fp, line);
        if( line[0] == '.' && line[1] == '\0' ){
            break;
        }
        c = sscanf( line, "%d %d", &id, &bytes );
                                 ← &idの価を代入
                                 ← &bytesの価を代入
        if( c != 2 ){
            fatal("illegual list command");
        }
        msg_regist( msg, id, bytes );
    }                            ← この2つをmsg構造体へ登録
}
```

(5) POPサーバとの接続を切り終了

ここまでで、すべてのメールを読み終わったことになるので、cmd_quit関数で接続を切断し、プログラムを終了する。

pop3gw.cgiのテスト

それでは、さっそくこのpop3gw.cgiを動作させてみよう。まず、POPサーバとPOPのユーザー名を指定するには、pop3gw.cの

```
#ifndef POP_SERV
#define POP_SERV        "mail.fujisawa.gr.jp"
#endif
#ifndef POP_USER
#define POP_USER        "pop-test"
#endif
```

という部分を書き換えればよい。そして、pop3gw.cをコンパイルして、pop3.cgiを作成し、それをWebサーバの/cgi-binディレクトリにftpで転送し、“chmod 755”として実行可能にしておく。なお、デバッグするには、ローカルホストでqpopperとhttpdを立ち上げておいて、testuserという名前のユーザー登録を行い、Makefileに、

```
CGI=/home/http/cgi-bin/pop3gw.cgi
debug: wplib.o wplib.h
       gcc -Wall -DPOP_SERV=\"localhost\"  \
                 -DPOP_USER=\"testuser\"   \
                  pop3gw.c wplib.o -o $(CGI)
```

というように書いておくと便利だ。-Dオプションでマクロを定義し、ソースファイルで#ifndefを使うとMakefileでのマクロ定義が可能になるわけである。

pop3gw.cgiのインストールが終わったら、それをデバッグしてみる。上記で指定したユーザーに、

```
$ echo "test message" | mail testuser
```

というようにしてテストメールを送り、テストユーザーのパスワードを一時的に“xxx-777”と変更しておく。そして、

```
$ telnet localhost 80
```

としてコネクトしたら

```
POST /cgi-bin/pop3gw.cgi HTTP/1.0
Content-Length: 12
                    ※ 空行（単にEnterを押す）
pass=xxx-777
```

という3行をタイプすれば、**画面1**のような返答が返ってくるはずだ。

ここまでうまく動いたら、このCGIプログラムを呼び出すページを作ってみよう。**リスト4**にあるようにフォームを使ってpop3gw.cgiを呼ぶようにし、NAME="pass"を指定し、フォームに入力した文字列が“pass=xxx”としてpop3gw.cgiの標準入力にわたるように“METHOD=POST”としておく。これをWebサーバの/pop3/index.htmlにおいてWebブラウザから、http://www.fujisawa.gr.jp/pop3/というようにアクセスすれば、パスワードを入力する画面が現れる（**画面2**）。ここでPOP3のパスワードを入力すれば、メールが表示されるはずだ。

問題点と次号の予告

以上で、当初の「どこからでもメールを読めるようにする」という目的は一応達成されたことになる。しかしながら、このままでは、単にWebブラウザの画面に表示されるだけで、MIMEエンコードされたSubjectなどは読むことができないし、何よりもふだん使いなれているメーラが使えないのでは、あまり意味がない。そこで、今回作成したpop3gw.cgiとダイレクトに通信し、MH形式のメールボックスにメッセージごとに切り分け、Mewなどで読めるようにするようなプログラムが必要になる。

このプログラムは、webpop.cとして今月号の付録CD-ROM内に収録してあるが、このプログラムに関する詳しい解説は次号で行う予定である。また、今月号で説明したプログラムでは、POPパスワードがネットワーク上を平文で流れるという問題や、途中で接続が切れたときにメールが失われてしまう問題があるので、これらの問題点についても具体的な対処法の考察も行うつもりである。

リスト4　index.html

```
<HTML>
<HEAD><TITLE>POP3GW</TITLE></HEAD>
<BODY>
<H3> POP3GW </H3>
<FORM METHOD=POST ACTION="/cgi-bin/pop3gw.cgi">
<P>PASS: <INPUT TYPE=TEXT NAME="pass"></P>
<P><INPUT TYPE=SUBMIT VALUE="GET">
</FORM>
</BODY>
</HTML>
```

今回の作成したCGIプログラム

引数の名前

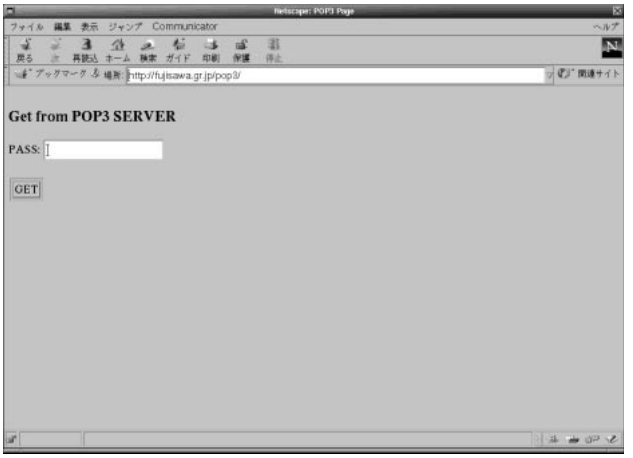
GETという名前のボタン

```
HTTP/1.1 200 OK
Date: Sat, 07 Aug 1999 04:59:17 GMT
Server: Apache/1.3.6 (Unix)
Connection: close
Content-Type: text/plain

1 messages
Return-Path: fujisawa
...
To: testuser
Date: Fri,30 Jul 1999 03:59:05 +0900 (JST)
From: Toshiki Fujisawa <fujisawa>

test message
```

画面1　testuserアカウントによる接続



画面2　パスワード入力画面

特別講座

ステップアップC言語

ポインタと構造体

■ポインタ（メモリブロックの先頭アドレス）

C言語を学習するにあたって、一番最初につまずきやすいのがポインタである。入門書などでは「実体を指し示すもの」といった説明がされていることも多いが、具体的なイメージをつかめない人も多いようだ。

このポインタを理解するには、C言語の入門書を読む前に、コンピュータがどのようにして動作するかを考えることが近道だ。コンピュータには、CPUとメモリがあり、CPUはメモリの特定の番地に保存されているデータを読み出して処理を進めていく。さまざまな処理の中では特定のメモリブロックを指定したいこともあるが、そのときには、そのメモリブロックの先頭アドレスを示せばよい。

この「先頭アドレスを入れておくもの」がまさに「ポインタ変数」である。たとえば、次のようなプログラムを考えてみよう。

```
double buf[10];
double *p;

p = buf;
for( i=0; i<10; i++){
        *p = 0.0;
             p++;
}
```

このプログラムではdouble型（8バイト）の変数10個分の「buf」という名前のメモリブロック（計80バイト）を確保しているが、ここではこのメモリブロックが1024番地から割り当てられていると仮定してみる。

まず、“p = buf”という文で、bufの先頭アドレスを“p”に代入しているが、この場合、1024という数値が“p”に代入されることになる。

次のfor文の中の、＊p = 0.0という文の意味は、「“p”に保存されている数値をメモリブロック（8バイト）の先頭番地とみなし、そのメモリブロックに“0.0”を書き込む」ことを意味している。つまり、“p”に保存されている数値が1024であった場合は、1024番地から1031番地までの8バイトにゼロが書き込まれることになる。

そして、p++を行うとdouble 1個分、つまり8バイト分だけ“p”の値が増加し、“p”の値は1032となる。

■構造体（メモリブロックの塊をまとめたもの）

構造体を用いると、変数の集合がひとかたまりのものとして表現されるため、プログラムが見やすくなったり、その内容を扱う関数を書くのが楽になったりする。

たとえば、田中さんの身長と体重を入れる構造体を定義し、180cm、79kgを代入するには、

```
typedef struct {
        long      height;
        long      weight;
} body_t;
body_t   tanaka;

tanaka.height = 180;
tanaka.weight = 79;
```

というような記述を行えばよい。ここで、heightとweightのことを構造体の「メンバ」と呼ぶ。

上記で、tanakaという構造体のメモリブロックの先頭アドレスが1024番地であると仮定すると、tanaka.weightというlong変数が割り当てられるメモリブロックの先頭アドレスは、heightの4バイト分（long）をスキップした1028番地となる。

なお、ここで“.”（ピリオド）で構造体のメンバを指定した場合には、そのアドレスがコンパイル時に決定される。つまり、1028番地のメモリブロックに“79”という数値が書き込まれるというオブジェクトコードが生成されることに注目してほしい。

さて、次に身長と体重を表示する関数を考えてみる。

```
void body_print( body_t *bp )
{
     printf("身長=%d\n", bp->height );
     printf("体重=%d\n", bp->weight );
}
```

上でbody_print関数の引数bpは構造体へのポインタ、すなわち、身長と体重が保存されているメモリブロックの先頭番地が格納される。先ほどは、構造体メンバ（heightとweight）をアクセスするのに“.”を用いていたが、ここでbpは構造体そのものでなく、構造体へのポインタであるため“->”（アロー演算子）を用いてアクセスしている。仮に引数bpに1024という数値が格納されていた場合、heightはlong変数（4バイト）であるので、bp->weightは、1028番地からの4バイトに保存されているlong型の変数を指すことになる。

なお、前述したように「構造体そのもの」のメンバをアクセスするには“.”を使うが、「構造体へのポインタ」が指し示す構造体メンバへアクセスする場合は“->”を使う。“.”はコンパイル時にアクセスするアドレスが決定されるが、“->”によるアクセスは、ポインタ変数に入っている数値（番地）に、“->”で指定されているメンバのオフセット値を足して、アクセスすべき番地をプログラムの実行時に計算することになる。

つまり、上記の例にある“bp->weight”の場合はメンバweightのオフセットが4なので、“1024+4”という演算が実行時に行われ、最終的に1028番地からの4バイト（long型）に記憶されている、79kgという数値を得てprintfすることができる。

構造体は難しいが、実際に自分で数行のテストプログラムを書き、メンバに代入して関数へ渡してみたり、ポインタのアドレスを表示したりという実験をすると理解が深まると思う。頭で考えて理解できないときは、キーボードを叩いて実験してみるのが一番である。

IMAP4（後編 クライアントの設定）

前回、ワシントン大学のIMAP4サーバ（以下WU-imapd）のインストール方法について説明した。今回は、WU-imapdの動作テストをかねて、IMAP4クライアントをいくつか紹介しよう。さらに、ApacheとWU-imapdを連動させるゲートウェイ、Wingについて紹介し、そのインストール方法も説明する。

IMAP4、生かすも殺すもクライアント次第

文：豊福 剛
Text:Tsuyoshi Toyofuku

IMAP4クライアントについて

まずは、Linuxで利用できるIMAP4クライアントをいくつか紹介しよう。

■Netscape Messenger

Netscape Messengerは、Netscape Communicatorに含まれるメーラだ。Netscape MessengerでIMAP4サーバを指定する方法を簡単に紹介しておく。以下の説明は、Linux版のNetscape Communicator 4.61を前提としている。

①Edit->Preferencesメニューを選択

左部のカテゴリからMail Serversを選ぶ。IMAP4サーバを新規登録する場合は、右部のIncoming Mail ServersにあるAddボタンをクリックする。すでにPOP3サーバを使っていて、POP3からIMAPに乗り換えるときは、Editボタンをクリックする。IMAP4の場合は、POP3と違って、複数のサーバを登録できる。

②Mail Server Propertiesダイアログの設定

GeneralタブのServer TypeからIMAP Serverを選択する。その下にあるUser Nameにユーザー名を入力、パスワードを保存したいときは、Remember Passwordをチェックする。

次にIMAPタブに移り、クライアント側でメッセージを削除したときにIMAPサーバ側でどのような処理を行うかを選択する。また終了時に、INBOXにある削除マークの付いたメッセージを削除するかどうか、Trashフォルダを空にするかどうか、といった処理も選択できる。Advancedタブについては、デフォルトのままで特に問題ない。

設定が終わると、IMAP4サーバとのセッションが開始され、フォルダ一覧にINBOXほかが表示される。

③購読フォルダの指定

購読するフォルダを指定するには、File->Subscribeを実行する。IMAP4サーバ側で提供しているフォルダの一覧が表示されるので、購読したいフォルダにチェックマークを付ける。

WU-imapdでは、デフォルトでユーザーのホームディレクトリにフォルダを作成する。このため、IMAP4のフォルダ以外のファイルもすべて表示されてしまう。これをWU-imapd側で調整する方法については後述する。

④その他の操作

フォルダを新規作成するときは、[File]-[New Folder]を実行する。フォルダ名を変更するときは、[File]-[Rename] Folderを実行する。いくつかのディストリビューションに含まれているNetscape Communicatorの日本語対応を強化したバージョンでは、日本語でフォルダ名を指定する

こともできる（Default Character SetをJapaneseにしておく必要がある）。これらの操作は、IMAPサーバ名を選択して右クリックで表示されるメニューからも実行できる。

メッセージをあるフォルダから別のフォルダに移動するには、メッセージ一覧からメッセージを選択し、それを移動先のフォルダにドラッグすればよい。元のメッセージには赤色の×印が付く。メッセージを削除するときは、DELキーを押す。これらの操作も、メッセージを選択し右クリックして表示されるメニューから実行できる。このメニューからだと、メッセージのコピーも実行できる。

検索[Edit]-[Search Messages]は、IMAP4らしい面白い機能で、IMAPサーバ側で実行できるのだが、残念ながら使用環境（Windows環境からでも）では検索条件に日本語は使えなかった。CHARSETの情報をうまくIMAPサーバに渡していないようだ。

また、フォルダ一覧に共有フォルダが表示されるものの、そこにあるはずのメッセージが表示されず、アクセスできなかった（共有フォルダの設定についても後述する）。共有フォルダのプロパティを見ると、“ This server does not support shared folders ”と表示される。いまだに原因がわからない。

■Wanderlust

Wanderlust（以下WL、http://www.gohome.org/wl/）は、寺西裕一氏を中心とするgohomeプロジェクトによる、Emacs系（Emacs、Mule、XEmacs）のための統合メール機能だ。elispというEmacsのLispで書かれている。IMAP4に特化したものではなく、POPやMH、はたまた圧縮アーカイブにもアクセスできる。WLは、Emacs系でMewを使用しているユーザーであれば、自然に移行できるインターフェイスになっている。

余談だが、wanderは、驚きではなく放浪のワンダーで、go homeもそうなのだが、レッド・ツェッペリンの歌詞によく出てくる。それにlustは、最後ではなく渇望のラストで、これもイギー・ポップのLust For Life（映画トレインスポティングでリバイバルした）を連想してしまう。なんだか70年代ロックの濃いテイストを彷彿させるネーミングだなあ。

WLのインストール、およびカスタマイズ方法については、詳細なユーザーマニュアル（日本語）が含まれているので、そちらを参照していただきたい。ここでは、IMAP4に関連する設定について簡単に説明しておく。

WLはEmacsで動くので、まずは、いずれかのEmacsがインストールしてあることが前提になる。さらに、WLは以下のelisp群を前提にしているので、これらもインストールしておく必要がある。

```
APEL    A Portable Emacs Library
FLIM    MIMEの構文解析、符号化/復号化機能
SEMI    MIMEメッセージの表示/コンポーズ機能
```

WLをインストールする前に、まずapel-9.13.tar.gz、flim-1.12.5.tar.gz、semi-1.13.3.tar.gzを入手しておく。これらは依存関係があるので、APEL、FLIM、SEMIの順にインストールする。XEmacsを使用している場合は以下のようになる。なおプロンプトが#のコマンド行はsuなどでスーパーユーザーになっていることを表している（以下同）。

```
$ tar zxvf apel-9.13.tar.gz
$ cd apel-9.13
$ make EMACS=xemacs
# make install EMACS=xemacs
```

残りのFLIM、SEMIについても同様だ。APEL、FLIM、SEMIのインストールが完了したら、次にWLをインストールする。

```
$ tar zxvf wl-1.0.2.tar.gz
$ cd wl-1.0.2
$ make
# make install
```

ちなみにelispのコンパイルとインストールはEmacs自身によって実行されている。Emacsの世界は、なんとも奥深いのである。

wl-1.0.2/docディレクトリにはtexi形式のマニュアルがある。これは、TexinfoというGNUプロジェクトで使われている汎用ドキュメント形式である。ちなみに、筆者はこれをmakeinfoしたところ、作成したInfoファイルの日本語が化けてしまった。Emacsの達人によれば、texiファイルを読み込んだ状態で、以下のコマンドをEmacsから実行するといいらしい。

```
M-x texinfo-format-buffer
```

※M-xは、Metaキー（PCの場合はAltキー）とxを同時に押下することを示す。

次にWLの動作をカスタマイズするための各種設定ファイルについて説明する。

~/.emacs　WLの起動に必要な設定情報を追加する。

```
(require 'mime-setup)
(autoload 'wl "wl" "Wanderlust" t)
(autoload 'wl-draft "wl" "Write draft with Wanderlust." t)
```

~/.wl　WLが参照する、各種変数の設定を記述する。

wl-icon-dirはWLがアイコン表示に使う画像ファイルのあるディレクトリを指すように指定する。wl-smtp-posting-serverは、メール送信に使用するSMTPサーバのドメイン名を指定する。付属するsample.dot.wlを修正するのがお手軽だ。

```
(setq wl-icon-dir "/wl-1.0.3/etc")
(setq wl-smtp-posting-server "my.smtp.server.com")
```

~/.folders　購読フォルダ名を以下の書式で指定する。

IMAPの場合、your.imap.server.comにあるINBOXにユーザー名tomでアクセスするとき、以下のように記述できる。通常は、elmo-default-imap4-serverなどの変数に指定しておき、フォルダ名にホスト名をいちいち書くことはしない。こちらもsample.foldersファイルを参照するとわかりやすい。最後の文字列で指定したエイリアス名が、WLでのフォルダ名の表示用に使われる。

```
%inbox:tom/auth@your.imap.server.com    "受信箱"
```

以上の作業が完了したら、XEmacsを起動して、次のコマンドをタイプする。

```
M-x wl
```

これで、WLが起動し、ロゴのはいったスプラッシュ画面が表示された後、IMAP4サーバにアクセスしにいく。購読するフォルダで指定したユーザー名に対するパスワードが聞かれるので、パスワードを入力する。IMAP4サーバへのアクセスに成功すると、画面には、次のように表示される。フォルダ名の後の“/”で区切られた各数字は、順に、新着、未読、総数を示す。

```
デスクトップ      0/0/14
  受信箱          0/0/8
  TestML          0/0/4
  Public          1/1/1
  Shared          0/0/1
```

カーソルをフォルダ名に移動して、スペースキーまたはEnterキーを押すと、**画面2**のような表示に切り替わる。

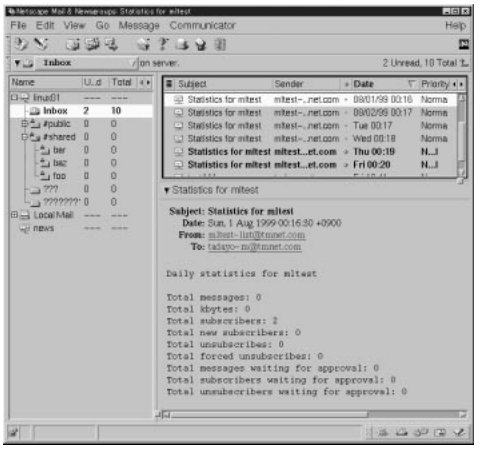
WLは、ELMOとWLの2階層で構成されている。

ELMO（elmo-*.el）　フォルダを仮装するモジュール（バックエンド）

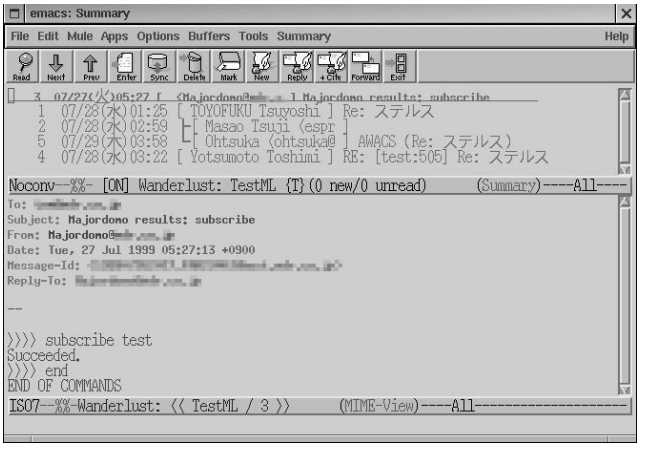
WL（wl-*.el）　本体の動作を決めるモジュール（フロントエンド）

ホームディレクトリを見ると、~/.elmo/imapというディレクトリが作成されている。この下にさらにサブディレクトリが以下の階層構造で作成され、mark、number、overview、seenという4つのファイルが保持されている。これらのローカルな情報とIMAPサーバ側の情報を照合して、フォルダの情報を更新しているのだ。

<ドメイン名>/<ユーザー名>/<フォルダ名>/



画面1
Netscape Messenger



画面2
WL on XEmacs

WU-imapdの調整

前回、WU-imapdのインストールの説明では触れなかった細かい点について補足しておこう。

①共有フォルダの設定

WU-imapdでは、#public/と#shared/の2種類の共有フォルダを提供する。#public/はimappublicユーザー、#shared/はimapsharedユーザーのそれぞれホームディレクトリに対応するように実装されている。

したがって、公共、共有フォルダを利用するときは、imappublicとimapsharedをユーザー登録しておけばよい。

使用するLinuxで、ユーザー名が8文字までしか使用できない場合、imap-4.5/src/osdep/unix/env_unix.cを修正する必要がある。env_unix.cにあるenv_init()を見ると、imappublicおよびimapsharedのユーザー名でgetpwnam()を実行している。ここで指定されているユーザー名を8文字以内にすればよい。

②各ユーザーのIMAP4ディレクトリの変更

各ユーザーが作成したフォルダは、デフォルトではホームディレクトリの直下に作成される。この場合、他のディレクトリやファイルもアクセスできてしまうので、できればIMAPフォルダは特定のサブディレクトリ以下にまとめたい。

これを変更するには、たとえば、~/imap/以下にフォルダを作成する場合、env_unix.cのenv_init()にある、

```
myHomeDir = cpystr (home);
```

を、

```
sprintf (tmp, "%s/imap", home);
myHomeDir = cpystr (tmp);
```

に変更する。

③認証機能（CRAM-MD5）の組み込み

WU-imapdでは、CRAM-MD5という認証方式が利用できる。CRAMとはChallenge-Response Authentication Mechanismの略。具体的なやりとりは次のようになる。

まずサーバはクライアントにチャレンジ値（BASE64で符号化されている）を送る。クライアントはチャレンジ値を復号化し、これとパスワードをもとにMD5のダイジェストを作成する。このダイジェストの先頭にユーザー名を連結し、さらにBASE64で符号化して作成されたレスポンス値を、サーバに返す。サーバはこのレスポンス値を復号化し、ユーザー名とダイジェストを取得する。このダイジェストと、サーバ側で作成したダイジェストが同一であるかどうかを調べることで、当該ユーザーの認証を行う。

これにより、ネットワーク上に実際のパスワード、もしくはパスワードを暗号化した値を流さなくて済む。実際には以下のようなやりとりになる（C：はクライアント側、S：はサーバ側のメッセージ）。詳しくはRFC2195を参照されたい。

```
C：A0001 AUTHENTICATE CRAM-MD5
S：+DE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2UucmVzdG9
uLm1jaS5uZXQ+
C：dGltIGI5MTNhNjAyYzdlZGE3YTQ5NWI0ZTZlNzMzNGQzODkw
S：A0001 OK CRAM authentication successful
```

この機能を組み込むには、次のオプションでmakeを実行する必要がある。

```
% make slx PASSWDTYPE=md5
```

さらに/etc/cram-md5.pwdを作成しておく。このファイルには、

<ユーザー名><タブ><パスワード>

の形式でパスワードのエントリを作成する。<パスワード>は平文で指定するので、/etc/passwd（/etc/shadow）で登録しているものと同じにしないほうがよい。当然、以下のコマンドを実行し、root以外アクセスできないようにしておく。

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

IMAP4のCAPABILITYコマンドを見ると、AUTH=CRAM-MD5が追加されているのがわかる。PASSWDTYPE=md5を指定してmakeしても、/etc/cram-md5.pwdが存在しないとAUTH=CRAM-MD5は有効にならないので、注意すること。また、

```
% make slx
```

で作成した場合でも、/etc/cram-md5.pwdがあれば、AUTHENTICATEを実行する。ただし、このときには、/etc/passwdのパスワードが照合に使用される。

このほかに、いくつか文書化されていない設定ファイルがある。たとえば、/etc/anonymous.newsgroupsを作成すると、AUTH=ANONYMOUSが組み込まれる。

```
# touch /etc/anonymous.newsgroups
```

ただし、ANONYMOUSを有効にすると、誰でもアクセスできてしまうので、これは設定しないほうがいいだろう。

WingでIMAP4とApacheを連動させる

CPANでIMAP4をキーワードに検索をかけたところ、Wing（Web IMAP/NNTP Gateway）というフリーソフトを見つけた。オックスフォード大学のMalcolm Beattieによるもので、ライセンス形態はGPLになっている。このソフトウェアは、その名が示すとおり、ApacheとIMAPを連動させるためのゲートウェイであり、IMAPをWebベースでアクセスできる環境を提供する。

Wingのインストールそのものは比較的簡単だが、実作業としては、Wingが前提としているサーバ環境の構築／設定が含まれるため、かなり複雑なものになる。具体的には、以下のソフトウェアが必要である。Wing全体のシステム構成は**図1**のとおり。

- **Apache/mod_perl**
- **PostgreSQL**
- **WU-imapd**
- **Sendmail**
- **Perl5.004以上**

Wingのシステム構成の特徴は、その論理構成にある。フロントエンドとWingサーバ、そしてIMAPサーバの3層構造になっている。この3層構造は、論理構成なので、実際には各層をどのマシンに割り当てるかを調整でき、もちろん3層を1台のマシン上で実現することも可能だ。またWingサーバは複数配置することもできる。

フロントエンドとWingサーバではApache/mod_perlを立ち上げる必要がある。Perlで開発されたWingのモジュールをここで実行する。mod_perlで実行させるのは、処理の高速化をねらってのことだろう。ちなみにmod_perlとは、ApacheとPerlの共同プロジェクトとして開発されているもので、PerlをApacheのモジュールとして実行させることができる。

フロントエンドにはApache/mod_perlのほかに、PostgreSQLを置く。フリーソフトのRDBMSであるPostgreSQLは、Wingではユーザー認証およびセッション管理のために使われている。また、ユーザー別の環境設定（アドレス帳、表示設定）もここに保存される。

フロントエンドはWingへの入口にすぎず、Wingの主要な処理はWingサーバ上で実行される。Wingサーバでは、Apache/mod_perlのほかに、maildというデーモンが稼動し、IMAPサーバへのアクセスを仲介する。

Wingでは、データベースへのアクセス、ソケット通信、MIME形式のメールの処理、クッキーの処理など、扱う内容が多岐にわたる。これだけの仕様を含むとなると、大規模なプログラムになってしまいそうだが、Perlのモジュールを徹底的に活用しているため、プログラム全体は非常にコンパクトにまとまっている。Webアプリケーションを開発するさまざまなテクニックが網羅されているので、CGIプログラマーにとっても非常に参考になるはずだ。

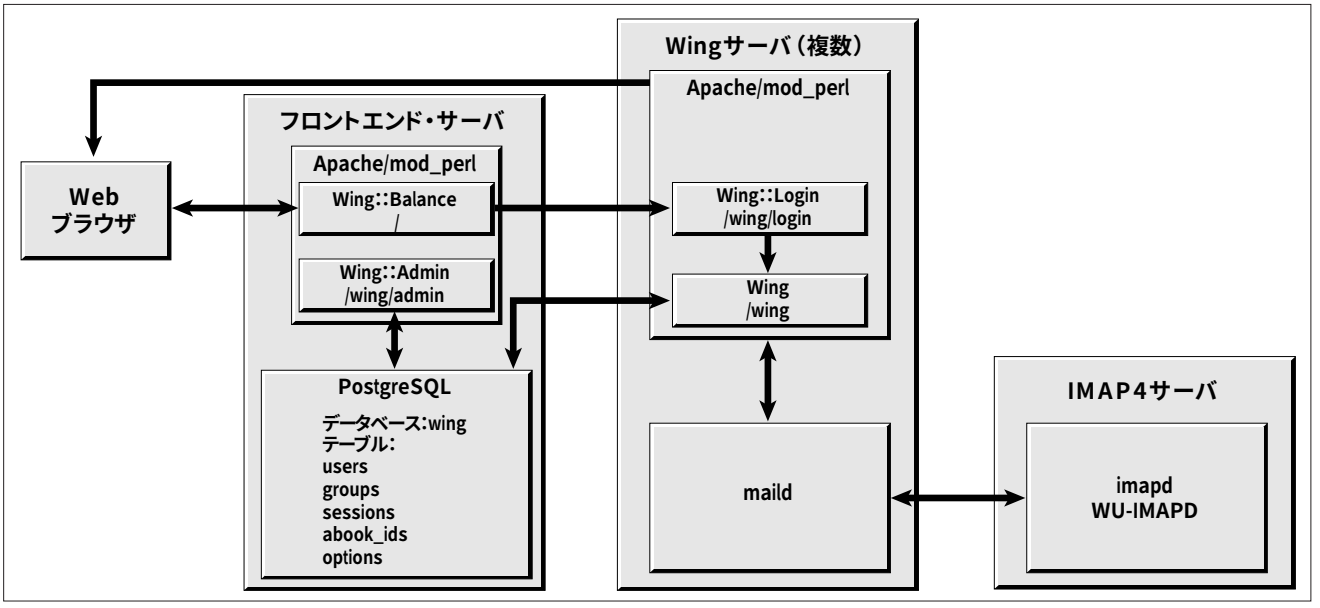


図1　Wingのシステム構成図

①Wingのインストール

Wingは以下のFTPサイトからダウンロードできる。

```
ftp://ftp.ox.ac.uk/pub/perl/Wing-0.8.tar.gz
```

アーカイブを展開すると、ディレクトリWing-0.8が作成され、以下のファイルが含まれている。

- **README、その他**

READMEではWingのインストール方法を簡単に説明してある。詳細なマニュアルはないが、Perlのプログラムには細かくコメントがつけられている。

- **Perlモジュール**

Wing.pmとOutline.pmがあり、さらにディレクトリWingに7つのpmファイルがある。これらはPerlがpmファイルを検索するパスのひとつであるディレクトリsite_perlにインストールする。

- **Perlスクリプト**

IMAP4サーバへのアクセスを仲介するmaild本体の他に、その起動スクリプトwing.initとテストプログラムがある。

- **HTML関連**

ディレクトリicons以下に、Wingがアイコンとして使用する画像ファイルがある。またディレクトリhelp以下に、同じくWingがヘルプ表示に使用するHTMLファイルがある。

- **sqlファイル**

PostgreSQLに作成するwingデータベースのテーブル、インデックスなどのSQLが記述されている。

Perlモジュールの中で、サーバの環境に合わせた変更が必要なものは、Wing/Shared.pmにまとめてある。HTML表示に関連する各種定数はとりあえずデフォルトのものを使うとして、最低限、以下の項目について変更が必要だ。

```
$WING_DOMAIN
```

デフォルトはwing.example.orgなので、これを実際にWingサーバが稼働するドメイン名に変更する。単体マシンでテストするときは、localhostにする。

```
sub initial_mailbox
```

このサブルーチンは、IMAPサーバのアドレスとそのメールボックスを含むアドレス指定を返す。デフォルトでは、ユーザー名が個々のマシン名に対応するクラスタ環境を想定しているので、利用するシステム環境に合わせて調整する必要がある。

たとえば、WingサーバとIMAPサーバを同一マシンで動かすのであれば、次のようにする。

```
return "{$WING_DOMAIN/imap}INBOX";
```

Wing/Shared.pmの変更ができたら、Perlモジュールをsite_perlにコピーする。

```
# cp Wing.pm Outline.pm /usr/local/lib/site_perl
# mkdir /usr/local/lib/site_perl/Wing
# cp Wing/*.pm /usr/local/lib/site_perl/Wing
```

HTML関連ファイルは、Apacheのドキュメントルートにコピーする。

```
# mkdir /var/www/wing-icons /var/www/wing-help
# cp icons/* /var/www/wing-icons
# cp help/* /var/www/wing-help
```

maild関連のファイルについては⑤で述べる。今回のインストールについては、Debian-JP 2.1で行った。これ以外のディストリビューションの場合、インストール先のディレクトリが異なると思われる。適宜変更していただきたい。

②PostgreSQLのデータベース設定

次にフロントエンドマシン上のPostgreSQLにWingデータベースを作成する。PostgreSQL本体のインストールについては、ここでは割愛させていただく。

Wingデータベースの作成には、wing-0.8ディレクトリにあるsqlファイルを実行する。Wingでは、Apacheの実効ユーザーがhttpdであることを前提にしている。したがって、httpd以外の実効ユーザー（たとえばwww-dataなど）になっている場合、これをhttpdに変更するか、Wingのプログラムを変更する必要がある。

ここでは、“usermod -l httpd www-data”としてLinuxのユーザーアカウント名を変更することにした。このため、後述するようにApacheのhttpd.confにおける指定も変更する必要がある。

次にPostgreSQLにWingデータベースを新規作成する。まず、PostgreSQLでユーザー登録を行う。

```
$ su
# su postgres
$ createuser root
```

```
$ createuser httpd
```

次にwingデータベースを作成する。

```
$ createdb wing
```

ユーザーrootでusers-init.sqlを実行する。

```
$ su
# psql -d wing -f users-init.sql
```

続いてユーザーhttpdでwing-init.sql、abook-init.sqlを実行する。

```
$ su
# su httpd
$ psql -d wing -f wing-init.sql
$ psql -d wing -f abook-init.sql
```

③Wing以外のPerlモジュールのインストール

前述したように、Wingのモジュールは多岐にわたるPerlモジュールに依存しているが、標準的なPerlの構成では含まれていないものが少なくない。そのため各種Perlモジュールをインストールする作業が必要になるのだが、これが思ったよりも時間を要してしまう。このような事情を察してか、以下のサイトにWingが依存する全ファイル（Wingも含む）が集められている。

ftp://ftp.ox.ac.uk/pub/linux/RPMS/i386　バイナリ配布(RedHat 5.x)
ftp://ftp.ox.ac.uk/pub/linux/SRPMS/　ソース配布

ソース配布として収録されているrpmファイルは次のとおり。

```
apache-mod_perl-1.2.6-5.src.rpm
exim-2.02-2.src.rpm
imap-4.4blackbox-5.src.rpm
perl-Apache-DBI-0.81-1.src.rpm
perl-CrackLib-0.1-1.src.rpm
perl-DBD-pg-0.73-1.src.rpm
perl-DBI-1.02-1.src.rpm
perl-DBO-0.2-1.src.rpm
perl-Data-Dumper-2.09-1.src.rpm
perl-HTML-Embperl-1.2b1-1.src.rpm
perl-HTML-Parser-2.20-1.src.rpm
perl-IO-stringy-1.203-1.src.rpm
perl-MIME-Base64-2.06-1.src.rpm
perl-MIME-Tools-4.121-src.rpm
perl-Mail-Cclient-0.4-2.src.rpm
                  0.5-1.src.rpm
                  0.6-1.i386.rpm
                  0.6-1.src.rpm
perl-MailTools-1.11-1.src.rpm
perl-Net-DNS-0.12-1.src.rpm
perl-Net-Telnet-3.01-1.src.rpm
perl-SQL-0.2-1.src.rpm
perl-Term-ReadKey-2.12-1.src.rpm
perl-Term-ReadLine-0.9904-1.src.rpm
perl-libnet-1.0605-1.src.rpm
perl-libwww-5.36-1.src.rpm
wing-0.6-1.src.rpm
wing-0.8-1.src.rpm
```

ファイル名がperlで始まっているファイルはPerlモジュールだが、CPANに収録されていないモジュールが2つある。CrackLibとSQLがそれで、Wingの作者Malcolm Beattieによるものだが、SQLにはバグが含まれていることを注意している。

今回はDebian-JP 2.1でインストールするため、以下の手順をとった。

・Debian-JP 2.1に収録されているPerlモジュールのパッケージをインストール
・それ以外のPerlモジュールはCPANからインストール
・CrackLibとSQLは、debファイル（Debianのパッケージ形式）に変換して、インストール

そのほか、Mail::Cclientをインストールした。このPerlモジュールは、WU-imapdに対する一種のAPIを提供するものであり、WU-imapdに含まれるライブラリを使用する。そこでCPANからMail-Cclient-0.5.tar.gzを入手し展開した後、以下の手順でmakeを行った。

```
$ perl Makefile.PL CCLIENT_DIR=/path/to/c-client
```

ここで、/path/to/c-clientには、WU-imapdをmakeしたときに作成されるc-clientディレクトリを指定する。このディレクトリに作成されたc-client.aからオブジェクト形式のファイルを抽出している。Makefileが作成されたら、makeを実行する。

```
$ make
```

```
# make install
```

通常のPerlモジュールでは、make installの前にmake testを実行するのだが、このモジュールで提供されているテストは、あくまでプログラミングのサンプルとしてのものであり、実際に実行すると終了しなくなるので注意が必要だ。

ちなみにrpmファイルのdebファイルへの変換には、alienコマンドを使用した。

```
$ alien perl-CrackLib-0.1-1.src.rpm
$ alien perl-SQL-0.2-1.src.rpm
# dpkg -i perl-cracklib_0.1-2_i386.deb
# dpkg -i perl-sql_0.2-2_i386.deb
```

すると、ルートディレクトリに次のファイルが展開される。

```
perl-CrackLib.spec
CrackLib-0.1.tar.gz
perl-SQL.spqc
SQL-0.2.tar.gz
```

これを適当なディレクトリに移動させ、展開して、make、インストールする。

```
$ perl Makefile.PL; make; make test
# make install
```

④Apache/mod_perlの設定

Debianでは、mod_perlはdselectを使ってパッケージlibapache-mod-perlをインストールする。インストール後、/usr/lib/apache/1.3/にmod_perl.soがあるのを確認する。

次に、apacheの設定ファイルhttpd.conf（Debianでは/etc/apache/httpd.conf）にある一連のLoadModuleをみて、perl_moduleの指定がコメントになっていたら、これを外して次のようにする。

```
LoadModule perl_module /usr/lib/apache/1.3/mod_perl
.so
```

余談だが、Apacheのモジュールにmod_imapというのがある。しかし、これはIMAP4ではなく、クリッカブルマップのためのモジュールである。

ついでに、httpd.confのuserとgroupの指定を、www-dataからhttpdに変更しておく。httpd.confの変更が終わったら、次のコマンドを実行してApacheを再構築する。

```
# apacheconfig
```

次に、access.confを編集する。

フロントエンドとWingサーバを同一マシンにする場合、このファイルの末尾に、以下の指定を追加する。

```
Alias /wing-icons/ /var/www/wing-icons/
PerlModule Wing
PerlModule Wing::Balance
<Location />
SetHandler perl-script
PerlHandler Wing::Balance
</Location>
<Location /wing>
SetHandler perl-script
PerlHandler Wing
</Location>
<Location /wing/login>
SetHandler perl-script
PerlHandler Wing::Login
</Location>
<Location /wing/admin>
SetHandler perl-script
PerlHandler Wing::Admin
</Location>
```

このAlias指定によって、/wing-icons/はWingの画像ファイルのあるディレクトリを参照する。PerlModule指定は、Apacheの起動時にここで指定されたPerlモジュールをロードする。頻繁に使われるPerlモジュールをこのようにあらかじめロードしておくと、オーバーヘッドが少なくなる。Locationの指定については、順番を変えてしまうと対応がうまくいかなくなるので、注意すること。

フロントエンドとWingサーバを分離する場合は、access.confの追加部分はそれぞれ以下のようにする。

フロントエンド側：

```
PerlModule Wing::Balance
<Location />
SetHandler perl-script
PerlHandler Wing::Balance
</Location>
<Location /wing/admin>
SetHandler perl-script
```

```
PerlHandler Wing::Admin
</Location>
```

Wingサーバ側：

```
Alias /wing-icons/ /var/www/wing-icons/
PerlModule Wing
<Location /wing>
SetHandler perl-script
PerlHandler Wing
</Location>
<Location /wing/login>
SetHandler perl-script
PerlHandler Wing::Login
</Location>
```

⑤maildの設定

Wingサーバに相当するマシンにmaildをインストールする。

```
# install maild /usr/sbin
```

maildがセッションの記録に使用するディレクトリを作成する。

```
# mkdir /var/lib/maild /var/lib/maild/sessons
# chown httpd /var/lib/maild /var/lib/maild/sessions
# chmod 700 /var/lib/maild /var/lib/maild/session
```

/var/lib/maildには、maildのログファイルmaild.log、プロセスIDを記録するmaild.pidが作成される。maildを起動／停止させるためのスクリプトがwing.initである。これを見ると、su httpdを使ってmaildを起動しているのがわかる。次のコマンドを実行して、このwing.initをmaildの起動スクリプトにする（スクリプトの場所は/etc/rc.d/init.dなどディストリビューションによって異なるので注意すること）。

```
# cp wing.init /etc/init.d/maild
# ln -s /etc/init.d/maild /etc/rc2.d/S20maild
# ln -s /etc/init.d/maild /etc/rc3.d/S20maild
# ln -s /etc/init.d/maild /etc/rc4.d/S20maild
# ln -s /etc/init.d/maild /etc/rc5.d/S20maild
```

最後にフロントエンドのマシンに/etc/wing.liveを作成する。このファイルには、Wingサーバのホスト名を指定する。この内容を見て、フロントエンドからWingサーバに処理がリダイレクトされる。Wingサーバを複数配置する場合は、１行につき１ホスト名にしたものを複数行指定する。先頭に#があれば、コメント行として解釈される。

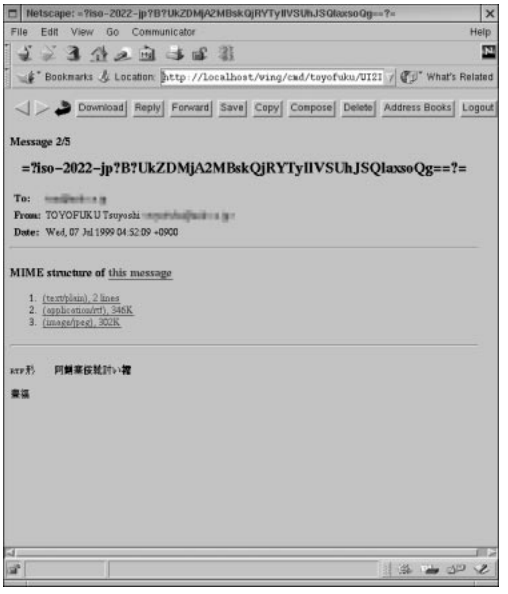
⑥テスト

フロントエンドとWingサーバの両方で、それぞれApacheを再起動する。またWingサーバではmaildを再起動する。

```
# apachectl restart
# /etc/init.d/maild restart
```

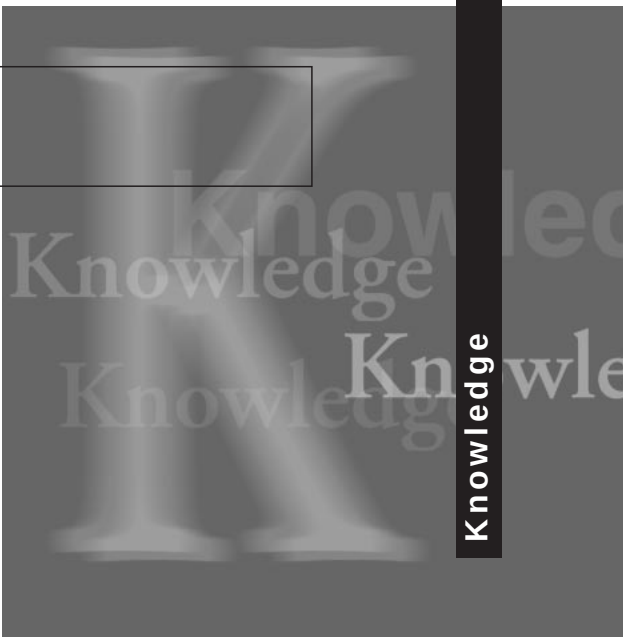
必要なPerlモジュールが欠落している場合、Apacheの起動時やWingの実行時にエラーが発生する。Apacheのログ（/var/log/apache/error.log）とmaildのログ（/var/lib/maild/maild.log）、WU-imapdのログ（/var/log/syslog）を見て、原因をつきとめるとよいだろう。

問題なければ、フロントエンドサーバにWebブラウザでアクセスしてみよう。添付された画像ファイルへのリンクをクリックすると、画像が画面表示される。本文が日本語のメールを表示すると、文字化けが発生する場合もあったが、設計そのものはかなりモジュール化されているので、見通しは悪くない。また、Wingをベースにすれば、従来のWebメールや掲示板システムとは、一味違ったWebアプリケーションが構築できるだろう。



画面3 Wingによる Web画面

ext2ファイルシステム



前回は、Ext2ファイルシステムの構造について見てきた。今回は、Ext2ファイルシステムの作成や各種パラメータの調整を行うユーティリティを紹介しよう。

ユーティリティ編

文：小松克行
Text:Ktsuyuki Komatsu

Ext2fs Home Page
http://web.mit.edu/tytso/www/linux/Ext2.html
E2fsprogs: Ext2 Filesystem Utilities
http://web.mit.edu/tytso/www/linux/e2fsprogs.html

Ext2ファイルシステムのユーティリティとしては、次のコマンドが用意されている。ここではそれぞれのコマンドの解説とちょっとしたポイントを紹介したいと思う。

chattr ファイルの属性を設定する
lsattr ファイルの属性を表示する
e2fsck ファイルシステムのチェックと修復を行う
e2label ファイルシステムのボリュームラベルを設定する
mke2fs ファイルシステムを作成する
tune2fs ファイルシステムの設定を変更する
dumpe2fs ファイルシステムの各種パラメータを表示する
debugfs ファイルシステムのデバッグを行う

chattr

chattrは、ファイルの属性を変更するツールである。Ext2ファイルシステムでは、ファイル単位で属性を設定できるようになっている。設定できる属性は、以下のとおり。

```
usage: chattr [-RV] [-+=AacdisSu] [-v version] files...
```

A アクセスタイムを更新しない
S 同期更新モード
a 追加書き込みのみ許可（設定変更はスーパーユーザーのみ可能）
c ファイル圧縮
i 変更不可（設定変更はスーパーユーザーのみ可能）
d dumpプログラムでのバックアップ候補から除外
s セキュア削除（使用ブロックをランダムバイトで埋めるなどしてからファイルを削除する）
u アンデリート可能

これらのうち、ファイル圧縮とアンデリートは、属性を設定できるものの、機能としては現時点では実装されていないので意味をなさない。ちなみに現在のExt2ファイルシステムのバージョンは“0.5a”である。

lsattr

ファイルの属性一覧を表示する。オプションなどは以下のとおり。

```
usage: lsattr [ -RVadv ] [ files...  ]
```

-R リカーシブモード
-V プログラムバージョンを表示
-a “.”（ドット）から始まるファイルを含めディレクトリ内のすべてのファイルの属性情報を表示
-d ディレクトリの属性を表示
-v ファイルのバージョンを表示

実行例：

```
[user@linux01 test]$ chattr +AacdisSu -V license
chattr 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Flags of license set as sucSiadA
chattr: Operation not permitted while setting flags
on license
```

（※aとi属性はスーパーユーザーのみ設定可能なので、コマンドは失敗している）

```
[user@linux01 test]$ chattr +AcdsSu -V license
chattr 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Flags of license set as sucS--dA
[user@linux01 test]$ lsattr
sucS--dA ./license
-------- ./jpnfont.tar.gz
-------- ./ILINXR.install
[user@linux01 test]$
[root@linux01 /root]# chattr +AacdisSu -V test.log
chattr 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Flags of test.log set as sucSiadA
```

（※スーパーユーザーでは、すべての属性が設定可能だ）

```
[root@linux01 /root]# lsattr
-------- ./nsmail
sucSiadA ./test.log
[root@linux01 /root]#
```

e2fsck

e2fsck（fsck.ext2）は、Ext2ファイルシステムの整合性チェックおよび修復を行うプログラムだ。

ファイルシステムの整合性のチェックは、ブート時のファイルシステムをマウントする際にクリーンフラグをチェックして自動的に実行されるように設定されているので、手動で実行することは少ない。/etc/fstabの6フィールド目が自動的にチェックするかどうかの設定だ（1は有効、0は無効）。またファイルシステムのクリーンフラグだけでなくマウント回数、経過時間などによってもチェックが有効になる。これに関しては後述するtune2fsコマンドなどを参照してほしい。

Usage：

```
e2fsck [-panyrcdfvstFSV] [-b superblock]
       [-B blocksize] [-I inode_buffer_blocks]
       [-P process_inode_size]
       [-l|-L bad_blocks_file] device
```

-a （-pオプションと同じだが互換性のために残っている）

-b superblock 通常のsuperblockの代わりに指定したsuperblockを使用する

-B blocksize ブロックサイズをsuperblockからではなくオプションで指定する

-c 不良ブロックのチェックを行ってi-nodeに登録する

-C fd fdで指定した番号のファイルハンドルに終了状況を書き込む。このオプションは、fsckを起動するプログラムで使用する

-d デバッグ情報を出力する

-f ファイルシステムの状態がfsckが必要な状態かどうかに関わらずチェックを実施する

-F 起動時にファイルシステムのデバイスのキャッシュをクリアする（ベンチマーク用）

-l filename ファイルで指定した不良ブロックのリストをファイルシステムに追加する

-L filename ファイルで指定した不良ブロックのリストをファイルシステムに設定する

-n ファイルシステムを読み取り専用で開き、すべての質問に「n」が入力されたものとみなしてプログラムを実行する

-p 修復に関する質問を行わずに自動で修復を行う

-r 互換性のために残されており、何もしない

-s ファイルシステムのバイトオーダーを標準のバイトオーダー（little endian）に変更する

-S ファイルシステムのバイトオーダーを現在のバイトオーダーに関わらず変更する

-t 処理時間の統計情報を出力する。オプションを2回指定するとより詳細な統計情報を出力する

-v プログラム動作の詳細出力を行う

-V プログラムのバージョンを表示し終了する

-y すべての質問に「y」が入力されたものとみなしてプログラムを継続する

なお、マウントした状態のファイルシステムは、チェックの最中に書き込み、変更が行われる場合があるので、以下のようなワーニングメッセージが表示される。一般にアンマウントしてからチェックするのが望ましい。

```
# fsck /dev/hda1
Parallelizing fsck version 1.12 (9-Jul-98)
e2fsck 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
/dev/hda1 is mounted.
```

```
WARNING!!!  Running e2fsck on a mounted filesystem
may cause
SEVERE filesystem damage.
Do you really want to continue (y/n)? no
check aborted.
```

e2label

e2labelは、Ext2ファイルシステムのボリュームラベルの表示と設定を行うコマンドである。ボリュームラベルの表示と設定は、後述するtune2fsなどでも行える。

```
# e2label
Usage: e2label device [newlabel]
# e2label /dev/hda1 "ROOT"
# e2label /dev/hda1
ROOT
#
```

mke2fs

mke2fs（mkfs.ext2）は、Ext2ファイルシステムを作成するプログラムである。

オプションを指定しないで実行するとブロックサイズ1024バイトでファイルシステムが作成される。オプションとしてよく使用されるものは、ブロックサイズを指定する-bとグループを何ブロックで構成するかを指定する-gだ。

```
Usage：
mke2fs [-c|-t|-l filename] [-b block-size]
       [-f fragment-size] [-i bytes-per-inode]
       [-N number-of-inodes]
       [-m reserved-blocks-percentage]
       [-o creator-os] [-g blocks-per-group]
       [-L volume-label] [-M last-mounted-directory]
       [-r fs-revision] [-R raid_opts]
       [-s sparse-super-flag] [-qvSV]
       device [blocks-count]
```

-b block-size　ブロックサイズをバイト単位で指定する。1024、2048、4096のいずれかが指定できる。デフォルトは1024バイト

-c　ファイルシステムを作成する前に不良ブロックのテストを行う。テストは読み出しのみ

-f fragment-size　フラグメントのサイズをバイト単位で指定する。1024、2048、4096のいずれかが指定できる。ただしブロックサイズ以下でなければならない

-i bytes-per-inode　ディスク容量に対するi-nodeの割合を、ディスクの何バイトに対して1つのi-nodeを作成するかで指定する。デフォルトは4096 [バイト/i-node]。1024以上でなければならない。これによりi-nodeの個数が決定される

-N number-of-inodes　ブロック数とbytes-per-inodeから計算されるi-node数をユーザーが直接指定する

-l filename　指定したファイルから不良ブロックのリストを読み込む

-m reserved-blocks-percentage　予約ブロックの割合をパーセンテージで指定する。デフォルトは5%

-g blocks-per-group　グループを何ブロックで構成するかを指定する。デフォルトはビットマップの構成上、1ブロックサイズ×8となる。つまりブロックサイズ1024バイトの場合、デフォルトは8192。これ以上は指定できない。これ以下なら指定できる

-o creator-os　作成OS名を指定する

-q　メッセージの出力を抑止する

-s sparse-super-flag　このフラグに1を指定すると、sparse superblockフラグをオンにする。0を指定するとsparse superフラグをオフにする（デフォルト）。sparse superフラグはカーネルが正しくサポートしていないので通常は指定しない

-v　動作の詳細を出力する

-F　指定したデバイスがブロックデバイス以外でも強制的にmke2fsを実行する

-L volume-label　ファイルシステムに付けるボリュームラベルを指定する

-M last-mounted-directory　最終マウントディレクトリを指定する

-r revision　ファイルシステムのリビジョンを設定する

-R raid_options　RAID関係のオプションを設定する

-S　superblockとgroupデスクリプタのみを書き込む

-V　バージョン番号の表示する

tune2fs

tune2fsは、Ext2ファイルシステムのパラメータを調整するツールである。Ext2ファイルシステムのパラメータは、スーパーブロックに書き込まれており、tune2fsはその内容を変更する。

tune2fsのスイッチには以下のものがある。

-c max-mount-counts　ファイルシステムに障害がなくてもfsckをかけるマウント回数を指定する
-e errors-behavior　エラーを検出した場合にカーネルが行う処理を指定する。errors-behaviorには、以下の種類がある

continue　処理を継続する
remount-ro　書き込み禁止でファイルシステムをマウントし直す
panic　panicを引き起こしてOSを停止する

-g group　予約領域を使用できるグループを指定する。グループは数値（グループID）かグループ名で指定する
-i interval-between-checks[d|m|w]　ファイルシステムに障害がなくてもfsckをかける時間間隔を指定する。0を指定すると時間間隔をfsckのタイミングとして使用しない。[d|m|w]はそれぞれ日／月／週を表し、3dなら3日、6mなら6カ月、9wなら9週間となり、指定しなければ日の指定とみなされる。デフォルトでは6mつまり6カ月である
-l　superblockの内容を表示する
-m reserved-blocks-percentage　予約領域の割合をパーセントで指定する
-r reserved-blocks-count　予約領域の大きさをブロック数で指定する
-s sparse-super-flag　sparse superblockフラグを設定する
-u user　予約領域を使用できるユーザーを指定する
-C mount-count　マウント回数を実際の値を無視して設定する
-L volume-name　ボリューム名を設定する（最大16文字）
-M last-mounted-directory　最後にマウントされたディレクトリ名を設定する。通常使用しない
-U UUID　UUIDを設定する

ファイルシステムのパラメータを変更する前には、-lスイッチを指定して現在の状態を必ず確認しておこう。また“tune2fs -l -c 30 /dev/hda1”のように指定すると現在の値と変更した値の両方を確認できる。

fsckのタイミングやディスクの予約領域の大きさは、よく調整される項目だ。

まずfsckのタイミングの制御は-cおよび-iスイッチで行う。マウントした回数が-cスイッチで指定した数に達した場合、ファイルシステムの状態がクリーンであってもfsckが実行される。また、以前のfsckから-iスイッチで指定した時間が経過した場合も同様にクリーンであるかどうかに関わらずfsckが実行されるようになる。シャットダウンをたまにしか行わないサーバでは短く、逆に煩雑に行うマシンでは長くしておくのがよいだろう。

ディスクの予約領域の制御は-mまたは-rスイッチで行う。予約領域は、ユーザーから見たファイルシステムの上限を設定する。ユーザーから見てディスクの空きがなくなった状態でも、スーパーユーザーなら予約領域を使用できるためファイルを消さなくても作業をすることができる。予約領域のデフォルトはディスクの5%である。5%という値は、ディスクのフラグメンテーションを避けるためにも使われる。しかし、フロッピーディスクでは予約領域を設定する意味はない。

なおスーパーブロックを更新するような操作をした場合、書き込み処理にそれなりの時間がかかるので注意すること。

dumpe2fs

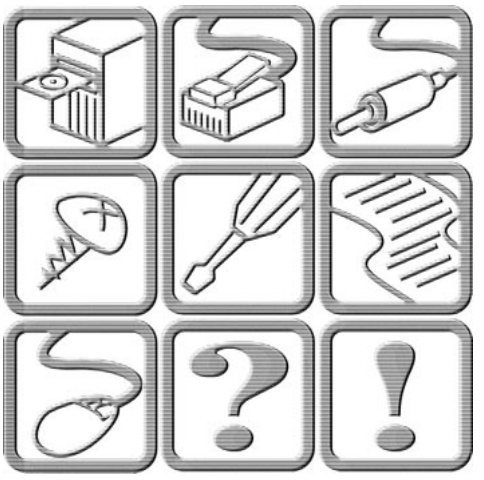
dumpe2fsは、Ext2ファイルシステムのパラメータ（superblockの内容）と各groupの割り当て状況を表示するコマンドだ。-bスイッチで不良ブロックを表示することもできる。ファイルシステムのパラメータの表示はtune2fsと同じである。

debugfs

debugfsはファイルシステムの内部構造操作用のツール（デバッガ）である。debugfsでは、ファイルシステムのディスク上のデータ構造を直接変更できるので、fsckで自動的に修復できないような場合でもファイルシステムの内容を変更することが可能だ。つまり、削除したファイルの復活にも利用できるということである。もちろん、ファイルシステムのディスク上のデータ構造を操作するのだから、Ext2ファイルシステムの知識が不可欠であり、操作に失敗すればディスクを修復不可能にまで破壊することができてしまう。注意して使用してほしい。ヘルプは?を入力すると表示される。

なおデフォルトは、リードオンリーモードでオープンされるので、書き込みも行いたい場合は、-wオプションをつける必要がある。

Try & [trái]
[trái] Try



Linuxカーネルと ダイナミックローダブルモジュール

文:政久忠由
Text:Tadayoshi Masahisa

カーネルのダイナミックなローダブルカーネルモジュールのサポート、つまりシステムの実行中に動的にデバイスドライバを読み込み、その機能を使用できるようにする（また取り外す）しくみは、一見、モノリシックカーネルのLinuxカーネルには似つかわしくない感じがする。しかし、それは偏見だ。この機能自体は、モノリシックであるかマイクロカーネルであるかは関係ない。

2.0以降のLinuxカーネルは、カーネルモジュールの依存関係を考慮したオンデマンド（必要に応じた）ダイナミックローディングをサポートしている。このとき、ユーザーは処理に伴うデバイスドライバのロードをまったく意識する必要はない。

たとえば、あるディスク（パーティション）をマウントする場合（ここではSCSI接続のディスク）、単に“mount /dev/sda1 /mnt/1”とするだけで、必要となるSCSIディスクの処理を行うドライバ、SCSIのジェネリックな処理を行うドライバ、そしてそれぞれのSCSI機器固有の下位ドライバがロードされる。さらに、適切なファイルシステムモジュールもロードされるのである。

ただし、これらが円滑に処理されるには、適切な設定がなされていることが必要十分条件である。特にハードウェアに直接依存したデバイスドライバの場合、どれが必要なのか設定されていないと処理のしようがない（でも、これは今のところの話であって、今後は、プラグアンドプレイがサポートされて完全自動ってことになるだろう。ISAデバイスはともかく、PCIデバイスに関しては）。

ローダブルモジュールをサポートするかどうかは、カーネルの構築時に決定される。現在、ほとんどのディストリビューションでは、ローダブルモジュールをサポートしたカーネルが採用され、ひとつのカーネルイメージでさまざまなユーザー環境に適応できるようになっている。ローダブルモジュールが本格的にサポートされる以前は、ハードディスクのインターフェイスやネットワークアダプタに応じた、数十種類のカーネルが別途用意されていたことを覚えている読者も多いと思う。今では、カーネルイメージとしては1種類（SMPの場合は別）だけを用意し、各デバイスドライバは、インストール時にチェックして、/etc/conf.modulesに設定するようになっている。

カーネル2.2系でローダブルモジュールとして設定できるのは、ファイルシステムから、SCSI、IDE、ネットワーク、サウンドなどの各デバイスドライバ、各バイナリフォーマット用のローダモジュールなど多岐にわたっている。またカーネル2.2系では、従来ローダブルモジュールのオンデマンドロードを処理していたkernelという外部プログラム（ユーザーレベルデーモンとして動作していた）は必要なくなり、代わりにkmodというカーネルモードで動作するプログラムで処理されるようになった。カーネル2.0から2.2にアップデートする場合は注意してほしい。

ローダブルモジュールのメリット

ローダブルモジュールをサポートするメリットは、先ほど述べたような、それぞれの環境で必要となるネットワークアダプタなどのデバイスの違いをカーネルの再構築なし

で解決できる点、そしてあまり使用されない機能をロードしないでおけるためカーネルが消費するメモリを軽減できる点などである。そしてモジュールは、リブートなしにロード、アンロードできるので、テストも非常に容易だ。

デメリットはというと、機能が要求されてから、最初にロードされ、利用できるようになるまでに多少時間がかかる（といっても数ミリから数秒程度、SCSIデバイスは初期化に結構時間がかかる）ことぐらいだ。一般的な利用においては、まったく問題にならないレベルである。あとは、スタティックなモジュールと何らパフォーマンス上の差はない。

ローダブルモジュールを利用するには

ローダブルモジュールを利用するには、ローダブルモジュールをサポートするカーネルと後述するモジュールユーティリティが必要となる。最近のディストリビューションでは、何の問題もないと思う。ここでは、カーネルを再構築する際のポイントだけ紹介しておこう。カーネルのコンフィギュレーションでは、次の項目を有効にする（カーネルは2.2を前提）。

```
Loadable module support：
[＊] Enable loadable module support
[＊] Kernel module loader
```

あとは、ブートして、ルートファイルシステムがマウントできて、initが実行できるのに最低限必要なドライバだけスタティックに組み込み、それ以外はモジュール＜M＞として設定するだけだ。実は、LILOなどのブートローダのサポートで、最低限必要なドライバさえもローダブルモジュールにしてしまうことができるのだが、とりあえずは、あまり無理をしないで設定しよう。どうしてもという人は、後述の「モジュールをプリロードする」の説明が参照になるだろう。

カーネルのコンフィギュレーションが終わったら、次のようにしてカーネル、各ローダブルモジュールをコンパイル、そしてインストールする。

```
# make bzImage modules modules_install
```

モジュールは、/lib/modules/＜カーネルバージョン＞内のblock、fs、netなどタイプ別のディレクトリにインストールされる。上記のコマンドでは、モジュールのインストールまで行っている。けれども、カーネルのインストールはユーザー環境に依存するので、あえて行っていない。一応、次のコマンドでカーネルをインストールすることができる（環境もある）。これは適宜環境に合わせて行ってほしい。

```
# make install
```

また、実際にモジュールを利用するには、以下のコマンドを実行してモジュールの依存関係をチェックしておく必要がある。通常は、/etc/rc.d/rc.sysinitなどの初期化スクリプトで実行されるようになっているのでユーザーがいちいち実行する必要はないのだが、知っておこう。

```
# /sbin/depmod -a ＜バージョン＞
```

ローダブルモジュールって？

ところでローダブルモジュールとしてコンパイルしたオブジェクトと、カーネルにスタティックに組み込むモジュールとしてコンパイルしたオブジェクトとでは、何が違うのだろうか？

ローダブルモジュールの場合、コンパイル時にMODULEを定義することでinit_module()とcleanup_module()を有効にしている。これらは、それぞれの名前から内容は想像できると思うが、カーネルがモジュールの初期化と登録抹消を行う時に実行する。スタティックなオブジェクトでは、これらの代わりに通常の初期化ルーチン（init_foobarとか）が有効になり、カーネルは起動時にそれを呼んでその機能を使用できるようにしている。違いとしてはこれだけで、init_module()と通常の初期化ルーチンは基本的には同じ内容である（違う必要はどこにもない）。異なるのは、カーネルがどのタイミングでどのように初期化しているかと、登録を抹消する機能があるかどうかだけである。

依存関係のあるモジュールのいくつかでは、request_module()を使用して、明示的に必要なモジュールを要求している。たとえば、pppでは、slhcを必ず先にロードする必要がある。これは、pppがslhcからエクスポートされている機能を使用しているためだ。

一方カーネルでは、モジュールを生成、初期化、削除、クエリーなどを行う機能が有効になっている（詳細は/usr/src/linux/kernel/module.cを参照）。request_module()はkmodが提供する機能だ。

ちなみにモジュールファイルは、foo.oなどのように拡張子に“ o ”がついているが、ユーティリティで暗黙のうちに処理されているので、ユーザーは単にfooとして扱うことができる。

モジュールユーティリティ

Linuxのローダブルモジュールのサポートでは、それを操作するユーティリティが不可欠である。ここでは、そのユーティリティを紹介しておこう。これらのコマンドは通常、/sbinに格納されている。

insmod	**ローダブルモジュールのインストール**
rmmod	**モジュールの登録削除**
ksyms	**シンボルの表示**
lsmod	**読み込まれているモジュール一覧の表示**
modinfo	**モジュール情報の表示**
modprobe	**設定ファイルに基づくモジュールの読み込み**
depmod	**modprobe用のモジュールの依存関係の生成**
genksyms	**モジュールシンボルのバージョン生成用プログラム**

■insmod

insmodコマンドは、引数として渡されたモジュールを実際にインストールするコマンドだ。ローダブルモジュールは、自動的に導入されるにしろ、手動で導入するにしろ、最終的には、このコマンドが使用される。このコマンド自身は、システムコールのsys_init_moduleを呼んで実際の登録を行っている。通常はinsmodよりも、後述のmodprobeコマンドを使用するほうがスマートだ。

コマンドのオプションは次のとおり。

```
Usage：
insmod [-fkmopsvVxX] [-o name] module [[sym=value].]
```

module　　モジュールのファイル名（*.o）
-f, --force　　強制的にロードする
-k, --autoclean　オートクリーンを有効にする
-m　　トレース用のロードマップを生成する
-o NAME
　--name=NAME 内部的なモジュール名
-p, --poll　　Pollモード、バージョンチェックを行う
-s, --syslog　　syslogにエラーをレポート
-v, --verbose　詳細を表示
-V, --version　バージョンを表示
-x　　エクスポートしない
-X　　エクスポートする（デフォルト）

■rmmod

rmmodコマンドは、ローダブルモジュールをカーネルからアンロードするコマンドだ。引数としてモジュール名を指定して実行するだけだが、実際に使用されているモジュールをアンロードすることはできない。現在使用されているかどうかは、後述のlsmodコマンドで確認できる。なお、-aオプションを指定すると使用されていないモジュールすべてをアンロードできるようになっている。

```
Usage： rmmod [-a] [-s] module ...
```

-a　使用されていないモジュールすべてを取り外す
-s　syslogに詳細をレポート

■ksyms

ksymsコマンドは、カーネルにロードされたローダブルモジュールのエクスポートしているシンボルを表示するコマンドだ。実際に各プロセスにマッピングされる際のアドレスとシンボル名、そしてモジュール名をリスティングすることができる。

オプションを何もつけないとローダブルモジュールのエクスポートシンボル情報だけが表示される。-aオプションをつけるとローダブルモジュール以外、最初からカーネルに組み込まれたモジュールのエクスポートシンボル情報も表示される。また、-mオプションをつけると各モジュールのロードアドレス、サイズなどの情報も表示されるようになる。これにより、エクスポートシンボル情報を持たないモジュール、ntfs.oなどの状況もわかる。

```
Usage： ksyms [-a] [-h] [-m]
```

-a　すべてのシンボルを表示
-h　カラムのヘッダーを表示しない
-m　モジュール情報を表示（ロードアドレス、サイズなど）

■lsmod

lsmodコマンドは、ロードしているモジュールの情報を表示する。システムコールquery_moduleを利用している。表示される情報は、モジュール名とサイズ、そして使用カウンタなどだ。オプションはない。

■modinfo

modinfoコマンドは、各モジュールオブジェクトの情報を表示する。

```
usage： modinfo <parameters> <module>
```

-a, --author	モジュールの作者を表示
-d, --description	モジュールの説明を表示
-f <str>	
--format <str>	任意の情報を表示
-p, --parameters	モジュールパラメータの表示
-V, --version	バージョン表示
-h, --help	ヘルプ

■modprobe

modprobeコマンドは、/etc/conf.modulesや後述するdepmodコマンドで生成された情報に基づき、柔軟なモジュールの管理を行うコマンドだ。内部的には、insmodコマンドを実行している。

このコマンドは、-cオプションで確認できるデフォルトの設定情報をコマンド自体に含んでいる。実行すると/etc/conf.modulesに設定した情報に加えて、物理的なデバイス機器に依存しない基本的なデバイス番号と対応するモジュール名などの設定が表示されるはずだ。つまりこのコマンドを使用する限り、ユーザーがいちいちリクエスト内容とモジュールの対応を設定しておかなくても問題はない。この設定には、モジュールファイルの格納先の設定も含まれている。pathで始まる項目がそれだ。基本は、各バージョンのディレクトリに格納するのだが、/lib/modules/defaultや/lib/modulesにそれぞれのタイプ（fsなど）のディレクトリを作成してファイルを格納しても参照されるようになっている。サードパーティ製のモジュールを利用していて、複数バージョンのカーネルを切り替えて使っているユーザーは参考にしてほしい。

なお、modprobeが把握しているモジュールファイル一覧は、-lオプションで確認できる。

```
Usage： modprobe [-a] [ -t TYPE ] MODULE [opt=val ...] ...
        modprobe -c
```

-a, --all	すべてのモジュールをロード
-c, --show-conf	モジュールの設定状況を表示
-d, --debug	デバッグモードで実行
-k, --kernel-daemon	（kerneldが利用）
-l, --list	モジュールファイルの表示
-r, --remove	カーネルからモジュールをアンロード
-s, --system-log	syslogにエラーをリポート
-t TYPE, -type TYPE	fsやnetなどカテゴリタイプの選択
--help	ヘルプ表示
-v, --verbose	詳細表示
-V, --version	バージョン表示

■depmod

depmodコマンドは、modprobeで利用するモジュールの依存関係情報を生成するコマンドだ。個別にモジュールを指定する方法もあるが、通常は、-aオプションをつけてすべてのモジュールの依存関係をチェックするのが一般的だ。これによりチェックされた依存関係情報は、/lib/modules/＜カーネルバージョン＞/modules.depファイルに保存される。このファイルの内容はプレーンテキストなのでlessなどで見てみるとよいだろう。各モジュール名のあとに、依存するモジュール名が絶対パスで指定されている。複数のモジュールに依存しているものもわかる。前述のmodprobeコマンドでは、これらの情報を元に、必要なモジュールを自動的にロードしている。

```
Usage： depmod [-e -s -v ] -a [FORCED_KERNEL_VER]
        depmod [-e -s -v ] MODULE_1.o MODULE_2.o ...
```

-a, --all	すべてのモジュールの依存関係をチェック
-d, --debug	デバッグモード
-e	未解決シンボルの表示
-i	シンボルバージョンを無視
-m, --system-map <file>	指定したファイルのシンボル情報を使用する
-s, --system-log	syslogにエラーをレポート
--help	ヘルプ表示
-v, --verbose	詳細表示
-V, --version	バージョン情報表示

■genksyms

genksymsコマンドは、モジュールのシンボルバージョン情報を生成するプログラムだ。ユーザーが直接使用することはなく、カーネルのコンフィギュレーションでバージョン情報を生成するように設定した場合に自動的に使用される。

オンデマンドローディング

それでは、オンデマンドローディングの実際の処理の流れを見ていくことにしよう。ローダブルモジュールがどの

ようにオンデマンドにロードされて使用できるようになるのかをまとめると次のようになる。

・まずカーネルは、あるリクエストを処理する際に、それを処理するための機能（モジュール）がカーネル内に登録されていないことに気づく。
・そこでカーネルは、kmodに必要な機能の情報を送る。
・kmodでは、送られてきた情報を元に、その機能を提供するモジュールをロードするためにmodprobeコマンドを実行する。
・modprobeは、/etc/conf.modulesや/lib/modules/<カーネルバージョン>/modules.depを走査してモジュールの実体と依存関係、さらにいくつかのオプション情報を入手し、insmodコマンドを用いて、必要なモジュールをインストールする。
・insmodコマンドによってローダブルモジュールは、カーネルのメモリ空間に登録され、ほかのカーネル内に配置されている機能の呼び出しと同じように使用できるようになる。

kmodがLinuxカーネルから依頼を受け、modprobeを実行するのに使用するリクエストは、次のようなものだ。

```
binfmt-?
block-major-?
char-major-?
eth?
net-pf-?
tty-ldisc-?
iso9660
```

上記のリクエストすべてをユーザーが設定する必要はない。ユーザーは、モジュールの設定を/etc/conf.modulesに記述するわけだが、コマンドの説明で述べたとおり、modprobeコマンド自体にデフォルトの設定情報があらかじめ組み込まれているので、足りない部分を補えばよいのだ。

ユーザーが設定しなければならないのは、下位のドライバである物理的なデバイス機器に依存する部分と、設定されていない特別なデバイスに関する部分だけである。一般に、ネットワークアダプタ、SCSIホストアダプタ、そしてサウンドカードである。これらに関しては、後述サンプル設定を参考にしてほしい。

ちなみに、kmodが実行するmodprobeでは、insmodを呼び出す際に、-kオプションをつけて実行している。つまり、モジュールをオートクリーン属性でインストールしている。そのため、モジュールは通常1分間使用されない状態でいると自動的にアンロードされるようになっている。これにより、カーネルのメモリ使用量を最小限に抑えることができる。カーネルモジュールはスワップアウトの対象ではないので、結構重要だ。

/etc/conf.modulesの設定

では実際に、/etc/conf.modulesの設定を行ってみよう。サンプルとして筆者のテストマシンの設定ファイルを紹介しておく（#以降はコメント）。

```
alias eth0 via-rhine
#alias eth1 smc-ultra
alias scsi_hostadapter aic7xxx
alias net-pf-3 off
alias net-pf-4 off
alias net-pf-5 off
alias net-pf-6 off
alias sound pas2
#alias char-major-14 pas2
#post-install pas2 /sbin/modprobe "sb"
options pas2 io=0x320 irq=5 dma=5
options sb pas2=1
```

1、2行目はネットワークアダプタの設定だ。ここでは、eth0にvia-rhineを指定している。eth1は、2枚目のネットワークアダプタだが、現在使用していないのでコメントアウトしている。

3行目はSCSIホストアダプタの指定だ。aic7xxxを指定している。

4～7行目の設定は、ネットワークプロトコルの設定で、それぞれ、ax25、ipx、appletalkのモジュールを使用しないという設定だ。わざわざこのように設定しなくても、実害はないのだが、ifconfigなどが実行されたときのワーニングメッセージがうざいので設定している。

8行目以降は、サウンド関連の設定だ。デフォルトで、alias char-major-14 soundと設定されているので、そのsoundのエイリアスとしてpas2を指定している。ユーザー設定が優先されるのでalias char-major-14 pas2としてもよい。pas2は、ISAバスのPro Audio Spectrum 16（PAS16）サウンドカードなので、オプションとして、リ

ソースの設定を行っている。最近のPCIカードであれば、IRQ、IOポート、DMAの設定はハードウェアのほうで自動的に行ってくれるので必要ないが、ISAカードには基本的にこのような仕組みはない（PnP ISAとかはあったけど）ので手動で指定する必要がある（11行目）。一応、PAS16には、SoundBluster互換モードがあり、そのドライバも用意されている。その設定が10行目と12行目で、post-installとしてpas2のロード後にmodprobeコマンドでインストールするように指示している。ただし、筆者はsbモードは使用しないので、コメントアウトしている。

post-install以外では、install、pre-install、さらにpost-remove、remove、pre-removeが指定できる。post、preはそれぞれ指定したモジュールがロードされる後、もしくは前に行う処理を指定する。単なるinstall、removeは、前か後の区別をしない時に指定する。

先ほどの設定中のchar-major-14は、カーネルで定義されたデバイス番号を指している。char-major-14だと、キャラクタデバイスのメジャー番号14番ということになる。マイナー番号まで指定したものだと、char-major-10-130などとなる。どの番号が何に割り当てられているかは、/usr/src/linux/Documentation/device.txtに記されているので、参照してもらいたい。ちなみに/dev/のファイル群は、このデバイス番号をマッピングしたスペシャルファイルである（ls -lで確認してみよう）。

モジュールをプリロードする

これまでの設定は、カーネルのイニシャライズ後のモジュールの扱いである。つまり、ローダブルモジュールファイル、モジュールユーティリティ、設定ファイルが格納されているデバイスにアクセスできることが前提だ。

では、そのデバイスにローダブルモジュールでしか、アクセスできない場合はどうしたものだろう。これにはいくつかの解決策がある。ひとつは必要なモジュールをカーネルに最初から組み込んでしまうことだ（要カーネル再構築）。もうひとつは、LILOなどのカーネルローダーのサポートで、RAMディスクを用いて、カーネルの初期化時に必要なモジュールをロードできるようにする方法である。この方法は、最近のディストリビューションではよく利用されている。プリロードするモジュールをRAMディスクイメージとして作成し、カーネルのロード時に同時に読み込み、RAMディスクに展開して、そこからモジュールをロードする。このRAMディスクイメージの作成には、/sbin/mkinitrdコマンドを利用する。このコマンドでは、/etc/conf.modulesのscsi_hostadapterとして設定されているモジュールと、--preload（SCSIモジュールの前にロード）、--with（SCSIモジュールの後にロード）で指定したモジュールをまとめてイメージファイルにできる（これを利用するのは通常SCSIアダプタなので、それが基本となっている）。このイメージはどこに配置してもよいが、通常はカーネル本体と同じディレクトリ（/bootなど）に置き、/etc/lilo.confのinitrd=の指定でそのファイルを指定する。たとえば次のような設定である。

```
image=/boot/vmlinuz-2.2.5-15
        label=linux.orig
        root=/dev/hda1
        initrd=/boot/initrd-2.2.5-15.img
        read-only
```

この機能を用いれば、可能なものすべてをローダブルモジュールにすることもできる。扱いやすいかどうかは別だけど。

ローダブルモジュールの実際

それでは、実際にローダブルモジュールの振る舞いを確かめてみることにしよう。まずは、オプションは指定しないでmodprobeでローダブルモジュールのいくつかをロードしてみた。その状態は次のようなものだ（lsmodで確かめている）。

```
# /sbin/lsmod
Module                  Size  Used by
vfat                   10812   0  (unused)
msdos                   5724   0  (unused)
fat                    31680   0  [vfat msdos]
ntfs                   39104   0  (unused)
ide-cd                 25140   0
cdrom                  14136   0  [ide-cd]
```

使用状況を表す部分では、依存関係のあるモジュール名が表示されている。たとえば、fatの[vfat msdos]という表示は、vfatとmsdosによってfatモジュールの機能が利用されているという意味だ。また（unused）という表記は、現状使用されていないことを表している。実際にWindows 98をインストールしたFATパーティションを

マウントしてみると次のように、使用されたモジュールfatとvfatの使用カウンタはインクリメントされ、vfatに表示されれいた（unused）もなくなる。

```
# /sbin/lsmod
Module                  Size  Used by
vfat                   10812   1
fat                    31680   1  [vfat msdos]
```

では、必要なモジュールを手動でロードしないで、システム任せで行ってみることにしよう。ここではvfatもfatもロードされていない状態で、“ mount /dev/hda3 /mnt/win98 ”を実行した。mountでは、-tオプションでファイルシステムタイプを指定したほうが確実なのだが、vfatなどはそれなりに判定してくれるのであえて指定していない。少なくとも筆者の環境では、vfatとntfsは問題なく判定できている。

```
# /sbin/lsmod
Module            Size    Used by
vfat              10812   1  (autoclean)
fat               31680   1  (autoclean) [vfat]
```

先ほどと異なり、ここでは、(autoclean)という表記があるのがわかる。これはinsmodコマンドの-kオプションつきでモジュールがインストールされたことを表している。つまり、オートクリーン有効である。前でも述べたが、このオートクリーンが有効な場合、1分間使用されない状態が続くと自動的にアンロードされるようになっている。

実は、カーネルから依頼を受けたkmodは、modprobe -s -k ＜モジュール名＞を実行している。modprobeコマンドの説明には、-kオプションはないが、キックするinsmodには渡しているのだ。もちろん、ユーザーが手動でモジュールをロードする場合に、上記のコマンドを実行すれば同様の結果となる。

ここでksymsコマンドでインストールしたモジュールがエクスポートするシンボルを見ることができる。ただし、ほかのモジュールから使われることのないもの、たとえばbsd_comp.oなどは、エクスポートしているものがないのでシンボルは表示されない。しかし、-mオプションを指定して実行すると、各モジュールが組み込まれマッピングされているアドレスとサイズが表示される。

```
# ksyms -m
Address  Symbol                                    Defined by
d081f000 (4k)                                      [bsd_comp]
d0818000 (21k)                                     [ppp]
d081be90 ppp_register_compressor_Rsmp_9682e733 [ppp]
d081bf00 ppp_unregister_compressor_Rsmp_a1b928df
[ppp]
d081ce60 ppp_crc16_table_Rsmp_cacc1f6a     [ppp]
d0815000 (5k)                              [slhc]
d081504c slhc_init_Rsmp_1ca65fca           [slhc]
d0815218 slhc_free_Rsmp_b99033d9           [slhc]
d0815ccc slhc_remember_Rsmp_07972313       [slhc]
d08152d8 slhc_compress_Rsmp_cfd3a418       [slhc]
d08158e4 slhc_uncompress_Rsmp_3bb36b01     [slhc]
d0815e84 slhc_toss_Rsmp_a152cec0           [slhc]
```

ちなみに、システムがエクスポートしているシンボルは、/proc/ksymsをcatすると見られる。ここには、ローダブルモジュールがエクスポートするシンボルだけでなく、カーネルのスタティックモジュールのシンボルも含まれる。

なおカーネルは、モジュールのロード、アンロードのたびに必要となるメモリ領域を確保、開放しているので必ずしも同じアドレスにマッピングされるとは限らない。

■

Linuxにおいてローダブルモジュールのサポートは、結構よくできた機能である。実際問題としてユーザーが行わなければならない作業は、ハードウェアに依存したドライバの設定だけである。設定自体は前述のとおり、少しも難しいものではない。むしろ一番大変なのは、ハードウェアに対応するモジュールを見つけることかもしれない。というわけで、対応モジュールを見つけるために、手当りしだいロードしてみるという手段もないわけではないと付け加えておこう。

下位のハードウェアまで自動認識してくれるようになれば、言うことなしなのではあるけれども、これにはもうしばらく時間がかかりそうだ。あと今後、Linuxが広く浸透するにつれて、Linuxカーネルに含まれるモジュールだけでなく、各社（人）から配布されたモジュールを利用することも増えてくる。この時、バイナリ配布では、バージョンチェックの問題が発生する場合があるので注意しておこう。

Internet Resources

～ちょっと気になるWebサイト12選～

Server checker

Netcraft—What's that site running?

http://www.netcraft.com/whats/

サーバのホスト名をFQDNで入力すると、そのWebサイトが使用しているWebサーバとOSが調べられるサイト。検索エンジンと同じノリで調べることができる。また、どのサイトの問い合わせが多いかも参照でき、違う意味で楽しめる。

Hardware database

CDB-Query Frontend

http://cdb.suse.de/cdb/english/

SuSEの「ハードウェア動作確認データベース」ページ。他に比類がないデータ量を誇り、ベンダー名や製品名から検索できる。x86 CPU用Linuxなら、ほかのディストリビューションを利用しているユーザーも参考にできるだろう。

Document

Microsoft Windows NT Server 4.0とUNIXとの比較

http://www.ne.jp/asahi/personal/kobayashi-osamu/Translation/kirch.net/unix-nt.j.html

John Kirch氏の「Microsoft Windows NT Server 4.0 versus UNIX」を翻訳工房の小林 修氏が日本語に翻訳したドキュメント。サーバ管理者向けに、コスト面、安全性などの比較や実例を通してNTとUNIX系OSの詳細な比較がなされている。

Document

ZERO Home

http://cwaweb.bai.ne.jp/~net_hal/main.htm

Linux初心者必見、net_hal氏のパソコン入門サイト。「Linux日記」にはVine Linuxを中心に、Linuxビギナーに役立つ内容が盛りだくさんだ。アプリケーションのインストール、Xやネットワークの設定方法などが具体的に解説されている。

Scientific applications

SAL (Scientific Applications on Linux)

http://sal.linet.gr.jp/index.shtml

Linux向けに用意されている科学分野の研究支援ツール検索とその紹介ページ。各分野用アプリケーションについてのコメント、リンクURL、パッケージ情報、フリーのものについてはFTPサイト情報まで得ることができる。

Remote control software

VNC

http://www.uk.research.att.com/vnc/index.html

VNC（Virtual Network Computing）のサイト。VNCは、異種OSをGUI環境のままネットワーク越しに操作できる優れものだ。高速な回線が確保できれば、リモートからGUIでの操作を可能にする。GPLに従って無料配布されているのが嬉しい。

News Topics

The comp.os.linux.announce home page

http://www.cs.helsinki.fi/u/mjrauhal/linux/cola.html

Linuxに関する新着情報を扱うニュースグループcomp.os.linux.announceのサイト。このグループに投稿された過去の記事を閲覧できる。Newsを読むのはつらいけど、Linux関連の最新情報はチェックしたいという人にお勧め。

Japanese imput tools

Canna Home Page

http://www.nec.co.jp/japanese/product/computer/soft/canna/index.html

UNIX用日本語入力システムの定番、「かんな」のページ。NECが開発し、ソースも含め公開されている。プログラムのダウンロードサイトがたどれるほか、マニュアルがPDF形式で公開されているので、プリントアウトして活用できる。

Japanese imput tools

FreeWnn Project

http://www.freewnn.org/

フリーで公開されているUNIX用かな漢字変換システム「FreeWnn」のメンテナンスを目的としたプロジェクトのサイト。FreeWnnは、商用のWnn6とは兄弟にあたり、有志によって開発、メンテナンスが進められている。現在の最新版はFreeWnn1.1だ。

Kernel

ELKS pages

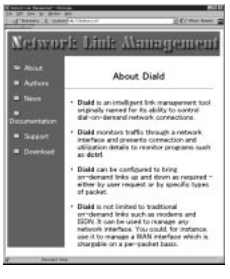
http://www.elks.ecs.soton.ac.uk/cgi-bin/ELKS/

ELKSは、パームトップPCや組込み用途をターゲットに、8086や80286などの16ビットCPUに特化したLinuxカーネルサブセット。たった400～512Kバイトのメモリで動作するという。家電製品がLinuxで制御される日も近い？

Network tools

Network Link Management

http://dialD.unix.ch/



煩わしいダイヤルアップ操作からユーザーを開放し、ネットワークインターフェイスを選ばないというon demandダイヤラ「dialD」のサイト。今年の6月に、オリジナルの作者Eric Schenk氏から、Mike Jagdis氏にメンテナンスが引き継がれた。

Software search

RPM repository on rpmfind.net

http://www.rpmfind.net/linux/RPM/

RPMパッケージの検索サイト。カテゴリ、名前、ディストリビューションで検索でき、欲しいパッケージがすぐに見つかる。また、個々のパッケージに含まれるプログラムやライブラリ、そのパッケージに必要なライブラリなどの情報もある。

Books

オープンソースソフトウェア

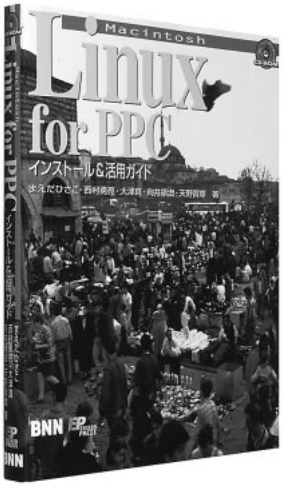
Chris DiBona、Sam Ockman、Mark Stone 編著　倉骨 彰 訳
オライリー・ジャパン
A5変形判／506ページ　本体価格 1900円

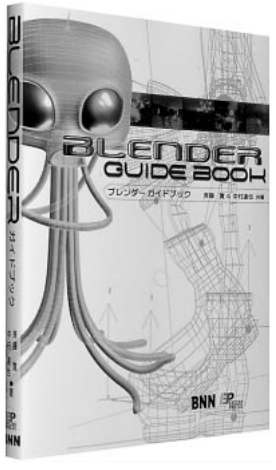
インターネットに接続しているWebサーバとして50%以上のシェアを持つApacheや、ネットスケープのWebブラウザであるNavigatorなど、インターネットの普及とともにオープンソースソフトウェアが注目されてきた。Linux自体もそうだが、Linuxで動作するソフトウェアの多くはソースコードを公開している。従来フリーソフトウェアと呼ばれていたように、フリーすなわち無料で配られているソフトウェアがオープンソースと名前を変えたとたんに、それに携わる会社がビジネス的に成功し始めたのだ。本書では、GNUプロジェクト創始者のリチャード・ストールマン氏や、Linux開発者であるリーナス・トーバルズ氏、Red Hatのボブ・ヤング氏ら14名の先駆者が、オープンソースの歴史や背景、意義について、それぞれの立場から語っている。彼らの話からソフトウェア開発の新しい流れを感じ取ってもらいたい。

Linux for PPCインストール&活用ガイド

まえだひさこ/西村勇亮/大津真/向井領治/天野賀章 著
BNN
B5変型判／282ページ／CD-ROM付き　本体価格 2500円

PowerPCを搭載するMacintoshに対応した、Linux for PowerPCの解説書である。インストールの章は短めだが、作業の要点は押さえてあるため、本書を片手に操作を進めれば、たいていはうまくいくだろう。活用方法としては以下の3つ、ネットワークの設定を含めたWebブラウザや電子メールの操作、GimpやMP3プレイヤーを使った音と映像、そしてNetatalkとSambaを用いたサーバ的な用途を紹介している。Macintoshは、Linuxプラットフォームとしてはマイナーだが、実現できることはPCと差がない。だが本書で最も多くページを割いているのは、ls、cp、rm...といったコマンドを始めとするUNIXの基本操作についてである。「MacOS使い」がLinuxを使うためには、このようなページが必要なのであろう。iMacのHDDが大きすぎて困っている人や、PC Linuxに食傷気味な人は、1ボタンで操作するX Window Systemに挑戦してみるのも良いだろう。





Blenderガイドブック

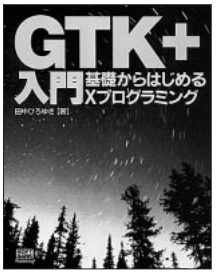
斉藤 寛、中村達也 著
エクシード・プレス
B5変型判／240ページ／CD-ROM付き　本体価格2800円

オランダのアニメーションスタジオNeoGeoによって開発された、3Dアニメーションソフトblenderは、SGI、SUN、Windows 95/98/NTのほか、FreeBSDやLinuxのX Window System上でも動作する。3Dウインドウで対話式にモデリングを行い、レンダリング、アニメーションをマウスでコントロールできる。付属CD-ROMには、フリー版のBasic Blenderを収録していて、ポリゴン、NURBSモデラー、メタボール、アニメーションテクスチャ、頂点ペイント、パノラマレンダリングなど、3Dアニメーションの作成に十分な機能を持っている。さらにライセンスを購入することにより、ラジオシティや環境マッピングのテクスチャ自動計算のような、ハイクォリティなCGを作成することができる。全ページカラーでBlenderの機能を手順に合わせて紹介しており、かわいらしい火星人やスターのサンプルを見ると、思わず自分でも3DCGを作ってみたくなる解説書だ。

LINUXがWindowsを超える日

脇 英世 著
日経BP
四六判／248ページ
本体価格 1200円

GTK+入門



田中ひろゆき 著
ソフトバンク パブリッシング
B5変型判／236ページ／CD-ROM付き
本体価格 2600円

TurboLinux4.0スタートブック

竹田善太郎、美濃村直之 著
アスキー
B5変型判／328ページ／CD-ROM付き
本体価格 2300円

Samba2.0によるWindowsネットワーク構築入門

浅野理森 著
技術評論社
B5変型判／292ページ／CD-ROM付き
本体価格 2280円

Virtual Network Computing入門〜OSの壁を越えて〜

白井徹也 編，VNC研究会 著
カットシステム
A5判／180ページ／CD-ROM付き
本体価格 2300円

人に聞けないLinuxの使い方（基本コマンド編）

アスキー書籍編集部 編
アスキー
B5変型判／232ページ
本体価格 980円

■■■■ Linuxユーザーにお勧めするUNIX書籍 ■■■■

良い参考文献を選んで作った本はよく売れると、私が長年コンピュータ書を担当していて実感しています。コンピュータのどの分野でも、本格的に勉強するなら参考文献に多く使われている書籍をお勧めします。Linuxを思う存分使いこなし、また自分自身がより深くLinuxの世界に入るには、UNIXの名著をぜひとも読んでいただきたいものです。

表の一番下の本はその後、'96年にLefflerを除いた著者陣により4.4BSD版が出版され、現在邦訳も進んでいるようです。さらに渋いところで、Silberschatz A／Galvin P著「Operating System Concepts」（'94 Addison Wesley）を習得すれば、怖いものナシです。

言語で言えば、Kernighan B W/Ritchie D M著「The C Programming Language」（'88 Prentice Hall）、Tondo C L著「The C Answer Book 2/E 」（'89 Prentice Hall）と、C言語の3部作といわれるKoenig A著「C Traps and Pitfalls」（'89 Addison Wesley）がお勧めです。

次回は、今回取り上げた何点かの出版よもやま話を少々と、そのほかのC言語やC++、コンパイラ、GNU、TCP／IPに関する本を紹介していきたいと思います。

（書泉ブックドーム　古田島）

著者	書名	発行年	出版社
Bach M J	「The Design of the UNIX Operating System」	'86	Prentice Hall
Tanenbaum A S	「Operating System：Design and Implementation」	'87	Prentice Hall
Kernighan B W	「The UNIX Programming Environment」	'84	Addison Wesley
Goodheart B／Cox J	「The Magic Garden Explained」	'94	Prentice Hall
Memeth E	「UNIX System Administration Handbook」	'95	Prentice Hall
Stevens W R	「UNIX Network Programming Ⅰ&Ⅱ」	'97/98	Prentice Hall
Stevens W R	「Advanced Programming in the UNIX Environment」	'92	Addison Wesley
Leffler／McKusick／Karels／Quaterman	「The Design and Implementation of the 4.3 BSD UNIX Operating System」	'89	Addison Wesley