News Express

Distribution Software Hardware Headline Event

Linuxカンパニーの目標は？

昨年から今年春にかけて日本や米国のコンピュータ業界では○○○バブルといった景気のよい話が花盛りだった。○○○には、「ネット」とか「Linux」というのが入るらしい。その間にLinux専業メーカーのRed HatやVA Linux、Caldera SystemsがNASDAQへ上場している。やはりバブルは弾けるのか、そういった企業の株価も最近はあまり芳しくないようだが……。

国内でも、ターボリナックス ジャパンやレーザーファイブが、ベンチャーキャピタルなどから出資を受けて、株式上場を目指している。

そんな中、業績好調で勢いのある日本オラクルが、Linux事業を行うためにミラクル・リナックス株式会社を立ち上げた。この夏よりOracle8iに特化したサーバOS（Miracle Linux？）を提供する計画になっている。

そのほかに日本市場へ進出してくることが決まっているディストリビューションを挙げておくと、9月にStorm Linux 2000日本語版が、2000年第4四半期にはCorel LINUX OSの日本語版が発売される予定だ。

また、すでに日本語版も発売されているOpenLinuxの開発元、米Caldera Systems社が、ここ数カ月のうちに日本法人を設立する見込みとなっている。

Linux市場の拡大スピード以上に、ディストリビュータとして参入してくる企業のほうが多いようにも思える。Linux業界は、まだしばらくホットな戦いが続いていくことだろう。

Hardware　発売日　2000年4月19日

Netfinityの全モデルでLinux正式サポート「Netfinity」新シリーズ、新モデル

URL　http://www.ibm.co.jp/

発売	日本アイ・ビー・エム株式会社
TEL	0120-04-1992
価格	13万8000円〜

日本アイ・ビー・エムは、PCサーバ「Netfinity（ネットフィニティ）」の新シリーズとして、Netfinity 7600、7100、5100、4500Rの4シリーズを追加し、現行シリーズNetfinity 5600、3000、1000に新モデルを追加した。

7600シリーズは、Pentium III Xeon 550MHz（2Mバイト L2キャッシュ）を搭載したハイエンド・エンタープライズサーバで、4ウェイSMPに対応。512Mバイトメモリ、RAIDアダプタを標準装備した高さ8Uのラックモデル。7100シリーズは、Pentium III Xeon 550MHz（512KバイトL2キャッシュ）を搭載したエンタープライズサーバで、4ウェイSMPに対応。256Mバイトのメモリを搭載し、ラック型とタワー型の2モデルが用意されている。

Netfinity 5100シリーズと4500Rシリーズは、メモリ128Mバイト標準搭載で、2ウェイSMPに対応。5100シリーズは、Pentium III 667MHzを搭載した部門アプリケーションサーバで、ラック型とタワー型の2モデルの構成。また、4500Rシリーズは、Pentium III 733MHzを搭載したラック型の1モデルのみの構成。オプションの拡張キットにより、ホットスワップHDDを6台まで搭載可能。

また、現行シリーズであるNetfinity 5600シリーズにPentium III 800MHzまたは733MHzを搭載した4モデル（ラック型またはタワー型）を追加。さらに、3000シリーズにPentium III 650MHz搭載の1モデル、1000シリーズに同600MHz搭載の1モデルを追加した。新モデルの希望小売価格は、5600シリーズが73万5000円から、3000シリーズが23万5000円から、1000シリーズが13万8000円からとなっている。

今回発表した新シリーズから、従来の1年間オンサイトサービスに加え、3年間の部品保証サービスを実施。Red Hat Linux 6.1日本語版とTurboLinux Server日本語版6.0が正式にサポートされた。また、Linux版の「ロータス ドミノ」を標準添付する（1000シリーズを除く）。

Netfinity 1000

Hardware　発売日　2000年5月24日

OpenGLアクセラレータ搭載のLinuxワークステーション「Silicon Graphics 230／330／550」

URL　http://www.sgi.co.jp/

発売	日本SGI株式会社
TEL	0120-161-086
価格	33万6000円〜

日本SGIは、OSにLinuxとWinodws NTを採用した「Silicon Graphics 230 Visual Workstation」、「同 330」、「同 550」を発表した。

同社初のLinuxワークステーションで、CPUにはインテルのPentium IIIまたはPentium III Xeonを採用している。プレインストールOSは、Red Hat Linux 6.1かWindows NT Workstation 4.0を選択可能。

OpenGLハードウェアアクセラレーションを実現する「SGI ProPack 1.2 for Linux」パッケージによって、Linux上でOpenGLの3Dグラフィックス機能をハードウェアレベルで高速化できる。

エントリーモデルの230は、CPUにPentium III 667MHzを採用したモデルと、同733MHzを採用したモデルの2種類が用意されている。

ミッドレンジモデルの330は、CPUにPentium III 800MHzを最大2基まで搭載可能。ハイエンドモデルの550は、CPUにPentium III Xeon-800MHzを最大2基まで搭載可能。

価格は、230の667MHzモデルが、Linuxプレインストールで33万6000円、Windows NT4.0プレインストールで34万8000円。733MHzモデルが、Linuxプレインストールで56万2000円、Windows NT4.0プレインストールで57万4000円。330と550は6月下旬発売で価格は未定。

Hardware　発売日　2000年5月10日

PowerEdgeシリーズ全機種でLinuxをサポート「PowerEdge」シリーズ

URL　http://www.dell.com/jp/

発売	デルコンピュータ株式会社
TEL	044-556-6190
価格	16万3000円〜

デルコンピュータは5月10日より、サーバ製品「PowerEdge（パワーエッジ）」の全製品ラインで、Red Hat Linux 6.2Jを標準搭載したモデルの販売を開始した。顧客の希望に応じて、同OSをサーバ製品にインストールして出荷する。販売されるモデルは、PowerEdge 1300、2400、4400、6400、2450、6450。

また、プレインストールモデル以外の製品を購入した顧客もRed Hat Linux 6.2Jを導入できるように、ドライバなどを同社のハードウェア構成に合わせたパッケージ「Red Hat Linux 6.2J SBE（System Builder Edition） 2」（サーバ製品に同梱）を、同日より1万5000円で販売開始した。

付属するアプリケーションは、Red Hat Linux 6.2J デラックスと同じだが、サポートがオリジナルのデラックス版より期間が長くなっており、電話とメールが90日、専用FTPサイトへのオンラインアクセスは180日間と単体で買うよりもお得な内容となっている。

PowerEdge 1300

Hardware

発売日 2000年5月22日

2Uサイズのラックマウント Linuxサーバ「NL Server 2100」

URL http://www.nlcomputer.com/

発売 ノーザンライツコンピュータ株式会社
TEL 044-850-2391
価格 69万円〜

ノーザンライツコンピュータは、LinuxをOSに採用したISP（Internet Service Provider）、ASP（Application Service Provider）向けサーバ「NL Server 2100」を5月22日より発売する。価格は69万円から。

CPUはPentiumIII 800MHzを最大２個まで搭載可能。また、ストレージ・サブシステムはホットスワップ可能で、ハードディスクを最大４基まで搭載できる。同時にソフトウェアRAIDに対応、オプションでハードウェアRAIDにも対応している。メモリは最大2Gバイトまで拡張可能、2Uラックマウントシャーシのため、ネットワークの成長に応じたサーバの追加が容易、OSには、Red Hat Linuxをプレインストール、あらかじめISP、ASPなどでの使用を前提としたサーバ用途の設定がなされているなどの特徴がある。

Hardware

発売日 2000年6月

高性能マイクロサーバ「silver neon」

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

発売 株式会社アクアリウムコンピューター
価格 40万円前後

アクアリウムコンピューターは、高性能なマイクロサーバ「silver neon」を発売する。同社の「blue grass」「white neon」でのデザイン重視のコンセプトを受け継ぎ、新機種ではCPUはCeleron 533MHzに、メモリは128Mバイトに、内蔵ハードディスクは20Gバイトに拡張し、大規模アプリケーションにも対応できるようにした。価格は40万円前後、出荷は6月末の予定となっている。

また、外部SCSI、10/100BASE-Tネットワーク、RS-232C×2ポート、パラレル、モニター、キーボード、マウス、USB×2、PCIスロットといった多くのインターフェイスを備えている。

データ保護用に緊急バッテリーを内蔵していて、万一停電した場合にシャットダウンする機能がある。

OSは購入時にTurboLinuxまたはRed Hat Linuxを選択する。サーバ管理ソフトとしてHDE Linux Controllerが付属する。

Hardware

発売日 2000年5月16日

Linuxをバンドルした高性能PC「Precision WorkStation 220」

URL http://www.dell.com/jp/

発売 デルコンピュータ株式会社
TEL 044-556-6190
価格 18万1500円〜

シリーズのエントリーモデル「Precision WorkStation 220」の、TurboLinux Workstation日本語版6.0バンドルモデルを5月16日から発売した。

Precision WorkStation 220は、CPUにPentiumIII 533MHz〜866MHz、メモリにPC800／PC700のECC RDRAMを採用した高性能モデルで、BTOにより組み合わせが選べる。

発売を記念して、同モデル購入者のうち限定300台にTurboLinuxラーニングキットが無償提供される。そして、TurboLinux認定トレーニング・ベーシックコース（兼松コンピュータシステム実施）の受講料が定価（3万5000円）の半額（1万7500円）になるという特典がある。

なお、ミッドレンジモデルの「Precision WorkStation 420」のバンドルモデルも後日発売の予定だ。

Software

発売日 2000年6月16日

eビジネス対応を強化したRDBMS「DB2 UDB（ユニバーサル・データベース）V7.1」

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

発売 日本アイ・ビー・エム株式会社
TEL 0120-04-1992
価格 3万8800円〜237万3000円

日本アイ・ビー・エムは、DB2の新バージョン「DB2 UDB（ユニバーサル・データベース）V7.1」を6月16日より出荷する。

インターネット技術で使われるXML（eXtended Markup Language）を直接扱えるので、XML文書の検索や、DB2の表からXML文書の作成が簡単にできる。単語や語句のあいまい検索、ワイルドカードを使用したメモリ内検索（イン・メモリ・サーチ）も可能にしている。オプションの高速テキスト検索「DB2ネット・サーチ・エクステンダー」を利用すれば、1日に9000万件を超えるテキスト検索が応答時間0.5秒未満で可能だ。

Webアプリケーションサーバ「WebSphere」も標準でバンドルしている。

DB2 UDB V7.1の価格は、パーソナル版が3万8800円、ワークグループ版が12万9900円、エンタープライズ版が184万8000円。エンタープライズ拡張版は237万3000円。AIX、HP-UX、NUMA-Q、Windows 95/98、Windows NT、Windows 2000、OS/2、Solaris、Linuxに対応する。

News Headline from 日刊アスキーLinux

www.Linux24.com

GNOME 1.2がリリース

2000年5月26日

GNOME Projectは5月25日、フリーなデスクトップ環境「GNOME」（GNU Network Object Model Environment）の安定最新版「GNOME 1.2」をリリースした。GNOMEはGNUプロジェクトの一部で、KDEと並んで人気のあるデスクトップ環境だ。

GNOME 1.2では「GNOME UIチーム」が結成され、ユーザーインターフェイスがより洗練された。また、安定性の向上やヘルプ／ドキュメントの充実、GNOME API仕様の文書化などもなされている。

ソースコードのダウンロードは、GNOMEのFTPサイトから行うことができる。または、GNOMEのサポートおよびネットワークサービスを販売する米Helix Code社が公開しているバイナリパッケージをインストールすることも可能だ。現状では、Solaris 2.7（UltraSparc）、LinuxPPC 2000、Debian GNU/Linux 2.3（Woody）、TurboLinux 6.0、Red Hat Linux 6.0、6.1、6.2、SuSE Linux 6.3、6.4、Yellow Dog Linux Champion Server 1.2、Linux Mandrake 6.1、7.0、7.1、Caldera OpenLinux eDesktop 2.4に対応したパッケージが存在する。

GNOMEプロジェクト
(http://www.gnome.org/)

PHP 4.0.0リリース

2000年5月26日

Webアプリケーション構築用スクリプト言語「PHP 4.0.0（以下PHP4）」がリリースされた。

「PHP」は、Webサーバ「Apache」のモジュールとして使用される、HTML埋め込み型のスクリプト言語である（CGIにより実行することもできる）。XML、IMAP、LDAPなどに対応しており、PostgreSQL、Oracle、Sybaseといったデータベースと連携したWebアプリケーションの作成などに用いられる。ライセンスは、PHP4本体はオープンソースライセンスのひとつである「The PHP License」、新たに採用されたスクリプトエンジン「Zend Engine」は「QPL（The Q Public License Version 1.0）」を用いている。

「PHP4」の最大の特徴は、スクリプトエンジンがイスラエルのZend Technologies社（PHP開発チームのコアメンバーが設立）の「Zend Engine」に変更されたことである。この変更により、大幅なパフォーマンスの向上が実現している。

「Zend Engine」は以下の特徴を持つ。（1）実行時にコンパイルすることで、大規模アプリケーションの速度向上が期待できる。（2）「リソース管理機能」により、メモリ節約、速度向上、リソース管理の自動化を実現。（3）IIS（Windows NT／2000上）やAOL Serverなどのネイティブ APIに対応（従来は、Apacheのモジュールとしてしか組み込めなかった）。（4）アドオンによる機能拡張が可能。（Zend Technologiesから商用ベースの「Zend Optimizer」、「Zend Compiler」、「Zend Cache」などがリリース予定）。

「PHP4」本体の主な変更点は以下のとおり。（1）セッション管理機能を標準的な機能として提供。（2）PHP3上位互換、Perlの「ヒアドキュメント」と同等な構文の追加、foreach文の追加などの言語仕様の強化。（3）標準クラスライブラリ「PEAR（PHP Extension and Add-on Repository）」の採用。

PHP（http://www.php.net/）
Zend Technologies
(http://www.zend.com/)

Linuxカーネル 2.4.0-test1リリース

2000年5月26日

Linus Torvalds氏は5月25日、Linuxカーネル 2.4.0-test1をリリースした。これはカーネル2.4.0ではないが、従来の2.4.0 pre-xバージョンと比べて、一段とリリースに近づいたものだという。

またAlan Cox氏は同日、2.4.0リリースに必要な作業のリストをLinux-kernelメーリングリストに投稿した。そのリストには、「セキュリティ」「ブート時のエラー」「修正は存在するもののマージされていないもの」「致命的ではないもの」など、100点以上のバグが取り上げられている。Linus Torvalds氏は3週間出かける予定があるので、そのあいだ、Alan Cox氏が「2.4.0-ac」（Alan Coxの「ac」）シリーズのカーネルをメンテナンスするという。

アナウンス
(http://www.kernel.org/pub/linux/kernel/v2.4/README-2.4)

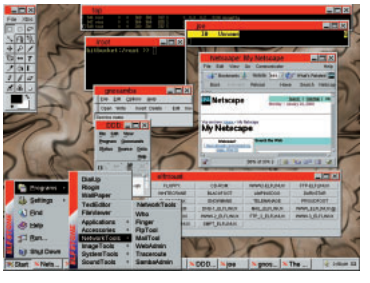
「Motif」を使用したディストリビューションがリリース

2000年5月23日

5月15日に、ソースコードが公開されたGUIツールキット「Motif」を使用したLinux ディストリビューション「Elfstone Linux」が、米Elfstone Softwareからリリースされた。

Elfstone Linuxは、ソフトウェア開発者やネットワーク管理者をターゲットとして開発されているディストリビューションである。「Motif」環境として、同社の開発した「Elfstone RTX（Run Time Xpress） Runtime Libraries」を使用している。対応プラットフォームはx86のみとなっている。価格は69.95USドル。

「Elfstone Linux」の主な特徴は以下のとおり。（1）カーネル 2.2.6、XFree86 3.3.3を採用。（2）GUIツールキットに「RTX」を使用したXFree86環境。（3）RPMパッケージだけでなく「Slackware」のパッケージも扱えるパッケージ管理ツールを収録（今後そのほかのパッケージ



形式にも対応する予定)。(4) 多くのライブラリをアセンブラで書き換えたことによる、パフォーマンスの向上。

ダウンロードは同社のWebサイトか、FTPサイトから可能。ファイルはCD-ROMイメージ(ISO)になっており、ファイルサイズは293Mバイト。

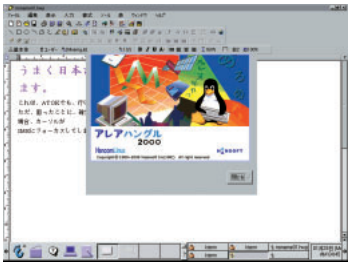
米Elfstone Software
(http://www.elflinux.com/)

韓国製のワープロが、Red Hat Linux日本語版にバンドル

2000年5月23日

レッドハットは5月23日、韓国のHancomLinux社と提携した。この提携により同社は、「Red Hat Linux日本語版」にHancomLinuxの開発するワードプロセッサ「HancomWord」日本語版を、独占的にバンドルすることになる。

現在、「HancomWord」日本語版はベータテスト中で、HancomLinuxのWebページからダウンロードできる。また、ベータ版の不具合の修正は、レッドハットと韓国のHancomLinuxの協同作業により、すでに行われており、バンドル版では不具合が修正されるという。



「HancomWord」は、5月に行われた「LinuxWorld Expo/Tokyo 2000」では「アレアハングル2000」として展示されていた製品だが、今後は、正式名称として「HancomWord」が使用される。

同製品の特徴としては、中国語、ハングル、英語、日本語の4カ国語への対応や、Microsoft Word、一太郎とのファイル互換性を持っていることなどがあげられる。

韓国のHancomLinuxは、「HancomWord」を主力製品とするLinuxソフトウェアベンダー。同社は、今回の提携を機に日本法人を設立する。また5月中には、「HancomWord」日本語版をサポートするために、顧客サポートセンターをレッドハットの社内に開設する予定だという。

レッドハット
(http://www.redhat.com/jp/)
HancomLinuxの日本語ページ
(http://www.hancom.com/jp/)

FD 1枚で、リアルタイム機能を持つLinux「miniRTL」がリリース

2000年5月22日

「The Thinking Nerds Projects」は、リアルタイム機能を持ち、1枚のフロッピーディスクに収まる容量のLinuxディストリビューション「miniRTL(Real-Time Linux)」をリリースした。

The Thinking Nerds Projectsは、「miniRTL」のほかにも、米FSMLabsの開発する「RTLinux」のためのリアルタイムコントローラなどを開発しているオープンソースプロジェクト。

miniRTLはの主な特徴は以下のとおり。(1) リアルタイム機能は米FSMLabsの開発する「RTLinux」がベース。(2) ハードディスクは不要で、フロッピーディスク、フラッシュメモリ(2Mバイト)から起動可能。(3) シェルは「ash」を採用。(4) glibc 2.0.7を採用(glibc 2.1.xはファイルサイズが大きすぎるため)。(5) CGIをサポートした、サイズの小さいWebサーバ(thttpd)を搭載し、Web経由でシステムのモニタリングが可能。

動作条件は、CPUがIntel386 DX/486 DXシリーズ、メモリ8Mバイトとなっている。

配布はディスクイメージで行っており、同プロジェクトのFTPサイトから、以下の2種類がダウンロードできる。どちらもファイルサイズは1.4Mバイト。(1) miniRTL2.0PRE1.img:カーネル2.2.12ベース、Pentium、Pentium II/IIIでも動作可能。(2) miniRTL2.0PRE3.img:カーネル2.2.13ベース、Intel386 DX/486 DXシリーズのみで動作。

The miniRTL project
(http://www.thinkingnerds.com/projects/minirtl/minirtl.html)

SuSE Linux、IBMの「S/390」に移植へ

2000年5月19日

ドイツのSuSEは5月17日、同社のLinuxディストリビューション「SuSE Linux」をIBMのメインフレーム「S/390」に移植すると発表した。6月下旬に、初のベータ版がSuSEのFTPサイトで公開される予定。

S/390は、独自アーキテクチャのCPUを搭載し、ハードウェアの機能としてVirtual Machine(VM)をサポートしている。これによって、1台のS/390の内部にいくつもの仮想的なS/390を持つことができ、それぞれのVM上で個別のOSを動作させることが可能だ。あるVMでOSがクラッシュしても、ほかのVMにまで影響が及ぶことはない。つまり、VMとはUNIXの「プロセス」という概念をマシンにまで拡張したものといえる。

S/390でLinuxを動作させることによって、Apacheなどのインターネットアプリケーションと、基幹系OS/390上で動作するメインフレームのI/O性能を生かしたデータベースなどを、同一マシンで稼動させることができる。

メジャーなディストリビュータによるLinuxのS/390への移植は、TurboLinuxに続いてこれで2件目。

プレスリリース
(http://www.suse.de/en/news/news/newpage/IBM_S390.html)

米Red Hat、IA-64対応Linuxディストリビューションをリリース

2000年5月18日

米Red Hatは5月17日、IA-64対応Red Hat Linuxの初期バージョンを開発者向けにリリースした。これには、GNUの開発ツールやXFree86、GIMPなどのプログラムが含まれている。Linuxカーネルは Trillian Projectによって、コンパイラは Red HatやSGIによってすでにリリースされており、ディストリビューション全体が移植されるのはTurboLinuxに続いてこれが2番目。

IA-64アーキテクチャに関する詳細な技術資料 は、5月10日に米Intelが同社Webサイトで広開しており、秘密保持契約に

サインすることなしに誰でも見ることが可能だ。Red Hatは、今回リリースされたディストリビューションとIA-64の技術資料を併せて参照することで、開発者はItanium出荷前からLinux/IA-64で動作するプログラムを開発することが可能になるとしている。

プレスリリース
(http://www.redhat.com/about/2000/press_itanium-alpha.html)

CorelとInpriseの合併計画撤回へ

2000年5月17日

カナダCorelと米Inpriseは5月16日、両社の合併合意を取り消したと発表した。この件に関して違約金などの支払いは発生しない。

CorelはCorel LINUXやCorel Officeなどを、InpriseはJBuilderやInterBaseなどのLinuxに対応した製品を持っている。合併により強力なLinux企業を目指すとしていた。

Inprise（http://www.inprise.com/）
Corel（http://www.corel.com/）

SGIがIA-64対応コンパイラをGPLでリリース

2000年5月17日

米SGIは5月15日、Intelの次世代CPUアーキテクチャ「IA-64」に対応したコンパイラ「Pro64」をオープンソースでリリースした。ライセンスはGPLで、C、C++、Fortran90／95をサポート、gccおよびg++との互換性を備えている。IA-64のプロセッサは、第1世代である「Itanium」の出荷が今年後半に予定されている。Pro64には以下のものが含まれる。（1）Complete C、C++、Fortran 90 optimizing compilers（2）Fortran runtime libraries（3）Vector math library（4）OpenMPTM support library（5）Man pages（6）Assign command

Linux/IA-64の開発は、主要なLinuxディストリビュータなどが参加する「Trillian Project」によって進められており、すでに成果のソースコードが公開されている。今回のコンパイラのリリースにより、Itanium出荷と同時にLinux/IA-64システムを稼動できる環境が揃ったことになる。

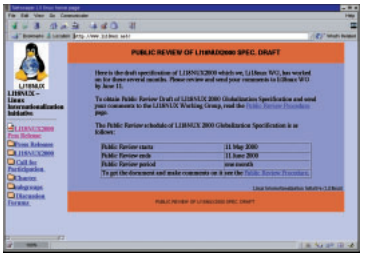
IA-64で採用されたEPICと呼ばれるアーキテクチャは、多数の命令を同時実行できるようにコンパイラが最適化を行うため、性能はコンパイラによって大きく左右されると言われている。現在のPro64は、IA-32で動作するクロスコンパイラだが、近い将来にはIA-64のネイティブでも動作可能になる予定。

Pro64についての情報
(http://oss.sgi.com/projects/Pro64/)
Trillian Project（http://www.linuxia64.org/）

LI18NUX、「LI18NUX 2000国際化仕様」ドラフトの公開レビューを開始

2000年5月11日

Linuxおよびオープンソースの国際化に関する標準化団体「Linux Internationalization Initiative」（LI18NUX）は5月11日、「LI18NUX 2000国際化仕様」（以下、LI18NUX2000）のドラフトをhttp://www.li18nux.net/で公開し、レビューを開始した。期間は1カ月後の6月11日までで、li18nux2000@sun.comでコメントを受け付ける。仕様確定は今年8月を目標としている。



LI18NUX2000は、商用UNIXが実現している国際化機能を元に作られた仕様で、国際化アプリケーションが必要とするOSの機能やインターフェイスを定義する。これに適合したLinuxディストリビューションはUTF-8を使用した多言語機能を提供することができる。また、APIのセットが規定されることから、ディストリビューション間で互換性のあるアプリケーションが容易に作成可能になる。

ドラフトでは（1）ベースライブラリ（2）シェルおよびユーティリティ（3）プログラミング言語（4）GUI（5）グラフィックライブラリ（6）グラフィックツールキットとXサーバ（7）入力メソッド（8）出力メソッド（9）ネットワークサーバ（11）インターネットツール（12）印刷などについて、それぞれ必要とする機能や関数、実装例、将来の方向性が規定されている。また、仕様への適合性に応じて「Level 1」と「Level 2」という2つのレベルが定義される。

LI18NUXは5月8日、Linuxディストリビューションの互換性向上を目標とする標準化団体「Linux Standard Base」との合併を発表している。

LI18NUX（http://www.li18nux.net/）

Linux カーネル 2.2.15がリリース

2000年5月8日

Linux カーネル 2.2.15がリリースされた。

Linuxのカーネルには安定版と開発版があり、カーネルのマイナーバージョンナンバー、つまりバージョンナンバーの小数点以下1ケタ目の数字が、偶数なら安定版で、奇数なら開発版となっている。今回リリースされた2.2.15は、安定版（2.2.x）の最新バージョンとなる。ちなみに、次の安定版となる2.4.xシリーズは、2000年夏のリリース予定となっている。

2.2.15は、前バージョンの2.2.14がそうであったように、安定版の特徴であるバグフィックスとドライバの追加、アップデートが主体になっている。このバージョンには、新たに加わったI2O（Intelligent Input／Output）のサポート、ISDN関連における「多数」のドライバの追加、アップデートなどが含まれる。

I2O（Intelligent Input／Output）は、OSに依存しないドライバの作成を目的とするPCIのスーパーセット。

ソースコードのダウンロードは、「The Linux Kernel Archives」の日本のミラーサイトや、Ring Serverプロジェクトのミラーサイトなどから可能。ファイルサイズはgzipで圧縮されたものが約16Mバイト、bzip2で圧縮されたものが約13Mバイト。

リリースノート
(http://www.linux.org.uk/VERSION/relnotes.2215.html)

Distribution

新着ディストリビューション

Linux2000G

グラフィック用途に特化したディストリビューションの誕生だ。Linux２０００Gは、堅牢なサーバ用ＯＳのLinuxを、タイトなグラフィック作業のプラットホームにしよう、というコンセプトのもと開発された。斬新なデスクトップのデザインをはじめ、デザイン重視の秘密を探ってみた。

Linux for PPC Japanese Edition 3

x８６に次ぐLinuxプラットフォームである、PowerPC用のLinuxが今、活発な動きを見せている。日本語化されたデスクトップ環境など前バージョンの長所を引き継いだうえに、MacOSエミュレータという新たな力を得たLinux for PPC Japanese Edition 3を紹介する。

Red Hat Linux 6.2J プロフェッショナル

e-コマースもLinuxの時代へ。Raw I/O、クラスタリングといった、基幹サーバに必須機能を搭載した、Red Hat Linux 6.2J プロフェッショナルのリリースだ。これまで商用ＵＮＩＸの牙城であった基幹分野へ、Linuxの先鋒としてなぐり込みをかける。基幹業務にも耐えるという、その新機能を見てみよう。

Linux2000G

Linux2000Gは、PCとMacintosh上で動作する、グラフィック機能を強化したディストリビューションだ。PC用はRed Hat Linux 6.0、Macintosh用はLinuxPPC 1999をそれぞれベースにして開発され、ヒロ杉山氏による、斬新なデスクトップ（**画面1**）やアイコン（**画面2**）のデザインが特徴だ。カーネルはバージョン2.2.15pre4、ライブラリはglibc2.1.3、XはXFree86 3.3.5が採用されており、株式会社ホロンが1万1800円で発売している。ユーザー登録後90日間は電話、FAX、メールのいずれかで、件数無制限のサポートを受けられる。

グラフィック専科

Linux2000Gの「G」はGraphicsの「G」である。PhotoshopなどをWindowsやMacに導入すると、びっくりするほど費用がかかるが、さいわいLinuxには、フリーで使えるGimpやTgifといったグラフィックツールがある。Linux2000Gは、デスクトップに起動用のアイコンを配置して、それらのツールが快適に使えるように環境が整えられている。

製品名　Linux2000G
価格　1万1800円
問い合わせ先　株式会社ホロン
03-5282-5101
http://www.linux2000g.ne.jp/

インストールCD（PC用とMac用）	
ソースCD	
インストーラ起動用フロッピー	
Shade for Linux Preview Kit（体験版）CD	3Dグラフィックツールの体験版
インストールマニュアル（PC用とMac用）	インストール解説
オペレーションマニュアル	Linuxの基本操作を解説
グラフィックマニュアル	グラフィックツール使用法の解説

Linux2000Gの収録物

■Gimp

GNUが開発するこの画像処理ツールは、とてもフリーとは思えないほど多くの機能をもち、「Photoshopキラー」の異名をもつ。GimpはSchemeという言語で書かれたプログラムを理解する。そこで、Schemeを使うScript-Fuというスクリプトを書けば、かなり凝ったロゴがなどが作成できる。LinuxのGimpは、とても安定しているので、タイトな画像処理を快適に行うことができる（**画面3**）。

■Tgif

Tgif（**画面4**）はフリーのドローツールで、XIMという入力方式で日本語も使える優れものだ。Linux上で図版を含んだ文書を作成するときに重宝する。

■Shade for Linux Preview Kit（体験版）

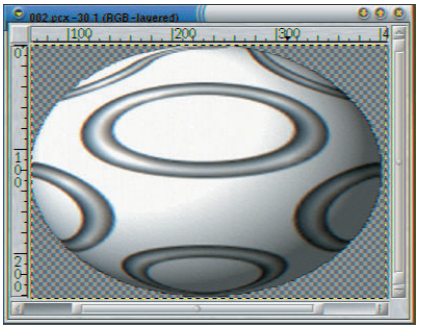
3Dグラフィックツールとしては、WindowsやMacで定番のShade（**画面5**）が収録されている。扱える画像サイズなどに制限がある体験版だが、Shadeの高機能を体験することができる。

初めての人も簡単インストール

LinuxはWindowsのアプリケーションと違い、インストールするのにはハードディスクに専用の領域を作る必要がある。この領域をパーティションと

画面1
斬新なデザインのデスクトップ

画面2
Linux2000Gオリジナルのアイコン



画面3
Gimpに用意されたスクリプトをもとに作成した画像

画面4
ドローツールのTgif

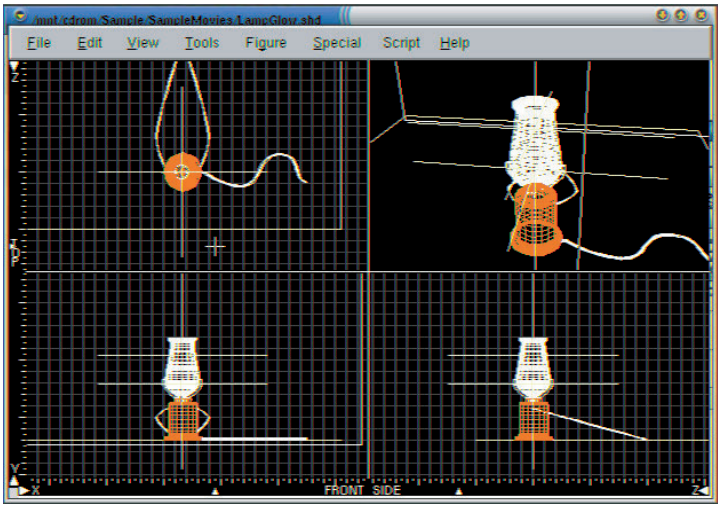
呼び、その作成は初心者にとって、つまずきやすいポイントとなっている。誤ったパーティション作成をすると、今までの大切なデータが消えてしまったり、OSが起動しなくなったりと、かなり深刻な問題を引き起こす。そこでLinux2000Gでは、直感的な操作で、Linux用のパーティションが作成できる、パーティションマジックプレップツール（**画面6**）を用意している。このツールは、Windowsマシンへインストールして使うのだが、実際に使ってみると、ハードディスク上でWindowsが使用する領域か、それ以外の領域かを選択するくらいで、パーティションサイズも自動で設定してくれる。このツールによって、初心者には鬼門のパーティション作成がとても楽になっている。

いちから始めるLinux

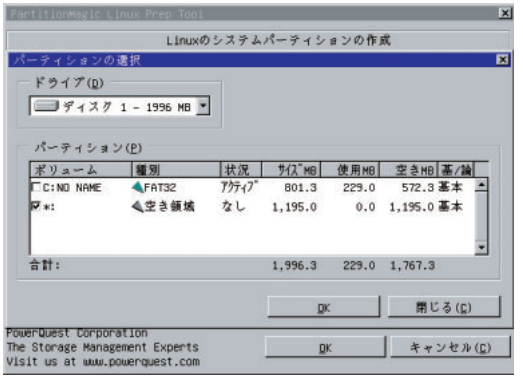
「WindowsやMacよりも安定して動作するするのは魅力だけど、Linuxは操作が難しいというし」という人は多いだろう。しかし、GimpやTgifといったツールは、あらかじめデスクトップ上に配置されたアイコンから起動できるようになっているし、これらのツールは、ほとんどマウスで操作できる。とはいえ、Linuxを使っていると、どうしてもコマンド操作が必要になることがある。そこで、Linux初心者が不慣れなコマンド操作について、ログインやログアウトといった基本操作とともに、付属マニュアルで解説している。マニュアルではこのほかにも、メールクライントや、RPMパッケージのインストール方法、さらにSambaやNetatalkを使ったファイルサーバの設定方法まで解説されているので、初心者も安心してLinuxが学べるだろう。

だってLinuxだもん

このように、グラフィックワークステーションとしての性格が強いLinux2000Gだが、Linuxが元々得意とする、ファイルサーバやプリンタサーバという使い方も有効だ。Linux2000G上で、SambaやNetatalkを立てれば、各プラットホームで作成したファイルがネットワーク上で共有できるので、複数マシンが混在する環境で、グラフィック作業を行っているユーザーにお勧めのディストリビューションといえるだろう。



画面5
3Dグラフィックツールとして人気のShade（体験版）



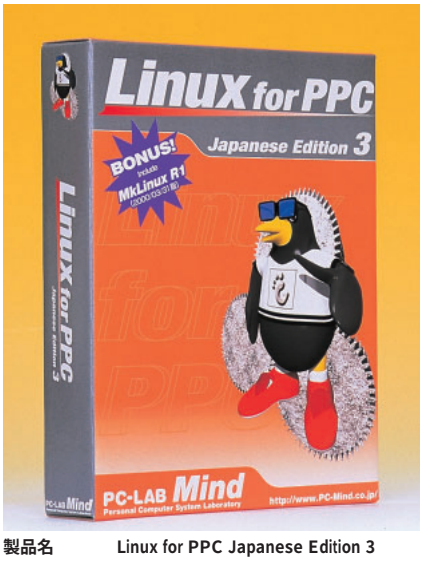
画面6
パーティションマジックプレップツール

Linux for PPC Japanese Edition 3

マインドは、Power Macintosh向けディストリビューションである、Linux for PPC Japanese Edition 3（以下Edition 3）の販売を4月25日より開始した。価格は6800円。同社が'99年の9月から発売していたLinux for PPC Japanese Edition 2（以下Edition 2）のバージョンアップ版である。

Editon 2では、オープンソフトウェアのみからなるスタンダード版と、商用ソフトとサポート権を含んだオフィシャル版の2種類があったが、Edition 3では、パッケージは1種類に統一され、パッケージを購入したユーザーは電子メール、FAXでのサポートが受けられるようになった。

対応機種は、Edition 2よりも大幅に拡大しており、最新のPower Macintosh G4、iMac、iBookなどでも動作するようになった。



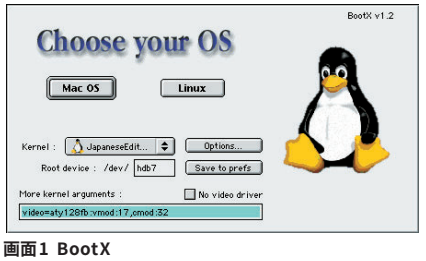
製品名　Linux for PPC Japanese Edition 3
価格　6800円
問い合わせ先　株式会社マインド
http://www.PC-Mind.co.jp/

Mk Linux R1は含まれず

パッケージ内容は、インストール（バイナリ＋ドキュメント）、ソースの各CD-ROMに加えて、MkLinux R1のCD-ROMが含まれている。現在のLinuxPPCは、初代Power Macintosh（使われている拡張バスに基づいて、NuBus Macintoshと呼ばれる）では動作しないので、これらの機種のオーナーには、嬉しい配慮だ。

ただ、現在出荷されているEdition 3の一部には、MkLinux R1のCD-ROMが含まれていないものがある。これは、発売後にMk Linux R1に含まれるソフトウェアの一部にライセンス上の問題が見つかったためのようだ。その場合でも、ユーザー登録はがきの返送時に、「MkLinuxを希望する」と書き添えておけば、問題解決後に送付してくれるそうだ。

マニュアルは、とても丁寧に書かれたインストールガイドが付属している。Power MacintoshとPCではハードウェアの構成が異なるため、ディスクのパーティション設定など、一部の重要な部分で操作方法が異なっている。そのためPCのLinuxに慣れているユーザーがインストールする際には、重宝するだろう。もちろん、初心者にとっても心強いものだ。



画面1 BootX
Linux for PPCは、MacOS上のBootXから起動する。カーネルに渡すオプションもここで指定するようになっている。

Power MacintoshでLinuxを起動する時は、MacOS上のプログラムであるBootX（**画面1**）を用いるようになっている。BootXは、MacOSの起動前でもデスクトップからでも実行できるようになっている。またカーネル起動時のオプション設定もBootXから行える。ここでグラフィックスドライバや、X Window Systemの解像度と色数の指定をするのが一般的だ。この時「No video driver」ボックスにチェックが入っていると、描画のアクセラレーションが行われず、Xのデスクトップ表示が非常に遅くなってしまうが、使用する機種に合わせてドライバを指定することで、かなり高速化できる。

Macintoshの1ボタンマウスで、X Window Systemの3ボタンをエミュレートする場合の設定もここで行えるが、かなり無理があるので、できれば3ボタン（ホイール付きがベター）のマウスを用意すべきだろう。編集部の青白Power Macintosh G3で試した際には、Microsoftのオプティカルマウス（USB）

画面2 デスクトップ
ウィンドウメニューは日本語化されており、操作にとまどうことはない。違うのは、壁紙のペンギンだけ？

を用いたが、ホイール機能も含めて完全に使用できた。

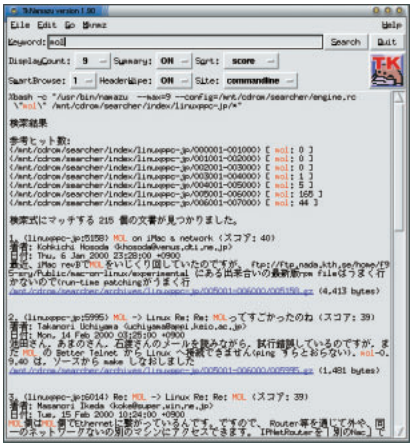
最新ソフトウェアを採用

カーネルは、安定版の2.2.15、Cライブラリはglibc 2.1.3をそれぞれ採用している。またXFree86 3.3.6（TrueTypeフォント対応パッチ）、GNOME 1.0.55、KDE 1.1.2とX Window System関係も最新版にアップデートされている。

インストーラは、Red Hatスタイルのテキストベースでマウスが使えないため、マウス依存度の高いMacOSユーザーは、多少面食らうことになるだろう。余談だが、このインストーラはRPMパッケージのインストール中に、画面の左上に新機能の紹介文などが表示されるようになっている。PC用の有名なOSのインストーラに合わせたのだろうか。このような表示をすることで、慣れないユーザーでも今何をしているのか見当がつけられるようにしてあるのだろう。

インストールが終了してX Window Systemが起動してしまえば、Power MacだろうとPCだろうと同じLinuxであり、使用感はほとんど変わらない（**画面2**）。メニューを見ても、PC用ディストリビューションと同様のソフトウェアが備わっている。

最近のディストリビューションでは、使用中のトラブルシューティングに役立てるために、Linux関連のメーリングリストのログファイルを収録しているものが多い。Edition 2にもlinuxppc-jpなどのログと全文検索システムのNamazuが収録されていたが、Edition 3ではGUIの検索用クライアントであるTkNamazuがGNOMEやKDEのメニューに組み込まれている。インストールCD-ROMをマウントして、メニューからTkNamazuを起動すれば、簡単にコミュニティのノウハウにアクセスできる（**画面3**）。



画面3 検索クライアントTkNamazu
インストールCD-ROMをドライブに入れておけば、メニューから一発で検索が行える。

目玉はMac-on-Linux

Edition 3最大の目玉は、Mac-on-Linux（以下MOL）だ。MOLは名前が示す通り、PowerPC Linux上でMacOSを起動させるエミュレータである。PC用のVMwareを思い浮かべれば、どのようなシステムか想像できるだろう。

一般にMacintosh用のエミュレータは、MacOSのROMイメージが必要だが、MOLは自身が動作しているマシンのROMを、ファイル化するユーティリティを同梱している。別のMacintoshからROMイメージを持ってきたりする必要はないので、法的にも問題はない。

7.5.2から9.0まで、すべてのMacOSがブート可能で、もちろん日本語版OSにも対応している（**画面4**）。メモリは、MacOSで使いたい分量だけ割り当てられるので、まともに動かしたければ、かなり大量のメモリが必要になる。デフォルトでは32Mバイトだが、これではOSを起動させただけで終わりだろう。設定は、/etc/molrcというテキストの設定ファイルを編集して行う。

編集部で青白Power Macintosh G3（400MHz）を用意し、Edition 3をインストール後、100Mバイトを割り当てたMOLでMacOS 8.6Jを起動して、Adobe Illustratorを起動して操作してみたところ、十分実用的な速度で、動作することがわかった（640×480の画面でIllustratorを使う気にはならないが）。ショートカットキーも一部X Window Systemが横取りするものを除けば、正しく動作していた。

サウンド、SCSI、リムーバブルメディアなど、まだ未対応の機能も多いが、Power Macintosh用Linuxの可能性を広げるシステムといえよう。

画面4 Mac-on-Linux
GNOMEデスクトップでMacOSが起動している不思議な画面。これは合成ではない。MacOSライクなデスクトップテーマを使ったら、混乱するかも？

Red Hat Linux 6.2Jプロフェッショナル

Red Hat Linux 6.2J プロフェッショナルは、Red Hat,Incが開発した、Red Hat Linux Professional Editionを、レッドハット株式会社が日本語化したサーバ用ディトリビューションである。4月に発売されたワークステーション用のRed Hat Linux 6.2Jデラックスと、インストールCDは共通で、カーネルのバージョンは2.2.14、ライブラリはglibc2.1.3、 Xは XFree86 3.3.6と、システムの基本部分は同じである。プロフェッショナルは、デラックスに表1に示すCD-ROMとガイドを追加した構成になっている。

Red Hat Linux 6.2J プロフェッショナルは、6月2日から販売され、電話で30日間、メールで90日間のサポートが提供される。さらに、正規ユーザーのみ利用可能なFTPサイトへのアクセス権が180日間与えられる。

セキュアWebサーバ

Red Hat Linux 6.2J プロフェッショナルの最大の特徴は、セキュアWebサーバのバンドルだ。インターネット上での電子商取引は、サーバ市場で今もっともホットな分野のひとつだが、個人データなどを扱うため、システムには強固なセキュリティが求められる。このセキュアWebサーバは、セキュリティ面で安全なWebサーバを構築するために、128ビット暗号技術を用いたWebサーバアプリケーションだ。128ビットの暗号技術は、米国の輸出規制で制限されているが、開発場所をヨーロッパへ移すことにより、アメリカ以外の国でも利用可能となっている。

データベース開発のプラットホーム

Red Hat Linuxは最新バージョンの6.2から、ハードディスクに直接アクセスする機能のRaw I/O、複数のマシンを1台のマシンとして運用するクラスタリング、4Gバイトまでの物理メモリの使用といった先進的な機能を装備している。

また、Red Hat Linux 6.2J プロフェッショナルには、IBMのリレーショナルデータベースシステムDB2（英語版）や、データベース開発環境のdbMAGIC version 8（評価版）がバンドルされ、これらのツールと、Red Hat Linuxをあわせて使うことにより、業務用途にも十分耐えうるデータベースシステムの開発と運用が可能である。

サーバ用アプリケーションCD	サードパーティ製のものを含む
Secure Web Server CD	セキュリティ面で堅牢なWebサーバ
CPAN収録CD	Perlのスクリプト集
IBM DB2開発版（英語版）CD	リレーショナルデータベース
Secure Web Server インストールガイド	

表1　プロフェッショナルで追加された収録物

IBM DB2 開発版（英語版）	リレーショナルデータベース
HDE Linux Controller	サーバ管理ソフト
ATOK12SE	日本語入力プログラム
Wnn6 ver.3	日本語入力プログラム
dp/NOTE for Linux/BSD Ver.2.01	日本語ワープロ
System Commander Lite	OSブートマネージャ
DynaFont 5書体	商用フォント

表2　バンドルされる主な商用アプリケーション

本誌付録CD-ROMに収録されているのは、Red Hat Linux 6.2Jの<FTP版>です。この記事で紹介している、Red Hat Linux 6.2J プロフェッショナルにバンドルされる商用ソフトウェアは含まれていません。また、レッドハット株式会社からサポートを受けることはできません。

エンタープライズという道

これまでのLinuxは、Webサーバとしては高いシェアを誇っていたものの、データベースやネットワーク基幹システムという、エンタープライズ分野では、先達のUNIXに一歩遅れをとっていた。しかし、IBMが自社サーバマシンにLinuxを本格的に採用するなど、Linux環境も着々とエンタープライズ分野への対応を強めているようだ。こういった意味で、信頼性の高いデータベースシステムに求められる先進機能で武装したRed Hat Linux 6.2J プロフェッショナルは、UNIXの牙城へ切り込む急先鋒となるだろう。

製品名　Red Hat Linux 6.2J プロフェッショナル
価格　2万9800円
問い合わせ先　レッドハット株式会社
03-3257-0411
http://www.redhat.com/jp/

Distribution ▶▶▶

プロサーバLinux Ver3.0発売

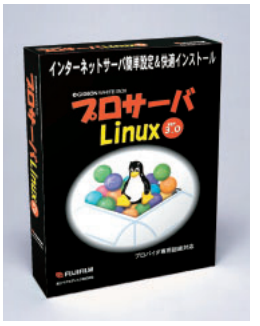
富士マグネディスクから、サーバ専用パッケージソフト「プロサーバLinux Ver3.0」が6月2日に発売された。価格は1万9800円で、ブータブルCD-ROMと日本語マニュアルが含まれる。サポートは、メールにより2件までが無償で提供される。

カーネル2.2.14、XFree86 3.3.6の採用や、Sendmail、Bind、Apacheなどのサーバアプリケーションも最新版を取り入れた。

プロバイダ固有の接続方式に各種対応し、OCNエコノミー専用回線、ODN、DION、そのほかのプロバイダ専用回線に接続可能だ。CATV回線のルータとしても利用できる。

GUIで各種の設定を行う管理メニューに、LAN環境での特定フォルダ作成と、アクセス権限の設定機能を追加し、Windowsファイル共用フォルダへのアクセスを、難しいコマンドを使わずに設定できる。

Webプロキシサーバによって、ホームページへのアクセスが、より速く快適になった。また、タイムサーバによるサーバ時刻調整機能の追加も行われた。



富士マグネディスク（http://www.fujifilm.co.jp/fmd/）

Kondara MNU/Linux Server 1.1発売

デジタルファクトリは、SOHOなどのサーバ向けディストリビューション「Kondara MNU/Linux Server 1.1」を6月8日から発売する。価格は通常のパッケージが2万4800円、アカデミックパックが1万9800円。ユーザー登録から90日間に5インシデント、インストールからサーバ設定までの範囲で、電話、FAX、メールによる24時間365日対応のサポートが含まれる。PC/AT互換機のほかに、Alpha CPUにも対応する。

サーバの各種設定作業を容易にするGUIツールとして、独自に開発した「mph」ツール群を利用することで、NFSやDNS、Sendmailの設定、Apacheのアクセス制限、各種サーバサービス機能設定、メール、FTP、エイリアスのユーザー設定、ファイアウォール構築、カーネルの再構築、バックアップなどをGUIで行うことができる。

また、有償サポートとして、インシデントパック（3インシデント、年間15万円から）や管理者アウトソーシングサービス（年間基本サポート200万円）も用意されている。

デジタルファクトリ（http://www.digitalfactory.co.jp/）

TurboLinux Server日本語版6.1、6月16日発売

ターボリナックス ジャパンは、サーバ向けディストリビューション「TurboLinux Server日本語版6.1」を6月16日から発売する。価格は3万9800円で、アカデミック版は2万4800円。サポートは、インストールからサーバ設定までの範囲で、電話、Web、メールによる90日間3インシデントが含まれている。

主に、セキュリティの強化やデータベース使用時のパフォーマンス向上などが行なわれ、特にセキュリティ面の強化のために米RSA Security社と提携、SSL（Secure Sockets Layer）の開発ライブラリ「RSA BSAFE SSL-C」を収録した。

カーネル2.2.15、glibc 2.1.3、XFree86 3.3.6を採用。最大4Gバイトメモリ、RAW I/O、Async I/O、I2O（Intelligent I/O）をサポートしている。

インストーラには「AutoCruise」という新機能が加わり、ISPなどで同一構成のサーバを複数構築することが簡単になった。

「セキュリティホール通知サービス」の提供と、年4回の最新アップデートモジュール「メンテナンスCD」も送付される。

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

LASER5 Linux 6.2、6月30日発売予定

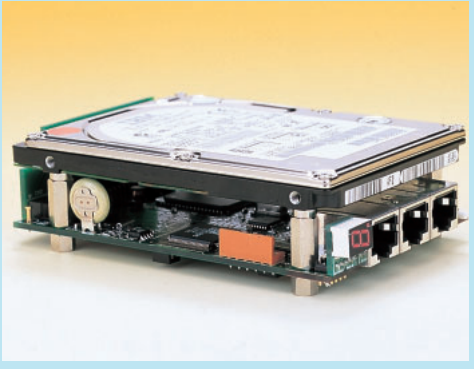
レーザーファイブは、LASER5 Linux 6.0 Rel.2の次期バージョン「LASER5 Linux 6.2」を6月30日に発売する。カーネルは2.2.14、XFree86は3.3.6、GNOME、KDEを採用したデスクトップ向けディストリビューションである。価格は未定。

LASER5 Linux 6.2には、米ThinkFree.com社が開発したJavaベースのオフィススイート「ThinkFree Office Linux」（日本語対応版）をバンドルする。ThinkFree Officeは、ワードプロセッサ、表計算、プレゼンテーション・グラフィックス、オンラインファイル管理機能を持っていて、Microsoft Officeと互換性がある。

レーザーファイブ（http://www.laser5.co.jp/）

Products

手のひらサイズの組込用超小型のLinuxサーバ



FutureNet eServer-200

CPU、メモリ、ハードディスク、ネットワークインターフェイスが揃っていれば立派なコンピュータである。はがきの半分以下のサイズにそれを凝縮して、Linuxが動作する本機は、さまざまな応用ができるベースシステムとして大きな可能性を秘めている。

製品名	FutureNet eServer-200
価格	5万9800円～9万9800円（サンプル価格）
問い合わせ先	センチュリー・システムズ株式会社 TEL 0422-37-8911 http://www.centurysys.co.jp/

FutureNet eServer-200は、センチュリーシステムズが開発した組み込み用の超小型Linuxサーバである。

10cm×7cmというコンパクトなメインボードだけでも動作可能だが、ほぼ同じ大きさの2.5インチハードディスクや（**写真1**）、PCMCIAインターフェイスボード（**写真2、3**）、コンパクトフラッシュインターフェイスボードを接続することで応用が広がるだろう。

現在は、OEM向けなので基板がむき出しのままだが、ケースに入れた「ディストリビューション版」も開発を進めており、7月より販売の予定となっている。なお、個人でもサンプル品を購入することが可能だ。

CPUは組み込み用PowerPC

メインボードのCPUは、MotorolaのPowerPC 860Tという、周辺装置コントローラを統合したマイクロプロセッサが使われていて、50MHzクロックで動作させている。4MバイトのFlash ROMと16MバイトのRAM、RS-232C、ハードディスクインターフェイスを搭載し、ネットワークインターフェイスは、10／100BASE-TXを1ポートと10BASE-Tを1ポートの合計2ポートが標準装備されている。RS-232Cインターフェイスはネットワーク用と同じRJ-45ソケットが使われている。電源は5VのACアダプタから供給し、7セ

写真1　FutureNet eServer-200
2.5インチのハードディスクと同じ大きさで、CD-ROMと比べると小ささがよくわかる。

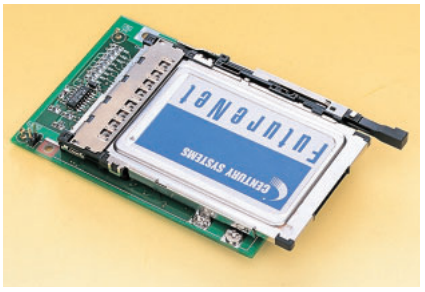


写真2　PCMCIAインターフェイスボード
このほかにコンパクトフラッシュインターフェイスボードも用意されている。

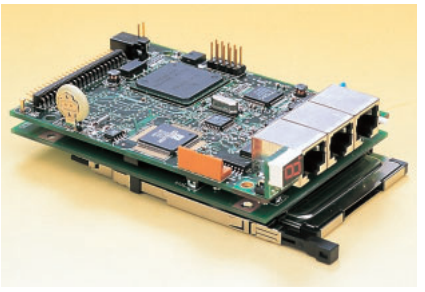


写真3　PCMCIAインターフェイスとの組み合わせ
ハードディスクレスで利用することも可能。メインボードに接続してFlash ROMカードを使うことができる。

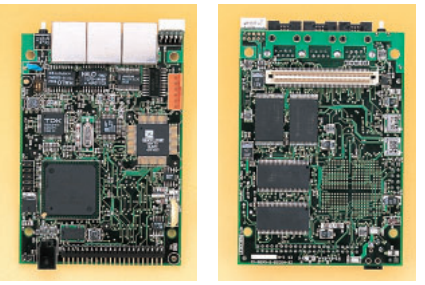


写真4　メインボード
CPUは組込用ワンチップPowerPC 860T。この1枚にすべての機能が凝縮されている。

グメントのLEDで各種ステータスを表示する（**写真4**）。

OSは、LinuxかiTRON準拠のリアルタイムOSを選択することができ、Flash ROMに書き込まれて出荷される。Linuxは、LinuxPPCのサブセットとなっていて、接続するストレージの違いでインストールされるサーバソフトウェアは変わってくる。メインボードだけで動作させる場合には、Flash ROMの容量によって機能が制限されるのはしかたがないところだろう。

OSのバージョンアップといったファームウェアの更新は、ユーティリティを使ってLAN経由で行うことができる。

Linuxならではの豊富な機能

Webサーバ（Apache）、メール、DNS／DHCP、FTP、NTPタイムサーバ、ダイヤルアップサーバなど豊富な機能を実現でき、ハードディスクを付ければNFS／Samba／Netatalkによって、UNIX／Windows／Macintoshのファイルサーバとして利用できる。

また、2つのネットワークポートを生かして、ファイアウォールやIPマスカレードといったルータ機能を持たすのもよいだろう。

今回試用したeServer-200は、6Gバイトのハードディスク付きで、デモ用に調整されたものだ。Webサーバ、FTPサーバなど一部の機能しかなかったため、telnetでログインしたり、ブラウザでホームページを見てみたりと、短い時間使っただけだがレスポンスは悪くない。ファンもなく、ノートPC用のハードディスクなのでシーク音もかすかに聞こえる程度で、ケースに入れたらまったくの無音だと思われる。

ユーザーアプリケーションの開発

製品版では、ネットワークやシステム設定、サーバ機能の動作、サービス稼働状況監視などが、すべてWebブラウザからコントロールできるようになるということだ。

eServerには、キーボード、マウス、ディスプレイを接続することができないため、ネットワーク経由で操作するためには、ネットワークアプライアンスと呼ばれている特定用途向け製品と同じように、Webブラウザでの操作は必須といえよう。

アプリケーションの開発は、Linux PPCが動作するMacintoshでコンパイルするのが簡単だが、PC上のクロスコンパイラを利用することもできるし、ハードディスク付きのモデルならeServer上でセルフコンパイルも可能だ（ただし、RAMを32Mバイトに増設が必要）。

応用製品としてはいろいろ考えられるが、PCMCIAインターフェイスを利用して、無線LANカードやPHSデータ通信カードを使ってアクセスポイントにしたり、SOHOや小規模オフィスなどでインターネットとのファイアウォール、Webサーバ、メールサーバとして利用したりするのがよいのではないだろうか。小型、無音、低消費電力という利点は家庭で使う場合にも向いていると思われる。

型番	FutureNet eServer-200
CPU	PowerPC 860T 50MHz
ROM（Flash）	4Mバイト
RAM	16Mバイト（最大32Mバイト）
シリアルポート	RS-232×1ポート
イーサネット	10／100BASE-T×1ポート 10BASE-T×1ポート
ディスクインターフェイス	IDEインターフェイス×1ポート
LED	7セグメントLED（電源、ステータス表示）
OS	LinuxまたはリアルタイムOS
ストレージオプション（システム、データ用）	2.5インチハードディスク（6Gバイト） PCMCIA Flash ROMカード（最大1枚） コンパクトフラッシュカード（最大2枚）
基板サイズ	100mm×70mm（突起物を除く）
外形寸法（mm）	120（W）×84（D）×50（H）
最大消費電流	2.5A（DC 5V、起動時）

表1　FutureNet eServer-200の主な仕様

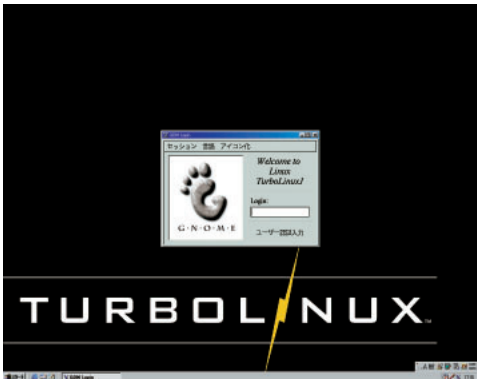
ASTEC-X 3.00

ASTEC-XはWindows用のXサーバソフトだ。X Window Systemをネットワーク接続したWindowsマシン上で使うことができる。Xをコンソールの呪縛から解き放ち、telnetでは実行できないGUIのアプリケーションが使えるスグレモノだ。

製品名	ASTEC-X
価格	7万8000円（個人利用の場合、3万9000円）
問い合わせ先	株式会社アステック・プロダクツ TEL 03-5804-1853 http://www.astec.co.jp/

Windows用PC Xサーバとして実績のあるASTEC-Xが3.00にバージョンアップした。PC XサーバはWindows上でX Window Systemを実行するためのソフトである。WindowsとLinuxの両方を使っているような場合に非常に便利なソフトウェアである。

今までPC Xサーバに触れたことのない人は、どんなものなのかちょっと想像がつかないかもしれない。Windows上でXのアプリケーションを動かすという感覚がつかめないだろう。まずは**画面1**を見てほしい。見慣れたGNOMEのログイン画面である。これはgdmというグラフィカルログインの画面だ。しかしよく見ると、下のほうにWindowsのスタートメニューとタスクバーが表示されているのがわかる。実はこれはLinuxのコンソールに表示されたgdmではなく、Windows上に表示されたgdmだ。もちろんログインすることができる。



画面1　グラフィカルログイン画面
よく見るとWindows上でgdmのログイン画面が表示されているのが分かる。下に見えるのはWindowsのスタートメニューとタスクバー。

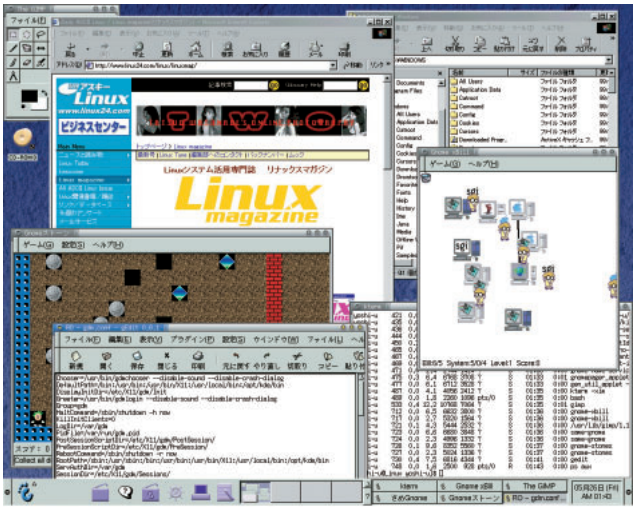
Windows上でXアプリケーション

次に**画面2**を見てほしい。これも見慣れたGNOMEのディスクトップだ。いくつかのウィンドウが表示されている。しかしよく見ると、Internet Explorerとエクスプローラのウィンドウが表示されているのがわかる。これは当然Windowsのアプリケーションである。このように、WindowsのアプリケーションとGNOMEのアプリケーションをシームレスに実行できるのがPC Xサーバの素晴らしい機能だ。

画面3は、Xの壁紙を隠してWindowsの壁紙を表示させた状態だ。この画面を見ればWindows上でXのアプリケーションが動いていることがよくわかる。もちろん、XサーバなのでGNOMEだけではなく、AfterStepやKDEを使うこともできる（**画面4**）。

ASTEC-XはWindows上でLinuxを動かしているわけではない。この画面はネットワーク上の別のマシンのLinuxにログインしているのだ。そのため、Xで動かしているアプリケーションのマシンパワーは、リモートログインしているマシンのものを使用する（Windows上での描画は別だが）。

Windows上のXサーバはいくつも存在しているが、特定のウィンドウ内で使用するものも多く、ASTEC-Xのようにマルチウィンドウで表示できるものは少ない。まして、日本語が使えて、実用に耐えられる実行速



画面2　GNOMEデスクトップ
左上のウィンドウはInternet Explorerで、右上のウィンドウはエクスプローラだ。GNOMEのデスクトップとシームレスに実行されているのがわかる。

度のものとなると探すのが難しくなる。ASTEC-Xは他のPC Xサーバと比べても最高峰に属しているといえるだろう。

どんなときに便利なのか

今となってはX Window Systemを使わずにLinuxを使っている人は少ないだろう。多くのアプリケーションはユーザーインターフェイスがGUIで提供されているし、実際GUIのほうが使いやすいと感じている人も多いだろう。しかも、コンソールでログインできない場合などはtelnetでログインすることになるが、これではGUIのソフトが使えない。また、場合によってはWindowsのアプリケーションを使わなければならない場合もあるだろう。そんなときにPC Xサーバを使えば1台のマシン上でWindowsとX Window Systemの両方を使うことができる。

どこが変わったのか

旧バージョンのASTEC-Xを使用していたなら、バージョン3.00になってどこが変わったのか気になるところだろう。主な変更点は次のようなものだ。

- **カラーエミュレーションの強化**
- **コンソールエミュレーションの強化**
- **設定ファイルの読み書き機能**
- **設定アシスタント**
- **X11R6.4対応**
- **XDMCP chooser機能**
- **高速化**
- **Xlaunchの改良**

特に、今回追加されたXDMCP chooser機能を使うことで、複数のLinuxマシンをメニューから選択することができるようになった。複数のマシンを使用してる場合に非常に便利だ。

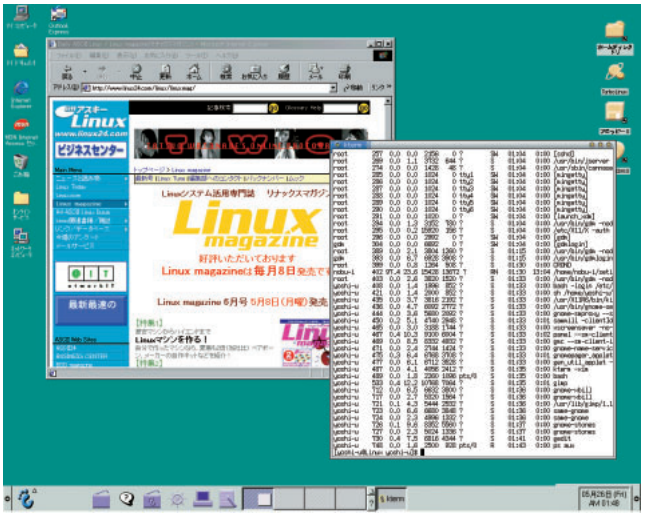
ASTEC-Xを動かす

最近のディストリビューションはセキュリティが厳しくなっており、ネットワークからのログインができないように設定されているものもある。このようなディストリビューションでASTEC-Xを使う場合は、設定ファイルの変更が必要になる。たとえば、GNOMEのgdmを使用する場合は、/etc/X11/gdm/gdm.confを変更する必要がある。変更する箇所は[xdmcp]セクションの「enable=0」を「enable=1」に変更する。もちろん、ASTEC-Xを実行するマシンからリモートログインできるように/etc/hosts.allowファイルを変更する必要もある。

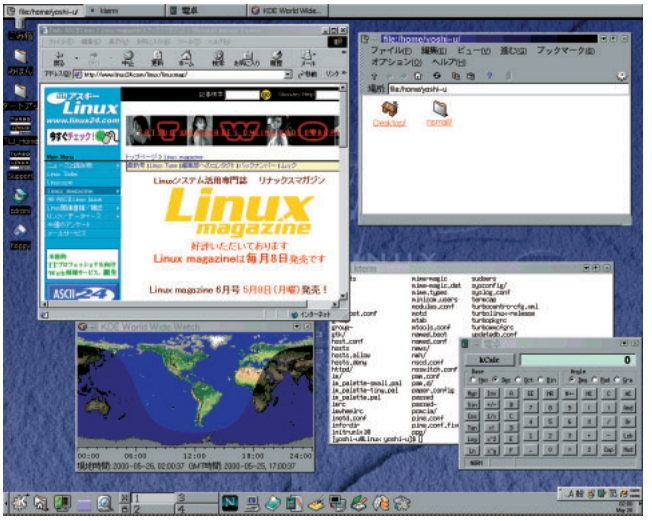
これでもログインできない場合は、[daemon]セクションの「KillInitClients=1」を「KillInitClients=0」に変更してみてほしい。

ASTEC-Xは非常に便利なソフトウェアで、一度使うと手放せなくなってしまう。WindowsのIMEをXのアプリケーションで使うこともできる。速度も申し分なく、ほとんどコンソールでログインしているのと変わらない。付録CD-ROMに体験版が収録されているので、ぜひ使ってみてほしい。

ただし便利なソフトである反面、価格がやや高価である。1ライセンス7万8000円（個人使用の場合は3万9000円）と気軽に手を出せる金額ではない。もう少し気軽に買える価格設定にしてほしいところだ。



画面3　Windowsの壁紙を表示
Xの壁紙を隠してWindowsの壁紙を表示してみた。Windows上でXのアプリケーションが動いているのがわかりやすいだろう。



画面4　KDEのデスクトップ
ASTEC-XはXサーバなので、KDEやAfterStepなどのデスクトップも表示できる。もちろん、SunやFreeBSDなどでも使用できる。

特報！
Linux World Expo/Tokyo 2000 開催

5月11、12日の両日、有明の東京ビッグサイトにおいて、LinuxWorld Expo/Tokyo 2000が開催された。「IT革命の中心にLinuxがある」というテーマが掲げられた今回は、85社が展示会に出展し、2日間で2万4000人を超える来場者を記録した。会場はLinuxの発展と普及を願う多くの人々の熱気につつまれていた。

この記事が掲載される頃には、開催からすでに1カ月近く経過していることになる。本誌の読者の方なら、すでに関連の情報を他のメディアを通じてご存じのことだろう。そこで、来場できなかった読者のために、できるだけイベント会場の生の雰囲気を伝えることにしよう。

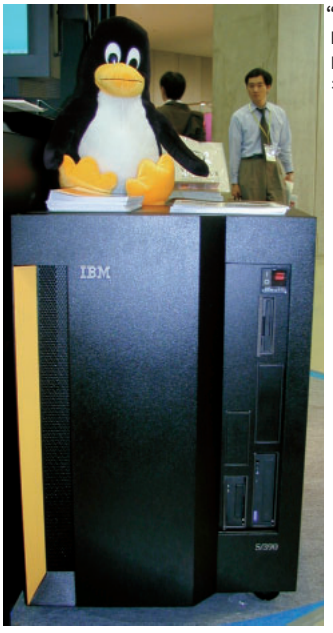
4月中に発表されていたとはいえ、データベース界の巨人Oracleと、世界有数のLinuxディストリビューターTurboLinuxのジョイントによる「ミラクル・リナックス」は大きな注目を集めていた。ミラクル・リナックス株式会社は、日本オラクルが中心となって、ターボリナックスジャパン、日本電気（NEC）、オービックビジネスコンサルタント（OBC）などの出資を受けて設立されたLinux事業会社である。

注目の基調講演

基調講演では、ミラクル・リナックスの代表取締役社長に就任する矢野広一氏が、設立の経緯と抱負を述べた。強調されたのは、同社の実績に裏付けられた技術とオープンソースコミュニティで企業が果たす役割である。日本オラクルのデータベース技術、ターボリナックスジャパンのOS技術、NECのPCサーバ技術を結集して、企業が安心して使うことのできるデータベース処理に最適化された日本語ディストリビューション（Miracle Linux）を誕生させるという目標が示された。

ほぼ同時刻に行われたターボリナックスジャパンの記者発表会場では、同社の小島國照社長が「Miracle Linuxは100%TruboLinuxとなる」と語った。

今回はラックマウントサーバの出展が目立った。写真は、各社の1Uサイズサーバマシン（フォースのGK-21、ノーザンライツのNL Server 1100、VA Linux SystemsのVA Linux 1000 1U、コンパックのAlphaServer DS10L）。



“ Penguin on S/390 ”というわけで、IBMのブースでは同社の汎用機S/390でLinuxが動作していた。アパッチとDB2を組み合わせたストリーミングをデモ。

コンパックはAlphaマシンでクラスタシステムをデモ。Linuxのハイエンドサーバ用途での利用は、今後ますます加速していきそうだ。

マーケットにおいてはライバル関係になることに触れ、「競争は歓迎する」という発言も飛び出した。技術を共有しながらも、ビジネス面では互いに新しい可能性にチャレンジしていくということなのだろう。

いっそう強まったビジネス色

今回で3回目となるLinuxWorld Expoだが（第1回はLinuxWorld Conference Japan '99として開催）、回を重ねるごとに規模が拡大し、同時にビジネス色も強まってきている。テーマとしてIT革命が据えられた今回は、本格的な企業レベルでの導入に向けて、各社とも力の入った展示内容だった。

会場入りして、まず目についたのが、クラスタリング技術を用いた大規模サーバシステムのデモンストレーションだ。ラックマウントタイプのマシンがずらりと勢ぞろいし、来場者の目を引いていた。熱心に担当者に話を聞く背広姿のビジネスマン風の人もいて、導入側の企業でもLinuxに対する関心が高まっていることをうかがわせた。

NEC、IBM、Compaqなど、大手コンピュータメーカーに混じって、日本進出を発表した米国のVA Linux Systems、国産ハードウェアベンダーであるノーザンライツやアダムネットなどもラックマウントタイプのサーバマシンを出展していた。

そんな中で異彩を放っていたのが、IBMのS/390だ。IBMのメインフレームマシンであるS/390上でLinuxが動作するという事実にまず驚かされた。S/390には、OS上で複数のバーチャルマシンを動作させる機能があり、そのバーチャルマシン上でLinuxが動作する仕組みになっているらしい。メインフレームの堅牢性や高可用性を活かしながら、Linuxの持つ柔軟性や豊富なソフトウェア資産といった利点も享受できる。会場では、S/390でOS/390とLinuxを並行稼動させ、Linux上で動作するApacheとOS/390上で動作するDB2との連携によるストリーミング配信のモデルをデモンストレーションしていた。

Linuxのこれからを担う主役たち

最近では一般メディアにも取り上げられるようになり、オープンソースによるソフトウェア開発が新しいビジネ

日本進出を発表したVA Linuxのブース。ラックマウントサーバマシンを中心に出展。今後の事業展開が期待される。

各ディストリビュータとも大型スクリーンを使ったデモで、多くの来場者を集めていた。会場内が熱気に包まれる。

Red Hat, Inc.のCEO、ボブ・ヤング氏もビデオで登場。

ディストリビュータ界の元気者、Kondaraペンギンは会場でも存在をアピール。住み慣れたデスクトップを離れ、大型スクリーンで大暴れしていた。

テンアートニのブースで配られていた「オープンソース」。同社の外食チェーン向け受発注システム「セルベッサ」もオープンソースソフトウェアだ。

スモデルとして注目を集めている。その代表格ともいえるのがRed HatとTurboLinuxだろう。

会場内では、最大級のブースを用意して大型スクリーンによるインストラクションを繰り広げたTurboLinuxが目立っていたが、Red Hatのブースも多くの来場者を集めていた。また、出展されたハードウェアには、「Powered by TurboLinux」の文字や、おなじみの「Red Hat Man」のステッカーが貼られたものが多く、ここでも、お互いの存在を主張し合っていた。

ほかのディストリビューターも、それぞれのブースでアピールを行った。Kondara MNU/Linuxの販売元デジタルファクトリは6月8日にリリース予定のKondara MNU/Linux Server 1.1をAlphaマシンでデモンストレーション。レーザーファイブも6月発売予定のLASER5 Linux 6.2を中心に、クラスタリングシステム、クレジットカード大の超小型マシンなどを参考出展して注目を集めた。

日本進出を予定しているOpenLinuxのCaldera Systems、Storm Linux 2000のStormix Technologiesといった海外ディストリビューション企業も参加して、その存在を将来のユーザーたちに示していた。

SOHO向けマシンも多数出展

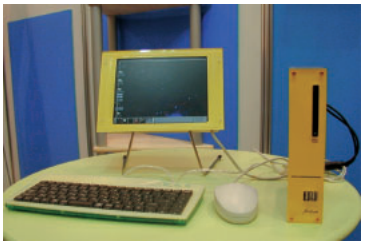
クラスターシステムやメインフレームと好対象をなしていたのが、SOHOや個人ユース向けの小型サーバマシンの展示ブースだ。

アクアリウムコンピュータが出展した小型サーバ「Silver Neon」は、少し照明を抑えたブース内で半透明の筐体が淡い光を放つ姿がとても印象的だった。もちろん、SOHO向けサーバとして十分な機能も備えている。CPUはCeleron 533MHz、128Mバイト（または256Mバイト。購入時に選択）のメモリを搭載し、20.4GバイトのHDD、内蔵UPS機能などを備える。TurboLinuxとRed Hat Linuxのプリインストールモデルがラインアップされる予定だ。

このほかにも、アダムネットのファイアウォール専用マシンFIREBOX SOHO、フォースのGK-21、サイバネテックのPathNavigatorなどの個性的なマシンが多数展示されていた。これらの元祖的存在であるCobalt Qubeも姿を見せていた。

アクアリウムコンピュータのSOHO向け小型サーバマシンsliver neon。中央部にある2つのLEDは、上がサーバの状態監視用、下の3色に点滅しているほうがプロセス監視用。

富士通のブースに出展されていた、サイバネテックの小型サーバマシン。サイズは20cm×3.9cm×15.5cmと、かなり小型で、底面の磁石で垂直面にも設置可能。



新規参入メーカーの出展も。写真はピノーのFolium。8.5インチの専用モニタもイカしている。Linuxをプリインストールして今夏にも発売予定。

ついにキャンギャルが！　華やかさを演出した彼女たちもイベントの盛り上げにひと役かっていた。

サーバ用途ではないが、コンパクトでキュートなマシンを発見した。ピノーが出展したFoliumとTruncusがそれだ。特にFoliumは黄色のフロントパネルが特徴的で、専用の8.5インチ液晶ディスプレイを合わせると、とてもインテリア性が高くおしゃれなイメージがある。Foliumのスペックを紹介しておこう。CPUにはNational SemiconductorのGeode GXLV 266MHzを採用し、64Mバイトのメモリを搭載。4.3Gバイトの2.5インチHDD、USB 2ポート、PCカード、さらにモデムまで内蔵している。今夏にも発売の予定だ。

ニューカマー

SGIといえば、ハリウッド映画でも使われているというハイクオリティCGなどの高度なグラフィックス処理環境を提供するプラットフォームとして有名だ。これまで、独自のUNIX系OSであるIRIXを採用してきたが、今後Linuxへのシフトを強めることを表明している。同時にハードウェアに関しても、Intelアーキテクチャを取り入れた新しいシリーズ展開を始めている。今回は、Intelアーキテクチャを採用したワークステーションマシンとクラスタリングシステムを組み込んだラックマウントサーバを展示していた。

Linuxとハードウェアだけでなく、アプリケーションの出展が目立ったのも今回の特徴だ。従来、Linuxが利用されてきたサーバ用途向けのツールやアプリケーションだけでなく、グループウェア、ワープロなどのビジネス用途向けの製品も登場してきた。

なかでは、ThinkFree.comのオフィススイート「ThinkFree Office」、HancomLinuxの多国語対応ワープロ「アレアハングル」、日本語化はまだだが、Windows版の評価が高いCorelの「WordPerfect for Linux」が注目だ。

冒頭でも紹介したように、ミラクル・リナックスに出資しているOBCもテレビCMでおなじみの「〜奉行」のLinux版を発表している。

●

Linuxを取り巻く環境は、ここ1年ほどで急激な変化を遂げた。IT産業にかかわる各企業がLinuxへの支援を次々と表明した反面、「ブームにすぎない」とする冷ややかな見方もある。ただ、今回の盛り上がりを見る限り、ビジネスシーンへのLinuxの浸透は当分やみそうもない印象を強く受けた。

ディストリビューション 戦国時代

その誕生以来、一貫してオープンソース／フリーソフトウェアを標榜してきたLinuxだが、相次ぐ商用ディストリビューションの登場により、いやがおうにもマーケットという戦場へと駆り立てられている。そこは厳しい競争原理のみが支配する世界。生き残るのは誰だ！

illustration : Aki with Tsubochi

ディストリビューションの現在

2000年に入ってから相次いだリリースラッシュも一段落し、各ディストリビューションがほぼ出揃った感がある。次期リリースはカーネル2.4を睨んで、ということになるだろう。

そこで今回の特集では、今年リリースされた日本語ディストリビューションの評価と今後の動向を、ディストリビューション選びのポイントなどをまじえながら展望していくことにする。

基本コンポーネントのバージョン

まずは、Linuxのコア部分について検討していこう。基本コンポーネントのバージョンは、サウンドやグラフィックス機能などのハードウェアのサポートに関連する重要なポイントだ。現状では、リリース次期によるばらつきはあるが、どのディストリビューションも

●カーネル：2.2.13または2.2.14
●Cライブラリ：2.1.X*1
●XFree86*2：3.3.5または3.3.6

でほぼ揃っている（64ページの機能一覧を参照）。この部分では大きな差異はないと考えてよい。また、各ディストリビューターとも、検証作業が済みしだい、新しいバージョーンのものをFTPサイトなどを通じて順次公開していくはずだ。

公開（配布）時期は、そのスタンスによって異なってくると予想される。新しいバージョンを積極的に取り入れていくディストリビューターもあれば、現行リリースの安定性を重視する方針のところもあるからだ。この点は、ディストリビューションを選ぶ際のポイントのひとつになるだろう。この点については、のちほどもう一度触れることにする。

RedHat系がメインストリームを形成

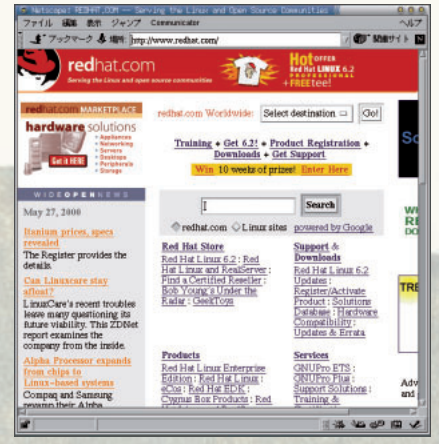
Linuxのディストリビューションは、次の3つの系列に大別される。

●Red Hat系
●Debian系
●Slackware系

世界中にはさまざまなディストリビューションが存在し、これら3つの系統に属さないパッケージもあるが、現在、多くのユーザーを集めているディストリビューションのほとんどは3系統のいずれかに属している。日本国内では、さらにその傾向が強い。

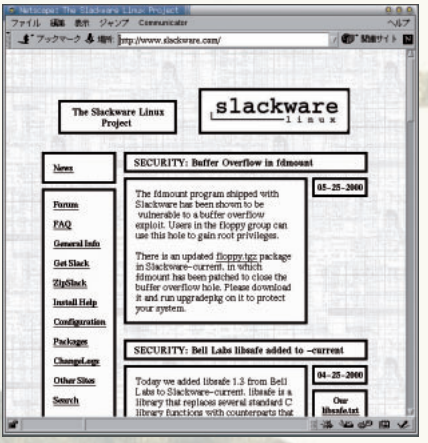
3つの系列は、おもにソフトウェアのバイナリパッケージの管理方式の違いで判別できる。バイナリパッケージによるインストールは、ソースコードからのコンパイルに比べて必要とされる知識が少なくて済む。ただし、あるパッケージをインストールし、正常に動作させるには、別のパッケージが必要となる、といった依存関係が存在する。この依存関係を管理して、ユーザーが適切にパッケージを導入できるように支援する仕組みが各系列で異なっているのだ。

Red Hat系のディストリビューションは、Red Hat, Incが開発したRPM形式という管理システムを採用してい



画面1　Red Hat, IncのWebサイト

画面2　DebianプロジェクトのWebサイト



画面3　SlackwareのWebサイト

る。Debian系には、Debianプロジェクトが開発したdeb形式のパッケージを管理するためのツールが用意されている。Slackwareは、パッケージを管理するための仕組みを持たない。

今回取り上げるディストリビューションは、Plamo 2.0（Slackware系）を除いて、すべてRed Hat系となった。他系列のディストリビューションを大きくリードし、Red Hat系が中心的な役割を果たしているといってよい。

Linuxの配布形態は、パッケージ販売、FTPサイト、雑誌・書籍の添付CD-ROMと多岐にわたっているため、インストールベースでのシェアを正確に把握することはできないが、当誌で行っている読者アンケートの結果（**図1**）などを見ても、Red Hat系の優位は確かなようだ。

これは、入手のしやすさ、商用パッケージを含めたバリエーションの豊富さ、RPMによるパッケージ管理の普及度などに起因していると思われる。特に、RPM形式によるバイナリパッケージの配布は、ソースからのコンパイルに自信のない（あるいはそれを好まない）ユーザー層に支持される大きな要因となっているはずだ。ゲームやドライバモジュールなどの配布元でも、RPM形式での配布を優先的に行うところが増えてきている。

ただし、ひとくちにRed Hat系といっても、あくまで開発ベースがRed Hatということであって、それぞれ独自のユーティリティやツール、より新しい機能を取り入れることで、独自性を強めつつある。ユーザーの立場からすると、選択の幅が広がるという利点がある一方で、コマンドやユーザーインターフェイスなどの統一性が損なわれるといったマイナス面があることも否めない。細かいことではあるが、ディストリビューション間の互換性について考えるとき、ユーザーインターフェイスや操作性の問題は無視できない要素である。

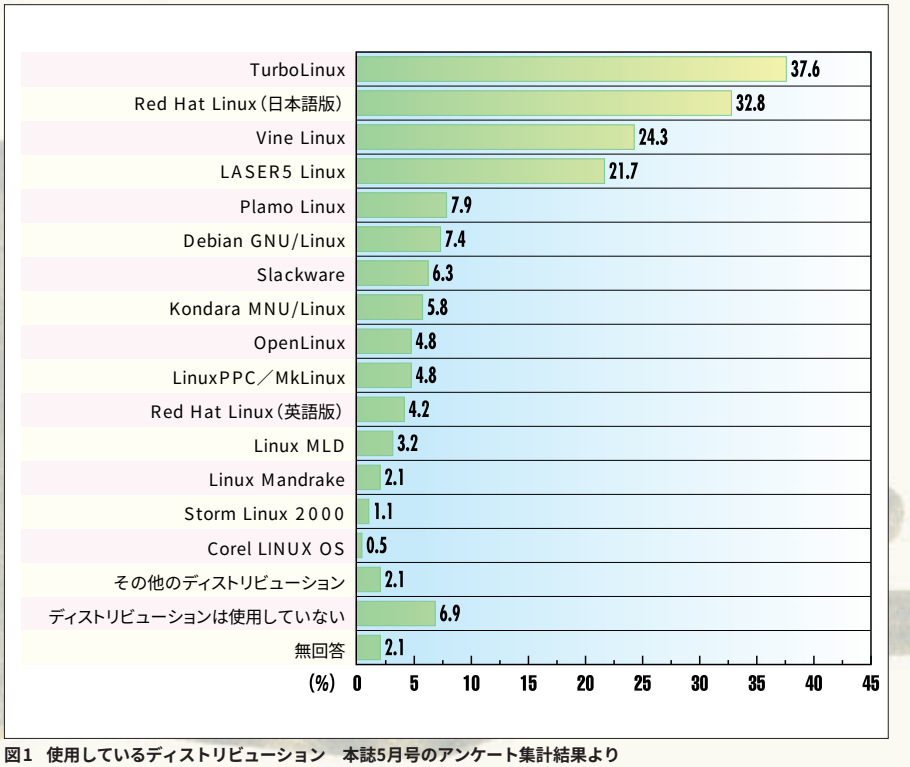


図1　使用しているディストリビューション　本誌5月号のアンケート集計結果より

系　列	ディストリビューション名	公式Webサイト
Red Hat系	OpenLinux	http://www.openlinux.ne.jp/
	SuSE Linux	http://www.suse.com/
	Linux Mandrake	http://www.linux-mandrake.com/en/
Debian系	Debian GNU/Linux	http://www.debian.org/
	Corel LINUX OS	http://linux.corel.com/
	Storm Linux	http://www.stormlinux.com/
Slackware系	Slackware Linux	http://www.slackware.com/

表1　各系統のおもなディストリビューション　※今回、特集としてとりあげたディストリビューションを除く

ユーザーインターフェイスに直接関連する要素として、X Window Systemのウィンドウマネージャとデスクトップ環境があげられる。これも後述の機能一覧を見てもらえばわかるのだが、現在の主流はEnlightment／Sawfish＋GNOME、KDEである。特にデスクトップ環境に関してはGNOMEとKDEがスタンダードの地位を確立しつつあるといってよい。今回紹介するディストリビューションのすべてで、この2つがデフォルトまたは、追加の設定で利用可能となっている。いまのところ、GUI環境の操作性に関して、ディストリビューション間での大きな差異はないといえるだろう。

Glossary

＊1　Cライブラリ
C言語でコーディングされたソフトウェアの作成と実行に必須の「関数」と呼ばれる小さなプログラムの集まり。バージョンによって機能が異なるため、アプリケーションの動作に影響を与える場合がある。

＊2　XFree86
UNIX系のOSでは、ユーザーにGUI環境を提供するための仕組みとしてX Window Systemを採用している。XFree86はフリーのX Window Systemで、ほとんどのLinuxディストリビューションに収録されている。

デスクトップ系？

この特集では、ディストリビューションをデスクトップ系とサーバ系に大別して紹介している。実際には、明確な区別があるわけではなく、どのディストリビューションのどのリリースを用いても、サーバを構築することは可能だ。必要なサービスアプリケーションを入手、インストールして、セキュリティなどの必要な設定を行えば、任意のディストリビューションをベースにサーバを構築できる。

商業ベースで販売されているサーバ向けディストリビューションは、必要なサービスアプリケーションや専用の管理ツールがバンドルされており、デフォルトの設定もサーバ用途に合わせてチューンアップされている。

今回は便宜上、このようにサーバ向けに特化されていないものをデスクトップ系と呼ぶことにする。繰り返しになるが、これらのディストリビューションの用途がデスクトップに限定されるわけではないことに注意してほしい。

今回の評価ポイント

すでに述べたように、基本となるカーネルのバージョンなどについて、各ディストリビューション間に大きな格差はない。現状では機能的に抜きん出たディストリビューションはない、という印象が強いのも事実だ。そのため、それぞれのアピールポイントと、改善が望まれる部分を重点的に評価していくことにする。

各ディストリビューションを紹介するページには、それぞれの特徴をわかりやすく示すために、タイトル部分に「判定チャート」を掲載している。

■先進性

前述した新バージョンや新機能の採用に関するアプローチは、「先進性」として示している。開発バージョンのソフトウェアなど、積極的に新しい機能を取り入れる姿勢を先進性とし、安定バージョンに限定し、採用に際しても一定の猶予期間をみていこうとする姿勢を安定志向として評価した。

■コストパフォーマンス

商用パッケージの値段と、バンドルソフトウェアの充実度を比較して評価している。Plamoについては、商用パ

Column

Linuxとは何か？

「Linux」とは、もともとOSのコア部分であるカーネルに付けられた呼称であるが、現在では、カーネルを中心とするシステム全般を示す言葉として用いられている。

■Linux誕生の経緯

ほとんどの読者はすでにご存じだと思うが、Linuxのカーネルは、当時フィンランド在住の学生だったLinus Torvalds氏が、1991年に開発を始めたプログラムが原点である。このプログラムは、80386に組み込まれたマルチタスク機能をOSとして実装するためのものであった。開発に際してTorvalds氏は、ソースコードをネットニュースを通じて公開し、広く開発への参加を呼びかけた。そして、世界中のプログラマーが開発に参加して改良されていき、現在へと至っている。

■GNUプロジェクト

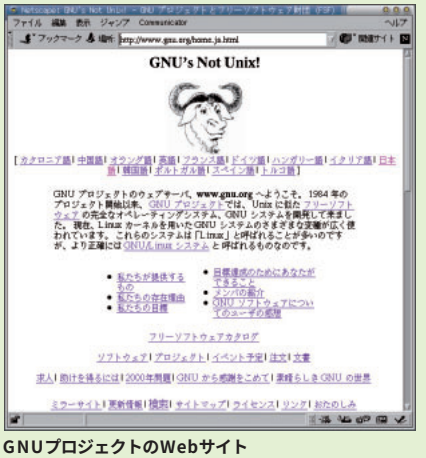
カーネルだけではOSとしても、コンピュータシステムとしても機能しないため、各種のサービスやコマンド、アプリケーションがLinuxに移植された。その多くは、Linuxカーネルと同じくフリーソフトウェアであるGNUプロジェクトが開発したものであった。このジョイントは、Linuxカーネルをコアとするシステムが発展し、広く普及するきっかけとなった。GNU/Linuxと表記されることがあるのは、それがGNUとLinuxカーネルを組み合わせたOSであることを示しているのだ。

LinuxもGNUプロジェクトが策定したフリーソフトウェアのライセンス規約であるGPL（GNU Public Licence）のバージョン2に基づいて配布されている。GPLでは、ソースコードの改変や再配布の自由が認められていると同時に、それを妨げないことを要求している。

■ディストリビューション

前述のとおり、カーネルだけではシステムとして完全ではない。そのため、Linuxカーネルをコアとするコンピューティング環境を構築するには、ほかにも多くのソフトウェアが必要となる。それらのソフトウェアをとりまとめてパッケージングし、ユーザーに配布するための完全なOSのセットが「ディストリビューション」である。

初期のディストリビューションは、フリーソフトウェアのみで構成されていたが、やがて、商用ソフトウェアやサポートなどを含み、商業ベースで流通するディストリビューションも登場するようになった。商用ディストリビューションであっても、フリーソフトウェア部分は、GPLなどのライセンスに準拠してFTPサイトなどを通じて公開されている。



GNUプロジェクトのWebサイト
http://www.gnu.org/

ッケージがないため対象外とした。

■サポート

電話、メールによる直接的なものだけでなく、Webサイトでの情報公開、教育／認定制度によるユーザー、エンジニアの養成活動、ドキュメントの日本語化についても考慮している。

「ユーザーのスキル」については、特に説明の必要はないだろう。

デスクトップ系の場合、ユーザーが選択時に考慮すべきポイントとしては、上記のものを含め、次のようなものがあげられる。

①バンドルされる商用アプリケーション、独自ユーティリティの必要性
②使用するハードウェア環境がサポートされているかどうか
③GUIの使用感、ルック＆フィール

①は、かな漢字変換システム、エディタ、メーラなど、常用するソフトウェアの有無がポイントとなる。②については、公式のサポートである必要はないかもしれない。個人のホームページや雑誌などのメディアで公開されている情報を参照してもよいだろう。当然、公式なサポートを最重視するスタンスも考えられる。③は長期にわたって使用する場合、意外に重要なポイントとなることがある。ただし、これはカスタマイズ可能な項目であるので、デフォルトの設定が好みに合うか、といった程度の問題である。

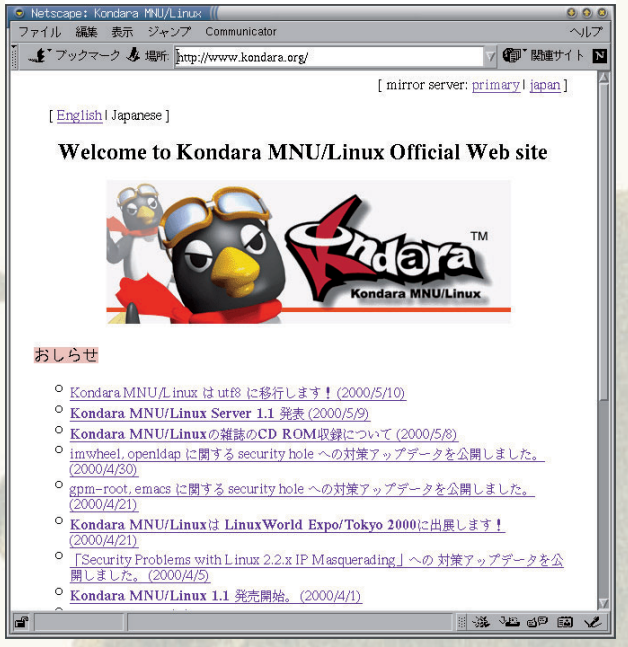
ユーザーとコミュニティ

最後に、Linux（フリーソフトウェア）ならではの選択のポイントをひとつあげておきたい。それは、プロジェクト（およびその成果物）へのシンパシーやリスペクトといったものだ。

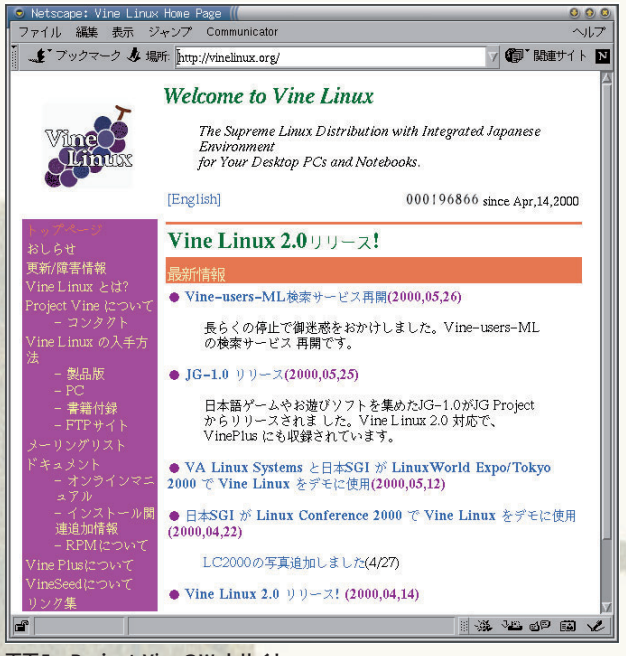
フリーソフトウェアの開発は基本的にボランティアベースである。Linuxディストリビューションの場合も、Plamoのようにまったくの非営利のものと、KondaraやVineのように、開発プロジェクト自体は本来営利を目的としておらず、商用パッケージ製品の販売を企業が担当しているケースがある。Red HatやTurboLinuxなどは、販売を行っている企業が直接開発も行っているが、その成果はコミュニティへと還元されている。

これらの開発プロジェクトへの共感や、その成果物としてのディストリビューションに対する評価として、それを選択（購入）し、使用していくという考え方もあってよいはずだ。

フリーソフトウェアの進展とか、社会的な浸透というのは、運動体としてのプロジェクトだけで成り立つものではない。それを下支えするユーザー層や協賛者が必要でなのである。具体的なフィードバックをもたらさないユーザーであっても、潜在的な支持者という一面を持っているのである。ユーザーの側も、コミュニティを形成するメンバーのひとりとして、その精神や目的といったものを理解するよう努めるようにしたい。



画面4　Kondara ProjectのWebサイト



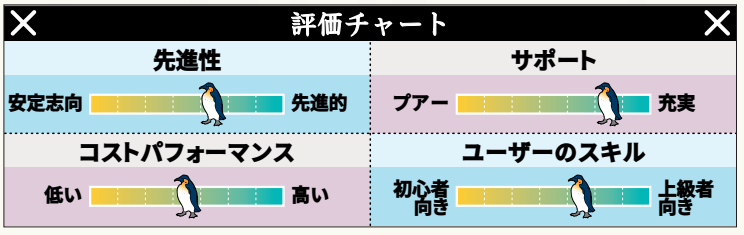
画面5　Project VineのWebサイト

系列ディストリビューションの宗家

Official Red Hat Linux 6.2J

レッドハット株式会社
03-3275-0411
http://www.redhat.com/jp/

スタンダード：3980円
デラックス：1万2800円



「Official」と銘打っているように、米Red Hat, Inc.の開発した「本家」Red Hat Linuxの日本語バージョンである。以前、日本国内での開発と販売は五橋研究所が行っていたが、1999年9月の日本法人の設立に伴い現在はレッドハット株式会社へと移行している。

3つの製品ラインナップ

Red Hat 6.2Jには、「スタンダード」、「デラックス」、「プロフェッショナル」の3つの製品ラインアップが用意されている。このうち、スタンダードとデラックスがデスクトップ系である。デラックスは、**表1**に掲載した商用アプリケーションがバンドルされており、30日間の電話によるインストールサポートと90日間の無料優先アップデートが提供される。

画面1　パーティションレスインストール
インストーラのパーティション選択画面でVFATを選択してインストールできる

「スタンダード」は商用アプリケーションと無料アップグレードを除いた廉価版だ。

新機能はサーバ寄り

前バージョンの6.1改訂日本語版では、カーネルなどの基本的なコンポーネントのバージョンアップは見送られた。6.2Jでは、カーネルが2.2.14、XFree86が3.3.6、Cライブラリがglibc 2.1.3にアップデートされている。

また、以下のような新機能も追加され、機能面での充実が図られている。

- **4Gバイトまでのメモリサポート**
- **パーティションレスインストール**
- **クラスタリング機能**
- **Raw I/Oのサポート**

ただ、デスクトップOSとして見た場合、ユーザーにとって有益といえるのは、パーティションレスインストールくらいだろう。これは、VFATファイルシステムでフォーマットされたパーティションへのインストールをサポートする機能だ。既存のWindows環境を変更する必要がないので、ちょっとLinuxを試してみたいといったときに便利である。

そのほかの新機能は、あくまでサーバOSとしての利用を前提としたもので、個人ユーザーにとって（特にホビーユースの場合）選択の決め手とはならないかもしれない。

デスクトップOSとしての評価は？

6.2Jへのバージョンアップでは、デスクトップ用途に関連する変更点として、ウィンドウマネージャ「Sawfish」（商標登録の問題でSawmillから名称変更された）の採用があげられる。SawfishはGNOMEとの親和性が高く評価されており、動作も軽快である。elispで開発されているので、カスタマイズの自由度も高い。今後、Enlightenmentに代わるウィンドウマネージャとして注目の存在だ。X関連ではMesaのグラフィックライブラリも標準で組み込まれるようになった。また、バージョン6.1で指摘された日本語

かな漢字変換システム	ATOK12 SE for Linux、Wnn6 Ver.3
日本語ワードプロセッサ	dp/NOTE
商用TruTypeフォント	DynaFont和文5書体
その他	System Commander Lite（ブート管理ツール）

表1　Red Hat 6.2Jの製品版にバンドルされるソフトウェア

TrueTypeフォントが表示できない問題も改善されている。

システムサービスに関しても細かな変更が施されている。インストール時に「GNOMEワークステーション」、「KDEワークステーション」のどちらかを選択した場合には、FTP、telnetなどのサーバサービスがインストールされなくなった。また、デフォルトでは、bind、DHCP、sambaなどのサーバサービスも起動しないように設定変更された。

インストール直後のデフォルト状態を制約することで、まずはデスクトップ向けに、より安全に、そして無駄なく使用できるよう配慮したものと思われる。つまり、知らないうちにネットワークサービスを起動してしまって、セキュリティが脅かされたり、余分なシステムリソースを消費することがなくなるわけだ。幅広いユーザー層を想定しているという点で、デスクトップOSとして評価できる。

用途を問わず、うれしいのが「ドキュメントCD」の添付である。このCD-ROMには、Red Hatのオフィシャルマニュアルである「インストールガイド」、「リファレンスガイド」とともに、JFやFMによる「HOWTO」や「FAQ」といったLinux関連の日本語ドキュメントが多数収録されている。設定に困ったときなどは、まずこのCD-ROMを調べるとよい。たいへん便利に使えるアイテムである。

中級者以上にお勧め

インストールに関して気になる点がある。バンドルされている商用アプリケーションは別CD-ROMにまとめて収録されており、通常のインストーラからはインストールできないのだ。ほかのディストリビューションでは、TrueTypeフォントや日本語入力システムなどが標準的にインストール・設定され、すぐに使えるものもあるだけに、個別にRPMパッケージをインストールし、設定を行わなければならないのは少し面倒に感じられる。これは初心者にとっては難しい作業だ。

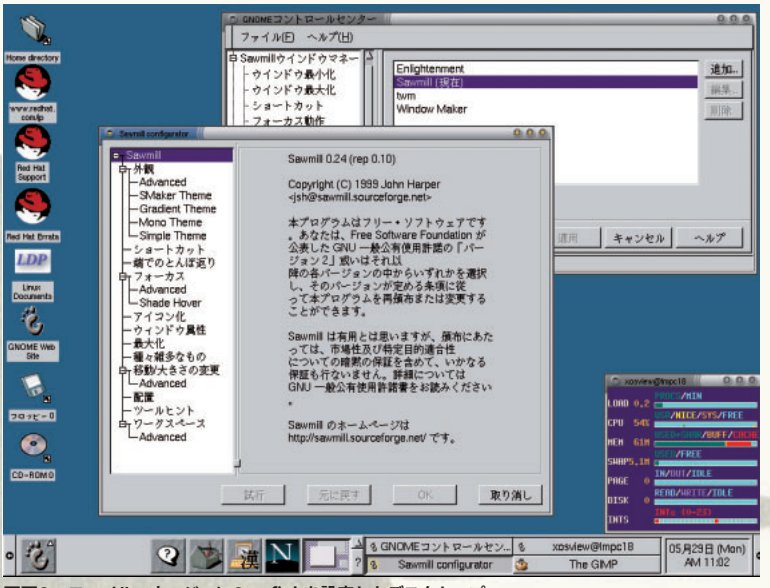
アプリケーションCD-ROMを別個にしているのは、製品版のバイナリCD-ROMと、FTP版との共通化を図るという意図があると思われる。これはこれで、評価できる姿勢なのだが、であれば、環境設定まで可能な専用のインストーラを用意するなどして対処してほしいところだ。

●

総合的には、収録アプリケーションも含め、パッケージ構成は「中庸」といった印象だ。ひととおりの機能が過不足なくそろっており、「ベーシックな」Linux環境の構築に適したディストリビューションといえる。そこから、用途に合わせた環境設定をユーザーが行っていくことになる。そのため、ある程度以上の知識を持ったユーザーに向いているだろう。

基本的な機能や設定は、ほかのRed Hat系ディストリビューションへ引き継がれる部分も多いはずで、その意味でも、スタンダードとして果たす役割は大きい。ユーザーにとっては、新しい機能をいち早く体験でき、アプリケーションやハードウェアの対応に関しても先行しているというメリットがある。各ディストリビューションが独自性を打ち出す中で、今後も中心的な存在としての地位を保っていくのか注目したい。

画面2　起動しているサービスの確認
ntsysvコマンドで起動しているサービスを確認したところ。DHCPのデーモン（dhcpd）が起動していないことがわかる。



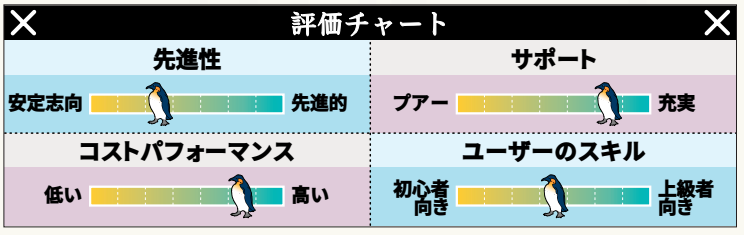
画面3　ファイルマネージャにSawfishを設定したデスクトップ
豊富なカスタマイズ項目が設定可能だ。現在でも設定画面などでは「Sawmill」と表示される。

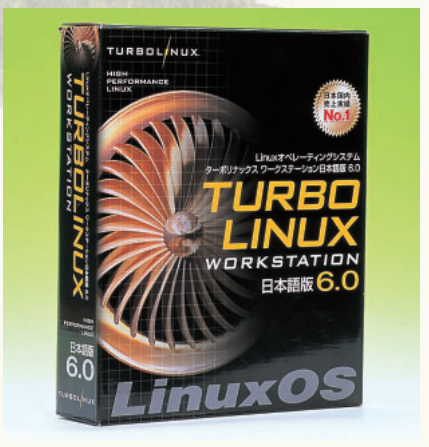
Red Hatとは袂を分かって独自路線を歩む

TurboLinux Workstation日本語版6.0

ターボリナックス ジャパン株式会社
TEL 03-5766-1660
http://www.turbolinux.co.jp/

1万2800円





TurboLinux Workstation日本語版6.0（以下TurboLinux 6.0）は、ターボリナックス ジャパンが開発、販売しているディストリビューションで、4月7日に発売された。前バージョンのTurboLinux日本語版4.0から、5を飛ばして6.0になったのはRed Hatを意識してのことだろう。そして、サーバ用途のTurboLinux Serverと区別するために、Workstationという文字が加わった。価格は1万2800円だが、初回3万本は9800円のキャンペーン価格で提供され、Linux上でWindows 95/98アプリケーションを実行可能な、PCエミュレータVMware Expressが登録ユーザーに無償提供される。また、6月23日発売分からはVMware Expressが標準バンドルされる。

改良されたインストーラ

インストーラは、従来と変わらずテキストモードだが、Xの設定をインストール後に変更してステップ数を減らしている。また、従来はLILOブートローダが対応していなかった1024シリンダを越える領域へのインストールが、LILOのオプション設定（**画面1**）にLBA32モードが追加されたことで可能になった。これにより、いまでは当たり前となった8.4Gバイト以上の大容量ハードディスクが活用しやすくなった。

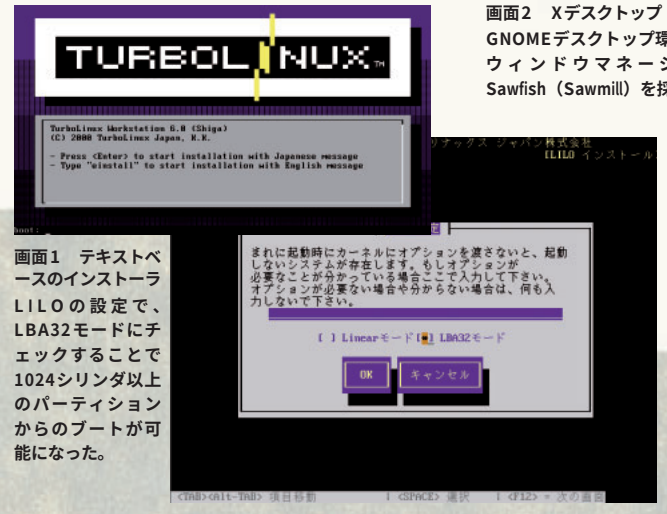
製品版のインストールCDには商用ソフトも含まれているため、インストールが終了してすぐに、ATOK12SE、Wnn6 Ver.3、リョービ商用フォント、翻訳魂が使える環境になる。

Xの設定では、ウィンドウマネージャとしてEnlightenmentも選べるが、GNOMEまたはKDEのデスクトップ環境を利用することが多いだろう（**画面2**）。

プリンタドライバも改良され、エプソン、キヤノンなどの汎用カラープリンタのフィルタを組み込んだ。エプソンのPM-770C、PM-800Cでの写真画質の印刷を実現するPIPSドライバも含まれている。

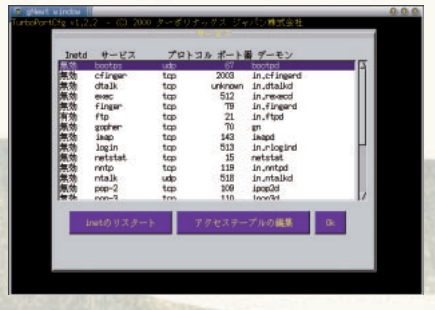
セキュリティを強化

TurboLinux 6.0からセキュリティ強化のために、インストール直後はネットワークのポートのほとんどが閉じた設定になっている。そのため、telnetやFTP



画面1　テキストベースのインストーラ
LILOの設定で、LBA32モードにチェックすることで1024シリンダ以上のパーティションからのブートが可能になった。

画面2　Xデスクトップ
GNOMEデスクトップ環境の標準ウィンドウマネージャにはSawfish（Sawmill）を採用。



画面3　TurboCentro（TurboTools）
専用設定ツールがXからも利用できるようになった。TurboPortcfg（右の画面）でネットワークサービスを設定する。

画面4　TurboLinuxソリューションガイド

で接続するためには、/etcディレクトリにあるinetd.confやhosts.allow、hosts.denyファイルを修正する必要がある。この変更は、専用ツールTurboPortcfgで行うことができる（**画面3**）。

また、telnetを利用してネットワーク経由でログインするときに、単に“login:”とだけ表示され、OSやバージョンが分からないようにするという工夫も取り入れられた。

CATVやNTTのIP接続サービスなど、常時接続環境が手軽に手に入る状況になりつつある現在、安全方向にセキュリティが設定されていることは、初心者にとってありがたいことだろう。

便利な付属ソフトウェア

TurboLinux 6.0には、バイナリ、ソース、コンパニオン、ソリューションガイドの4枚のCD-ROMが付属する。

製品には**表1**のような商用ソフトが付属する。ATOKは、ATOK12SE/R.2にバージョンアップされ、カタカナ入力、on the spot入力、カタカナ英語辞書、フェイスマーク辞書などの新機能が追加され、使い勝手が向上した。商用フォントはリョービフォントを採用、かな漢字変換には、Emacsとの相性がよいWnn6 Ver.3も選べる。

日英／英日翻訳ソフトである翻訳魂は、英文が多いLinuxのドキュメントを読む助けとなるだろう。また、ブートマネージャのSystem Commander Liteを使うことで、WindowsとLinuxのマルチブートが容易になる。

さらに、コンパニオンCDには、4Gバイトメモリ対応カーネル、XFree86 4.0、Mesa 3Dライブラリなど、多くのLinux用フリーソフトの最新版RPMパッケージが納められている。

TurboLinuxソリューションガイドは、TurboLinuxで稼動が検証されているアプリケーションやツール、対応周辺機器のほか、トレーニングコースなど、約100件の情報が収録されていて、Webブラウザで説明を見ることができる（**画面4**）。これには、試用版、体験版ソフトも多数収録されていて、必要に応じて試すことが可能だ。

充実したマニュアル、サポート

マニュアルは、インストール解説の「ユーザーリファレンスガイド」と、「活用ガイド」の2冊が付属する。

オンラインサポートは、従来のWeb、E-mailに加えて、電話およびFAXサポートも受けられるようになった。サポート範囲はインストールに関して90日間3件までとなっている。

同社のWebページから、セキュリティ情報やアップデート情報が提供され、アップデートパッケージは無償でダウンロードが可能である。また、登録ユーザーには最新情報がメールで送られる。

バランスの取れた構成

当初、Red Hatベースで開発されていたTurboLinuxだが、インストーラもテキストベースのままで、TurboToolsといったTurboLinux専用設定ツールを用意するなど、独自路線を歩んでいる。そのため少し癖があるともいえるが、付属するアプリケーションも十分な内容であり、説明不足といった点もほとんどなく完成度は高い。初心者から上級者まで満足するできといえるだろう。

日本語かな漢字変換	ATOK12 SE/R.2 for Linux Wnn6 Ver.3
日本語TrueTypeフォント	リョービ商用日本語TrueTypeフォント 明朝、ゴシック、羽衣、シリウス、サンセリフの5書体
日英/英日翻訳ソフト	翻訳魂
マルチブートマネージャ	System Commander Lite
PCエミュレータ	VMware Express

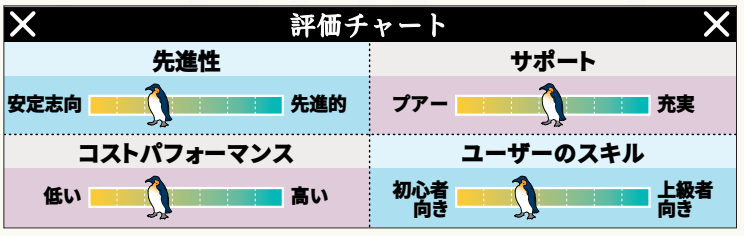
表1　TurboLinux Workstation日本語版6.0に含まれる主な商用ソフト

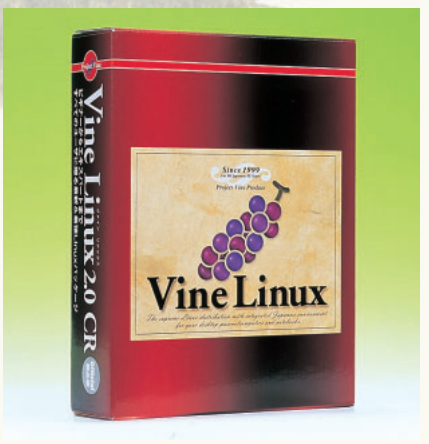
充実の日本語環境で勝負

Vine Linux 2.0 CR

技術評論社
03-3225-3481　http://www.gihyo.co.jp/
Project Vine
http://vinelinux.org/

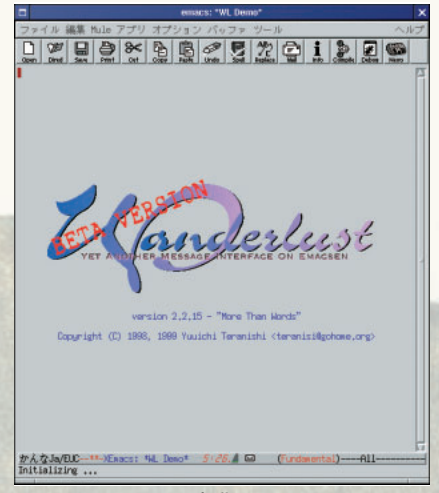
9800円





Vine Linuxは、使いやすい日本語環境を提供するディストリビューションとして高い評価を受けている。最新のバージョン2.0は、Red Hat 6.1をベースに開発され、今年4月にリリースされた。製品版のVine Linux 2.0 CRは、コンピュータ関連書籍の出版でもおなじみの技術評論社より発売されている。

Vine 2.0は、グラフィカルインストーラの採用、GNOMEとKDEの採用によるデスクトップ環境の充実、Emacsで動作する人気のメーラWanderlustなどのアプリケーションの収録により、デスクトップ用途での利用を強く意識したことをうかがわせるパッケージ構成になっている。



画面1　Wanderlustの起動画面

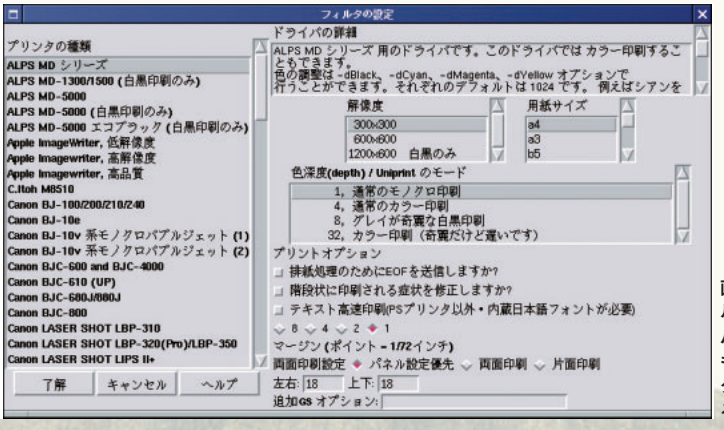
PJEの成果を継承

Vineの開発は、PJE（Project Japanese Extensions）のメンバーが中心となって運営するProject Vineが行っている。PJEはLinuxの日本語環境の整備を目的としたプロジェクトで、その成果はRed HatやSlackware向けのアドオンパッケージとして、これらのディストリビューションに組み込まれていた。現在、PJEは休眠状態になっているようだが、その成果はVineに引き継がれている。Vineは、PJEの活動を背景に、日本語環境の使い勝手を重視したディストリビューションとして当初から開発が進められてきたのである。

このため、デフォルトの設定でも十分な日本語環境が提供できるよう、次のような配慮がなされている。

- **日本語入力に対応した豊富なEmacs設定ファイル**
- **独自のX用固定ピッチフォント**
- **国内メーカーのプリンタに対応したプリントフィルタの開発（このプリントフィルタはRed Hat 6.2Jにも採用されている）**
- **cp、mvなど、主要なコマンドのメッセージの日本語化**
- **日本語検索エンジンNamazuの標準採用（デフォルトでデスクトップ環境から利用可能）**

ほかにも、独自開発の「Vine Tools」という日本語アプリケーションが付属している。いまのところは、メーラ（Vmail）とテキストエディタ（Vedit）のみで、それほど高機能ではないが、



画面2　プリンタフィルタの設定
バージョン1.0に比べても選択できるプリンタの種類が増えている。

どちらもGUIアプリケーションとして初級者でも直感的に操作しやすい作りになっている。上記の項目とあわせて、インストールしてすぐに、ひととおりの日本語デスクトップ環境が整うというのが大きなアピールポイントだ。これは、デスクトップ用途を考えた際に、大きな利点となるだろう。

待望のバージョンアップ

1999年6月のバージョン1.1のリリースから、今回のバージョンアップまで、ほぼ1年の期間を要した。そのため、カーネルなどの基本コンポーネントの部分で、ほかのディストリビューションにやや遅れをとっていた。

しかし、バージョン2.0のリリースにより、この問題は解消されている。カーネル 2.2.14、glibc 2.1.2、XFree86 3.3.6という基本コンポーネントの構成は、今回紹介するディストリビューションにみられる標準的なものだ。

■GUI環境も進化

前述したように、統合デスクトップ環境としてGNOME（ウィンドウマネージャはEnlightenment）とKDEも標準採用された。デフォルトは従来どおりWindowMakerだが、グラフィカルログイン画面での切り替えが可能になっている（この画面では、使用する日本語入力システムも切り替えられる）。デフォルトではインストールされないが、製品版のVine Plus CD-ROMには、Sawfish（Sawmill）も収録されている。このCD-ROMにはほかにも、冒頭で触れたWanderlust、高機能ファイルマネージャFileRunnerといったProject Vineお勧めのアプリケーションが含まれている。一度チェックしておこう。

開発ペースからもわかるように、どちらかといえば安定指向のディストリビューションだが、GUI環境でも最近のトレンドに追従した形となった。もちろんこれはユーザーの側から見ても、より充実したデスクトップ環境が得られることから、歓迎すべき進化だといえるだろう。

かな漢字変換システム	Wnn6 Ver.3
商用TruTypeフォント	DynaFont和文5書体

Vine Linux 2.0 CRにバンドルされるソフトウェア

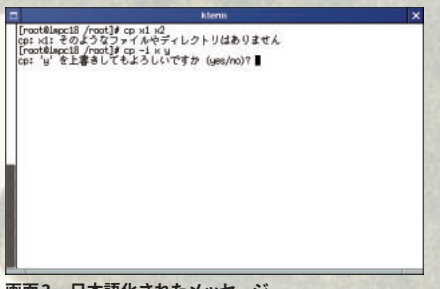
課題はコストパフォーマンス？

Vineの製品版にバンドルされる商用ソフトウェアは、**表1-3**に示すように、Wnn6 Ver.3とDynaFontの和文5書体のみで、充実しているとは言いがたい。価格（9800円）を考えるとコストパフォーマンスの点では、TurboLinuxやLASER5などのバンドルソフトが充実した製品版に比べると、少しもの足らない気がするユーザーがいるかもしれない。基本となるディストリビューション部分がしっかりしているだけに、かえって製品版の価格が高いと受け取られてしまう可能性がある。

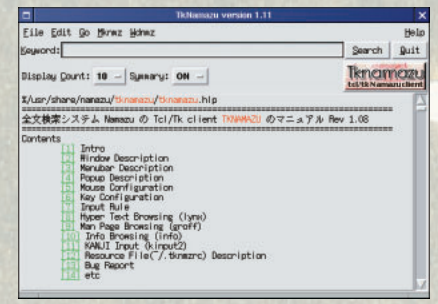
しかし、マニュアルもインストール方法だけでなく、基本操作や各種の設定、RPMによるパッケージ管理の手順などがわかりやすくまとまっているし、整備された日本語環境、そして製品版で受けられるサポートのことも考え合わせれば、納得できる範囲内だろう。

総合的にみて、「ユーザーを選ばない」ディストリビューションだということができるだろう。しっかりした日本語環境は、初心者にとっても、中・上級者にとっても有益なことだ。特に、使い始める前に、あれこれ設定したり、余計な手間をかけたくない人にはお勧めだ。

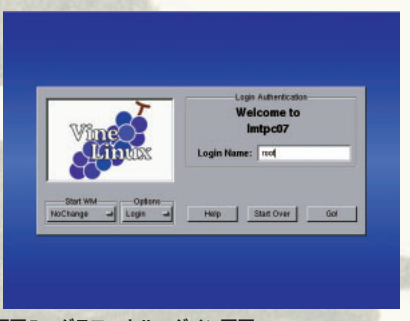
各ディストリビューションともバージョンアップに伴い、日本語環境の整備が進んできている。Vine Linux最大の特徴であるこのポイントも、ほかとの差はずいぶん縮まってきた。今後は、フォントや印刷環境のいっそうの整備と、Vine Toolsの充実といった面での発展に期待したい。



画面3　日本語化されたメッセージ
メッセージが日本語のため、日本語表示が可能なKtermやKon上でないとメッセージが文字化けする。設定を変更して英語メッセージにすることも可能。



画面4　TkNamazu
デフォルト設定のままでも、Vineのマニュアル、JF、JMの文書を検索できるようになっている。



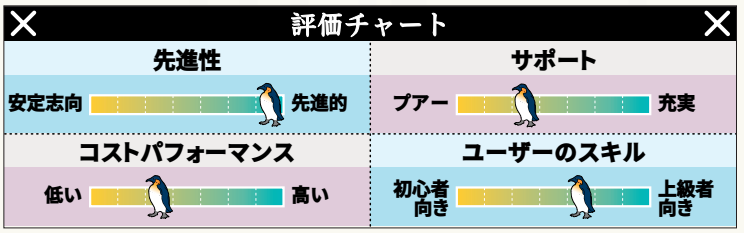
画面5　グラフィカルログイン画面
ウィンドウマネージャや言語設定をログイン時に切り替えられる。

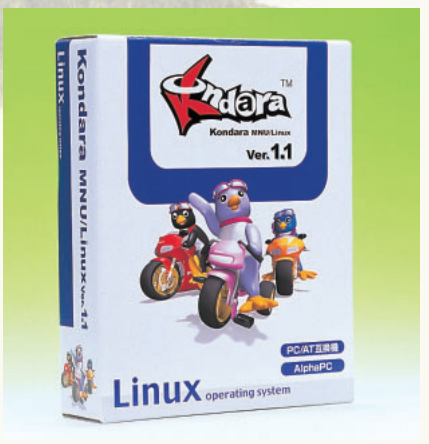
積極的な取り組みで勢いを増す風雲児

Kondara MNU/Linux 1.1

デジタルファクトリ株式会社
06-6882-5850　http://www.digitalfactory.co.jp/
Kondara Project
http://www.kondara.org/

7800円





Kondara Projectのオフィシャル Webサイトによると、当初はディストリビューションとしてリリースする予定はなかったそうだ。Kondara MNU/Linux（以下、Kondara）も、あくまで最新のコンポーネントをまとめたパッケージ集という体裁だったらしい。新しいCライブラリなど、プロジェクトのメンバーが使いたい（興味のある）ものを日本語化したり、Linuxで動作するように作り込んだりしていたことがプロジェクトの出発点であったようだ。「使いたい人＝作る人」という発想が原点にあるのだろう。

ディストリビューションとしてのKondaraは、Red Hatの先行開発バージョンRawhideをベースに開発され、リリースに至っている。昨年12月から、デジタルファクトリジャパン（当時。現在は「ジャパン」がとれてデジタルファクトリ株式会社に社名変更）とのジョイントにより製品版の発売も開始された。

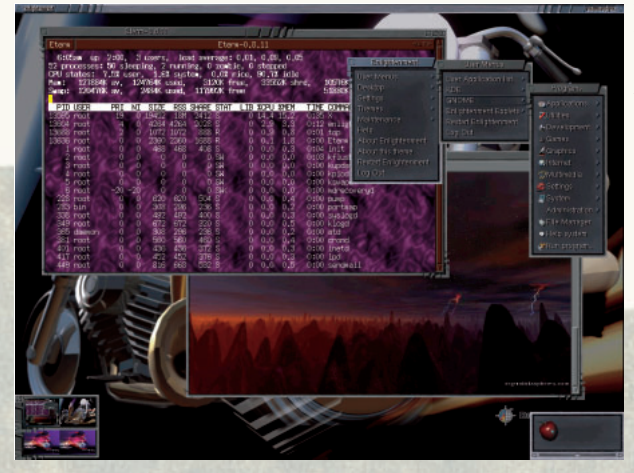
最新のパッケージをいち早く

Kondara Projectは、発足の経緯からもわかるように、最新の機能をいち早くとり入れるフットワークの軽さが身上だ。その成果はWeb（FTP）サイトにおいて、すばやく公開されている。Webサイトでの開発日誌の公開や、掲示板での活発なやり取りなどを通じて、プロジェクト活動を積極的にオープンにしているのも特徴である。

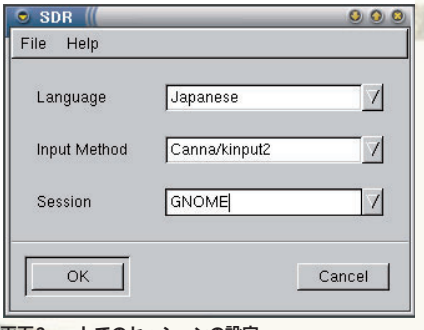
製品版も、発売当時からGUIインストーラ「Anaconda」を採用するなど、プロジェクトの姿勢が反映されたパッケージ構成となっている。最新の製品パッケージであるバージョン1.1では、基本コンポーネントこそ他のディストリビューションと差異がないものの、開発バージョンのGNOMEコア 1.1.4、フリーのサウンドドライバALSA 0.5.5などを取り入れることで、先進性をアピールしている。

楽しい？デスクトップ環境

1.0から1.1へのバージョンアップに伴い、標準のXウインドウマネージャは、EnlightenmentからSawfishに変更された。また新たに、統合デスクトップ環境のGNOMEとKDEが追加されている。GNOME＋Sawfishという組み合わせは今後のトレンドになると予想される。デスクトップの環境はグラフィカルログイン画面、および独自開発の



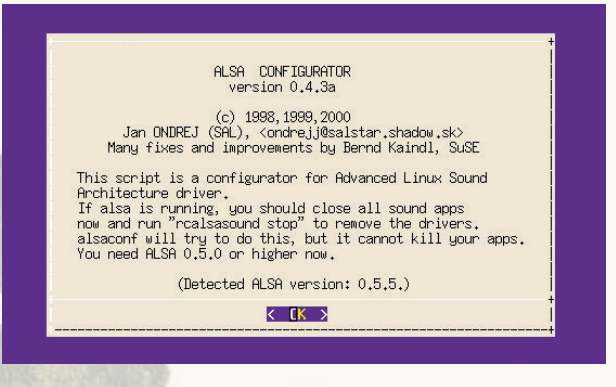
画面1　デフォルトのデスクトップ
画面はEmlightenment。どのウィンドウマネージャも、デフォルトの設定をKondaraオリジナルに変えてある。どれもかなり派手。



画面2　sdrでのセッションの設定
言語、かな漢字変換と入力メソッド、ウィンドウマネージャをX上で切り替えられて便利である。

スイッチングツールsdrで切り替えることが可能になっている。

オリジナルの壁紙やデフォルト設定の凝り方も「らしい」ところか。ただし、この点は好みの問題でもあるので、賛否が分かれるところだろう。シンプルなデスクトップ設定が好きなユーザーは設定作業が必要となる。



画面4　ALSAの設定ツールalsaconf
収録されているALSAドライバのバージョンは0.5.5だ。

■豊富なアプリケーション

製品版にバンドルされる商用ソフトウェアは、DynaFont和文4書体のみだが、Wnn6 V3（日本語入力システム）、dp/NOTE（ワープロ）、翻訳魂（日英翻訳ソフト）などの優先販売クーポンが同梱されており、必要なソフトを選んで割引で購入できる。

商用アプリケーションこそバンドルされていないが、独自に開発したものも含めて、多くのフリーアプリケーションやツールが収録されている。前述のsdrをはじめ、テキストエディタmgedit、コンソールで動作するメーラMuttとニュースリーダslrn、Vineのところでも触れた人気のメーラWanderlustなど、もりだくさんだ。

マニュアルがしっかりしている点も評価したい。インストールはもちろん、PPPやNetscapeの設定から、ウインドウマネージャごとのデスクトップ環境のカスタマイズについて、そして収録ソフトウェアの簡単な使用方法まで、きちんと説明されている。製品版を購入すれば、初心者でも、インストールするだけでなく、マニュアルを見ながらLinuxを使ってみることができるように配慮されている。

多言語対応とAlphaへの取り組み

もうひとつの大きな特徴は、国際標準化への取り組みだ。1つの環境で多言語に対応できる方向で開発が進められており、現行でもログイン時に「英語のみ」、「英語と日本語」、「日本語のみ」の中から言語環境を選べるようになっている。

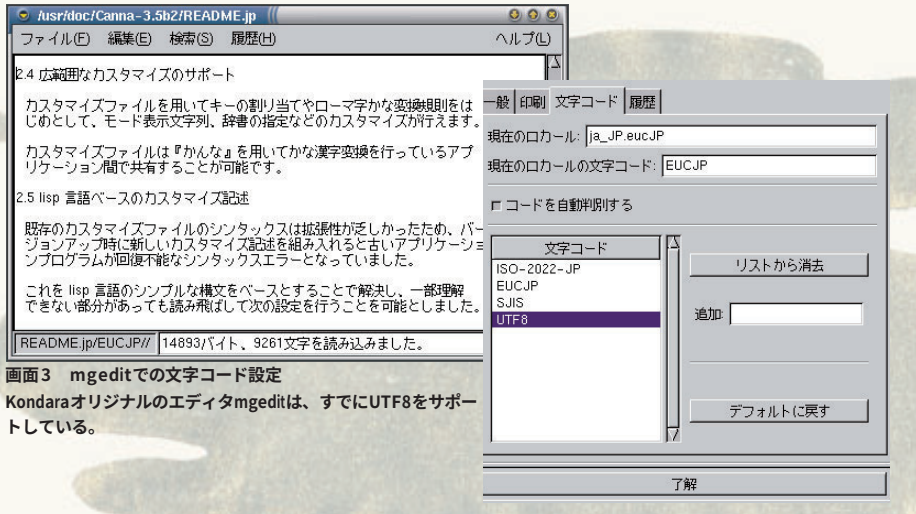
Kondara Projectは、UTF-8への移行も表明している。UTF-8はUnicodeの文字コードセットのひとつで、Linuxに関してはLi18nuxなどで国際化への取り組みとして推進されている。Kondaraでは、すでに前出のmgeditがUTF-8に対応している。今後は、UTF-8のロケールを用意し、アプリケーションでの対応やフォントの整備などを進めていく方針のようだ。

Kondaraは、バージョン1.0の頃からコンパックの64ビットAlphaアーキテクチャをサポートしていた。バージョン1.1では、PC用とAlphaチップ用のソースコードを統合し、両アーキテクチャで共有するようになっている。これにより、PC版と同じタイミングでAlphaもサポートされるようになるはずだ。

●

とにかく新しい機能を早く試したいという人や、開発プロジェクトの動向や雰囲気に触れてみたいといった、Linuxに対してテクニカルな興味のあるユーザーにお勧めのディストリビューションだ。逆に企業ユースなど、安定性や実績の評価が重視されるケースでは、この点は説得力をもち得ないかもしれない。

日本語環境の整備も高いレベルにあり、マニュアルの充実も含めて初心者にも勧められるパッケージングであることもつけ加えておきたい。



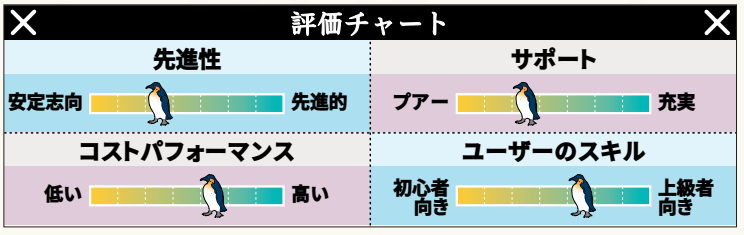
画面3　mgeditでの文字コード設定
Kondaraオリジナルのエディタmgeditは、すでにUTF8をサポートしている。

豊富なバンドルソフトを誇る老舗

LASER5 Linux 6.0 rel.2

レーザーファイブ株式会社
03-5818-6626
http://www.laser5.co.jp/

1万2800円



五橋研究所は、1993年以来、日本語環境をまとめたLinuxパッケージの販売を行ってきた老舗である。1996年からは、Red Hatの国内バージョンを「日本語Red Hat Linux」として開発・販売してきた。レーザーファイブは、五橋研究所のOS部門が独立して1999年8月に設立されたディストリビューターだ。以後、Red Hat互換の日本語ディストリビューションを独自に開発・販売している。LASER5 Linux 6.0は、1999年9月にリリースされ、今年の1月にマイナーバージョンアップをうけて現行の6.0 rel.2に至っている。ベースディストリビューションはRed Hat 6.0である。

あくまでもマイナーチェンジ

6.0からrel.2へのバージョンアップでは、基本コンポーネントの変更は行われていない。カーネル2.2.5、glibc 2.1.1、XFree86 3.3.5が採用されている。安定性を重視したためだろうが、6.0のリリース開始からすでに1年近く経過しており、現在では基本コンポーネントに関して、ほかのディストリビューションに若干見劣りすることは否めない。

ただ、これらのコンポーネントが極端に陳腐化しているわけではないので、特に問題はないはずだ。UltraDMA/66やi810など、手持ちのハードウェアがサポートされていない場合もあるだろうが、動作確認のとれた環境で、安定した動作を望むのであれば、気にする必要はまったくない。製品版に含まれる「contrib CD」には、カーネル2.2.13、glibc 2.1.2、XFree86 3.3.6も含まれている。それぞれインストール用のスクリプトも用意されているので、必要に応じて、それほど手間をかけずにアップデートすることが可能だ。

また、発売日は確定していないが、次期バージョン（LASER5 Linux 6.2）のリリースも準備されている。カーネルが2.2.14に、glibcとXFree86が、それぞれ2.1.3と3.3.6にアップデートされる予定で、近いうちに正式に発表されるはずだ。

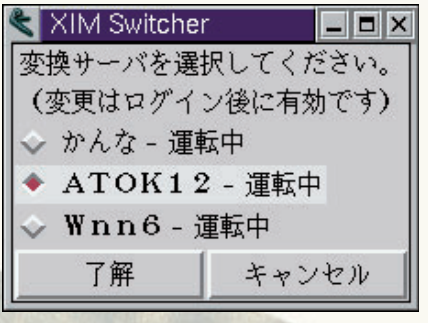
使いこなしに配慮

LASER5の最大の売りは、製品版にバンドルされた豊富なソフトウェアだろう。日本語入力から、英和辞書、3Dレンダリングソフトまで、デスクトップOSとして幅広い用途に対応できるパッケージ構成となっている。

さらに、ウイルス対策ツール、セキュリティ関連ソフトウェア、Webベースのリモート管理ツール、周辺機器の商用ドライバなどもバンドルされており（**表1**）を参照）、Linuxをメインの OSとして利用できるよう配慮されている。製品版のパッケージを購入するだけで、Linuxを十分に活用できる環境が整うのがうれしいところだ。1万2800円という値段も、これだけパッケージ

かな漢字変換システム	ATOK12 SE for Linux、Wnn6 Ver.3
商用TrueTypeフォント	DynaFont和文5書体
日英辞書引きソフト	eWnn Ver.1.0 for Linux（研究社「新英和・和英中辞典」）
セキュリティ関連ソフト	F-SECURE SSH1/SSH2、Roxen Challenger（168ビット暗号）、Phoenix Adaptive Firewall
アンチウイルスソフト	F-SECURE Anti-Virus、Sophos Anti-Virus
3Dレンダリングソフト	BLENDER
その他	System Commander Lite（ブート管理ツール）、Linux Controller（サーバ管理ツール）、商用周辺機器ドライバ、UPS管理ソフト

表1　LASER5 Linux 6.0 Rel.2にバンドルされる商用ソフトウェア



画面1　XIM Switcher

内容が充実していれば納得できる。

Linux自体をホビーとして楽しむのではなく、本来のOSとしての用途で使ってほしいというディストリビューター側からの主張も感じられる（もちろん、企業への導入を視野に入れたマーケティング戦略的な狙いもあるのだろうが……）。次期バージョンでは、Microsoft製品ともデータ互換性のあるオフィススイート「ThinkFree Office」もバンドルされる予定になっている。

Linuxを使いこなすための配慮として、もうひとつマニュアル類の充実があげられる。製品版には、「インストールガイド」、「ユーザー活用ガイド」、「システム管理ガイド」のメインとなるマニュアルのほかに、バンドルソフトのマニュアル（ATOK12 SE for Linux、Wnn6/eWnn、System Comander Lite）と、JFやJMによるLinux関連の日本語ドキュメントを収録したCD-ROM「LaserDoc」が同梱されている。

一部のバンドルソフトのインストール方法を説明したドキュメントが（READMEなど）、CD-ROM上の各ソフトウェアが収録されたディレクトリに分散して置かれているのはもったいない。もちろん、ディストリビューションによってはすべてがそうなっている場合もあるし、「とにかくREAD MEを読め！」が基本ではあるのだが、ここまできたら、全バンドルソフトについて、（最低インストール方法だけでも）マニュアルに記載してはどうだろうか。そこまで徹底できれば、言うことなしである。

GUI環境はどうなっているのか

X Window Systemのデスクトップ環境は、GNOMEとKDEが採用されている。デフォルトはGNOMEで、ウィンドウマネージャはEnlightenmentだ。グラフィカルログイン画面での切り替えにも対応している。

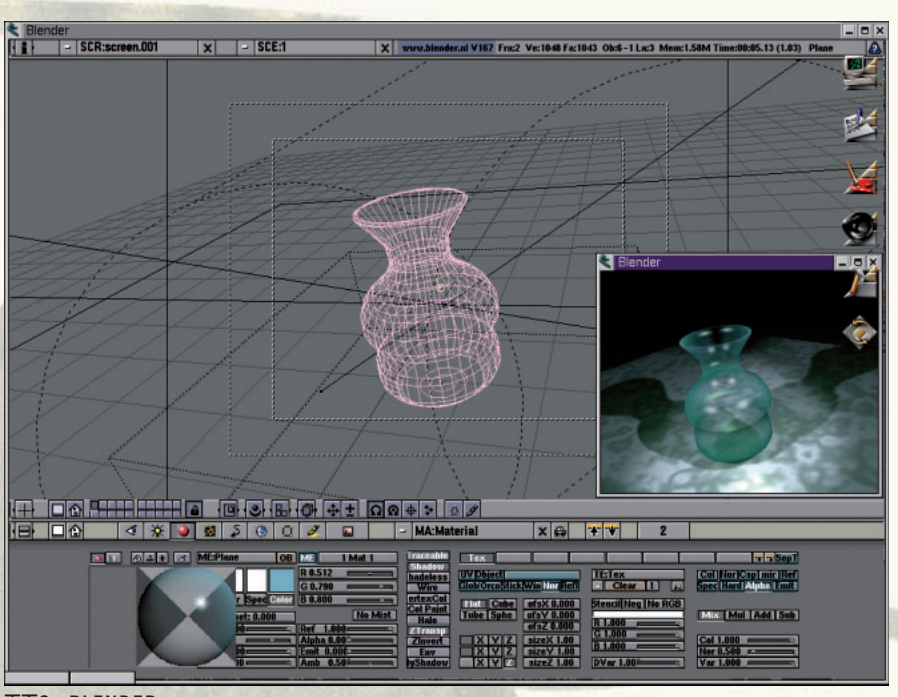
まったく個人的な意見で申し訳ないのだが、デフォルトでEnlightenmentの結構派手なデスクトップアイコンが設定されているのは気に入らない。デフォルト設定はシンプルにしておき、変更はユーザーの好みにまかせたほうがベターではなかろうか。細かいことだが、いちおう指摘しておこう。

ユーザーインターフェイス面での優れものとして評価したいのが、日本語入力システムを切り替えるグラフィカルツール「XIM Switcher」だ。Xで使用する日本語入力システムの切り替えは、初心者には難しい作業だが、このツールを使えばGUI操作で簡単に切り替えられて便利である。ただし、設定を有効にするにはログインし直す必要がある。デスクトップ用途からは離れるがWebベースのサーバ管理ツール「Linux Controller」も操作性の高いGUIツールだ。

●

総括すると、Linuxを使いこなすために必要なすべてがそろったパッケージ構成は高ポイント。マニュアルの充実と合わせて、安心して使えるディストリビューションとして初心者にもお勧めだ。

画面2　Linux Controller
WebベースでDNS、DHCP、Apacheなどの各種サーバを管理できる。



画面3　BLENDER
Linuxで動作する3Dレンダリングソフトの定番だ。

グラフィック用途を推奨して新規参入

Linux2000G

株式会社ホロン
03-5282-5101
http://www.holonsoft.co.jp/

1万1800円

評価チャート

先進性	サポート
安定志向 ― 先進的	プアー ― 充実
コストパフォーマンス	**ユーザーのスキル**
低い ― 高い	初心者向き ― 上級者向き

Linux2000Gは、タイピング習得ソフト「激打」と「闘打」でおなじみのホロンより発売された新しいディストリビューションだ。2000Gの「G」はグラフィックを示しており、その名のとおり、グラフィックス処理にLinuxを使うことをコンセプトとしている。

インテル版とMacintosh版を収録

グラフィックスといえばMacintoshということなのか、製品パッケージには、AT互換機用の「インテル版」とPowerPCに対応した「Macintosh版」が収録されている。インテル版はRed Hat 6.0、Macintosh版はLinuxPPC 1999がベースとなっている。2バージョンを収録した意図は、PCとMacで共通の環境を提供し、統一した操作性による処理を実現することにあるようだ。

コストパフォーマンス面からみても、Linuxをグラフィック用途で使うことには利点がある。Macには、Photoshopなどのフォトレタッチソフトをはじめとするグラフィック処理アプリケーションが豊富にそろっているが、これらのグラフィック系ソフトは高額なものが多い。Linuxには、「Photoshopキラー」の異名をとる優れたフォトレタッチソフトであるGimpがほとんどのディストリビューションに収録されている。3Dグラフィックス作成に関しても、POV-Rayというフリーソフトが動作する（POV-RayはMac OSにも対応しているが……）。

ユーザーフィードバックを重視

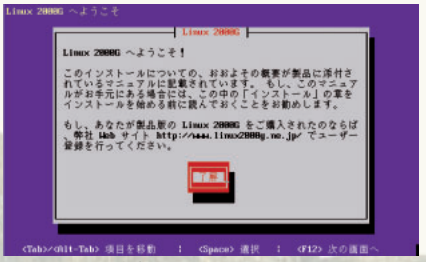
4月以降にリリースされたRed Hat系ディストリビューションの多くがグラフィカルインストーラ（Anaconda）を採用しているなか、Linux2000Gはキャラクタベースのインストーラをあえて採用している。これは、事前のユーザー調査で、初心者の場合はGUIのほうが誤った操作をする確率が高いという結果が得られたからだそうだ。

今後、ユーザーからの要望が高ければグラフィカルインストーラの採用も検討する予定だ。その際には、独自開発のものになる可能性もあるらしい。このインストーラの件だけでなく、ユーザーからのフィードバックを積極的に製品に反映して開発を進めていくとしている。発売されて間もないため、具体的な成果はまだだが、Webサイトに掲載されているFAQなども徐々に充実してきている。エンドユーザーを意識したディストリビューションとして、この点には期待したい。

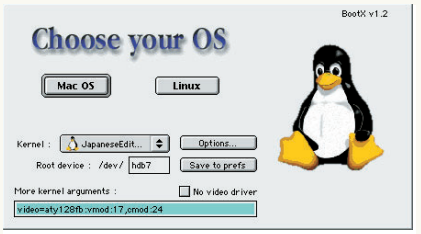
●

2バージョンが収録されているとはいえ、製品版バンドルされる商用アプリケーションがShadeの体験のみで、1万1800円という価格は少し高い気がする。

今後、Shadeのフルバージョン（現在は試用版をバンドル）や3Dレンダリングソフトなどをバンドルして、3Dグラフィックスライブラリなどのグラフィックス用途での使用環境の整備を進めていけば、独自性を持ったユニークな存在として注目を集めるようになっていくのではないだろうか。



画面1　キャラクタベースのインストーラ
Red Hat系のグラフィカルインストーラと設定項目などはそれほど変らないので、特に問題とはならないだろう。

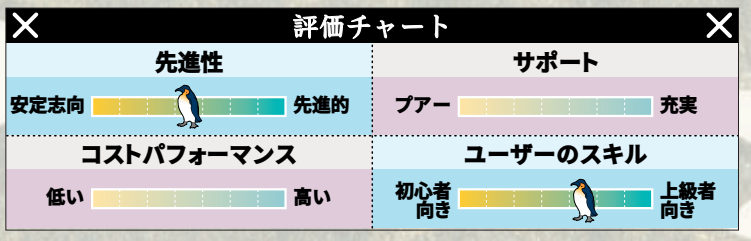


画面2　Mac用のブートローダーBootX
Macはブートローダーもグラフィカル。少し見ずらいが「hda7」がroot（/）として設定されている。

非商用ディストリビューションの雄

Plamo Linux 2.0

Plamoのホームページ
http://www.linet.gr.jp/~kojima/Plamo/



Plamo Linuxは今回紹介するうちでは唯一、商用ベースで製品版が販売されていないディストリビューションである。Slackwareをベースに、Linuxの日本語対応に取り組んでいたPJE（Project Japanese Extensions）提供の日本語版ソフトウェアをまとめ、インストーラの日本語化、独自のカーネルのチューニングなどを施してリリースされている。こじまみつひろ氏が中心となって、ボランティアベースで開発が進められている。

PC-98をサポート

Plamoのディストリビューションとしての特徴としてまずあげられるのが、NECのPC-98x1シリーズに対応したバージョンが用意されているということだ。

PC-98へのLinuxの移植は京大マイコンクラブによって行われ、Linux/98として公開されている（http://www.kmc.kyoto-u.ac.jp/proj/linux98/）。Plamo 2.0では、Linux/98の2.2系カーネルの最新版を採用している。なお、AT互換機対応版のカーネルバージョンは2.2.14である。

カーネル以外の基本コンポーネントについても触れておこう。Cライブラリは、glibc 2.1.2にLinux研究会のNLS分科会（http://www.linuxjp.org/lrs/nls/）が作成した日本語localeパッチをあてたバージョンを採用している。XFree86は3.3.6。GNOMEとKDEもバイナリが用意されている。ただし、デフォルトではインストールされないので、別途セットアップが必要だ。

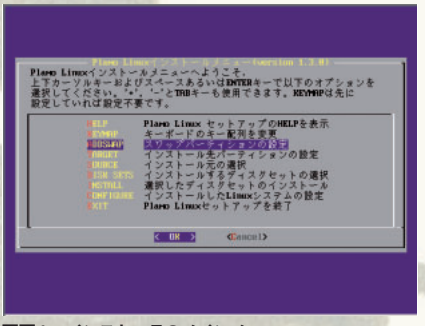
細かな配慮がうれしい

インストールの基本的な手順や操作性はベースとなったSlackwareのインストーラと変わらないが、メッセージが日本語化されていて、この手順ではなにをどうすればいいのか、わかりやすく説明されている（**画面1**）。メッセージに従って、作業を進めていけば、それほどインストールで迷うことはないはずだ。

また、パッケージ選択時に「お勧めパッケージセット」が選べるようになってる。これは開発者の、こじま氏が実際に使っている環境をもとにしたソフトウェア集で、デスクトップマシン向けと、ノートPC向けのものが用意されている。ハードウェア環境に合ったパッケージセットを選択すれば、開発者みずからが選んだアプリケーションやツールがインストールされる。パッケージの選択に迷うことなく、Linuxを活用するために必要なものがひととおり揃うわけだ。

国内で利用されることの多いネットワークカード（3Comの3c509シリーズ、Intelの EtherExpress Proなど）や、SCSIカード（Adaptec、Tekramなど）のドライバを組み込んだ、カーネルが用意されている点も見逃せない。対応ハードウェアを使っている環境であれば、ドライバをあらためて設定する必要はない。初心者にはうれしい配慮である。

Red Hat系のRPMやDebian系のdebのようなパッケージの依存関係を管理する仕組みがないため、インストールや設定が難しいと言われるSlackwareだが、Plamoでは上記のような対応がなされており、初級ユーザーでもインストールできるように配慮されている。ただし、使いこなすためには、それなりの知識が必要となることも確かだ。



画面1　インストーラのメインメニュー
日本語化されたメッセージに従ってインストールの手順を確認しながら、ひとつひとつ実行できる。

製品名		Official Red Hat Linux 6.2J	TurboLinux Workstation日本語版6.0	Vine Linux 2.0 CR
販売元		レッドハット株式会社	ターボリナックスジャパン株式会社	株式会社技術評論社
URL		http://www.redhat.com/jp/	http://www.turbolinux.co.jp/	http://www.gihyo.co.jp/
提供メディア		CD 4枚／FD 1枚※1	CD 4枚／FD 2枚	CD 4枚／FD 2枚
マニュアル		1冊	1冊	1冊
価格		1万2800円※2	1万2800円	9800円
動作条件				
	CPU／メモリ	Pentium互換／16Mバイト以上	Pentium互換／32Mバイト以上	Pentium 100MHz以上／32Mバイト以上
	ハードディスク	500Mバイト以上	2Gバイト以上推奨	600Mバイト以上
インストール				
	インストーラ	日本語／グラフィカル	日本語／テキストベース	日本語／グラフィカル
	ブートメディア	CD-ROM、FD、DOS	CD-ROM、FD、DOS	CD-ROM、フロッピーディスク、DOS
	インストール元メディア	CD-ROM、HDD、FTP、NFS、HTTP	CD-ROM、NFS	CD-ROM、ハードディスク、FTP、NFS、SMB
システム構成				
パッケージ管理システム		RPM	RPM	RPM
デスクトップ環境		GNOME、KDE、Sawfish、Enligtenment、TWM、Window Maker、AfterStep、fvwm	GNOME、KDE、Enlightenment、TWM	Window Maker、GNOME、KDE、Enlightenment、Fvwm2、TWM
日本語入力システム		ATOK12 SE、Wnn6 Ver.3、Canna	ATOK12 SE、Wnn6 Ver.3、Canna	Canna、Wnn6 Ver.3
バージョン				
（注）	カーネル	2.2.14	2.2.13	2.2.14
	標準Cライブラリ	glibc 2.1.2	glibc 2.1.2	glibc 2.1.2
	XFree86	3.3.6	3.3.6	3.3.6
	GNOME	1.0.55	1.0.55	1.0.55
	KDE	1.1.2	1.1.2	1.1.2
設定ツール				
	ユーザー管理	linuxconf、kuser	kusr	linuxconf、kuser
	ネットワーク	netcfg、linuxconf、netconf	turbonetcfg	netcfg、linuxconf、netconf
	Xサーバ設定	XF86Setup、Xconfigurator	turboxcfg、XF86Setup	XF86Setup、Xconfigurator
	ウィンドウマネージャ変更	switchdesk-gnome	turbowmcfg	—
	サウンド	sndconfig	alsaconf	sndconfig
	起動デーモン	linuxconf、Control Panel、ksysv	turboservice、tksysv、ksysv	linuxconf、Control panel
	パッケージ管理	gnorpm、kpackage	turbopkg、gnorpm、kpackage	gnorpm、kpackage
主なバンドルソフト				
	日本語入力	ATOK12 SE for Linux、Wnn6 Ver.3	ATOK12 SE for Linux、Wnn6 Ver.3	Wnn6 Ver.3
	日本語TrueTypeフォント	DynaFont和文5書体	リョービ日本語TrueType5書体	DynaFont和文5書体
	ブートマネージャ	—	System Commander Lite	—
	そのほか	dp/NOTE、一太郎Ark for Java Technology Preview	OSS、一太郎Ark for Java Technology Preview	—
サポート				
	サポート内容	インストール、無料優先FTPアクセス、Red Hatアップデートエージェンシー	対応ハードウェアの認識・設定から、インストール完了まで※1	ユーザー登録後インストールまで
	サポート方法	電話、Web（メール）	電話、FAX、Web（メール）	書面、FAX、E-Mail
	サポート期間	30日間の電話サポート、90日間のWeb（メール）サポート、180日間の無料FTPサイト優先使用権（デラックスのみ）	90日間	ユーザー登録から3カ月間
	サポート件数	制限なし	3件まで	特に制限なし
備考		※1　スタンダードはCD 2枚／FD 1枚 ※2　スタンダードは3980円（商用ソフトなし）	※1　インターネット接続、プリンタ／サウンド設定などの有償サポートあり	

ディストリビューション戦国時代

Kondara MNU／Linux Ver.1.1	LASER5 Linux 6.0 Rel.2	Linux2000G	Plamo 2.0
デジタルファクトリ株式会社	レーザーファイブ株式会社	株式会社ホロン	—
http://www.digitalfactory.co.jp/	http://www.laser5.co.jp/	http://www.digitalfactory.co.jp/	http://www.linet.gr.jp/~kojima/Plamo/
CD 4枚／FD 2枚	CD 6枚／FD 2枚	CD 3枚／FD 2枚	—
1冊	9冊	5冊	—
7800円	1万2800円	7800円	—
Pentium以上／32Mバイト以上	i486互換／16Mバイト以上	i486以上／16Mバイト以上	i486以上／16Mバイト以上
1.2Gバイト以上	600Mバイト以上	600Mバイト以上	600Mバイト以上
日本語／グラフィカル	日本語／テキストベース	日本語／テキストベース	日本語／テキストベース
CD-ROM、FD、DOS	CD-ROM、FD、DOS	CD-ROM、FD、DOS	CD-ROM、フロッピーディスク、DOS
CD-ROM、HDD、FTP、NFS	CD-ROM、HDD、FTP、NFS、HTTP	CD-ROM、HDD、FTP、NFS	CD-ROM、ハードディスク、FTP、NFS
RPM	RPM	RPM	—
GNOME、KDE、Sawfish、Enligtenment、Window Maker、amaterasu、fvwm	GNOME、KDE、fvwm、fvwm2、Window Maker、AfterStep、Lesstif TWM	GNOME、KDE、Sawfish、Enligtenment、twm、Window Maker、AfterStep	fvwm2、twm、Window Maker、AfterStep
Canna、Wnn6 Ver.3	ATOK12 SE、Wnn6 Ver.3、Canna	Canna、Wnn6 Ver.3	Canna、Wnn4
2.2.14	2.2.5	2.2.15 pre4	2.2.14
glibc 2.1.3	glibc 2.1.1	glibc 2.1.3	glibc 2.1.2
3.3.6	3.3.5	3.3.5	3.3.6
1.1.4	1.0.11	1.0.55	(1.0.55)
1.1.2	1.1.2	1.1.2	(1.1.2)
linuxconf、kuser	linuxconf、kuser	linuxconf、Control panel	linuxconf、Control panel
netcfg、linuxconf、netconf	netcfg、linuxconf、netconf	netcfg、linuxconf、netconf	linuxconf、netcon f
XF86Setup、Xconfigurator	XF86Setup、Xconfigurator	XF86Setup、Xconfigurator	XF86Setup
sdr	switchdesk-gnome	switchdesk-gnome	switchdesk
alsaconf、sndconfig	sndconfig	sndconfig	sndconfig
linuxconf	linuxconf、Control Panel、ksysv	linuxconf、Control panel	linuxconf、Control panel
kpackage	gnorpm、kpackage	gnorpm	—
—	ATOK12 SE for Linux、Wnn6 Ver.3	—	—
DynaFont和文4書体	DynaFont和文5書体	—	—
—	System Commander Lite	—	—
—	ATOK12 SE for Linux、eWnn Ver.1.0 for Linux、BLENDER、SSH1、SSH2、Adaptive Firewall、Atnti-Virusなど	—	—
ユーザー登録後インストールまで	基本サポート（インストールまで）※1	ユーザー登録後インストールまで	—
電話、FAX、メール	電話、FAX、Web（メール）	電話、メール	—
ユーザー登録から90日間	ユーザー登録後から90日間	ユーザー登録から90日間	—
3件	3回	制限なし	—
	※1　インターネット接続、出張サービスを受けられるオプションあり		

そのほかのディストリビューションの動向

ここまでに取り上げた以外にも、さまざまなディストリビューションが存在する。製品としてリリースされているものを中心にいくつか紹介しておこう。

そのほかの日本語ディストリビューション

まずは、日本国内で販売されている商用ディストリビューションを見ていく。ユニークなインストーラを持つものや、1台のマシン上でのWindowとの共存を図ったものなど、いずれも個性派揃いだ。

■OpneLinux

米国のCaldera Systems, Inc.によって開発されたディストリビューション。日本語対応の最新バージョンは、昨年11月にリリースされたOpenLinux 2.3日本語版（1万2800円）で、株式会社ネオナジーが販売およびサポートを行っている。

独自開発のグラフィカルインストーラ「LIZARD」は、その名のとおり（LIZARD＝Linux Wizard）Windowsのウィザードに似た画面デザインと操作性を備えている。サウンドカードのテストを行えるのも特徴だ（**画面1**）。機能とは直接関係ないが、インストールファイルのコピーを待つ間、インストーラ画面でゲームを楽しめるようになっているのがユニークである。

パッケージ管理方式にはRPMを採用しており、基本的にはRed Hatベースだが、上記のインストーラやWindowsに近いルック＆フィールを持つKDEを標準のデスクトップ環境として採用するなど、独自色を打ち出している。

製品版のOpenLinux 2.3日本語版には、ATOK12 SE for Linux、Wnn6 Ver.3、dp/NOTE（日本語ワープロ）などのアプリケーションのほかに、パーティション管理ツールPartition MagicとブートローダBootMagicがバンドルされている。

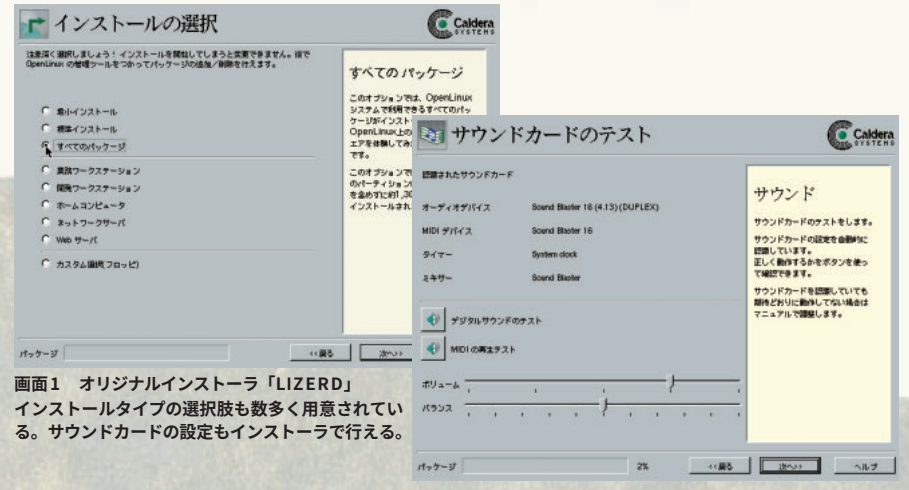
■Linux MLD4

最大の特徴は、Windows95/98のファイルシステムであるFAT16またはFAT32にインストールできる点だ。Red Hatもバージョン6.2から、FATパーティションにインストールできる「パーティションレスインストール」という機能をサポートしているが、MLD4はWindows環境での動作に特化されたディストリビューションで、Red Hatにはない機能をいくつか持っている。

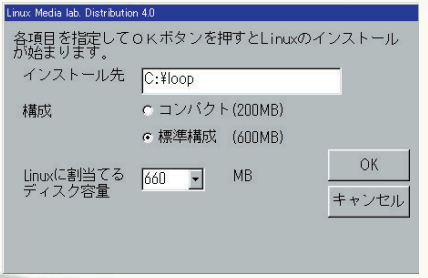
まず、専用のインストーラがWindows上で動作する点。このインストーラは、Windowsのレジストリに記録されているハードウェアやネットワークの構成情報を収集し、自動的にそれらをLinuxに適用してくれるので、非常に手軽にインストールできる。インストール時に空き領域にext2パーティションを作成して、そこにインストールすることも可能だ。

もうひとつの違いは起動方法だ。Red Hatの場合は専用の起動ディスクが必須だが、MLD4は起動ディスクとDOSプロンプトの2つの起動方法をサポートしている。

製品版の開発および販売元はメディアラボ株式会社。価格は8800円で、Wnn6、DynaFont和文5書体などがバンドルされている。同社からはCD-ROMブートで動作する「Live Linux」



画面1　オリジナルインストーラ「LIZERD」
インストールタイプの選択肢も数多く用意されている。サウンドカードの設定もインストーラで行える。



画面2　Windowsから起動するMLD4のインストーラ
操作は実にシンプル。Linux用に割り当てる領域のサイズを指定するだけだ。

も発売されている。

■Linux Mandrake

日本語対応された製品にはめずらしい、フランス生まれのディストリビューション。木村順一氏が作成した日本語化パッケージを組み込んで、「マイスターMandrake 6.1」として五橋研究所から発売されている。ユーザーサポートは提供されないが、DynaLabの商用フォントも付いて3840円というリーズナブルな価格が魅力である。

ディストリビューション	カーネル	glibc	XFree86	標準デスクトップ
OpenLinux 2.3日本語版	2.2.10	2.1.1	3.3.5	KDE
Linux MLD4	2.2.12	2.1.2	3.3.5	KDE
マイスターLinux Mandrake 6.1	2.2.13	2.1.1	3.3.5	KDE

表1　各ディストリビューションのスペック－その1

Debainの動き

Debian GNU/Linuxは、dbe方式と呼ばれる優れたパッケージ管理システムと厳格な方針（DFSG：Debian Free Software Guidelines）に基づいた100%フリーなソフトウェアを特徴とするディストリビューションである。

DFSGに適合した100%フリーのもの（mainと呼ばれる）だけで2000を超えるソフトウェアがパッケージとして収録されている。その多くがGNU由来であることから「GNU」の名が冠せられている。DFSGはライセンス規約というより、Debian Projrctが考えるフリーソフトの配布に対してのポリシーを示した「ガイドライン」としての性格が強い。DFSGで認めている「フリー」ソフトウェアライセンスにはGNUのGPLも含まれている。

ボランティアとして世界中の多くの人々がDebian Projectに参加して、開発を進めている。現在、次期バージョン2.2（開発コード「Potato」）がコードフリーズされ、最終的なテストサイクルに入っている段階だ。Potatoというかわいらしい名称がつけられているが、約4000ものソフトウェアをパッケージングした非常にパワフルなディストリビューションである。

■Debian JP Project

日本でもDebian JP Project（http://www.debian.or.jp/）のメンバーが、1996年8月の発足以来、日本語パッケージの開発にあたっている。その成果は、日本語パッケージとしてリリースされてきた。現行のバージョン2.1にDebian JPの日本語パッケージを組み合わせた製品は、ネットビレッジ株式会社から「Debian JP Project Officail CD DEBIAN GNU/LINUX 2.1」として販売されている（5800円）。

Debian JPは、昨年5月に今後は日本パッケージをリリースしないことを発表したが、これは「本家」Debianに日本語パッケージを組み込むよう方針を変更したためで、バージョン2.2では、Debian Projrctのパッケージのみで日本語に対応できるようになるはずだ。

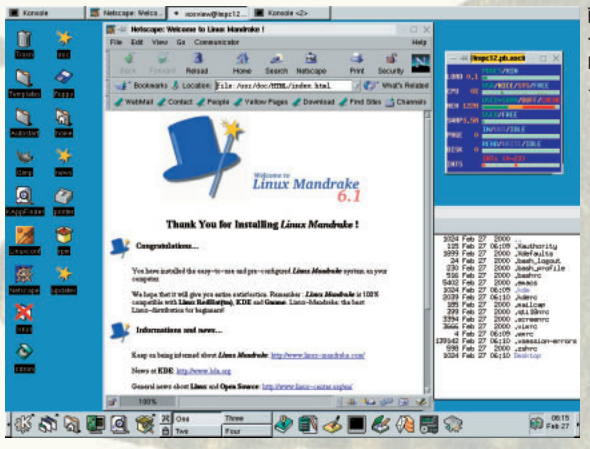
●

Debian JP以外のDebian関連のプロジェクトには、でびまる作成委員会（http://llc.linet.gr.jp/~yochi/debimaru/）とProject Dice（http://dice.debian.gr.jp/）がある。

■Omoikane GNU/Linux

オモイカネ株式会社より4月にリリースされたDebianベースのディストリビューション。Omoikaneとは、日本書紀や古事記に登場する思索の神「思兼命（オモイカネノミコト）」からとられている。考え方と知識をシェアする（兼ねる）という意味で、フリーソフトウェア、オープンソースソフトウェアの精神も表す。

今回リリースされたバージョン1.0は（Intel版とAlpha版を同時リリース）、Debian GNU/Linux 2.2をベースとし



画面3　Mandrakeのデスクトップ
デフォルトのデスクトップ環境はKDE。もちろんGNOMEなど他のデスクトップを選ぶことも可能。

画面4　Debian JPのWebサイト
設立の趣旨、開発方針、規約などが詳しく記載されているのが特徴。「本家」Debianのポリシーが生きている。

ており、パッケージ自体には変更を加えていないため互換性は高い。膨大なDebianのパッケージ群の中から、必要性の高いバイナリパッケージだけを選び、そのソースコードと合わせて1枚のCD-ROMに収録できるサイズまでにスリム化している。また、インストーラも独自に開発し、テキストベースながら対話的な設定を減らすことで作業を簡素化している。

オモイカネでは、製品版の販売を予定しておらず、有料サポートや技術者の教育などの面でビジネス展開していく予定だとしている。

注目の海外ディストリビューション

正式に日本語対応されていない海外ディストリビューションの中で注目したいのが、「Storm Linux 2000」と「Corel LINUX OS」だ。両者ともDebianをベースとしているが、独自のインストーラを備えるなどの拡張を施しており、Debianの長所と使い勝手の良さの両立を図っている。

■Storm Linux 2000

カナダのStormix Technologies Inc.（http://www.stormix.com/）が開発し、昨年12月にリリースされた。グラフィカルなインストーラ（**画面**）、ブートローダ、管理システム（Storm Administration System：SAS）、パッケージ管理システム（Storm Package Manager）など、多くの独自開発された機能を備えている。

SASは、モジュールを追加することで管理項目を増やせるようになっている。統一されたインターフェイスでさまざまな管理項目を操作できるというコンセプトは、Windows 2000のMMC（Microsoft Management Console）に似ている。現在のところ、ネットワーク、ダイヤルアップ、ユーザー管理の3項目が用意されており、今後順次モジュールを開発していく予定だ。

Storm Package Managerは、Debianのパッケージ管理プログラムであるdpkgにGUIフロントエンドをかぶせたもので、DebianオリジナルのCGIインターフェイス（dselect）に比べ、操作性が大幅に改善されている。

このユーザーフレンドリな「ネオDebian」は、9月に日本での発売が予定されている。

ディストリビューション	カーネル	glibc	XFree86	標準デスクトップ
Storm Linux 2000	2.2.13	2.1.1	3.3.5	KDE（1.1.2）
Corel LINUX OS	2.2.12	2.1.2	3.3.5	KDE（1.1.2）

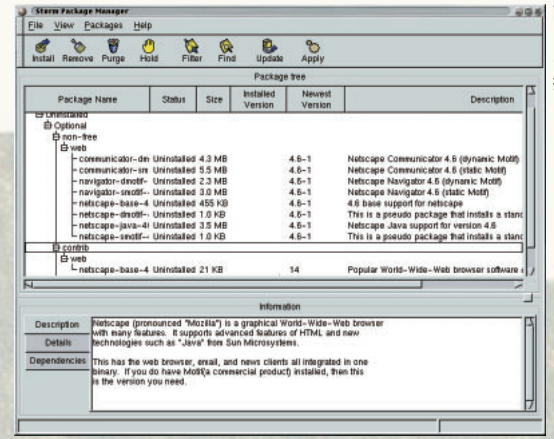
表2　各ディストリビューションのスペックーその2

■Corel Linux OS

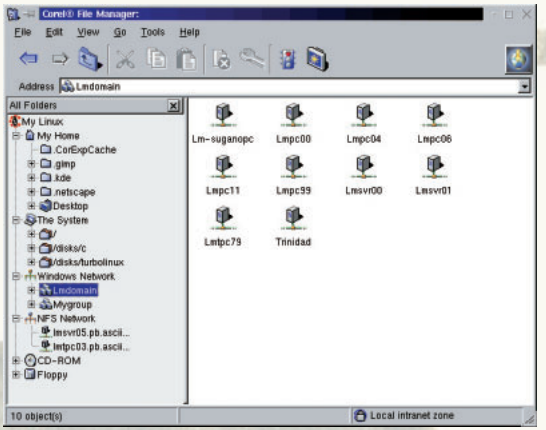
WordPerfectなどのWindowsアプリケーションでおなじみのCorelが開発・販売しているディストリビューション。Storm Linuxと同様に、グラフィカルインスートーラを備え、ほかにも独自の機能を盛りこんで操作性を向上させている。

驚かされるのが、わずか3ステップで設定が完了するインストーラだ。設定項目を絞り込んで、自動認識によるハードウェアやネットワーク構成の検出に従ってインストールが実行される。正式にサポートされているハードウェア環境であれば、非常に簡単にインストールが行えるだろう。インストールする環境の既存OSも自動的に検出され、起動リストに登録されるようになっている。

もうひとつの注目は、独自開発されたファイルマネージャである（Corel File Manager）。WindowsのExplorerに似た操作性を備えているだけでなく、ネットワーク上のWindowsサーバを検索して、シームレスにアクセスすることが可能になっている。LinuxとWindowsが混在した環境では、とても有効な機能である。



画面5　Storm Pakage Manager
GUIインターフェイスの採用により、操作性が大きく向上している。



画面6　Corel File Manager
ネットワーク上のWindowsサーバにシームレスにアクセスできる。

Linuxのユーザーインターフェイスを考える

かつて、Linuxは初心者には使いこなしが困難なOSというイメージで捉えられることが多かった。公開されたソースコードを入手し、ユーザーが自らコンパイルしていた時代から比べれば、バイナリパッケージとしてディストリビューションが配布されている現在は、インストールと設定は格段に楽になっている。ハードウェアの自動検出もサポートされ、この点でもユーザーの負担が軽減は軽減された。しかし、デスクトップ用途ということを考えると、まだ不十分な面残されている。ここでは、Linuxがデスクトップ用途で普及していくための課題としてユーザーインターフェイスについて考えてみたい。

インストール

Red Hat系のディストリビューションの多くは、新しいリリースで「Anaconda」というグラフィカルなインストーラを採用した。少なくともマウス操作によるGUI環境がユーザーに提供されたわけだ。しかし実際には、以前のテキストベースのインストーラをGUIに置き換えただけのもので、設定項目やメニューの遷移といったユーザーインターフェイスの基本部分は変更されていない。

今回取り上げた中では、OpenLinux、Storm Linux、Corel LINUX OSが独自のインストーラを採用しており、改善の可能性を示している。今後、そのほかのディストリビューションでも、改良されていくことを望みたい。

インストーラが行うデフォルトの設定についても改善が図られている。Red Hat 6.2では、起動するネットワークサービスを制限することで、知らないうちにセキュリティ上の問題を抱えてしまうという問題への対応がなされた。TurboLinuxもこの点に配慮して、ネットワークサービスのデフォルト設定を変更している。

デスクトップ環境

GNOMEやKDEをはじめとする統合デスクトップ環境の登場によって、LinuxのGUI環境は大きく前進した。ユーザーによるカスタマイズの自由度や操作性ではWindowsと遜色ないレベルになってきている。

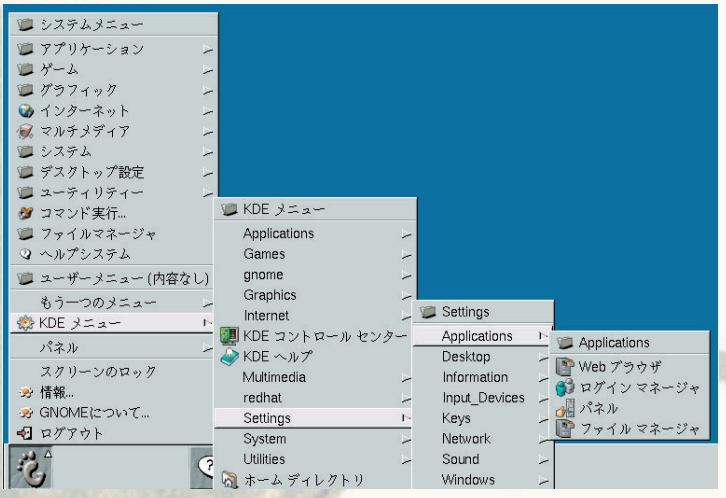
ただ、デフォルトのメニュー構成にはやや難点がある。どこに何が配置されているのか把握しずらいのだ。たとえばGNOMEを使用する場合に、同じマシンにKDEがインストールされていると、GNOMEのメニューにKDEのメニューが取り込まれる。両方の環境をシームレスに利用できて便利なように思えるが、メニューの階層が複雑すぎてわかりずらい。初めて使うユーザーはとまどうのではないだろうか。

メニューについては、同じ設定項目に対して、異なるツールが使用可能になっていたり（tksysvとksysvなど）、ほぼ同等の機能を持ったアプリケーションがメニューに複数ある（複数のターミナルなど）といった場合もある。

初期状態はなるべくシンプルで、よく使われるアプリケーションなどの必要最低限のものだけがメニューに登録されているほうが、どこに何があるか混乱することなくスムーズに使い始められるはずだ。その後、ユーザーの側で、スキルの向上に合わせてカスタマイズしていけばよい。

標準のデスクトップだけでなく、インストールされる各デスクトップ環境ごとに、きちんと初期設定されていれば、初心者でも好みのデスクトップを安心して使えるようになるはずだ。

GUIはユーザーとシステムをつなぐ大切な窓口である。今後も、さらに改良されていくことを期待したい。



画7　GNOMEメニューの中のKDEメニュー
メニュー階層が深すぎて複雑化してしまっている。このメニュー階層内だけで4つの「アプリケーション」メニューがある。

Linuxサーバ系徹底比較

最近のディストリビューションでは、サーバ系といわれるものがいくつか発売されるようになった。Linux（というかUNIX）には、サーバ機能が含まれているので、Windows NTやWindows 2000のようにWorkstationとServerといった明確な機能の違いはない。もちろん、サーバと銘打って販売されていないディストリビューションにもサーバとしての機能はきちんと備わっている。それでもサーバとして販売されているディストリビューションがあるのは、Workstation版に比べて、セキュリティ設定や、パッケージのインストール内容などがサーバ用にチューンされているからである。これらのディストリビューションを使えば、最小限の設定でサーバマシンを構築できる。もちろん設定だけではなく、UPSドライバなどサーバを運用するうえで役立つソフトがバンドルされている。

もともとサーバ版といわれるディストリビューションは昨年3月18日に発売されたTurboLinux Server日本語版1.0が最初である。これを機に、各ディストリビューターからサーバ版が発売されるようになった。サーバといっても、家庭内や小規模オフィスで使用されるようなSOHO向けのものから、ISPやASPが使用するようなeビジネス向けのサーバまで幅広いものがある。今回取り上げる4つのサーバ版は、どちらかというとeビジネス向けのディストリビューションだが、もちろん小規模なサーバとしても使用できる。

サーバ系の5つのキーワード

サーバ系に求められる機能をいくつか挙げてみよう。

- **大容量メモリサポート**
- **セキュリティ**
- **Raw I/Oモード**
- **クラスタリング**
- **遠隔操作**

これらはデスクトップ用途ではほとんど使われない機能だ。逆にいえば、サーバ系ディストリビューションの要といえるだろう。それではこれらの機能の概要を順に説明する。

大容量メモリサポート

エンタープライズ向けサーバには、高速なレスポンスが必要になる。そのため、プログラムやデータ領域がハードディスクにスワップされずにメモリ上で動作することが要求される。また、大容量のメモリを搭載することで、大量の同時アクセスを扱うことができる。このような理由で、サーバ系のディストリビューションでは、大容量メモリをサポートする必要がある。

セキュリティ

ここでいうセキュリティとは、サーバのポートやサービスを閉じるという意味ではなく、eビジネスソリューションに関するセキュリティを指す。

eビジネスでは、Web上での決済がベースとなるが、その際には暗号化などが必須となる。これらの機能はデスクトップ用途には必要ないものだ。そのため、サーバ系ディストリビューションは、どれもeビジネス向けにセキュ

ターボリナックス ジャパン株式会社最初にサーバ系を発売したディストリビュータである。TurboLinux Server日本語版6.1を発売している。
http://www.turbolinux.co.jp/

レッドハット株式会社
http://www.redhat.com/jp/
老舗のディストリビューターである。Red Hat Linux 6.2Jプロフェッショナルを発売している。

リティ機能が強化されている。ただ、その内容は各ディストリビューションごとにセキュリティライブラリなどが異っている。

Raw I/Oサポート

データベースのように頻繁にディスクにアクセスするアプリケーションの、信頼性とパフォーマンスを向上させるため、Raw I/Oがサポートされるようになっている。Raw I/Oは、アプリケーションがカーネルのディスクバッファを使用せずに直接ハードディスクに書き込むことで、高い信頼性とパフォーマンスを実現する。このような機能はデスクトップの用途としては必要がないし、SOHOサーバ程度ではあまり利用されない。

クラスタリング

クラスタリングはLinuxのもっともホットな話題のひとつだ。クラスタリングとは、ネットワークで接続された2台以上のマシンを使って、フェイルオーバー（障害復旧）と、ロードバランシング（負荷分散）を行うテクノロジーだ。外部からは複数のサーバが1台のマシンであるかのように機能する。1台のマシンに障害が発生したとしても、他のマシンが処理を継続するのでサーバ機能が完全にダウンすることがない。また、負荷の高いサーバの処理を分散することで、高いレスポンスを提供することができる。このような高い可用性とスケーラビリティはエンタープライズのサーバに要求される。

クラスタリング機能は一部のディストリビューションに装備されており、今後のサーバ系ディストリビューションでは標準的な機能となっていくだろう。

遠隔管理

また、どこにいてもサーバ状態を設定できるように、遠隔地からのリモート設定機能を強化しているものもある。24時間オンサイトのサーバを管理するのは非常に大変だ。常に管理者を待機させるのはコストがかかる。そこで、遠隔地からでもサーバ状態をコントロールできるようにしてあるのだ。遠隔地から操作する場合、セキュリティが心配されるが、もちろんそれも考慮されている。

紹介ディストリビューション

今回ここで取り上げるサーバ版は、次の4つである。

- **TurboLinux Server日本語版6.1**
- **Red Hat Linux 6.0Jプロフェッショナル**
- **OpenLinux eServer 2.3日本語版**
- **LASER5 Linux 6.0 Server**

どれも人気ディストリビューションのサーバ版という位置づけだ。このほかにも今回の記事には間に合わなかったが、Kondara MNU/Linux Server 1.1がデジタルファクトリから発売される（6月8日発売）。詳しくは、オフィシャルページ（http://www.kondara.org/）を参照してほしい。

マニュアル

サーバ版は通常のディストリビューションと比べて、システム管理者が利用するものと考えられているためか、初心者向けのマニュアル類が充実していないことが多い。商用ソフトなどのインストール方法に関しても詳しく記述されていなかったり、CD-ROMに英語ドキュメントが収録されているだけだったりする。これは、ワークステーション版ほど数が出ないためかもしれないが、できれば改善してほしいところだ。せめて、商用アプリケーションのインストール方法や、セキュリティがどのように設定されているのかを詳しく解説したマニュアルをバンドルしてほしい。

株式会社ネオナジー
http://www.openlinux.ne.jp/
Caldera Systems, Inc.のOpenLinuxを販売している。

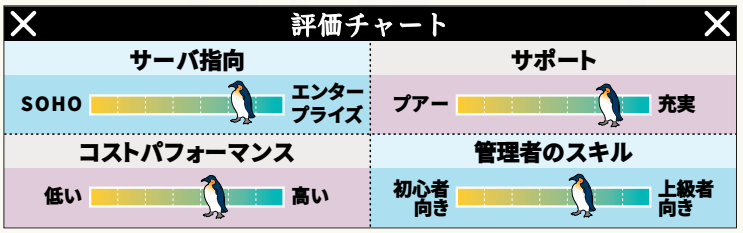
レーザーファイブ株式会社
http://www.laser5.co.jp/
日本のディストリビュータの老舗である。以前はRed Hat Linuxの日本国内向けのディストリビュータだった五橋研究所の子会社だ。

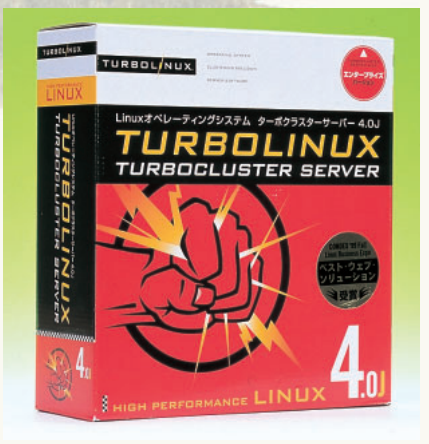
ひとつ未来をみすえた

TurboLinux Server 日本語版 6.1

ターボリナックス ジャパン株式会社
03-5766-1660
http://www.turbolinux.co.jp/

3万9800円





TurboLinux Server日本語版6.1（以下、Turbo Server 6.1）は、ターボリナックス ジャパンから6月16日に発売される、セキュリティ対策を充実させたディストリビューションだ。安全なサーバを構築するための機能が充実しているのが特徴だ。

ターボリナックス ジャパンは、以前からデスクトップ系とサーバ系を明確に差別化して提供しているディストリビューターで、デスクトップ系はWorkstationという名前で販売している。

なお、今回の記事ではβ3を使用しているので、製品版とは若干異なる可能性がある。

カーネル

カーネルのバージョンは2.2.15、標準Cライブラリにglibc 2.1.3と最新のものを使用している。これはTurboLinux Workstation 日本語版6.0よりも新しい（カーネル2.2.13、glibc 2.1.2を採用）。

また、Raw I/Oモードにおける Asynchronous I/Oをサポートしている点に注目したい。Raw I/Oモードは、データベース使用時の信頼性とパフォーマンスを向上させるものだが、通常は同期書き込みを行っている。Asynchronous I/Oは、ディスクの読み書きを非同期に行うことで、ディスクアクセスのパフォーマンスが大幅に向上する。

さらに、サーバでは大容量のメモリサポートが求められるが、最大4Gバイトのメモリに対応しているので、Webサーバなどを大量に同時実行させても安心だ。

インストーラ

Turbo Server 6.1のインストーラは、TurboLinux Workstation 日本語版6.0と同等のもので、キャラクタベースのものである。しかし、インストーラが商用バンドルソフトや、商用フォントを一緒にインストールしてくれるという非常にありがたい仕組みになっている。このため、他のサーバ系ディストリビューションのようにインストール後にひとつひとつインストールしていくような煩わしさがない。このような機能を持っているのは今回紹介するサーバ系4ディストリビューションの中では、Turbo Server 6.1が唯一である。

セキュリティ

セキュリティ対策のため、セキュリティホールになりやすいポートとサービスはすべて閉じられている。また、もっともクラックされやすいバッファオーバーフロー対策にもぬかりはない。さらにOpenSSHを使うことで、遠隔地からtelnetでリモートコントロールする場合のセキュリティも万全だ。

eビジネスが普及するに従い、eコマースでの決済に対する暗号化の強化も求められている。多くのディストリビューションではOpenSSLを採用しているのに対し、Turbo Server 6.1ではより解読されにくい128ビットRSA BSAFE SSLライブラリを使用し、安全性を高めている点が大きく異なる。

なお、後述するTurbo Cluster Server 4.0Jと組み合わせれば、大規模なeコマースサイトの構築も可能となる。

商用バンドルソフト

商用ソフトには、BRU（バックアップユーティリティ）、HDE Linux Controller 1.0（Webベースシステム管理ツール）、Norton Ghost（ディスクイメージバックアップツール）、リョービ日本語TrueTypeフォント5書体、EnlightenDSM（システム管理スイート）と、前述したRSA BSASE SSLを収録している。決して多くはないが、サーバ運営に必要な十分なソフトウェアが揃っている。

サポート

サポートは、90日間3件まで（電話、

Web、メール）のサポート権が付属している。このサポート権には、インストールだけでなくインターネットサーバの設定までが含まれている。また、新たに発見されたセキュリティホールをユーザーにメールで通知するなど、安心のサポート体制だ。

さらに、年4回のメンテナンスCDの無償送付サービスが付属している。なお、サーバの構築から運用までをカバーする有償サービスも用意されている。

クラスタリング

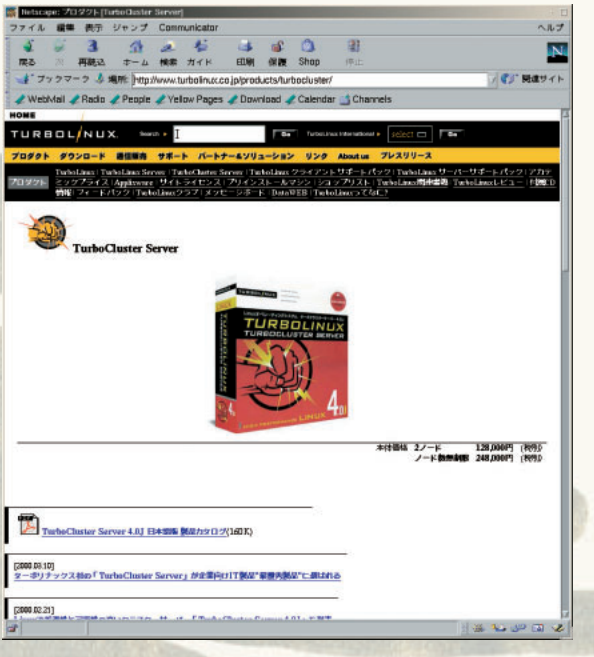
Turbo Server 6.1ではクラスタリングをサポートしていないが、Turbo Cluster Server 4.0J（以下、Cluster 4.0J）を使えば、高性能なクラスタリングが実現できる。Cluster 4.0Jのクラスタリング機能は、Linux Virtual Server Projectで開発していたものをベースに、ターボリナックスが独自に開発したもので、フェイルオーバーとダイナミックロードバランシングを実現している。

フェイルオーバー機能は、クラスタノードに障害が発生した際に、自動的にノードを切り替えて対処する機能で、ダイナミックロードバランシングは、クライアントからのトラフィックを自動認識して最適なサーバに処理を分散し、ピーク時の負荷を軽減する機能だ。いずれも信頼性の高いサーバを実現するうえで重要な機能だ。

また、独自のクラスタ管理／設定ソフトであるTCSWATが付属している。TCSWATはWebベースのソフトで、クラスタのステータスを表示したり、構成を変更したりすることができる。このソフトを使えば、リモートでも簡単に管理作業が行える。

Cluster 4.0Jは、「J」とはなっているものの、実際の中身は英語版である。インストーラもマニュアルも英語版そのものだ。konコマンドも、ktermも収録されていないため、日本語の表示はできない。Cluster 4.0Jに付属しているディスク2の「アップデートCD」にクラスタリングのテストに関する日本語のドキュメントが収録されているが、このドキュメントも見ることができなかった。リモートで使うことを前提にしているのかもしれないし、事実それで不便はないが、コンソールで使う場面もないとはいえないので、最低限の日本語化はほしいところだ。

画面1　ターボリナックス ジャパンのホームページ
クラスタ機能についての解説がある。



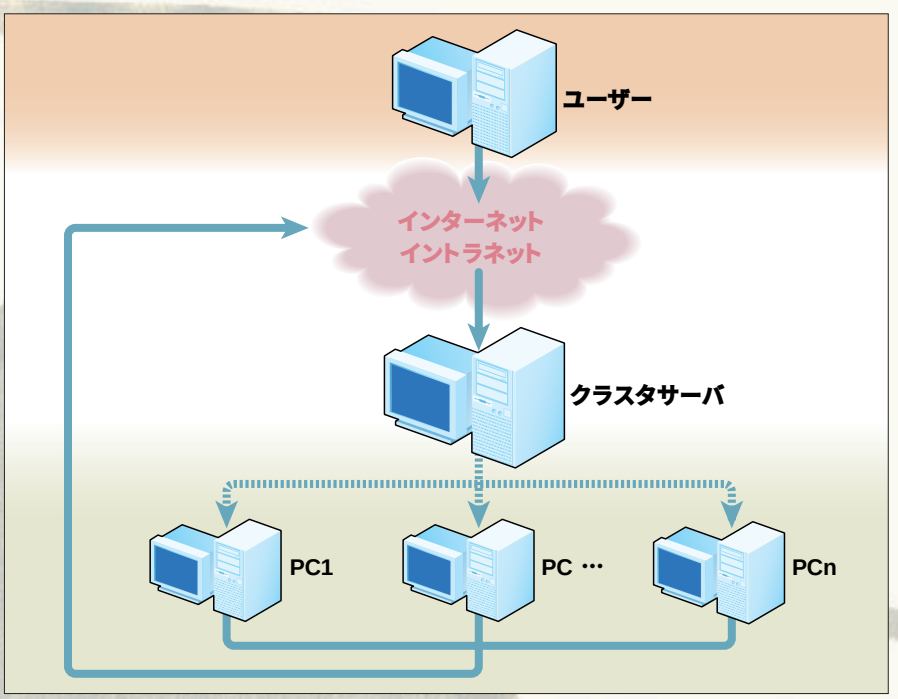


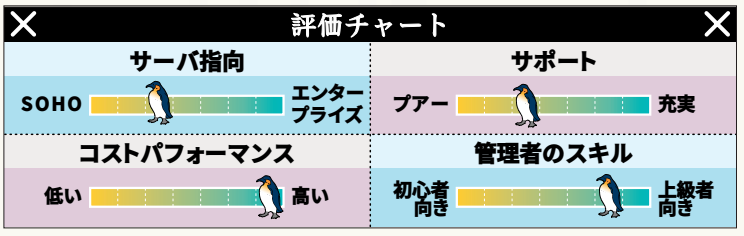
図1　クラスタの概念図
クラスタサーバがユーザーからのリクエストを分散する。

玄人好みの総本山

Red Hat Linux 6.2J プロフェッショナル

レッドハット株式会社
03-3257-0411
http://www.redhat.com/ja/

2万9800円

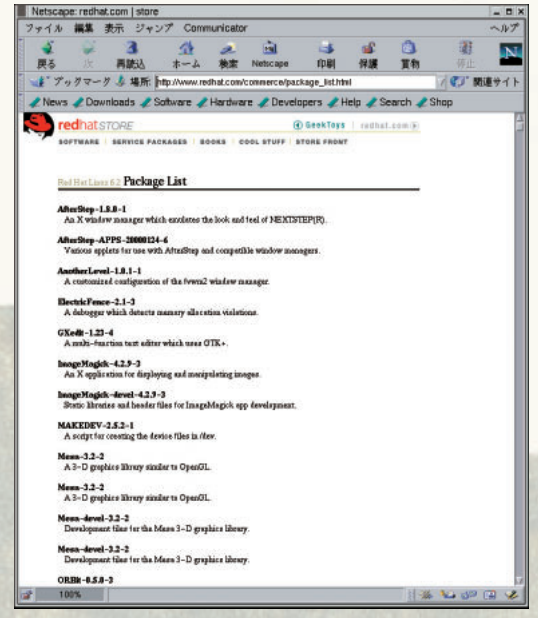


Official Red Hat Linux 6.2日本語版プロフェッショナル（以下、Red Hat 6.2 Pro）は、先日発売されたOfficial Red Hat 6.2日本語版のサーバ版だ。これまでレッドハットでは、デスクトップ系とサーバ系とを明確に分けていなかったが、この6.2からRed Hat 6.2 Proを発売し、サーバ系として位置づけるようになった。スタンダード、デラックスとの大きな違いは、セキュアWebサーバの対応やUPSなど、サーバ系ソフトのバンドルだ。

カーネル

Red Hat Linux 6.2Jでは、3種類のバージョンを用意しているが基本的なスペックは同じである。プロフェッショナルでもスタンダードと同一のCD-ROMが付属しており、まずこのCD-ROMを使ってインストールする。したがって、インストール直後は、スタンダードと変わらない。サーバパッケージはその後からインストールすることになる。このような構成のため、カーネルのバージョンは2.2.14、標準Cライブラリにglibc 2.1.3とデスクトップ系と全て同じ構成だ。

また、Raw I/Oモードと4Gバイトまでのメモリの自動認識をサポートしている。これもプロフェッショナルのみの機能というわけではなく、スタンダードでも利用できる機能で、サーバ系独自の機能というわけではない。



画面1　Red Hat Linux 6.2パッケージリスト（US）
US版に収録されているパッケージリストと簡単な説明を見ることができる。
http://www.redhat.com/commerce/package_list.html

インストーラ

インストーラには、トレンドであるAnacondaを使用したGUIを採用しており、非常にわかりやすい。ただし、これは前述したように全てのバージョンを同一の構成にするためで、インストールできるのはスタンダードの機能だけである。プロフェッショナルではこの後にバンドルソフトをインストールすることになるが、こちらのインストールはあまりわかりやすいとはいえない。必要なパッケージを自分でひとつひとつインストールしていくことになる。もちろん、対象がシステム管理者ということなのでこれでも問題はないのかもしれないが、Windows系の技術者などへの配慮があってもよかったかもしれない。もっとも、慣れるまでは苦労するかもしれないが、慣れればこちらのほうが使いやすいのだろう。

ちなみに、IBM DB2はテキストベースのインストーラが付属している（**画面3**）。

なお、パッケージの詳しい内容はホームページを参照するといいだろう。英語だが、パッケージの名前と簡単な説明が付いている。

画面2　IBMのDB2のホームページ
http://www.jp.ibm.com/software/data/udb/index.html
PCから汎用機に至るまで一貫したシステムを提供

画面3　DB2のインストール
たとえCUIでもインストーラがついているとありがたい。

セキュリティ

スタンダードと基本構成は同一であるため、ポートやサービスは完全に閉められていない。telnetなどもデフォルトの状態で使えるようになっているので、各自必要ないポートやサービスを閉じる必要がある。

eコーマスに対するWebサーバセキュリティは、Apacheのmod_sslと、OpenSSLライブラリを利用している。

クラスタリング

Red Hat Linux 6.2Jでは、標準でクラスタリングをサポートしている。この機能は、スタンダードでもサポートされているものだ。クラスタリングはインストーラのカスタムインストールから選択することでインストールできる。Turbo Cluster Server 4.0Jと同様に、フェイルオーバー機能とロードバランシング機能を実現し、サーバの信頼性とトラフィックの軽減を行う。

クラスタリングのアーキテクチャに関しては、Turbo Cluster Server 4.0Jとの違いはない。Red Hat 6.2Jのクラスタリングは、Linux Virtual Server Projectで開発されていたものであり、もともとターボリナックスも開発に参加していた。しかし、ターボリナックスはクラスタリングのスループットを向上させるためにフレームワークを見直し、独自開発を行った。このようにして同一のアーキテクチャのクラスタが2つ生まれることになったのだが、ターボリナックスはAdvanced Traffic Manager（turboclusterd）をGPLで公開せず、再配布も許可しないということなので、オープンソースということにはならないようだ。一方、Red Hat 6.2JのクラスタはGPLで公開されているのでこちらが一般的になるだろう。

商用バンドルソフト

商用ソフトには、HDE Linux Controller、Jun for Java、ARC Power Chute（UPS管理ソフト）、パワーバイザ（UPS管理ソフト）、FREQUPS（UPS管理ソフト）、Comprehensive Perl Archive Network（CPAN。パールスクリプト集）、IBM DB2（英語版）がバンドルされている。DB2は、PCから汎用機に至るまで、一貫したシステムを提供するデータベースだ。インターネットにおけるスケーラビリティの高いデータベースとして実績がある。

そのほか、デラックスに付属しているソフトも全て付属している（詳しくは、本誌52ページのRed Hat 6.2Jを参照）。さらに、US版に収録されているソフトもそのまま収録されているので、バンドルしているソフトが最も多いディストリビューションでもある。US版のバンドルソフトの詳細は、レッドハットのホームページ（**画面1**）に記載されている。

サポート

サポートは、90日間のメールと、30日間の電話によるインストールサポートを行っている。また、180日間のアップデートプログラムへの優先アクセス権が付いている。

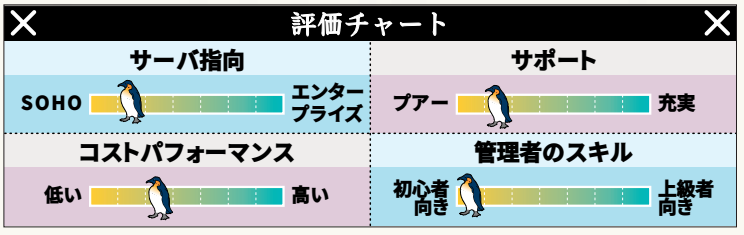
インストール後のサポートに関しては、ワールドワイドサポートグループが企業向けにApache、Samba、Sendmailのサポートを行うということが発表された。詳細は不明だが、ワールドワイドサポートグループということで、英語サポートになると思われる。

独自の路線で使いやすく

OpenLinux eServer 2.3日本語版

株式会社ネオナジー／Caldera Systems, Inc.
03-3252-4300
http://www.openlinux.ne.jp/

2万9800円

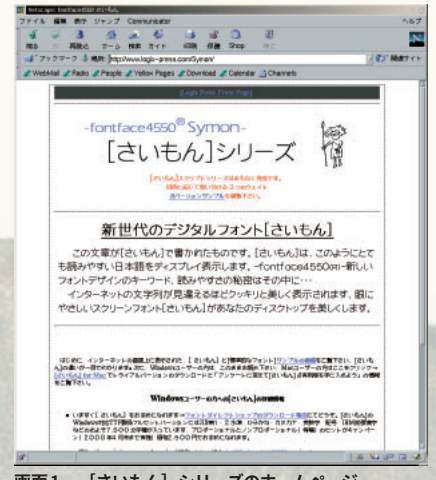


OpenLinux eServer 2.3日本語版（以下、eServer 2.3）は、eビジネス向けサーバだ。OpenLinuxは米国で非常に評価の高いディストリビューションだ。開発はNovelの子会社であるCaldera Systems, Incが行っており、日本語化されたのはつい最近である。販売は株式会社ネオナジーが行っている。

以前からKDEを標準デスクトップ環境に採用しており、サーバ系といえども使いやすいディスクトップ環境が提供されている。また、COASという独自のGUIによる管理ツール群を使用する点にも特徴がある。

カーネル

カーネルは2.2.14、標準Cライブラリは2.1.2と比較的新しいものを採用している。同じバージョンのOpenLinux 2.3日本語版が採用しているカーネル2.2.10、glibc 2.1.1よりも新しいものだ。さらに、大規模サーバ向けの大容量メモリとRaw I/Oなどをサポートするためにカーネルにパッチをあてて独自拡張を行っている。ただし、4Gバイトまでのメモリを使用するには、カーネルの再構築が必要になる。



画面1　［さいもん］シリーズのホームページ
http://www.logix-press.com/Symon/
書体のサンプルをみることができる。

インストーラ

インストーラには、OpenLinux独自のLizardが使われている。GUIのグラフィカルなインストーラで非常にわかりやすい。さらに、細部まで細かい配慮がされており、インストールの煩わしさを軽減してくれる。インストールパーティションなどはツリー表示するので、マウントポイントを分ける場合など直感的で間違えない。非常によくできたインストーラだ。

また、Lizardの面白い特徴だが、パッケージのインストール中にテトリスライクなゲームで遊べるようになっている。本当にこのような機能が必要なのかどうかは賛否両論だろうが、新しいアイデアとして注目したい。

バンドルソフトのインストールはeServer 2.3のインストール後に個別に行うことになる。インストール方法はマニュアルには書かれておらず、CD-ROM内に収録されているドキュメントを参照する必要がある。商用フォントもRPMパッケージのみで提供されており、インストーラは付属していない。また、設定ファイルをエディタで修正しなければならないなど、やはりシステム管理者向けという印象だ。商用ソフト1本、トライアル版1本、試用版1本とバンドルしているソフトも少ないので、今後はインストーラを用意してもいいのではないだろうか。

セキュリティ

標準状態のセキュリティはそれほど厳しくはなっていない。telnetなどのポートやサービスも閉じられていないので、実際にサーバを運用するには、マニュアルを熟読してセキュリティを設定することになる。また、OpenSSLなどは収録されていないので、eコーマスサーバを構築する場合は別途用意する必要がある。

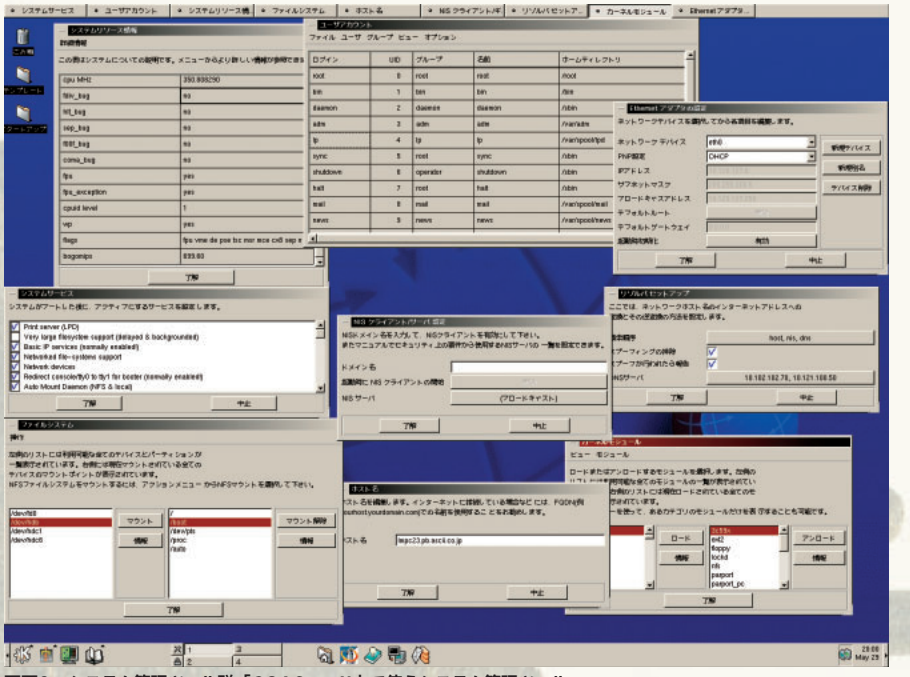
商用バンドルソフト

商用ソフトには、IBM Visual Age for Java for Linux（Java統合開発環境）のみが収録されている。また、商

用フォントとしてFontface4550 Symon［さいもん］バージョン1.2が収録されている。そのほか、IBM Web Sphere Application Server Standard Edition, V2.03 for Linux 90日試用版が収録されている。

商用バンドルソフトは1本と少ないが、管理ツールなどのソフトがバンドルされていないわけではない。管理ツールはCaldera Systems, Incが開発したオープン管理システムであるCOAS（**画面2**）と、ブラウザベースのリモート管理ツールであるWebmin（**画面3・4**）が収録されている。COASは、X上で使用する管理ツールで、ネットワークの設定やデーモンの実行、ファイルシステムの操作などが行え、非常にわかりやすいツールだ。Webminは同様のツールだが、HDE Cotrollerと同じく、Webブラウザから設定を行うためのツールだ。Webminを使えば、ネットワーク経由でサーバの設定が可能になる。

そのほかにもPostgreSQLや、ほかのディストリビューションでは珍しい、MySQL（リレーショナルデータベース）などサーバ用途のソフトが多数収録されている。



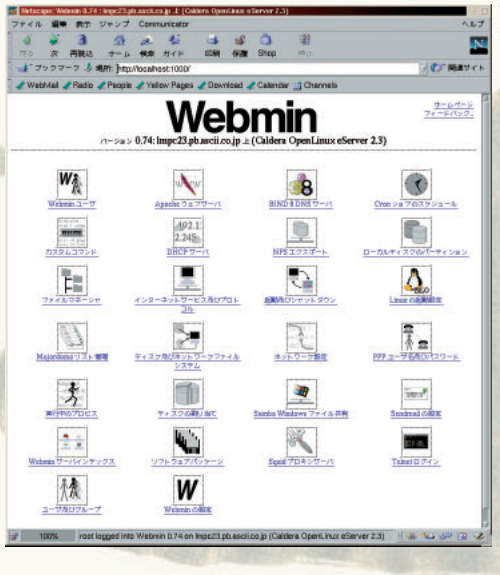
画面2　システム管理ツール群「COAS」　X上で使うシステム管理ツール。

マニュアル

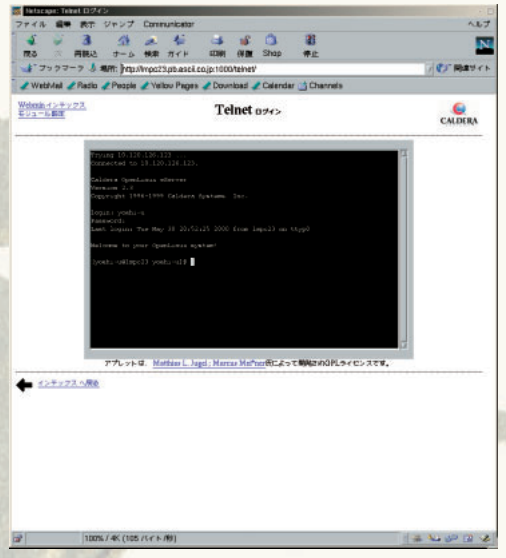
マニュアルは「システム管理者ガイド」1冊のみだが、インストールからサーバ運用、セキュリティまで詳しく解説されている。構成もよくできており、非常にわかりやすい。特に、NetwatchやtcpdumpやNetwatchを用いたネットワークの監視や、パケットフィルタリングの解説など有用な情報も網羅されている。なかなかよく出来たマニュアルだ。初めてサーバを構築する人でもマニュアルをじっくり読めばひととおりの設定ができるだろう。

サポート

サポートは、30日間の電話によるサポートまたは、90日間までのメールサポートを最大5件まで行っている。サポート内容はインストールまでだ。



画面3　システム管理ツール「Webmin」
Webベースでシステムを管理するツールだ。



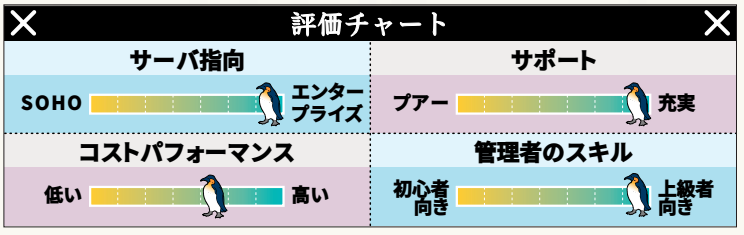
画面4　Webminでのシステム管理
Webブラウザ内からtelnetでリモートログインしている。

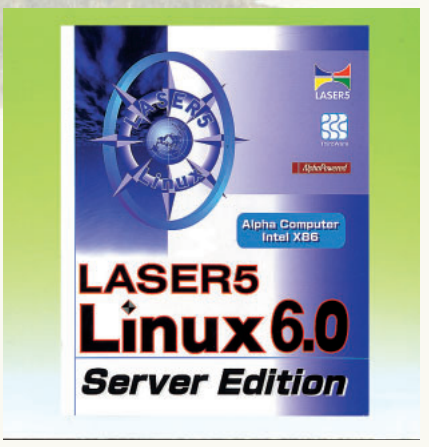
eコーマスもお手のもの

LASER5 Linux 6.0 Server Edition

レーザーファイブ株式会社／株式会社サードウェア
03-3818-6626
http://www.laser5.co.jp/

4万9800円





LASER5 Linux 6.0 Server Edition（以下LASER5 6.0 Server）は、もともとRed Hat Linux 6.0 Server Editionとして発売されていたディストビューションである。レーザーファイブのRed Hat社との商標権の使用期限が切れたために、LASER5 6.0 Serverとしてリファインして発売されたものだ。基本的な機能はRed Hat Linux 6.0 Server Editionと変わらないが、新たにいくつかの商用ソフトをバンドルした。また、Alpha版も同梱されている点が大きく異なる。今回紹介するサーバ系ディストリビューション中、唯一Alpha版が含まれている。

LASER5 6.0 Serverは、今回紹介するサーバ系4ディストリビューションの中で、もっともエンタープライズ向けの商品であり、ISP/ASP向けのディストリビューションといえるだろう。

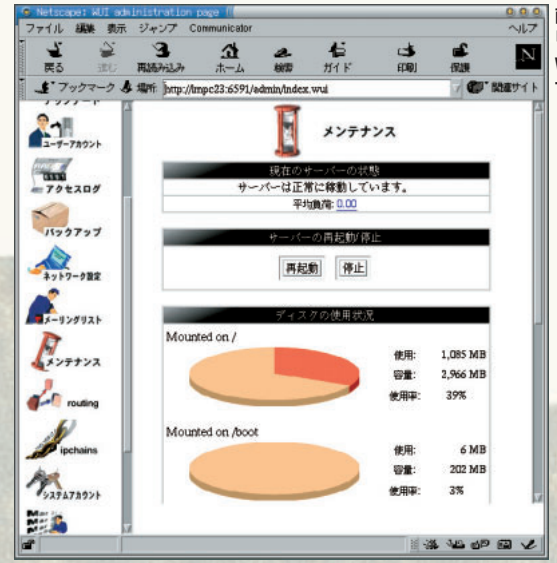
カーネル

基本的な構成はRed Hat Linux 6.0 Server Editionと変わらない。カーネルも2.2.5、標準Cライブラリにglibc2.1.1と同じ構成だ。Alpha版も同様の構成である。今となっては若干古いという感は否めないが、サーバとして安定した動作を望むのであればかえって安心といえるかもしれない。なお、カーネル2.2.14（セキュリティパッチ済み）を採用したLASER5 Linux 6.5 SecureServer Editionが7月に発売される。価格は4万9800円（予定）。

インストーラ

インストーラには、少し前まで使われていたRed Hat系のCUIベースのインストーラが使われている。使いやすいとはいえないが、以前からのユーザーは違和感なく使用できるだろう。

LASER5 6.0 Serverは、多くのバンドルソフトが付属しているが、やはりバンドル版のソフトはひとつひとつインストールしていくことになる。CD-ROMに収録されているソフトの一覧が添付されており、おのおののソフトのインストール方法は、各ディレクトリのドキュメントを参照する。慣れないとなか



画面1　システム管理ツール「HDE Linux Controller」Webminと同様にWebブラウザでシステムを管理する。

画面2　HDE Linux Controllerでサーバの設定を変更遠隔地から自由しサーバを設定できる。

なか大変な作業なので、初めてインストールする場合は1日作業になる。

なお、標準ではX Windows Systemの設定はされていない。リモートで使うことを前提にしているためだ。ISPやASPがeビジネス用のサーバとして設置するような場合、「X Windows Systemのような無駄なリソースを使いたくない」、「キャビネットに納めた状態で使用するのだからリモートログインしか使わない」、「X Windows Systemの導入はセキュリティホールを生む」といった考えが前提にあるようだ。レーザーファイブの、サーバ用途でX Window Systemは使わないというポリシーは明確で、7月に発売されるLASER5 Linux 6.5 SecureServer Editionは、X Windows Systemを収録しないということだ。しかし、使い方はユーザー次第なので、インストーラで選択できるようにしてほしいと思うユーザーも多いのではないだろうか。

セキュリティ

セキュリティポリシーは、インストール時に選択するサーバのタイプによって最適化される。セキュリティマネージャを使えば、パケットフィルタリングによるファイアウオール設定が簡単に行え、インターネット経由の不正アクセスを防止することができるようになっている。さらに、SSHを使うことでtelnetを使って安全にサーバをコントロールすることができる。

商用バンドルソフト

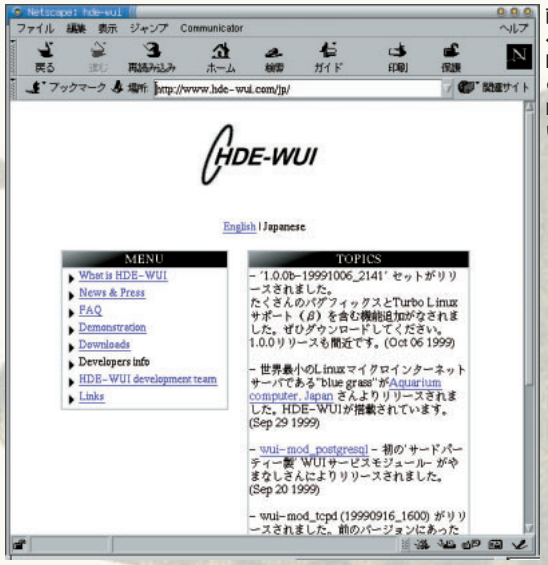
商用ソフトには、BRU、HDE Linux Controller 1.0が付属している。それ以外にも多くの非商用版ソフトや試用版、体験版が収録されている。UPSドライバはAPS、アルファテック、関西蓄電池、三菱電機など、多くのUPSに対応したものが収録されている。また、強力な128／168ビットの暗号をサポートするRoxen Challenger 1.3が収録されている（現在はRoxen WebServerに名称変更）。Roxen Challenger 1.3を使えば、安全なWebサイトを構築することができる。

マニュアル

マニュアルは非常に充実しており、「サポートガイド」、「ご利用案内」、「インストールガイド」、「Alpha版インストールガイド（追補）」、「サーバリファレンス」の計5冊が付属している。総ページ数は300ページ超という立派なものだ。インストールからサーバ運用までを図入りで詳しく解説している。これらのマニュアルは、バインダーにCD-ROM/FDメディアと一緒に綴じられていてとても使いやすい。

サポート

サポートは、各種サーバ系ディストリビューションの中で、最も充実しており、インストールサポートだけでなく、サーバ機能の設定や、運用方法までをカバーしている（Alpha版はインストールからサーバ設定まで）。サポート期間は半年間と長く、電話、FAX、メールで行う（インストールサポート3件まで、それ以外は2件まで、合計3回まで）。さらに、延長サポートサービス（1年間5件までのフルサポート。12万円）と、追加サポート(5件までのフルサポート。5万円）が用意されている。さすがに老舗だけあって、長年のノウハウの蓄積があってこそできるサポートだろう。安心のバックアップ体制だ。



画面3　HDE-WUIのホームページ
http://www.hde-wui.com/jp
HDE-WUIはHDEが開発している管理ツールである。

画面4　Enhanced Software Technologies, Inc.のホームページ
http://www.hde-wui.com/jp/
BRUは、ESTが開発しているバックアップツールである。今後、このようなツールが増えるだろう。

製品名		TurboLinux Server 6.1	TurboCluster Server4.0J
	販売元	ターボリナックスジャパン株式会社	ターボリナックスジャパン株式会社
	URL	http://www.turbolinux.co.jp/	http://www.turbolinux.co.jp/
	提供メディア	CD4枚/FD2枚	ＣＤ２枚/FD2枚
	マニュアル	3冊	1冊（英語）
	価格	3万9800円	2ノード：12万8000円 ノード無制限版：24万8000円
動作条件			
	CPU	Pentium相当以上	i486DX以上（Pentium相当以上推奨）
	メモリ	32Mバイト以上（64Mバイト以上推奨）	32Mバイト以上 （128Mバイト以上推奨）
	ハードディスク	600Mバイト以上推奨 （最低200Mバイト）	150Mバイト〜700Mバイト
インストール			
	インストーラ	日本語／テキスト	英語／テキスト
	ブートメディア	CD-ROM、FD、DOS	CD-ROM、FD、DOS
	インストール元メディア	CD-ROM、NFS	CD-ROM、HDD、NFS、FTP
システム構成			
	パッケージ管理システム	RPM	RPM
	デスクトップ環境	GNOME、fvwm、fvwm2	TurboDesk/AfterStep、fvwm、 fvwm2、twm
	日本語入力システム	Wnn6 Ver.3、Canna	—
バージョン			
	カーネル	2.2.15	2.2.12
	標準Cライブラリ	glibc 2.1.3	glibc 2.0.7
	XFree86	3.3.6	3.3.5
	GNOME	1.0.55	—
	KDE	—	—
設定ツール			
	ユーザー管理	—	turbousercfg
	ネットワーク	turbonetcfg	turbonetcfg
	Xサーバ設定	turboxcfg、XF86Setup	turboxcfg、XF86Setup
	ウィンドウマネージャ変更	turbowmcfg	turbowmcfg
	サウンド	—	—
	起動デーモン	turboservice	turboservice
	パッケージ管理	turbopkg	turbopkg
主なバンドルソフト			
	日本語入力システム	Wnn6 Ver.3	—
	日本語TrueTypeフォント	リョービ和文5書体	—
	ブートマネージャ	—	—
	そのほか	BRU、HDE Linux Controller、Norton Ghost	BRU
サポート			
	サポート内容	インストールおよび、 インターネット系サーバ設定	インストールからサーバ設定まで
	サポート方法	メール、Web、電話	メール、Web、電話
	サポート期間	90日	90日
	サポート件数	3件	3件
	備考	セキュリティホール発見時の緊急メールアナウンス、年間4回のメンテナンスCD無償送付	—

ディストリビューション戦国時代

	Red Hat Linux 6.2J Professional	OpenLinux eServer 2.3 日本語版	LASER5 Linux 6.0 Server Edition
	レッドハット株式会社	株式会社ネオナジー	レーザーファイブ株式会社
	http://www.redhat.com/jp/	http://www.openlinux.ne.jp/	http://www.laser5.co.jp/
	CD8枚/FD1枚	CD3枚	ＣＤ6枚/FD4枚
	2冊	1冊	5冊
	2万9800円	2万9800円	4万9800円
	x86プロセッサシステム	Pentium以降	i486相当以上またはAlpha Chip
	16Mバイト	64Mバイト以上 （128Mバイト以上推奨）	16Mバイト以上（Intel） 128MB以上（Alpha）
	500Mバイト（推奨）	40Mバイト以上 （フルインストール時1.3Gバイト）	800Mバイト以上推奨
	日本語／グラフィカル	日本語／グラフィカル	日本語／テキスト
	CD-ROM、FD、DOS	CD-ROM、FD、DOS	CD-ROM、FD、DOS
	CD-ROM、HDD、NFS、FTP、HTTP、PCMCIA	CD-ROM、NFS	CD-ROM、ＨＤＤ、NFS、FTP、HTTP
	RPM	RPM	RPM
	GNOME、KDE、fvwm、fvwm2、Window Maker、Enlightenment、AfterStop	KDE、twm	GNOME、KDE、fvwm、fvwm2、Window Maker、Enlightenment、AfterStop
	ATOK12 SE、Wnn6 Ver.3、Canna	Wnn Ver.4、Canna	Canna
	2.2.14	2.2.14	2.2.5
	glibc 2.1.3	glibc 2.1.2	glibc 2.1.1
	3.3.6	3.3.5	3.3.3.1
	1.0.55	—	1.0.7
	1.1.2	1.1.2	1.1.2
	linuxconf、kuser	COAS、Webmin	linuxconf、kuser
	netcfg、linuxconf	COAS、Webmin	netcfg、linuxconf、netconf
	XF86Setup、Xconfigurator	XF86Setup	XF86Setup、Xconfigurator
	switchdesk-gnome	—	switchdesk-gnome
	sndconfig	—	sndconfig
	linuxconf	COAS、Webmin	linuxconf、tksysv、ksysv
	rpm、gnorpm	COAS、Webmin、rpm	gnorpm、ｒｐｍ
	ATOK12 SE、Wnn6 Ver.3	—	—
	DynaFont和文5書体	Fontface 4550 Symon［さいもん］1書体	—
	System Commander Lite	—	—
	Deluxeに加え、IBM DB2など多数	IBM VisualAge for Java for Linux	BRU Personal Edition、HDE Linux Controller
	インストールサポート	インストールサポート	指定なし
	メール、電話	メール、電話	メール、電話、FAX
	90日（メール）、30日（電話）	90日（メール）、30日（電話）	180日
	制限なし	—	インストール3件、その他2件、最大3件
	180日間の無料FTPサイト優先使用権	カルデラ・システムズのサポートデータベース（英語）	追加サポートあり

1G対決！　Linuxで速いのはどっちだ！

対決！最速マシン

2000年はPC界にとって歴史的な年となった。ついにCPUのクロックが1GHzを超えるという歴史的瞬間を迎えることができたのだ。まず、3月6日にAMDがAthlon 1GHzを発表すると、2日後の3月8日にはインテルがPentium III 1GHzを発表した。僅差でAMDが先に1GHzオーバーの名誉を手中に収めることとなった。

すでにクロック数ではAMDに先行されていたインテルだが、1GHzオーバーという名誉は手中に収めたかったに違いない。この時期、1GHz CPUの発表日の情報は二転三転し、両社の「歴史的瞬間」争いが激化していたことがうかがえる。また、両社とも発表と同時という迅速な出荷（インテルは限定出荷）にも歴史的瞬間にかける意気込みが感じられた。

10年前のPCといえば、まだNECのPC-9800シリーズ全盛期であった。CPUは80386DX 20MHz、メモリは1.6Mバイトという構成だった。クロックだけを単純に考えると、50倍の性能になっているということになる。わずか10年の間にすさまじい進歩を遂げたということをあらためて実感する。インテルとAMDのCPU開発戦争が、CPU高速化に拍車をかけたことは間違いない。

そして待ちに待った1GHzマシンが市場に出回るようになった。これら両雄の戦いの結果は…果たしてLinuxにおいてPentium III 1GHzとAthlon 1GHzはどちらが速いのかを徹底検証する。

Photo : Tetsu Ohgushi (Dee)

Gateway Select 1000

日本ゲートウェイ株式会社

価格：28万9800円（標準構成）
http://www.gw2k.co.jp/

Gateway Select 1000（以下、Select 1000）は世界で最初に発売された1GHzマシンだ。CPUにAMD Athlon 1GHzを搭載している。Athlon 1GHzを搭載したマシンは米国ではコンパックも販売しているが、日本国内で販売しているのは日本ゲートウェイだけである。Athlon 1GHzは、OEMメーカーへ優先的に出荷されているため秋葉原でも入手するのは困難であり、事実上日本で入手できる唯一のAthlon 1GHzマシンだ。

スペック

チップセットはAMD-750を採用している。AMD-750チップセットは、AMD-751とAMD-756の2チップで構成されている。AMD-756はUltra ATA/66に対応しているが、AMD-751はAGPは2×、メモリはPC100と若干古さは否めない。システムバスクロックは200MHzとなっているが、CPUとAMD-751システムコントローラ間だけが、ダブルデータレート転送を使い200MHzで稼働し、AMD-751とメモリ間は100MHzで稼働する。安価なPC100のSDRAMが使用できるというメリットはあるものの、システム全体の高速化は望めないだろう。ただし、AMDは100MHzまたは133MHzのFSBに比べ50%～100%高速であると公表している。

写真1　ケース側面を開けたところ
ヒートシンク付きファンのみでクーリング対策を行っている。発熱はさほど大きくない。

消費電力が高く、高熱を発するAthlonのクーリング対策は、CPUに直付けされたファン付きヒートシンクで行っている（**写真1**）。その他の特別なクーリング対策は行っていない。一部のAthlonマシンではケース側面に排気口を設け、ファンで強制排気を行っているものすらあるのに、本当に大丈夫だろうかと心配になるが、編集部の連続稼働では全く問題なかった。実際にCPUを触ってみても、さほど熱くはならなかったので、いわれるほど発熱に対して神経質になる必要はないのかもしれない。

マザーボードは1AGP、5PCIスロットを備え拡張性が高い。DIMMスロットも3本用意されている。メモリスロットが多いというのは、個人ユーザーにとってありがたい拡張性だ（**写真2**）。標準構成では、サウンドカードとモデムカードを装着しているので、空きPCIは3スロット、空きDIMMスロットが2本になっている。

ゲートウェイのAthlonマシンで使用されているマザーボードは共通で、先月号の特集で紹介した「JI-SAKU-KI」とマザーボードは同じだった。このマザーボードはUSBポートが3ポートついているというユニークな特徴を持っている。

ビデオカードは標準構成でnVIDIA GeForce 256 DDR 32MBを搭載しているが、今回テストに使用したモデルはBTOで選択できるnVIDIA RIVA TNT2 M64 16MBを搭載している（共にOEM版）。これはTNT2の廉価版であり、すでに1世代前のスペックだ。Athlon 1GHzのシステムとしては相応しくないと思える。2つのビデオカードの差額が1万3千円であることを考えると、TNT2 M64をラインナップする必要性は低いと思われる。

ハードディスクも標準で10Gバイトと今となっては少ない部類である。価格を抑えるために周辺機器を抑えてあるのだろう。もっとも、Athlon 1GHzを搭載してサウンドカード、56kモデム、最大40倍速CD-ROM、17インチディスプレイ込みで30万円を切るという価格設定はかなりお買い得であろう。

1GHz! Athlon

Athlonの1次キャッシュは128Kバイトと Pentium IIIの32Kバイトに比べ4倍もの容量を誇っている。しかし、Athlon

1GHzの2次キャッシュのクロックは333MHzとコアクロックの3分の1での動作だ。これは、2次キャッシュのクロックが350MHzのAthlon 700MHzより遅い。今までは同クロックのAthlonとPentium IIIであれば、Athlonのほうが速いといわれていたが、ここにきて2次キャッシュと等速で動作するCoppermineコアのPentium IIIに比べて遅いといわれるようになった。2次キャッシュが遅い理由は、Athlonの2次キャッシュであるSRAMがCPUコアダイに内蔵されていないからだ。しかし、CPUコアダイにSRAMを内蔵していないために歩留まりがよく、512Kバイトという容量の2次キャッシュを搭載しているにも関わらず、CPU自体を低価格に抑えられるというメリットもある。ただし3月の発表時では、Athlonの単価は15万5880円（1,000個ロット）とPentium IIIの10万8460円に比べて5万円も高い。

Athlonの最大の特徴は、新たに追加されたスーパースケーラ浮動小数点ユニットにより浮動小数点演算が高速だということだ。これはPentium IIIに比べ、3Dジオメトリ演算を大量に行う3Dゲームなどに適しているといえるだろう。ただしソフトが対応していなければこのありがたい機能を利用することはできない。

CPU	1GHz AMD Athlon
メモリ	64Mバイト PC100 SDRAM （64Mバイト×1）
チップセット	AMD-750チップセット
ハードディスク	10Gバイト Ultra ATA HDD
ビデオカード	nVIDIA DDR GeForce 256 32MB w/DVI 出力 AGP グラフィックス
フロッピードライブ	3.5インチ フロッピードライブ
CD-ROMドライブ	最大40倍CD-ROM ドライブ
サウンドカード	Sound Blaster Live! Value w/デジタル出力 カード
スピーカ	なし
モデム	56K PCI モデム
モニタ	17インチカラーモニタ
ケース	ミッドタワーケース
キーボード	109日本語キーボード
マウス	MSインテリマウス
OS	Windows 98
価格	28万9800円

表1　Gateway Select 1000標準構成のスペック

Linux Ready!?

Select 1000に標準で搭載されているOSはWindows 98 Second Editionだ。Windows NT Workstation 4.0も選択できる。もちろんLinuxは…選択できない。当然ながら自己の責任においてインストールすることになる。しかし、カーネル2.2.14、XFree86 3.3.6以降では問題ない。ただし、内蔵されているCONEXANTのチップを使っているモデムに関してはドライバが存在しないために利用できない。Linuxを使用するつもりならBTOでモデムを削除したほうがいいだろう。

なお、Ultra ATA/66に関しては、正式対応ではないもののTurboLinux Workstation 6.0 など、最新のディストリビューションを使用すれば利用できる。ただし、デフォルトでは設定されないのでhdparmコマンドで転送モードを変更する必要がある。

Ultra ATA/66はともかく、標準ではDMAがオフになっているのでかなり遅くなってしまう。高速なAthlonを使うのであれば忘れずに設定しよう。

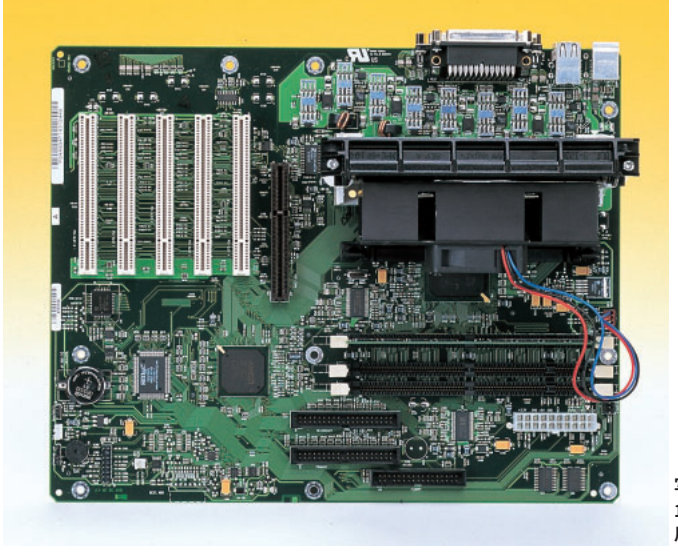


写真2　マザーボード
1AGP／5CPI。拡張性が高い。RIMMスロットも3本用意されている。USBが3ポートというのも珍しい。

写真3　1GHz Athlon
これが世界初の1GHz CPUだ。見た目はいままでのAthlonと変わらない。

Dimension XPS B 1000r Special Edition

デルコンピュータ株式会社

価格：39万7300円～
http://www.dell.com/jp/

Dimension XPS B 1000r Special Edition（以下、XPS B 1000r）は世界で最初に発売されたPentium III 1GHzマシンだ。Pentium III 1GHzを搭載したマシンはHP、IBMからも発表されているが、現時点で実際に製品として出荷しているのはデルコンピュータだけだ。Pentium III 1GHzは現状では限定生産のためAthlonよりも入手困難であり、日本では事実上デルコンピュータからのみ入手できる。

スペック

XPS B 1000rはCPUだけでなく、その他のスペックでも最強マシンとして相応しいシステム構成だ。今回テストに使用したモデルはPC700 RDRAM 256Mバイトメモリ、CD-R/RW、DVD-ROMドライブが内蔵されている。

クーリング対策に巨大なヒートシンクと排気ダクトを設け、ファンによる強制排気を行っている。このため、ケースを開けただけでは排気ダクトに隠れてCPUは見えない（**写真4**）。

マザーボードはインテルのVC820を使用している。1AGP、5PCIとGateway Select 1000と同様に拡張性が高い。チップセットはi820だ。i820チップセットは、インテルのハイエンド向けのチップセットで、FSB 133MHz、Ultra ATA/66に対応している。しかしなんといってもi820チップセットの最大の特徴は、PC800 RDRAMの対応だ。

CPUが高速化するに伴い、高速なメモリが必要になる。互換チップセットメーカーはSDRAMの高速化でその場をしのいでいるが、インテルはラムバスが開発した高速RDRAMを使用することで対処している。PC800 RDRAMを使えば、100MHz SDRAMに比べ約2倍のメモリ帯域幅である1.6Gバイト／秒が得られる。Pentium III 1GHzの性能を引き出すのに十分なチップセットである。しかし、RDRAMはまだ高価で、SDRAMに比べ8倍近い値段の開きがある。このため、基本的にPentium III 1GHzマシンはAthlon 1GHzより高くなってしまう。

なお、RDRAMを使ったシステム構成では、すべてのRIMMスロットが埋まっていなければならない。これはRambusインターフェイスの反射を防ぐためで、RIMMスロットに空きがある場合は、C-RIMMと呼ばれる導電モジュールを装着しなければならない（**写真5**）。また、i820チップセットのバグの問題でRIMMスロットは2本しか搭載されていない。このため最大メモリは512Mバイトとなっている。

ビデオカードは現時点の最高スペックに位置するnVIDIA DDR GeForce 256 64Mバイト(OEM版）を搭載している。BTOでもこれ以外のビデオカードは選択できない。

この構成で45万3300円という価格設定になる。もっとも、PC700 128MバイトとPC700 256Mバイトの差が6万円ということで、いかにRDRAMの値段が高いかがわかる。

ハードディスクは標準で30Gバイトと申し分ない容量である。BTOで選択できる構成に34Gバイトもある。

ゲートウェイのGateway Select 1000がハイエンドマシンを低価格で提供する

写真3　DVDドライブ（上）とCD-RWドライブ（下）
標準構成ではCD-RWドライブのみ

写真4　ケース側面を開けたところ
巨大な排気ダクトが目立つ。このダクトで強制排気している。

路線であるのと対照的に、デルコンピュータのXPS B 1000rでは最高スペックのハイエンドマシンという路線を展開している。

CPU	1GHz Pentium III
メモリ	128Mバイト ECC PC700 RDRAM（128Mバイト×1）
チップセット	Intel 820チップセット
ハードディスク	30GB Ultra ATA66 HDD
ビデオカード	nVIDIA DDR GeForce 256 + 3D 4× AGP 64MB
フロッピードライブ	3.5インチ フロッピードライブ
CD-R/RWドライブ	読み込み最大32倍/書き込み8倍/書き換え4倍速CD-RWドライブ
サウンドカード	Sound Blaster Live! Value 512ボイスサウンドカード
スピーカ	Altec ACS340
モデム	なし
モニタ	17インチカラーモニタ
ケース	109日本語キーボード
マウス	MSインテリマウス
OS	Windows 98
価格	39万7300円

表2　Dimension XPS B 1000r Special Edition標準構成のスペック

1GHz! Pentium III

CoppermineコアのPentium IIIの特徴は、Athlonと違いCPUコアダイに内蔵されCPUコアと等速に動作する2次キャッシュにある。このため、1次キャッシュ32Kバイト、2次キャッシュ256KバイトとAthlonに比べ少ないキャッシュ容量だが、高速化されている。しかし、CPUコアダイにSRAMを内蔵しなければならないため歩留まりが悪い。実際問題として、インテルはPentium III 1GHzの発表を急ぐあまり、量産体制が整っていない。このため限定生産という生産体制をとっている。CPUの単価はAthlonよりも安く設定されているが、出荷量が少ないのであれば意味はない。当然入手困難なCPUはプレミア価格がつくだろう。

Linux Ready!?

XPS B 1000rのOSはWindows 98 Second Editionしか選択できない。Linuxに関しては Gateway Select 1000同様だ。こちらも、カーネル2.2.14、XFree86 3.3.6以降では問題ない。ただし、編集部で確認したところ今月号の付録CD-ROMに収録されているVine Linux 2.0やRed Hat 6.2Jのインストーラではvga表示になってしまった。XPS B 1000rに付属しているnVIDIA DDR GeForce 256を認識が認識できないようだ（XFree86 3.3.6では対応済み）。もしかするとAnacondaを使用したインストーラでは認識できない可能性があるので注意が必要だ。もっとも、セットアップ後にX Window Systemの設定をすれば正常に認識させることができたので問題ないだろう。

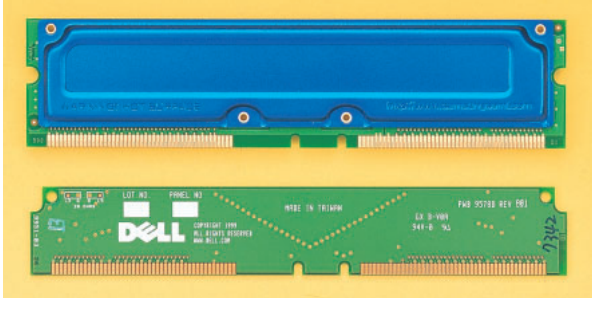


写真5　RDRAM（上）とC-RIMM（下）
RIMMスロット2本を埋めるためにC-RIMMが装着されている。

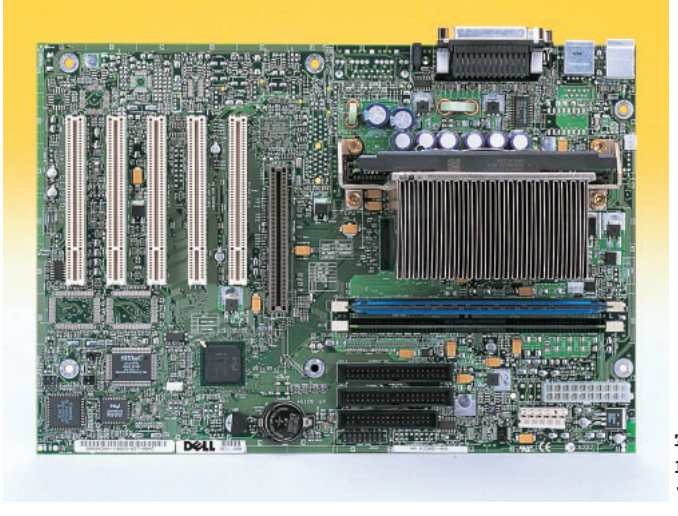


写真6　マザーボード
1AGP／5PCI。i820チップセットの問題で2本のRIMMスロットしかない。巨大なヒートシンクがひときわ目立つ。

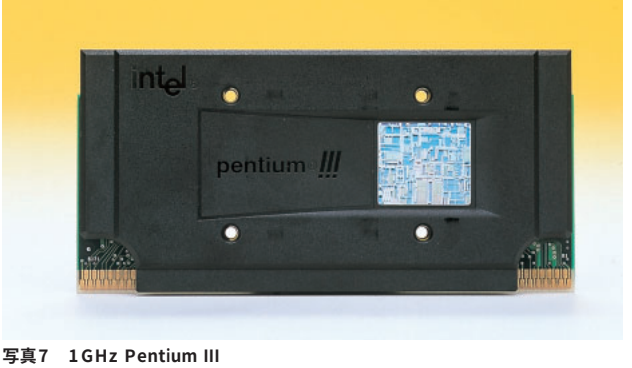


写真7　1GHz Pentium III
Coppermineコアを採用。コアダイに2次キャッシュを内蔵して高速化している。

1G対決！　結果は予想どおりほぼ互角…

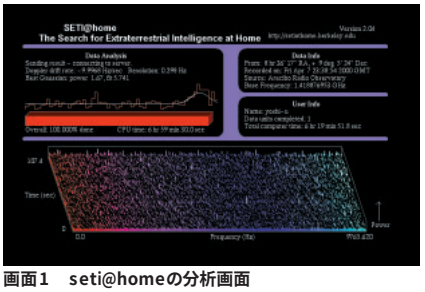
Pentium III 1GHz vs. Athlon 1GHz

今回紹介した2機種を使ってPentium III 1GHzとAthlon 1GHzの実力を調べてみることにする。もちろん2つの機種はハードウェア構成が異なるため、純粋にCPU同士の対決とはいえない点に留意してほしい。それでもCPU自体のポテンシャルは十分感じられるはずだ。特にCPUキャッシュの容量と、キャッシュの速度がどのように影響するのか、SDRAMとRDRAMの性能にどのくらいの違いがあるのかは十分わかるはずだ。ベンチマークは、TurboLinux Workstation 6.0で行った。ベンチマークソフトの測定誤差をなくすためにベンチマークを3回行い平均している。

まず最初に本誌でもおなじみのHDBENCH clone 0.14.1を使ってベンチマークを取ってみた（**図1**）。結果はPentium III 1GHzの圧勝であった。期待された浮動小数点も3割ほどAthlon 1GHzのほうが遅いという結果が出た。メモリに関してはPentium III 1GHzはAthlon 1GHzに比べて2倍の性能が出ている。しかし、HDBENCH cloneの作者は互換CPUの結果が遅くなると明言しているのでベンチマークの結果ほどの差はないだろう。参考までにWindowsのHDBENCH 3.22の結果を載せておく（**図2**）。Windowsでは両者は僅差である。

次に、ベンチマークではないがseti@homeを実行してみた。seti@homeは宇宙からの微弱な電波をインターネット上のコンピュータを使って解析し、地球外知的生命体を検索するという壮大なプロジェクトだ。seti@homeのサーバからデータを受け取り分析する。この分析時間をベンチマークとして利用した（**画面1**）。もちろん、公正さを保つために同一のデータを使用する。結果は僅差でPentium III 1GHzが勝利している（**図3**）。しかし、7時間近い計算を行っている中で僅か14分ほどの違いはほとんど無視できるだろう。

最後にnbench 2.1.0を実行してみた（**図4**）。nbenchはソートやフーリエ計算など、さまざまな演算を行いメモリ性能と正数演算、浮動小数点演算の能力を測定するベンチマークソフトだ。自分の環境でコンパイルすることもできるが、今回は公平性を保つためにコンパイル済みで提供されているバイナリRPMパッケージを使用した（gcc 2.7.23でコンパイル済み）。全てのベンチマークの中で、nbenchでの結果だけがAthlon 1GHzの圧勝である。nbenchは比較的小さなプログラムのため、512Kバイトという巨大なAthlon 1GHzの2次キャッシュによる効果があったものと思われる。このベンチマークの結果から、高速なキャッシュよりも、大容量のキャッシュのほうが効果が高いということが推測できる。ただし、通常使われるソフトは当然キャッシュに収まることはあり得ないので、ベンチマークほどの大きな差は生まれないだろう。これらのベンチマークの結果から、若干1GHz Penitum IIIのほうが有利に思えるが、さほどの差はないと思われる。



画面1　seti@homeの分析画面
膨大な量のデータを分析し地球外知的生命体を探索する。残念ながら地球外知的生命体は見つかっていない…

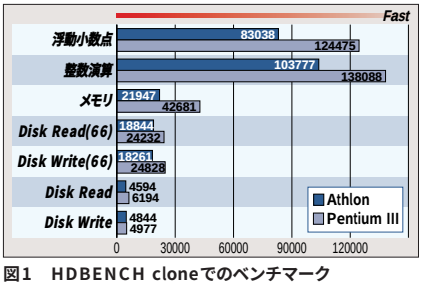


図1　HDBENCH cloneでのベンチマーク
Disk Read（66）はDMA66オン、Disk ReadはDMAオフ。

HDBENCH clone	http://www.enjoy.ne.jp/~gm/index-ja.html
HDBENCH	http://www.lares.dti.ne.jp/~ep82kazu/
seti@home	http://setiathome.ssl.berkeley.edu/
nbench	http://members.xoom.com/Gavrilov_Con/

表3　ベンチマークに使用したソフトの入手先URL

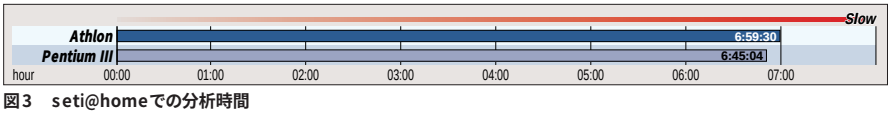


図3　seti@homeでの分析時間
ほとんど差は出ていない。ベンチマークよりも実際の体感速度に近い。

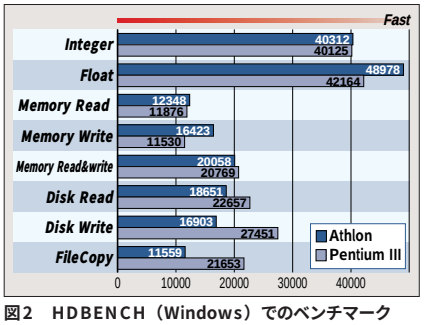


図2　HDBENCH（Windows）でのベンチマーク
浮動小数点がAthlonのほうが速い。メモリもRDRAMの効果はあまりない。

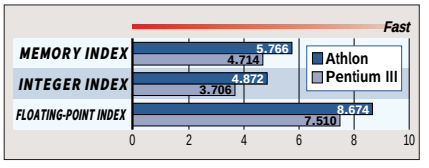
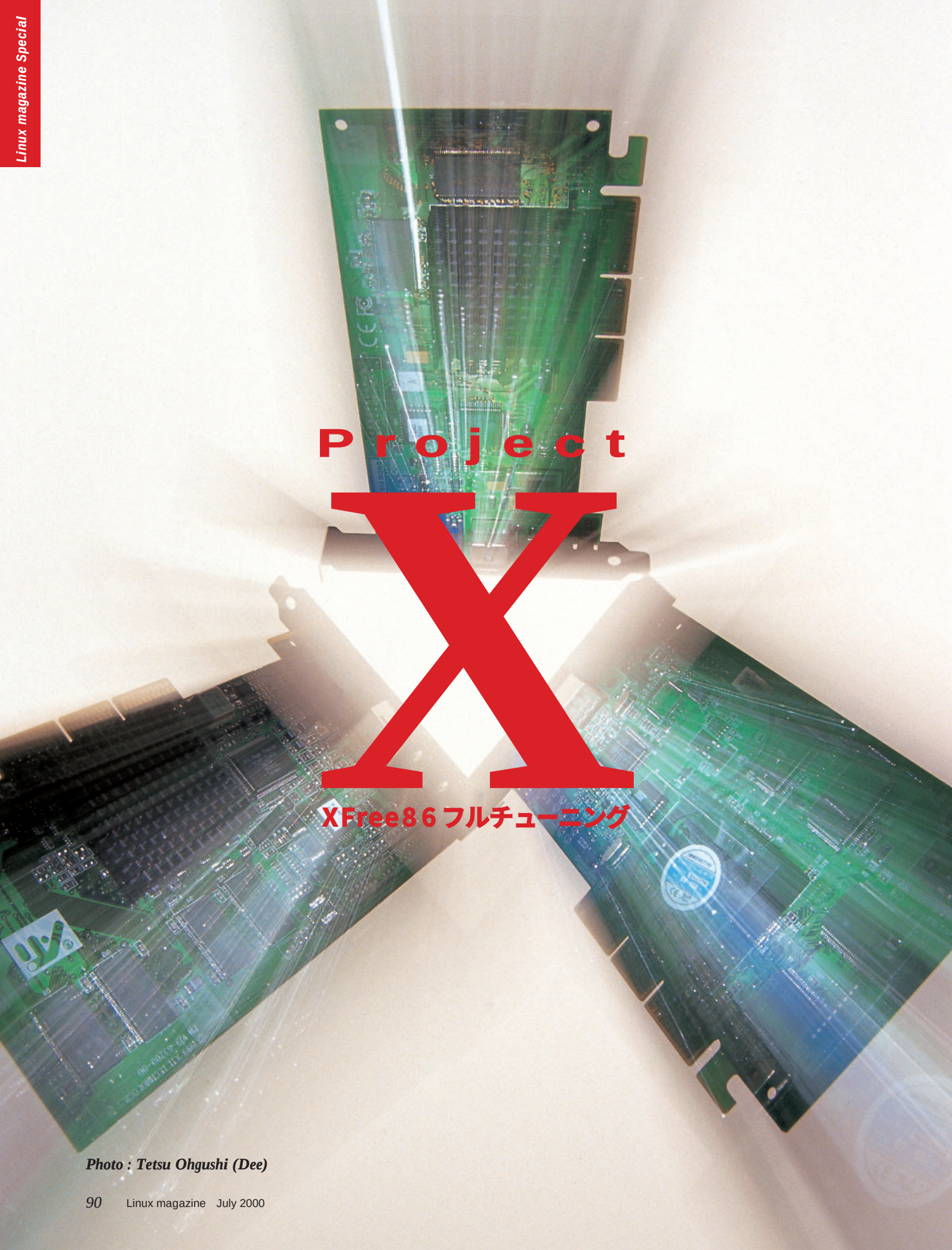


図4　nbenchでのベンチマーク
メモリを含め、すべてのベンチマークでAthlonが速い。キャッシュの問題と思われる。



Project X

XFree86フルチューニング

Photo : Tetsu Ohgushi (Dee)

Part 1

Xの仕組みとXFree86

文：山岸典将
Text : Norimasa Yamagishi

Linuxのウィンドウシステムといえば、X Window Systemだ。そして、実際にはLinuxを利用しているほとんどの人がX Window Systemを利用しているといってもよいだろう。また、X Window Systemがなければ Linuxもこれほど利用されることはなかっただろう。

ウィンドウシステム

ウィンドウシステムとはなにか。見た目だけでいえば、画面上にプログラムがいくつもの窓を開いていれば、それがウィンドウシステムだといえるだろう。しかし、表示はもちろんのこと、キーボードやマウスなどからの入力をどのプログラムに渡すのかを制御することもウィンドウシステムの役割だ。

そして、現実には、そのシステム上で動作するためのアプリケーションプログラム、さらに、そのプログラムを開発するためのライブラリが存在しなければ、そのシステムは役には立たない。さらに操作方法に統一性を持たせるための規約が用意されている場合がほとんどだ。

X Window Systemはもちろん、Microsoft WindowsやMacintoshのMacOSといったウィンドウシステムもこれらを備えている。

X Window Systemの特徴

X Window Systemには以下のような特徴がある。

1. ソースコードが公開されていること。
2. サーバ／クライアント方式のシステムであること。
3. ネットワーク透過型のシステムであること。

それぞれ、順を追って見ていこう。

●ソースコードの公開

X Window System自体は、初期のバージョンからソースコードが公開されている。ソースコードの公開がそのソフトウェアの普及に果たす役割は、本誌読者にはいうまでもないだろう。LinuxやBSDはもちろん、多くのUNIXシステムや、Microsoft Windows、Macintoshなどで動作するXサーバも存在する。

Linux（やFreeBSDなどの俗にいうPC UNIX）上で、もっとも使われているXFree86も、i386系のCPU上で動作するフリーのX Window Systemのひとつである。XFree86以外にもi386系のCPUで動作する商用のものと

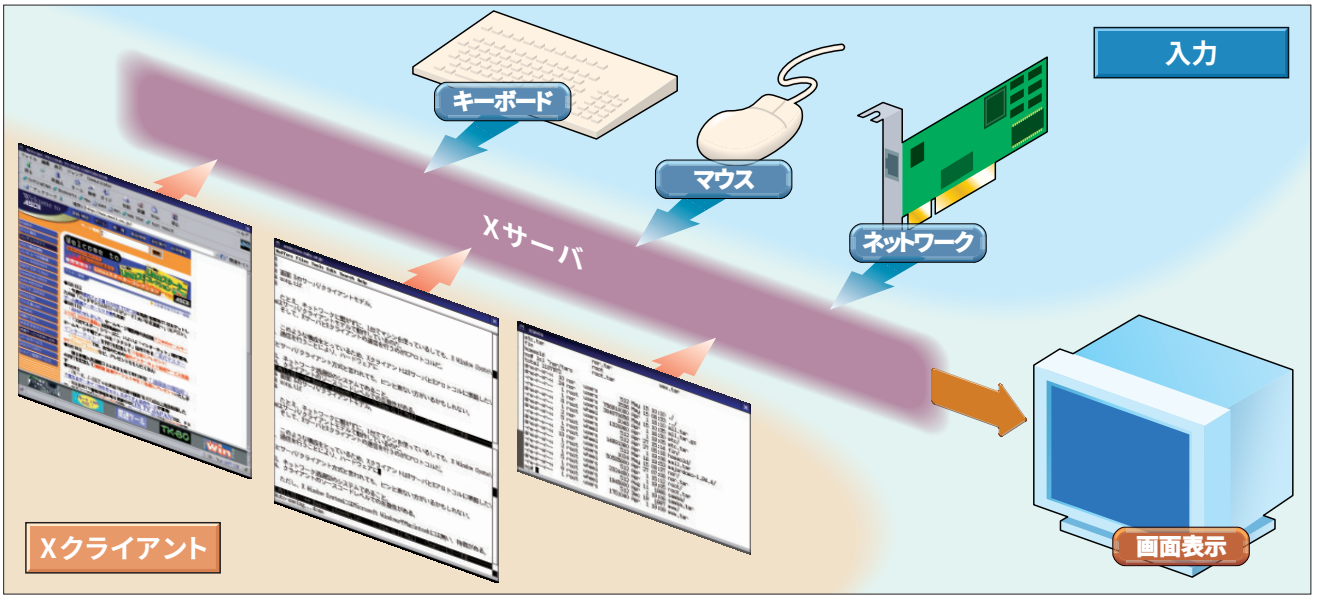


図1　Xのサーバ／クライアントモデル
Xは、Xクライアント（アプリケーションプログラム）と、画面表示や各種デバイスからの入力をXクライアントに伝えるXサーバのサーバ／クライアントモデルで動作する。

してXi Graphics社のAccelerated-Xや、Metro Link社のMetro-Xがある。

●サーバ／クライアント方式

X Window Systemは、XサーバとXクライアントから構成される。

XクライアントとはX Window System上で動作するアプリケーションプログラム。そして、XサーバはXクライアントとさまざまな入出力機器との仲介を担うプログラムだ（**図1**）。

たとえ、ネットワークに繋がずに、1台でマシンを使っているとしても、X Window Systemはサーバ／クライアントモデルで動作しているのだ。

Xサーバの役割は基本的な役割は以下の2つ。

1. Xクライアントからの要求に従い画面に表示を行う。
2. キーボード、マウス、ネットワークなどからの入力をXクライアントに伝える。

Microsoft WindowsやMacintoshのMacOSと違い、X Window System自体にはファイルマネージャなどの機能は存在しない。

そして、XサーバとXクライアントの通信方法はXプロトコルで規定されている。

実際の表示そのものに関してはXサーバが受け持つため、XクライアントはXプロトコルに準拠した通信を行うだけで、どのようなグラフィックハードウェアを使っているかは気にしないでよい。入出力についても同様だ。そのため、特殊なハードウェアに直接アクセスするようなプログラムでもない限りは、同一のソースコードをコンパイルし使用することができる。

●ネットワーク透過型のシステム

X Window Systemがネットワーク透過型のシステムだということは、Microsoft WindowsやMacintoshとの大きな違いだろう。これは簡単にいえば、XサーバとXクライアントは同じマシン上で動作している必要がない、ということだ（**図2**）。

実は、キーボードなどからの入力、ディスプレイへの出力は、すべてネットワーク上で通信されている。これはたとえ同一のマシン上でサーバとクライアントが動作していても同じである。

すなわち、ネットワーク上で通信が可能であれば、（理論的には）地球の裏側にあるマシン上でWebブラウザを立ち上げ、その操作と表示を手元のマシンで行うことも可能なのだ。イメージとしては、Webサーバで実行されるCGIプログラムをWebブラウザから動かしているのと似ているかもしれない。

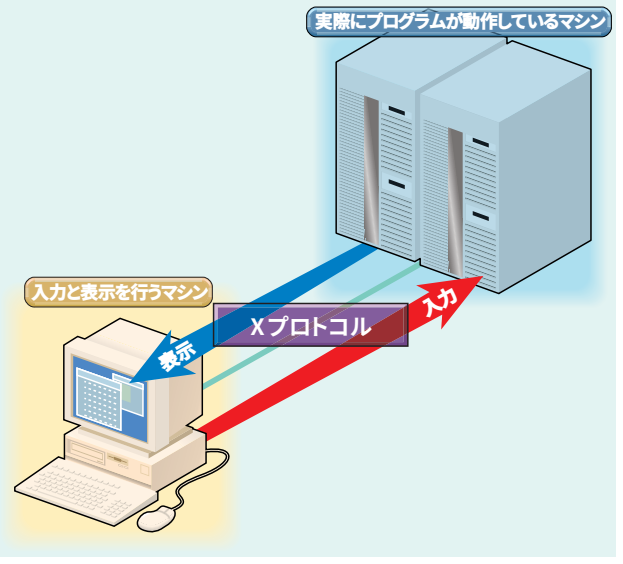


図2　ネットワーク透過型のウィンドウシステム
他のマシン上でプログラムを実行し、手元のマシンでは表示と入力を行うことができる。

Column

X Window Systemの歴史

X Window SystemはMIT（マサチューセッツ工科大学）とDECによるAthena Projectにおいて開発された。Athena Projectはネットワークを利用したサーバ／クライアント構成の環境を整える目的のものだった。なぜ、「X」なのかといえば「W」というウィンドウシステムを元に開発されたものだからだ。

X Window Systemのソースコードは、ハードウェアやソフトウェア（OSなど）に依存した部分と、それ以外の部分が切り放されて開発されており、ハードウェアに依存した部分のみを開発することにより、新しい環境でX Window Systemを利用できる。XFree86もそのような形で開発された。

X Window Systemは、そのバージョンをX11R6といった形で表現する。これはバージョン11リリース6と読む。最初に一般公開されたのは1986年のX10R3だ。そして1987年のX11R1でプロトコルの変更が行われ、現在の最新バージョン、X11R6.4は1998年に公開された。

なお、開発の管理は、MIT内のX Consortiumから、X Consortium, Inc.、The Open Groupを経て、現在はX.Orgにより行われている。

Windows NT Terminal Server、Metaframeといったソフトウェアを使用すれば、Windowsでもネットワークの向うにあるマシンのプログラムを利用することが可能だが、そのライセンス料等を考えると、限られた予算の範囲で簡単に導入できる代物ではない。基本的にフリーであるというのもX Window Systemの大きなメリットだ。

余談だが、ネットワーク上でX Window Systemを使う場合、通常のサーバとクライアントのイメージとは逆になるので、気をつけよう。X Window Systemの場合、手元で動作しているのはXサーバで、ネットワークの向うで動作しているのはXクライアントだ。

ウィンドウマネージャ

「サーバ／クライアント方式」のところでXサーバは表示および入出力を担当すると書いた。しかしXサーバによる表示とは、本当に単にウィンドウを表示するだけの、非常にシンプルなものだ。ウィンドウの切り替えや、ウィンドウタイトルの表示といった機能は一切ない。このような機能はウィンドウマネージャが提供する。ウィンドウマネージャもXクライアントの一種だが、X Window Systemのルック&フィールを管理するという意味で特殊なものといえる。もちろん、ウィンドウマネージャなしでもX Window Systemがまったく使えなくないというわけではないが、普通はウィンドウマネージャを利用することになるだろう（**画面1**）。

ウィンドウマネージャには多くの種類があり、これが一番良いというものは存在しない。好みに応じて選択するのがよいだろう。

ここではその中でもよく使われていると思われるものを紹介する。もっとも、当然筆者もすべてのウィンドウマネージャを試して使い込んでいるわけではない。ちなみに筆者が好んで使うのはWindow Makerだ。これは、画面の狭いノートパソコンを使っているので仮想デスクトップが必須であること、そして、使い勝手と設定のしやすさ、動作の軽快さのバランスから選択した。また、CPUパワーのあるマシンではGNOMEも使いたいため、GNOMEが対応しているというのも、選択のポイントとなった。

Column

X端末とは

昔（といっても7、8年前）はX端末という、Xサーバのみを実行する端末がよく使われていた。これは現在のようにX Window Systemが動作するようなPCが安く手に入る時代ではなかったためだ。そのため開発現場でもなければ、1人に1台UNIXワークステーションとはとてもいかない。そこで、強力なパワーのある1台のサーバにプログラムを実行させて、表示は手元のX端末という使い方をしていた。

現在ではマシンの値段も下がり、1人1人がそれなりのパワーを持ったマシンが使えるようになった。そのため1台の強力なマシンを何人もが同時に使うというやり方は当時に比べれば減ってきている。

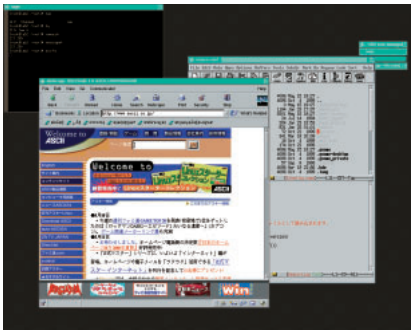
そして逆に、クラスタリングを利用し、何台ものマシンを1つの強力なシステムにまとめあげるという方法が、主流になりつつある。もっとも、この場合も、その強力なマシンを何人かで使うためにもX Window Systemが使われることになるだろう。

●twm

X11R4から標準として含まれているウィンドウマネージャがtwm（Tab Window Managerの略）だ（**画面2**）。そのためX Window Systemがインストールされていれば、まずtwmも使えると考えてよいだろう。古くからあるウィンドウマネージャで、見た目も非常にシンプル、機能もシンプルだ。

設定についても、最近のウィンドウマネージャのように専用のツールは用意されておらず、自分で設定ファイル.twmrcを記述する必要がある。また、最近のウィンドウマネージャにはほとんど用意されている仮想デスクトッ


画面1　ウィンドウマネージャなしのX


画面2　twmのデスクトップ


画面3　fvwm2のデスクトップ

プ機能はない。

もっとも、機能が少ないといっても、もちろん本来ウィンドウマネージャに必要とされる機能は備えているので、「ウィンドウマネージャには余計な装飾は必要なく、軽いほうがよい」ということで、昔から使い続けているユーザーもいる。

なお、twmを拡張し仮想デスクトップ機能を使えるようにしたvtwm、tvtwm、ctwmというウィンドウマネージャもある。

●fvwm2

fvwm2も今となってはかなりシンプルなウィンドウマネージャだ（**画面3**）。現在の華やかなフリーのウィンドウマネージャのはしりといえるfvwmの進化したものだ。twmと比較すると、仮想デスクトップ機能を持っていること、Motif風、Open Look風といった複数のルック＆フィールのサポートが特徴といえる。

fvwmから派生したウィンドウマネージャには、Windows 95のユーザーインターフェイスをまねたfvwm95がある。

設定は.fvwm2rcを書き換えて行う。

（fvwmのWebページ　http://www.fvwm.org/）

●AfterStep

AfterStepはNEXTSTEPライクなウィンドウマネージャだ。その外観こそが一番の特徴だろう（**画面4**）。

AfterStepはfvwmを元にしたウィンドウマネージャBowManの発展型であるため、機能的にはfvwmとよく似たものになっている。

Wharfと名付けられた、右側に表示されるアプリケーションランチャは、アイコンを階層的にして、収納することができる。もちろん仮想デスクトップ機能もある。

設定はAfterStep-1.0では.steprcを書き換えて、AfterStep-1.6以降ではGNUstep/Library/AfterStepの中のファイルを書き換えて行う。

（AfterStepのWebページ　http://www.afterstep.org/）

●Window Maker

Window MakerもNEXTSTEP風のウィンドウマネージャだ。仮想デスクトップ、テーマのサポート、またGUIによる設定の変更も可能になっている（**画面5**）。

特に設定の容易さが特徴といえるだろう。これまでに出てきたウィンドウマネージャなら設定ファイルを書かなければいけなかったようなことも、GUIの設定ツールで行うことができる。

また、右側にあるのはDockと呼ばれるアプリケーションランチャだが、これへの登録は、アプリケーションを起動すると下に表示されるアイコンをドラックして一番左に持っていくだけでよい。

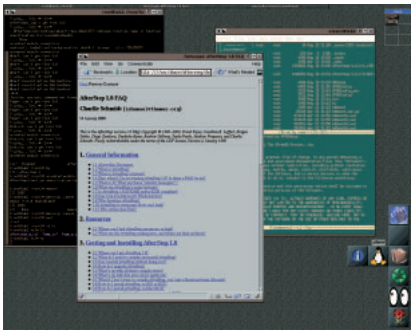
そして、これは賛否両論あるだろうが、マウスのクリックにより表示されるメニューを、そのままウィンドウとして残すこともできるのも便利だろう。Vine Linux標準のウィンドウマネージャに採用されている。

（Window MakerのWebページ　http://www.windowmaker.org/）

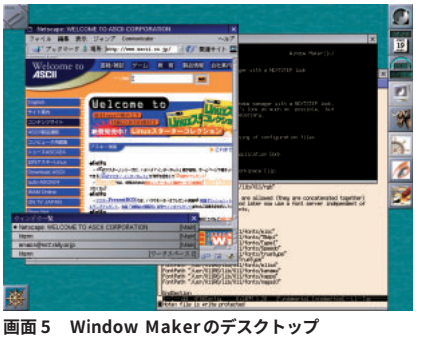
●Enlightenment

「E」とも呼ばれるEnlightenment（**画面6**）の特徴は、何といってもそのテーマ機能だろう。テーマを切り替えることにより、まったく違ったウィンドウマネージャにしたような雰囲気を味わうことができる。カスタマイズできる機能も豊富で、自分で使いやすいように設定できる。設定はGUIのツールで行うことができる。

また、GNOMEに対応しているウィンドウマネージャとしても有名で、Red Hat Linux 6.2J、Laser5 Linux6.0 Rel.2といったGNOMEを採用しているディストリビューションの多くがEnlightenmentをウィンドウマネージャとして採用している。



画面4　AfterStepのデスクトップ



画面5　Window Makerのデスクトップ

画面6　Enlightenmentのデスクトップ

（EnlightenmentのWebページ　http://www.enlightenment.org/）

●sawfish（sawmill）

sawfishはrepというlisp言語処理系を内蔵したウィンドウマネージャだ（**画面7**）。そのためrepを記述することによりさまざまな拡張が可能になる。もちろん、GUIのツールでもかなり細かく設定を変更できる。

またGNOMEとの親和性も非常に高くTurboLinux Workstation 日本語版 6.0 、Kondara MNU/Linux 1.1でも標準のウィンドウマネージャとして採用されている。Enlightenmentに比べると動作が軽いので、あまりパワーのないマシンでも使うことができるだろう。

なお、sawfishは今年の5月まではsawmillという名前だったが、バージョン0.27より名前を変更した。

（sawfishのWebページ　http://www.dcs.warwick.ac.uk/~john/sw/sawfish/）

デスクトップ環境

ウィンドウマネージャは、ウィンドウシステムの管理を目的としている。そして、X Window Systemをさらに使いやすくする目的のソフトウェアにデスクトップ環境がある。

デスクトップ環境は、ファイルマネージャ、ヘルプ、システム設定ユーティリティ、そして、メールやワープロ、表計算といったさまざまなGUIのアプリケーションプログラムを提供する。

現在のディストリビューションの多くは、GNOMEかKDEのデスクトップ環境を利用できるようになっている。

●GNOME

GNOME(GNU Network Object Model Environment)はGNUライセンスに基づくフリーなデスクトップ環境だ（**画面8**）。GTK+をツールキットとして利用している。対応するウィンドウマネージャはEnlightenment、sawfish、Window Makerなどがある。

設定はGUIのツールGNOMEコントロールセンターの中から変更できる。

GNOMEを標準で採用しているディストリビューションはTurboLinux Workstation 日本語版 6.0、Red Hat Linux 6.2J、Kondara MNU/Linux 1.1、LASER5 Linux 6.0などだ。

（GNOMEのWebページ　http://www.gnome.org/、日本GNOMEユーザー会のWebページ　http://www.gnome.gr.jp/）

●KDE

KDE（The K Desktop Environment）は、ノルウェーのTroll Tech社のQtというツールキットを利用して作成された、フリーの統合デスクトップ環境だ（**画面9**）。

標準でkwmというウィンドウマネージャが用意されているが、EnlightenmentもKDEに対応している。ファイルマネージャやヘルプ、メールソフトなどは日本語化されている。

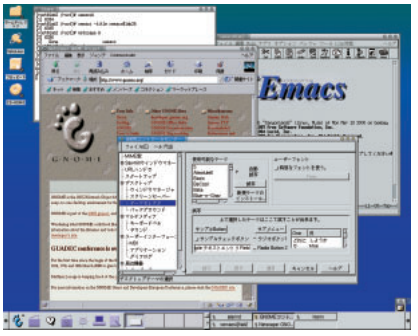
設定に関しては、KDEコントロールセンターというGUIのツールが用意されている。この中にあるシステムインフォメーションでは、現在のハードウェアの環境を知ることができる。

日本製の電子辞書検索やゲーム、エディタといったKDE用のソフトウェアもリリースされている。また、日本語化はされていないがKOfficeというオフィススイートもある。

標準で採用しているディストリビューションはCaldera OpenLinux 2.3、Corel LINUX OSなどだ。

（KDEのWebページ　http://www.kde.org/、日本KDEユーザ会のWebページhttp://www.kde.gr.jp/）


画面7　sawfishのデスクトップ



画面8　GNOMEのデスクトップ


画面9　KDEのデスクトップ

Part 2

XFree86 3.3.6の設定方法

文：山岸典将
Text : Norimasa Yamagishi

最近では、ディストリビューションのインストーラによってLinux本体とともにXFree86もインストールされることが多くなった。ただし、すべてのハードウェアが自動認識されるわけではないし、グラフィックスカードやモニタを入れ替えた場合、またディスプレイドライバをアップデートする場合などは、再設定が必要になることがある。

XF86SetupによるXFree86の設定

XFree86の設定は、/etc/X11ディレクトリなどに置かれたXF86Configというファイルに書かれている。このファイルはテキストファイルなので直接編集してもよいが、なにもないところから作成するのは大変だ。そこで、VGA用のXサーバを利用して動作するGUIのツール、XF86Setupを使ってXFree86の設定をしてみよう。ディストリビューションによっては、独自の拡張を行ったXFree86の設定ツールが付属するが、ここではXFree86のパッケージに含まれる標準の設定ツールであるXF86Setupを取り上げる。

なお、ハードウェアを交換したり設定を変更したりすると、Xの表示がうまくいかない場合がある。その場合は、Ctrl＋Alt＋Back Spaceキーを押せば、XFree86が強制終了することも覚えておいてほしい。

画面1　XFree86 Setupの起動画面
XがVGA 16色モードで起動し、GUIで設定できる。

●XF86Setupの起動

XF86SetupはXが起動していない状態でスーパーユーザーになって使用する。そのため、すでにXが立ち上がっている状態であれば、いったんXを終了する必要がある。グラフィカルログインで使用しているなら、テキストログインに変更しよう。一時的な変更であれば、起動時にLILOのbootプロンプトで「linux 3」のようにテキストモードのランレベル（この例では3）を指定すればよい。

テキストモードからroot（スーパーユーザー）でログインしたら、XF86Setupを起動しよう。マシンの中にXの設定が存在すれば、まず最初に「Would you like to use the existing XFree86Config file for defaults ?」と聞かれる。設定をちょっと変更するのが目的であれば既存のXF86Configを使うと楽なので＜YES＞を、まったく新規に作るなら＜NO＞を選べばよいだろう。

XF86Setupでは、最上部にあるタブをクリックして設定項目を選択する（**画面1**）。

●マウスの設定

特殊なマウスを使っていない限り、マウスの設定はほとんど必要ないはずだ。念のため、マウスを動かしたりボタンを押したりして、右側にあるマウスの絵が反応するかどうかをチェックしてみよう（**画面2**）。

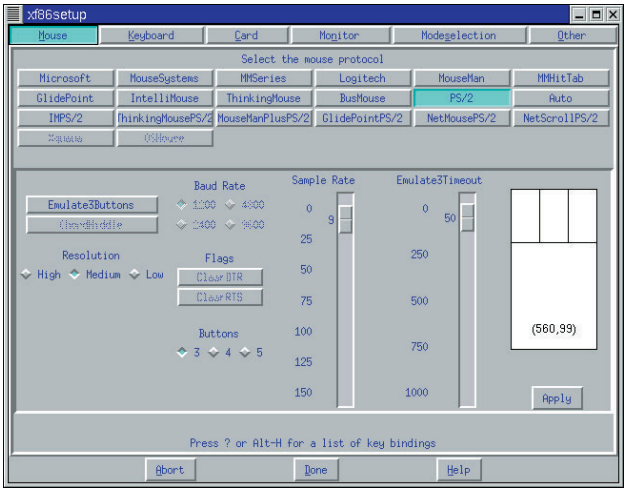
2ボタンマウスを使っている場合には「Emulate3

Column

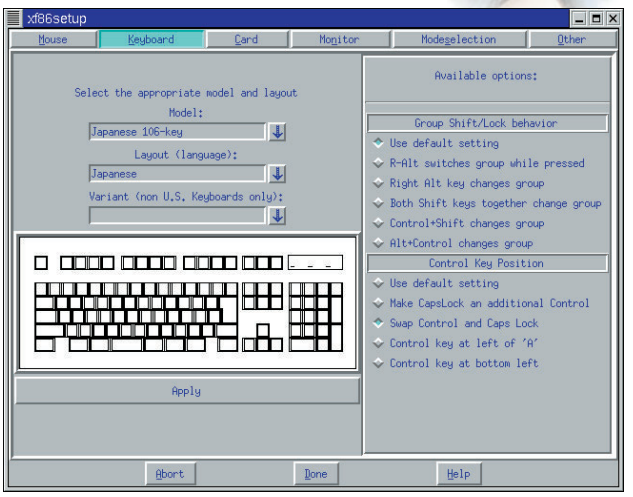
システムにXF86Setupが存在しない！

Red Hat Linux 6.2Jでは、インストール時にXF86Setupはインストールされない。以下の手順でXF86Setupとその実行に必要となるVGA16サーバをCD-ROMからインストールしよう。もちろん、GnoRPMやKpackageを使用してもよい。

```
$ su      （スーパーユーザーになる）
# mount /mnt/cdrom
# cd /mnt/cdrom/RedHat/RPMS
# rpm -ihv XFree86-VGA16-3.3.6-20j1.i386.rpm
# rpm -ihv XFree86-XF86Setup-3.3.6-20j1.i386.rpm
```



画面2　マウスの設定
画面右側のマウスの絵を見ながら設定を確認しよう。



画面3　キーボードの設定
異常終了する場合は直接XFree86Configを書き換える。

Buttons」をチェックしておくと、2つのボタンを同時に押したときに真ん中のボタンを押したことになる。3ボタンマウスによる動作を基本としているX Window Systemを使うためには便利だ。

●キーボードの設定

キーボードの設定では、主に「model」と「Layout」を選ぶことになる（**画面3**）。「model」はキーボードにひらがなが印刷されているキーボードであれば「Japanese 106-key」を選べばよい。「Layout」も「japanese」だ。

なおLinuxではCtrlキーと何かのキーを組み合わせて使用する場合も多い。右側の「Swap Control and Caps Lock」を選ぶと、Caps LockキーとCtrlキーの位置が入れ替わる。CtrlキーがAキーの左隣にないとダメというユーザー向けの機能だ（アメリカ人はキーボード左下にあるCtrlキーを手の腹で押すらしいが……）。

●グラフィックスカードの設定

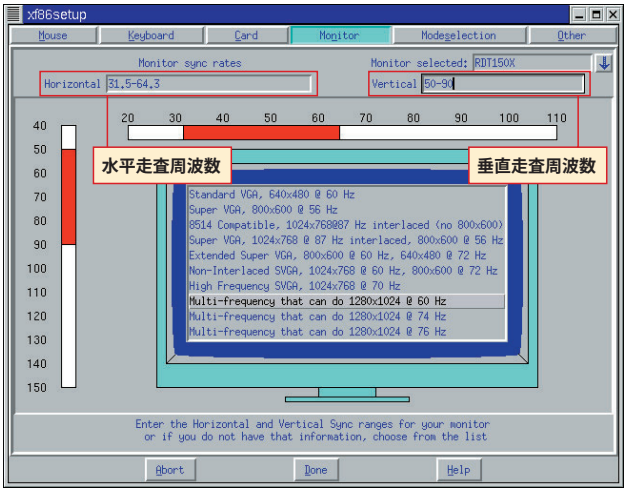
グラフィックスカードの設定はXの設定でも最も重要な部分だ。まず、「Card List」をクリックし、自分の使用しているグラフィックスカードを選ぶ（**画面4**）。それから、「Detailed Setup」を選んで、「Video RAM」の項目を設定すればよい。もっとも現在は「Video RAM」に関してもかなりの確率で自動認識が成功するはずだ。その場合は「Probed」という文字が表示されているだろう（**画面5**）。

●モニタの設定

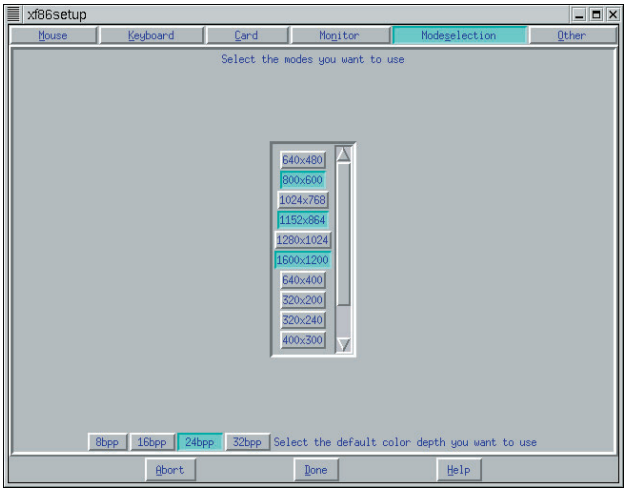
まず、自分のモニタと同じ最大解像度のものをリストから適当に選ぼう。次に、モニタのマニュアルを見て、水平走査周波数を「Horizontal」に、垂直走査周波数を

画面4　グラフィックスカードをリストから選ぶ
使用しているカードか、同じチップのカードを選ぶ。

画面5　グラフィックスカードの詳細設定
RAMDMCやVideo RAMは自動認識が成功する場合が多い。



画面6　モニタの設定
選択肢は多くないが、解像度と周波数さえ正確に入力すればほとんど問題はない。



画面7　ビデオモードの設定
利用したい解像度をすべて選ぶ。

「Vertical」に入力する。最大と最小は「－」で結び「31.5－64.3」というように入力すればよい（**画面6**）。

●ビデオモードの設定

使用する解像度は複数選ぶことができる（**画面7**）。ここで選んだ解像度は、すべてXF86Configに登録され、簡単に変更することができる。また、色数は32ビットカラーまで選べるが、グラフィックスカードが32ビットカラーをサポートしているかは調べてくれない。

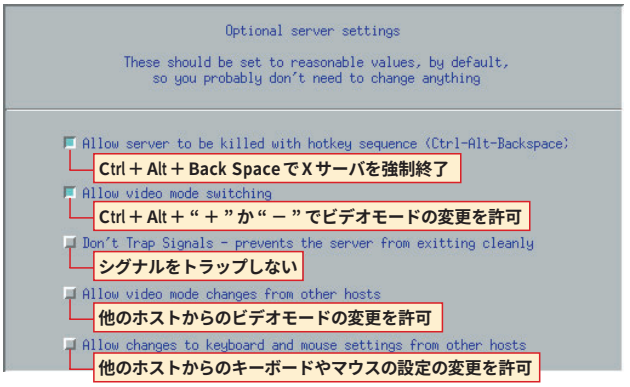
●その他の設定

この中には、チェックしておきたい項目が2つある（**画面8**）。ひとつは、Ctrl＋Alt＋Back SpaceキーでXサーバを強制終了させるための「Allow server to be killed hotkey sequence」。もうひとつはCtrl＋Alt＋（テンキーの）“＋”か“－”でビデオモードの変更を許可する「Allow video mode switching」だ。

この2つは、これでもうXの設定は変更しない、というときまではチェックしておいたほうがよいだろう。

以上を設定したら画面の下にある［Done］をクリックする。そのあと、xvidtuneというツールでディスプレイの設定を確認することができる（**画面9**）。

xvidtuneでは、XF86Setupでの設定に基づいて画面が表示される。「Next」ボタンを押し、すべての解像度での設定をチェックしたら、「Quit」をクリックし、設定を保存して終了する。



画面8　その他の設定
Xの強制終了と解像度変更のホットキーを有効／無効にできる。

XF86Configを自分で書こう

XF86Setupを使えば、XFree86の設定ファイルXF86Configを作成することができる。多くの場合はこれでうまくいくのだが、きめ細かい設定をしたい場合や、XF86Setupではうまく設定できない場合は直接XF86Configを編集する必要がある。

XFree86の設定ファイルは、下記の順に検索され、最初に見つかったものが使用される。なお、Linuxの場合、通常XF86Setupで変更されるファイルは/etc/X11/XF86Configだが、これは/usr/X11R6/lib/X11/XF86Configからリンクされている。

```
/root/XF86Config          （スーパーユーザーの場合のみ）
/etc/XF86Config
/usr/X11R6/lib/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.hostname
```

●問題の発見

Xがうまく起動しないなど、なにか問題がある場合は、

まず現在のXF86Configのどの部分でエラーが出ているのかを調べることが重要となる。これを調べるには、Xサーバのprobeonlyオプションを利用しよう。

```
# X -probeonly &> x.log
```

このようにすることで、x.logというファイルにエラーメッセージが出力されるのでよく読んでみよう。

XF86Configは基本的に**表1**にある7つのセクションから構成されている。Xサーバがうまく動かない場合は、「Monitor」「Device」「Screen」の3つのセクションの設定が間違っている場合が多い。ここでは、この3セクションを中心に見ていこう。

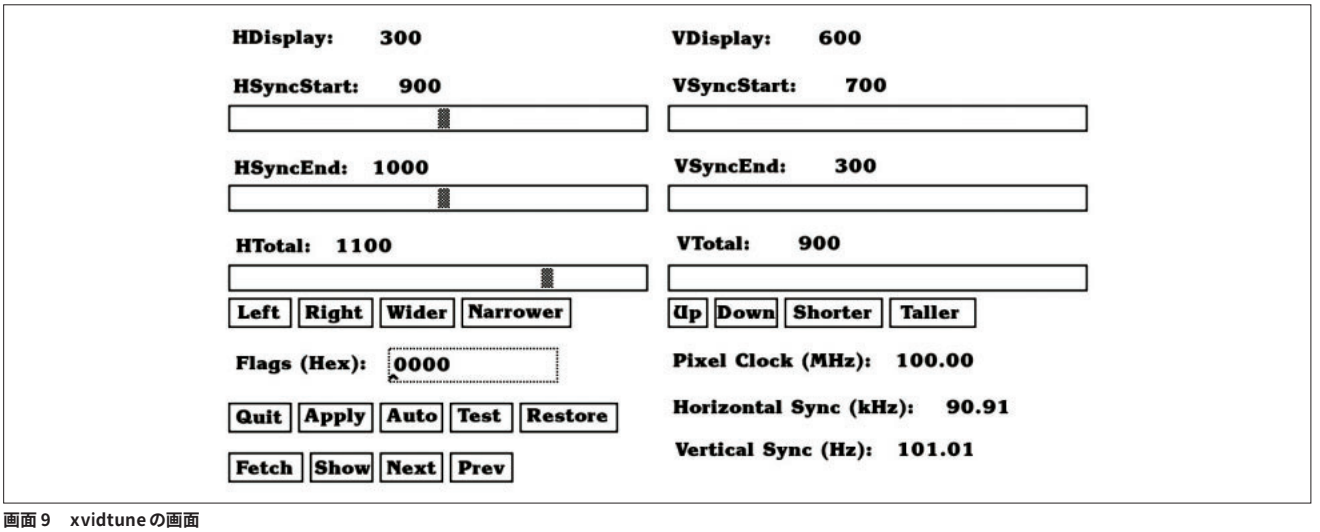
●Monitorセクション

モニタの設定はかなり重要な部分となる。というのは、このセクションを正しく設定していないと、ハードウェアが表示可能なビデオモードでも、Xサーバが立ち上がらなくなってしまうからだ。

セクション	説明
Files	フォントやカラーデータベースのファイルのパス。ここで指定されているファイルがまったく見つからないと、Xサーバは起動しない
ServerFlags	XF86SetupのOtherタブで設定する、Xサーバを強制終了するかどうかなどの設定
keyboard	キーボードの設定。XF86SetupのKeyboardに対応する
Pointer	マウスやその他のポインタデバイス（タブレットなど）の設定。XF86SetupのMouseに対応する
Monitor	モニタの設定。XF86SetupのMonitorに対応する
Device	グラフィックスカードの設定。XF86SetupのCardに対応する
Screen	Monitorセクションと Deviceセクションの関連づけ

表1　XF86configの各セクション

リスト1にMonitorセクションの例を示す。この中でもModelineの内容が最もわかりにくいだろう。これは各解像度でのモニタ走査周波数などの情報を記述している。ここで記述された解像度が、このモニタで利用可能だということになる。逆にいえばここに書いていない解像度ではXサーバを動作させることはできないということだ。特にノ



画面9　xvidtuneの画面
本来は表示を調整するためのツールだ。ここで確認したHsyncStartなどの値をXF86Configに記述することによって、モニタ上の表示位置を微調整することができる。

▼リスト1　Monitorセクションの設定例

```
Section "Monitor"
    Identifier  "RD17G3"              …Screenセクションでモニタの判別に使う
    VendorName  "MITSUBISHI"          …製造社名
    ModelName   "RD17G3"              …モデル名
    HorizSync   31.5-82.0             …水平走査周波数
    VertRefresh 40-100                …垂直走査周波数

    Modeline "1280x1024" 135    1280 1312 1416 1664  1024 1027 1030 1064      …各解像度ごとの設定
    Modeline "1024x768" 98.9  1024 1056 1216 1408   768 782 788 822 -HSync -VSync
    Modeline "800x600" 69.650 800  864  928 1088   600  604  610  640 -HSync -VSync
    Modeline "640x480" 36     640  696  752  832   480  481  484  509 -HSync -VSync
EndSection
```

ートパソコンで、特殊な画面の大きさのものを利用している場合は、このセクションを自分で変更しなくてはならないことが多い。

ただし、解像度以外の値はかなり難解だろう。なにもないところからこの行を書くのは難しいので、/usr/X11R6/lib/X11/doc/Monitorsに記載されているいくつかのモニタの設定値を参考にトライしてみてほしい。もちろん、XF86Setupで設定できた解像度については、すでにModeline行は記載されているはずだ。

Xが起動したら、xvidtuneを使いそれぞれの値を確認し微調整することができる。これらの値はxvidtuneで表示される以下の値に対応している。

解像度 `PixelClock HDisplay HsyncStart HsyncEnd HTotal VDisplay VsyncStart VsyncEnd VTotal Flags`

●Deviceセクション

ここには、グラフィックスカードの設定が記載される（**リスト2**）。この項目については自動認識される確率が高いので、実際に自分で指定しなくてはならない項目はほとんどないだろう。

もっとも、一部のグラフィックスカードについては特殊な設定が必要になる。それらについては、/usr/X11R6/lib/X11/doc/README や同じディレクトリにあるドキュメントを参考に記述するしかない。

●Screenセクション

ここは前述の、Monitorセクションと Deviceセクションを関連づけて、実際にどのモードで起動するかが記載されている（**リスト3**）。

Xサーバはまず、DefaultColorDepthで指定した色数のDisplayサブセクションを見る。そして、Modesに記述された解像度を行頭から順に見て、Monitorセクションの Modelineと合致するものを調べる。最初に合致した解像度で、Deviceで指定されたグラフィックスカードを使い表示しようとする。ハードウェアの性能に比べて、Xの解像度が低いようであれば、Modesと Monitorセクションの Modelineの記述を見直してみよう。

なお、（Identifierの異なる）複数の Monitorセクションを記述しておき、使われるモニタによって Screenセクション内の Monitorの値を変更するといったことも可能だ。

また、リストには存在しないが、Displayサブセクションの中に仮想スクリーンを指定する Virtualというキーを記述することも可能だ。

```
Virtual 1024 768
```

上のように記述すると実際のモニタが800×600のサイズだったとしても、画面全体をスクロールさせながら1024×768のサイズを使うことができる。

▼リスト2　Deviceセクションの設定例

```
Section "Device"
    Identifier  "Matrox G200"      …カード名
    VendorName  "Matrox"           …製造社名
    BoardName   "G200 AGP"         …モデル名
    VideoRam    8192               …VRAM容量（Kバイト）
EndSection
```

▼リスト3　Screenセクションの設定例

```
Section "Screen"
    Driver     "svga"                   …使用するXサーバ
    Device     "Matrox G200"            …カードの指定（Deviceセクションの Identifierで指定した名前）
    Monitor    "RD17G3"                 …モニタの指定（Monitorセクションの Identifierで指定した名前）
    DefaultColorDepth    24             …デフォルトの色数
    Subsection "Display"
        Depth     8     …色数
        Modes    "1280x1024" "1024x768" "800x600" "640x480"    …解像度
    EndSubsection
    Subsection "Display"
        Depth     16    …色数
        Modes    "1280x1024" "1024x768" "800x600" "640x480"    …解像度
    EndSubsection
    Subsection "Display"
        Depth     24    …色数
        Modes    "1280x1024" "1024x768" "800x600" "640x480"    …解像度
    EndSubsection
EndSection
```

Part 3

XFree86 4.0の新機能と設定

文：山岸典将
Text : Norimasa Yamagishi

今年3月にXFree86 4.0がリリースされた。XFree86 3.3系列はX11R6.3系列をベースに開発されていたが、XFree86 4.0は、X11R6.4をベースにしている。これ以外にもさまざまな新機能が搭載されている。ここでは、XFree86 4.0の新機能を紹介して、その後、インストールについて解説しよう。

XFree86 4.0の新機能

X11R6.3からX11R6.4への変更点は、ほとんどがそのまま4.0の新機能となる。X11R6.4の新機能としては、複数のディスプレイを1つのデスクトップとして利用するXinerama、カラーマップの利用規定であるTOG-CUP（the Open Group Colormap Utilization Policy）、Xクライアントのリソースを再起動なしに変更できるRCM（Resource Configuration Management）があげられる。

そして、XFree86 4.0固有の新機能は、主に以下のものがあげられる。

●モジュールのダイナミックロード

XFree86 3.3では、Xサーバのプログラムがグラフィックスチップによって異なっていたが、4.0では、Xサーバプログラムが1つとなった。そして、必要とされる機能や各ハードウェアに依存した部分は、ダイナミックにロードされる個別のモジュールとして用意されている。

●XAAの見直し

アクセラレータチップの機能を利用して描画を高速化するXAA（XFree86 Acceleration Architecture）関係のコードが新たに書き起こされ、速度の改善、機能強化がなされた。ただし、この書き直しにより、3.3系列でサポートされていたグラフィックチップの中には、4.0ではサポートされなくなったものもある。

●マルチヘッドのサポート

マルチヘッドと呼ばれる、複数のディスプレイや入力デバイスを利用できる機能が追加された。このマルチヘッドをX11R6.4で実装されたXineramaとともに利用することで、複数のディスプレイを1つのデスクトップとして利用できるようになった。

●VESA DCCサポート

VESA DCCとは、ディスプレイとグラフィックスカードの双方向の通信機能の規格だ。これにより、今までマニュアルを見て入力しなければならなかったディスプレイの仕様などを、自動認識することが可能になる。もっとも、これはその通信機能がサポートされただけであり、現在はまだすべての仕様が自動認識されるというわけではない。

●フォントフォーマット追加

新規に追加されたのは、TrueTypeフォントとCIDフォントのサポートだ。日本において特に重要なのはTrueTypeフォントのサポートだろう。XFree86 4.0では標準でX-TrueType、xfsftに対応し、XFree86 3.3のようにX-TrueTypeパッチをあてなくてもスケーラブルなTrueTypeフォントのラスタライズが可能だ。

●Unicodeサポート

Unicodeサポートにより、Unicodeでエンコードされたヨーロッパ系のいくつかの言語をxtermで表示することが可能になった。またUnicodeのフォントも追加されている。

●GLX／DRIのサポート

これらにより3D描画の高速化が期待できる。GLXはMesa 3Dライブラリを利用することによりOpenGLの表示を行う。またDRI（Direct Rendering infrastructure）は、グラフィックチップの3Dアクセラレーション機能をXFree86で利用するためのシステムだ。

現時点では、4.0の新機能はすぐユーザーのメリットに結び付くものではない。これらの機能の多くは、あくまでも、基礎的な部分を用意したに過ぎないからだ。

安定したX環境を求める場合には、まだその信頼性が実証されていないXFree86 4.0はお勧めできない。しかし、新しい技術を追い求めたいならば、4.0を試してみる価値はあるだろう。

設定ファイルXF86Configの変更点

新機能への対応のため、XF86Configについても記述方法が変わった。大きく変わった部分は、以下のとおりだ。

なお、/usr/X11R6/lib/doc/XF86Config.egというサンプルも用意されているのでそちらも見てほしい。

●Moduleセクション

ここは、ロードするダイナミックモジュールを定義する。TrueTypeやGLX、DRIの利用などもモジュールとして実現されているので、ここに定義しなくてはならない。

●Filesセクション

フォントやカラーデータベースの定義をする。また、モジュールを標準以外のディレクトリにインストールした場合もここで定義できる。

●ServerFlagsセクション

XFree86 3.3と同様に、サーバの動作に関する定義はここに書く。記述内容に関して大きな変更はない。

●InputDeviceセクション

キーボードやマウスなど、入力デバイスの定義はInputDeviceセクションに統合された。ここには、複数のキーボードやマウスなども定義できる。それぞれのデバイスについて別々のInputDeviceセクションを作成し、IdentifierとDriverを書けばよい。Driverはキーボードの場合は「keyborad」、マウスの場合は「mouse」だ。それぞれのデバイスのオプションは、4.0になって記述方法が若干変わっている。

なお、XFree86 3.3との互換性を確保するため、KeyboardやPointerセクションも残されている。

・InputDeviceセクションのキーボードの定義の例

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "keyboard"
    Option "XkbRules" "xfree86"
    Option "XkbModel" "jp106"
    Option "XkbLayout"        "jp"
EndSection
```

●Monitorセクション

Monitorセクションについては、3.3からの変更はほとんどない。

●Deviceセクション

Deviceセクションの大きな変更は、BusIDというエントリーが加わったことだ。これはPCI／AGPのカードが複数あった場合、どのカードに対応するかを判別するために使われる。カードが1つの場合は特に必要ない。

●ServerLayoutセクション

4.0では、マルチヘッドをサポートしたため、複数のディスプレイや入力デバイスに対応できる。そのため、利用するデバイスごとにServerLayoutセクションを書く。そして、Screen、Monitor、InputDeviceセクションで定義したIdentifierの組み合わせを書いておく。

-layoutオプションで使用するレイアウトを決定でき、指定がなければ最初に定義されたセクションが利用される。

・ServerLayoutセクションの例

```
Section "ServerLayout"
    Identifier   "Server1"
    Screen       "screen1"
             :
    InputDevice  "keyboard1"
    InputDevice  "Mouse1"
             :
    options
             :
EndSection
```

●DRIセクション

DRIの設定を記述する。

XFree86 4.0のインストール

XFree86 4.0を標準のX Window Systemとして採用しているディストリビューションはまだない。そのため、4.0を使うには、FTPやCD-ROMを利用してパッケージを入手し、自分でインストールする必要がある。FTPで入手する場合は、XFree86 ProjectのWebページhttp://www.xfree.org/を見て近くのミラーサイトを探すとよいだろう。Linux用のバイナリは、libc5、glibc 2.0、glibc 2.1対応のものが用意されている。また、本誌5月号の付録

CD-ROMにソースコードと、glibc 2.0、glibc 2.1用のバイナリを収録しているので、お持ちの方はご利用いただきたい。

今回使用したディストリビューションはTurboLinux Workstation 日本語版 6.0なので、glibc 2.1用のものを使用することになる。インストールを開始する前に、XFree86 4.0が自分の使うグラフィックスカードに対応するかをRelease Notesなどで確認するのをお忘れなく。

●準備とインストール

ここでは、glibc 2.1用のバイナリパッケージを例としてインストール作業を説明する。すべての作業はスーパーユーザーになって行う。

まず、XFree86 3.3用のXF86Configのバックアップをとっておこう。これは後で4.0用の設定ファイルを作成するのにも使う。

```
# cp /etc/X11/XF86Config ~/XF86Config.33
```

4.0をそのままインストールすると3.3の環境が上書きされてしまうので、3.3の環境を保存しておこう。そのためには、3.3がインストールされているディレクトリの名前を変更する。

```
# mv /usr/X11R6 /usr/X11R6.old
```

3.3環境の保存ができたら、XFree86 4.0のバイナリパッケージを展開したディレクトリに移動し、インストールスクリプトを実行する。

```
# cd Linux-ix86-glibc21
# sh Xinstall.sh
```

スクリプトを実行すると、現在利用しているX Window Systemの環境をバックアップしておくことを勧めると表示され、インストールを続行するか尋ねられる。すでにXFree86 3.3.x環境をバックアップしていれば「y」を入力し、あとは質問に答える形でインストールをしよう。

●設定ファイルXF86Configの作成

インストールが終了したら、いよいよXFree86の設定に入る。しかし、XFree86 4.0には3.3系列で用意されていたXF86Setupがまだ存在しない。そこで、コマンドラインから対話的に使うxf86configコマンドを使ってXF86Configを作成することにする。

xf86configの欠点は、モニタの細かい設定ができないこと、古いグラフィックスカードしかサポートしていないことだ。そこで、まずxf86configで基本的なXF86Configを作成し、モニタの設定は3.3のものを流用する。そしてグラフィックスカードの設定は、4.0のXサーバに搭載された、XF86Configを「ある程度」作成する機能を利用することにする。では、xf86configを実行しよう。

```
# xf86config
```

マウスとキーボードの設定は慎重に行う。モニタとグラフィックスカードの設定はあとから入れ替えてしまうので適当なものを選べばよい。設定が終わると、/etc/X11/XF86Configが作成される。

次に、Xサーバの設定ファイル作成機能を利用して、グラフィックスカードの設定ファイルを作成しよう。

```
# XFree86 -configure
```

これを実行すると、ホームディレクトリにXF86Config.newというファイルが作成される。

さて、この時点で以下の3つのXF86Configファイルが存在することになる。

/etc/X11/XF86Config	元になる4.0用の設定ファイル
~/XF86Config.new	グラフィックスカードの設定の元にするファイル
~/XF86Config.33	モニタの設定の元にする3.3用の設定ファイル

これらを統合してXFree86 4.0を利用するための設定ファイルを作成しよう。まず、/etc/X11/XF86Configの Monitorセクションと Deviceセクションをすべて削除する。そして~/XF86Config.newの Deviceセクションと、~/XF86Config.3.3の Monitorセクションを、/etc/X11/XF86Configに書き込む。

そして、/etc/X11/XF86ConfigのScreenセクションの DeviceとMonitorをそれぞれのセクションのIdentifierで指定されているものに書き換える。

これで設定ファイルXF86Configの作成は完了だ（**リスト1**）。

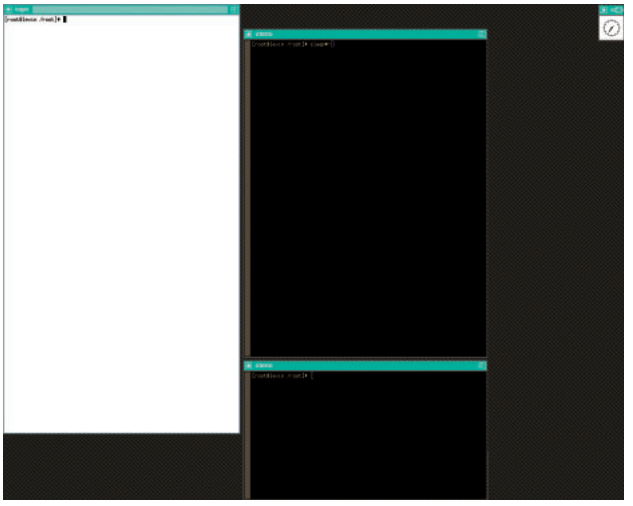
XFree86 4.0の起動

設定ファイルが完成したら、あとは立ち上げるだけだ。

```
# startx
```

ウィンドウマネージャをtwmとしてXが起動する（**画面1**）。起動しない場合はログファイル/var/log/XFree86.0.logを参照すると、設定の問題を見つけることができるだろう。

うまく起動したらウィンドウマネージャなどの設定をすれば作業は終了だ。それでは成功を祈る。



画面1　XFree86 4.0の画面
XFree86を入れ換えることで、ウィンドウマネージャがXFree86標準のtwmに換わる。

▼リスト1　作成した/etc/X11/XF86Configの一部

```
Section "Monitor"                ...Monitorセクションは~/XF86Config.33のものに入れ替える
    Identifier  "RD17G3"
    VendorName  "MITSUBISHI"
    ModelName   "RD17G3"
                      :
                      :
    Modeline "640x480" 36     640  696  752  832   480  481  484  509 -HSync -VSync
EndSection
                      :
                      :
Section "Device"                 ...Deviceセクションは~/XF86Config.newのものに入れ替える
        ### Available Driver options are:-
        #Option     "SWcursor"
                      :
                      :
        #Option     "Rotate"
        Identifier  "G200"
        Driver      "mga"
        VendorName  "Matrox"
        BoardName   "MGA G200 AGP"
        BusID       "PCI:1:0:0"
EndSection

Section "Screen"
    Identifier  "Screen 1"
    Device      "G200"           ...DeviceセクションのIdentifierの値を指定する
    Monitor     "RD17G3"         ...MonitorセクションのIdentifierの値を指定する
    DefaultDepth 24
                      :
                      :
    Subsection "Display"
        Depth       24
        Modes       "640x480" "800x600" "1024x768"     ...優先したい順に並べ替える
        ViewPort    0 0
    EndSubsection
EndSection
```

Part 4

AGPグラフィックスカード8機種をLinuxで使う

文：Linux magazine ラボ
Text : Linux magazine Lab.

近ごろでは大手メーカーのPCも低価格化が進み、安く上げるためにPCを自作する意味も薄れている。それでもPCを自作するユーザーが減らないのは、完成後もパーツを交換していくという楽しみがあるからだ。

なかでもグラフィックスカードは、交換することで、解像度、色数、3D描画機能などが劇的に改善する場合もあるので、交換しがいのあるパーツといえるだろう。

2Dの描画性能に関しては、どのカードを用いても十分な速度が得られるようになっており、ほとんど差がない状態だ。現在、差別化のポイントは、画質と3D描画機能である。

画質についてはユーザーの主観や好みがあるので、万人が「良い」と評価するベストの製品は選びにくい。それに対して3D描画機能は、速ければ速いほどよいわけだし、結果が数値で表されるため、優劣は簡単に判定できてしまう。メーカーにとっては厳しい分野である。

どのメーカーも短いサイクルで新製品を投入するため、競争は非常に激しい。また高性能3Dエンジンの設計と、対応するドライバの作成には高い技術が要求されるため、脱落するメーカーも多く、アクセラレータチップのメーカーは淘汰が進んでいるというのが現状だ。

Linuxの場合

以上の状況は、基本的にWindows環境での話であり、Linuxでは少々異なっている。ほとんどのディストリビューションで標準のX Window Systemとして採用されているXFree86のドライバは、メーカーによって公開されている情報を元に作成されている。そのため、ドライバ（Xサーバ）のチューニングの程度はアクセラレータごとに大きく異なっている。また、多く使われているカードほど開発者も多くなるので、人気のあるカードほどドライバもチューニングが進む傾向がある。

3D描画機能に関してはどうであろうか。X Window System上での3D環境は、OpenGLまたは、フリーの互換環境であるMesa3Dが一般的だ。XFree86 3.3.xまでは、統一されたインターフェイスが用意されていなかったため、3D描画をハードウェアでサポートする方法は、メーカーやプロジェクトごとに異なっていた。

このような状況下で、性能的には3dfx社のVoodoo3／Bansheeが頭ひとつ抜きん出ていた。それは、同社がGlideと呼ばれる3Dライブラリを有しており、これを利用してMesa3Dのハードウェアアクセラレーションを実現していたからだ。

しかし今年の3月に発表されたXFree86 4.0では、DRI（Direct Rendering Infrastructure）やGLXといった仕組みが提供されており、ハードウェアによる3D描画機能が利用しやすくなっている。

最新グラフィックスカードカタログ

今回のテストでは5月現在、パソコンショップで購入可能なグラフィックスカードを、アクセラレータチップ別に7＋1機種用意した。「＋1」というのは、チップセットにアクセラレータ機能を統合したインテルのi810を用いたマザーボードである。先月号の自作PC特集で紹介したようなベアボーンキットや、メーカー製の低価格PCに採用されることが多く、普及が進んでいると考えて、取り上げることにした。

機種選定の基準は、実際に購入可能なAGP（Accelerated Graphics Port）バスの製品である。PCIバス用のグラフィックスカードは、種類が少ないうえに、マルチモニタ環境での利用を想定しないかぎり、積極的に選択する理由がないと考えたからだ（AGPバスを持たないマザーボードを利用している場合は別だが）。

購入時にXFree86で利用可能かどうかは、特に確認しなかった。これはちょっとした賭けだったが、最終的にすべての機種でXFree86を利用できた。設定に工夫が必要な機種については、個別に記載した。

今回紹介する製品には、パッケージに「Linux対応」をうたったものはないので、メーカーへの問い合わせは遠慮していただきたい。また編集部で動作を保証するものでもないので、こちらもご了承いただきたい。

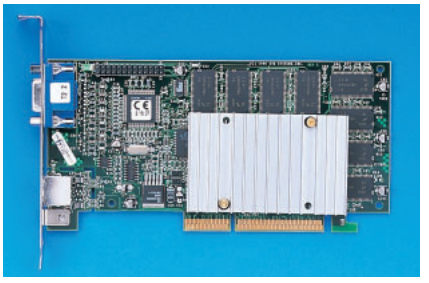
3dfx Interactive, Inc.

Voodoo3 3000 AGP

アクセラレータチップ：Voodoo3

購入価格：1万7800円

http://www.3dfx.com/



3dfx社は、3D専用のVoodooアクセラレータシリーズで知られていたが、Voodoo Banshee以降は他社と同じく2D／3D統合型のアクセラレータのみをリリースしており、本カードも2D／3D両方をハードウェアで描画できる。以前はグラフィックスカードメーカーにアクセラレータを提供していた同社だが、現在ではグラフィックスカードの自社生産のみを行っている。そのため「Voodoo3」は、アクセラレータと同時に、グラフィックスカードの名称でもある。

Voodoo3シリーズは高性能の代償として、非常に発熱量が多いことで知られている。本カードにもかなり大きめなヒートシンクが取り付けられているが、動作中は触れないほど高温になる。長時間にわたって運用するつもりなら、ケース内の換気に注意する必要があるだろう。

XFree86 3.3.6／4.0ともにサポートされており、2D環境のみなら、特にトラブルもなくインストールが完了し、使用できる。3Dアクセラレーションは、同社のWebサイトで提供されているGlide、XFree86 4.0、Direct Rendering Managerモジュールの3つのrpmパッケージをインストール後、ライブラリのリンクを調整することで利用できる。

3dfx Voodoo3
Voodoo2の後継というよりは、Bansheeの後継。本カードでは166 MHzで動作している。

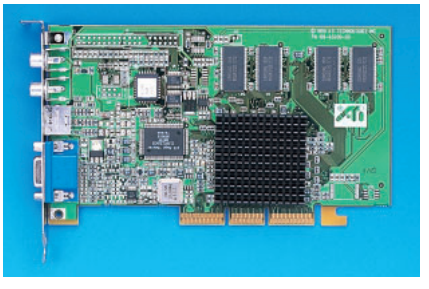
ATI Technologies Inc.

ATI RAGE FURY PRO

アクセラレータチップ：RAGE 128 PRO

購入価格：1万8800円

http://www.ati.com/



ATI社は最大規模のグラフィックスカードメーカーのひとつで、同社のグラフィックスカードは、大手メーカー製のPCへのOEMが多いことで知られている。RAGE 128は、同社初の128ビットコアを採用したアクセラレータチップで、RAGE 128 PROはその高クロック版である。99年の7月にリリースが予定されていたが、半年以上遅れて今年の2月に市場に出回り始めた。

ATI RAGE 128はXFree86 3.3.6でサポートされているが、本カードはRed Hat Linux 6.2に含まれているXFree86 3.3.6のXサーバが起動できなかった。そこで独SuSE社が提供する最新のXサーバを手に入れて試したが、結果は変わらなかった。インストール時に「RAGE 128（PF）」と認識はされるので、RAGE 128 PROは単なる高クロック版ではなく、何らかの変更が施されている可能性がある。

その代わりXFree86 4.0では最初からサポートされており、普通にインストールしたRed Hat Linux 6.2を、rawhide（Red Hat Linuxの開発版）のXFree86 4.0 RPMパッケージを使ってアップデートすることで、X Window Systemを利用できた。現状では3Dアクセラレーションは利用できないが、対応が予定されている。

ATI RAGE 128 PRO
ATI社のRAGE 128シリーズの第二世代に相当する。コアを高速化し、AGP 4xモードに対応している。

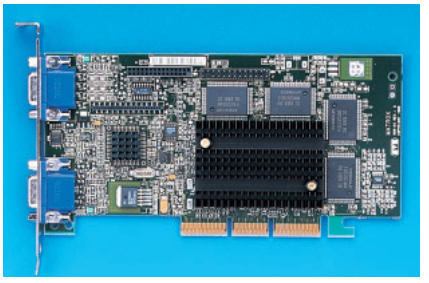
Matrox Electronic Systems Ltd.

Millennium G400 DH

アクセラレータチップ：G400

購入価格：1万8800円

http://www.matrox.com/



Matrox社の製品は初代Millennium以来、高速／高画質の評判が高く、昔からの自作PCユーザーなら1枚は同社のカードを持っていることだろう。

G400は、本格的3D描画機能を持った同社初の製品だ。また「DH」という型番は「Dual Head」を表し、1枚のカードで2台のモニタへの表示ができるという特徴を持つ。残念ながら、現状ではG400 DH 1枚でXFree86 4.0のマルチヘッド機能を用いることはできないようだ。

初代Millennium以降の同社製品に対応するXサーバは、XFree86 3.3.6／4.0ともに非常にチューニングが進んでおり、2D環境は快適に利用できた。

3Dアクセラレーションについては、XFree86 4.0での対応が予定されている。XFree86プロジェクトが提供しているソースを見ると、カーネルモジュールを収録したディレクトリに「mga」で始まるファイルがたくさん置かれているので、マイナーバージョンアップ時に対応されると思われる。

またXFree86 3.3.6環境では、accelerated GLX projectによるUtah-GLX（後述）を用いることで、G400のハードウェア機能を用いた3D描画を行うことが可能だった。

Matrox G400
Matrox社製品の高画質には定評があり、G400も画質で選択されることが多いという。

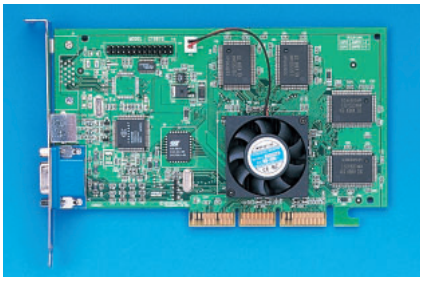
クリエイティブメディア

3D BLASTER RIVA TNT2 Ultra

アクセラレータチップ：Riva TNT2 Ultra

購入価格：1万5800円

http://www.creative.com/



クリエイティブメディアは、音源カードのSound Blasterを販売している会社だが、今ではグラフィックスカード、MP3プレイヤーなど多くの製品を扱っている。最新のアクセラレータを用いたグラフィックスカードを、いちはやく製品化することで知られている。

TNT2 Ultraは、NVIDIA社のアクセラレータで、1クロックに2ピクセルのテクスチャ処理（TwiN Texel）が可能なことから「TNT」と名付けられている。なかでもTNT2 Ultraは、シリーズ中最高速のコアクロック（150MHz）で動作する。

XFree86 3.3.6では、インストール時に自動認識され、そのまま問題なく利用できた。また4.0でも、XFree86本体をインストールした後に、NVIDIAのWebサイト（http://www.nvidia.com/）で提供されているカーネルモジュールとGLXライブラリをインストールすることで、2D／3Dともに完全に動作することを確認した。3D描画はきわめて高速で、動作も安定していた。

同社は3.3.6用にもドライバを提供していたが、それほど高性能ではなく、ドキュメントにも率直に「それほど速くない」と記されていた。4.0用ドライバのドキュメントには、当然そのような記述はない。NVIDIAの本気が感じられる。

NVIDIA RIVA TNT2 Ultra
TNT2シリーズの最上位品であるUltraは、150MHzという高クロックで動作する。

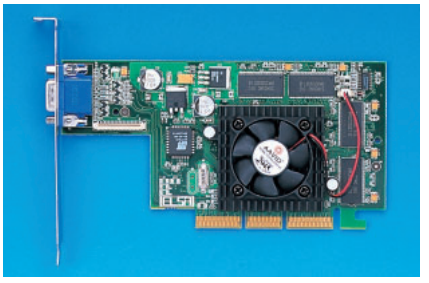
ELSA AG

ELSA ERAZOR X

アクセラレータチップ：GeForce256

購入価格：2万1500円

http://www.elsa.com/



ELSAはドイツの会社で、一般的な製品に加えて、CAD／CGなど業務用の高級グラフィックスカードも扱っている。ハイエンド製品が好まれる日本市場では人気のあるメーカーだ。本カードは、ゲームなど一般的な用途向けのラインに含まれる製品だが、用途によっては業務用に迫る性能を発揮する。

GeForce256は、NVIDIAが開発したアクセラレータで、3D描画に加え、座標計算（ジオメトリ演算）もハードウェアで行える、一般的なPC用のアクセラレータとしては初めての製品だ。同社では3Dアクセラレータではなく、GPU（Graphic Processing Unit）と呼んでおり、CPUに匹敵するほどの機能があることを示している。

NVIDIAのアクセラレータは、RIVA 128以降GeForce256まで、共通のドライバで動作する。そのためRIVA TNT2 Ultraと同様に、XFree86 3.3.6ではインストール時に自動認識され、利用できるし、4.0でもXFree86の本体をインストール後、同社が提供しているカーネルモジュールとGLXライブラリをインストールすれば4.0上で2D／3Dともに利用可能だった。

3D描画の性能は非常に高く、OpenGL（Mesa3D）の性能基準を大幅に上げた感がある。

NVIDIA GeForce256
PC用アクセラレータで、ハードウェアによるジオメトリ演算を可能にした最初のチップだ。

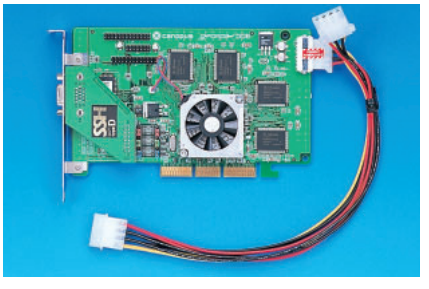
カノープス

SPECTRA 7400 DDR

アクセラレータチップ：GeForce256 DDR

購入価格：4万1800円

http://www.canopus.co.jp/



カノープスは、独自のハードウェア設計技術とドライバにより、チップメーカーのリファレンスデザインそのままの製品よりも高性能なグラフィックスカードを開発・販売している。また画質についても気を使っており、本カードでもコネクタに至る部分を別基板にするなどの工夫が施されている。また消費電力が大きいため、ケース内の12V電源コネクタから電力を供給するようになっている。

このような独自のブランドイメージを持つため、同社の製品は、ほかのメーカーよりも高めの価格にもかかわらず、新製品が発売されるとすぐに売り切れてしまうほどの人気だ。

アクセラレータチップのGeForce256 DDRは、ハードウェアによるジオメトリ演算をサポートし、クロック信号の両エッジに同期可能なDDR（Double Data Rate）メモリに対応している。大量のテクスチャデータを必要とする3Dアクションゲームなどでは、効果がある。それ以外はGeForce256とまったく同じで、XFree86 3.3.6／4.0ともに利用できた。

今回のテストでは、通常のSDRAMを用いているELSA ERAZOR Xとほぼ同等の結果が得られた。メモリ帯域が必要な用途では差がでるだろう。

NVIDIA GeForce256 DDR
DDR（Double Data Rate）メモリに対応しており、広帯域が必要な処理に威力を発揮する。

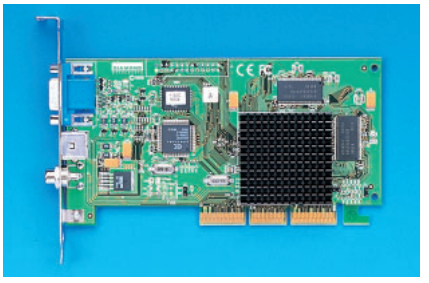
ダイアモンド・マルチメディア・システムズ

VIPER Ⅱ

アクセラレータチップ：Savage2000

購入価格：1万7800円

http://www.diamondmm.com/



ダイアモンド・マルチメディアは、グラフィックスカードメーカーとして始まり、現在では各種周辺機器を開発・販売している。アクセラレータチップを開発しているS3社に買収されたため、主に同社のチップを用いたカードを製造していた。

S3社は、PC用の2Dアクセラレータチップを最初に開発した会社だが、そのグラフィックスLSI開発部門をVIA Technologiesとの合弁会社に移管することになった。同社がアクセラレータの開発をやめてしまうというのは、グラフィックスカードの競争の厳しさを象徴するようなできごとといえよう。

Savage2000は3Dの描画に加えて、3D描画時の座標計算（ジオメトリ演算）もハードウェアで行えるはずだった。実際には、Windows版でさえも、ジオメトリ演算に対応したドライバがリリースされておらず、今後新しいドライバが出る可能性はないだろう。

XFree86上でも事情は同様で、3.3.6はSavage2000に対応しているため、インストール時に自動で認識され、動作させることができた。しかし4.0では、Savageシリーズはサポートされていないので、同シリーズを利用しているユーザーは、XFree86 3.3.6を使うことになる。

S3 Savage2000
S3社はアクセラレータ開発から撤退したため、おそらくこれが最後の製品であろう。

PC CHIPS

SF-BKi810

アクセラレータチップ：i810

購入価格：2万3980円

http://www.pcchips.com/

インテルのi810は、チップセットのMCH（Memory Controller Hub）と2D／3Dグラフィックスコアを1つのチップに統合した製品である。メモリは、メインメモリの一部を使うようになっており、独立したグラフィックスカードなしでPCが完成するようになっている。そのため低価格でコンパクトなベアボーンキットでシェアを広げている。またメーカー製の低価格PCでも採用されることが多いようだ。

i810はXFree86 3.3.6でサポートされているが、「agpgart.o」というAGP用のドライバモジュールをマニュアルで入れる必要があり、使うにはちょっと工夫がいる。しかし正しく設定できれば、安定して動作させることができた。

Matrox G400と同様に、accelerated GLX projectによるUtah-GLXにも対応しているはずだが、今回テストした環境では、Utah-GLXを組み込んだ状態でX Window Systemを動作させることができなかった。

i810は、XFree86 4.0でも対応機種にあげられている。だが、編集部でRed Hat Linux 6.2をフルインストール後に、rawhide（Red Hat Linuxの開発版）のXFree86 4.0パッケージをインストールしてみたところ、ハードウェア情報を取得するためのXサーバのオプション（XFree86 -configure）でエラーが出てしまい、結局動作させることができなかった。

4.0の機能を利用した3Dアクセラレーションについても、現在開発が進められており、マイナーバージョンアップ時（4.0.1？）に対応されると予想される。

Part 5

ベンチマーク
～最適なグラフィックスカードは？～

文：Linux magazine ラボ
Text : Linux magazine Lab.

グラフィックスカードは、モニタへの出力という目に見える形でその違いがわかるため、交換したときの効果が大きく、自作PCユーザーが最も関心を寄せるパーツである。

今回揃えた7＋1機種のグラフィックスカードはいずれも、5月の時点で現行製品として容易に入手可能な製品だ。これらをLinux上のXFree86で用いたときの描画性能を比較し、ベストな製品を探っていく。

使用したプログラム

X Window System上の描画系ベンチマークプログラムには、定番と言えるほど一般的なものはない。2D描画能力のテストにx11perfが用いられることもあるが、これは直線や四角形などの基本的な図形の描画を繰り返すもので、実際の使用感を反映したベンチマークというよりは、ドライバプログラムのチューニング用途に適したものと言えるだろう。

Part3でも紹介したように、XFree86 4.0にはOpenGL（Mesa3D）の表示をX Window System上で行うためのGLXや、ハードウェアの3Dアクセラレーション機能を直接利用するDRI（Direct Rendering Infrastructure）が組み込まれており、各グラフィックスカードの3D描画性能についても気になるところだ。

だが、今までPC UNIX上でハードウェアアクセラレーション可能な環境がそれほど多くなかったせいか、X Window System上で3Dの性能を評価する一般的な方法もまだ確立していない。

今回Linux magazine ラボでは、2D描画能力の測定には、二之宮氏の作成したHDBENCH clone Ver 0.14.1を用いた。また3D描画能力の測定には、id Software社のQuake III Arena（デモ版）と、川瀬氏の作成したglclock Ver 5.0を使用した。なおglclockは、5月の時点ではVer 6.0のβ版が川瀬氏のWebサイトで公開されている（**表1**）。

テスト環境には、i810を搭載したベアボーンキットと標準的な自作PCを1台ずつ用意し、CPU、メモリの条件を揃えて測定を行った（**表2**）。ディストリビューションは、カーネル2.2.14を採用したRed Hat Linux 6.2（英語版）を使用している。XFree86 3.3.6は、ディストリビューションに含まれているRPMパッケージをそのまま使用し、4.0は「Rawhide」版のRPMパッケージを利用した。

Webサイト	URL
3dfx	http://linux.3dfx.com/
NVIDIA	http://www.nvidia.com/
Utah-GLX	http://utah-glx.sourceforge.net/（G400、i810などに対応）
glclock	http://www.daionet.gr.jp/~masa/glclock/（現在は6.0βを公開中）
HDBench clone	http://www.enjoy.ne.jp/~gm/program/hdbench/index-ja.html
Quake III Arena	http://www.quake3arena.com/

表1 関連URL

ハードウェア		
マザーボード	MSI MS-6309	PC CHIPS SF-BKi810（ベアボーン）
チップセット	VIA Apollo Pro 133A	Intel i810
CPU	Celeron 500MHz（66×7.5）	Celeron 500MHz（66×7.5）
メモリ	PC100 SDRAM 128Mバイト	PC100 SDRAM 128Mバイト
ハードディスク	Quantum FBlct17.3AT（17.3Gバイト）	IBM DPTA351500（15Gバイト）
グラフィックスカード	実験ごとに交換	チップセットに統合
ソフトウェア		
OS	Red Hat Linux 6.2	Red Hat Linux 6.2
カーネル	2.2.14	2.2.14
XFree86	3.3.6／4.0	3.3.6／4.0

表2 テスト用PCのスペック

設定方法

XFree86 4.0は、リリースされてからあまり時間がたっていないため、ハードウェアアクセラレーションを利用するドライバのインストール方法は、各カードによってまちまちである。

またaccelerated GLX projectによるUtah-GLXを組み込むことで、ハードウェアアクセラレーションを利用できるカードもあるが、Utah-GLXはXFree86 3.3.x用であり、これもまたインストール方法は異なっている。

今回のテストのために、Linux magazine ラボで行った設定方法を紹介するが、ハードウェア構成、ディストリビューション、ソフトウェアのバージョンの違いにより、ここの記述どおりにはいかない可能性があることを

お断りしておく。

実際に試してみる際には、プログラムのパッケージに付属しているドキュメント類や、配布しているWebサイトの情報ページを参考にしよう。また、うまく動かない場合には、エラーメッセージやログファイルを参照して、どこに問題があるのかを見極めて、解決していこう。

●3dfx Voodoo3 3000 AGP

3dfxのVoodoo3／Bansheeは、同社のグラフィックスカード専用の3Dライブラリであるを利用して、高速な3D描画を実現している。XFree86上でもこのGlideライブラリをラッパーを介して利用できる。3.3.x／4.0の両環境用のドライバ類が提供されているが、今回は4.0に対応したものを用いた（**表3**）。Red Hat Linux 6.2で全パッケージをインストールしたマシンに、上記のドライバ類をインストールする。以下の作業は、root権限が必要だ。

インストール手順は**リスト1**のとおりだ。XFree86の設定ファイルのバックアップを作成し、ソースRPMをリビルドする。できあがったパッケージをGlideライブラリ、専用のXFree86、ローダブルモジュールの順番でインストールする。その後、設定ファイルのバックアップから必要な設定を書き戻して終了だ。

以上のように設定した後、Mesa3Dを起動した時に、「必要なライブラリがない」というようなエラーが出ることがある。これはMesa3Dライブラリが更新されているのに、シンボリックリンクが以前のままになっていることが原因だ。**リスト2**を参考に修正すればよい。

●Matrox G400

acclerated GLX projectのUtah-GLXを用いると、XFree86 3.3.xでGLXを利用してハードウェアによる3D描画が実現できる。同プロジェクトのWebサイトによると、Matrox G400／G200やi810以外にATI Rage Pro、S3 VIRGEなどにも対応しているようだ。だがG400以外は、3Dの性能がそれほど高い製品ではないので、アクセラレーションが有効になってもあまり性能は出ないと思われる。

インストールは、Utah-GLXのWebサイトから、Utah-GLXとMesa3Dを組み合わせたソースRPMを取得して行っ

▼リスト1　操作手順（Voodoo3 3000 AGP）

```
# cp /etc/X11/XF86Config /etc/X11/XF86Config.orig   ← 設定ファイルのバックアップ
# rpm --rebuild Glide_V3-DRI-3.10-6.src.rpm
# rpm --rebuild tdfx_dri-4.0.00-1.src.rpm           ← リビルド
# rpm --rebuild tdfx_drm-1.0-1.src.rpm
# cd /usr/src/redhat/RPMS/i386
# rpm -Uvh Glide_V3-DRI-3.10-6.i386.rpm
# rpm -Uvh tdfx_dri-4.0.00-1.i386.rpm               ← インストール
# rpm -Uvh tdfx_drm-1.0-1.i386.rpm#
```

▼リスト2　リンクの修正の例

```
# cd /usr/X11R6/lib
# ls -l libGL*
     669 Mar 10 09:40 libGL.la*
      14 Apr 12 05:18 libGL.so -> libGL.so.1.2.0*
      14 Apr 12 05:18 libGL.so.1 -> libGL.so.1.2.0*
 3647156 Mar 10 09:40 libGL.so.1.2.0*
     622 Mar 10 09:40 libGLU.la*
      15 Apr 12 05:18 libGLU.so -> libGLU.so.1.0.0*   ← リンク間違い！
      15 Apr 12 05:18 libGLU.so.1 -> libGLU.so.1.2.0*
  306044 Mar 10 09:40 libGLU.so.1.2.0*
# ln -sf libGLU.so.1.2.0 libGLU.so                    ← 修正
```

▼リスト3　操作手順（G400）

```
# rpm -e Mesa-devel
# rpm -e Mesa
# rpm --rebuild glxMesa-20000328-1.src.rpm
# cd /usr/src/redhat/RPMS/i386
# rpm -Uvh glxMesa-20000328-1.i386.rpm
```

	ファイル名	内容
Voodoo3／Banshee	Glide_V3-DRI-3.10-6.src.rpm	Linux用Glideライブラリ
	tdfx_dri-4.0.00-1.src.rpm	専用のXFree86 4.0
	tdfx_drm-1.0-1.src.rpm	DRI用モジュール
Riva TNTシリーズ／GeForce256	NVIDIA_GLX-0.9-2.src.rpm	Mesa3Dライブラリ
	NVIDIA_kernel-0.9-2.src.rpm	DRI用ローダブルモジュール
	XFree86-4.0-0.12.i386.rpm など	XFree86 4.0本体（Rawhide版）
Matrox G200、G400、Intel i810	glxMesa-20000328-1.src.rpm	Utah-GLX＋Mesa3D

表3 機種別に必要なパッケージ　※バージョンナンバーは異なっている可能性がある。

た。**リスト3**のように、あらかじめMesa3Dのパッケージをアンインストールしておくほうがよいだろう。その後ソースRPMをリビルドし、インストールすればよい。

なお同様のやり方で、i810を搭載したマシンにもインストールを試みたが、Xサーバを起動することはできなかった。GLXの最新のスナップショット版、Mesa、i810用のagpgart.oなど考えつくものはすべて新しいバージョンに変更してテストを重ねたが、結果は同じだった。

▼リスト4　操作手順（RIVA TNT2 Ultra／GeForce256）

```
# ls XFree86*
XFree86-4.0-0.12.i386.rpm        XFree86-xdm-4.0-0.12.i386.rpm
XFree86-devel-4.0-0.12.i386.rpm  XFree86-xf86cfg-4.0-0.12.i386.rpm
XFree86-libs-4.0-0.12.i386.rpm   XFree86-xfs-4.0-0.12.i386.rpm
XFree86-tools-4.0-0.12.i386.rpm
# rpm -Uvh --force --nodeps XFree86*
# ls NVIDIA*
NVIDIA_GLX-0.9-2.src.rpm  NVIDIA_kernel-0.9-2.src.rpm
# rpm --rebuild NVIDIA_GLX-0.9-2.src.rpm
# rpm --rebuild NVIDIA_kernel-0.9-2.src.rpm
# cd /usr/src/redhat/RPMS/i386
# rpm -Uvh NVIDIA_kernel-0.9-2.i386.rpm
# rpm -Uvh NVIDIA_GLX-0.9-2.i386.rpm ―― 警告が表示される

# cd /usr/X11R6/lib
# mkdir orig
# mv libGL.so* orig
# cd /usr/X11R6/lib/modules/extensions
# mkdir orig
# mv libGLcore.a libglx.a orig          ―― ファイルの待避
# cd /usr/src/redhat/RPMS/i386
# rpm -Uvh NVIDIA_GLX-0.9-2.i386.rpm ―― 成功

# xf86config         ←/etc/X11/XF86Configを作成
# XFree86 -configure  ←~/XFree86Config.newを作成
```

▼リスト5　/etc/X11/XF86Configの修正

```
# This loads the GLX module
#   Load       "glx"  ―― 行頭の「#」を取る

EndSection

:
:

Section "Device"
    Identifier  "geforce256ddr"
        Driver      "nvidia"
        VendorName  "NVidia"
        BoardName   "GeForce DDR"
        BusID       "PCI:1:0:0"   ―― XF86Config.newの「Device」セクションに変更する
#    Driver      "vga"
        # unsupported card
    #VideoRam    32768
    # Insert Clocks lines here if appropriate   ―― 元の「Device」セクション
EndSection
```

● NVIDIA RIVA TNT2 Ultra／GeForce256

NVIDIAのRIVA TNTシリーズ、GeForce256は、同社が提供する専用のMesa3Dライブラリと、ローダブルモジュールを利用することで、ハードウェアによる3D描画を実現できる。これ以外にXFree86 4.0の本体が別途必要である。XFree86プロジェクトが提供するtarボールや、ディストリビューションのRPMパッケージを利用しよう。今回のテストでは、Rawhide版のRPMパッケージを利用した。

インストールはリスト4の手順で行う。XFree86 4.0のパッケージをインストール後、NVIDIAが提供するドライバをリビルドする。NVIDIA_kernel-0.9-2.i386.rpmは問題なくインストールできるが、NVIDIA_GLX-0.9-2.i386.rpmは、ファイルの依存関係に問題があると警告が表示されるはずだ。指摘されたファイルを待避してから、再度インストールしてみよう。

パッケージのインストール終了後に、設定ファイルを作成する。まずxf86configコマンドで/etc/X11/XF86Configを作成する。さらにXFree86の設定機能を利用して、~/XFree86Config.newを作成し、リスト5のように/etc/X11/XF86Configを修正する。

最後に/usr/X11R6/lib/modules/drivers内にあるnv_drv.oというモジュールを、別のディレクトリに移動しておく。これをしないと、NVIDIAのパッケージからインストールされるnvidia_drv.oがうまく読み込まれずにエラーになるからだ。

ベンチマーク結果

では、ベンチマークの結果を見ていこう。テストを行う際には、解像度を1024×768×16ビットカラー（65536色）、リフレッシュレート75Hzにしてサーバを起動している。Quake III Arenaデモ版は、640×480のフル画面で動作させた。glclockは、320×320のウィンドウで1秒あたりの表示画面数（フレームレート）で評価している。

●HDBENCH Clone

2D描画速度の評価には、HDBENCH Cloneを用いた。「VIDEO」のテストは、矩形、円、テキスト、スクロール、イメージの計5種類の項目がある。項目間で数字の大きさにばらつきがあるため、ATI RAGE IIcという古い（1995年の製品）グラフィックスカードでも測定を行い、同カードの値を1とした相対値で評価した。

まずXFree86 3.3.6上での結果を**グラフ1**に示す。どのカードともリファレンスのRAGE IIcを大きく上回っており、製品間の差もi810を除けばそれほど大きくないことがわかる。実際に使用している際にも、2Dの描画でストレスを感じることはまったくない。RAGE IIcでは大きいウィンドウを移動する時に少し動作が遅れたりするが、最も値の低いi810でもスムーズに動かせた。RAGE IIcはXF86_Mach64というXサーバを使用するが、このXサーバは2Dのアクセラレーションを行わないので、これだけの大差がついたと考えられる。

2Dの描画については、どのカードを選択してもストレスなく使用できるレベルに達している。また画質の点でも、特に気になるような問題点はなかった。X Window SystemでもWindowsと同様に、2D描画ではもはや差がつかないレベルに到達しているようだ。

さらにXFree86 4.0でも動作させることができたVoodoo3、RIVA TNT2 Ultra、GeForce256について、3.3.6／4.0間で比較した結果を**グラフ2**に示す。

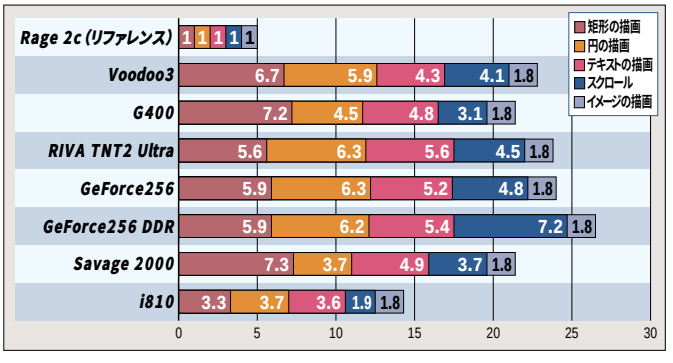
実際の使用感はともかく、ベンチマーク上はどのカードも4.0になってかなり性能が上がっていることがわかるだろう。4.0は、XAA（XFree86 Acceleration Architecture）を一新しており、その効果が明確に表れているといえよう。

●Quake III Arena

Quake III Arena（**画面1**）は、非常に高負荷の3Dアプリケーションであり、ハードウェアの3D描画機能が使えない環境で動かしても無意味だ。そのため実際にテストしたのは、Voodoo3、NVIDIAの3機種、G400の計5種類のカードのみだ。

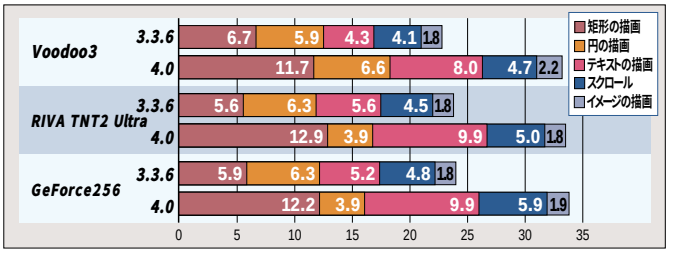
測定方法は、「`」（バッククオート）キー（日本語キーボードでは半角／全角キー）を押してコンソールを表示させ、

グラフ1　2D描画速度の比較（XFree86 3.3.6）

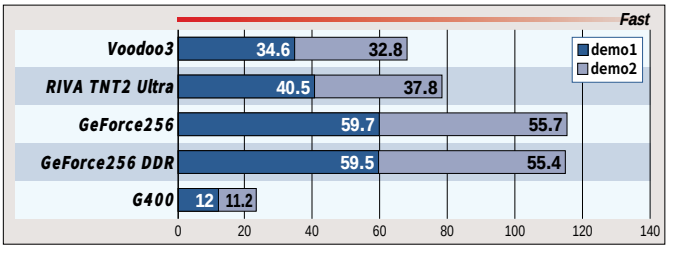


画面1
Quake III Arena

グラフ2　2D描画速度の比較（XFree86 3.3.6 vs 4.0）



グラフ3　Quake III Arena（fps）



```
/timedemo 1
```

と入力し、用意されている2種類のデモをそれぞれ実行して、1秒あたりの表示フレーム数を確認することで行った。

テキスチャの綺麗さ、ポリゴン数、光源の処理などの画面設定は、デフォルトのままで行っている。この条件で、Voodoo3やNVIDIAのチップを用いたカードは、実用的に使えるかどうかの基準である30フレーム／秒（fps）を超えている（**グラフ3**）。特にGeForce256を用いた2機種は、約60fpsにも達しており、別格の高性能である。またGeForce256では、画面設定をすべて最高にして、1024×768×32ビットカラーに設定しても、DEMO1で57.7fpsという結果が得られており、GeForce256にはまだまだ余裕があることがわかる。フルに性能を発揮させるためには、もっと高速なCPUを用意する必要があるということだ。

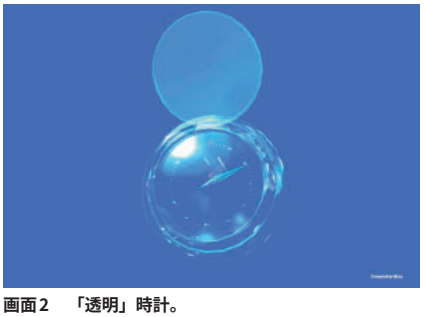
●glclock

glclockは、さまざまな設定で3Dの懐中時計を表示させられるアプリケーションだ。配布されているtarボールには「benchclock」というスクリプトが含まれており、条件を変えながらfpsを算出することで、3D描画能力の評価に用いることができる。Quake III Arenaと同様に、ハードウェアで3D描画が行えるカードに限って測定をしたが、ソフトウェアで描画した場合の例として、Matrox G400を3.3.6で動作させた環境でも測定を行った。ソフトウェアでの描画では、カードの種類によらずほぼ同等の値が出ることは確認済みである。

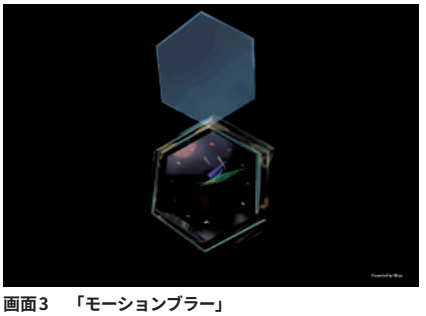
benchclockを実行すると、9通りの条件をダブルバッファ／シングルバッファの両方で測定し、合計18通りの値が得られる。ダブルバッファとは、フレームバッファ（画像データを保持するメモリ領域）を2つ用意し、非表示のフレームバッファで描画作業を行ってから表示するフレームバッファに切り替えることで、表示のちらつきをなくす仕組みである。すべてを掲載するスペースはないので、興味深い結果をいくつか選んで紹介する。

グラフ4の「透明」は、クリスタルのような時計（**画面2**）を表示するもの、「モーションブラー」（**画面3**）は、ボリゴン数を減少（時計が六角形！）させるかわりに、少しずつずらした画像を作って平均化することで、動く物体の「ぶれ」を表現したものだ。比較的重い処理だが、GeForce256はここでもほかを圧倒する高性能を見せつけた。

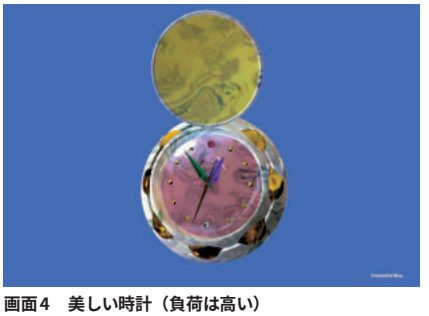
Voodoo3は、「透明」の項目で75fpsとなっているが、もっと負荷が小さいテストでもほとんど75fpsという結果が出ている。同じ設定の時計をシングルバッファで表示させると、より高い値が得られることから、これはおそらく3dfx社のソフトウェアの仕様だと推定できる。つまり、ダブルバッファで表示する場合には、画面のリフレッシュ



画面2　「透明」時計。

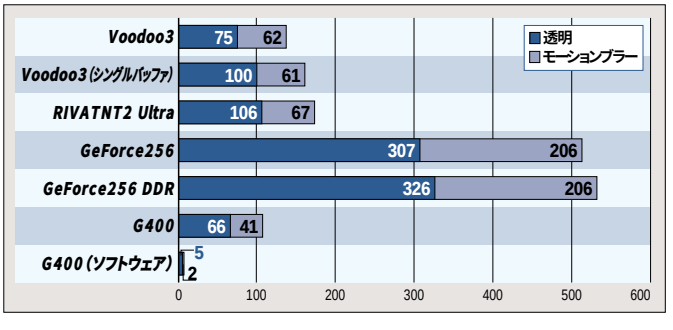


画面3　「モーションブラー」

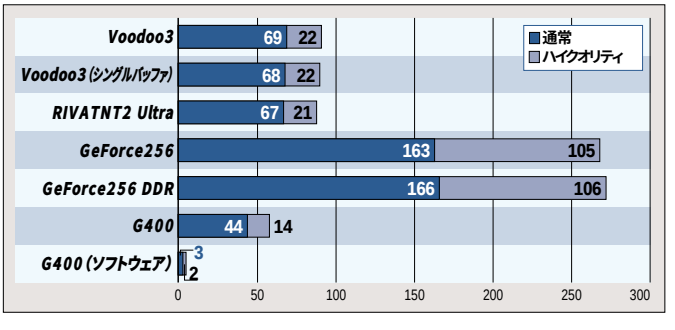


画面4　美しい時計（負荷は高い）

グラフ4　glclockその1（fps）



グラフ5　glclockその2（fps）



レート（75Hz）以上には表示を行わないということだ。3Dゲームなどでは、画面表示以外にもCPUパワーが必要なので、このような仕様はリーズナブルである。

グラフ5の「通常」は、多くの処理を重ねて行う、美しい時計（ただし負荷は高い）を表示するもの、「ハイクオリティ」は、さらにポリゴン数を「通常」の4倍に増やしているものだ（**画面4**）。両者を比較すると「ハイクオリティ」にすることで、GeForce256以外のカードがどれも3分の1くらいまでfpsが低下するのに対して、GeForce256では3分の2程度の低下に収まっていることがわかる。これはGeForce256が3D描画を行う際の座標計算（ジオメトリ演算）をハードウェアで実行できるためである。RIVA TNTシリーズと共通のドライバを用いているが、チップの種類を判別してジオメトリ演算を行っているのであろう。

以上のように、2Dではほとんど差がつかなかったが、3Dの場合はGeForce256がほかを圧倒する高性能を示している。

グラフィックスカードの選び方

LinuxでXFree86を利用するにあたって、どのグラフィックスカードを選べばよいのか、今回のテスト結果を踏まえてまとめてみよう。

グラフィックスカードを選ぶ際には、まず次の2点を確認したい。

1. **XFree86で利用できること**
2. **必要とする性能を有していること**

1.についてはいうまでもないが、XFree86が動作しなければお話にならない。本特集や、Webなどの情報をもとに対応機種を確認しよう。

また、初心者は設定に手間のかかる製品を避けたほうがよいだろう。今回取り上げた7＋1機種のうち、RAGE 128 PROはXFree86 3.3.6では動作せず、X環境を使うためにはXFree86 4.0をインストールする必要があった。現状ではXFree86 4.0を標準構成としているディストリビューションはないため、インストールは簡単とはいえない。もちろん、多少の苦労はいとわないという方や、動かなければドライバをハックしてしまおうという方はこの限りではない。

次に、2.の描画性能だが、自分が求めるスペックに合わせてカードを選ぶ必要がある。たとえば、3D描画を利用することがなければ、高価な3Dアクセラレーション機能付きカードを選ぶ必要はないのだ。

最近のグラフィックスカードを使う限り、2Dの描画性能はまったく問題ないといえる。今回テストした機種の中では、i810の2D描画性能がいくぶん劣るが、実際には体感できるほどの差はなかった。i810で問題になるのは、あとからマシンの拡張をしたくなっても外付けのグラフィックスカードを使うことができないという点かもしれない。一方、3Dの描画性能を見ると、カードごとの差は非常に大きくなる。3D描画を行うアプリケーションプログラムを利用するなら、ハードウェアによる3Dアクセラレーションは必須といえよう。ハードウェアの性能もさることながら、ドライバなど、ソフトウェアの対応状況も重要だ。

どのカードを選ぶべきか

自分の求めるスペックが決まったら、あとは財布と相談するだけだ。

写真1　NVIDIA GeForce256 DDR
現在、Linuxで最強のグラフィック環境を提供する。しかし、ライバル達がその座を虎視眈々と狙っている。

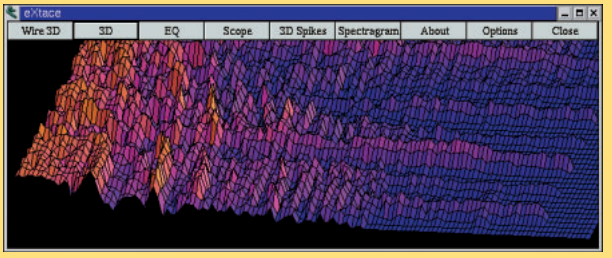
2Dのみで利用するということであれば、比較的新しいカードからXFree86で利用できるものを選べばまず問題はない。数年前では、高解像度ではリフレッシュレートを上げられず、チラチラした画面になるグラフィックスカードもあったが、現在のグラフィックスチップに統合されたRAMDACは十分な性能を持つので、その心配は無用だ。i810を搭載したベアボーンキットなどのコストパフォーマンスも大きな魅力だ。用途に合わせ、賢い選択をしたい。

Linux上で3Dゲームを快適にしたいなら、現状ではGeForce256以外の選択肢はありえない。ただし、これはあくまで「現状」での話だ。グラフィックスチップの世代交代は非常に速く、まもなくVoodoo4／5（チップ名：VSA-100）、ATI Radeon256など強力な新製品が各メーカーから登場する。特にVSA-100は、XFree86 4.0のDRI対応ドライバも開発中であり、その性能には期待できる。NVIDIAも最新チップGeForce2 GTSでさらに性能を上げてきている。

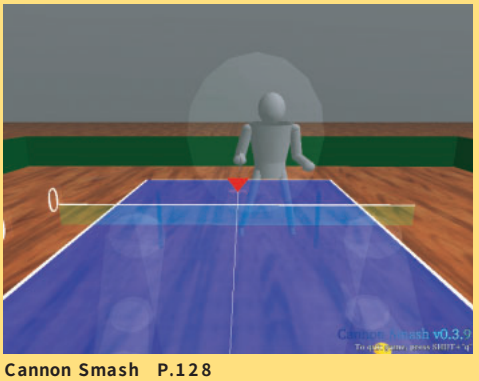
XFree86 4.0への移行、ドライバモジュールの開発状況などからも目が離せない。最強のグラフィックス環境を構築するなら、常に最新の情報をチェックしておこう。Linuxのグラフィックス環境は、これから急速な進化を見せることになりそうだ。

Free Application Showcase

文：出井 一
Text : Hajime Dei



eXtrace P.131



Cannon Smash P.128

Sylpheed P.117

※紹介したソフトは、すべて付録CD-ROMに収録されています。

安定して使いやすいメールクライアント

Sylpheed

バージョン：0.3.8　　ライセンス：GPL

http://www.kcn.ne.jp/~hiro-y/
http://www.linuxstart.com/~sunnyone/mylinux/sylpheed.html（RPM）

Sylpheedは日本発の軽快で使いやすいメールクライアント兼ニュースリーダだ。Outlook Expressに代表されるWindows系メールクライアントと同様の外見と、MewやWanderlustといったEmacs系メールクライアントの操作感を併せ持つ。複数アカウントへの対応をはじめ、スレッド表示、振り分け、マルチパートMIME、クリッカブルURLなど、機能も充実している。動作にはGTK+ 1.2.6以降が必要だ。

●ビルドとインストール

Sylpheedは、ソース一式をtar＋bzip2したtarボールのほか、RPMのパッケージが別サイトで配布されている（上記URLを参照）。

tarボールはbzip2により圧縮されているので、「bzip2 -dc sylpheed-0.3.8.tar.bz2 | tar xvf -」として展開する。ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。

一方、RPMはソースとバイナリのパッケージがそれぞれ用意されている。ただし、バイナリパッケージはVine 2.0用なので、他のディストリビューションではソースパッケージからリビルドしたほうがいいだろう。

「rpm --rebuild sylpheed-0.3.8-1.src.rpm」とすると、ソースパッケージからバイナリパッケージが作成されるので、「rpm -Uvh /usr/src/redhat/RPMS/i386/sylpheed-0.3.8-1.i386.rpm」としてインストールする。

●画面構成とアカウントの作成

「sylpheed&」として起動するとメインウィンドウが開く。画面構成はOutlook ExpressなどのWindows系メールクライアントでおなじみの3ペイン方式だ。[受信箱]などのフォルダが並ぶ「フォルダツリー」、メールの送信者やサブジェクト（件名）などが一覧表示される「サマリビュー」、メールの本文が表示される「メッセージビュー」で構成される（**画面1**）。

まずは、アカウントの登録を行おう。ツールバーの[アカウント]ボタンを押し、「アカウントの設定」ダイアログの[基本]ページで名前やメールアドレス、POP／SMTPサーバ名などを設定する（**画面2**）。他の設定は後で行えばいい。

登録したアカウントは「アカウントの編集」ダイアログに一覧表示される。Sylpheedでは1人のユーザーが複数のアカウントを扱えるため、このダイアログから別のアカウントを追加したり、既存のアカウントを編集・削除するといった操作が可能だ。

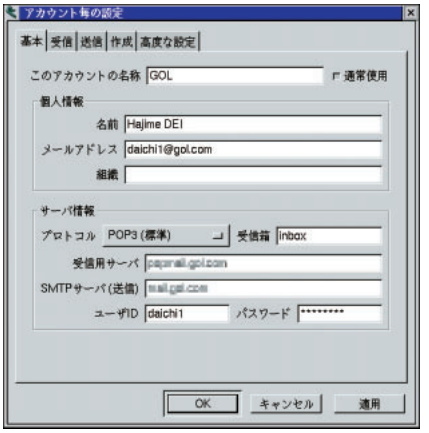
なお、メインウィンドウで現在選択しているアカウント名は、ウィンドウの右下に表示される。別のアカウントに切り替えたい場合は、メニューの[設定]－[現在のアカウントを変更]以下の項目から、目的のアカウント名を選択すればいい。

●メールの受信と閲覧

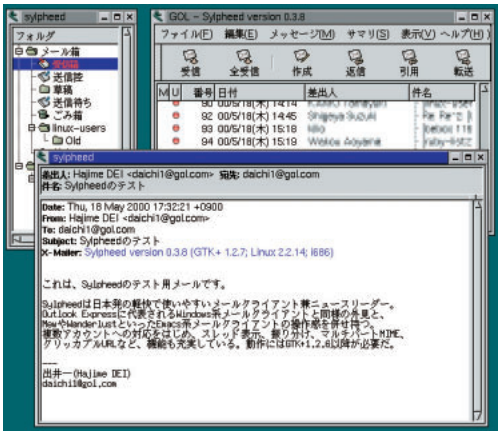
ツールバーの[受信]ボタンを押すと、現在のアカウントのメールサーバに接続して新着メールを受信する。登録したすべてのアカウントの新着メールを一度に受信する[全受信]ボタンも用意されている。このほか、ローカルなメールボックスの内容をインポートすることも可能だ。

受信したメールは[受信箱]フォルダに格納される。[受信箱]フォルダを選択すると、メールのタイトルや日付、送信者などがサマリビューにスレッド表示され、サマリビューで選択したメールの本文がメッセージビューに表示される。なお、フォルダツリーやメッ

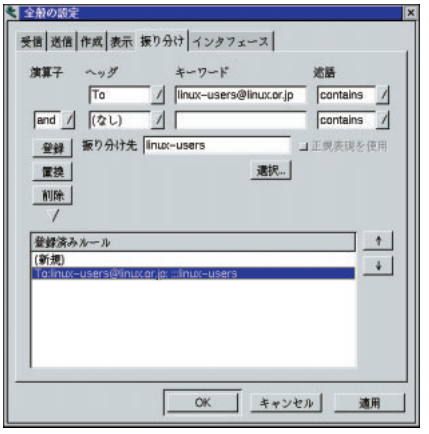
画面1
Outlook Expressなどと同じ3ペイン構成のウィンドウ。



画面2
アカウントの設定はこのダイアログで。複数アカウント対応だ。



画面3
フォルダツリーやメッセージビューをウィンドウ表示した状態。



画面4
メールをフォルダに振り分けるためのルールを設定する。

セージビューを別ウィンドウとして表示する設定も可能だ（**画面3**）。

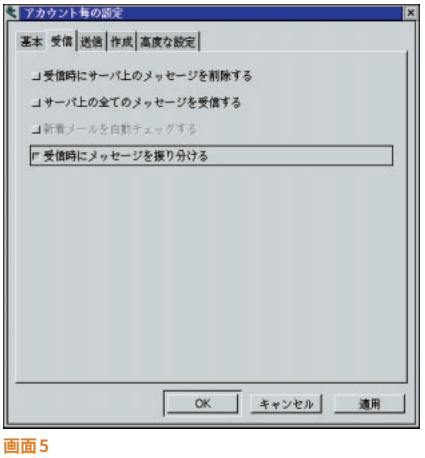
未読メールを閲覧するには、スペースキーを押せばいい。次の未読メールへの切り替え、画面に収まりきらない本文のスクロール、次のフォルダの切り替えなどをインテリジェントに処理してくれる。また、Emacs系メールクライアントの挙動をエミュレートする機能も用意されている。

●フォルダの作成と振り分け

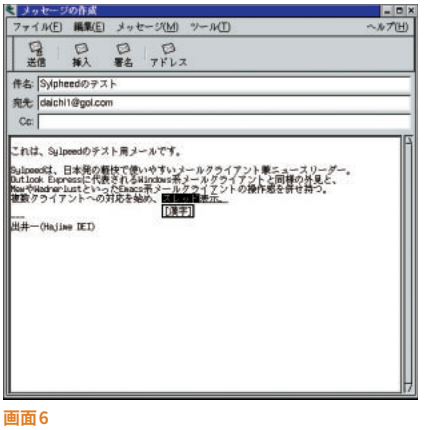
流量の多いメーリングリストに加入している場合は、振り分け機能を利用してメールをフォルダに分類しよう。まずは、整理用のフォルダ（階層化可能）を作成する。フォルダツリー上での右クリックメニューから[新規フォルダを作成]を選択し、フォルダ名を入力すればいい。

続いて、ツールバーの[設定]ボタンで設定ダイアログを開き、[振り分け]ページに切り替える（**画面4**）。ここでは、振り分けに利用するヘッダ名（Subject、From、Toなど）とその内容（キーワード）、移動先フォルダを「ルール」として登録する。複数のルールを登録した場合は、上位のルールから順番に適用される。

メール受信後に、[サマリ]—[メッセージを振り分ける]を選択すると、これらのルールに基づいて振り分けが行われる。なお、メールの受信と同時に自動的に振り分けるには、「アカウントの設定」ダイアログの[受信]ページで、[受信時にメッセージを振り分ける]をチェックしておく必要がある（**画面5**）。



画面5
受信したメールを自動振り分けするにはここで設定を変更。



画面6
メッセージ作成ダイアログで送信するメールを書く。

●メールの作成

メールを新規作成する場合は、ツールバーの[作成]ボタンを押す。また、メールの返事を書くには[返信]ボタン、相手の発言を引用して返事を書くには[引用]ボタン、相手とは別の人にメールの内容を転送するには[転送]ボタンをそれぞれ押せばいい。

いずれにせよ、メッセージ作成ウィンドウが開くので（**画面6**）、件名（Subject）や宛先（To）、本文を入力しよう。返信などの場合は、件名や宛先はすでに入力済みだ。本文には自動的に署名（シグネチャ）が挿入される。

なお、マルチパートMIMEによるファイル添付は、現時点では読み込みのみ対応している。指定したファイルの内容をメール本文に挿入する機能が用意されているので、メールにファイルを添付する場合は、自分でファイルをエンコードして挿入しよう。

●アドレス帳はXMLで管理

メインウィンドウのメニューで[ツール]—[アドレス帳]を選択するか、メッセージ作成ウィンドウのツールバーの[アドレス]ボタンを押すと、メールアドレスを管理するアドレス帳ウィンドウが開く（**画面7**）。

画面7
XMLで管理されるアドレス帳。メールアドレスの検索も可能だ。

画面8
設定内容は名前とメールアドレス、備考の3種類だけ。

アドレス帳に登録するには、このウィンドウで[登録]ボタンを押し、メールアドレスと名前、備考を入力すればいい（**画面8**）。このほか、分類用の仮想フォルダを作成したり、名前を指定してメールアドレスを検索する機能も用意されている。

メッセージ作成ウィンドウからアドレス帳を開いた場合は、[宛先:]や[Cc:]ボタンが有効になる。一覧中のメールアドレスを選択してからこれらのボタンを押すと、メッセージ作成ウィンドウの宛先やCcにそのメールアドレスが設定される。

なお、アドレス帳の内容はXMLを使って記述されており、~/.sylpheed/addressbook.xmlに保存されている。メールアドレスを大量に登録しなければならない場合には、このファイルをテキストエディタやスクリプト言語で直接変更するとよいだろう。

●ニュースリーダ機能を使う

Sylpheedには、ネットニュースを購読するニュースリーダとしての機能も用意されている。メールを読むのと同じ感覚でニュースグループの記事を閲覧可能だ。ただし、現時点では記事を投稿できない。

ニュースリーダとして使うには、利用するニュースサーバやニュースグループの名前をSylpheedに登録する必要がある。

まずは、フォルダツリーの[NetNews]フォルダ上で右クリックメニューを開き、[ニュースサーバを追加]を選択してサーバ名を入力しよう。[NetNews]フォルダの下に、サーバ名のフォルダが作成される。

続いて、そのフォルダ上で右クリックメニューを開き、[ニュースグループを購読]を選択して、グループ名（たとえば「fj.os.linux」など）を入力する。これを、購読したい各グループに対して行えば準備完了だ。

ニュースグループの記事を取得するには、そのグループ名のフォルダをクリックするだけでいい。自動的にニュースサーバに接続して、新しいニュースを取得する。

サマリビューには、ニュースの件名や投稿者の一覧がスレッド表示され、メッセージビューには選択したニュースの本文が表示される（**画面9**）。あとは、メールを読む場合と同じようにして閲覧すればいい。もちろん、スペースキーだけを使って未読のニュースを読み進められる。

なお、フォルダツリーのペインを拡大すると、ニュースの新着数・未読数・総数が表示される。ニュースグループ名は長くなることが多いので、メニューの[表示]－[フォルダツリーを分離]を選択してフォルダツリーを独立させ、横幅を広げると見やすいだろう（**画面10**）。上記のメニューを再度選択すれば、ウィンドウが消えて元のペイン表示に戻る。

画面9
ネットニュースをメールと同じ感覚で読み進められる。

画面10
フォルダツリーをウィンドウ表示にして横幅を拡大する。

日本語に対応した使いやすいHTMLエディタ

Bluefish

バージョン：0.4　　ライセンス：GPL

http://bluefish.openoffice.nl/
http://www.weblint.org/（Weblint）
http://www.sfc.keio.ac.jp/~mimasa/jweblint/index.html.en（jweblint）

Bluefishは、カラー構文表示に対応したタグ挿入型のHTMLエディタだ。ツールバーのボタンを使ってタグを効率よく入力できる。日本語カタログによるメニューの日本語化や、日本語の入力・編集にも対応している（日本語EUCのみ）。また、作成したWebページをWeblintでチェックしたり、実際の表示をNetscapeで確認できるなど、外部ソフトとの連携も考えられている。動作にはGTK＋が必要だ。

●ビルドとインストール

BluefishはtarボールのほかRPM／DEBなどのパッケージでも配布されている。ただし、RPMはi586／686用のバイナリパッケージしか用意されていないので、tarボールを利用したほうがいいだろう。ビルドとインストールは「./configure」→「make」→「make install」という一般的な手順だ。

このバージョンからは日本語カタログ（ja.po）が標準添付され、日本語環境ではメニューやダイアログが日本語表示されるようになった。また、ロケール設定やフォントセットにも対応しているため、ソースを修正することなく日本語を入力できる。

●Weblintのインストール

続いて、Webページの文法チェックを行う「Weblint」の導入を行う。日本語を使ったWebページのチェックを行う場合は、オリジナルに加えて日本語化パッチ（jweblint）も必要だ。いずれもtarボールでソースが配布されているので、同じディレクトリに展開しよう。

まず、Red Hat系ディストリビューションの場合は、Perlをパッケージでインストールしているはずなので、ライブラリのインストール先を変更する必要がある。具体的には、Makefile.jaの11行目のLIBDIRの設定を、「/usr/local/lib/perl」から/usr/lib以下のディレクトリ（/usr/lib/perl5/など）に変更する。

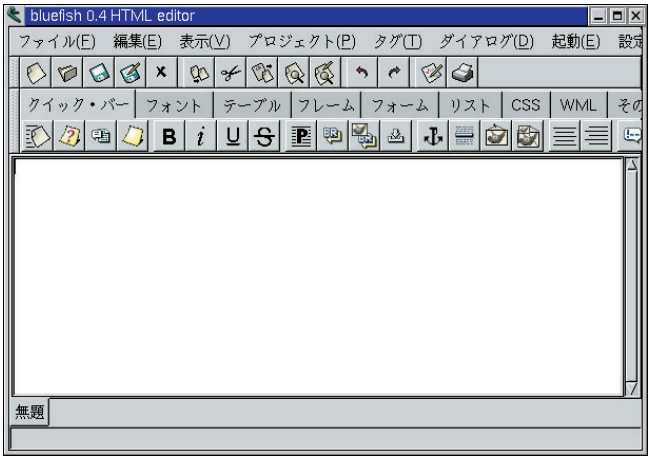
変更が終わったら、「make -f Makfile.ja」→「make install」とすると、パッチ当てなどが行われ、/usr/local/binにweblintとjweblintがインストールされる。

●Bluefishの起動と画面構成

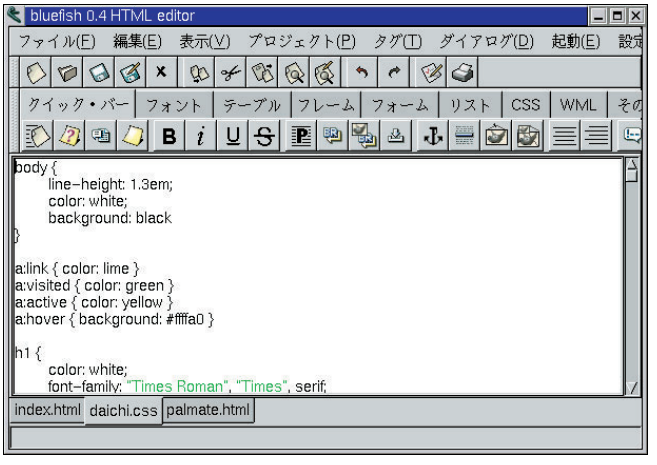
「bluefish&」として起動すると、スプラッシュが表示された後、ツールバーが並んだウィンドウが開く（**画面1**）。日本語環境では、メニューなどが日本語で表示されるはずだ。

ツールバーは2段構成で、上段がファイルオープンや保存、カット＆ペーストなどの編集用、下段がタグ入力用となっている。タグ入力用のツールバーはジャンル別にページ分けされており、目的のタグを素早く見つけられる。各ボタンの機能は、マウスポインタを置くと表示されるチップヘルプ（日本語）で調べられる。

その下の編集領域には、HTMLファイルをはじめ、CSSスタイルシートや通常のテキストなどを記述できる。複数ファイルを編集している場合は、ウィンドウ下のタブで切り替え可能だ（**画面2**）。なお、Ctrl－Oキーはオープンダイアログに割り当てられているため、日本語入力のオン／オフにはShift－スペースキーを利用する。



画面1
日本語カタログによってメニューなどが日本語で表示される。



画面2
複数のファイルを同時にオープンし、タブで切り替え可能だ。

画面3
HTMLファイルを新規作成する場合に便利な「クイック・スタート」

画面4
イメージタグの設定ダイアログでは画像をプレビューできる。

画面5
スタイルシートの設定はこのダイアログで行う。

●HTMLファイルの入力

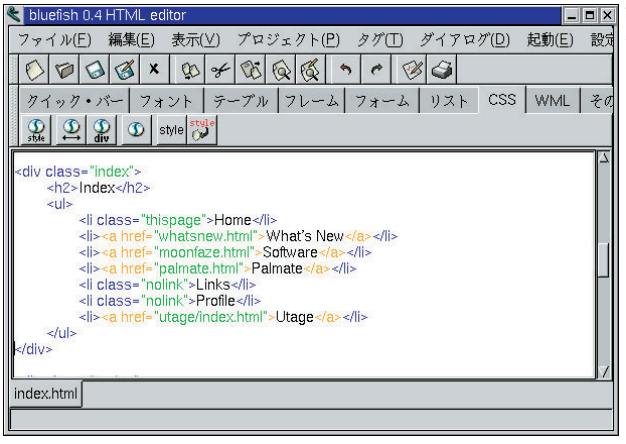
使い方は一般的なタグ挿入型のHTMLエディタと同じで、ツールバーのボタンを押してカーソル位置にタグを挿入し、要素をキーボードから入力する。もちろん、タグをキーボードから入力してもいい。

なお、HTMLファイルを新規作成する場合、まずはツールバー下段の[クイックバー]ページ左端にある[クイックスタート]ボタンを押そう。DTD（文書型定義）やタイトルなど、基本的な設定をダイアログで行い、まとめてタグと要素を挿入できる（**画面3**）。

空要素タグ（
など）の場合は、ボタンを押すだけでタグが挿入される。一方、要素の前後を挟むタイプのタグ（<P>など）では、ボタンを押してから要素を入力してもいいし、先に入力した要素を選択した状態にしてボタンを押してもいい。

なお、<A>や<IMG>のように属性の指定が不可欠のタグでは、それぞれ専用のダイアログが開いて属性値を設定する（**画面4**）。[OK]ボタンを押すと、指定した属性と属性値を持ったタグがカーソル位置に挿入される。

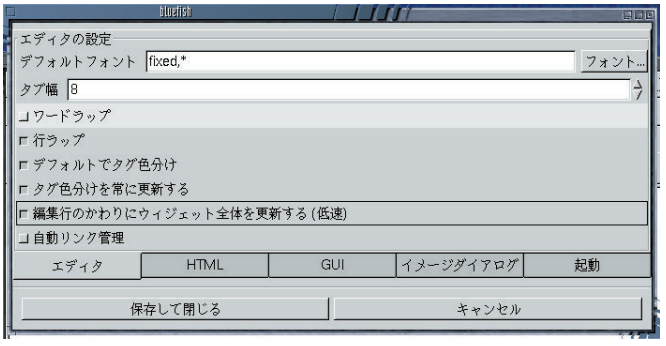
スタイルシートの設定では、[CSS]ページの左端のボタンでジャンル別にページ分けされたダイアログが開き（**画面5**）、すべてのスタイルを設定できる。HTMLファイルのタグへのクラスの追加や、要素を<DIV>や<SPAN>で囲む作業もボタンを押すだけだ。



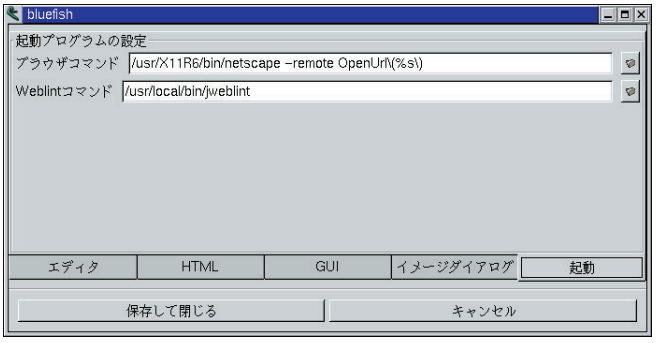
画面6
タグや属性値などを色分け表示するカラー構文表示。

●日本語の色分けには問題あり

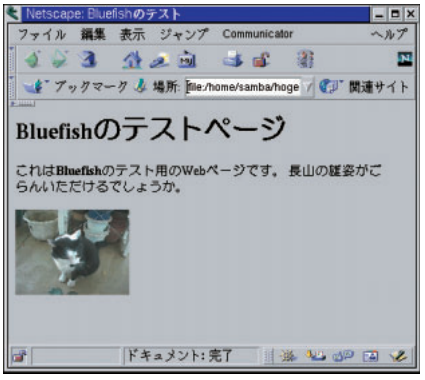
Bluefishの特徴のひとつに、タグや属性、属性値などをカラー表示する色分け（カラー構文表示）機能がある（**画面6**）。なお、初期設定ではカラー表示が自動更新されないので、適当な



画面7
色分け表示に関する設定はこのダイアログで行う。



画面8
NetscapeとWeblintのコマンドの設定は必ず行うこと。



画面9
Netscape Navigatorで編集中のページの実際の見栄えを確認。

タイミングでF5キーを押して手動で更新する必要がある。

編集行のカラー表示を自動的に更新させることも可能だ。ツールバー上段右から2番目の[詳細設定]ボタンを押して設定ダイアログ（**画面7**）を開き、[タグ色分けを常に更新する]をチェックすればいい。さらに、[編集行のかわりにウィジェット全体を更新する（低速）]もチェックすると、編集領域全体が自動更新されるようになる（処理はそれだけ遅くなるが）。

残念ながら、現時点では日本語に関する色分け処理にバグがあり、日本語を含むHTMLファイルは正常に色分けされない。日本語を含むHTMLファイルを編集する場合は、メニューの[表示]－[タグの色分け]のチェックを外しておく必要がある。また、常に色分けを行わない状態にするには、設定ダイアログの[デフォルトでタグ色分け]を外せばいい。

●Netscapeで実際の表示を確認

Bluefishでは、編集中のHTMLファイルの実際の表示を、Netscape Navigator（以下Netscape）で確認できる。

まずは、設定ダイアログの[起動]ページで、Netscapeの設定を行う（**画面8**）。[ブラウザコマンド]の「##WHERE_IS_NETSCAPE ##」を実際のパス（/usr/X11R6/bin/netscapeなど）に変更し、「OpenUrl\\(%s\\))」の最後の「)」を削除する。

Netscapeの起動は、ツールバー下段[その他]ページ左から4番目の[Netscapeで確認]ボタンで行う。自動的にファイルの保存が行われた後、Netscapeにより編集中のWebページが表示される（**画面9**）。すでにNetscapeが起動している場合は、それを利用してWebページの表示が行われる。

●Weblintによる構文チェック

WeblintでHTMLファイルの構文チェックを行うには、設定ダイアログの[起動]ページの[Weblintコマンド]を「/usr/local/bin/weblint」から、日本語対応版であるjweblintのフルパス（通常は「/usr/lcoal/bin/jweblint」になる）に変更する。

jweblintの起動は、メニューの[起動]－[Weblint]で行う。子プロセスでjweblintが実行され、HTMLファイルのチェック結果が別ウィンドウに表示される（**画面10**）。

チェック結果をダブルクリックすると編集領域の対応行が反転する。内容を修正したら、[再チェック]ボタンで再度チェックしよう。

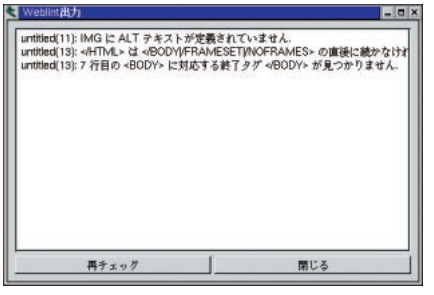
●プロジェクトの作成

一般に、Webサイトは複数のHTMLファイルで構成されている。Bluefishには、こうした複数のファイルを「プロジェクト」としてまとめて扱う機能が用意されている。

プロジェクトを新規作成するには、[プロジェクト]－[編集]を選択してダイアログを開き（**画面11**）、ベースディレクトリやWebディレクトリなどを設定する。

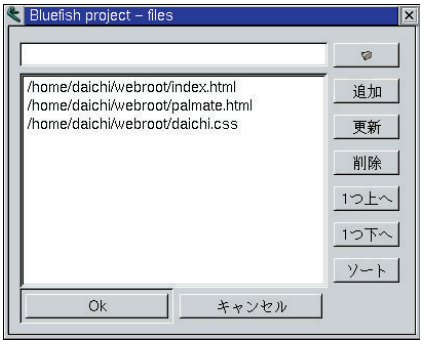
編集中のファイルをプロジェクトにファイルを追加するには、[プロジェクト]－[現在編集中の文書を追加]を選択する。プロジェクトへのファイルの追加や削除といった処理は、このダイアログの[ファイル]－[ファイル]で開くダイアログで行う（**画面12**）。また、[リスト]以下のメニューで、各ファイルで共用する色やURL、CSSなどのリストを作成可能だ。

作成したプロジェクトは、[プロジェクト]－[名前をつけて保存]で保存しておけば、次回からは[プロジェクト]－[開く]で読み込める。



画面10
Weblint（jweblint）によるHTMLファイルの構文チェックも可能だ。

画面11
複数のファイルをプロジェクトしてまとめて扱える。



画面12
プロジェクトへのファイルの追加・削除はここで行う。

WindowsからLinuxのパーティションを読み書きする

Explore2fs

バージョン：1.00 pre3　　ライセンス：フリー

http://uranus.it.swin.edu.au/~jn/linux/explore2fs.htm

Explorer2fsは、Windows 9x／NTでLinuxのext2ファイルシステムを読み書きできるソフト。ルック＆フィールはエクスプローラによく似ており、パーティションごとのディレクトリツリーとファイル表示領域で構成される。ドラッグ＆ドロップや右クリックメニューでファイルを自由に閲覧／コピー／移動／削除可能だ。WindowsとLinuxをデュアルブートしているマシンで導入すると便利なソフトだ。

●インストール

Explorer2fsは、ソースとバイナリのZIPファイルがそれぞれ配布されている。通常は、実行ファイルやDLLファイルなどが含まれるバイナリ版を取得すればいい。インストーラは付属しないので、適当なフォルダで展開し、実行ファイルのショートカットを作成しておこう。

●使い方はエクスプローラと同じ

起動すると2ペイン構成のウィンドウが開く（**画面1**）。左側にはLinuxのext2ファイルシステムのパーティションのディレクトリ構造がツリー表示され、右側には選択したディレクトリのファイル一覧が表示される。ファイルの表示は大小アイコン／一覧／詳細を切り替えられる。

ファイルに対する操作は、エクスプローラと同様にドラッグ＆ドロップや右クリックメニューを使用する。たとえば、Windowsのパーティションにファイルをコピーするには、エクスプローラなどにファイルをドラッグ＆ドロップすればいい。

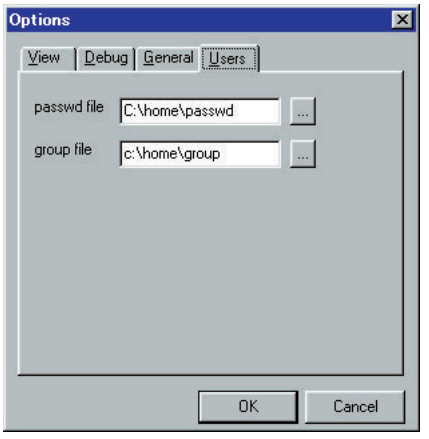
また、テキストの内容を閲覧するには、右クリックメニューから[View]を選択する。日本語EUCを自動判別するビューアを用意すれば、日本語テキストの閲覧も可能だ。

●ユーザー名の表示も可能

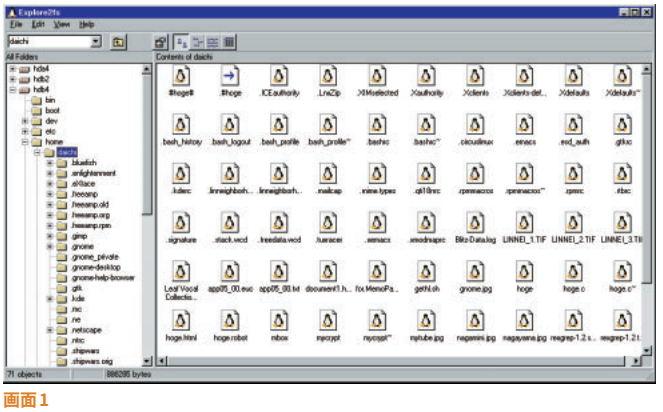
初期設定では、詳細表示のユーザーIDとグループIDが数値のまま表示される。これをユーザー名やグループ名に変更するには、「/etc/passwd」（シャドウ化済みのもの）と「/etc/group」をWindowsの適当なフォルダにコピーし、オプションダイアログ（**画面2**）の[Users]ページで指定すればいい。それらのファイルの情報に基づいて、ユーザーIDとグループIDが対応する名前で表示されるようになる（**画面3**）。

Explorer2fsは、Linux上で設定されているファイルのアクセス権限に関係なく、全ファイルを読み書きできる強力なツールだ。起動時スクリプトなどの間違いを修正したり、Ext2ファイルシステム内のデータファイルを再起動することなくWindowsから利用するといった便利な使い方ができる。

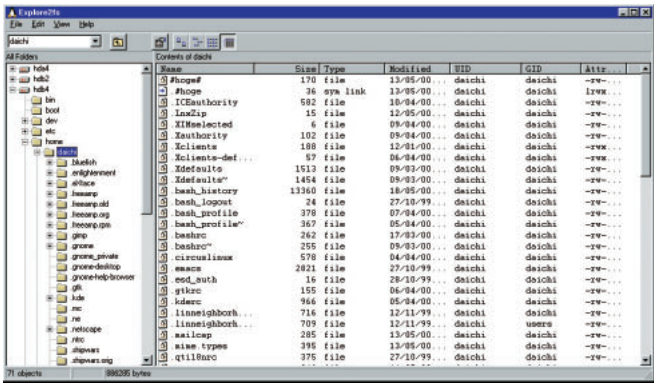
だがその反面、大きなセキュリティホールにもなりかねないので、他人が直接パソコンに触れられる環境での運用には注意が必要だ。またLinuxの起動に不可欠なファイルを消してしまうことも可能なので、注意しよう。



画面2
オプションダイアログでユーザー名とグループ名のリストを設定。



画面1
ex2fsファイルシステムのパーティションの内容が表示される。



画面3
詳細表示のユーザーIDとグループIDが名前で表示される。

定番のプレゼンテーションソフトがPNG対応に

MagicPoint

バージョン：1.07a　　ライセンス：フリー

http://www.mew.org/mgp/
http://puchol.com/cpg/software/mgp/（RPM）

MagicPointは、スライドやOHPの代わりにパソコンのディスプレイや液晶プロジェクタでプレゼンテーションを行うためソフト。WindowsのPowerPointとは異なり、テキスト形式のmgpファイルから各ページの画面を生成する。TrueTypeやType1フォントによる高品質な文字に加え、背景にグラデーションを使ったり、各種形式の画像を表示することも可能だ。最新版ではPNG形式にも対応した。

●ビルドとインストール

MagicPointは、tarボールのほかにRPMパッケージも配布されている。ただし、バイナリパッケージはVFlibを無効にした状態でビルドされているので、日本語フォントの扱いに難がある。RPMを利用する場合は、以下のようにして、ソースパッケージからリビルドしたほうがいいだろう。

「rpm –rebuild mgp-1.07a-1.src.rpm」とすると、ソースパッケージからバイナリパッケージが作成されるので、「rpm -Uvh /usr/src/redhat/RPMS/i386/mgp-1.07a-1.i386.rpm」としてインストールしよう。

一方、tarボールからのビルドとインストールは、「./configure」→「xmkmf」→「make Makefiles」→「make」→「make install」→「make install.man」という手順になる。

なお、RPMパッケージでは日本語ドキュメント（USAGE.jpなど）がインストールされない。必要ならばtarボールから取得しよう。

●サンプルで操作方法を学ぶ

まずは、付属のチュートリアルで基本的な操作方法を学んでみよう。tarボールなら展開先、RPMパッケージの場合は/usr/doc/mgp-1.07a以下のsampleディレクトリに、MagicPoint用のデータファイル（mgpファイル）のサンプルが用意されている。

たとえば、日本語のチュートリアル（tutorial-jp.mgp）を表示するには、ktermなどのコマンドラインで「mgp tutorial-jp.mgp」とすればいい。通常は、各ページの内容がフルスクリーン表示される（**画面1**）。－gオプションでサイズを指定してウィンドウ表示にすることも可能だ（**画面2**）。

画面操作にはキーボードかマウスを使用する。スペースキーや↓キー、マウスの左ボタンなどで次のページに進み、Delキーや↑キー、マウスの右ボタンで前のページに戻る。MagicPointを終了するには、EscキーかQキーを押せばいい。

さらに、「ページ番号」＋Gキーで指定したページに直接飛んだり、Ctrlキーで左下に表示されるページ一覧をマウスでクリックしてページを切り替えることも可能だ（**画面3**）。また、Shift－Gキーで画面の下に前後のページのタイトルや現在のページ番号がガイドライン表示される。

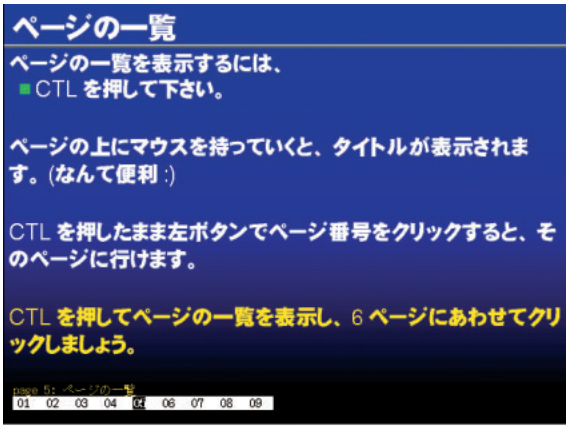
ちょっと変わった機能として「落書きモード」がある。これは、Xキーを押すとマウスポインタがペンの形に変わり、マウスのドラッグ操作でページ上に描画できるというもの（画面4）。プレゼンテーション中に内容を補足したり、アンダーラインを引くのに使えるだろう。Shift－Xキーで色を変更することも可能だ。なお、描いた内容は、ページを切り替えるか、Lキーを押すと消去される。落書きモードから抜けるには再度Xキーを押せばいい。

●mgpファイルを記述する

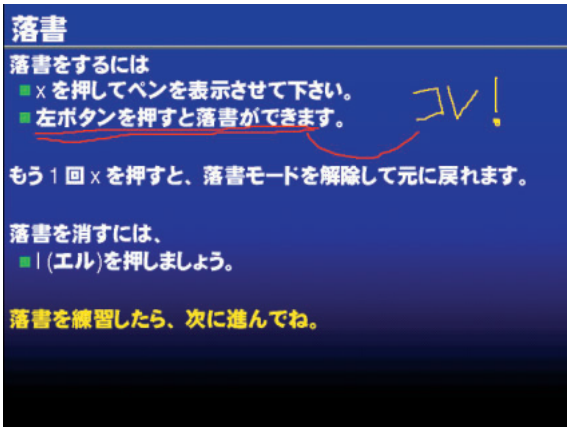
続いては、自分でmgpファイルを作

画面1
通常はフルスクリーンでページの内容が表示される。

画面2
起動時オプションによりウィンドウに表示することも可能だ。



画面3
Ctrlキーでページ一覧を表示し、マウスのクリックで切り替え。



画面4
表示中のページに自由に書き込みを行える「落書き」機能。

成してみよう。テキスト形式なので、ふだん使っているEmacsなどのエディタで作成可能だ。

mgpファイルは行指向のデータ形式を採用しており、先頭が「%」で始まる行はコマンドと解釈される。文字の大きさや色などを変えるコマンドなどが用意されており（**リスト1**）、複数のコマンドを「,」で区切って1行に並べることもできる。

まず、mgpファイルの先頭には「%include default.mgp」と書こう。これは、各ページのスタイルの初期設定（/usr/X11R6/lib/X11/mgp/default.mgp）を読み込むコマンドだ。

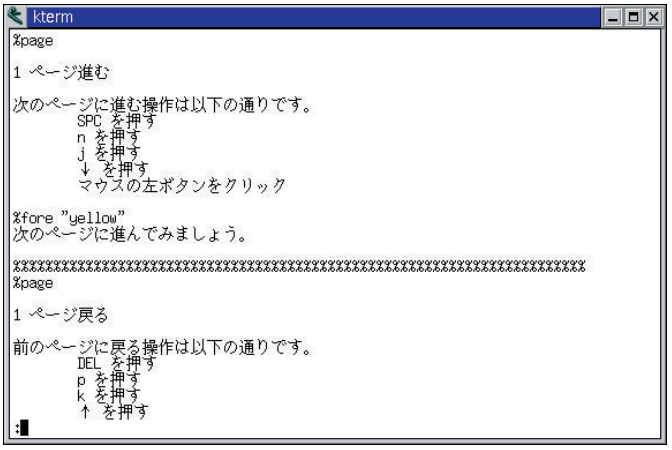
続いて、ページごとの内容を次のような構造で記述する。まず、ページの開始を「%page」というコマンドで宣言し、空行、タイトル、空行、本文と続く（**画面5**）。

コマンドの内容は、通常はそれ以降の行で有効になる。例外として、行番号を指定して属性を設定する「default」と、タブで字下げした段数を指定して属性を設定する「tab」がある（通常はdefault.mgpで指定された内容がそのまま使われる）。

●画像データの埋め込みも可能

MagicPointには、画像データをmgpファイルに埋め込む「mgpembed」、mgpファイルをPostScript形式に変換する「mgp2ps」、HTTPサーバとして動作し、ネットワーク上でプレゼンテーションできる「mgpnet」などのツールが付属している。各ツールについての詳細は、manコマンドで表示されるマニュアルを参照されたい。

mgpembedは、「mgpenbed 元ファイル名 > 埋め込み先ファイル名」として使う。通常は画像ファイルを外部から読み込むが、mgpembedで作成されたmgpファイルには、画像をuuencodeでテキストデータに変換した内容が埋め込まれている。このため、mgpファイルをメールやWebで配布したり、持ち運ぶときに便利だ。



画面5
各ページのソースはこのようにテキストで記述されている。

%size	文字のサイズ
%fore	前景色
%back	背景色
%bgrad	背景グラデーション
%left	左寄せ
%leftfill	左寄せ（長い行）
%center	センタリング
%right	右寄せ
%lcutin	左端からカットイン
%rcutin	右端からカットイン
%cont	改行なしで表示
%deffont	フォント名定義
%font	フォントの指定
%bar	横線表示
%image	画像表示
%prefix	前置文字
%icon	アイテム用アイコン
%bimage	背景画像表示
%default	指定行の属性
%tab	指定タブ数の属性
%include	ファイルの読み込み
%page	ページ開始
%vgap	行間隔
%hgap	文字間隔
%pause	表示の一時停止
%mark	現在位置をマーク
%again	上記マークに戻る
%system	子プロセス実行
%filter	同上（テキスト指定）
%endfilter	テキスト指定の終了
%pcache	ページの先読み

リスト1
mgpファイルで使えるコマンド（SYNTAXより抜粋）

RPMの作成も可能な統合アーカイブソフト

LnxZip

バージョン：0.1　　ライセンス：GPL

http://visionary-hawk.webjump.com/lnxzip.html

LnxZipは、tar／gzip／bzip2／zip／arjなどでアーカイブ・圧縮されたファイルを統一的に扱うフロントエンドだ。見た目はWindows用の統合アーカイブソフト「WinZip」に似ており、アーカイブの形式を気にすることなく、ファイルの追加や削除、閲覧を行える。また、RPMパッケージの作成を支援する機能を持ち、一般ユーザーのままRPMパッケージを作成可能だ。動作にはGNOMEが必要となる。

●ビルドとインストール

LnxZipは、ソース一式をtar＋gzipしたtarボールのほかに、RPMパッケージも配布されている。tarボールからの作成手順は、「./configure」→「make」→「make install」という一般的なものだ。

RPMはソースとバイナリのパッケージがそれぞれ用意されている。ただし、バイナリパッケージはMandrake用なので、他のディストリビューションではソースパッケージからリビルドしたほうがいいだろう。

「rpm --rebuild LnxZip-0.1-1.src.rpm」とすると、ソースパッケージからバイナリパッケージが作成されるので、「rpm -Uvh /usr/src/redhat/RPMS/i386/LnxZip-0.1-1.i386.rpm」としてインストールしよう。

●起動と初期設定

「lnxzip&」として起動すると、ツールバーとファイルリストで構成されたウィンドウが開く。なお、初めて起動したときには、RPM作成関連の設定ファイルやディレクトリを作成するかどうか尋ねられるので、[OK]ボタンで許可しよう。

既存のアーカイブを開く場合は、[Open]ボタンを押してダイアログからアーカイブを選択する。また、[New]ボタンでアーカイブを新規作成することも可能だ。いずれにせよ、ウィンドウにはアーカイブに含まれるファイルが一覧表示される（**画面1**）。

ファイルの展開、追加、削除、閲覧などの操作には、ツールバーのボタンや右クリックメニューを利用する。たとえば、アーカイブ内のファイルを外部に展開するには、[Extract]ボタンを押し、展開先ディレクトリをダイアログで指定する。対象となるファイルを選択（複数可）してからボタンを押せば、一部のファイルだけ展開することも可能だ。

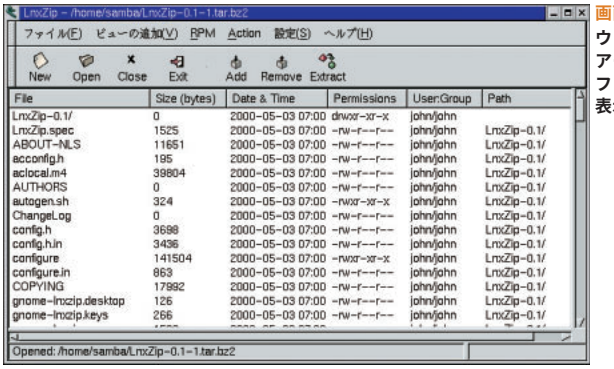
外部のファイルをアーカイブに追加する場合は、[Add]ボタンを押して、対象となるファイルをダイアログで選択する。逆に、アーカイブ内のファイルを削除するには、対象となるファイル（複数可）をマウスで選択してから[Remove]ボタンを押せばいい。

また、ファイルを選択してから右クリックメニューの[View]を選択すると、別ウィンドウにその内容が表示される（**画面2**）。日本語EUCで書かれた文章も表示可能だ。

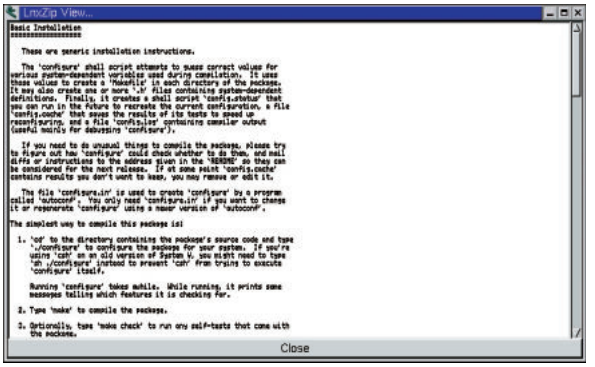
●設定変更とRPM作成の準備

メニューから[設定]－[設定]を選択すると、設定ダイアログが開く（画面3）。[General]ページには、デフォルトのアーカイブタイプ（初期設定は「.tar.bz2」）などが設定されており、[Archiver]ページには、アーカイブや圧縮を処理する各種コマンドのフルパスが設定されている。

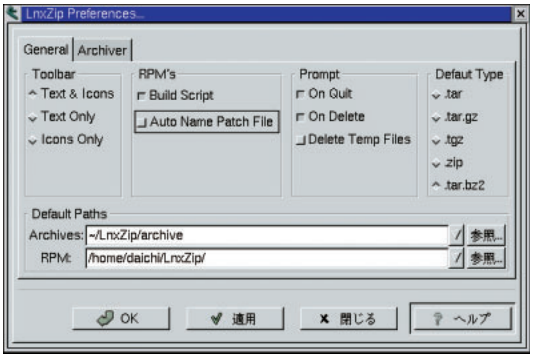
このあとで説明するRPM作成機能を使う場合は、[General]ページで以下の2箇所の設定を変更する必要がある。まずは、[Default Paths]の[RPM]の設定だ。初期設定では、ユーザーのホームディレクトリを意味する「~」を使って「~/LnxZip」に設定されているが、このままでは正常に動作しないた



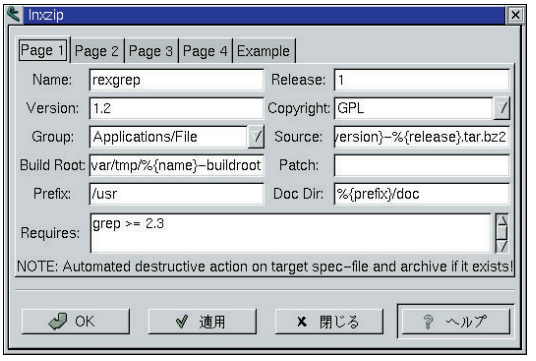
画面1
ウィンドウにはアーカイブ内のファイルが一覧表示される。



画面2
内蔵ビューアでテキストファイルの内容を閲覧できる。



画面3
設定ダイアログ。RPM作成機能を使う場合は変更の必要あり。



画面4
RPMのSPECファイルの内容をダイアログで設定する。

めフルパスで再設定する。たとえば、ホームディレクトリが「/home/hoge」の場合、「/home/hoge/LnxZip/」と設定すればいい。末尾に「/」が必要なことにも注意されたい。

次に、[RPM's]の[Auto Name Patch File]のチェックを外す。これがチェックされたままだと、実際にはパッチファイルが必要ない場合でもSPECファイルにパッチファイル情報が設定されてしまい、RPMパッケージ作成時にエラーが起こるからだ。

このほか、初回起動時にホームディレクトリに作成される「.rpmmacros」も一部修正する必要がある（そのままだと、RPMパッケージの作成時にエラーとなる）。修正内容は単純で、最後の行「%_tmppath …」の末尾に改行を追加するだけだ。

●RPMパッケージを作成する

それでは、LnxZipの特徴であるRPM作成機能を説明しよう。RPMのパッケージ作成の中心となるのは、「SPECファイル」と呼ばれる設定ファイルの作成だ。パッケージ名や依存関係、インストールするファイルやディレクトリなどの情報は、すべてこのファイルに書かれている。

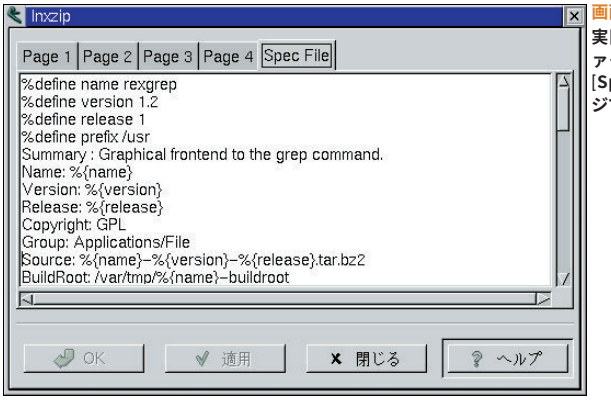
メニューから[RPM]－[Spec File]を選択すると、SPECファイルの設定内容がジャンル別にページ分けされたダイアログが開く（**画面4**）。[Page 1]ページでは、パッケージ名やバージョン、リリース番号、依存関係などを設定する。以下、[Page 2]ページではパッケージに関する説明、[Page 3]ページではビルドやインストールの際のコマンド、[Page 4]ページではインストールするファイルのフルパスといった情報を設定する。

各項目の意味や設定内容についての詳細は、本誌2000年3月号の記事「賢く使うUNIX」や、Webページでは「Making RPM」（http://vinelinux.org/MakingRPM/index.html）などの情報を参考にされたい。

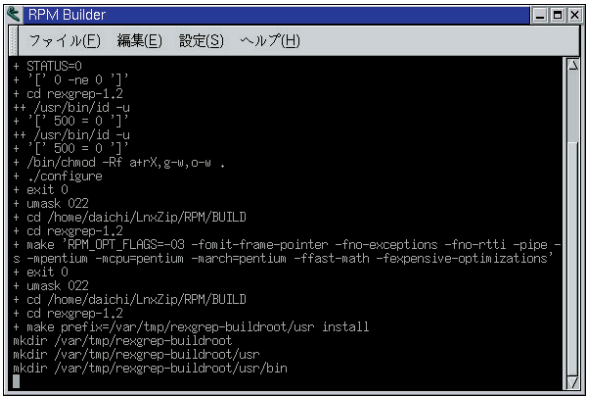
[適用]ボタンを押すと、設定内容に基づくSPECファイルが~/LnxZip以下に作成される。内容はダイアログの[Spec File]ページで確認できる（画面5）。続いて、ソースのディレクトリを指定すると、そこに含まれるファイルがtar＋bzip2でアーカイブ・圧縮され、LnxZipのウィンドウに表示される。これで準備完了だ。

メニューから[RPM]－[Build RPM]を選択すると、Gnome terminal上でrpmコマンドが実行され（**画面6**）、ソースパッケージは~/LnxZip/RPM/SRPMSに、バイナリパッケージは~/LnxZip/RPM/RPMS以下に作成される。

なお、rpmコマンドの実行中にエラーが起きてパッケージが作成されなかった場合は、コマンドの出力を記録したログファイル（~/LnxZip/tmp/rpm-build.log）で原因を調べよう。



画面5
実際のSPECファイルの内容を[Spec File]ページで確認。



画面6
rpmコマンドによってソースとバイナリのパッケージが作られる。

ラリーが気持ちいい本格卓球ゲーム

Cannon Smash

バージョン：0.3.9　　ライセンス：GPL

http://cannonsmash.sourceforge.net/
http://www.utmc.or.jp/~nan/csmash/（日本語）
http://mesa3d.sourceforge.net（Mesa3D）

Cannon Smashは、本格的な3D卓球ゲームだ。プレイヤーのタイプが3パターン用意されており、多彩な戦術が要求されるスポーツとしての卓球を楽しめる。また、サウンドドライバEsounD（esd）のある環境では、球がテーブルやラケットに当たる音が再現され、気持ちよくラリーを続けられる。動作には3Dライブラリの Mesa3DとGLUTが必要になる。快適にプレイするには3Dアクセラレータが必須だ。

●ビルドとインストール

Cannon Smashは、ソース一式をtar＋gzipしたtarボールのほか、RPMパッケージやKDE用GUIインストーラ付きのバイナリも配布されている。ただし、これらはバージョンが古いので、tarボールの最新版をインストールしたほうがよいだろう。

ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。なお、ビルドと実行に必要な3Dライブラリ（Mesa3DおよびGLUT）をあらかじめインストールしておく必要がある。

●ゲームの起動とプレイヤーの選択

「csmash」として起動すると、メニュー画面が表示され、バックで白熱したラリーが展開される（**画面1**）。はじめてプレイする場合は、[How to play]を選択して、操作を解説したデモを見るとよいだろう。[Start Game]を選択するとゲーム開始だ。

まずは、以下の3種類のプレイヤーの中から、好みのプレイタイプを選択する（**画面2**）。

- **Pen Attack（前陣速攻型）**

最もテーブルに接近した状態で戦う攻撃型。浮いた球を逃さず叩き付けるスマッシュが武器。

- **Pen Drive（ドライブ主戦型）**

テーブルから少し下がった状態で戦う攻撃型。球に強烈な前進回転をかけるドライブが武器。

- **Shake Cut（カット主戦型）**

テーブルからかなり下がった状態で戦う守備型。球に逆回転をかけるカットが武器。

はじめのうちは、攻撃が入りやすい前陣速攻かドライブ主戦がお勧めだ。一方、対戦相手となるコンピュータのタイプは、この3種類の中から毎回ランダムに決定される。

●マウスとキーボードで操作

プレイヤーの操作には、キーボードとマウスを併用する。キーボードは球を打ち込む位置の指定に、マウスはプレイヤーの移動とショットのために使用する。

球を打ち込む位置は、キーボードのフルキー部分を相手コートに見たて、左奥の1キーから右手前の\キーまで44カ所にも及ぶ細かな指定が可能だ（**画面3**）。指定した位置にはインジケータ（▼）が表示される。もちろん、プレイ中にも位置を変更できるので、「フォアの浅い位置にサーブして、レシーブされた球をバックの深い位置へ返す」といった戦術を実現できる。

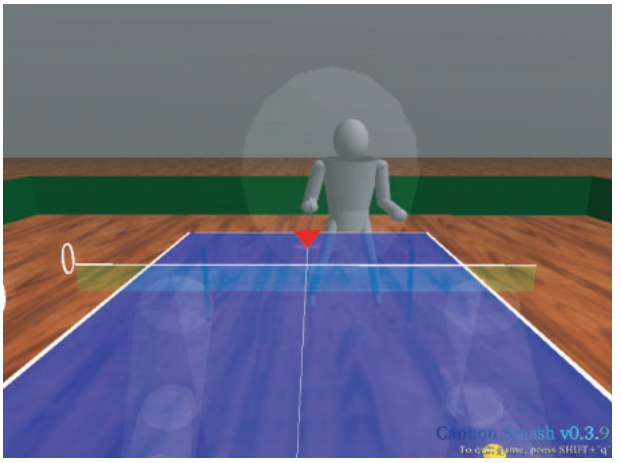
マウスのボタン（どれでも可）をクリックすると（サーブを含む）ショット動作が行われる。こちらのサーブから始まるので、ボタンを押してショットしよう。インジケータ（▼）に向かって球が飛んでいく（**画面4**）。

コンピュータが操作する相手プレイヤーがレシーブすると、返ってくる球の軌道が点線で表示される（**画面5**）。最適な打点が赤く表示されるので、赤い点のそばに球が来るのを待ってボタンをクリックしよう。

画面1
メニュー画面のバックでは白熱したラリーが続く。

画面2
操作するプレイヤーを3種類のタイプから選択。

画面3　相手のコートに打ち込む位置はキーボードで指定する。



画面4　こちらが打った球はインジケータ（▼）めがけて飛んでいく。

うまくタイミングが合うと「Nice!」と表示されて強い球を打てる。一方、タイミングがずれると「Bad」と表示され、返球が浮いて相手にスマッシュされたり、ひどいときにはそのまま空振りしてしまう。

相手が返す球の強さやコースによっては、マウスを動かしてプレイヤーを移動させないと、最適打点を示す赤い点が表示されない場合もある。たとえば、相手が短いサーブをしてきたときには前進して返さなくてはならないし、球が浮いてスマッシュされるときには、後ろに大きく下がってロビングする必要がある。

なお、ドライブ主戦型のプレイヤーはバックハンドよりフォアハンドのほうが強い球を打てるため、操作に慣れたら、バックに返された球に対しても、フットワークを生かしてフォアで返すようにするとよい。

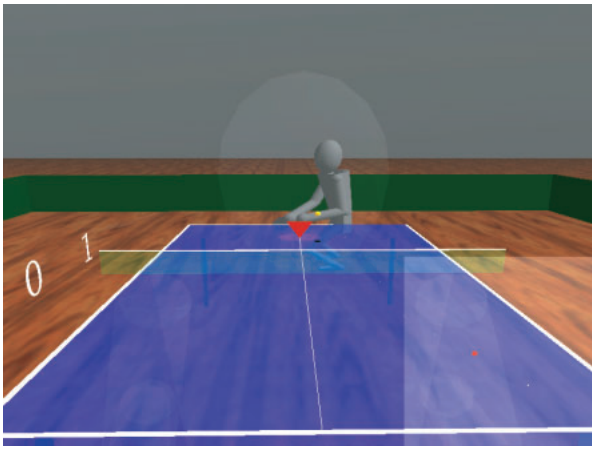
●ルールと設定変更

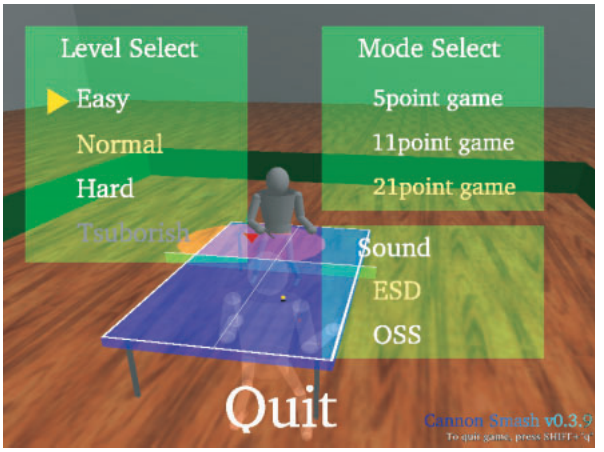
ルールは実際の卓球に基づいており、5球ごとにサーブをチェンジし、先に21ポイントを取ったほうが勝ちだ（1セットマッチ）。設定によっては、5ポイント先取や11ポイント先取に変更することもできる。

メニュー画面で[Config]を選択すると設定変更画面になる（**画面6**）。勝利ポイントの変更のほか、コンピュータの強さやサウンドドライバの切り替えも可能だ。たとえば、コンピュータが強すぎて何度対戦しても勝てない場合は、[Level Select]を[Easy]に変更するとよいだろう。

これまでの卓球を題材としたゲームは「温泉でピンポン」レベルのものが多かったが、Canon Smashは球のスピードや軌道がスポーツとしての卓球にかなり近い。このため、ラリーを続けているだけで実に爽快感がある（試合をしないでラリーを続けられる練習モードも欲しいところだ）。球がテーブルやラケットに当たる音を聞いていると、卓球部に所属していた人はありありと記憶がよみがえってくるのではないだろうか。

なお、作者のWebページの情報によれば、サウンドエフェクトの充実などに加え、人間同士の通信対戦についても検討されているようだ。実現する日を楽しみにしている。


画面5　返球の軌道上に表示された赤い点が最適打点だ。



画面6　勝利ポイントやコンピュータの強さはこの画面で変更する。

プロセス情報をグラフィカルに表現する

LavaPS

バージョン：1.10　　ライセンス：GPL

http://www.isi.edu/~johnh/SOFTWARE/LAVAPS/

LavaPSは、プロセス情報を不定形な「ブロブ（メタボール）」状のグラフィックで表示するソフトだ。各プロセスのメモリ消費量やCPU使用量、名前などの情報をブロブの大きさや動き、色相、彩度などのアナログな要素によって表現するため、現在のシステムの状況を感覚的に理解できる。さらに、ポップアップウィンドウで各プロセスに関する詳しい情報を表示することも可能。動作にはTcl/Tkが必要だ。

●ビルドとインストール

LavaPSは、tarボールとRPMパッケージの両方で配布されている。Red Hat系のディストリビューションではRPMパッケージを利用したほうが簡単だ。バイナリパッケージのほか、ソースパッケージも用意されている。tarボールからビルドする場合は、「./configure」→「make」→「make install」という一般的な手順をとる。

●プロセス情報を視覚的に表現

「lavaps&」として起動すると、さまざまな色のブロブを含んだウィンドウが開く（**画面1**）。ウィンドウのサイズは自由に変更可能だ。ブロブとプロセスは1対1で対応しており、プロセスが生成されると新たなブロブが追加され、プロセスが終了する対応するプロブが消滅する。

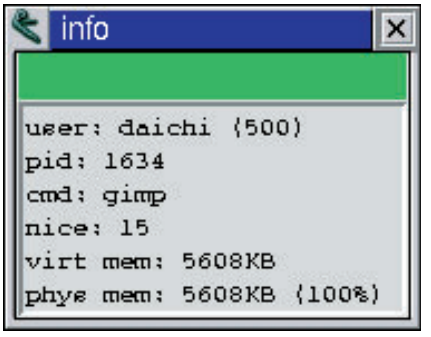
各プロセスの情報は、対応するブロブの大きさや動き、色などによって表現される。たとえば、ブロブの大きさはメモリ消費量、動きはCPU使用量に比例する。つまり、メモリを大量に消費しているプロセスのブロブは大きく表示され、CPUを多く使うプロセスのブロブはよく動く。

また、ブロブの色合い（色相）はプログラム名をハッシュした数値によって決まる。たとえば、Xのブロブはオレンジ、Emacsのブロブはブルーといった具合だ。また、色の鮮やかさ（彩度）はプロセスの実行頻度に比例しており、実行されないプロセスのブロブはくすんだ色に変わっていく。

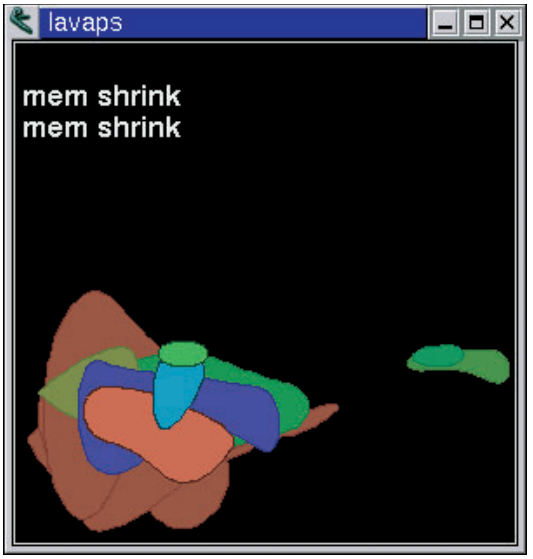
●詳細情報の表示

ブロブ上でマウスの左ボタンを押し続けると、ウィンドウがポップアップして、そのブロブに対応するプロセスの情報（プログラム名や実行ユーザー名、プロセスID、メモリ消費量など）が表示される（**画面2**）。

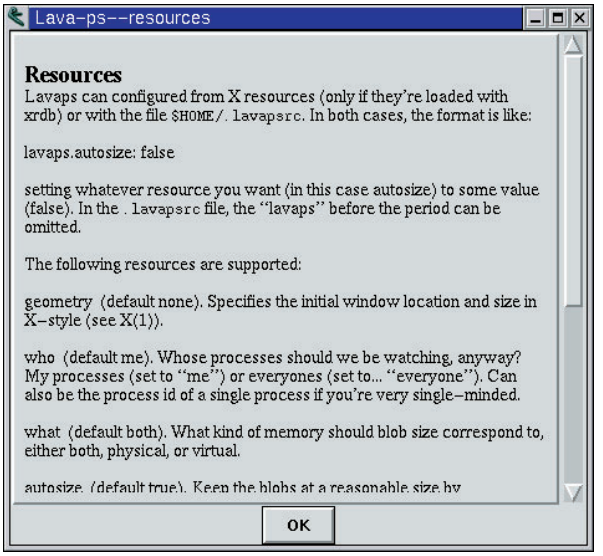
また、右ボタンのクリックで開くポップアップメニューでは、ユーザーごとのプロセス表示と全プロセス表示の切り替えや、LavaPSに関する各種ヘルプを表示を行う。ヘルプの中には、ドキュメントに書かれていないリソース情報も含まれているので、目を通しておきたい（**画面3**）。



画面2
プロセスの詳細な情報を表示するポップアップウィンドウ。



画面1
実行中のプロセスに対応する「ブロブ」が表示される。



画面3
リソース情報などをヘルプで確認できる。

再生中のサウンドをさまざまに視覚化する

eXtace

バージョン：1.2.23　　ライセンス：GPL

http://extace.sourceforge.net/
http://www.fftw.org/（FFTW）
http://sourceware.cygnus.com/gsl/（GSL）

eXtaceは、再生中のサウンドを高速フーリエ変換（FFT）して得られたスペクトル情報をリアルタイムに視覚化するソフトだ。表示方法は、3Dワイヤフレーム、サーフェス、スパイク、グラフィックイコライザ、グラディエントスコープ、スペクトログラムの6種類で、再生中もボタン操作でこれらを自由に切り替えられる。動作にはサウンドドライバEsounD（esd）と、FFTライブラリが必要だ。

●ビルドとインストール

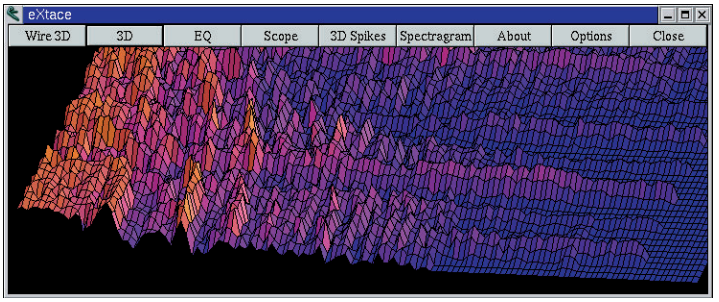
まずは、FFTライブラリのFFTWかGSLをインストールする。FFTWのほうがGSLよりも30%ほど高速に動作するとのことなので、特に理由がなければFFTWを使うとよいだろう。

eXtaceは、ソース一式をtar＋gzipしたtarボールで配布されている。ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。

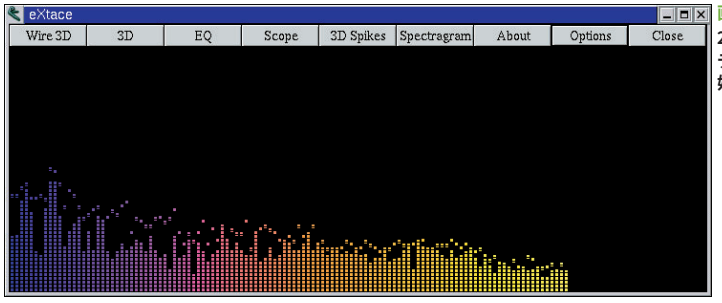
なお、ソース展開先のextaceディレクトリに、テスト用ツール「sine」が用意されている。これは、周波数をが段階的に変化するサイン波を発するもので、eXtace起動後に「./sine ¦ esdcat」とすることでeXtaceの動作確認が可能だ。

●ボタンひとつで表示を切り替え

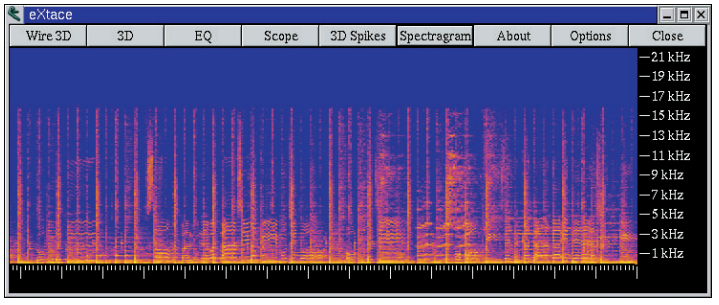
「extace&」として起動すると、上部にボタンが並んだウィンドウが開く。



画面1
再生されるサウンドのスペクトル情報を3Dサーフェス表示。



画面2
2Dのイコライザ表示。ディケイをかけると格好いい。



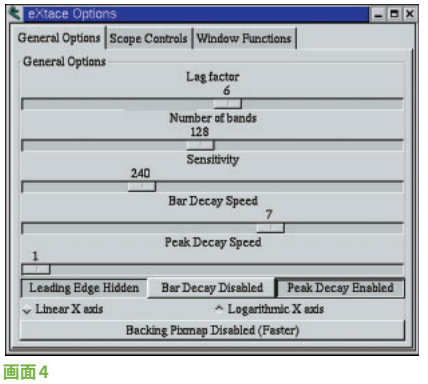
画面3
スペクトログラム（スペクトルの時間的変化）の表示も可能だ。

その後、EsounD（esd）を利用したサウンド再生ソフト（FreeAmpなど）を使用すると、FFTにより得られたスペクトル情報がリアルタイムに3Dサーフェス表示される（**画面1**）。

表示方法を変更するには、ウィンドウ上部のボタンを押す。サウンド再生中の切り替えも可能だ。3D表示には、サーフェス以外にもワイヤフレーム（[Wire 3D]）とスパイク（[3D Spikes]）が用意されており、表示の傾きや更新速度を「Direction」ウィンドウで変更できる。

上記の3D表示以外に、イコライザ（**画面2**）やグラディエントスコープ、スペクトログラム（**画面3**）などの表示方法が用意されており、再生中でも切り替え可能だ。

また、[Option]ボタンで開くダイアログでは、バンド数やディケイ速度（バーなどが落下していく速度）などを細かく調整できる（**画面4**）。



画面4
バンド数やディケイ速度の設定はこの設定ダイアログで行う。

短期集中連載

Linux上でのビジュアル開発を実現する

JBuilder 3.5 Foundation日本語版

第2回 アプレットを作る

文：加藤大受
Text：Daiju Kato
daiju@ms.tokyo.jcom.ne.jp

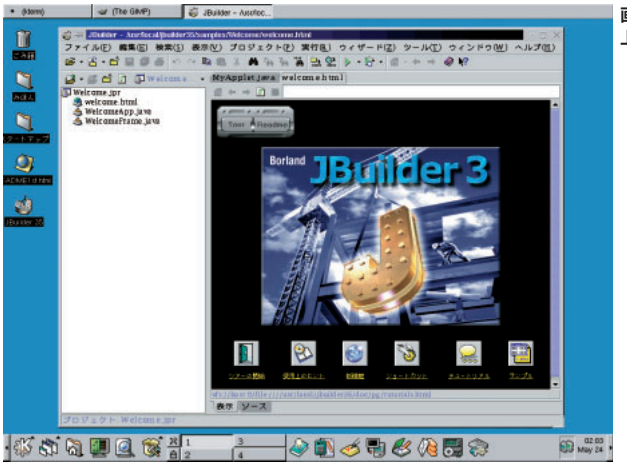
前回はJBuilder 3.5 Foundationのインストール、デバッグ機能の説明、簡単なアプリケーションの作成を行いました。今回はJBuilder 3.5 Foundationの提供しているコンポーネントについて解説し、簡単なアプレットを作成してWebサーバへ配布する方法について解説します。

前回はLASER5 Linux 6.0 Rel.2を使用して説明しましたが、Red Hat 6.1日本語版をサポートしているJBuilder 3.5 FoundationはRed Hat 6.1をベースとしているVine Linux 2.0上でも問題なく動作します（**画面1**）。Vineがお好きな方はVine上で使用してみてください。今回はVine Linux 2.0を使って解説していくことにします。

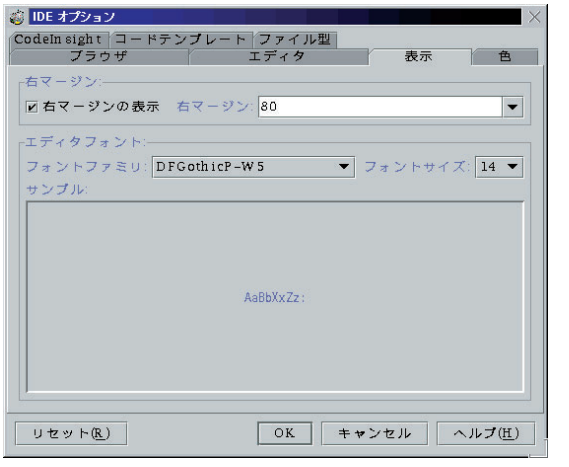
今月号の付録CD-ROMにVine Linux 2.0が収録されていますので、これを使うのもいいでしょう。ただし、Vine Linux 2.0はJBuilder 3.5製品版でオフィシャルにサポートしているディストリビューションではありません。JBuilder 3.5 Foundationにはサポートがありませんので関係ありませんが、製品版を使用される場合は注意が必要です。

環境設定

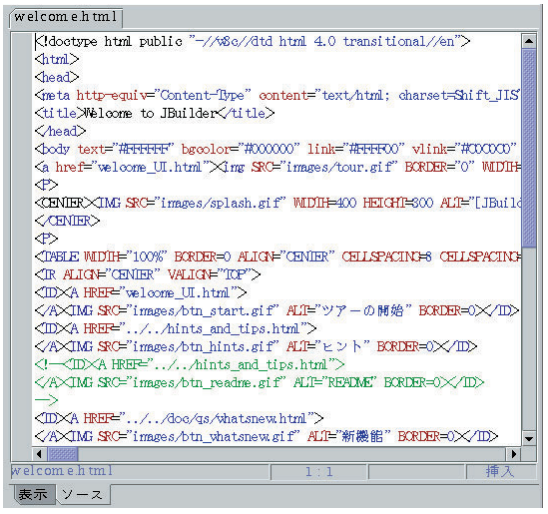
さて、前回のJBuilder 3.5 Foundationのインストールのときに表示されるフォントがあまりきれいではないと思われた方も多いと思います。これはLinux版のJDKがデフォルトで使用しているフォントの問題です。今回はまず、デフォ



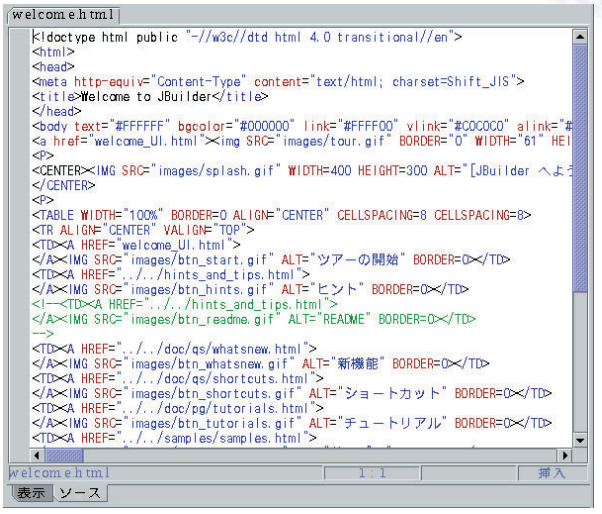
画面1　Vine 2.0上のKDEで実行



画面2　フォントの選択



画面3　デフォルトのfont.properties.jaの場合



画面4　DynaLabを使用するように変更した場合（フォントサイズは同じ14に設定）

ルトで使用しているフォントを商用フォントに変更してみましょう。ただし、この変更はディストリビューションのFTP版を使用している方は行わないでください。商用フォントが利用できないと、まったく日本語が表示されなくなりますので注意してください（付録CD-ROMも同様）。

JDKが使用する日本語フォントの設定はfont.properties.jaというファイルに記述されています。前回、/usr/local/jdk1.2.2にJDKをインストールしましたので、このファイルは/usr/local/jdk1.2.2/jre/libのディレクトリにあります。このファイルには**リスト1**のようにフォントの設定が記述されています。このファイルをよく見ると、wadalabフォントを使用しているのが分かります。このwadalabフォントの設定を商用フォントの設定に変更することで、きれいなフォントを表示することができます。

emacsなどのエディタでfont.properties.jaを開き、**リスト2**を参考にしてwadalabの設定をDynaLabに置換してください。

使用しているディストリビューションがLASER5 Linux 6.0でも同じ設定でフォントを変更できます。Vine 2.0、LASER5 Linux 6.0以外のディストリビューションの場合は、使用している商用フォントを確認のうえ行ってください。ここで行った設定は筆者の環境の例ですが、このファイルをチューニングしていくことでフォントをよりきれいに表示することができます。

font.properties.jaを変更したら、続いて環境変数を設定します。JDK（およびJRE）は、環境変数JAVA_FONTSに設定しているパスからfont.properties.jaに書かれているフォントを探します。

次のようにして環境変数JAVA_FONTSを設定してください。

▼リスト1　font.properties.jaの一部

```
# Serif font definition
#
serif.0=-b&h-lucidabright-medium-r-normal--*-%d-*-*-p-*-iso8859-1
serif.1=-wadalab-mincho-medium-r-normal--*-%d-*-*-c-*-jisx0201.1976-0
serif.2=-wadalab-mincho-medium-r-normal--*-%d-*-*-c-*-jisx0208.1983-0
```

▼リスト2　フォントの置換文字列

```
■明朝の設定
置換前　"-wadalab-mincho-medium-r-normal--*-%d-*-*-c-*-jisx0201.1976-0"
置換後　"-DynaLab-DFMinchoP-W3-r-normal--0-0-0-0-c-0-jisx0201.1976-0"
置換前　"-wadalab-mincho-medium-r-normal--*-%d-*-*-c-*-jisx0208.1983-0"
置換後　"-DynaLab-DFMinchoP-W3-r-normal--0-0-0-0-c-0-jisx0208.1983-0"

■ゴシックの設定
置換前　"-wadalab-gothic-medium-r-normal--*-%d-*-*-c-*-jisx0201.1976-0"
置換後　"-DynaLab-DFGothicP-W5-r-normal--0-0-0-0-c-0-jisx0201.1976-0"
置換前　"-wadalab-gothic-medium-r-normal--*-%d-*-*-c-*-jisx0208.1983-0"
置換後　"-DynaLab-DFGothicP-W5-r-normal--0-0-0-0-c-0-jisx0208.1983-0"
```

▼リスト3　.bash_profileに追加

```
PATH=$PATH:$HOME/bin
BASH_ENV=$HOME/.bashrc
USERNAME=""
JAVA_FONTS=/usr/X11R6/lib/X11/fonts/TrueType:/usr/local/jdk1.2.2/jre/lib/fonts
export USERNAME BASH_ENV PATH LESSOPEN JAVA_FONTS
```

```
$ export JAVA_FONTS=/usr/X11R6/lib/X11/fonts/TrueType:/usr/local/jdk1.2.2/jre/lib/fonts
```

ここで設定しているのはDynaLabのフォントのパスとJDKに含まれているフォントのパスです。この設定を.bash_profileに書き込んでおけば、ログイン時のデフォルトの環境になります。.bash_profileはホームディレクトリ（/home/＜ユーザー名＞）にあります。こちらのファイルを開いて、**リスト3**のように追加してください。

続いてJBuilder 3.5 Foundationを起動し、メインメニューから［ツール（T）］－［IDEオプション（O）］を選択して、［IDEオプション］ダイアログボックスを表示し、［表示］のページを選択します。［フォントファミリ］のリストボックスで「DFGothicP-W5」を、［フォントサイズ］のリストボックスから「14」を選択します（**画面2**）。選択したら［OK］ボタンをクリックしてダイアログボックスを閉じます。

ソースを表示させて見てみると前よりもきれいに表示されていることがわかると思います（**画面3・4**）。

AWTとJFC/Swing

Javaでユーザーインターフェイスを作成する場合、AWTまたはJFC/Swingと呼ばれるGUIのクラスを利用します。AWTはAbstract Window Toolkitの略で、JDK 1.0の時代からあるJavaの基本的なGUI用のコンポーネントです。このコンポーネントの提供しているユーザーインターフェイスの機能は低く、Windows 3.1レベルのものです。一方、JFC/Swingはその後に登場し、より洗練されたGUIを提供するコンポーネントです。JFC/Swingを使用すると非常にきれいなGUIを作成することができるのですが、アプレットで使用した場合、ファイルサイズが大きくなる問題が発生します。そのため、アプレットはAWTベースで作成することをお勧めします。これらのコンポーネントを設計画面で使用していくことで、JBuilderではGUIを簡単に作成することができます。

アプレットとは？

前回は、簡単なJavaアプリケーションを作成しました。今回はブラウザ上で実行されるJavaプログラムであるアプレットに挑戦してみましょう。まず、アプレットの仕組みを説明します。

図1はJavaアプレットの仕組みについて簡単に図にしたものです。

ブラウザがWebサーバにURLを渡すとWebサーバは指定されたファイルをブラウザに送り返します。ブラウザが受け取ったHTMLファイルに<APPLET>タグがある場合、ブラウザはWebサーバにアプレットファイルのダウンロード要求を送り、希望するアプレットを受け取ります。受け取ったアプレットはブラウザに組み込まれているJava VMの上で実行されます。つまり、Javaアプレットとはブラウザ上で実行されるJavaプログラムのことです。クライアント側にプログラムの配布の必要がない非常に便利なテクノロジーなのですが、配布されるアプレットのサイズが大きい場合、時間がかかるなどの問題点もあります。また、ブラウザで使用できるJDKのバージョ

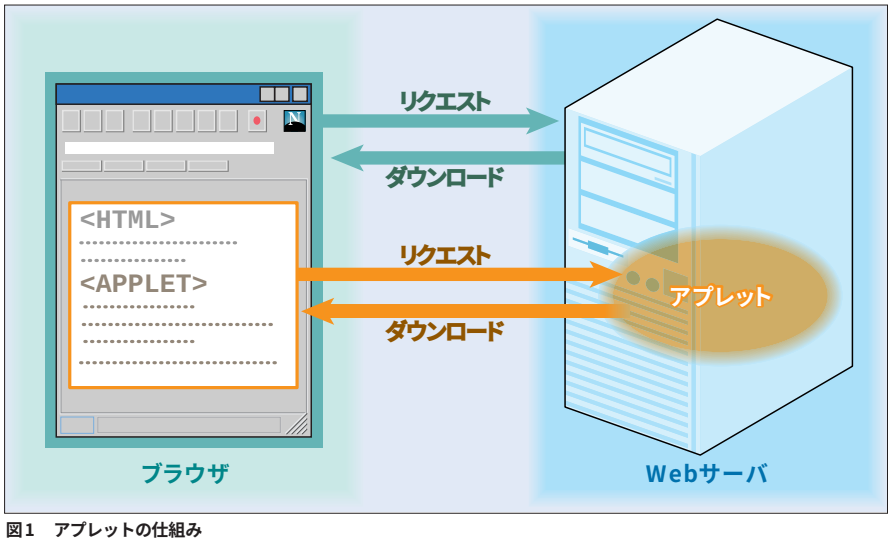
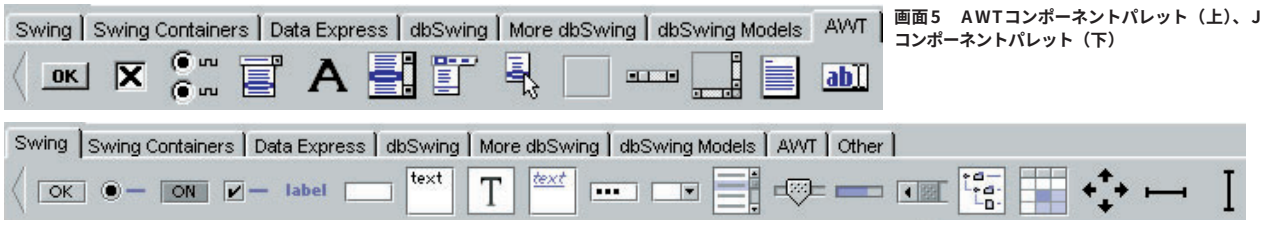


図1　アプレットの仕組み



画面5　AWTコンポーネントパレット（上）、JFC/Swingコンポーネントパレット（下）

画面6　プロジェクトウィザードステップ 1/2

画面7　アプレットウィザードステップ 1/3

画面8　アプレットウィザードステップ 3/3

ンは1.1がまだ一般的なため、JBuilder 3.5 FoundationなどJava 2に特化した製品を使用する場合は多少の注意が必要です（JBuilder 3.5の製品版ではJDK 1.1を使用することができます）。

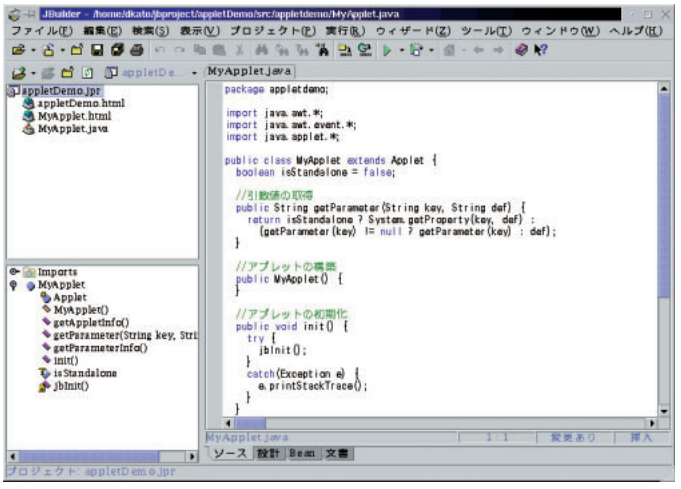
アプレットの作成

それでは、JBuilder 3.5 Foundationを使用して簡単なアプレットを作成してみましょう。今回作成するアプレットは「Welcome!」という文字列がゆっくりと1文字ずつ点滅しながら表示されるアプレットです。

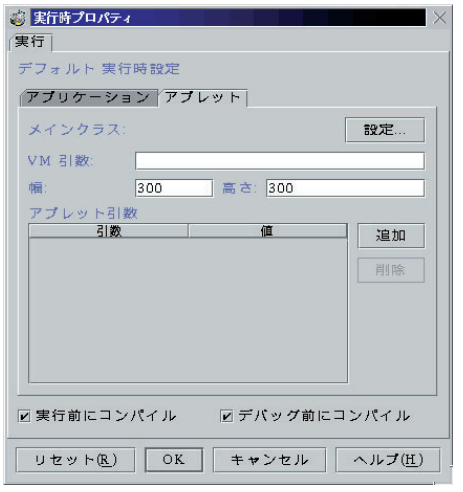
JBuilder 3.5 Foundationを起動し、「Welcomeプロジェクト」などのプロジェクトが開いている場合はメインメニューの［ファイル（F）］－［プロジェクト"＜プロジェクト名＞"を閉じる（E）］で開いているプロジェクトを閉じてください。続いて、［ファイル（F）］－［新規（N）］で表示される［オブジェクトギャラリ］のダイアログボックスから［アプレット］を選択します。プロジェクトを作成していないため、**画面6**のようなプロジェクトウィザードが起動します。プロジェクトファイルの名前を「/home/＜ユーザー名＞/jbproject/appletDemo/appletDemo.jpr」と入力し、［終了］ボタンをクリックします。続いて、**画面7**のようなアプレットウィザードが起動されます。クラス名を「Applet1」から「MyApplet」に変更し、ベースクラスのリストボックスで「java.applet.Applet」を選択します。選択したら、［次へ］ボタンをクリックしてウィザードを進めます。「ステップ 2/3」でアプレット引数の画面が表示されますが、今回は引数を使用しませんので、［次へ］ボタンをクリックしてウィザードを進めます。「ステップ 3/3」ではアプレットを使用するHTMLについての設定になります。タイトルを「JBuilder Foundation Applet Demo」に、高さを「40」に変更し、［終了］ボタンをクリックします（**画面8**）。アプレットウィザードが終了すると、アプレット本体である「MyApplet.java」とアプレットを呼び出すHTMLファイルである「MyApplet.html」が生成されました（**画面9**）。ここで、［ファイル（F）］－［すべて保存（A）］を選択し、生成されたすべてのファイルを保存します。プログラムを書くときは不慮の事故に備えてこまめに保存する習慣をつけておくことをお勧めします。

アプレットの実行

ではここで生成されたアプレットを実行してみましょう。［プロジェクト（P）］－［プロジェクトのメイク"appletDemo.jpr"（M）］を選択してコンパイルを実行します。続いて、［実行（R）］－［プロジェクトの実行（N）］を選択します。デフォルトで実行されるファイルが設定されていない場合、［実行時プロパティ］のダイアログボックスが表示されます。ここで、アプレットのページを選択します（**画面10**）。「メインクラス」の［設定］ボタンをクリックして、「appletDemo.MyApplet」クラスを選択します（**画面11**）。ここ



画面9　MyApplet.javaとMyApplet.htmlが生成された

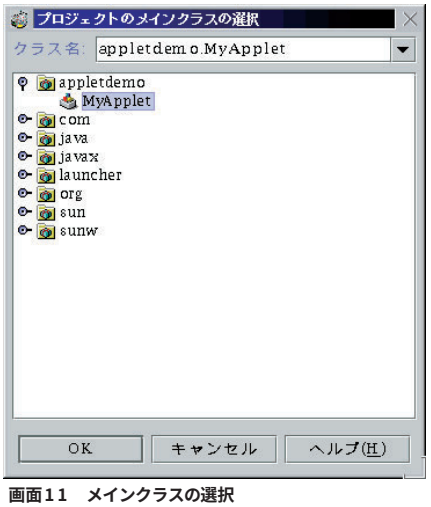


画面10　実行時プロパティ

で［OK］ボタンをクリックすると、指定されたメインクラスが実行されます。本来なら、これで作成したアプレットが実行されるはずなのですが、**画面12**のようなエラーに遭遇します。これはJBuilder 3.5 FoundationのSolaris版、Linux版の既知の問題で、インプライズ社のWebサイトに回避方法が書かれています（**表1**）。この回避方法に従って、アプレットが正しく実行できるように設定しましょう。

まず、rootでログインまたはsuでスーパーユーザーになり、JBuilder.config（デフォルトでは/usr/local/jbuilder35/binディレクトリ内）をエディタで開きます。次に「addjars」エントリを相対パスから絶対パスに変更します。変更したJBuilder.configは**リスト4**のようになります。この設定を行うとJBuilder Foundation内からアプレットを実行することができます。

設定が終了したら、再度JBuilder Foundationを起動して［実行（R）］－［プロジェクトの実行（N）］を選択してみましょう。**画面13**のような画面が表示されます。この画面はappletviewerと呼ばれるアプレットのテスト用のツールです。JBuilder 3.5 Foundationはアプレットの作成時は自動的にこのツールを呼び出して、作成しているアプレットのテスト実行を行えるようにしています。アプレットの動作をこの画面でテストしながらアプレットを作成していくことになります。実行時は画面下に標準出力の内容が表示されます。このウィンドウは実行したメインクラス名のタグの上で右クリックすることで表示されるスピードメニューから、［すべてのタブを削除（A）］を選択することで非表示にすることができます。



画面11　メインクラスの選択

画面13　アプレットを実行したところ

```
Exception in thread AgmainAh
java.lang.noClassDefFoundError:com/Borland/jbuilder/runtime/AppletTested
```

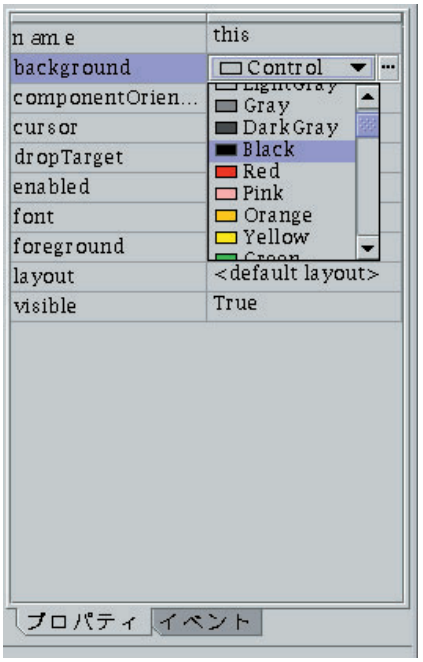
画面12　アプレット実行時のエラー

URL	http://www.inprise.co.jp/jbuilder/papers/javamadness/javamadness_append.html

表1　Linux、Solarisでアプレットが実行できない現象について

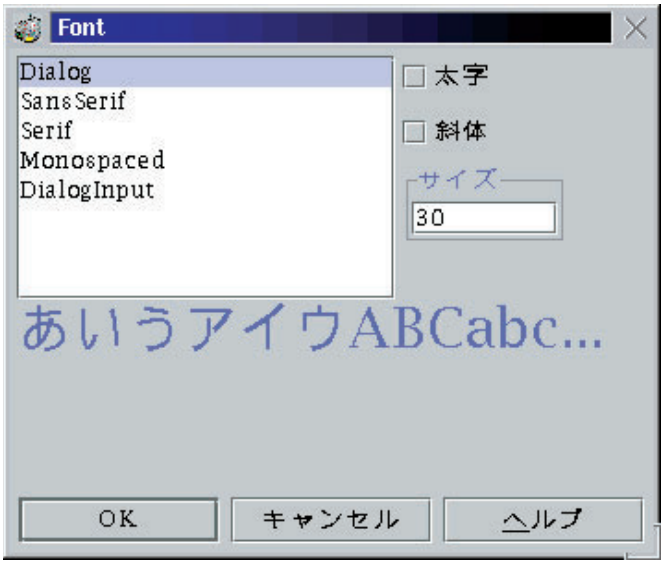
▼リスト4　修正したJBuilder.config

```
・
省略
・
#addjars ../lib
#addjars ../lib/ext
#---アプレットの実行の不具合の回避
addjars /usr/local/jbuilder35/lib
addjars /usr/local/jbuilder35/lib/ext
・
省略
・
```



画面14　背景色の変更

画面15　Labelコンポーネント



画面16　フォントサイズの変更

アプレットの実装

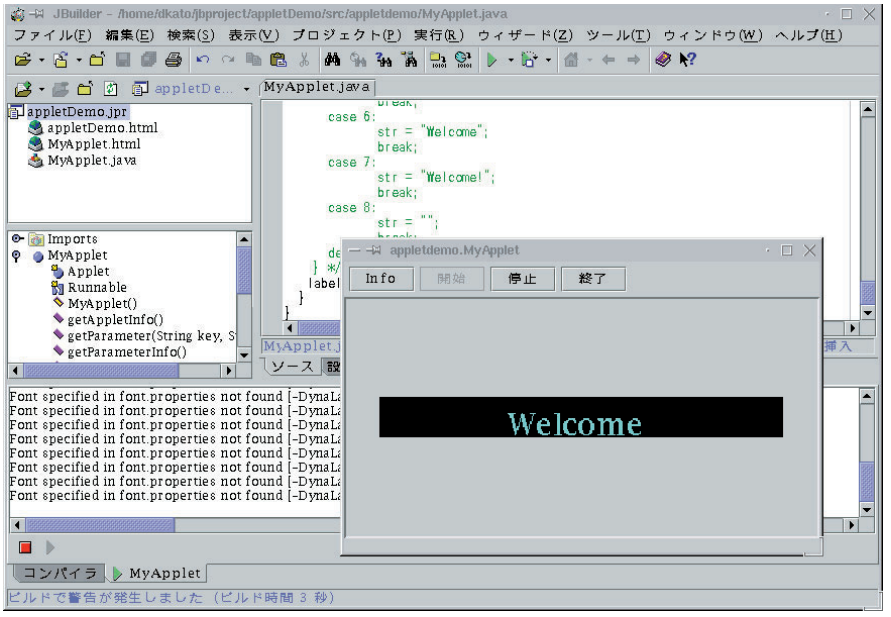
それではアプレットの中身を実装していきましょう。設計のタグを選択して、設計画面を表示します。まず、アプレットの背景色を変更します。インスペクタでbackgroundのプロパティ横のリストボックスから「Black」を選択します（**画面14**）。続いて、AWTのページから「A」の絵のアイコンのLabelコンポーネント（**画面15**）を選択し、フォーム上に貼り付けます。貼り付けたLabelの背景色と文字の色が黒になっているので見えませんが、構造ペインを確認するとlabel1というオブジェクトがあることがわかります。構造ペインでlabel1が選択されていることを確認して、label1のforegroundプロパティを「Black」から「Cyan」に変更します。文字の色が変更されたのでlabel1という文字列が表示されます。続いて、textプロパティの値を「label1」から「Welcome!」に変更します。ここで、デフォルトのフォントサイズでは小さいのでフォントサイズを大きくしましょう。fontプロパティを選択して、「Font」ダイアログを表示し、サイズを12から30に変更します（**画面16**）。これで、表示する文字列の設定が終わりました。ソースのタグを選択して、ソースエディタに戻ります。

メインの処理とは別に他の処理を行うにはスレッドという方法を利用します。メインの処理では表示しているアプレットに関する処理、たとえば位置を移動したときにそれに対応する処理などを行います。スレッドを使用することで、文字の描画が可能になります。もし、スレッドを使用しないで行った場合、文字の描画以外の処理はいっさいできなくなってしまうという問題が発生します。複数の処理を同時に実行するときに使用される方法がスレッドというわけです。Javaはマルチスレッドに対応した言語であり、スレッドを簡単に扱える特徴があります。JavaAPIにThreadクラスが用意されており、次の3つのメソッドが用意されています。

`start()` **スレッドの開始**
`run()` **スレッドの中身**
`stop()` **スレッドの停止**

スレッドを開始するとrun()メソッド内の処理が実行されていくわけです。今回のアプレットでは文字を1文字ずつ描画していく処理をこちらに書いていきます。描画する文字列は8文字（Welcome!）です。例では文字数が9文字以上にならないようにする必要があります。**リスト5**は完成したアプレットのソースコードです。また、**リスト6**はアプレットを呼び出すHTMLファイルです。<APPLET>タグでアプレットを指定していることが分かります。完成したソースコードを利用しながら説明していきましょう。

Appletクラスを継承してMyApplet



画面17　JBuilder 3.5 Foundation内で実行したところ

クラスを作成していますが、アプレットウィザードで生成したコードに変更を加え、implements Runnableを追加しています。Runnableというのはインターフェイスです。アプレット自体はスレッドを扱うことができません。スレッドを扱う場合はこのインターフェイスをインプリメントすることになります。インターフェイスとは、その名のとおりインターフェイスだけを実装したクラスのことをいい、中身がありません（正確に説明すると抽象メソッドのみからなるクラスです）。Runnableインターフェイスはrun()というメソッドを持っています。インターフェイスを使用する場合、そのインターフェイスが持っているメソッドをすべて実装しなければなりません。しかし、スレッドを使用する場合、必ずrun()メソッドは実装しますので、それほど気にすることはないでしょう。

run()メソッド内でアプレットの実行中、ずっとループするようにしています。ここではwhile文を使用しています。while文などの処理を書くときはCTRL+Jを押すことで、コードテンプレートという機能を呼び出すことができます。コードテンプレートはwhile、ifなどの条件文などのテンプレートが入っている入力支援機能です。

```
while (条件) {
  <条件に一致する場合に実行する処理>;
}
```

また表示を遅らせるため、毎回200ミリ秒だけ待たせるようにしています。

なお、スレッドに関する処理などを行うときは必ず例外処理を書きます。例外処理はtry…catch構文を使用します。例外処理を使えば、例外の内容によって処理を変更することができます。たとえば、入出力に関する例外であれば、IOExceptionを指定します。今回使用しているExceptionはすべての例外に対応できる一番上位のクラスです。

```
try {
  <処理>;
} catch (例外クラス) {
  <例外が発生したときに行う処理>;
}
```

run()メソッドではwriteText()メソッドを呼んでいます。writeText()メソッドは文字列を書いていく処理を行っています。文字型の変数を引数に取り、現在のlabel1オブジェクトの文字列を渡しています。「Welcome!」という文字列は8文字ですので、switch文を使用して文字数ごとに処理を行っています。switch文で指定できる変数は文字型または整数型です。break文を入れないと条件に一致するすべてのケースに対してそれらの処理が行われますので、かならずbreak文を入れて、switchのループから抜け出すようにしてください。文字列の設定が終了したら、label1オブジェクトにテキストを設定します。label1の文字列を取得するときはgetText()メソッド、設定するときはsetText()メソッドを使用しています。Javaでは直接プロパティにアクセスすることが禁止されていますので、必ずset～()/get～()というメソッドを利用してプロパティにアクセスします。これらのメソッドをセッター、ゲッターと呼んでいます。

```
switch (変数) {
  case <値1> :
       <処理>;
  case <値2> :
       <処理>;
   ....
  default:
       <デフォルトの処理>;
}
```

アプレットの実行と配布

以上でアプレットの作成が終了しましたので、JBuilder Foundation内から実行してみましょう。appletviewer内で

▼リスト5　MyApplet.java

```
/*-------------------------------------------------------
  JBuilder Foundationで作成するアプレットのデモ
-------------------------------------------------------*/

package appletdemo;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;

//アプレットにRunnableインタフェースを指定する
public class MyApplet extends Applet implements Runnable{
  boolean isStandalone = false;
  Label label1 = new Label();

  //引数値の取得
  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }

  //アプレットの構築
  public MyApplet() {
  }

  //アプレットの初期化
  public void init() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  //コンポーネントの初期化
  private void jbInit() throws Exception {
    label1.setFont(new java.awt.Font("Dialog", 0, 30));
    label1.setForeground(Color.cyan);
    label1.setText("Welcome!");
    this.setBackground(Color.black);
    this.add(label1, null);
  }

  //アプレットの情報取得
  public String getAppletInfo() {
    return "Applet Information";
  }

  //引数情報の取得
  public String[][] getParameterInfo() {
    return null;
  }

  //スレッド変数の作成
  Thread thr=null;

  //スレッドの開始メソッド
  public void start() {
    if (thr==null) {
       //スレッドを新規に作成
       thr = new Thread(this);
       thr.start();
    }
  }

 //スレッドの停止メソッド
  public void stop() {
    if (thr != null) {
       thr.stop();
    }
  }

  //スレッドの中身
  public void run() {
    while (true) {
      try {
        //200ミリ秒待つ
        thr.sleep(200);
        //文字列を書くメソッドを呼び出す
        writeText(label1.getText());
      } catch (Exception e) {
        //エラー情報の表示
        e.printStackTrace();
      }
    }
  }

  //文字列を書くメソッド
  private void writeText(String str) {
    //今の文字数を判断して次の文字を指定
    switch (str.length()) {
      case 0:
             str = "W";
             break;
      case 1:
             str = "We";
             break;
      case 2:
             str = "Wel";
             break;
      case 3:
             str = "Welc";
             break;
      case 4:
             str = "Welco";
             break;
      case 5:
             str = "Welcom";
             break;
      case 6:
             str = "Welcome";
             break;
      case 7:
             str = "Welcome!";
             break;
      case 8:
             str = "";
             break;
      default:
    }
   //ラベルに設定
    label1.setText(str);
  }
}
```

```
$ ps aux | grep 'httpd'
root       532  0.0  0.6  2576 1080 ?        S    03時07分   0:00 httpd
nobody     538  0.0  0.5  2764  916 ?        S    03時07分   0:00 httpd
                                    ・
                                  省略
                                    ・
nobody     548  0.0  0.6  2764  984 ?        S    03時07分   0:00 httpd
dkato     2256  0.0  0.2  1628  480 pts/0    S    12時00分   0:00 grep httpd
```

画面18　起動している場合

```
$ ps aux | grep 'httpd'
dkato     2256  0.0  0.2  1628  480 pts/0    S    12時00分   0:00 grep httpd
```

画面19　起動していない場合

作成したアプレットが実行され、「Welcome!」の文字列が1文字ずつ点滅しながら描画されていきます(**画面17**)。描画の速度を変更したい場合は、sleep()メソッドの値を変更してください。

続いて、Webサーバに配布します。マシンにApacheがインストールされているかを確認します。

```
$ rpm -q apache
```

「apache-1.3.11-0v11」と表示されればApacheがインストールされています。「パッケージapacheはインストールされていません。」と表示された場合はApacheがインストールされていないので、CD-ROMからインストールします。

```
$ su
# mount /mnt/cdrom
# rpm -ivh /mnt/cdrom/Vine/RPMS/
apache-1.3.11-0v11.rpm
```

続いて、Apacheが起動しているかどうかを確認します。

```
$ ps aux | grep 'httpd'
```

起動している場合、**画面18**のように表示されます。**画面19**のように表示された場合は起動されていませんので、Apacheを起動します。

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

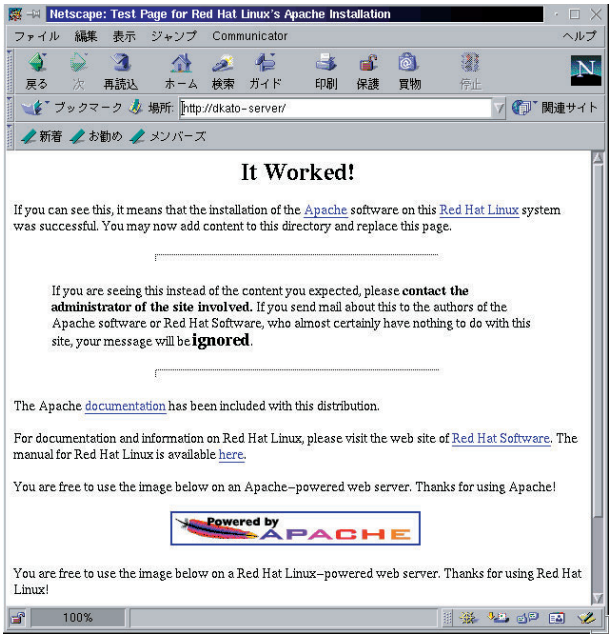
Apacheが起動したら、Netscapeを実行してサーバにアクセスしてみましょう。トップのindex.htmlが表示されると思います（**画面20**）。

それでは、作成したアプレットをコピーしてWebサーバ経由でアクセスしてみましょう。Apacheのドキュメントディレクトリにhtmlファイルとクラスファイルをコピーします。クラスファイルはappletdemoディレクトリごとコピーします。これは、HTMLファイルからパッケージ.クラス名で呼び出していますので、パッケージ名にあたるディレクトリがないとクラスファイルが見つからないためです。

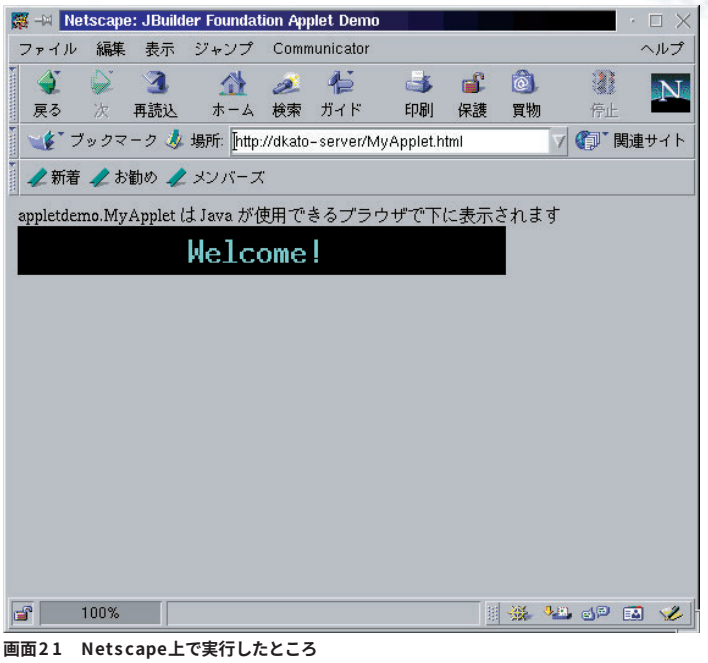
```
$ su
# cd/home/dkato/jbproject/appletDemo
# cp src/appletdemo/MyApplet.html/
home/httpd/html
# cp -r classes/appletdemo /home/
httpd/html
```

▼リスト6　MyApplet.html

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=EUC-JP">
<TITLE>
JBuilder Foundation Applet Demo
</TITLE>
</HEAD>
<BODY>
appletdemo.MyApplet は Java が使用できるブラウザで下に表示されます
<BR>
<APPLET
  CODEBASE = "."
  CODE     = "appletdemo.MyApplet.class"
  NAME     = "TestApplet"
  WIDTH    = 400
  HEIGHT   = 40
  HSPACE   = 0
  VSPACE   = 0
  ALIGN    = middle
>
</APPLET>
</BODY>
</HTML>
```



画面20　トップのindex.html



画面21　Netscape上で実行したところ

ファイルのコピーが終了したら、URLに＜サーバ名＞/MyApplet.htmlを入力してアクセスしてみましょう。Javaが起動され、Netscape上でアプレットが起動されます（**画面21**）。

まとめ

アプレットはWebサイトの表現力を高める有効な手段です。また、アプレットから他のJavaプログラムを呼び出すことでさまざまな処理を行うことも可能です。アプレットを作成するときはダウンロードするサイズをできるだけ小さくすることと、JDK 1.1ベースで動作することを確認することが重要です。JBuilder 3.5 Foundationで使用しているJDKが 1.2.2なので、JDK 1.1ベースの開発は多少面倒ですが、AWTベースで作成している限りそれほど問題はありません。

今回はできるだけシンプルに作成できるように文字列を点滅させる処理のためにラベルを使用しましたが、本来文字列を点滅したりする場合は、イメージバッファというものを使用して行います。インターネット上にはたくさんのアプレットのサンプルがあります。また、書籍も数多く出版されていますので、それらの書籍を利用してJavaの持っている可能性の深さを理解していただきたいと思います。

なお、今回作成したアプレットのソースコードが付録CD-ROMに収録されています。ソースコードの入力が面倒な方や、うまく動かない方は付録CD-ROMに収録されているソースコードを利用してください。

CPU	Intel Pentium II 233MHz以上
メモリ	128Mバイト以上
ハードディスク	150Mバイト以上の空き容量
モニタ	SVGA以上(256色以上)
OS	LASER 5 Linux 6.0 Red Hat Linux 6.1 日本語版
その他	CD-ROM、マウスなどのポインティングデバイス

表2　JBuilder 3.5 Foundationの動作環境

Column

JBuilderとJDKの入手方法

JBuilder 3.5 Foundation日本語版とJDK（Java 2 SDK Standard Edition version 1.2.2）は先月号のCD-ROMに収録されています。詳しいインストール方法も記載されていますのであわせてご覧ください。なお、JBuilder 3.5 Foundation日本語版はインプライズのWebサイトから、JDKはSun MicrosystemsのWebサイトからダウンロードすることもできます。WebサイトのURLについては、表3を参照してください。ただし、サイズが非常に大きいので注意してください。また、JBuilder 3.5 Foundationを使用するためにはライセンスキーも必要になります。ライセンスキーはインプライズのWebサイト（JBuilder 3.5 Foundationの入手先と同じ）で登録することで入手できます。Webサイトで必要な情報を記入すると、記載したメールアドレスにインストールに必要なインストール番号とインストールキーが送付されます。

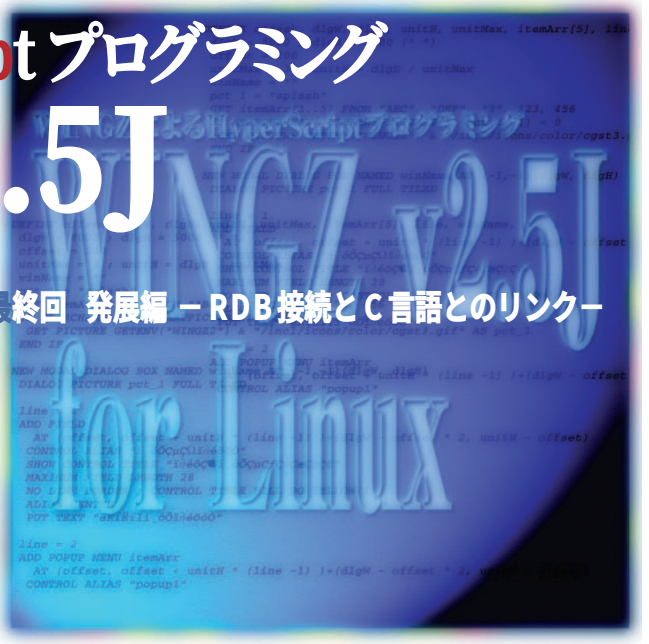
Jbuilder 3.5の入手先	http://www.inprise.co.jp/jbuilder/foundation/download/
Java 2 JDKの入手先	http://java.sun.com/products/jdk/1.2/ja/download-linux.html

表3　JBuilderとJDKの入手方法

WINGZによるHyperScriptプログラミング

WINGZ v2.5J for Linux

最終回　発展編 ー RDB接続とC言語とのリンクー



WINGZは、スプレッドシートをベースとした簡易GUI構築を実現するアプリケーションです。この連載ではWINGZ v2.5J for Linuxの体験版をもとに、WINGZのHyperScriptを使ったプログラミングを紹介しています。今回は、HyperScriptプログラミングの発展例として、DataLink機能を使ったRDBへの問い合わせと、C言語を使ってWINGZにアドイン機能を追加する外部ツールについて解説します。

文：株式会社アイフォー　久米 繁之
Text：i4 CORPORATION　Shigeyuki Kume

前回までの復習

第3回の内容に入る前に、これまでの連載内容を簡単に復習しておきましょう。

■WINGZについて

WINGZはスプレッドシート（表計算）機能をベースとしたソフトウェアで（**画面1**）、搭載しているHyperScript言語を使ってGUI（グラフィカルユーザーインターフェイス）ベースのアプリケーションを作成することができます。また、アドインのDataLink機能を使ったリレーショナルデータベースとの接続や問い合わせ処理も可能なため、社内業務システム、データベースのフロントエンドなどとして幅広く利用されています。

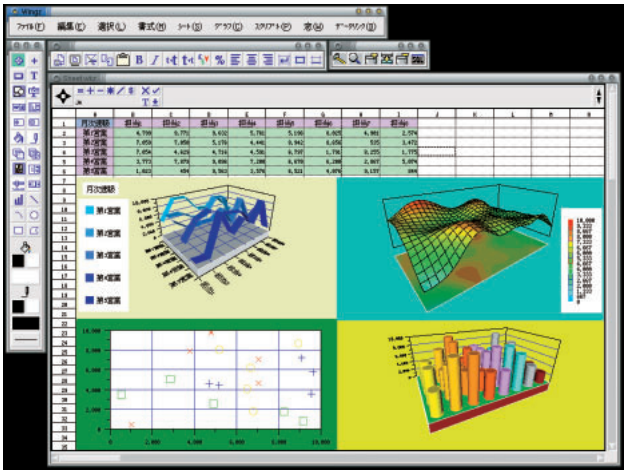
■HyperScriptについて

HyperScriptはBASICに似た言語形態のインタープリタです。実行環境としてはWINGZ自身が必要となりますが、インタープリタであるため試作～実行のサイクルが短くてすむなど、一般のツールキットよりも比較的気軽にプログラミングすることができます。

HyperScriptで扱う変数には基本的に型の制限がなく、255バイトまで格納できる変数、3次元まで定義できる配列変数を、グローバル変数やユーザー関数内のローカル変数として使い分けることができます。またワークシートのセルの座標や範囲、セル範囲名を扱うプログラミングも可能です。

◇スクリプトの実行形態

HyperScriptの実行方法としては、スクリプトエディタに記述しておいたスクリプトを直接実行する方法、ワークシートのエントリーバーに入力した文字列をHyperScriptとしてダイレクトに実行する方法、対象（ワークシート、ダイアログボックス、コントロール）に処理とその実行のタイミング（イベント）をスクリプトで記述しておくことで、指定されたイベントが行われた際に記述した処理を実



画面1　スプレッドシート機能をベースとしたWINGZ

行させる方法があります。もちろん、それぞれのスクリプトの中で、条件分岐や変数、ユーザー定義関数の定義を行うことができ、ほかのスクリプトから変数を参照したり、ユーザー関数の呼び出しを行うことも可能です。

◇イベント処理

「マウスが押された」、「ウィンドウのサイズが変わった」……などといった場合に、それぞれの動きに対応した処理が行われることをイベントドリブンな処理といいます。HyperScriptでは、対象となるコントロールやウィンドウに対し、イベントハンドラと呼ぶ文節にスクリプトを記述することでイベント処理を行います。

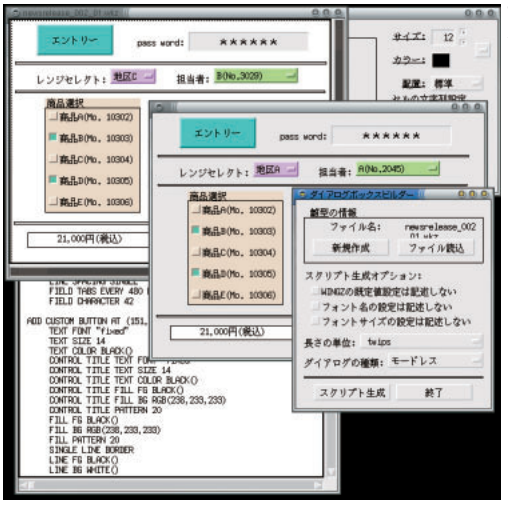
◇ダイアログボックスビルダー

WINGZでは、もちろんスクリプトの記述のみでも、コントロールを作成できますが、スクリプトを意識せずにコントロールをGUIベースで作成できるダイアログボックスビルダー（**画面2**）もバンドルしています。

スプレッドシート、ダイアログボックス、コントロールなどを組み合わせた処理を比較的簡単に作成できるのがHyperScriptの特徴です。WINGZはもちろんスプレッドシート（表計算）アプリケーションとしても十分な機能を備えていますが、この連載では表題のとおりHyperScriptのプログラミングを主題に解説しています。

復習の最後として、HyperScriptで作成したサンプルをいくつかご紹介します。スプレッドシートという枠を超えたWINGZの可能性を感じていただけるとさいわいです。

●WINGZでLifeGame（sample3_01.scz、画面3左）



画面2
ダイアログボックスビルダー

●WINGZでフラクタル（sample3_02.wkz、画面3右）
●WINGZでパズル
●html形式データの出力

■体験版について

本号の付録CD-ROMに、WINGZ v2.5J for Linuxの体験版を収録していますのでご利用ください。体験版はインストールから3カ月間、使用可能です。ただし、最終使用期限を2000年11月末日までとさせていただきます。インストール手順や動作環境については、CD-ROMに添付のドキュメンを参照してください。

体験版には、WINGZの関数／コマンドリファレンス、DataLinkのユーザーズマニュアル（いずれもPDF形式。体験版をインストールしたディレクトリに作成されるmanualディレクトリにあります）も付属しています。本文中のサンプルスクリプトで使用している関数の詳細については、こちらのマニュアルを参照してください。

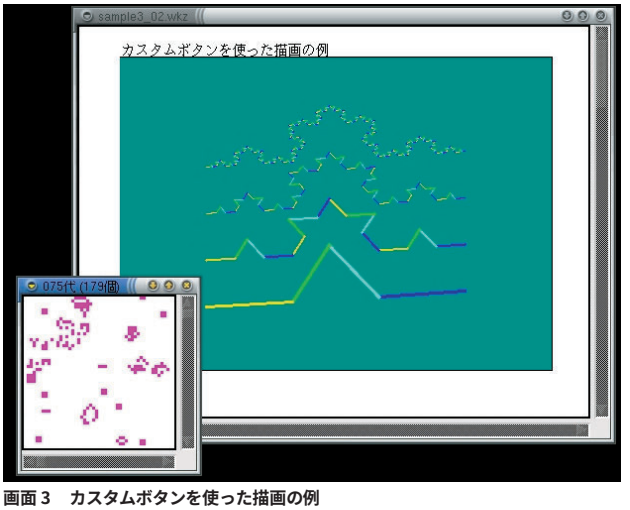
編集部からのお知らせ

CD-ROM作成の日程の都合上、本号のサンプルスクリプトは付録CD-ROMに収録されていません。筆者の久米氏および株式会社アイフォーのご好意により、Webサイトで公開させていただいておりますので、お手数ですが下記のURLからダウンロードしてご覧ください。

URL http://www.ifour.co.jp/wingz/

DataLinkの概要

WINGZは、後述する外部関数という機構を使って、機能を追加、拡張することができます。DataLinkは、



画面3　カスタムボタンを使った描画の例

WINGZとリレーショナルデータベース（RDB）との間でデータのやり取りを行うための外部関数（DataLink関数）と、それを使ったダイアログボックスなどのインターフェイス部をまとめたものです。

DataLink機能を使うことで、WINGZをRDBのフロントエンドとして使用することが可能になります。DataLink関数にはRDBにログオンする関数から、組込みSQL文の発行、エラーステータス確認、ワークシートへのデータの一括出力の指定、配列や変数へのデータの格納／データの更新削除を行う関数など、RDBとのやり取りに必要となる関数が用意されています。このためDataLink関数を使ったHyperScriptプログラミングでは、動的な問い合わせの実現や、ストアドプロシージャのトリガ発行なども行えます。

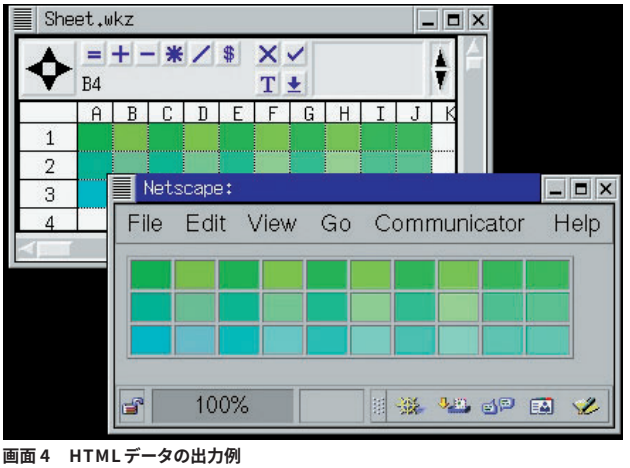
ここでは、DataLinkのユーザーインターフェイス部の解説とDataLinkダイアログボックスを使った問い合わせの実際、DataLink関数を使ったプログラミングについて解説します。

■体験版での制限事項

体験版ではInformix社のRDB製品（v7.3以降）との接続が行えるバイナリを用意しました。製品版では、Oracle社のサーバーにつながるバイナリも、バンドルされます。

■接続サーバについて

INFORMIX DynamicServer for Linuxのトライアルバージョンが公開されています（http://www.informix.com/jp/を参照してください）。今回のサンプルでは、RDBへの接続はローカルで行っています。サンプルには、サーバ製品にバンドルされているデモンストレーション用データベースstores7を使っているものがありますので、必要に応じてこのデータベースを作成しておいてください。



画面4　HTMLデータの出力例

■DataLinkのための環境設定について

WINGZインストール先のbin/ディレクトリに、WINGZとDataLinkを起動するシェルスクリプトdlを用意しています。同ファイルをホームディレクトリにコピーするなどしたうえで、環境を設定してお使いください。同ファイルで記述している環境変数の意味と設定個所は次のとおりです。

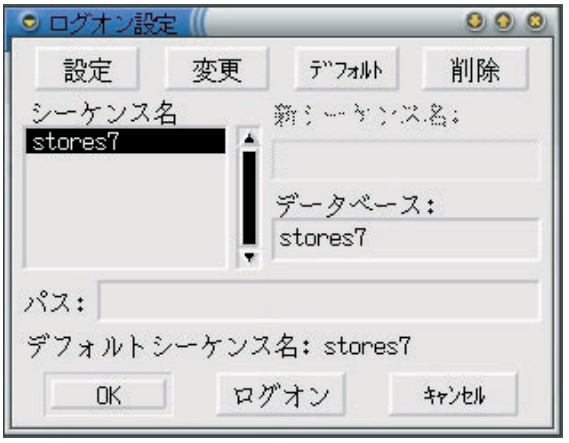
- **INFORMIXDIR：INFORMIXインストール先ディレクトリ**
- **INFORMIXSERVER：サーバの指定**
- **DB_LOCALE：接続先データベースで運用する言語ロケールの指定（オプション）**
- **DBLANG：クライアントアプリケーションに表示するINFORMIXのメッセージの言語ロケール（オプション）**
- **CLIENT_LOCALE：クライアントアプリケーションで使用する言語ロケールの指定（WINGZではja_jp.sjis-sを指定。オプション）**
- **WINGZ2：WINGZのインストール先ディレクトリ**
- **WINGZDL：DataLinkのインストール先（"$WINGZ2"を指定）**
- **DATALINK：DataLinkが作成する一次ファイルの作成ディレクトリを指定（オプション）**

■DataLinkの起動

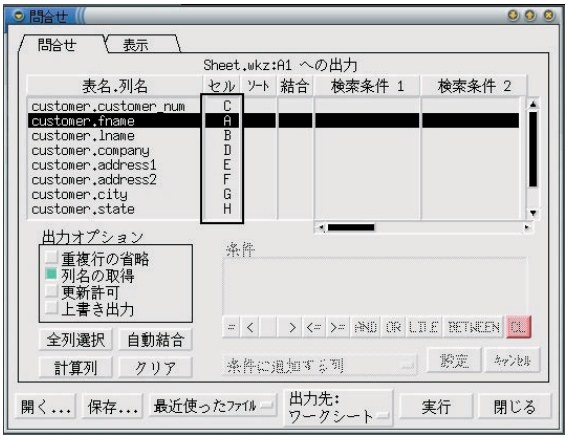
DataLinkの環境設定が有効なコンソールから、WINGZを起動し、WINGZのインストール先にあるスクリプトDLRun.sczを実行します（**画面5**）。あるいは、以下のラインコマンドで直接DataLinkの起動まで行うこともできます。

```
$WINGZ2/bin/Wingz $WINGZDL/DataLink
```

画面5　DataLinkの起動画面



画面6
ログオンダイアログボックス



画面7
問い合わせダイアログボックス

DataLink既存インターフェイスの利用方法

それでは、DataLinkを使ったRDBへの接続と問い合わせの実際を見ていきましょう。

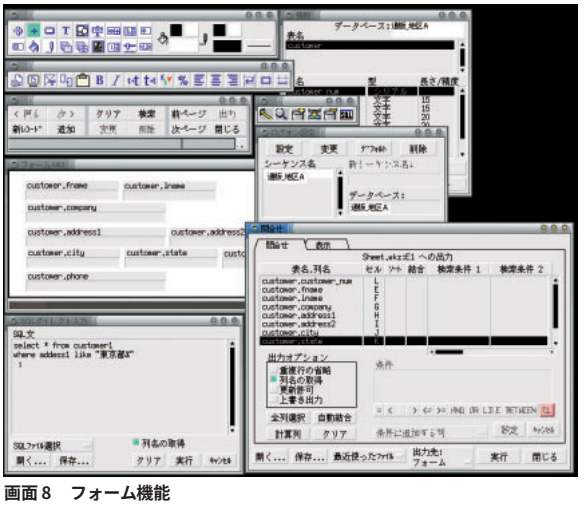
◇データベースへのログオン

WINGZの［データリンク］メニューからログオンを選ぶと、接続先データベースの設定を行うダイアログボックス（**画面6**）が表示されます。このダイアログボックスには、データベース名や、必要に応じてデータベースパスを指定し、その組み合わせに対する名称を設定します。一度設定した名称は、［データリンク］メニューの［ログオン］サブメニューに登録され、次回以降はメニューから選ぶだけでログオンまで行われるようになります。

◇問い合わせ（ワークシートへの出力例）

WINGZの［データリンク］メニューの［問合せ］―［問合せ実行］を選ぶと、問い合わせのダイアログボックス（**画面7**）が表示されます。ダイアログボックスには、接続先データベースの有効な表と列の情報が表示されます。それぞれの目的に合わせて、条件や出力先を指定することで、SQL文をいっさい意識せずにデータベースへのデータの問い合わせを行うことができます。

また、ダイアログボックスで指定した内容と同義のSQL文を発行したり、問い合わせ用のフォームを自動生成することも可能です。



画面8　フォーム機能

◇フォーム機能

接続先データベースの中から、利用したい表と列を指定して、簡易検索フォームを作成することができます。作成したフォームは、フォームパレットを使って操作します。フォーム機能（**画面8**）は、簡易データベースフロントエンドの作成を支援します。

◇その他のダイアログボックスの紹介

その他の特徴的なダイアログボックスを紹介しておきます。これらのダイアログボックスを使って、DataLinkの既存インターフェイスだけでもRDBとのやり取りをひととおり行うことが可能です。利用できる部分はできるだけDataLinkの既存インターフェイスを利用し、カスタマイズする必要のある部分のみをDataLink関数を使って作り込むなどすることで、RDBフロントエンド構築のための工数を削減できます。

- **［データベース情報の表示］ダイアログボックス**
  接続先データベースを切り替えます（**画面9**、**画面10**）。
- **［表・列の編集］ダイアログボックス**
  既存の表や列の設定を変更できます（ログオンユーザーが

画面9
データベース選択ダイアログボックス

画面10
データベース情報の表示

変更の権限を持つ場合に限ります。**画面11**)。

●［SQLダイレクト実行］ダイアログボックス

SQL文を直接ダイアログボックスに書き込んで発行することもできます。このダイアログボックスは主にメンテナンス時に使用します。

WINGZで扱えるコントロール

次に、DataLink関数を使って直接スクリプトでRDBとのやり取りを行う例を紹介します。DataLink関数の使用はもちろん、DataLinkの既存インターフェイス機能と組み合わせて使用することもできます。

■ハイレベル関数とローレベル関数

DataLink関数は動作上、ハイレベル関数とローレベル関数の2つに分類されます。ハイレベル関数は、エラーチェックをDataLinkが管理している関数で、エラー時のインターフェイスなども用意されています。 たとえばWINGZからRDBにデータの問い合わせを行った際に、該当するデータがないなどのエラーになった場合には、必要に応じて所定のダイアログボックスが表示され、ユーザーが対応を指示できます。またハイレベル関数では戻り値も設定されています。

一方ローレベル関数は、DataLinkによるユーザーインターフェイスやエラーチェックはありません。ステータスや状況に対する対応は、すべてユーザーが設計、管理することになります。このようにローレベル関数は、より単純な処理に特化した個別関数となっていますが、その分ローレベル関数を組み合わせることで、用途に合わせた独自関数を作成することができます。実際には、ハイレベル関数についても、ローレベル関数とHyperScript関数を組み合わせることで実装されています。

■DataLinkハイレベル関数を使った問い合わせの例

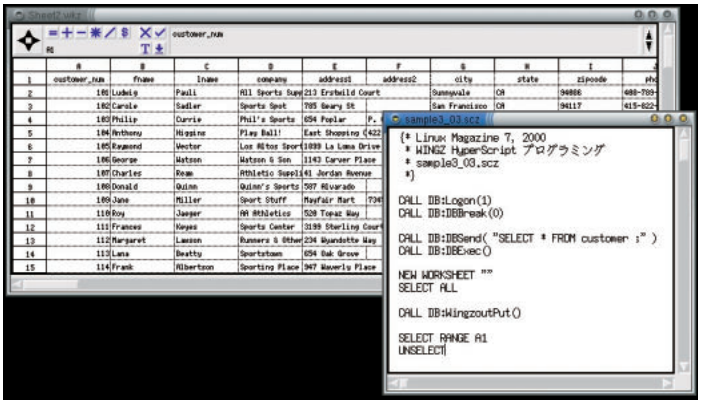
sample3_03.scz

```
CALL DB:Logon(1)
CALL DB:DBBreak(0)

CALL DB:DBSend( "SELECT * FROM customer ;" )
CALL DB:DBExec

NEW WORKSHEET ""
SELECT ALL

CALL DB:Wingzoutput()

SELECT RANGE A1
UNSELECT
```

この例（sample3_03.scz）では、ハイレベル関数WINGZOUTPUT( )を使った問い合わせを行っています。DBSend( )、DBExec( )で発行したSQL文をワークシートに出力するところで、WINGZOUTPUT( )を使用しています（**画面12**)。

このスクリプトでは、事前に新規ワークシートを開いて、SELECT ALLを発行していますので、該当データが自動的に出力されますが、たとえば選択範囲を限っている場合などでは、出力するデータの数と選択中のセルの数が異なっていると、選択範囲外にもデータを出力してもよいかを確認するダイアログが表示されます。このように、ハイレベル関数を使用すると、DataLink側で状態を判断して適切なダイアログを表示するなどの対応がとられます。逆にダイアログボックスの表示といった、いっさいの割り込みを行いたくない場合は、ローレベル関数を使用します。

画面11 表と列の編集ダイアログボックス



画面12 WINGZOUTPUT( )の実行結果

◇問い合わせの初期化

問い合わせ状態の初期化は、DB:DBBreak(0)関数をCALLして行います。問い合わせ文を発行する前に、必ず問い合わせの初期化を行ってから、問い合わせ文を発行する必要があります。

◇ステータスの確認

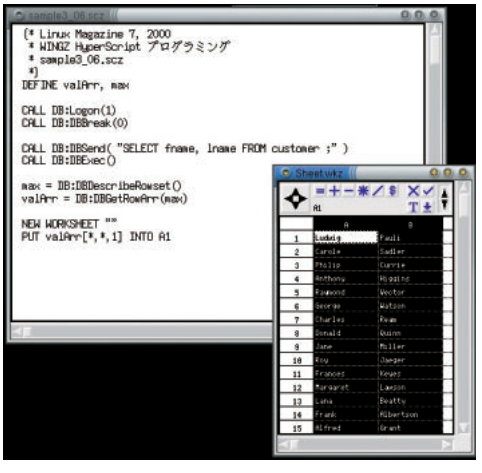
DataLink関数の中には戻り値を返すものもありますが、この関数の戻り値は、関数を発行した時点でのDataLinkの状況を返す値がほとんどであることに注意をしてください。DataLink関数を発行した結果データベースの状況については、 DB:DBState( )関数を発行し、この関数の戻り値を確認することで行います。

◇その他のハイレベル関数

sample3_04.scz では、列を追加するハイレベル関数ADDROWS( )を使用しています。この関数は、ワークシートセルに追加する列名を記述しておき、その下の領域に追加するデータを配置します。そして、設定した列名とデータを選択した状態で関数を発行することで、データを追加しています。

```
PUT "fname" INTO A1
PUT "lname" INTO B1

PUT "Linux" INTO A2
PUT "Magazine" INTO B2

SELECT RANGE A1..B2
CALL DB:ADDROWS()
```

データ型が違うなどのエラー時には、DataLinkが用意しているダイアログボックスが表示され、処理を促します。



画面13 sample3_06.sczの実行例

■DataLinkローレベル関数を使った問い合わせの例

sample3_05.sczでは、ローレベル関数の GETVAL( )を使って、問い合わせたデータを取り出しています。該当するデータがまだ残っているか否かは、DBGETROW( )の状態を確認することで行います。処理によっては、この例のように逐一データを変数に格納する方法も選べますが、結果セットを配列に格納するローレベル関数GETROWARR( )を使うと、問い合わせ結果とそのステータスを一気に処理できるので便利です。sample3_06.sczでは、配列を使った結果セットの格納例を記述しています(**画面13**)。

■DataLinkアプリケーションの例

このようにDataLinkではRDBへの問い合わせをWINGZから直接行える機能を提供しています。WINGZ本体のHyperScriptの機能と組み合わせることで、データベースのGUIベースのフロントエンドを構築したり、問い合わせた結果をグラフ化するなどして、結果をHTML形

式で定期的に出力、保存するといった定型業務の自動化を行うことも可能です（sample3_07.scz、**画面14**）。

外部関数

システムコール、リソースへの直接アクセス、独自モジュールとのデータ交換などといった、WINGZ本体には用意されてない機能については、Cによるモジュールを WINGZの外部関数として使用することで、機能拡張できます。外部関数では引数や戻り値を利用して、WINGZ本体とCの関数の間でデータのやり取りを行うことが可能です。前述のDataLink機能も、WINGZの外部関数を使った機能拡張のひとつです。

HyperScriptプログラミングの連載の最後の項目として、外部関数によるWINGZの機能拡張について解説します。なおサンプルは、Linuxのシステム情報取得などの基本関数のみを使用するものとし、WINGZとCのmake環境があれば作成な可能なようにしています。簡単な内容ですのでぜひお試しください。

■WINGZの外部ツール

WINGZに機能追加するためにCで作成したモジュールの実行形式ファイルを「WINGZの外部ツール」と呼んでいます。この外部ツールの中に、複数の外部関数を定義することができます。

GET EXTERNALコマンドでメモリに外部ツールを組み込むことで、外部関数は実行が可能になります。用途に応じて、それに関連する外部関数を同一の外部ツールにまとめておくと、メンテナンスなどの面で有効です。

■実習1－数値（UserID）を戻す外部関数

では、実際に外部ツールを作成し、WINGZから外部関数を呼び出してみましょう（sample3_08）。

外部関数で扱えるデータ型は数値、文字列、配列、およびエラータイプ（WINGZへ戻す外部関数のエラーコード）です。この中では数値が一番単純な扱いとなるでしょう。

次の外部ツールtest1には、外部関数が1個含まれています。外部関数の名前はuserid( )で、引数の指定はなく、WINGZを起動しているユーザーのユーザーIDをOSから取得し、単純に数値で戻すだけの機能とします（**画面15**）。なおサンプルでは、関数内でのエラーチェックについては、単純化のために省略しています（オンラインマニュアルによると、使用しているgetuid( )関数は必ず成功する仕様でありますし:-） Cf. man getuid）。

◇make手順（外部ツールと外部関数）

Makefileを用意していますので、環境が合っていればmakeだけで外部ツールが作成できます。必要に応じてMakefileを変更して、外部ツールtest1を作成してください。

```
# make
# ls test1*
test1*    test1.c
```

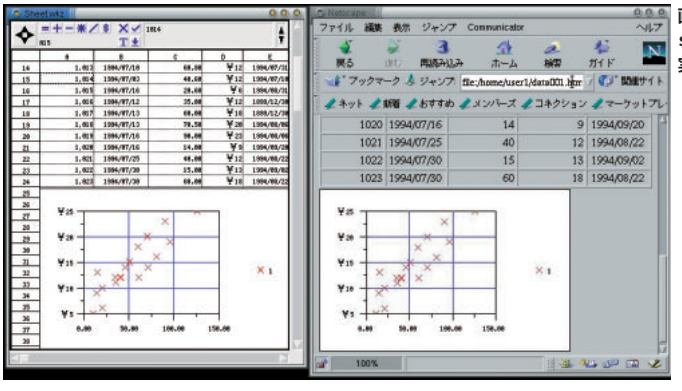
Makefileではなく、以下の手順でも作成できます（環境変数WINGZ2にはWINGZのインストール先ディレクトリを指定しておきます）。

```
# cc test1.c -L$WINGZ2/lib -lwztools -I$WINGZ2/incl
-o test1
```

◇WINGZからの呼び出す手順

まずは、作成した外部関数を実行してみましょう。

外部関数を実行するには、まず、外部ツールをGET EXTERNALコマンドでメモリに格納してWINGZから呼び出し可能な状態にします。外部ツールがGETできたら、



画面14 sample3_07.sczの実行例

画面15 sample3_08で表示されるメッセージダイアログ

MESSAGEコマンドを使って、外部関数を呼び出します。あとは必要に応じて、REMOVE EXTERNALコマンドを用いて、メモリから外部ツールを削除します。

外部ツールは一般のHyperScriptコマンドと異なり、ディレクトリとファイル名を指定してGETする必要があります。このサンプルでは単純にフルパスで指定していますが、実際にアプリケーションを作成する際には、HyperScriptで事前に外部ツールのファイルが存在していることをチェックしてからGETするようにしましょう。サンプルでは、実行中のスクリプト自身の情報を戻す関数CURRENTSCRIPTNAME( )を使っています。

◇外部関数の機能記述部分

外部ツールtest1と外部関数userid( )の内容を見ていきましょう。

test1.c

```
#define numfuncs 1

#include <pwd.h>
#include "WZTools.h"

extern void    xfTest1();
extern void    ExitFunc();

ROUT rout =
    {
        numfuncs,
        ExitFunc,
        {
            {xfTest1, "\006userid", 0}
        }
    };

WZMAINFUNC()
{
    return(&rout);
}

void xfTest1(pret)
PVAL    pret;
{
        pret->flag = NUMERIC;
        pret->val.numeric = getuid();
    return;
}

WZFUNC ExitFunc()
{
    return;
}
```

◇外部関数の構造とCの書式について

外部ツール、外部関数が一般のCの書式と異なる点は以下のとおりです。

●メイン関数はWZMAINFUNC( )
●外部関数の実体のC関数の型はextern voidを指定
●WINGZとの引数、戻り値のやり取りはVAL構造体を使用
●外部ツール自身の定義はROUTE構造体で行う
●個々の外部関数の定義はRERUT構造体で行う
●WINGZでは文字列はパスカル文字列を使用

◇メイン関数WZMAINFUNC( )

外部ツールでは、WZMAINFUNC( )がメイン関数の役割を担います。 WZMAINFUNC( )では、外部ツール自身の定義を行っているROUTE構造体を引数に指定することで、外部ツール自身の初期化が行われます。

◇C関数の定義について

外部関数の実体であるCの関数は、戻り値を持たないextern voidで指定します。WINGZとのデータの交換は、後述するVAL構造体を使って行います。

◇ROUTE構造体

ROUTE構造体で外部ツール自体の定義を行います。構造体のメンバーは次の通りです。

●nrout：外部ツールに含まれる外部関数の総数
●exitfunc：exit時のクリーンナップ処理など
●relt：このROUTE構造体で個々の外部関数の定義を行う

◇RERUT構造体

RERUT構造体で個々の外部関数の定義をします。構造体のメンバーは次の通りです。

●pfunc：該当する外部ツール内のC関数名の指定を行う
●name：WINGZから呼び出す際の関数名を定義する。関数名はパスカル文字列（先頭に文字長を格納）を定義する
●narg：WINGZから呼び出す際の引数の個数を定義する

◇VAL構造体

WINGZと外部ツールとの間でデータのやり取りを行うために用意されているのがVAL構造体です。WINGZから外部関数へ引数を渡す場合も、外部関数からWINGZへ戻り値を渡す際もすべてVAL構造体（を示すポインタ）を使用します。

VAL構造体では、数値、パスカル文字列、配列、エラー値が扱えます。VAL構造体には、どのデータ型を使用するかの指定をするメンバーflagと、型ごとに値を格納する共用体valが用意されています。VAL構造体の定義は、$WINGZ2/incl/WZTools.hで確認できます。

外部関数userid( )（C関数名xfTest1( )）では、数値であるユーザーIDをWINGZに戻していますので、戻り値が数値であることをflagに格納して、valの共用体のうち、数値を格納するval.numericに値を格納することで、機能を実装しています。

```
pret->flag = NUMERIC;
pret->val.numeric = getuid();
```

なお、外部関数の実体のC関数では、第1引数がWINGZへの戻り値、第2引数がWINGZから外部関数へ与えられる引数を格納したVAL構造体のデータになります。

■実習2－文字列を戻す外部関数

次に、外部関数で文字列を扱ってみましょう（sample3_09.scz、**画面16**）。前述のとおり、VAL構造体で扱える文字列は先頭に文字数が格納されたパスカル文字列であるため、通常のCの文字列をそのままVAL構造体で扱うことはできません。用途によってCの文字列とパスカル文字列を変換して使用します。WINGZでは、C文字列からパスカル文字列へ、またはその逆の変換を行うなどといった外部関数作成のためのプロトタイプ関数を用意しています。プロトタイプについての解説は、$WINGZ2/manual/wz_external_prototype.txtにありますので参照してください。この例では、C文字列からパスカル文字列への変換を行うプロトタイプ関数c2pcopy( )を使用しています。

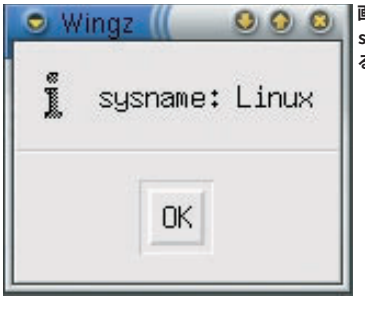
◇外部関数でのエラー処理

外部ツールtest2には、外部関数info1( )が含まれます。info1( )のC関数xfTest( )には、簡単なエラー処理も加えています。この例では、uname( )の実行が行えなかった場合、WINGZへエラーが戻る仕様になっています。

WINGZの外部関数におけるエラー処理は、戻り値用のVAL構造体のメンバーflagにエラーを設定し、メンバーvalの共用体errに外部関数用のエラー番号を格納することで実現しています。ここで使用しているNOMEMはメモリ不足を表すエラー番号です。エラー番号の定義は、$WINGZ2/incl/WZTools.hで行っています。

test2.c

```
void xfTest(pret, parg)
PVAL    pret;
PVAL    parg;
{
    struct utsname uts;

    if (uname(&uts) != 0)
    {
        pret->flag = ERR;
        pret->val.err = NOMEM;
    }
    else
    {
        pret->flag = STRING;
        c2pcopy(uts.sysname,pret->val.string);
    }

    return;
}
```



画面16
sample3_09/test2で表示されるメッセージダイアログ

◇実習3―配列でデータを戻す外部関数

次は、配列を戻す外部関数の例です（sample3_10.scz）。外部ツールtest3には外部関数info2( )が含まれます。WINGZからinfo2( )を実行すると、ホスト情報などの文字列を配列で戻します（**画面17**）。

外部関数info2( )のC関数xfTest( )の中では、まず外部関数のプロトタイプxfNewArray( )を使って配列の領域を設定しています。領域が確保できたら、配列の個々の要素を格納していきます。要素ごとにデータ型を変えることも可能ですが、この例では、tmpVal.flagにはすべてSTRINGを格納し、文字列情報を戻すようにしています。Cでは配列の最初の要素は0と数えますが、HyperScriptの配列要素は1から始まる点に注意してください。

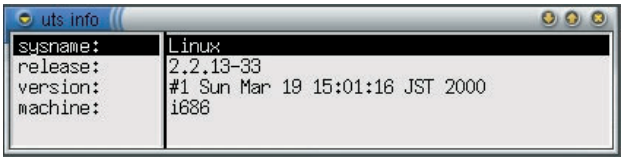
◇実習4－WINGZから引数を渡してアプリケーションを起動

これまでは、WINGZからは引数を与えずに、システム情報などを外部関数からWINGZへ戻す例をあげてきました。今回は逆に、WINGZから引数を与える例と、与えられた文字列を使って、所定のアプリケーションを外部関数から起動する例を示します。

外部ツールtest4には外部関数exec( )が格納されていて、起動するアプリケーション名を文字列で与える仕様になっています（**画面18**）。外部関数info2( )のC関数xfTest( )はVALで指定された引数を2つ持ち、2つ目の引数がWINGZから外部関数に与えられる引数になります。外部関数の引数は0～50個まで設定できますが、この例のように引数を1つ持つ関数の場合は、VALの0番目の要素がそれに相当します。

また、文字列の扱いについても、前述のとおりWINGZ内部ではパスカル文字列を使用しているため、外部関数のプロトタイプp2ccopy( )を使って事前にC文字列へ変換しています。

```
char   execname[64];

p2ccopy(parg[0].val.string, execname);
system(execname);
```



画面17　sample3_10/test3で表示されるメッセージダイアログ

このように、外部関数を使うことでシステム情報をWINGZで直接取り扱ったり、他のアプリケーションをWINGZから制御したりすることが可能になります。ここではごく簡単な例を使って外部関数の概要を紹介しましたが、外部関数で扱える情報の取扱いについて、ひととおりおわかりいただけたのではないでしょうか。外部関数によるオリジナルの機能をWINGZに付加してみてください。

最後に

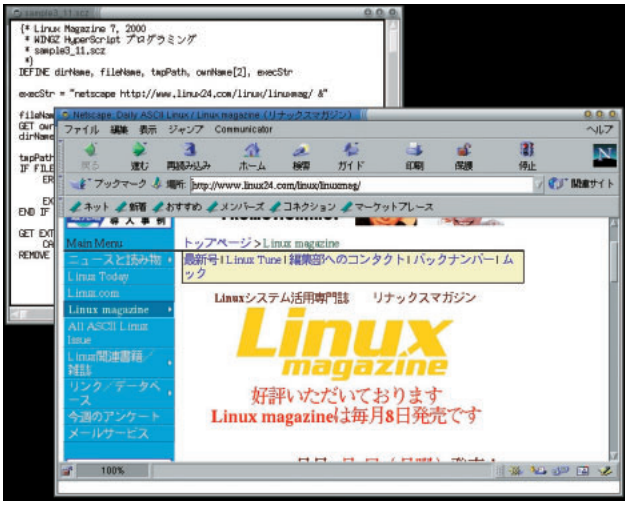
3回にわたって連載してきたHyperScriptプログラミング、如何だったでしょうか？できるだけ簡単な例によるイントロダクションを記述するよう心がけてきましたが、WINGZをご存知の読者にはやや易しい内容だったかもしれません。また機会がありましたら、さらに応用編などにも取り組みたいと思います。ご意見、ご感想をお寄せいただけますと幸いです。ありがとうございました。

お知らせ

これまで体験版として付録CD-ROMに収録してきたWINGZ for Linuxの発売が決定しました。GUIベースのアプリケーションの簡易作成やリレーショナルデータベースのフロントエンドとしての利用など、ビジネスシーンでLinuxのクライアントデスクトップ環境の利用を大きく推進します。

WINGZ v2.5J for Linux
発売日：2000年7月14日
価格：1万2000円（6カ月間のインストールサポート付）

問い合わせ先: 株式会社アイフォー 営業部
TEL　03-5436-7850
URL　http://www.ifour.co.jp/wingz/



画面18　sample3_11/test4の実行例

隠喩としてのコンピュータ

ソフトウェアはなぜ動物に隠喩されるのか？

文：豊福 剛
Text：Tsuyoshi Toyofuku

Linuxはなぜペンギンなのか

ペンギンはLinuxのシンボルである。正確には、Linuxのカーネルのシンボルである。それでは、なぜペンギンなのか、理由については、よくわからない。ペンギンは、Linuxカーネルそのものをシンボライズしているというよりは、むしろ、カーネルをオープンソースで開発するという、その開発の運動の質をシンボライズしているように思える。多数のペンギンが集まって、大集団をつくる。何十万羽にもおよび集団を形成することもあるらしい。そうした集団形成の性質は、オープンソース運動のバザール的な側面をシンボライズするのにふさわしいだろう。

また、Linuxカーネルそのものは、ペンギンが住んでいる南極大陸によってシンボライズさせることもできるだろう。南極大陸は、南極点を中心に広がる大陸である。南極点を地球のひとつの中心とみなすことができるように、カーネルもソフトウェアのひとつの中心とみなすことができる。もちろん、カーネルは点ではなく、さまざまな拡張性という広がりがあるのだから、その意味で、大陸的である。また、アルゼンチン、オーストラリア、チリ、フランス、イギリス、ニュージーランド、ノルウェーは、南極大陸の領有を主張しているが、1961年の南極条約によって、これらの主張は保留されていて、国際協力による科学研究が進められている。南極大陸が特定の国家の領土ではないのと同じように、Linuxカーネルも特定の企業の占有技術ではない。

このように、Linuxのシンボルがペンギンであることを、いろいろ解釈することはできるのだが、それだけでは何かが欠けている気がしてならない。このシンボルについては、2つの側面から考えることができる。ひとつは、ソフトウェアのシンボルに、ペンギンという動物が使われることの意味、もうひとつは、シンボルの表現のテイストについて、である。

オライリー本のカバーの意味

動物といえば、たとえばオライリー社の本のカバー・イラストがある。

1998年、ラリー・ウォールとティム・オライリーが来日したとき、池袋のジュンク堂という本屋さんで講演会があった。そのとき、オライリー氏にカバー・イラストで動物のイラストを使っている理由を質問してみた。オライリー氏によると、もともとデザイナーのアイデアであるらしい。それまで、オライリー社はコンピュータ関連のマニュアル

制作などを手掛けていたのだが、いかにもハイテクな感じのブックデザインに飽き飽きしていたこともあって、そういうのとは違うデザインを求めていたらしい。そのデザイナーには、UNIXのコマンドの名前の多くが、まるで動物の名前のように感じられたそうだ。動物のイラストは、それまでのデザインにない、独特の暖かい感じがあるということで、採用されたそうだ。

オライリー本のカバーでは、たとえばPerlのシンボルとして駱駝が使われている。ラリー・ウォールは、駱駝は（外見が）汚らしいけれども、働きもので、そういうところはPerl的だ、と説明している。プログラミング言語でいえば、最近ではPythonのシンボルに蛇が使われている。ギリシア神話で、アポローン神がデルポイ（デルファイ）で退治した大蛇の名前ピュートーンが、その語源になっている。

このほかにもたくさんの動物がカバーに使われているのだが、個人的には、『Sendmail』のカバーの蝙蝠（コウモリ）が気に入っている。蝙蝠といえば、吸血鬼ドラキュラが連想されるのだが、これについては後述する。

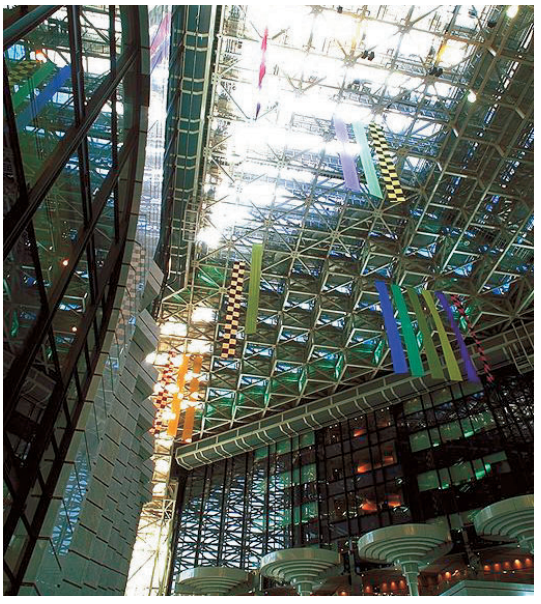
もちろん、オライリー本のカバーにかぎらず、それ以前から、コンピュータにおけるある種の擬似生物的な隠喩というのは、存在していた。たとえば、プログラムのバグという言い方や、コンピュータ・ウイルスという言い方などは、その代表例といえるだろう。

動物、あるいは生物的な隠喩が成立するのは、おそらくソフトウェアが複雑であることと関係しているのだろう。生命をある種の情報システムとして見る観点は、サイバネティクス以来のものだし、複雑系の科学にも継承されている。生命が情報システムとみなせるのであれば、その逆に、情報システムを擬似生命とみなしたとしても、それほど違和感がないだろう。

不気味なものと可愛いもの

Linuxで注目すべきは、そのシンボルがペンギンであることではなくて、そのペンギンがどのようなテイストで表現されているか、であるように思う。Linuxのペンギンは、オライリー本のカバーのような、写実的な表現ではなく、キャラクターとして表現されている。

こうしたシンボル・キャラクターに感じられるポップさは、Linuxだけでなく、BSDのデーモンくんの特徴でもある。キャラクターの可愛さという点では、Linuxのペンギンよりも BSDのデーモンくんのほうが、若干勝っている気がするが、いずれにせよ、どちらのキャラクターも、オラ



イリー本のカバーの動物よりは、むしろ、たとえばポストペットのMOMOに近いテイストであるように感じられる。

このようなテイストは、少なくとも、アップルやマイクロソフトのパソコン文化には存在しなかった。可愛いさを表現したシンボル・キャラクターの台頭は、コンピュータ文化における新感覚の台頭を感じさせる。この新感覚の背景には、いったいどのような変化があったのだろう。

東浩紀は『サイバースペースは何故スペースと呼ばれるか』で、情報技術の進展とともに「不気味なもの」が出現することの意味を考察している。この「不気味なもの」という概念は、フロイトによるもので、「生命か非生命か決定不可能な自動人形、分身（ドッペルゲンガー）、死者との対話、反復される謎めいたメッセージ、アニミズム的精霊」などが、これに該当する。これらは、情報技術以前から存在していたのだが、現在においては、テレプレゼンスの効果として、自己の分身化が実現されたり、あるいはビデオなどの記憶媒体によって、自己の幽霊化が実現されるなど、「不気味なもの」の様態は、常にテクノロジーの進展と密接に連動している。

こうした「不気味なもの」に深い関心を寄せていた作家として、東はフィリップ・K・ディックを参照している。東は、『アンドロイドは電気羊の夢を見るか？』の自動人形、『暗闇のスキャナー』の分身、『ユービック』の死者との対話、『銀河の壷直し』の謎めいたメッセージ、などなど、ディックの小説に「不気味なもの」のモチーフが頻出することを指摘したうえで、「不気味なもの」の描写を「可愛いもの」へと偏差させる戦略に長けていた点に注目し、また、ディックの世界における「可愛いもの」とは、「不気味さ＝擬似生命性」に対する感情的負荷が逆転したものとして説明している。

メディア理解の二重性

東が指摘する、ディックの作品における「感情的負荷の逆転」（ディック自身の表現では「感情移入」）について考察する前に、「不気味なもの」について、もうしばらく検討してみたい。

東は、電子メディアの本質に関する理解の仕方には、速度＝距離的なものと空間的なものの2種類があることを指摘する。

ハイデガーは30年代にラジオを「遠隔性の克服」を目指し、「世界の近さを保証する装置」と解釈していた。現実空間の距離性が、ラジオというメディアによって克服さ

れた点が強調されているのだ。このように「情報の移動とモノの移動を同じカテゴリーで思考する」観点を、東は「速度＝距離的メディア理解」と形容している。

『グーテンベルクの銀河系』を書いたマクルーハンは、電子メディアの発展によって、グローバル・ビレッジという地球規模の村が成立することを予見したが、このことを、現実空間としての世界が小さくなったと理解するとき、その理解は「速度＝距離的」といえる。

マクルーハンの「地球村」という概念には、ある神秘的な過剰さを含んでいる。マクルーハンは、地球規模の「集団意識」が形成され、人々は地球規模の「ミサ」に参加すると予見した。東は、マクルーハンのメディア論で語られている「精神圏」という概念に着目する。

――そこではもはやメディアは視点と終点（発信者と受信者）、二つの現実空間に挟まれた単なる時間経過ではない。両者のあいだには情報経路がある。そしてその経路のネットワークにある奇妙な「空間性」が宿る。

東は、このような神秘思想的なメディア理解を、「速度＝距離的」と対比して、「空間的」と形容する。ここでいわれている「空間」とは、現実空間ではなく、隠喩としての空間を意味する。こうした「空間」の隠喩が成立するのは、前回述べたように、電子メディアによる「不気味なもの」の効果が悪魔祓いされるかぎりにおいてである、と東は考察している。

ただし、メディアにおける「不気味なもの」の効果に注目する観点は、ポスト構造主義において、かなり共有された概念であるようだ。ジル・ドゥルーズ／フェリックス・ガタリは『カフカ』の中で、次のように書いている。

――カフカは二つの系列の技術的発明を区別する。ひとつは距離を克服し、人間たちを接近させることによって《自然な関係》を復活させようとする技術（汽車・自動車・飛行機）であり、もうひとつは、亡霊の吸血鬼的な復讐を表象するか、《人間のあいだにある亡霊的なもの》を再導入するような技術（郵便・電信・電話・無線電信）である。

ドゥルーズ＝ガタリは、文学機械／表現機械としてのカフカにおける「亡霊的」技術系例を代表するものとして、手紙に注目している。引用にある「吸血鬼」とは、ドラキュラをさしているのだが、蝙蝠がドラキュラの分身であるのと同じように、手紙はカフカにとって分身なのだ。

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。

オープンソース
・
アクティビズム

100億ドルの間抜けたちと ソフトウェアの遺伝子

文：安田幸弘
Text : Yukihiro Yasuda

やっぱり「愛」ってやつは、国境を越えたがるものなんだろうか。ちょっと前に「I LOVE YOU」というメールが世界を駆けめぐった。被害額は10億ドルとも100億ドルともいわれるが、小さなプログラムが一夜にして世界中にこれだけの衝撃を与えることができるのだからすごい。

幸か不幸か、ぼくのところにはこの有名なラブレターが1通も飛んでこないので、いまいちピンとこないのだが、要するに、「世界には100億ドルに値する間抜けの大集団が存在する」ということを証明した事件なんじゃないかと思っている。

うっかり被害にあった人にはお気の毒だけど、これが実生活の場面だったら、玄関のチャイムが鳴ってドアを開けたら強盗だった、というような話である。強盗が犯罪であることは言うまでもないが、しかし相手を確認せずにドアを開けた人にも過失がある。まして、マスコミでそんな強盗の話題が毎日のように騒がれていたとすれば、簡単にドアを開けてはいけないことぐらい誰にでもわかる。

ところがサイバーワールドでは、なぜか強盗を招き入れただけじゃなく、強盗に知り合いの家まで教えて送り出した間抜けの大群が、「被害にあった、どうしてくれよう!」と大騒ぎをしている。その上、亜種・変種の登場で、二度も三度も同じ手口にひっかかる間抜けがいるらしい。こうなると、絶対、ひっかかる方が悪い。どうして誰も「あんたも加害者なんだよ」と言わないんだろう?

フィリピン人は痛くもかゆくもない

7月に開かれる九州・沖縄サミットの大きなテーマのひとつが情報分野なんだそうだ。ここでは、IT革命に伴う格差の解消、途上国の支援、そしてネットワークの不正な利用やハッカーなど国境を越えた犯罪の防止などが取り上げられる予定だという。

インターネットのメリットはいうまでもないが、問題はこれが先進国に有利に進められているということ。しかしその先進国では、今回のワームのように無防備に広がったネットワークの脆弱性が問題にもなっている。

今回、FBIが見つけ出したI LOVE YOUワームの容疑者とやらが、途上国フィリピンの人間だったそうだが、少なからぬフィリピン人は、自国のイタズラ小僧が先進国の100億ドルの間抜け集団をきりきり舞いさせたことに、内心「ざまみろ」と思っているらしい。フィリピンのコンピュータ普及率は全人口の1%以下。どんなに強力なワームがネットを駆けめぐっても、99%以上のフィリピン人は痛くもかゆくもないわけだ。

もっとも、インターネットのグローバル化は、フィリピンでもそれなりに進んでいるらしい。フィリピンのあるNGOのスタッフから聞いた話だけれど、自分でパソコンを持っていなくても、無料メールサービスにアドレスを持っていて、自転車で週に1回、パソコンを使わせてくれるセンターに出かけ、外国に出稼ぎに行った家族とメールのやりとりをしている人もいるという。毎日のようにxDSLだの超高速ネットだのという話を聞き、どうでもいいようなメールを山のように受け取っていると、自転車で週1回ぐらいのペースでメールをチェックしに行くぐらいのほうが、人間的でいいなあと思ったりもする。

FBIのパフォーマンス

話はちょっと横道にそれるけど、今回のワームは、米国のFBIがILETSという枠組みの中で、何年も前から日本やヨーロッパの捜査当局を相手に繰り広げてきたネットワークの規制の根拠としては、格好の事件だったのだろう。マスコミを通じて「被害」が大げさに語られ、第三世界に住む「犯人」の検挙という見せ場を作ることができたという点で、沖縄サミットでの議論で世論を盛り上げるのに一役買ったのである。でも、そんな

FBIのパフォーマンスにやすやすと乗ってしまえば、それこそ強盗に扉を開くようなことになりかねないのである。

実はFBIの本当の狙いは、「I LOVE YOU」みたいなケチなイタズラを防ぐことじゃない。あんなワームは、Outlook ExpressからVBScriptが起動しないようにすればいいだけの話だ。FBIの本当の目標は、暗号の規制、ネットワークや通信回線の盗聴の権利を獲得すること。昨年成立した日本の盗聴法も、こうした米国の努力の成果なのだが、最近になって米国がECHELONで産業スパイをしたり、市民運動を監視していることが暴露され、問題になっている。日本の当局も、いつまでも米国の腰巾着をやっていては、決して国益にはならないと思うんだけれどね。

コンピュータの「多様性の保護」の必要性

ブラジルのNGOで仕事をしていた知人から、世界で作られている大豆やトウモロコシといった穀物の品種が急速に減っているという話を聞いた。便利だから、栽培しやすいから、といった理由から、同じ種類の種ばかりが使われるようになったのだそうだが、これはつまり遺伝子のバリエーションが減るということを意味する。その結果、わずかな気候の変動や病虫害が、作物の収穫に広大な地域にわたり被害を与えることがあるそうだ。このような傾向が広がれば、地球規模の大飢饉もあり得ると彼は警告した。

パソコンの世界も同じだ。電子メールクライアントがOutlook系一色になり、ちょっとしたイタズラが世界を揺るがすようになった。また、Windowsの独占的なシェアに依存することは、悪質なクラッカーによるセキュリティへの脅威ばかりでなく、横暴な当局の盗聴をやりやすくし、それは結果としてわれわれの自由な通信やプライバシーをいとも簡単に脅かす。

オープンソースやフリーソフトウェアは、そんなデジタル世界の「遺伝子」の多様性を確保するキーワードではないだろうか。

I LOVE YOUには、次々と変種が誕生しているのだそうだけど、これはあのワームがVBScriptで書かれた「オープンソース」だったからだ。もちろん、だからいいんだなんて言うつもりは毛頭ないが、ソースが見えていたおかげで、ちょっとやそっとではへこたれないワームになった（なってしまった、と言うべきだろうか）。ところが、狙われる側のWindowsのモジュールは、世界中で同じ物が使われているため、ちょっとした隙が世界的な被害をもたらすことになった。これは、単一の遺伝子を持つ作物が、変異しやすい病原ウィルスのおかげで一気にやられてしまうという構図とそっくりだ。

ワームの作者を弁護する気はないけれど、作者を逮捕すればいいとか、サイバー犯罪取り締りの法律を強化すればいいという問題じゃない。そんなことをしたって、バグやセキュリティホールのないソフトウェアはありえず、人騒がせなワームを作る連中の種は尽きない。本当にネットワーク時代の社会的な安全対策が必要なんだったら、現在のソフトウェアのあり方を根本から問い直さなきゃいけない。そして、オープンソースソフトが持つさまざまな社会的価値をしっかりと評価しなきゃいけない。

マイクロソフト社は、米国司法省の分割という決定に「ユーザーの不便」を主張して反論している。分割の是非はともかく、ぼくは100億ドルの間抜けたちの不便よりも、やっぱり最終的には世界中のアイデアに満ちたオープンソースの豊かな未来の可能性を確保するほうが、ずっと大切なことなんじゃないだろうかと思うのである。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ドクターShiodaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shioda

5・10 Intel MTHの欠陥を確認
5・8 SUN Java2 Ver.1.3リリース
5・8 NASDAQ Japan開設
4・28 USB 2.0の最終仕様が確定
4・28 司法省Microsoftの2分割案を裁判所に提案
4・27 AMD、Duronの名称発表
4・11 S3、ディスプレイチップ部門をVIAに売却

I Love Youウィルス来ました？ うちにはちっとも来なかったんだけど、来る人のところには何通も来たのだとか。来ないのは友達が少ないせいかしら。それとも、来るのはつき合いに問題があるから？ でも、あれが問題になるってことは、世界中のほとんどの人は、Windowsに付いてくるOutlook Expressをそのまま使っているんでしょうね。メールソフトぐらい、ほかにいいのがいっぱいあるのにねぇ。

Microsoft、大ピンチ？

Microsoftと司法省の独禁法を巡る裁判は、Microsoftが有罪ということになり、現在、その「是正措置」について検討が行われている。司法省はMicrosoftをOSとアプリケーションの会社の2つに分割するという案を裁判所に提出した。もっとも、この裁判はまだ「地方裁判所」のレベルなので、実際、司法省の案が通ったとしてもMicrosoftが控訴して、さらに裁判が続くわけなのだが、少なくとも司法省はそこまでしないと是正できないという認識をしていることがはっきりした。

Microsoftのほうは、自らに課する「制限」をプレスリリースなどで提案、こうした範囲のものであれば和解可能であるとし、分割案を棄却するように裁判所に要求した。Microsoftが主張する「義務」とは、Windowsの起動画面をOEMメーカーがカスタマイズできるようにする、APIを差別なく公開、新規OSのロイヤリティを既存のOSのロイヤリティ以下にする、などというもの。要は、Windowsのインストールについても制限しないし、情報は公開、新しいものは値上げしないというもの（このほかにOEMによる他社製品の扱いを妨害しないというものもある）。

だが、有罪になったMicrosoftが提案する案が受け入れられることがあるんでしょうか？ とりあえず、今回の裁判は分割ということになって、控訴という流れになるでしょう。これは、かつてのAT&TやIBMの裁判も同じで、必ず長期化する。ただし、司法省の態度というのは、その時の大統領によって違うので、もしかしたら、今度の大統領選挙の結果によっては、いきなり和解なんてこともあるかも。

さて、そのMicrosoftが次期主力OSとして開発中のWhistlerがまた、社外に流出した。3月にも同様の事件があったのだが、今回も同じ社内の人間によるものらしいとか。Microsoftは犯人捜しをしているらしいが、自分たちも間違って、Macintosh版Officeの情報を公開してしまったという。やっぱり、分割に動揺したのか？

インテルはちょっとピンチ

AMDに1GHzのAthlonを先に発表されてしまったインテルだが、いま、CPU不足を起こしているらしい。なんでも、需要予測を誤ったのが原因とか。このため、Celeronの633/667MHzの製品発表も延期したらしい。

それに加え、最新のチップセットと組み合わせてSDRAMを使うためのMTH（Memory Translator Hub）に欠陥があることが判明した。これは、Direct RDRAM用の設計されたチップセット820でSDRAMを接続するための部品。RDRAMが高価なので、これを使って安価に入手できるSDRAM対応にしたマザーボードは多い。なんでも、高負荷時にシステムがハングするなどの現象が発生するとか。820チップセットを使ったPCを持っている人は要注意である。

なんだが、あまりいいことがないインテルだが、これもAMDとの競争で製品計画前倒しを続けた影響か？ 今年のインテルは少しペースダウンしそう。

これに対してAMDは、いまのところずいぶん調子がいい。Athlonの低価格バージョンにDuronという名前を付け、Celeronに押され続けているK6がカバーする低価格分野での巻き返しを計るようだ。

ちょっと弱り目に祟り目といった感じのインテルだが、なんかMicrosoftと同じく弱気になったみたい。というのは、現在のPentium IIIに組み込まれているプロセッサシリアル番号を、次期CPUであるWillametteでは組み込まないことに決めたのだとか。実際、利用例も少なく、ほとんど使われていないし、プライバシー保護団体からの突き上げが厳しかったので、これ以上やるとかえってマイナスって考えたのでしょうか。それとも、個々のプロセッサに番号を入れるコストもやはり切りつめたいってことでしょうか。

株、株、株

Microsoftが有罪判決を受け、さらに分割の報道がなされると、米国でハイテク株が下がり始めた。Linux関係でも、株式公開したLinux関連企業の株価がずいぶんと落ちたようである。ただし、株はあくまでも株。もともと、Linux関連企業はほとんどが赤字で、利益を出せるのは来年以降といったところばかり。業績の悪化で株価が下がったわけでもないのである。

日本でも、インターネット関連などの企業で、どーんと株価が下がったところもあり、株をお持ちの方は気が気でないのかもしれないが……。まあ、小金しかない小市民（筆者のその一人である）は株などに手を出さないほうがよいのかも。

AMDはAthlonの低価格バージョンDuronを投入、K6に代わって低価格分野での巻き返しを計る。
http://www.amd.com/

トラブル続きながら人気の衰えないiモード。広告自粛後も変わらず1日2万件の契約。総数は700万に迫っている。
http://www.nttdocomo.co.jp/

NASDAQ Japanが5月8日に開設したが、ここもハイテク関係が多いようだけど、大丈夫かね。最近では、インターネットで簡単に株が買えてしまうし、ハマる人も出てくるかも。だいたい、アダルトサイトで国際電話やダイヤルQ2の高額な電話料金払わせられた人もいるんだから、気軽にボタン押して、大損なんてことはありえない話ではないよね。アダルトサイトには、勝手にダイヤルするソフトなんかが置かれているみたいだけど、そのうち、勝手に株取引するソフトなんかが出てきたりして……。

QualCommが日本で携帯電話事業？

次世代の携帯電話システムIMT-2000の国内での事業認可申請をQualCommが検討しているという話が出た。国内では、電波割り当ての問題などから、同一地域で営業できる事業者は3つまでとなっており、順当ならNTTドコモ、J-Phone、DDIの3社が申請して問題なく認可が行われる予定であった。ところが、QualCommの開発したCDMA2000方式をDDIが採用せず、NTTなどの開発したW-CDMA方式を検討しているという話が出てきた。QualCommの申請は、このDDIの動きを牽制するもの。なんでも、海外の携帯電話事業者と組んで、事業申請を行うのだとか。これを門前払いすると、国際問題にもなりかねず、ちょっとした騒ぎ。申請の締め切りは5月12日だったのだが、DDIはこれに押されてCDMA-2000を採用することにして、めでたく国内3社の事業申請になった。なんだか、米国企業の辣腕を見た感じだけど、同時に外圧に弱い日本の姿を再確認した。

NTTドコモは、iモード機が売れてしょうがないが、どうもサーバに問題があるらしい。急激にユーザーが増えたのはわかるが、サーバ障害でサービスができないんじゃねぇ（いったいどこのOS使っているんだ？）。なんでも、広告を自粛するとか。いまさら広告やめたって、そんなに変わらないんじゃないんでしょうか？

日刊アスキーLinux on Linux magazine

http://www.linux24.com/

日刊アスキー Linuxの舞台裏

～本格的2バイトワープロ「HancomWord」登場!～

ハンコムワード（旧アレアハングル2000）の起動ロゴ。

5月11日～12日にかけて行われた「LinuxWorld Expo/Tokyo 2000」会場で、ひときわ目を引いたのが、韓国のHancomLinuxのワープロ「アレアハングル2000」である。日本語化はほぼ完璧に行われており、使い勝手もなかなかいい。そして、試用中にさらなるニュースが飛び込んできた！

多言語対応ワープロ

5月11～12日に、東京ビッグサイトで行われた「LinuxWorld Expo/Tokyo 2000」（以下LW2000）は、85にのぼる出展社と2万5000人近い来場者を数え、盛況に終わった。この会場でことさらインパクトがあったのが、韓国のHancomLinux（http://www.hancom.com/jp/）ブースである。

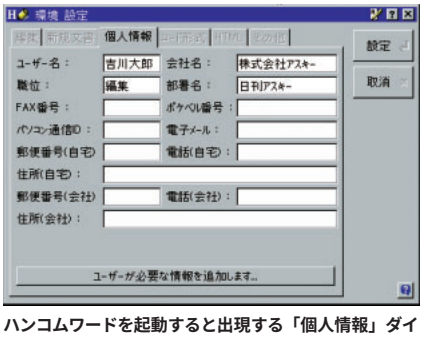
サーバ製品が圧倒的に多い中、展示してある製品はワードプロセッサであり、その名前が「アレア“ハングル”2000」である。それだけでも目を引くのに（今回、LW2000への韓国企業出展社は2社のみだった）、ブース中央の大型モニタでは、すでに同製品が日本語化されて動作しているのだ。その、まるでWindowsアプリケーションのような派手で取っつきやすそうな外観も印象的であった（実は、もともとはWindows用のアプリケーションであり、Wineを使ってLinux上で動作させているから、当たり前ではある）。

さて、実際の製品だが、第一に「わかりやすい」のが特徴だといえる。Wordをはじめとしたワープロを使った経験のあるユーザーならば、インストール直後から使い始めることができるだろう。実際の機能は、画面を多数キャプチャしたので、そちらをご覧いただければと思う。ちなみに、同製品の日本語化は、ほぼ完了しているといえるだろう。若干ダイアログの中にハングルが現れたりするが、実用に際して困ったことはなかった。特筆すべきはそのマニュアルで、完全に日本語化された160ページにおよぶマニュアルが付属している。

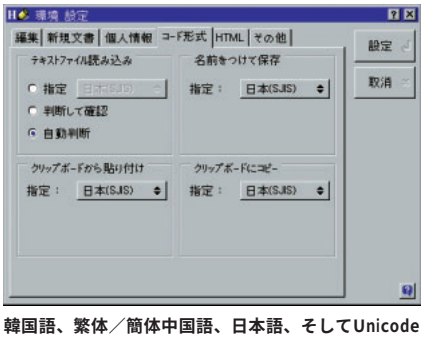
ソフト試用中にビッグニュースが！

インストールも終わり、製品を色々といじっていたら、なんと「レッドハット、ハンコムリナックスと提携契約を締結、ハンコムワード日本版の独占バンドルによるリリース開始を発表」というプレスリリースが、レッドハットからメールで送られてきた。それによると、アレアハングル2000は「ハンコムワード日本語版」と改名され、Red Hat Linux日本語版の次のリリー


画像の貼り付けや文字属性の変更、段組み、Webへのリンク、表、図形描画といった、一通りの動作はマニュアルを見なくても行えた。



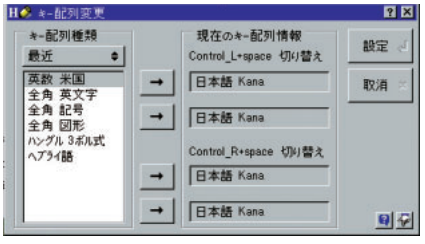
ハンコムワードを起動すると出現する「個人情報」ダイアログボックス。文書ごとにプロパティとして保たれるようだが、筆者の環境では今のところ入力情報を記録しておいてくれなかった。



韓国語、繁体／簡体中国語、日本語、そしてUnicodeなどが使用できる、マルチリンガルに対応したアプリケーションのため、ファイル操作やクリップボード操作時に、どのような文字コードを使用するのか設定できる。

スからバンドルされるという。同時にHancom Linuxは日本法人を設立、本格的に日本に進出するとのことだ。

注目度は高くても、まだまだ国内ではマイナーな存在（韓国ではトップシェアの製品）だと思っていたら、いきなり大手ディストリビューションへのバンドルとはさすがに驚いた。リリースにザッと目を通し、レッドハットに電話をかけたところ、すでにHancom Linuxとレッドハットの作業により、ベータ版の不具合解消作業が進行しているという。確かに、アレアハングル2000改めハンコムワード日本語版ベータを試用していると、日本語の入力などでいくつかの不具合が見つかった。その不具合をレッドハットのスタッフに告げると「ああ、それも直っていますよ」というお返事。製品ではさらに、DynaFontが使用可能になったり（ベータ版は、Hancom Linuxが用意したフォントしか使用できない）、lpr（プリンタ）周りもきちんと動くことになるそうだ。



キー配列の設定ダイアログボックス。日本語のローマ字入力やかな入力もサポートされている。「ハングル3ボル式」という見慣れない設定も。

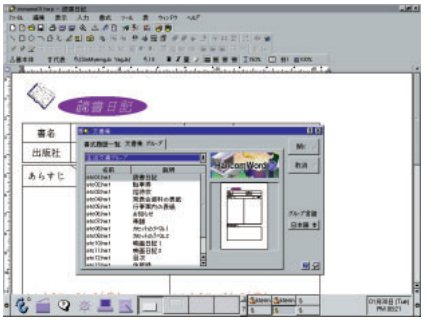
レッドハットに話を聞いたら、次は開発元のHancom Linuxに話を聞かねばなるまい。Hancom LinuxのWebページには、前述した日本語ページがあるし、日本語のメールも受け付けてくれているので、早速ハンコムワードの特徴を教えてほしい旨のメールを出してみた。すると、「まず、Linux環境でワード作業がすぐにできるというのが大きなメリットではないでしょうか」というお返事をいただいた。また、「ユーザーフレンドリー」というのもメリットだという。

●

ハンコムワード日本語版は、Red Hat Linuxへのバンドルだけではなく、単体のパッケージ販売も、6月中旬を目途に計画しているという。さらに、Hancom Linux製品はワープロにとどまらず、プレゼンテーションや表計算ソフトもリリースされるという最新情報も入手できた。Wineを使ったオフィス製品といえば、カナダCorelのWordPerfect Office 2000ばかりに目が行っていたが、同じ2バイト圏の国から来たマルチリンガルワープロということと、すでに日本語版が動いているということで、Linux上の日本語ワープロとして注目したい。次期LASER5 Linuxにバンドルされる予定のThinkFree Office Linuxや、ソース公開が決まった一太郎Ark for Javaなど、Linux上のオフィス製品の選択肢が増え、ユーザーとしては嬉しい限りだ。

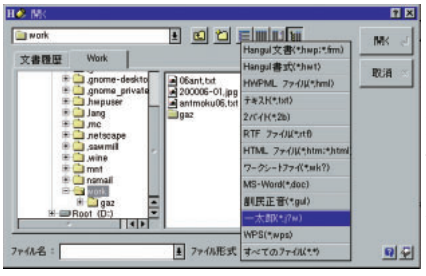
（日刊アスキー編集部・吉川）

プレゼンテーション機能。[ツール]−[プレゼンテーション]を選択すると、ワープロの1ページがそのままプレゼンテーション用のページとして生成される。

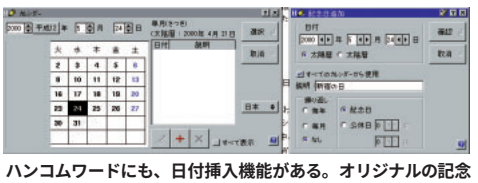


オフィス系のアプリケーションでは定番ともいえる各種文書ひな形。「住所録」はもちろんだが「ファッション名刺」や「駐車券」、「読書日記」や「映画日記」といった変り種もある。

クリップアート素材集「図片集」。「矢印集」や「アラビア数字集」、「昆虫」、「オフィス」、「コンピュータ」など、8種類のグループが用意されている。



ハンコムワード日本語版のファイル操作ダイアログボックス。左側のファイルリストは、ツリー形式のほかに、大きなアイコンを使った形式にすることもできる。ハンコムワード日本語版が扱えるファイルフォーマットはご覧のとおりだが、Microsoft Word 2000の「docファイル」や、Microsoft Word 2000で作成したHTMLファイルは、読み込んでも表示がおかしかったり、読み込んだとたんハンコムワードが異常終了した（テキストファイルは指定した文字コードさえ正しければきちんと読んでくれる）。なお、「Hangle文書」とは、ハンコムワード形式のファイルフォーマット。



ハンコムワードにも、日付挿入機能がある。オリジナルの記念日を追加することも可能だ。日本語化に合わせ、きちんと「平成」と入っているのが分かる。

正しきタコへの道

初級Linuxer養成講座

第10回　ファイルの見分け方

Linuxでいろいろなアプリケーションを使い始めたり、インターネットなどを通じてファイルをやり取りするようになると、いろいろな種類のファイルが手元に集まるようになってくる。はたして、削除してしまってよいファイルなのか、テキストエディタで中身をいじれるファイルなのか、判断に困ることもあるだろう。そのようなときに役に立つツールが、Linuxにはいくつか用意されている。

文：竹田善太郎
Text : Zentaro Takeda

今年のゴールデンウィークの話題といえば、やはり**I LOVE YOUワーム**の騒動に尽きるだろう。WindowsのVBScriptを悪用したマクロウィルスの一種だったようだが、伝わってきた情報から察すると、単に「マクロを悪用した」というだけでなく、現在のWindowsが抱えている大きな弱点を突いているのではと考えることができる。

Windows OSでは、ファイルの名前に付けられている「拡張子」によってファイルの種類を見分け、適切なプログラムに処理させるようになっている。ところが、現在のWindowsのデフォルトの設定では、ユーザーにはこの**「拡張子」が見えない**ようになっているため、一般のユーザーは画面に表示される「アイコン」の形だけでファイルの種類を判別しなければならない。

画面1 LOVEワーム（もちろんダミー）の本体はこう見える
拡張子「.vbs」が省略されると、無害な「テキストファイル」であるかのように表示される。かろうじて、アイコンの画像が「VBSファイル」のものになっていて見分けがつくのだが、「VBSファイルのアイコン」がこんな形をしているということを知っているユーザーが、はたしてそれほど多くいるものだろうか？

くだんの「I LOVE YOUワーム」では、ワーム本体のプログラム（VBScriptの添付ファイル）の名前が「love_letter_for_you.txt.vbs」というような名前になっているようだが、Windowsのデフォルトの設定では、このようなファイル名は拡張子の部分（.vbs）が見えなくなって、「love_letter_for_you.txt」という、**一見すると無害**なテキストファイルのように表示されてしまう（**画面1**）。幸い、Outlook Expressなどを含めて、多くのメールソフトは、Windows側の設定のいかんにかかわらず、添付ファイルの拡張子を省略してしまうことは少ないようだが、たとえば添付ファイルを通常のファイルとして保存してしまうと、有害なスクリプトファイルをテキストファイルと間違えてしまう危険性が高い。今回の「LOVEワーム」でも、Windowsのこのような弱点によって被害が増えた可能性もあるだろう。

Microsoftがどのような考えから、このような設定をデフォルトにしているのか分からないが、少なくとも筆者はこの設定では**きわめて使いづらい**と感じるので、Windowsをインストールしてまず最初に行うのは、Explorerの設定を変えて、すべての拡張子とすべてのファイルを表示させるようにすることだ（**画面2**）。

拡張子がユーザーに見えなくなっていることは、アプリケーションソフトを配布したり販売したりする側にとってもやっかいなことと思われる。Windows用のアプリケーションのインストールマニュアルを読むと、CD-ROMからの自動起動ができない場合に、セットアッププログラムを起動するのに、Explorerのファイル一覧から起動させるのではなく、わざわざ面倒な［スタート］メニューの［ファイル名を指定して実行］を使うように指示しているのはなぜだろうかと、つねづね疑問に思っていた。どうやら、Explorerのデフォルトの設定ではファイルの拡張子が表示されないので、たとえば、「setup.txt」、「setup.exe」、「setup.hlp」などのファイルが、すべて「setup」と表示され、**素人にはまったく見分けがつかない**ことを嫌ってのようだ。**setupファイルをクリックしたのに、説明書が表示されるだけで先に進まない**というような苦情が殺到する

のを避けたいので、わざわざ面倒な［ファイル名を指定して実行］からインストールプログラムを起動するように指示しているのだろう。

ファイルの種類を意識する

Windowsがこのような**おかしな設定**になっているのは、おそらくMacintoshを意識してのことだろう。Macintoshの場合、ファイルの種類の区別は、ファイル名の拡張子ではなく、ファイル自身に記録されているファイル情報で行うようになっている。ユーザーから見た画面上では、これもファイル自身に記録されているアイコンデータ（あるいはファイルの種別に応じてシステムが持っているアイコンデータ）を使用して、ファイルの種別によって異なるアイコン画像が表示されるようになっている。

Windowsの場合も、DOSの時代から使い慣れていて、**ファイル名**や**拡張子**を扱い慣れているユーザーはともかく、最近になってパソコンを使い始めたユーザーの場合は、アイコンの画像を頼りにファイルを区別している人も多いだろう。

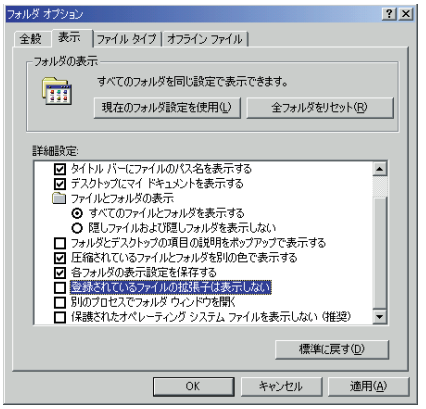
使っているアプリケーションの種類が少ないうちは、アイコン画像によるファイルの種類の見分けは簡単なのだが、多数のアプリケーションをインストールして使い分けるようになってくると、アイコンの種類は増えて、どのアイコンがどのアプリケーションに対応するものなのか、だんだんわかりにくくなってしまう。また、自分の管理しているマシンはよいのだが、他人が使っているマシンや、共用しているマシンなどでは、**見慣れないアイコン**が表示されていると、つい身構えてしまう。見慣れないファイルのアイコンをクリックすると、思いも寄らないアプリケーションが起動して、あわてさせられることにもなるからだ。

無害なアプリケーションが起動するだけならよいのだが、今回のLOVEワームなどの有害なプログラムや、システム設定を変更したり、あるいはディスクの中身を書き換えるようなプログラムが起動してしまうと、もっと困ったことになる。

なにがいいたいかといえば、Macintosh、Windows、Linuxを問わず、パソコンを使ううえで、自分が開こうとしているファイルがどのような性質のものなのか、中身にはなにが入っているのか、については、常に注意すべきであるということだ。その手がかりのひとつとなるのが、ファイルにつけられている名前なのだが、現在のWindowsでは、その重要な情報源である拡張子が、意図的に隠されている。最近のMicrosoftは（そしてApple Computerも）、**由らしむべし、知らしむべからず**の**超保守お役所的施策**が目立つのだが、今回のワーム騒動の教訓として、エンドユーザーの側でも、もっと**ファイルの中身を意識する**ようになってもらいたい。少なくとも、見慣れないファイルをいきなり実行したりアプリケーションで開くようなことは慎むべきだ。

「拡張子」には意味がない

Linuxの場合も、Windows以上にファイルの種類を見分けることは重要だ。というのも、Linuxでは**ファイル名とファイルの中身はまったく関係ない**からなのだ。たとえば、バイナリ形式のプログラムファイルに「readme.txt」という名前がついていたとする。Windowsの場合はこのファイルをダブルクリックすると、そのプログラムコードを実行するのではなく、メモ帳などのテキストエディタを起動して、ファイルの中身をそのエディタに読み込もうとする（バイナリファイルを開けば、当然、画面には意味のない文字列が表示されることになる）。しかし、Linuxの場合は、コマンドラインで、



画面2 Explorerの設定を変える
Windowsをインストールして真っ先にすべきなのが、Explorerの設定を「拡張子を省略しない」ように変更することだ。筆者の場合、ついでに「すべてのファイルとフォルダを表示する」、「保護されたオペレーティングシステムのファイルを表示する」ように変更している。自分のパソコン上のファイルは、すべて自分で管理できるようにしておかないと不安だし不便だからだ。もちろん、パソコンの使い方によっては、このような設定ではよくないこともある。

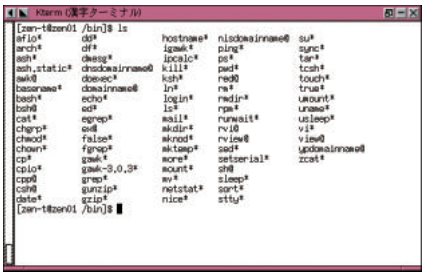
```
$ ./readme.txt
```

というように入力してやれば（readme.txtファイルに実行属性がつけられている場合）、ファイルの中身のデータをメモリに読み込んで実行（しようと）する。「.txt」という「拡張子」の部分は、Linuxにとっては何の意味もない、ただの**ファイル名の一部にすぎない**のだ。すでにご存じかと思うが、通常、Linuxのコマンドファイルは、「.exe」や「.com」などの拡張子を一切つけないファイル名になっている。/binなどのディレクトリをlsコマンドで表示させてみればわかるだろう（**画**

面3)。Windowsの場合は、「.exe」、「.com」、「.bat」などの拡張子がファイル名についている場合は、無条件に実行可能なコマンドファイルであると判断するようだが、Linuxの場合は、ファイル名にかかわらず、実行可能属性がつけられているファイルなら、ユーザーが「実行せよ」と指示すれば、とりあえずその内容を読み込んでみて、それからどのように処理するかを判断しているのだ(実際には、この処理の大部分を行っているのはLinuxの本体ではなく、csh、bash、あるいはX Window Systemのウィンドウマネージャなどの「シェルプログラム」である)。

通常は、**よほどひねくれた人間**でない限り、ファイルの中身が何なのかを想像できるような名前をつけるし、Linuxでは絶対的な意味がないとしても、DOSやWindowsのユーザーが多い現状に則して、「.txt」、「.html」、「.c」などの**よく知られた拡張子**をつけるのが普通だ。また、このように種類別の拡張子をつけておけば、ファイルを操作する場合に「ls ./*.c」とか「rm ./*.mp3」などといった「ワイルドカード」的な表現が使えるので便利である。

ところで、たった今「Linuxでは拡張子に意味はない」と書いた直後で、それを覆すようで恐縮なのだが、最



画面3 Linuxのコマンドファイル
/binや/usr/binなどのディレクトリには、さまざまなコマンドファイルが保存されているのだが、Windowsなどとは違い「.exe」や「.bat」などの「拡張子」はつけられていない。

近のX Window SystemのGUI環境(GNOME環境など)では、ファイル名からファイルの種類を判断して、アプリケーションを起動するようになっている場合がある(**画面4**)。このような環境では、ファイル名の「拡張子」も意味を持つようになり、画面上ではファイルの種類ごとに異なるアイコンが表示され、Windows環境に似た使い方ができるようになるのだが、ファイルの中身と拡張子が正しく対応しているかどうかは保証されておらず、間違ったファイルを実行したり、開いてしまう可能性があることに変わりはない。

ファイルの種類

ここで、Linuxにはどのような種類のファイルがあるのか、おさらいしておこう。

ファイルの種類分けをする方法はいろいろあるし、厳密な分類をするのは実はとても難しいことなのだが、ここではまず、ファイルを「テキストファイル」と「バイナリファイル」の2種類に分ける。

・テキストファイル

ASCIIコード、あるいは、JIS、シフトJIS、EUCなどの文字コードの規格に沿って、文章やソースプログラムなど、文字の羅列されたデータからなるファイル。catコマンドやlessコマンドで画面上に表示すれば、**人間にも意味がわかる**ようなもの。ここで「意味がわかる」とは、それが何かを表現した文字の並びになっていると判断できることであって、**プログラミング言語はさっぱりわからん**という人にとって、プログラムのソースコードなどは**意味不明の文字の羅列**に見えるかもしれないが、これも「テキストファイル」として分類できる。乱暴に言ってしまえば、**テキストファイルとは人間にもそのまま読めるファイル**とすることができる。

ところで、ASCIIコードの文字しか表示できない環境で、たとえば日本語の文字コードを含むファイルを表示させると、画面上には意味不明の文字列が表示されることになる。このような場合、日本語EUCのファイルはバイナリファイルとみなすのか、テキストファイルとみなすのかは難しいところだが、ここでは深く考えないで、なにかしらの文字コードの規則に準拠していれば、それはテキストファイルであると考えることにしよう。

・バイナリファイル

「テキストファイル」に分類されないファイルは、すべて「バイナリファイル」ということになる。バイナリとは、**2進数**という意味で、いわば、コンピュータの側から扱うのに都合のよいファイルということになる。バイナリファイルをcatコマンドやlessコマンドで表示させても、画面上には意味不明の文字ばかりが表示されるほか、場合によってはそれ以降の画面表示がおかしくなってしまうこともある。ワードプロセッサや表計算などのソフトで作成した文書ファイルは、内部にはテキスト形式のデータも含まれていて、catコマンドなどで表示すると、**部分的には人間にも読める**文字が表示されることがあるのだが、さまざまな書式設定や付加情報などは、文字コードの規格には収まらないバイナリ形式のデータとして記録されているので、このようなファイルもバイナリファイルとみなすことにする。

ハイテータイプで作成したり変更したりできるかどうかが重要だからだ。Linuxの場合、ワープロソフトなどのアプリケーションがまだまだ未発達で、文章の作成やデータ入力には、テキストエディタを使うことが多く、また、最近ではX Window System上のGUIツールが増えてきたものの、システムの各種設定を行う場合でも、テキスト形式の設定ファイルをテキストエディタで編集することが多い。ファイルの内容がプログラムコードなのか、文章なのか、データなのかはおいておいて、ともかく**テキストエディタでいじれるかどうか**を第一に意識するのが重要なのだ。

テキストファイルとバイナリファイルに分類したあとは、それぞれ、そのファイルがどのような目的に使われるものなのかに応じて、さらに細かく分類することになる。この分類の方法はいろいろあって、**これが正しい**という分類法を決めるのは不可能なのだが、Linuxのエンドユーザーの立場から見れば、次の2種類に分類できるだろう。

・プログラムファイル

バイナリ形式のプログラムファイルや、テキスト形式のスクリプト（Perl、シェルスクリプトなど）といった、Linux上で「実行」することのできるファイル。

・データファイル

アプリケーションからデータを書き込んだり読み出したりするためのファイル。あるいは、アプリケーションやシステムの動作方法を指示するためのくて、たとえば、C言語などのコンパイラ型言語で処理するためのソースファイルは、プログラムファイルと考えることもできるが、そのままコマンドラインなどから「実行」することはできないので、単なるデータファイルとみなすこともできる。ともあれ、ユーザーの立場からは、**コマンドラインでファイル名を入力しても、直接には実行できないファイルはデータファイル**と考えておけばよいだろう。

ファイルを見分けるコマンド

では、目の前にあるファイルが、バイナリ形式のプログラムファイルなのか、テキスト形式のスクリプトファイルなのか、データファイルなのかを見分けるにはどうすればよいのだろうか。実行可能属性がついているから、プログラムファイルだろう。**とりあえず実行してみよう**というのは、大昔ならともかく、素性の知れないファイルが紛れ込む可能性がある、殺伐とした現在のネットワーク環境ではかなり危険な行為だ。特に、rootユーザーとしてログインしている状態では、厳として慎まなければならない。また、Linuxではそれほど頻繁に起こるわけではないが、正しくないバイナリファイル（たとえば、自分のLinux環境とは異なるLinuxシステム用にコンパイルされたファイルなど）を実行してしまうと、システムの動作がおかしくなったり、画面が乱れるなどの不具合が生ずる可能性もある。

画面4 GUI環境でファイルを見る
GNUのファイル管理ツール「Midnight Commander」では、MIMEの設定に基づいて、ファイルの種類別に異なるアイコンでファイルを表わすようになっている。ここでは、ファイルの種類を区別するのにファイル名が使われている。アイコンを右クリックして「プロパティ」メニューを選択しても、MIMEに設定された情報以上のファイル情報は表示されない。GUI環境においてもfileコマンドは有用なのだ。

Linuxでファイルの素性を探る場合、最初に使うコマンドが**fileコマンド**だ。このコマンドは、ファイルの内容を解析して、どのような目的に使われるファイルなのかを表示してくれる、とても便利なものだ。使い方は簡単で、コマンドラインに調べたいファイルのファイル名を指定して実行するだけだ。たとえば、「weirdfile」というファイルの素性を知りたければ、コマンドラインで次のように入力すればよい。

```
$ file weirdfile
```

すると、次のようなメッセージが表示される。

```
weirdfile: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), not stripped
```

すべて英語のメッセージで、しかも**専門用語**が並んでいるのでちょっと手ごわい感じもするが、たとえば上の例では、

「ELF 32ビット LSB形式の実行ファイルで、CPUはIntel 80386用、バージョンは1、動的リンク（共有ライブラリを使用）、デバッグ用文字列を含む」

という意味になっている。簡単に言えば、Intel CPU用のLinuxで動くバイナリファイル、ということになる。現在のLinuxディストリビューションのほとんどは、「ELF形式」のバイナリファイルが標準になっているので、fileコマンドで「ELF …… executable」というメッセージが表示されるファイルは、「Linux用のバイナリプログラムファイル」であると考えておけばよい。

同じようなプログラムファイルでも、

```
weirdfile: perl commands text
```

とか、

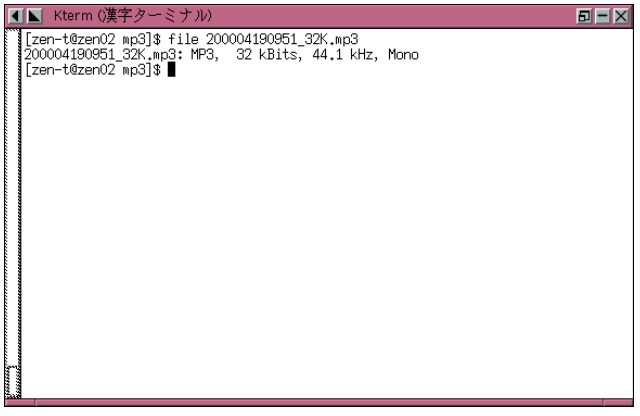
```
weirdfile: Bourne shell script text
```

などと表示される場合がある。これらはそれぞれ、perl用、bash用のスクリプトファイルで、テキスト形式のファイルであることを示している。このようなファイルは、コマンドラインから実行できるほか、テキストエディタで開いて内容を確認したり、変更したりすることができる。

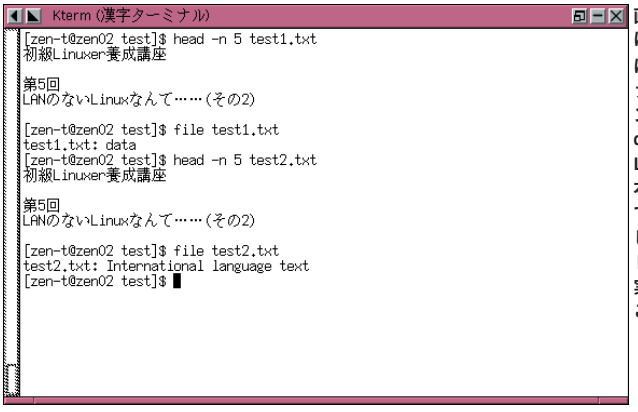
fileコマンドで識別できるファイルの種類は、Linuxのディストリビューションによって異なる（実際には、/usr/share/magicというファイルに、ファイルの見分け方が記述されている）。最近では、MPEGやAVIといった画像、音声系のファイルの種類なども見分けられるようになっていて、かなり便利である（**画面5**）。とはいえ、fileコマンドも常に正しいわけではなく、よく似た構造のバイナリファイルを見分けられなかったり、間違った結果を返す場合もあるので、注意が必要だ。

fileコマンドで判断できない内容のファイルについては、テキストファイルである場合には、「English text」や「ASCII text」などと表示される。バイナリ形式のデータファイルの場合には、単に、「data」としか表示されない。

問題なのは日本語のテキストファイルで、「International Language」と表示される場合もあれば、「data」としか表示されない場合もあって一定しない（**画面6**）。日本語の文字コードにJIS、SJIS、EUCなどさまざまな種類があって、fileコマンドでは判別が難しいからだろう。このあたりは、今後の



画面5 MP3ファイルの種類を見分ける
fileコマンドでは、ファイルの大まかな種類だけでなく、内容の細かな種類まで判別してくれることもある。MP3形式の音声ファイルにfileコマンドを使うと、ビットレートなどの情報も表示してくれる。かなり便利である。



画面6 日本語のテキストファイルは……
ほぼ同じ内容の2つのテキストファイル（日本語EUC）をfileコマンドで判別させてみると、一方はdataと、もう一方はInternational Languageであると判断される。本当なら、文字コードなども含めてきちんと判別してくれるとうれしいのだが……。日本語文字コードの種類を完全に見分けるのは、実はプログラムを使っても難しいことらしいのだ。

ディストリビューションでは改善されてゆくだろう。

バイナリファイルを覗く

fileコマンドで判断がつかないバイナリファイル、あるいは、さらに詳しくファイルの素性を知りたい場合、ヘビーユーザーならodコマンドを使って、ファイルの内容を16進数などの形式で表示させて、ファイルの構造を解析するところだろう。しかし、一般のユーザーではそこまでしようとは思わないだろう。このようなユーザーでもstringsコマンドを使えば、ファイルの種類についてある程度の判断がつく場合もある。stringsコマンドは、バイナリファイル中に埋め込まれているテキスト文字列を抽出してくれるコマンドだ。stringsコマンドに中身を知りたいファイル名を指定して起動すれば、すぐに文字列が表示される。バイナリのコマンドファイルなどに対してstringsコマンドを実行すると、かなり大量の文字列が表示されるので、lessコマンドなどを併用するのが無難だろう。

```
$ strings ./weirdfile | less
```

stringsコマンドは、テキストデータと思わしき部分をすべて抜き出してくるので、実際には意味のない文字列もかなり多く含まれる（**画面7**）。しかし、lessコマンドなどで出力された文字列をスクロールしていけば、ファイルの素性を知るキーワードがどこかに見つかる可能性がある。もちろん、stringsコマンドの出力からファイルの正体を知るには、Linuxやプログラミング言語についての知識がある程度は必要になるが、勘を働かせれば大体のところはわかるようになるだろう。ときには、プログラムの作者が埋め込んだ、隠しメッセージなどを発見できることもあるので、それを探してみるのも面白いだろう。

stringsコマンドは、デフォルトでは4文字以上続くテキスト文字列を見つけるようになっているが、-nオプションでこの文字数を指定すれば、もっと長い文字列だけ見つけるようにすることもできる。たとえば、10文字以上の文字列だけ探すようにするには、

```
$ strings -n 10 ./weirdfile | less
```

とすればよい。

テキストファイルの先頭だけ見る

テキストファイルの中身を知りたい場合は、catコマンドやlessコマンドで表示することができる。テキストエディタで開いてみても一目瞭然だが、いちいちファイル全体を開かなくても、CDプレイヤーやビデオデッキなどのイントロサーチ機能のように、ファイルの先頭部分（あるいは末尾部分）だけ見れば事足りる場合もある。このようなときに便利なコマンドがheadコマンドとtailコマンドである。

headコマンドはファイルの先頭部分、tailコマンドはファイルの末尾の部分だけを出力してくれる。たとえば、

```
$ head ./weirdfile
```

とすると、weirdfileの先頭10行ぶんだけを表示してくれる。同様に、

```
$ tail ./weirdfile
```

とすれば、weirdfileの末尾の10行ぶんだけが表示される。どちらのコマンドも、-nオプションを使えば出力される行数を変えることができる。

```
$ head -n 5 ./weirdfile
```

とすれば、先頭の5行だけ表示してくれるのだ。

Linuxでファイルの種類を見分けるには、いままで説明したいろいろなコマンドを組み合わせて、自分で推理することになるのだが、実際にはここまで必要になる場面は少ない。たいがいは、ファイルにはあらかじめわかりやすい名前（もちろん「拡張子」つき）がつけられているからだ。しかし、見慣れないファイルや中身を忘れてしまったファイルが見つかった場合には、ここで説明した方法を使ってファイルの素性を探るように心がけたほうがよいだろう。

画面7 stringsコマンドを使う
バイナリ形式のコマンドファイルには、ヘルプメッセージやエラーメッセージを含む、表示用に定義されたテキストデータが埋め込まれていることが多い。これをstringsコマンドで抜き出してみれば、そのファイルの用途などが判断できることがある。ただし、大部分の文字列は、プログラムが内部で使っている関数名や、デバッグ用の情報、あるいは、バイナリコードがたまたま文字と同じコード範囲に収まっている部分などである。

Javaプログラミング入門
Javaで3Dゲームを作ろう

今回から始まるこの連載では、今話題のJava言語を使ったプログラミングを基礎から解説していきます。もちろん、単なるプログラム入門では面白くないので、なんと3Dゲームを作成するという趣旨を含めてみました。極力やさしく説明していきますので、Java言語の魅力を堪能してください。

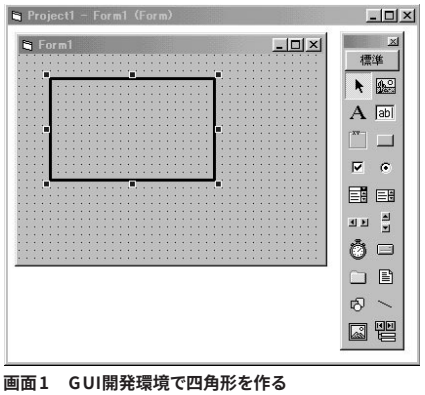
第1回　Javaとは

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote／Kazaguruma

プログラミング入門

本誌をお読みになっている読者の中には、プログラミングに興味を持っている方も結構おられることでしょう。

最近は大半の操作をマウスなどで行えるGUIでの開発が主流になりつつあります。GUIでは部品を画面上に切ったり貼ったりして、それらの動きをチョコチョコっと（時にはたくさん）指定すればプログラミングができるので、たしかに初心者にもとっつきやすく、短時間で開発が完了するというメリットがあります（**画面1**）。



画面1　GUI開発環境で四角形を作る

少し昔を振り返ると、WindowsやMacintosh、X Window Systemが生まれる以前は、キャラクタベースのC言語やBASICといったプログラミング言語を駆使してほとんどすべてのソフトウェアの開発を行っていたものです。グラフィックス中心のゲームを開発する場合も同様で、カラフルでスピード感あふれるゲームほど、何千（万）行もの無味乾燥したプログラムコードが必要でした。画面上に色のついた四角形を1つ表示するにも、数行のコード（命令）を並べて書く必要があり、しかもそのコードを書いた時点では正しく表示されるかどうかもわかりません（**リスト1**）。コンパイルしてできたプログラムを実際に動かしてみてはじめて結果がわかるといったありさまです。四角形の大きさや位置、色あいなどが満足できなければ、またコードを修正してコンパイルして実行して…といった繰り返しがいわばプログラミングだったのです。

▼リスト1　非GUI開発環境で四角形を作る

```
public class paint(Graphics g) {
    g.setColor(Color.black);
    g.drawRect(20, 20, 160, 100);
}
```

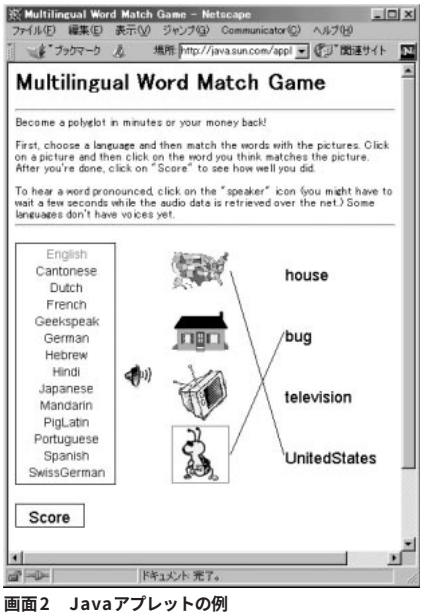
とはいえ、経験を積んでくると何度も書いたことのあるコード、たとえば線を引くだの、キーボードから入力するだのといったことは、コードを書いたあといちいち実行して確かめずに、実行結果を想像しながら書けるようになります。適当なところでコンパイルして実行し、結果を確かめるのです。そうすればその都度実行するよりも数倍効率がよくなりますし、なによりも想像力を身につけることができるようになります。

この想像力は結構重要で、コードをどう書けばどのように動いてくれるのかがイメージできるようにならなければ、プログラミングは上達しません。特にプロのプログラマーを目指す方には必要不可欠な能力といえるでしょう。

もうひとつ、非GUIプログラミングで得られることがあります。非GUIプログラミングでは、最初からできあがった部品を使わないので、かなり細か

いところまでその動きやふるまいをプログラミングしてやらなければなりません。それだけ作業が多くなるわけですから、一見不便な面と思えるかもしれませんが（実際、不便で大変ですが…）、おかげでソフトやコンピュータが裏側でどう動いているかを理解できるのです。これもプログラマーにとって大切なことで、何よりも自分で書いたプログラムが思い通りの動きをしてくれなかったときに、何が原因なのか、どう改修すればよいのかを解決する力が生まれてきます。

もちろん、GUI開発環境はそれらをおぎなってあまりあるぐらい魅力的な機能を与えてくれます。ですから今後のプログラミングの主流になってもおかしくはないでしょうし、読者諸氏がGUI開発環境を使っていかれることに何の異論もありません。ただ、このような昔ながらの非GUIプログラミングを通して基礎力を鍛え、うわべだけでなくじっくり基礎を充実させることがその道をめざすための早道なのです。それはプログラミングのみならず、どの世界でも同じことなのかもしれません。



画面2　Javaアプレットの例

Java言語とは

前置きが長くなりましたが、本題の「Java」です。Javaはインターネット時代を迎えるようになってから、サンマイクロシステムズによって開発された、新しい「プログラミング言語」です。もともと多目的、多用途のために生まれたJavaですが、これまでのところ、インターネットが普及を後押ししてくれたともいえるようです。生みの親がサンマイクロシステムズなら、育ての親がインターネットといったところでしょうか。

冒頭でGUI開発環境か、非GUIかという話をしましたが、「JavaはGUI開発環境か」という議論には意味がありません。というのも、Javaはプログラム言語であり、GUIか非GUIかというのは、あくまでもプログラミングをするための環境／ツールの問題だからです。いいかえれば、GUI開発環境を使っても、あるいは非GUIでもJavaプログラミングはできるということになります。

現状ではまだ、JavaのためのGUI開発環境は、BASIC、C++などに比べればそれほど一般に普及していないようです。今後、PC用のソフトウェアを開発するための言語の中で、Javaのウェイトが高くなれば、いろいろなGUI開発環境がリリースされ、普及していくことでしょう。本誌でも、先月号からインプライズの「JBuilder」というJavaのGUI開発環境が解説されています。先月号の付録CD-ROMには「Foundation版」が収録されていましたので、興味ある方は使ってみるのもよいでしょう。しかし、むしろJavaの生まれた経緯を考えると、Javaの世界の全てをGUI開発環境が占めるというようなことはないようです。

そもそもJavaが開発されたのは、各種家電製品用のソフトウェアを作成するためで、たとえば洗濯機などに組み込むソフトウェアの開発用だったわけです。家電向けゆえ、かろうじて小さな液晶パネルを使える程度で、大きなディスプレイやマウス、キーボードといった、PCでは定番のインターフェイスが使えません。ですから、ウィンドウやボタンといったGUIお定まりの部品は使う必要がなく、どちらかといえば、開発したソフトウェアが多くのメモリを使わないとか、実行速度がどうだという、よりハードウェア寄りの問題を解決する必要にせまられます。この種のソフトウェアを開発する場合、GUIに彩られた開発環境はあまり役に立たないでしょう。それよりも、シェイプアップされてかつ性能の高いプログラムコードが書ける優秀なプログラミング技術が必要になるはずです。

本連載では、まさにその非GUI環境でのJavaプログラミングを学習します。残念ながら、洗濯機に組み込むプログラムとまではいきませんが、開発環境の便利さにとらわれず、プログラミングそのものの楽しさと真髄にぜひとも触れていただけるよう、非GUIのプログラミング環境で、じつにトレンディな新プログラミング言語「Java」を学習してみましょう。

OSに依存しないプログラム言語

もうひとつ、Javaにはとっておきのセールスポイントがあります。そもそもPCはパーソナルコンピュータという名のごとく個人利用のために生まれたものですが、1990年代後半より、インターネットの普及とともに「ネットワーク」を意識するようになりました。つまり、今やPCは単独で存在するもの

でなく、世界中に広がるネットワークの一部として利用されることが当たり前になってきているのです。個々のPCはCPUも違えばOSも違うため、目的や用途が同じソフトウェアであっても、違ったプラットフォーム（CPUやOS）上では当然のように動きません。

そこに一石を投じたのがJavaです。Javaは生まれてきた経緯からすでにインターネットのようなコンピュータネットワークを意識しているため、これらプラットフォームに依存することなく、同一のプログラミングが可能なのです。ようするに、OSやCPUが違っても、全く同じJavaのプログラムコードで、それぞれのソフトウェアが作成できるということです。

最近では、Webアプリケーションのようなインターネットを考慮したアプリケーションの登場とともに、多くの人が好んで使う定番ソフトウェアといえるものがよりはっきりしてきました。最低限、ワープロやWebブラウザ、メールソフトなどがあれば、高価で雑多なソフトウェアを買い揃えることもなく、用途に合わせてインターネット上で配布されているソフトウェアを利用し、各種サービスを受ければよいというようなオンデマンド（欲しいときに欲しいものを）的発想に移り変わりつつあります。となれば、OSに依存せず、しかもオールインワンでなく、単用途で小回りの利くソフトウェアの開発が必要とされてくるでしょう。

Javaの発想はまさにそこにあります。Visual C++あたりが得意としてきた特定のハード向けの大掛かりな製品を開発するよりも、インターネットやPDAなどを念頭において、身近なソフトウェア環境を充実させる小気味よいアプリケーション開発に、皆さんも一役買ってみてはどうでしょうか。

アプリケーションとアプレット

このように、さまざまな用途のプログラムを作成できるJavaですが、PCで実行可能なプログラムの形態として「アプリケーション」と「アプレット」があります。まず、アプリケーションとは、PC上で直接実行できるプログラムで、皆さんが日頃使っているソフトウェアと同じように操作することができます。OSやCPUに依存しないJavaにはからくりがあって、JVM（Java Virtual Machine）と呼ばれるJavaプログラムの土台というべき実行環境を必要とします。JVMは多くのプラットフォーム用に開発されており、LinuxにはLinux用の、WindowsにはWindows用のJVMをインストールしてあれば、Javaアプリケーションを実行することができます。

一方アプレットとは、Netscape NavigatorなどのWebブラウザ上で実行できるのが特徴です。現在、ほとんどのWebブラウザで実行することができます。つまりJavaアプレット用のJVMはすでにWebブラウザに組み込まれています。さらに、アプレットはネットワークを通じて転送できるので、Webブラウザを使ってHTMLのホームページなどと同じようにサーバからアプレットをダウンロードし、実行することができます。Webブラウザやプラットフォームを問わず、たったひとつのJavaアプレットを用意するだけで、あらゆるユーザーがJavaを使えるわけです。

本連載では、このJavaアプレットの開発を学習していきます。Webが閲覧できる環境であれば、誰でも実行することができますし、作成したアプレットは、自分のホームページなどに埋め込んで広く使ってもらうことができます。せっかく学んだプログラミングですから勉強だけで終わるのではもったいないので、実用的に使っていくためにもアプレットはもってこいでしょう。

Javaアプレットのサンプルは、開発元のサンマイクロシステムズのサイトや（http://java.sun.com/applets/）、その他のホームページで公開されています。どんなことができるのか、一度見ておくことをおすすめします。

3Dゲームに挑戦

本連載では、プログラミング入門としてJavaのプログラミングを学ぶのですが、あまり教科書然とした題材ではきっと退屈してしまうでしょうから、プログラミングも楽しめて、なおかつ作ったあとでも楽しめるゲームを開発することにします。しかも3Dゲームです。みなさん、遠慮なく感動してください。

とはいえ、市販されているような高度なアクションゲームなどを開発しようとすると、この本1冊くらいでは足りそうもないので、そこそこ遊べるレベルのゲームだと思ってください。

画面3がこのゲームの画面です。奥行きのある壁の様子と、ボールに影があるのがわかると思います。立派（？）な3Dゲームですね。

図1にこのゲームを構成する部品の説明があります。基本的に「ブロックくずし」です。くずされるブロックと、ブロックをくずすボールと、それを打ち返すパドルがあればOKです。3Dということで、ここではパドルの代わりにラケットと呼ぶことにします。しかしこのラケット、ガットも張ってなければ、グリップもありません。これは、できるだけプログラムを省略したせいです。ゲームを行うには、ラケットの面だけがあればこと足りるでしょう。

ラケットの面に薄く色を塗ることも考えたのですが、ボールが見えにくくなるため今回は透明にしました。透明といえば聞こえはよいのですが、ようするになにも描いていないということです。このあたりの装飾は、本連載終了後の皆さんの力量におまかせします。

もうひとつ、昔のブロックくずしでも忘れてはならない重要なパーツが、左右の壁です。プレイしているときには、ボールが左右の端ではね返るなんてことは当たり前で気にすることもなかったでしょうが、プログラムを作る場合にはこれが結構重要だったりします。ボールが壁ではね返る動きをプログラムすることを忘れてしまうと、ボールは壁に当たったが最後、そのまま壁に吸い込まれて永久に戻って来ないなんてことになってしまうのです。ともかく、ボールがはね返るとはどういうことなのか、どのようにプログラミングすればよいのかなど、プログラム上では考えることが多くなりそうです。しかも、2次元のときはボールの動く方向が前後左右だけでよかったのに、3次元になると上下も加わります。つまり天井と床を作ってやらなければならないのです。

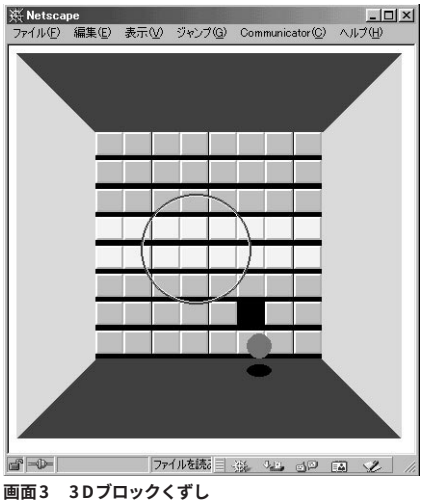
さらに、はね返りの方向もややこしそうです。2次元なら反射角と入射角などと、とりあえず紙に書いてみれば比較的理解できたかもしれません。しかし、3次元での反射となると…。このあたりは、次回以降にコツのようなものを解説していきますので、ご心配なく。あわてて数学の教科書を引っぱり出すにはおよびません。

とにかくJavaを書いてみる

あれこれ説明ばかりでも退屈でしょうから、まずはとにかくJavaのプログラムを実際に書いてみることにしましょう。習うより慣れろというわけです。

Javaのプログラミングには最低限、「JDK」と呼ばれる「開発ツールキット」が必要になります。これは、「JBuilder」のような、プログラミングを便利にする開発環境ではなくて、Javaプログラムを動かすために必要なJVMなどを含んだソフトウェア開発キット（SDK）です。

JDKは、サンマイクロシステムズのサイト（**表1**）からダウンロードするか（約21Mバイト）、6月号の付録CD-ROMからインストールすることができます。6月号をお持ちでない方は、Webでバックナンバーを購入することができます（255ページ参照）。



画面3　3Dブロックくずし

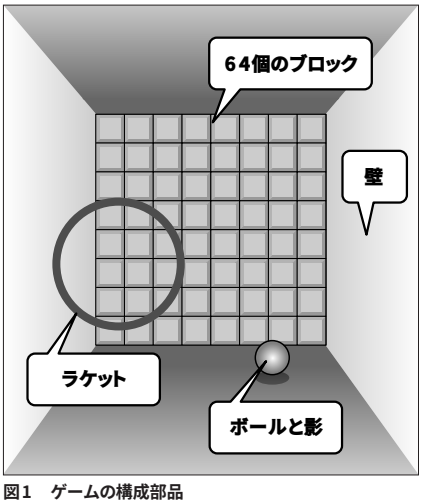


図1　ゲームの構成部品

URL	http://java.sun.com/products/jdk/1.2/ja/download-linux.html

表1　JDKのダウンロードサイト

JDKのインストール

JDKをインストールするには、スーパーユーザーとしてログインします。次に、JDKの収録されているディレクトリに移動します。そしてJDKを/usr/localにインストールします。

```
# tar  zxvf jdk1_2_2-linux-i386.
tar.gz  -C  /usr/local
```

これで、/usr/local　に、jdk1.2.2というディレクトリができていれば、インストールは完了です。

続いて、一般ユーザーとしてログインし、JDKのツールを使うためにパスを設定します。

```
$ PATH=/usr/local/jdk1.2.2/bin:$PATH
```

以降、いつでもこのパス設定を有効にしておきたい場合は、ホームディレクトリの.profile（または.bash_profile）に、この1行を書き加えてください（シェルがbashの場合）。

JDKが正しくセットアップされていれば、

```
$ java  -version

java version "1.2.2"
Classic VM (build 1.2.2-L . . . . )
```

と表示されます。

これでJDKの準備が整いました。いよいよ本題に突入です。

最初のJavaプログラム

ともかく、まずはJavaのプログラムに触れてみましょう。**リスト2**が一番簡単なJavaアプレットのプログラムです。

このリストを入力して、「app01.java」というファイル名で保存してください。ファイル名はかならずこの名前にする必要があります（理由は後述）。

アプレットビューア

アプレットは、最終的にHTMLとともにWebブラウザ上で実行するものですが、いちいちそのためのHTMLを書いて、Webサーバに転送してブラウザで確認するというのも大変です。そこで、アプレットビューアというJDKのツールを使って、アプレットの動作を確認しましょう。

アプレットビューアはXやGNOMEなどのGUI環境で起動しますが、まず、入力したプログラムをコンパイルする必要があります。

次のように入力してプログラムをコンパイルします。

```
$ javac app01.java
```

javacはJavaのプログラムを、実行可能なバイナリコードにコンパイルするためのツールで、app01.javaというプログラムファイルをコンパイルすると、app01.classというファイルが生成されます。

もし、プログラムに間違いがあれば、ここでエラーメッセージが表示されます。エラーの内容は少し難しいかもしれませんが、間違っている箇所も一緒に表示されますので、その部分をリストと見くらべてみて、正しい文字（記号）を入力してください。app01.classというファイルができあがっていればOKです。

コンパイルが無事完了したら、アプレットビューアで実行してみましょう。今度は次のように入力してください。

```
$ appletviewer  app01.java
```

画面4が実行画面です。アプレットビューアが起動する時に、エラーメッセージのようなものが何行も表示されるかもしれませんが気にしないでください。画面4のような画面が現れれば成功です。

アプレットビューア画面の最下行に表示されるメッセージは、アプレットビューア自身が出してくるレポートですので、プログラムで表示させたものではありません。リスト2で行ったのは、

```
Hello java world !
```

というテキストをアプレットビューアに表示させることです。

どうですか、同じ画面が表示されましたか。これが、あなたの第1号のJavaアプレットになります。

Javaアプレットの構造

では、もう一度リスト2を見てください。

冒頭の５行は、アプレットのために必要な記述です。

最初の2行で、実行に必要なライブラリを呼び出しています。1行目がアプレットとして必要なライブラリ、2行目がグラフィックス表示に必要なライブラリです。

```
import java.applet.*;
import java.awt.*;
```

今後、必要とする機能に応じて別のライブラリを呼び出すこともありますが、Webページ上でなんらかの表示を行うアプレットを作る場合は、最低限この2つのライブラリが必要になります。この2行は、いつもプログラムの最初に書くものだと憶えておいてください。

▼リスト2　app01.java

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="app01" width=200 height=200></applet>
*/

public class app01 extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello java world !",20,100);
    }
}
```

画面4　リスト2実行画面

／＊と＊／ではさまれた次の3行は、アプレットビューアで実行するためだけに必要な記述で、Webブラウザのみで実行するなら書く必要はありません。もともと、／＊と＊／ではさまれた部分は、コメントといってプログラム上無視される部分なので、javacがコンパイルするときにも一切無視されます。ただ、アプレットビューアがこのアプレットを実行するときに、画面の大きさをどれくらいにすればよいかという情報を与えるためだけに存在します（Webブラウザの場合はHTML内に記述します）。

ようするに、この5行はどんなアプレットを作る場合にも必要になる部分ですから、コピーするなどして必ずプログラムの最初に書いてください。

このアプレットは、画面の大きさを横200ピクセル×縦200ピクセルに設定しています。その記述が、<applet…の行の「width=200 height=200」という部分です。画面の大きさを変えたいときは、この2つの値を変えてください。

また、実行するアプレット名をcode="" の部分に記述します。今回は、app01.classですから、拡張子のclassをとって、「code="app01"」と書いています。app01がアプレット名になります。

先ほど、この先頭の5行はコピーなどしてと書きましたが、プログラムごとにこの画面サイズとアプレット名だけは書きかえるのを忘れないようにしてください。特に、アプレット名が違うとアプレットビューアは実行してくれません。

さて、いよいよ後半5行がJavaプログラム本体です。

プログラムをアプレットとして動かしたいときは、

```
public class ～ extends Applet {
```

の決まり文句で始まります。～の部分には、先程と同じくアプレット名を書きます。

classとありますが、Javaのプログラムはすべてこのclass（クラス）単位で作ることになっています。プログラム全体（アプレット自体）もクラスですし、その中に登場する部品や機能もクラスとして作ります。実行手順としてのプログラムと、データなどの区別をせず、すべてをクラスとして扱うという「オブジェクト指向」なのです。ここでは詳しくは説明できませんが、興味があればオブジェクト指向について学んでみてください。数々の書籍が出版されています。

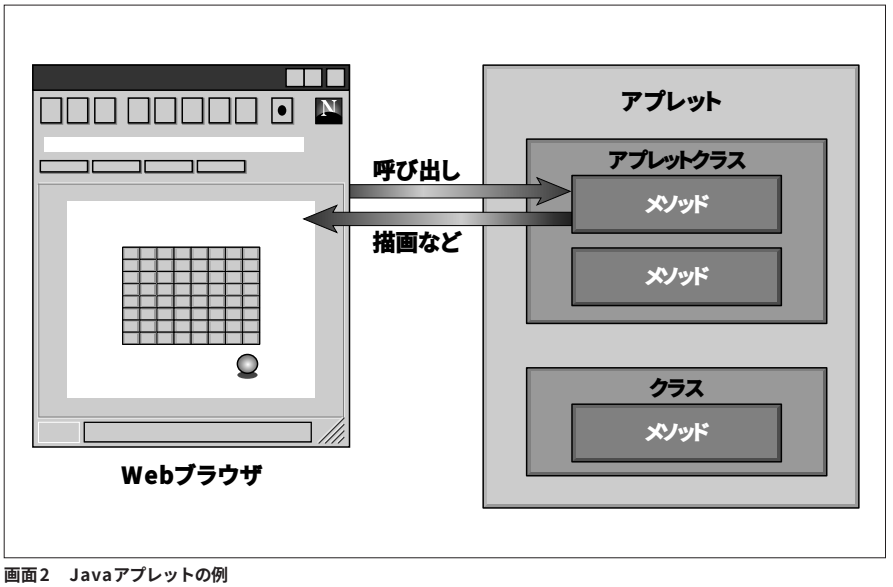
とにかく、このアプレット自体もクラスになります。このクラス（アプレット）はアプレットビューアやWebブラウザなど、外部から呼び出すことができなくてはいけませんので、publicと書いて「外部に公開するよ」と宣言します。

public class～と書けば、それはそれで立派なクラスなのですが、私たちが作りたいのはアプレットであり、アプレットはアプレットとしての基本的な機能や性質を持っていなければなりません。そのようなアプレットとしての共通機能は、すでにJavaの中に基本クラスとして用意されています。

これから作るアプレットクラスの基本部分は「標準のアプレットクラスと同じでいいよ」というのが、「extends Applet」という部分です。このAppletは、Javaが用意している基本アプレットのクラスで、今ここで作る新しいクラスは、基本アプレットのextend、つまり拡張ですよという意味です。

オブジェクト指向のプログラミングというのは、基本となるクラスに次々と機能を付け加えていって新しいクラスとし、またそのクラスを基本として新しいクラスを作るという手法なのです。

このAppletクラスのように、Javaには最初からあらゆる基本クラスが揃っており、あたり前の共通機能については改めてプログラミングする必要はありません。自分に必要な機能だけを書いて、基本クラスを拡張してゆく。これができればりっぱなオブジェクト指向プログラミングといえるでしょう。



画面2　Javaアプレットの例

以上ような理由から、今回のアプレットは、

```
public class app01 extends Applet {
```

で始まります。「アプレットの基本クラスを拡張して、app01というクラスを作り、それを外部に公開する」という意味になるのです。

今後、自分でプログラミングを続けてゆく場合でも、この読み替えのクセをつけておいてください。意味がわかっていれば、extendsやAppletを書き忘れるような、とんでもない文法上の間違いがなくなります。

次の行は、

```
public void paint(Graphics g) {
```

で、これはapp01クラスのメソッドといわれるもので、クラスに持たせたい機能をメソッドとして記述します。

通常、自分で作ったクラスは自分なりにどのような機能を持ってもよいのですが、Appletクラスだけは、アプレットとしてWebブラウザなどから実行される関係上、決まった機能を持つ必要があります。このpaintメソッドはアプレットの標準機能の1つで、画面を描画する機能です。

この行は、paintメソッドの始まりを表します。publicはクラスの時と同じ意味で、このメソッドが外部から呼ばれる必要がある場合に記述します。次のvoidは、メソッドを実行したあとに実行結果などのデータを返す必要がある場合、そのデータのタイプ（数値か文字かなど）を記述しますが、paintメソッドの場合、何も返さなくてよいので、void（何もなし）と書きます。これを戻り値といいます。

次にメソッド名、ここではpaintと記述します。その後には（ ）を必ず記述し、その中に呼び出す側から与えられるデータが何であるかを書きます。これは引数（ひきすう）とかパラメータと呼ばれますが、paintメソッドの場合これも決まっており、描画するための画面はどこなのかを指定します。この場合、画面そのものもクラスとして扱われますので、対象となる画面をGraphicsクラスとして与えてもらいます。

さて、これでpaintメソッドが用意できましたので、Webブラウザなどがこのアプレットを実行する際に、「画面を表示しなさい」という命令としてpaintメソッドが呼び出されます。

アプレットには、それ以外にも必要なメソッドがあるのですが、とりあえず特別なことをしなくてよいのであれば、先程の基本Appletクラスがすべてのメソッドを用意してくれていますので、そちらに任せることができるのです。つまり、他のメソッドについてはプログラムに何も書かなくても、自動的に基本Appletクラスのメソッドが実行されます。

実は、paintメソッドも基本Appletクラスに用意されてはいますが、基本AppletクラスのpaintメソッドはNo Imageを描画しないというメソッドです。

最後になりましたが、次がこのアプレットの動作です。

```
g.drawString("Hello java world !",
20,100);
```

これは、「Hello java world!」というテキストを表示するためのプログラムです。「g.drawString(.....);」が、「テキストを表示しなさい」というメソッドで、パラメータには、「表示するテキスト」「表示位置の横座標」「表示位置の縦座標」を指定します。

つまりこの例では、「画面上の（20,100）という座標にHello java world! というテキストを表示しなさい」という意味になります。

座標の数値を変えれば表示される位置が、テキストを変えれば表示される内容が変わりますので、ぜひ自分でいろいろと書き替えてから、実行してみましょう。

絵を描く

次は、画面にグラフィックスを描いてみましょう。

プログラムは**リスト3**です。入力して、app02.javaというファイル名で保存したら、app01の時と同じように

▼リスト3　app02.java

```
import java.applet.*;
import java.awt.*;
/*
        <applet code="app02" width=200 height=200></applet>
*/

public class app02 extends Applet {
        public void paint(Graphics g) {
                g.fillOval(40,100,30,30);
                g.fillOval(70,100,30,30);
                g.fillOval(100,100,30,30);
        }
}
```

javacでコンパイルしアプレットビューアで実行してみましょう（**画面5**）。

クラス、メソッドの宣言部分は、名前が違うくらいでリスト2とほとんど同じです。今回は、テキストでなく円を描画しています。

```
g.fillOval(40,100,30,30);
g.fillOval(70,100,30,30);
g.fillOval(100,100,30,30);
```

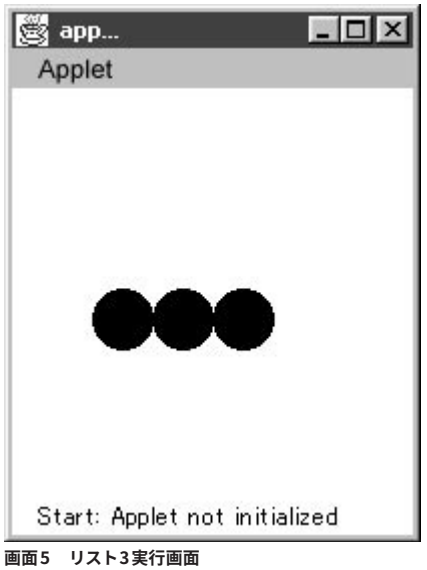
g.fillOval( )というのは、円または楕円を塗りつぶすメソッドです。パラメータは、「横座標」「縦座標」「横直径」「縦直径」になります。リスト3では位置を変えて、3個の円を描画しています。ここでも自分なりに数値を変えて、結果がどのように描画されるかを必ず試してみてください。

動きのあるアプレット

文字やグラフィックスを表示できても、表示して止まったままではプログラムとはいいがたいところです。まして、Webページでは静止した文字や画像などは、なにもアプレットに頼らなくてもHTMLで十分なのですから。

そこで、最後に本格的なアプレットを作ってみます。単純に動きのあるアプレットとはいっても、実は「動かす」というだけで、考えなければいけないことが結構出てきます。プログラムも、急に行数が増えますが、やっとプログラムらしくなったくらいに思ってください（**リスト4**）。

実行画面は載せてありませんので、ぜひとも皆さんのアプレットビューアで見てください。



画面5　リスト3実行画面

次回への宿題

Javaはプログラムのすべてがクラスで構成されていますので、プログラムの持つあらゆる機能は、必ずいずれかのクラスのメソッドになります。とすると、リスト2の「g.drawString(.....);」という「テキストを表示する」メソッドは、どのクラスのメソッドなのでしょうか。また、そのクラスはリスト2の中のどこに現れているのでしょうか。考えてみてください。

付録のCD-ROMに「3Dブロックくずし」が収録されています。Webサーバにコピーして遊んでみてください。なお、3DブロックくずしはWebサーバの設定に合わせて、Webサーバのルートディレクトリの「java3d」というディレクトリにコピーしてください（たとえば、/home/httpd/html/java3d）。詳しくは240ページからの付録解説を参照してください。

▼リスト4　app03.java

```
import java.applet.*;
import java.awt.*;
/*
        <applet code="app03" width=200 height=200></applet>
*/

public class app03 extends Applet implements Runnable {
        Thread          t;
        int             x = 0;

        public void init() {
                t = new Thread(this);              // スレッドを開始する
                t.start();
        }

        public void run() {
                try {
                        while(true) {
                                repaint();                // 再描画を要求する
                                Thread.sleep(100);        // 100ms制御を解放
                                x += 10;
                                if ( x > 200 )            x = 0;
                        }
                }
                catch(Exception e) {}
        }

        public void paint(Graphics g) {
                g.fillOval(x,100,30,30);
        }
}
```

Webサーバ構築術(第11回)

Apacheは、モジュール化という機能によって成り立っている。なにげなく使っているログの設定機能も追加されたもののひとつであり、Apacheのモジュールとして加えられている機能だ。今回はApacheのモジュール組み込みについて簡単に触れてみたい。

Apacheの機能を拡張する

文:中島昌彦
Text : Masahiko Nakajima

Apacheのモジュール機能

Apacheは、モジュールによってその機能が拡張されている。なにげなく使っているログや、MIMEによるファイルの関連付け、SSIといった機能も、Apacheのモジュール機能で追加されているものだ。

標準的に組み込まれている機能がApacheのモジュールのすべてではなく、モジュールとして組み込まれていないものの、ソースとして存在しているモジュールもApacheには多数ある。そうした中から有用なものを利用したり、Apacheプロジェクト以外で開発されたモジュールを組み込むことで、さらにApacheは拡張されていく。

かつては、モジュールの追加というと、コンフィグレーションファイルを修正し、再度makeし直すという作業が必要で、とっつきにくい機能拡張であった。しかし、Apache 1.3以降はDSO(Dynamic Shared Object)という方法も使え、イメージ的にはサーバに組み込むというよりも必要なときに読み込む形式になっている。まさにモジュールという名前にふさわしいものになってきた。

さらには、DSOを使うことで、Apache本体の再コンパイルが不要となり、モジュールも利用しやすくなったといえる。これまでなら、動作させているサーバの環境を考え、不要なモジュールはできる限り組み込むべきではなかったが、DSOならモジュールを利用するまではサーバのリソースを消費しない。セキュリティ面や安定動作を考えると、公開サーバにはできる限り不要なモジュールを組み込まないというポリシーは生き続けるべきであるが、テスト環境であれば、あまり深く考えずに手軽にモジュールを組み込み評価できるようになっている。

また、Apache本体にモジュールを組み込む方法として、configureプログラムが動作環境を調べて適切なMakefileを作成する、APACI(Apache Autoconf-style Interface)がApache 1.3から導入された。統合化されたコンフィグレーションインターフェイスであるAPACIを使えば、モジュール追加のためにコンフィグレーションファイルを修正するといったややこしい作業が不要になっている。

APACIやDSOを使うことで、Apacheのモジュール追加はソースファイルさえあれば比較的手軽に実行できるものとなっている。

DSOで組み込むのは便利だが、何か問題が生じたときに、原因の追求に手間取ることになる。また、多くのモジュールを組み込むことは、セキュリティホールを増やすことにつながるかもしれない。結局、DSOでテストを行い、公開サーバは安全性を重視してAPACIを使って必要なモジュールだけを組み込むのがよいだろう。

標準搭載モジュールの追加

Apacheでごく普通に利用されている機能は、モジュールという形で用意されている。このあたりをいちど見なおしてみよう。

この作業を始める前に、最新のApache（http://www.apache.org/dist/から、近所のミラーサイトを選択するといい）を入手する。現在の最新バージョンは1.3.12なので、今回はこのバージョンをベースにしていこう。いろいろと試したところ、既存のApache1.3.xベースのものに手を加えていたりすると、モジュールの追加がうまくいかないケースがあるので、今回はまっさらな環境で組み込むことにする。

Apacheのソースファイル一式が/usr/local/src/apache_1.3.12/以下にあるとすると、その下のsrc/Configurationでモジュール組み込みを設定している。けっこう巨大なファイルなので、AddModuleという名前をgrepコマンドで検索してみよう。

```
# AddModule モジュールオブジェクト
```

と書いてあるような行がいくつも見つかるはずだ。行頭が#でコメントアウトされているものは、モジュールとして組み込まれていない機能だ。逆にいってしまえば、ほとんどのモジュールがApacheには組み込まれていないことに気づくだろう。

かつてなら、組み込みたいモジュールがあれば、コンフィグレーションファイルを修正し、行頭のコメントアウトを外し、モジュールのmake、インストールといった作業をしていかないといけなかったが、せっかくAPACIがあるのだから、APACIを使ったインストールをしたい。

mod_rewriteをAPACIで組み込む

mod_rewriteというモジュールがある。何をするものかといえば、Apacheに渡されたURLを元に、特定の処理をして、新たなURLを戻すというものだ。リダイレクトやエイリアスのもっと便利なものと考えるといい。これは、Apacheプロジェクトの中で作られていて、Apacheのソース一式の中に組み込まれている。しかし、デフォルトではApacheのモジュールとして有効になっていない。有効になっていないとはいえ、モジュールとして組み込まれるケースが多いと思われるモジュールだ。mod_rewriteをベースに、標準的なモジュールの追加作業を追ってみよう。

作業として、Apacheのソースディレクトリに移動して、APACIを使ってApacheをビルドする。

```
# cd /usr/local/src/apache1.3.12/
# ./configure --enable-module=
rewrite
# make
# make install
```

作業は、4つのコマンドを実行するだけで、コンフィグレーションファイルの修正や環境ファイルの修正作業はまったく不要だ。これだけ？とあっけにとられるほど手軽だ。makeという作業が加わっているのでそれなりに時間はかかるが、mod_rewriteは確実に組み込まれている。

どんなことをしているかといえば、./configureすなわちAPACIを使い、rewriteモジュールを組み込んでいる。ただし、これはDSOとしてではなく、あくまでもモジュールとしての組み込みだ。DSOで組み込む場合は、--enable-sharedというオプションを使う。

make、make installで、そのあとにApacheを再ビルドしてモジュールを組み込んでいる。

本当にmod_rewriteが動いているかどうか確認するために、次のように行ってみよう。

1.httpd.accessのAllowOverrideを修正する

AllowOverrideは、ドキュメントディレクトリ中の.htaccessによる制御を受け付けるかどうかの指定だ。テストのため、/usr/local/apache/conf/httpd.conf中にある“ <Directory "/usr/local/apache/htdocs"> ”の16行下にある、AllowOverrideの指定を、

```
AllowOverride All
```

としておく。

2.ドキュメントディレクトリ中に.htaccessを記述する

mod_rewriteによる制御をしたいドキュメントディレクトリ上に、次のような.htaccessを記述して置いておく。

```
RewriteEngine On
RewriteRule                    (.+)
http://www.testserverB.com/$1 [L,R]
```

RewriteRule行のhttp://...のホスト名は、自分の環境のテストサーバや公開サーバにしておく。

たとえば、http://www.testserverA.com/のドキュメントルートに記述した場合、testserverA.comへのアクセスは、mod_rewriteによってhttp://www.testserverB.com/へリダイレクトされ、ビジターのWebブラウザには、新たなURLが表示される。mod_rewriteをわざわざ使わなくとも、RedirectMatchを使えば済む話だが、mod_rewriteはRedirectMatchにはない高度な指定ができるので、さらに複雑なRewrite条件が指定できる。

デフォルト組み込み以外のモジュールを利用する

mod_rewriteは、Apacheが標準で持っている機能であったが、それ以外のモジュールを導入する場合は、もう少しだけ手間がかかる。

Perlで書かれたCGIを使うなら一般的ともいわれているmod_perlを組み込むケースで解説しよう。mod_perlは、Perlを呼び出さずに処理するもので、PerlスクリプトをApache内部で処理する。CGIが呼び出されるたびに、いちいちPerlが呼び出されて処理をするということがなくなるため、サーバの負荷を下げられる。しかも、スクリプトのバイナリをキャッシングしているため何度も翻訳することがなくなる。CGIの環境によってはかなり明確な速度差が出てくることだろう。

mod_perlを使うために、当然だがmod_perlのソースが必要だ。これは、The ApachePerl Integration Project（http://perl.apache.org/dist/）から入手しよう。今回はmod_perl-1.23を使っている（最新版のmod_perlは3月16日に1.24になっている）。

入手したmod_perl-1.23.tar.gzは、/usr/local/src/に置き、そのまま展開する。

```
# tar xvfz mod_perl-1.23.tar.gz
# cd /usr/local/src/mod_perl-1.23
# perl Makefile.PL APACHE_SRC=../
apache_1.3.12/src DO_HTTPD=1 USE_
APACI=1 PREP_HTTPD=1 EVERYTHING=1
```

この作業で、環境に合わせたMakefileができあがる。引き数の内容は、APACHE_SRCでApacheのソースディレクトリを、DO_HTTPD=1で、APACHE_SRCを利用する指定だ。USE_APACI=1にすることで、Apacheの新たな環境構築インターフェイスであるAPACI（Apache Autoconf-style Interface）を利用する指定となる。PREP_HTTPD=1は、APACHE_SRCで指定したディレクトリの下のmodules/perlにApacheモジュールを構築するという指定だ。最後のEVERYTHING=1は、すべてのモジュールを組み込む指定となる。

このあたりの詳細な解説は、mod_perlディレクトリ中のINSTALL.apaciの中で解説されているので、そちらを参照してほしい。

mod_perlのインストール

mod_perlのMakefileができあがるので、このまま、

```
# make
# make install
```

として、mod_perlをインストールする。ただし、このままでは、mod_perlのインストールが終わり、必要なMakefileをApacheのソースディレクトリに作っただけで、ApacheのMakeはできていない。そこで、

```
# cd ../apache1.3.12
# ./configure --activate-module=
src/modules/perl/libperl.a
# make
# make install
```

という作業が必要だ。

ここまでの作業では、mod_perlは組み込まれているものの、動くような状態にはなっていない。そこで、httpd.confを修正し、動作状態を確認するCGIを動かしてみよう。

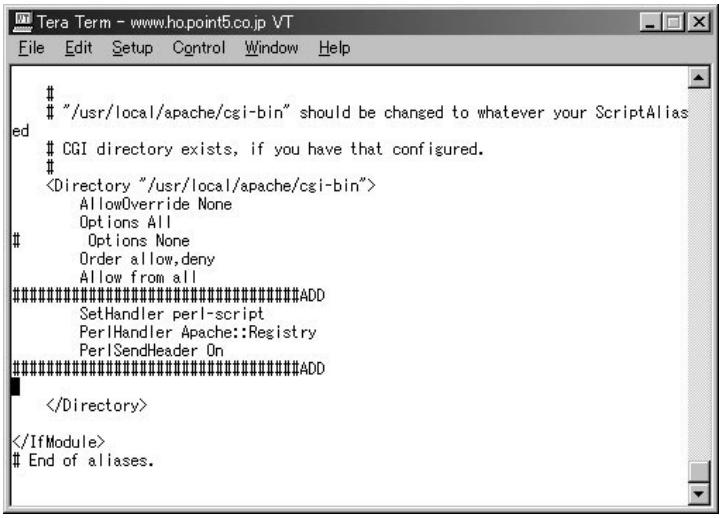
手っ取り早く動作確認するために、/usr/local/apache/cgi-binのディレクトリを使うことにする。手順としては、次のようになる。

1.httpd.confの編集

/usr/local/apache/conf/httpd.confのファイルを開き、mod_perlが有効になるように、“ <Directory "/usr/local/apache/cgi-bin"> ”の行を探す。このDirectory中で指定している「Options None」を「Options All」に変更し、ExecCGIを有効にする。mod_perlでは、ExecCGIになっていないといけない。

2.httpd.confにハンドラを設定

さらに次の3行を追加する（**画面1**）。



画面1
httpd.confを修正し、mod_perlハンドラを設定

▼リスト1　test.cgi

```
#!/usr/bin/perl

print "Content-type: text/plain\n\nCGI is running ";
if ($ENV{MOD_PERL}) {
        print("under a $ENV{MOD_PERL}.");
} else {
        print("under a perl.");
}
```

```
SetHandler perl-script
PerlHandler Apache::Registry
PerlSendHeader On
```

この設定によって、/cgi-binディレクトリにはmod_perlのハンドラが動く。

3.動作確認

動作確認のCGIは、**リスト1**（test.cgi）のようなものだ。これを、test.cgiという名前で、/usr/local/apache/cgi-bin/に保存する。ファイルのパーミッションは、0755にしておく。mod_perlを組み込まずに、http://＜サーバ名＞/cgi-bin/test.cgiへアクセスすれば、**画面2**のような表示になり、正しくmod_perlが組み込まれていれば、**画面3**のような表示となる。

正しく動作しているようならば、cgi-bin以外に置いているCGI中で、こうした設定をすればいい。ちなみに、PerlHandlerは.htaccess中で記述できるので、httpd.confのallowOverrideの設定をしっかりしていれば、好きな場所で自由に利用できるようになる。

複数のモジュールを有効にする

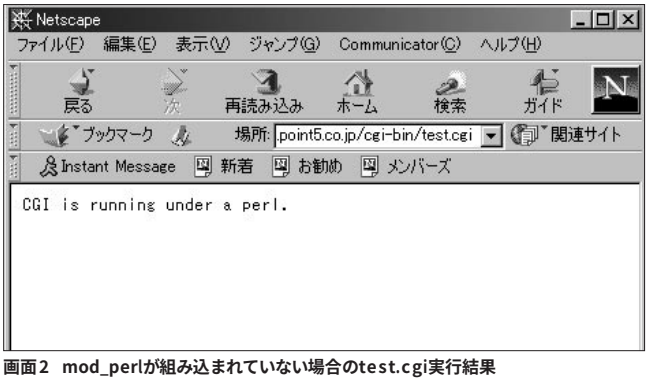
この記述どおりに作業をしていくと、最初はmod_rewriteが正しく動くが、mod_perlを組み込んだ段階で、インストールしたはずのmod_rewriteが使えなくなる。現状のAPACIは、そういう仕様だ。ほとんどのWebサーバは、1回インストールしたら、セキュリティホールが見つかってOSをアップデートするとか、機能拡張が必要になったりするまでそのまま運用するのが普通で、モジュールの差し替えを年中するようなものではない。そういう意味では、インストールの手間が省けるAPACIの現仕様は、それなりに意味のあるものになっている。

では、複数のモジュールを加えるときにどうするのだろうか。今回のmod_rewriteとmod_perlの2つを組み込むとき、もっとも手間をかけずに確実なのは、mod_perlしか有効にならなくなったあとに、次のようにmod_rewriteを組み込むことだ。

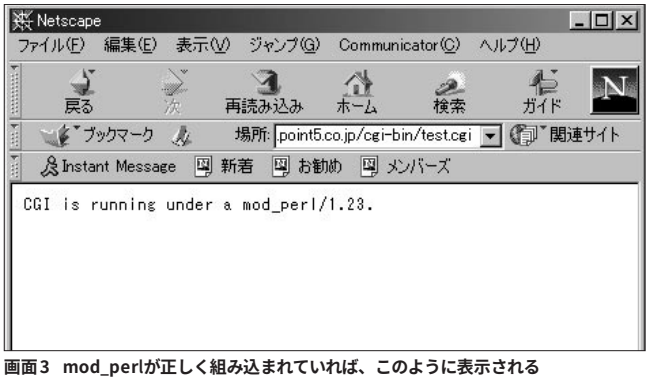
```
# cd /usr/local/src/apache1.3.12
# ./configure --enablemodule=rewrite
--activatemodule=src/modules/perl/
libperl.a
# make
# make install
```

として、./ configureのあとに必要なモジュールを指定する。

すでに動いているApacheにモジュールを追加するというときには、すでに組み込んだモジュールがなんであるかわからないと作業が進まない。そんなときは、httpd -lを使う。たとえば、mod_rewriteとmod_perlを組み込んでいる状態ならば、**画面4**のように表示され、モジュール一覧を参照できる。ただし、運用中のサーバにモジュールの追加をすることはかなり危険だ。あくまでも一度停止させてメンテナンス中に作業するか、別サーバにセットアップすることを勧めたい。なお、APACIのconfigureに--helpとすると、--disable-moduleのあとにモジュールの組み込み状態のようなものが表示される（**画面5**）。しかし、この内容と現実は一致していないようなので、何が組み込まれているかは、httpd -lで確かめるほうが確実そうだ。



画面2　mod_perlが組み込まれていない場合のtest.cgi実行結果



画面3　mod_perlが正しく組み込まれていれば、このように表示される

mod_perlを使ってApacheにPerlモジュールを加える

mod_perlの隠れた効用というか、優れた機能として、PerlでApacheのモジュールを記述できるという機能がある。Apache::RegistryというPerlのモジュールを使うことで、そのAPIを実現する。http://perl.apache.org/からPerl Apache Modulesをクリックすると、詳細の情報を参照できるので、気になる人はそちらを参照するとよいだろう。いくつか試してみたが、これこそはというようなPerl Apacheモジュールは見当たらなかったので、詳細には触れない。ただ、Apacheのモジュールを作りたいという人にとって、C言語でなくても作れるという点は、メリットといえそうだ。

mod_perlがあるならEmbperlも

mod_perlと比較してよく出てくるものが、Embperlである。HTML内にPerlコードを書き込めるもので、Javaでいえば JSPにあたるもののPerl版だ。PHP3使いの人にとってみれば、HTML中にEmbedできることは、完成品ができあがるまでの時間短縮につながる。

mod_perlを組み込むならば、いっそついでにEmbperlもという人には、http://perl.apache.org/embperl/からHTML-Embperl-1.2.1.tar.gzを入手してmakeするといい。

手順は、Embperlのドキュメントどおりでうまくいく。

```
# perl Makefile.PL
# make
# make install
```

でできるが、この方式を使うと、mod_rewriteが使えなくなるので、mod_perlのときと同じように、改めてconfigureをする。

```
# cd /usr/local/src/apache1.3.12
# ./configure --enable-module=rewrite
--activate-module=src/modules/perl/
libperl.a
# make
# make install
```

とすると、mod_rewriteとmod_perl、Embperlの共存がうまくいく。

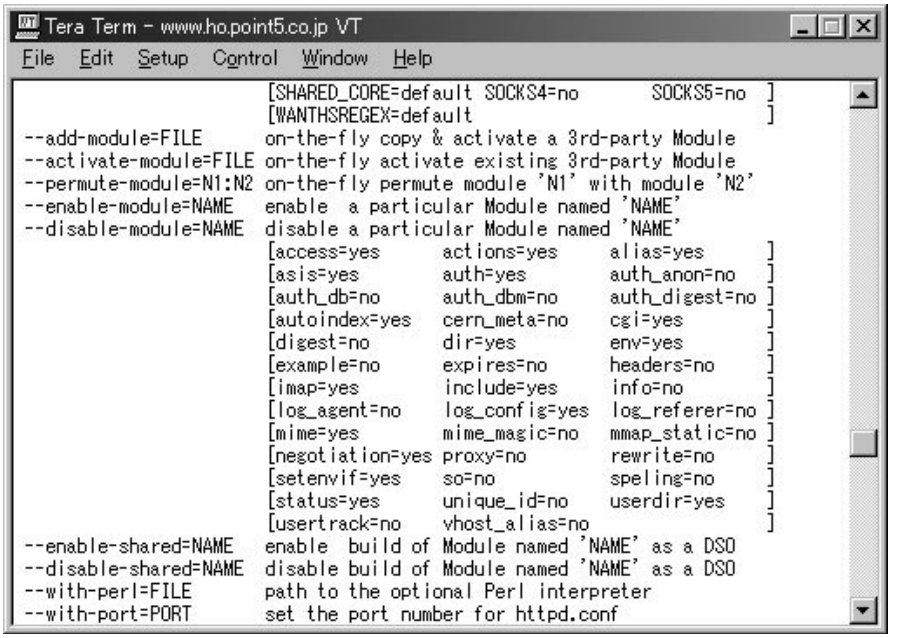
さらに、Embperlを動かすディレクトリ上に.htaccessとして次のファイルを作っておく。

```
SetHandler  perl-script
PerlHandler HTML::Embperl
Options     ExecCGI
```

Embperlを利用した例を示そう。**リスト2**のindex.htmlは普通のHTMLで、FORMを利用して変数に値を入力している。送信ボタンを押すことで、**リスト3**のget.htmlを呼び出している。Embperlで拡張された点を利用しているのは、“ [+$fdat{name}+] ”といった行で、先ほど入力した変数の値に置き換えられる。

```
# httpd -l
Compiled-in modules:
  http_core.c
  mod_env.c
  mod_log_config.c
  mod_mime.c
  mod_negotiation.c
  mod_status.c
  mod_include.c
  mod_autoindex.c
  mod_dir.c
  mod_cgi.c
  mod_asis.c
  mod_imap.c
  mod_actions.c
  mod_userdir.c
  mod_alias.c
  mod_rewrite.c
  mod_access.c
  mod_auth.c
  mod_setenvif.c
  mod_perl.c
suexec: disabled; invalid wrapper
/usr/local/apache/bin/suexec
```

画面4 Apacheに組み込まれているモジュールを表示



画面5 APACIのヘルプ出力（./configure --help）

ブラウザで実行すると**画面6、7**のようになる。単純に実行画面だけを見ている限りでは、特別なことをしているようには見えないが、FORMの値の受け渡しをHTMLレベルで制御でき、FORMのクエリーをそのまま変数として受け取れる。このあたりはPHP3と同様に便利な点だ。

Embperlは、その文法さえ覚えればかなり役立ちそうだが、HTML中にEmbedするにはかなり癖がある。また、現在のバージョンでは、日本語の引き渡しがうまくいかない。2バイトコードを使う環境での導入はまだ早そうだ。

▼リスト2　index.html

```
<HTML>
<HEAD>
<TITLE>HTML POST</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=EUC-JP">
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<FORM METHOD="GET" ACTION="get.html">
  <TABLE BORDER="1" CELLPADDING="0">
    <TR>
      <TD WIDTH="100">名前</TD>
      <TD WIDTH="230">
        <INPUT TYPE="text" NAME="name" MAXLENGTH="30"
SIZE="30">
      </TD>
    </TR>
    <TR>
      <TD WIDTH="100">年齢</TD>
      <TD WIDTH="230">
        <INPUT TYPE="text" NAME="year" MAXLENGTH="3"
SIZE="3">
      </TD>
    </TR>
    <TR>
      <TD WIDTH="100">郵便番号</TD>
      <TD WIDTH="230">
        <INPUT TYPE="text" NAME="post" SIZE="8"
MAXLENGTH="8">
      </TD>
    </TR>
  </TABLE>
  <INPUT TYPE="submit" NAME="Submit" VALUE="送信">
</FORM>
</BODY>
</HTML>
```

▼リスト3　get.html

```
<HTML>
<HEAD>
<TITLE>HTML POST</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=EUC-JP">
</HEAD>

<BODY BGCOLOR="#FFFFFF">
  <TABLE BORDER="1" CELLPADDING="0">
    <TR>
      <TD WIDTH="100">名前</TD>
      <TD WIDTH="230">
        [+$fdat{name}+]
      </TD>
    </TR>
    <TR>
      <TD WIDTH="100">年齢</TD>
      <TD WIDTH="230">
        [+$fdat{year}+]
      </TD>
    </TR>
    <TR>
      <TD WIDTH="100">郵便番号</TD>
      <TD WIDTH="230">
        [+$fdat{post}+]
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

画面6　Embperlを使った例（リスト1のFORMで値を入力する）

画面7　送信ボタンでget.htmlを呼び出すと、入力された値をHTML上で利用できる

プログラミング工房

Xのプログラミング解説として3回目となる今回は、GTK＋を例に「ツールキット」の本質とは何であるかを解説したあと、先月号で説明できなかったX Window Systemのメカニズムを詳しく見ていく。

第8回　Xのプログラミング（3）

文：藤沢敏喜
Text : Toshiki Fujisawa

今月はXlibプログラミングの第3回目である。最初の回で書いたように、この3回の連載ではXの実践的なプログラミングの解説を第一目的とするのではなく、ネットワークを通してやり取りされるXプロトコルの基本概念や、ツールキットの概念など、X Window Systemの設計思想の理解を目的としている。

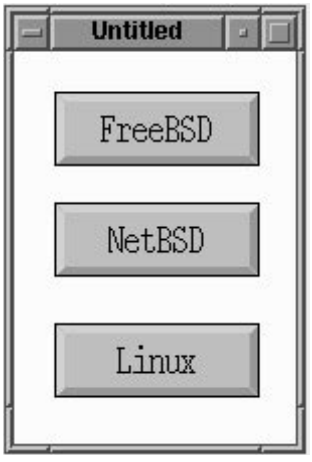
これらの概念は、自分ではプログラムを書かずに、単にXを使うだけのユーザーにとっても、知っておくと非常に役に立つ知識である。そして、X Window Systemの歴史とそのオープンな思想を理解することは、GNUソフトウェア、BSD、Linux、そしてGimpをはじめとするすばらしいソフトウェア文化のより深い理解にもつながるのではないかと思う。

また、Xがほかのウィンドウシステムと違って、高パフォーマンスで動作できるように設計されているということは、LinuxやBSDを他人に勧めるうえでも、有力な説得材料となるであろう。

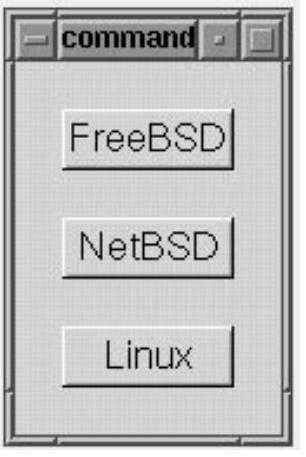
先月号のサンプルプログラムをGTK+で記述

先月号では、Xの基本であるXlibを直接扱い、**画面1**のようなサンプルプログラムを作成したが、今月号ではこのプログラムと同様な動作をするものをGTK+を使って書いてみる。

GTK+とは、画像を扱うソフトとして有名なGimpを開発するために作成された「ツールキット」である。なお、



画面1
Xlibを使用したボタンを表示するプログラム。先月号の付録CD-ROMに収録されたプログラムを「make 3」でコンパイル・実行した結果の画面。



画面2
GTK＋を使用したボタンを表示するプログラム。今月号の付録CD-ROMに収録されているgtktest.cを「make」でコンパイル・実行した結果の画面。

今回GTK+のプログラムを解説するのは、先月号で解説したXlibを直接使ったプログラムと比較することにより、Xlibとツールキットの関係についての理解をより深めるためである。

今月号のGTK+サンプルプログラムを実行するには、ルート権限で付録CD-ROMをマウントし、一般ユーザーでログインしてから本連載のプログラムがあるディレクトリからgtktest.cとMakefileをユーザーのホームディレクトリにコピーし、

```
$ make
```

を実行すると、プログラムが自動的にコンパイルされ、**画面2**のようなウィンドウが現れる。

なお、このプログラムは、Red Hat Linux 6.2（英語版）をフルインストールした状態で動作を確認した。GTK+がインストールされていないシステムでは、GTK+をあらかじめインストールしておく必要がある。また、FreeBSDなどではMakefileのgtk-configの部分をgtk12-configとする必要がある場合もある。

GTK+で書いたサンプルプログラムの概略

■ベースとなるウィンドウを開く

先月号のXlibを直接呼び出すサンプルプログラムでは、XOpenDisplayでXサーバに接続して、XCreateSimple

▼リスト1　GTK+でのbase_window_open関数

```
#include <stdio.h>
#include <gtk/gtk.h>
/*------------------------------------------------------------------------*/
static GtkWidget *
base_window_open(void)
{
    GtkWidget *root_window;
    int    argc;
    char **argv;

    argc    = 1;                                   ← 本来はコマンドラインで与えられた数
    argv[0] = "command_name";                      ← コマンド名だけをダミーで渡す
    gtk_init(&argc, &argv);                        ← gtkライブラリの初期化
    root_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);   ← ベースとなる新しいウィンドウを開く
    gtk_signal_connect(GTK_OBJECT(root_window),
               "destroy", GTK_SIGNAL_FUNC(gtk_main_quit), NULL);   ← ウィンドウを閉じる操作
    gtk_container_border_width(GTK_CONTAINER(root_window), 20);   ← gtkを終了するコールバック関数
    gtk_widget_show(root_window);                  ← ボーダー幅を20ドットに設定
                                                   ← ウィンドウを表示
    return root_window;
}
```

▼リスト2　vertical_box_open関数

```
static GtkWidget *
vertical_box_open(GtkWidget *root_window)
{
    GtkWidget *vertical_box;

    vertical_box = gtk_vbox_new(TRUE, 20);         ← 垂直ボックスの作成
    gtk_container_add(GTK_CONTAINER(root_window), vertical_box);   ← ベースウィンドウに垂直ボックスを載せる
    gtk_widget_show(vertical_box);                 ← 垂直ボックスの表示
    return vertical_box;
}
```

Windowでウィンドウを開いた。またXmapWindowやXSelectInput関数などのXlib関数を用いてベースとなるウィンドウを開いた。

一方、GTK+でのbase_window_open関数は、**リスト1**にあるように定義される。ここではgtk_init関数でGTK+ライブラリを初期化し、Xサーバとの接続などを行う。そして、gtk_window_new関数でベースウィンドウを作成し、gtk_signal_connect関数でウィンドウが閉じられたときの動作を登録し、gtk_container_border_width関数でウィンドウのボーダー幅を設定する。最後にgtk_window_show関数でウィンドウを表示している。

■垂直ボックスの作成

GTK+には、「垂直ボックス」という概念がある。これは、ボタンなどを縦に均等に見栄え良く並べることを目的としたもので、gtk_vbox_new関数を使って「垂直ボックス」を作成し、gtk_container_add関数を使ってその「垂直ボックス」にボタンを詰め込んでいくことになる。

これを行うのが、**リスト2**に示すvertical_box_open関数である。

■ボタンの作成

先月号のXlibを直接使ったサンプルプログラムでは、ボタンを表示するために、button_openという関数を定義し

▼リスト3　button_open関数

```
static void
button_open(GtkWidget *position, char *text, void (*func)(void) )
{
    GtkWidget *button;

    button = gtk_button_new_with_label(text);
    gtk_box_pack_start(GTK_BOX(position), button, TRUE, FALSE, 0);
    gtk_signal_connect(GTK_OBJECT(button),
                "clicked", GTK_SIGNAL_FUNC(func), NULL);
    gtk_widget_show(button);
}
```

新しく作成するボタンに表示する文字列
垂直ボックスにボタンをパック
クリックされたときに呼ぶコールバック関数
マウスクリック

▼リスト4　main関数

```
static void func_0(void) { printf("push FreeBSD\n"); }
static void func_1(void) { printf("push NetBSD\n");  }
static void func_2(void) { printf("push Linux\n");   }
/*------------------------------------------------------------------------*/
int
main(void)
{
    GtkWidget *root_window;
    GtkWidget *vertical_box;

    root_window  = base_window_open();

    vertical_box = vertical_box_open(root_window);
    button_open(vertical_box, "FreeBSD", func_0 );
    button_open(vertical_box, "NetBSD",  func_1 );
    button_open(vertical_box, "Linux",   func_2 );

    gtk_main();
    return 0;
}
```

垂直ボックスの作成
イベント待ちや再描画などすべての処理を行う

て、

```
button_open(20, 20, 100,35, "FreeBSD", func_0 );
button_open(20, 75, 100,35, "NetBSD",  func_1 );
button_open(20,135, 100,35, "Linux",   func_2 );
```

というようにして、ボタン左上の座標とボタンの幅および高さの座標を引数に指定して、表示を行った。

今回記述したGTK+の場合も、同様にbutton_open関数を定義して、

```
button_open(vertical_box, "FreeBSD", func_0 );
button_open(vertical_box, "NetBSD",  func_1 );
button_open(vertical_box, "Linux",   func_2 );
```

というようにボタンを表示している。なお、ここで指定するvertical_boxという名前の引数が、前述した垂直ボックスを示す変数である。

このbutton_open関数は、**リスト3**に示すように定義され、gtk_button_new_with_label関数の引数としてボタンに表示する文字列を与えて呼び出すことにより、ボタンを生成している。そして、gtk_box_pack_start関数により、「垂直ボックス」に生成したボタンを上から順に並べていく。また、gtk_signal_connect関数によりボタンが押されたときに呼び出すべきコールバック関数を定義している。そして最後に、gtk_widget_show関数を使ってそのボタンを表示している。

■main関数とgtk_main関数

リスト4に示すのがmain関数である。ここでは、上で定義したbase_window関数、vertical_box_open関数、そしてbutton_open関数を使って3つのボタンを表示している。

最後に、gtk_main関数が呼び出されている。この関数が、上記で定義したボタンが押されたときの処理を一手に引き受けている。すなわち、マウスやキーボードのイベントが発生したら、そのイベントに対応するコールバック関数を呼び出すなどの仕事をしているわけである。

また先月号で解説した再描画イベントなどの処理も、このgtk_main関数の中で自動的に処理されることになる。

ツールキットとは何か?

先ほど述べたように、GTK+は「ツールキット」であるが、ツールキットとは具体的には何を指すものなのであろうか？　先々月号の本連載では、

「ツールキットライブラリではボタンや、スクロールバー

Column

WindowsとXの両方で使えるツールキット

筆者は、1996年頃にFreeBSDとLinuxの上で動作するXプログラムを作成したことがある。当時は、GTK+などのように強力で格好の良いツールキットは存在しなかったし、FreeBSDやLinux対応の商用Motifを使うことも難しかった。また、最終的にはWindows 95でも同じソースコードを使って動作させることをも視野に入れていたため、結局独自のツールキットを作成せざるをえなかったのである。

たとえば、Windows 95でボタンを表示する場合、Win32ライブラリ関数を次のように用いている。

```
CreateWindow("button",  text,
      WS_CHILD | WS_VISIBLE | WS_BORDER
      | BS_PUSHBUTTON | BS_VCENTER,
      x, y, w, h, (HWND)hImageDlg,
      (HMENU)(id + IDX_OFFSET),hDSInst, NULL );
```

極端な話として、Xlibを使えばこの関数をそのままコンパイルできるようなXツールキットを作成することも不可能ではない（このようなWin32やMFC互換のフリーなXツールキットがあれば、過去にWindows用に開発されたソフトのXへの移植が進むかもしれない）。

しかしながら、Windows 2.0の時代から増築に増築を重ねたためか、Win32ライブラリやMFCはむやみやたらと数が多く、洗練されているとはいいがたい。したがって、完全互換のライブラリを作成するにはかなりの労力を必要とする。

そこで、筆者が作成したツールキットでは、XとWin32の両方で使えるような関数を実装することにした。このようなアプローチは実装が楽で、両方のOSで性能を落とさずに済むため、最近出現したいくつかのツールキットでも採用されているようである。

なお、独自にツールキットを作成したため、このプログラムで必要とするのはlibc.aとlibX11.a（Xlib）の基本関数だけとなり、ライブラリの互換性問題にもあまり悩まされずにプログラムを保守できるというメリットもあった。

などの部品が用意されていて、簡単なプログラムで、その部品を使用することができるようになっている」

と解説したが、これだけでは具体的にはどんなものなのかが、よくわからなかった読者も多かったと思う。

しかしながら、ここまでの解説でウィンドウを開くgtk_window_new関数や、gtk_button_new_with_label関数、そしてgtk_main関数などを集めたものが、GTK+ツールキットの実体であることがわかっていただけたのではないだろうか。もちろん、これらの関数の内部ではXlib関数が使われていて、先月号で解説したプログラムと同じような実装が行われてるはずである。

つまり、先月号でのサンプルプログラムのように、Xlibを直接使って、base_window_open関数やbutton_open関数を定義することが、ツールキットを作成するということにほかならないのである。

すべての基本はXlib

Xのツールキットとして、Motifなどいくつかのものが存在していたにもかかわらず、Gimpを作成したメンバーは自分達で独自にGTK+というツールキットを新規に開発した。なぜならばMotifなどのツールキットの仕様やライセンス条件などが、彼らの目的と合致しなかったためである。

しかしながら、GTK+もまた万能ではなく、目的によっては使いにくいことがある。たとえば、ボタンなどの見た目を好きなように変えたい場合で、オブジェクトの継承での実現が難しいような、まったく新しい概念の部品を追加したいときには、GTK+自体の改造が必要になる。

また、LGPLライセンスを適用したくない商用プログラムを作成したいような場合、GTK+を使うことは難しいと考えられる（再リンク可能なオブジェクト単位ではなく、GTK+を再リンク不可能な単一の実行ファイルのみとして配布したい場合）。

このような場合でも、Xlibの知識があれば、GTK+を自分の好きなように改造することができるし、GTK+とはまったく別の独自の機能やライセンスを持つツールキットを作成することもできる。もしかすると、数年後には、GTK+よりももっと優れたツールキットが開発されるかもしれない。しかし、Xlibの知識はそのときでも無駄になることはないであろう。

Xlibのさらなる理解

さて、ツールキットの本質がどんなものであるかがわかったところで、先月号で解説できなかったXlibのいくつかの概念について説明を続けよう。

■Pixmap（グラフィックイメージデータの入れもの）

先月号で述べたように、ボタンを実現するための情報として、次のような構造体を定義した。

```
typedef struct {
        Pixmap      pixmap;
        Window      window;
        int         width;
        int         height;
        void        (*callback)(void);
```

Column

VNCを使った遠隔操作とXプロトコル

読者の方には、VNCという名前のソフトウェアをインストールしたことがある方も多いのではないかと思う。このソフトは、遠方にあるWindowsマシンなどの画面を、ネットワークを通して操作するというGPLライセンスのソフトウェアで、下記のURLで配布されている。

http://www.uk.research.att.com/vnc/

VNC自体はたいへんすばらしいソフトウェアであるが、肝心のWindowsには、Xのようにネットワークを通して効率よく描画するという概念がないため、VNCではWindowsの画面イメージをそのまま転送するしか方法がない。そのため、フォントも再描画データもすべてイメージデータとして毎回転送されるので、1.5Mbps程度の回線で遠方のWindowsを操作すると耐えがたいくらいに時間がかかってしまう。

イライラしてあまり実用にならないWindowsの遠隔操作と、Xプロトコルを使った快適な遠隔操作を実際に体験してみれば、X Window Systemの設計がいかに優れているかを実感することができるはずだ。

X Windows Systemは、Microsoft Windowsの出現よりも、はるか前に開発されたことを考えると、その先見性には本当に驚かされる。Xプロトコルの詳細を知れば知るほど、多くの優秀な人がオープンな開発を行うことのすばらしさを感じることができるだろう。

```
} button_t;
```

ここで定義されているPixmap型の変数pixmapは、先月解説したようにボタンのグラフィックデータを保存する変数である。この変数に保存するpixmapは、

```
XCreatePixmap(base_disp, window, w, h, depth)
```

という関数で作成することができる。ここで、w, h, depthは、グラフィックデータの幅と高さおよび色数を示し、base_dispは、接続しているXサーバを示す識別子である。この場合、window識別子で示されるのと同じスクリーンにグラフィックデータを保持することになる。

ここで重要なのは、このpixmapに入れるイメージデータは、Xクライアントが実行されているマシンではなく、Xサーバが実行されている側のマシンへ保存されるということである。

先月号で解説したように、ウィンドウが隠された場合には再描画する必要がある。しかし、大きなグラフィックデータを再描画のたびにネットワークを通して転送していては、たいへん効率が悪い。ところが、Xサーバが動作しているマシンにpixmapとして保存しておけば、再描画の際には、Xサーバを動かしているマシンの中でのメモリ転送だけで済むことになる。

したがって、ボタンの立体感を出すための枠や、文字列をあらかじめ書いたpixmapを作成しておき、再描画の際にこのpixmapをウィンドウにコピーすることにより、ネットワークを通した場合でも、快適なスピードで操作を行うことができるのである。

さて、勘の鋭い読者はこの説明を聞いて、pixmap以外のデータもXサーバが動作しているマシン上にロードすることにより、パフォーマンスを改善できることに気がついたかもしれない。実はその通りである。後述するフォントデータなども、同じような手法によって高速な描画を実現している。

■ Pixmapへの描画とフォントのロード

Pixmapの概念が理解できたところで、Pixmapへどのようにして実際に描画を行うのかを見てみる。

pixmapへの描画を行う処理は、button_setup関数として定義されている。この関数が最初に呼ばれたときは、font_info変数がNULLになっているため、XLoadQueryFont関数が呼ばれてフォントがロードされる（ステップアップ講座参照）。

フォントデータは画像イメージで送ると膨大な量のデータとなるが、Xプロトコルで送るのは文字コードだけであり、英字であれば1バイト送るだけで済む。このように、フォントデータはXサーバ上に置くことにより、劇的にパフォーマンスを改善できる。

さて、フォントがロードされたら、ボタンの枠の色（黒）、表面の色、立体感を出すための右と下のボーダーの色、そして上と左のボーダーの色を示す数値を、次のようにget_colorという関数を用いて静的変数に保存している。

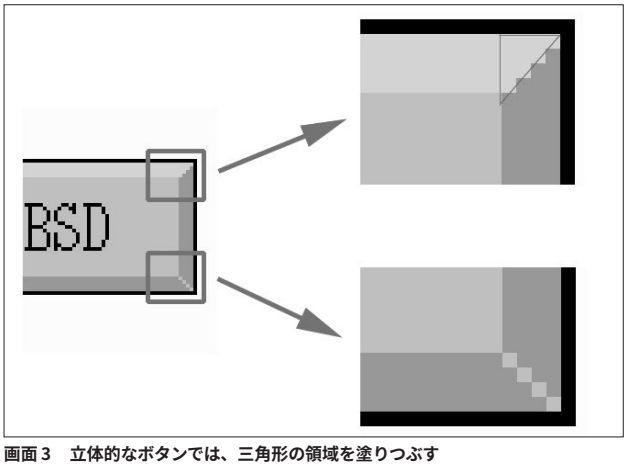
```
color_black      = get_color(   0,   0,   0 );
color_surface    = get_color( 178, 195, 223 );
color_down_right = get_color( 138, 154, 182 );
color_up_left    = get_color( 199, 215, 227 );
```

ここで、get_colorの引数は、赤、緑、青の順で、0～255の範囲となる。

なお、Xでの色の扱いは結構難しく、解説してある参考書も少ない。そのため、筆者がXのプログラミングをするうえで一番難しく感じた部分である。今回は残念ながら誌面の都合で解説することができないため、詳細はサンプルプログラムを参照してほしい。

さて、フォントと色の初期化が終わったところで、ボタンを描画するために必要なグラフィックコンテキスト（GC）を、XCreateGCを用いて作成する。GCは、先月号で解説したように、パフォーマンスを上げる目的で描画する線の幅や色をXサーバ上に保持するものであるが、ここでは、XSetForeground関数を用いて描画する色をGCに設定している。

そして、XFillRectangle関数を使って指定したpixmap



画面3　立体的なボタンでは、三角形の領域を塗りつぶす

の座標の長方形領域を、GCに設定した色で塗りつぶす。

なお、ボタンの四隅では、**画面3**のように三角形の領域を塗りつぶす必要がある。そのためには、多角形を塗りつぶすXFillPolygon関数を用いている。

つまり、この関数では、

- **ボタンの表面**
- **上と左の影**
- **下と右の影**
- **左下の三角形領域**
- **右上の三角形領域**
- **右上と左下の斜線**

を順番にそれぞれ指定した色で塗りつぶしている。

またボタンの文字は、先ほどXサーバ上にロードしたフォントデータをGCに指定して、XDrawString関数により描画を行い、最後に使用したGCを解放する。

Pixmapを目に見える形で描画する

今作成したpixmapは、単なるメモリ領域であるので、ビデオメモリに転送しないと表示されない。この転送を行うのがXCopyArea関数である。この関数は、

```
XCopyArea( base_disp, src, dst, gc,
           src_x, src_y, w, h, dst_x, dst_y );
```

というような引数を取り、srcで指定したpixmapからdstで指定したウィンドウ、つまりビデオメモリへの転送を行う。ここで「src_x, src_y」が転送するpixmapの長方形領域の左上の座標で、「w, h」が長方形領域の幅と高さである。また、「dst_x, dst_y」が転送先のウィンドウの左上の座標である。なお、「src,dst」はウィンドウ、pixmapのどちらでもよく、たとえばあるウィンドウから別のウィンドウへのコピーを行うことも可能となっている。ただ、いずれの場合もXサーバ上にあるデータのコピーであるため、XクライアントとXサーバとの通信では、座標データだけがやり取りされるだけとなっているわけである。

最後に

が、Xプロトコルがネットワークを通して効率よく実行するように作成されていることがわかっていただけたのではないと思う。

しかし、Xプロトコルの概念とXツールキットの役割については、実際のプログラムソースコードを追いかけてみないとなかなか理解しにくい部分もある。付録CD-ROMにサンプルソースコードがあるので、ソースコードをいろいろと変更しながら、自分で実際にプログラムを実行してみてほしい。

※来月号は、筆者の都合により休載となります。

Column

Xにおける カラーイメージデータの扱い

今でこそ、32Mバイトものビデオメモリが搭載されたカードが信じられないくらい安く売られているが、Xが開発された'80年代にはメインメモリでさえ数Mバイトしかなく、高速アクセスが必要な高価なビデオメモリを大量に搭載することは非常に難しかった。このため、メモリを節約するなどの理由により、ビデオメモリのビットパターン配列にはさまざまな形式があった。

一方、当時のUNIXマシンのCPUはマニュアルを表示（man ls）するだけでも数分かかるほど遅く、数Mバイトのビデオメモリを扱うのは速度的にも困難だった時代である。

このような状況の中、Xでは受け取ったイメージデータを変換したりするためのメモリ領域と変換時間を削減するために、どのような種類のビットパターン配列でもそのまま扱えるようなプロトコルを採用したのである。

このプロトコルでは、Xサーバによりさまざまなビットパターン配列を扱う必要があるが、同じハードウェア（ビデオカード）でも、XFree86やWindows上で動作するXサーバの種類によって、配列はバラバラである。今回のサンプルプログラムでは、XサーバからRGB各色のビットパターン配列情報を得て、各色のビットシフトを行っている。しかし、この方法では時間がかかるため、各色がバイト境界になっている場合などは、もっと最適化してスピードアップをする必要があり、結構面倒なプログラミングが必要になる。

以上のように、Xlibでのカラーイメージデータの扱いは、ハードウェアやOS環境と密接に関係しており、dib（Windowsビットマップ）という単一のイメージ配置だけを考えればよいWindowsと違って、Xlibでのカラーイメージの扱いはたいへん難しく見える。

しかしながら、Xの実装は効率を上げるためには重要であり、下層のXlibではメモリ効率とスピードだけを考え、Xlibでの複雑さは上層のツールキットが吸収すればよいというのは、たいへん理にかなった考え方である。

特別講座

ステップアップC言語

static変数

■auto変数とstatic変数

関数内で変数を定義する場合、デフォルトではその変数はauto変数になる。

auto変数とは、その記憶領域がスタック上に確保される変数であり、スタックのメモリ領域が有効な間だけ、その変数にアクセスできる。

なお、スタックとは関数を呼ぶ際に、その関数から戻るときのアドレスを保持したり、その関数を呼ぶ前のCPUのレジスタの状態を保持するメモリ領域のことである。この領域は、関数を何段も呼び出すとどんどん確保されていくが、関数からリターンするたびに次々と解放されていく、という性質をもっている。

このため、関数内だけで有効な変数もこのスタック領域に割り当てると、メモリの有効な利用ができるというわけである。

一方、スタックではなく、大域的なメモリに固定的に割り当てられる変数を静的な変数（static）と呼ぶ。

■auto変数とstatic変数の初期値

オート変数を初期値付きで宣言した場合は、その関数が実行されるたびに、その初期値が代入されることになる。

たとえば、

```
int func(void)
{
        int i = 0;

        if ( i == 0 ) {
                i = 1;
                printf("Hello!!");
        }
}
```

という関数を3回呼び出したとすると、3回とも“Hello!!”がプリントされることになる。

一方、

```
int func(void)
{
        static int i = 0;

        if ( i == 0 ) {
                i = 1;
                printf("Hello!!");
        }
}
```

としてiをstaticとして定義すると、iが0に初期化されるのは、プログラムが起動した直後だけになる。したがって、この関数を最初に呼び出したときだけ“Hello!!”が表示される。

つまり、この関数を最初に呼び出したときだけ、実行したい処理を行うことができるというわけである。

なお、「int i = 0」を関数内でなく、関数外で定義しても同じ効果が得られる。しかし、関数外に定義すると、ほかの関数からもこの値を書き換えることができてしまう。

したがって、ひとつのファイルに多数の関数が定義されている場合、ほかの関数でこの変数を書き換えている可能性が出てくる。すべてのコードを詳細に検討しないと、本当に最初の1回だけしか実行されるかどうかは判断できないのだ。

■関数の外で定義するstatic変数

関数定義の外で変数を定義すると、デフォルトでは外部変数（extern）として、定義されたファイルとは別のファイルにある関数からも参照することができる。

一方、定義の際にstaticを付加すると、その変数は同じファイル内にある関数からはアクセスできるが、違うファイルで定義される関数からはアクセスできなくなる。

つまり、file1.cとfile2.cという2つのファイルから構成されるプログラムの場合は、

```
---------- "file1.c" ----------
int a;

---------- "file2.c" ----------
extern int a;
func(void)
{
        printf("%d\n", a);
}
```

というようにプログラムを書くと、file1.cのaという変数を、“file2.c”でアクセスすることができる。

一方、“file1.c”で、

```
static int a;
```

として定義すると、“file2.c”からは“file1.c”のaという変数は見えなくなり、リンク時にエラーが出ることになる。

■static変数の有効な使い方

上述のようにstaticを使うと、ある変数を関数（正確にはブロック）やファイルの中だけに閉じ込め、外からは見せないようにすることができる。

数百のファイルから成る数十万行の大規模なプログラムで、むやみやたらと外部変数を使うのは非常に困る場合が多い。

あちこちのファイルで変数名が重複してしまうことがあるばかりでなく、間違って、似たような大域変数に誤った値を代入してしまうという、きわめて発見が難しいバグを生み出す元凶となることもある。

こういったバグを防ぐには、使用する変数の有効範囲をできるだけ狭くし、余計なところからアクセスできないように隠蔽することが重要になる。

また、staticは変数だけでなく関数の定義時にも指定することができ、変数の場合と同様にその関数の有効範囲はそのファイルの中だけとなる。

オブジェクト指向言語では、あるモジュールの内部で行う作業を隠蔽し、そのモジュール外からの余計な操作をできないようにすることを可能としている。C言語でもstaticを用いて、ファイルの構成を適切に設計することにより、外部からの操作が不必要な変数や関数を隠蔽した、理解しやすいプログラミングを行うことができる。

PostgreSQLを極める

前回よりPostgreSQLとWindowsの連携について紹介しているが、今回はその続きとしてODBCを利用したPostgreSQLデータベースの利用例をいくつか紹介する。なお、この連載は、今回をもってひとまず終了となる。

最終回　Windowsとの連携（2）

文：片岡裕生
Text : Hiroki Kataoka

PostgreSQL用ODBCドライバのインストールとセットアップ方法は、先月号で解説しました。今回は、ODBCドライバを実際に利用して、Windows環境からPostgreSQLデータベースへアクセスする具体的な方法を紹介します。

ODBC利用の具体例

まず、ODBC利用の解説のために用意したPostgreSQL側のデータベースについて説明しておきます。PostgreSQL側には、この連載でも何回か取り上げたことがある、顧客管理テーブル“customer”を含んだ“ascii9”という名称のデータベースがあるとします。

ちなみに、customerテーブルの構成は**表1**のとおりで、その内容をpsqlコマンド内からSELECT文で表示したところが**画面1**です。

MS-Access 2000の場合

Windows環境からPostgreSQLデータベースを利用する場合の代表的な例として、MS-Access 2000（以下、Access 2000）での場合を取り上げてみます。ご存じの方も多いとは思いますが、Access 2000は、それ自体がデータベース管理システムでもあります。しかし、ODBC対応

カラム名	データ型	説明
ccode	datetime	顧客コード
name1	text	姓
name2	text	名
addr1	text	住所1（都道府県）
addr2	text	住所2
addr3	text	住所3
zip	text	郵便番号
email	text	電子メールアドレス

表1　顧客テーブル（customer）の構成

```
ascii9=> SELECT * FROM customer;
ccode |name1  |name2  |addr1    |addr2    |addr3          |zip       |email
------+-------+-------+---------+---------+---------------+----------+------------------
    1 |和栗   |真由美 |東京都   |杉並区   |高円寺北1-15   |166-0002  |mayu@co.jp
    2 |松林   |啓介   |東京都   |世田谷区 |代田5-10       |155-0033  |keisuke@co.jp
    3 |渡辺   |順子   |東京都   |八王子市 |旭町3-2        |192-0083  |jun@co.jp
    4 |斎藤   |純子   |東京都   |中野区   |東中野5-3      |164-0003  |junko@co.jp
    5 |須田   |美加子 |埼玉県   |大宮市   |東町4-9        |330-0841  |mk@co.jp
        :
(29 rows)
```

画面1　customerテーブルの内容

アプリケーションでもあるため、ODBCドライバが提供されているほかのデータベースを取り扱うことも可能です。

そこで、PostgreSQLデータベースをAccess 2000でグラフィカルに扱う方法を紹介します。

Access 2000でODBC経由のテーブルを利用する最も一般的な方法は、リンクテーブルを利用することです。リンクテーブルとは、実体がAccess 2000の外部にあるテーブルのことです。ですから、最初にcustomerテーブルを指すリンクテーブルを作成します。

まず、Access 2000を起動します。**画面2**は“ascii9.mdb”という名前の新しいAccess 2000データベースを開いたところです。そして、メニューから［ファイル］→［外部データの取り込み］→［テーブルのリンク...］を選び、リンク画面を表示します（**画面3**）。デフォルトでは、ファイルを選択する画面になっていますので、ここではODBCデータベースを選択する画面に切り替えるために、［ファイルの種類］欄で“ODBC Datasource”を選びます。そして、データソースの選択画面からcustomerテーブルにアクセス可能なデータソースを選択して（**画面4**）、［OK］を押します。

ここで選択するデータソースは事前に作成しておきますが、選択したデータソースに必要な情報が足りなかった場合には、直後にPostgreSQL Connection画面が表示されて、足りない情報の入力が促されます（**画面5**）。

データソースの選択が終わると、PostgreSQLデータベース内に含まれるテーブルの一覧が表示されますので、アクセスしたいテーブル（ここではcustomerテーブル）を選択します（**画面6**）。

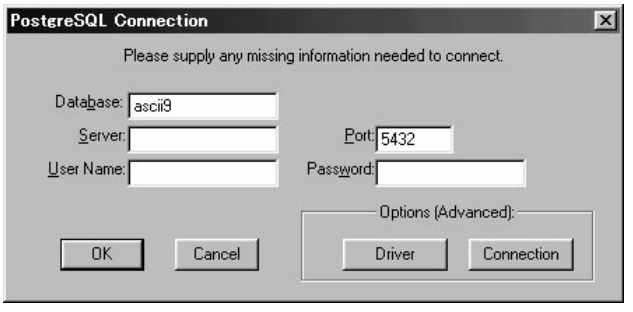
これで［OK］を押せばリンクテーブルが作成されるのですが、もしも、Access 2000が該当テーブルに主キーを見つけられなかった場合には、ここで固有レコード識別子の選択画面が表示されます。Access 2000で主キーが見つからないままテーブルをリンクすると、そのテーブルは更新できなくなってしまいますので、更新する予定があるのであればここで主キーとして代用可能なカラム（ユニークな値が格納されたカラム）を選択します。

この画面では複数のカラムを選択することも可能になっ

画面2　新規MS-Accessデータベース

画面3　リンク画面

画面4　データソースの選択画面



画面5　PostgreSQL Connection画面

画面6　PostgreSQLデータベースのテーブル一覧

ていますが、特にPostgreSQL 7.0以前のサーバでは不都合が発生することがありますので、必ず1つのカラムだけを選択してください。なお、複数のカラムの組み合わせでなければ主キーにできないテーブルの場合には、代替の方法としてPostgreSQLのシステムカラムであるoidカラムが利用できます。oidカラムの値は必ずユニークになっていますので、これを主キーとして代用するのです。

oidカラムを利用するには、データソースの設定のAdvanced Options（Driver）画面内にある［Show Column］オプションをチェックします。またこの場合には、oidカラムに別途インデックスを作成しておくことをお勧めします。そうしないとAccess 2000からのアクセスパフォーマンスが非常に悪くなってしまいます。

さて、これでリンクテーブルは作成できました。Access 2000のテーブル一覧には、今作成したcustomerテーブルへのリンクテーブルが表示されています（**画面7**）。

それでは、とりあえずcustomerテーブルの中身を表示してみましょう。customerテーブルをAccess 2000で開いた状態が**画面8**です。**画面1**のSELECT文の結果と比べてみてください。たしかに、PostgreSQLのデータベースにアクセスできているのがわかると思います。もちろんこの画面から直接データを書き換えたり、レコードを追加したりすることも可能です。

Access 2000では印刷も簡単に行えます。実際に需要のありそうな例として、次にタックシールへの印刷例を紹介します。

Access 2000で印刷を行うにはレポートを利用しますので、まずは新しいレポートを開きます。そして、利用するタックシールに合わせて、たとえば**画面9**のようにレイアウトします。ここでのレイアウトはあくまで1件分ですが、実際のタックシールは左右2段レイアウトになっていたりしますので、そのような場合には、メニューの［ファイル］

画面7　リンクテーブル作成後のMS-Accessデータベース

画面8　Access 2000に表示されたcustomerテーブルの内容

画面9　タックシールのレイアウト

画面10　タックシールの印刷プレビュー

画面11　集計クエリの設定

→［ページ設定...］を開き、段組などのレイアウトを調整します。タックシール印刷に関する作業はたったこれだけです。早速、このレポートの印刷プレビューを見てみましょう（**画面10**）。それらしくできあがっています。

このように、PostgreSQLデータベースのタックシール印刷も、ODBC＋Access 2000を使えば非常に簡単に行えてしまいます。

それでは次に、簡単な集計を行ってみましょう。例として、都道府県別の顧客数を降順で集計してみます。Access 2000で集計を行うには選択クエリを利用しますので、新しい選択クエリを開き、**画面11**のように設定します。そしてこのクエリを実行したものが**画面12**です。

この程度では、単にCOUNT集合関数とGROUP BY句を指定したSELECT文の実行となんら変わりません。そこで今度は、この集計結果をグラフに表示させてみましょう。Access 2000で画面を作成するには、フォーム機能を利用します。最初に、適当な大きさの新しいフォームを用意します。そしてメニューの［挿入］→［グラフ...］を選択してグラフ挿入モードにした後、フォーム上の適当な範囲をマウスでドラッグしてグラフを配置します。

Access 2000の設定によっては、ここでグラフウィザードが起動することがあります。通常ならばこのままグラフウィザードを利用してもいいと思いますが、グラフウィザードではメモ型フィールドが扱えませんので、今回の場合はあまり役に立ちません（ODBCドライバのデフォルト設定ではaddr1フィールドがメモ型になってしまいます）。

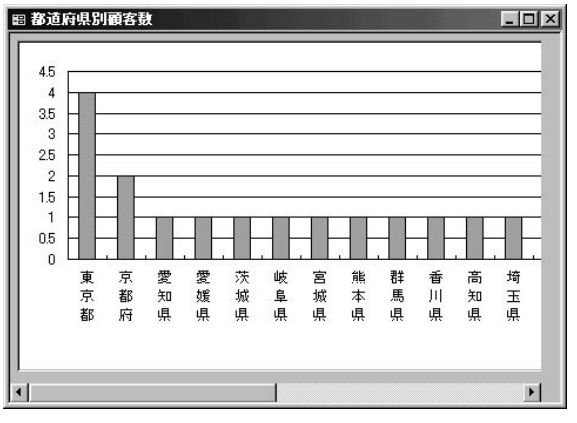
そこで、今回はグラフの種類やプロパティは手作業で設定します。まず、グラフをダブルクリックして環境をMS-Graphに切り替えます。そして、メニューから［グラフ］→［グラフの種類...］を選択し、利用したいグラフの種類を指定します。そのほかの細かな設定を済ませたら、グラフ以外のエリアをマウスでクリックして、MS-Graphの環境から抜け出します。最後にグラフのプロパティを表示して、値集合ソースに先ほど作成した集計用のクエリを指定（**画面13**）すれば完成です。このフォームを表示させたものが**画面14**です。

Access 2000からPostgreSQLなどの外部データベースを利用する場合によく見かける誤解として、SQL文の文法があります。Access 2000上で使用するSQL文の文法は、アクセス対象のテーブルがPostgreSQLデータベースを指すリンクテーブルであったとしても、Access 2000のSQL文法になります。ですからこの場合、PostgreSQL特有の文法で表現する機能（“::”による型変換など）は利用できません。

それでは、PostgreSQLのSQL文法を一切利用できないのかというと、そうでもありません。Access 2000には「パススルークエリ」という機能が用意されており、これを利用した場合にのみ、接続先のデータベース管理システムのSQL文法を利用することができます。つまり、パススルークエリを利用すれば、SQL文で可能なほとんどのPostgreSQL操作がAccess 2000からも可能になるのです。

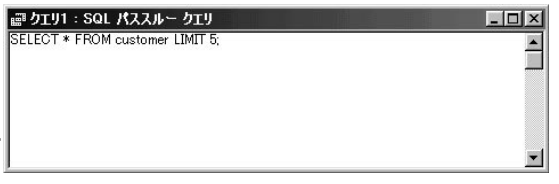
それではパススルークエリを作成してみましょう。最初の例は、PostgreSQL特有の機能である“SELECT～LIMIT～”文の利用例です。SELECT文にLIMIT句を

画面12 集計クエリの実行結果



画面14 集計フォームの実行結果

画面13 グラフのプロパティの値集合ソースの設定



画面15 パススルークエリ画面

指定すると、検索結果の件数を制限できますが、これをパススルークエリから実行してみます。

まず、新しく選択クエリを開きます。そしてファイルメニューから［クエリ］→［SQL］→［パススルー］を選択します。すると、選択クエリ画面がSQLパススルークエリ画面に変わりますので、ここにPostgreSQLの文法でSQL文を記述します。今回は例として、customerテーブルから5レコードだけを取り出す次のようなSELECT文を指定します（**画面15**）。

```
SELECT * FROM customer LIMIT 5;
```

次に、メニューから［表示］→［プロパティ］を選択してクエリプロパティを表示し、［ODBC接続文字列］を指定します（**画面16**）。このとき、右端の［…］ボタンを押すと、ODBC接続文字列を簡単に作成できます。なお、ここで指定するODBC接続文字列とは、このパススルークエリを実行するときの接続先データベースです。

これで、パススルークエリの準備は整いましたので、実行してみます。たしかに、PostgreSQL特有のSQL文法であるはずのLIMIT句が効いているようで、結果として5件しか表示されていません（**画面17**）。

もう1つパススルークエリの例を示します。今度はレコードを返さないSQL文の例として、東京在住の顧客データを別のテーブル（customer_tokyo）に格納する“SELECT～INTO～”文の例です。

先ほどのパススルークエリと同様に、新しくSQLパススルークエリ画面を開き、次のSQL文を入力します。

```
SELECT * INTO customer_tokyo FROM customer
    WHERE addr1 = '東京都';
```

パススルークエリのSQL文法はPostgreSQLの文法ですから、文字列を囲む記号は“'”（シングルクォーテーション）になります。次に、クエリプロパティを表示して、先ほどと同様に［ODBC接続文字列］を指定しますが、今回はさらに［レコード表示］を“いいえ”に設定します（**画面18**）。ここを“いいえ”にすることによって、レコードを返さないSQL文を実行できるようになります。

これで準備はできました。このパススルークエリを実行してみます（メニューから［クエリ］→［実行］）。特に、エラーが表示されなければOKです。本当にcustomer_tokyoテーブルにデータが格納されたかどうかは、読者の皆さんご自身で確認してみてください。

ところで、PostgreSQLは複数のSQL文を一度に実行することができます。これを利用すれば、パススルークエリをもう少し便利に使うことも可能です。

たとえば、上のパススルークエリで作成されるcustomer_tokyoテーブルは、このままではレコード数が増える一方です。そこで、先ほどのパススルークエリに、処理に先だってテーブルを空にするDELETE文を追加してみましょう。具体的には、次のSQL文をSQLパススルークエリ画面に入力します。

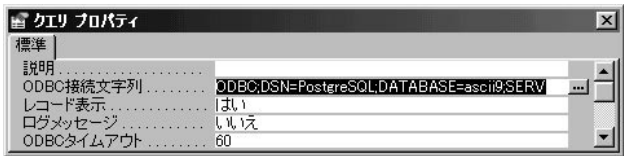
```
DELETE FROM customer_tokyo;
SELECT * INTO customer_tokyo FROM customer
    WHERE addr1 = '東京都';
```

2つのSQL文を続けて指定していますが、これは正しく動作します。

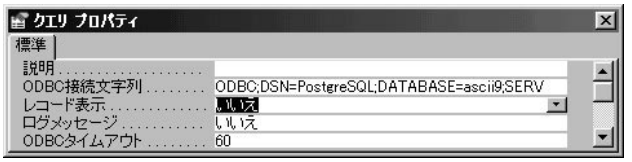
以上のようにAccess 2000は、PostgreSQLデータベースとともに利用してもたいへん便利なのですが、注意すべきこともあります。PostgreSQLのtext型は、ODBCドライバのデフォルト設定では、Access 2000のメモ型に対応付けられています。Access 97以前ではメモ型フィールドにインデックスがあることは許されませんので、もしもPostgreSQL側テーブルのtext型カラムにインデックスがあった場合、Access 2000から見た場合にはメモ型フィールドにインデックスがあることになってしまいます。

結果として、Access 97以前では、そのようなテーブル

画面17　SELECT～LIMIT～文の実行結果



画面16　クエリプロパティのODBC接続文字列の設定



画面18　クエリプロパティのレコード表示の設定

のリンクはできないのです。どうしてもリンクしたい場合には、PostgreSQL ODBC Driverの設定のAdvanced Options（Driver）画面にて、［Text as LongVarChar］のチェックを外し、PostgreSQLのtext型をAccess 2000の文字列型に対応させることで可能にはなります。

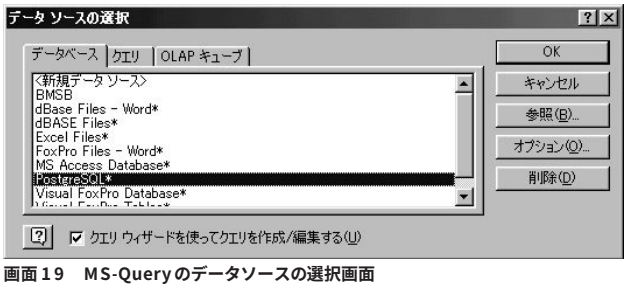
しかし、Access 2000の文字列型は254バイトまでしか扱えませんので、それ以上の長さのデータは正しく扱えなくなってしまいます。いずれにしても注意が必要です。

MS-Excel 2000の場合

次に、Access 2000よりも身近なアプリケーションと思われるMS-Excel 2000（以下、Excel 2000）での場合を取り上げてみます。Excel 2000もODBC対応アプリケーションですので、PostgreSQLデータベースにアクセスすることができます。ただし、基本的には、PostgreSQLデータベースからデータを読み込んでワークシートに貼り付けるという、参照中心の使い方になります。

それでは、Excel 2000で、先ほどのcustomerテーブルにアクセスする例を紹介します。

Excel 2000でODBCデータベースにアクセスするには、メニューから［データ］→［外部データの取り込み］→［新しいデータベースクエリ...］を選択します。すると、裏でMS-QueryというODBCアクセスツールが起動して、データソースの選択画面が表示されます（**画面19**）。そこで、customerテーブルにアクセス可能なODBCデータソースを選択して［OK］を押します。



画面19　MS-Queryのデータソースの選択画面

画面20　MS-Queryに表示されたcustomerテーブルの内容

その後の操作はクエリウィザードを使うかどうかで異なってきますが、最終的にMS-Queryの画面上にcustomerテーブルのデータが表示された状態が**画面20**です。この画面で、レコードを絞り込んだり並べ替えたりしながらデータを確認し、最後にメニューから［ファイル］→［MS-Excelにデータを返す］を選択して、MS-Queryを終了します。すると、Excel 2000への外部データの取り出し画面（**画面21**）が表示されますので、データを貼り付ける先を指定した後に、［OK］を押せば作業は完了です。PostgreSQLデータベースの内容がExcel 2000のワークシートに貼り付けられました（**画面22**）。

ここまでくれば、あとはごく普通にワークシート操作をすればいいわけです。なお、Excel 2000には、いったんワークシートに貼り付けた外部データを最新に保つ機能がありますので、時間とともに変化するPostgreSQLのテーブルをExcel 2000で解析するというような利用も可能です。

Visual Basic利用のヒント

最後に、Visual Basic（やVisual Basic for Application）からPostgreSQLデータベースにアクセスする場合のヒントをいくつか紹介します。Visual BasicからODBCデータベースにアクセスする一般的な方法は、DAO、RDO、ADOなどのライブラリを使うことだと思いますが、ここでは筆者がよく使うDAOやRDOを利用した場合の注意点などを紹介しておきます。

画面21　外部データの取り出し画面

画面22　Excel 2000に表示されたcustomerテーブルの内容

リスト1は、DAO（Jetワークスペース）を使ってcustomerテーブルにアクセスするコードの例です（ODBC接続文字列は省略してあります）。

この環境からPostgreSQL ODBC Driverを利用する場合の注意点としては、次のものがあります。

・主キーがないテーブルは更新できない

Jetワークスペースの制限のため、UPDATE文などのSQL文を利用した場合でも、更新はできません。だたしパススルークエリは除きます。

・利用可能なレコードセットの種類

キーセット――dbOpenDynaset
静的――dbOpenSnapshot
前方参照のみ――dbOpenForwardOnly

・トランザクションは実質的に利用できない

DAO（Jetワークスペース）では、単独のSQL文の実行時に複数の接続を張ることがあり、またPostgreSQLはトランザクションを接続単位で管理するため、この環境からトランザクションを利用すると、矛盾やデッドロックな

リスト1　DAO（Jetワークスペース）利用の例

```
Sub ScanByDAOJet()
    Dim wsp As DAO.Workspace
    Dim dbs As DAO.Database
    Dim rst As DAO.Recordset
    Dim i As Integer

    'ワークスペースの作成
    Set wsp = DAO.CreateWorkspace( _
        "", "admin", "", dbUseJet _
    )

    'データベースへ接続
    Set dbs = wsp.OpenDatabase( _
        "", dbDriverComplete, False, _
        "ODBC;DRIVER=PostgreSQL; … ;" _
    )

    'レコードセットのオープン
    Set rst = dbs.OpenRecordset( _
        "SELECT * FROM customer", _
        dbOpenSnapshot, 0, dbOptimistic _
    )

    'レコードの取り出し
    While Not rst.EOF
        For i = 0 To rst.Fields.Count - 1
            Debug.Print rst.Fields(i).Name;
            Debug.Print " = """;
            Debug.Print rst.Fields(i).Value;
            Debug.Print """"
        Next i
        rst.MoveNext
    Wend

    'クローズ
    rst.Close
    dbs.Close
    wsp.Close

End Sub
```

リスト2　DAO（ODBCワークスペース）利用の例

```
    Dim wsp As DAO.Workspace
    Dim dbs As DAO.Database
    Dim rst As DAO.Recordset
    Dim i As Integer

    'ワークスペースの作成
    Set wsp = DAO.CreateWorkspace( _
        "", "", "", dbUseODBC _
    )

    'カーソルライブラリの指定
    wsp.DefaultCursorDriver = dbUseODBCCursor

    'データベースへ接続
    Set dbs = wsp.OpenDatabase( _
        "", dbDriverComplete, False, _
        "ODBC;DRIVER=PostgreSQL; … ;" _
    )

    'レコードセットのオープン
    Set rst = dbs.OpenRecordset( _
        "SELECT * FROM customer", _
        dbOpenSnapshot, 0, dbOptimisticValue _
    )

    'レコードの取り出し
    While Not rst.EOF
        For i = 0 To rst.Fields.Count - 1
            Debug.Print rst.Fields(i).Name;
            Debug.Print " = """;
            Debug.Print rst.Fields(i).Value;
            Debug.Print """"
        Next i
        rst.MoveNext
    Wend

    'クローズ
    rst.Close
    dbs.Close
    wsp.Close

End Sub
```

どの致命的な障害を起こすことがあります。

・SQL文法の違い

DAO（Jetワークスペース）で用いるSQL文は、パススルークエリを除いて、Jetデータベースエンジンの SQL文法であり、PostgreSQLのSQL文法ではありません。

リスト2は、DAO（ODBCDirectワークスペース）を使ってcustomerテーブルにアクセスするコードの例です（ODBC接続文字列は省略してあります）。この環境からPostgreSQL ODBC Driverを利用する場合の注意点としては次のものがあります。

・利用可能なカーソルドライバの種類

ODBCカーソルライブラリ――dbUseODBCCursor

・利用可能なレコードセットの種類

静的――dbOpenSnapshot
前方参照のみ――dbOpenForwardOnly

・利用可能なロックタイプ

読み取り専用――dbReadOnly
行の値に基づいた共有的ロック――dbOptimisticValue

リスト3は、RDOを使ってcustomerテーブルにアクセスするコードの例です（ODBC接続文字列は省略してあります）。RDOからPostgreSQL ODBC Driverを利用する場合の注意点としては次のものがあります。

・利用可能なカーソルドライバの種類

ODBCカーソルライブラリ――rdUseODBC
クライアントバッチカーソルライブラリ――rdUseClientBatch

・利用可能な結果セットの種類

静的――rdOpenStatic
前方参照のみ――rdOpenForwardOnly

・利用可能なロックタイプ

読み取り専用――rdConcurReadOnly
行の値に基づいた共有的ロック――rdConcurValues
バッチモード更新を用いた共有的ロック――rdConcurBatch

■ありがとうございました！

一歩踏み込んだPostgreSQLの使い方を中心に紹介してきた連載「PostgreSQLを極める」ですが、気が付くと今回で9回目となっておりました。連載は、今回をもって終了です。長い間ありがとうございました。

リスト3　RDO利用の例

```
Sub ScanByRDO()
    Dim env As RDO.rdoEnvironment
    Dim con As RDO.rdoConnection
    Dim rst As RDO.rdoResultset
    Dim i As Integer

    '環境の作成
    Set env = RDO.rdoCreateEnvironment( _
        "", _
        "", _
        "" _
    )

    'カーソルライブラリの指定
    env.CursorDriver = rdUseOdbc

    'データベースへ接続
    Set con = env.OpenConnection( _
        "", _
        rdDriverComplete, _
        False, _
        "ODBC;DSN=;DRIVER=PostgreSQL; … ;" _
    )

    'レコードセットのオープン
    Set rst = con.OpenResultset( _
        "SELECT * FROM customer", _
        rdOpenStatic, _
        rdConcurValues _
    )

    'レコードの取り出し
    While Not rst.EOF
        For i = 0 To rst.rdoColumns.Count
            Debug.Print rst.rdoColumns(i).Name;
            Debug.Print " = """;
            Debug.Print rst.rdoColumns(i).Value;
            Debug.Print """"
        Next i
        rst.MoveNext
    Wend

    'クローズ
    rst.Close
    con.Close
    env.Close

End Sub
```

Rubyで行こう

プログラムが終了すると、そのプログラムが作り出したすべての情報は煙のように消えてしまいます。プログラムの実行が終了しても、情報を保存することを「データの永続化」と呼びます。今回はこの「永続化」について考えてみましょう。

第7回　データよ永遠に

文：赤松智也
Text : Tomoya Akamatsu

この連載では、このところ具体的なプログラムなしで解説してきたので、ややおもしろみに欠けたようです。今回は反省して、例題を中心にして解説してみようと思います。

今回の例題は「Rubyクイズ」です。プログラムの実行例を**図1**に示します。

ファイルからの情報取得

プログラムの実行を終了しても情報を保存するには、ファイルのような外部記憶に頼らなければなりません。外部記憶の使い方には、いろいろと気をつけなければならない点があるのですが、それはおいおい説明するとして、まずは簡単なファイルからの情報取得を考えてみることにしましょう。

Rubyクイズプログラムにおいて、外部からの情報取得といえば、クイズ内容の取得があります。どこからクイズの内容を読み込んでくるかは、いろいろな方法が考えられますが、今回はまず一番簡単な「プログラムそのものから読み込む」方法から考えましょう。

プログラムそのものから読み込む？ 外部からの情報取得じゃないのでは、と思われた方は鋭いです。しかし、Rubyのようなインタプリタ言語の場合は、プログラムそのものも単なるテキストファイルにしか過ぎません。ファイルからの入力と見なしてもよいでしょう。とりあえず一番簡単ですし。

ファイルにクイズを埋め込んだバージョンのRubyクイズを、**リスト1**に示します。

なんだか手抜きなプログラムですが、動作を説明するにはこれで十分でしょう。手続きquiz()は、クイズの内容を以下のような形式の配列で返します。

```
[[問題, [選択肢, 正誤], ...], ...]
```

「問題」と「選択肢」は文字列、「正誤」はtrueまたはfalseです。プログラム中盤のfor文以降からが本体で、ク

```
Rubyの作者の名前は？
1: まつもとゆきひろ
2: まつもとあきひろ
3: まつもとひろゆき
4: まつもとひろのぶ
answer? 1
正解

Rubyの開発が始まったのは？
1: 1986年
2: 1989年
3: 1993年
4: 1996年
answer? 2
はずれ

解答終了
成績: 2問中1問正解
正答率: 50%
```

図1　Rubyクイズ実行例

イズ内容のデータに対してループを回し、入力を受け付け、解答をチェックしています。

最後に、手続きresultで成績を集計して表示しています。このプログラムでは、グローバル変数を2つ使っています。それぞれ正答の数($yes)と問題の総数($total)になっています。グローバル変数はあまりお勧めでないと言われますが、この程度のサイズのプログラムであれば、さほど問題にならないでしょう。

とはいえ、プログラムに問題を埋め込むというのは手軽でよいのですが、クイズの問題が固定的になるという欠点があります。やはり、クイズの内容は別のファイルに置いたほうがよかったかもしれません。

そこで別ファイルからクイズを読み込むバージョンを作ってみましょう。要するに、ファイルを読み込んで、手続きquizが返していたのと同じ構造の配列を返す手続きを書けばよいわけです。そのようにして書き換えたものが**リスト2**です。

例によって手抜き例題なので、読み込むファイル名はカレントディレクトリのquiz.datに固定です。読み込むファイル名やそのディレクトリを任意に指定する方法は、読者への宿題として残しておきます。

quiz.datのファイルフォーマットは、以下のようになります。

```
Rubyの作者の名前は？

o まつもとゆきひろ
x まつもとあきひろ
x まつもとひろゆき
x まつもとひろのぶ

Rubyの開発が始まったのは？

x 1986年
x 1989年
o 1993年
x 1996年
```

つまり、問題と選択肢が空行をはさんで繰り返す形式です。選択肢には、正解の前に“ o ”を、間違いの前には“ x ”を置くことにします。問題が空行を含む場合には、“ . ”(ピリオド)だけからなる行を置けば、空行に置換されます。

この手続きquizで注意すべき点は、以下の2点になります。

▼リスト1　quiz0.rb（問題埋め込み版）

```
#!/usr/bin/ruby

def quiz
  [
    ["Rubyの作者の名前は？",
      ["まつもとゆきひろ", true],
      ["まつもとあきひろ", false],
      ["まつもとひろゆき", false],
      ["まつもとひろのぶ", false]],
    ["Rubyの開発が始まったのは？",
      ["1986年", false],
      ["1989年", false],
      ["1993年", true],
      ["1996年", false]],
  ]
end

$yes = 0
$total = 0

def result
  STDOUT.print "解答終了\n"
  STDOUT.printf "成績: %d問中%d問正解\n", $total, $yes
  STDOUT.printf "正答率: %d％\n", 100*$yes/$total
end

for quiz, *answer in quiz()
  STDOUT.print "-"*10, "\n"
  STDOUT.print quiz, "\n"
  n = 0
  loop do
    i = 1
    for a, in answer
      STDOUT.printf "%d: %s\n", i, a
      i += 1
    end
    STDOUT.print "answer? ";
    STDOUT.sync
    line = STDIN.gets
    if /^\d+$/ =~ line
      n = line.to_i
      break
    end
    print "数字を入力してください\n"
  end
  $total += 1
  if answer[n-1][1]
    STDOUT.print "正解\n"
    $yes += 1
  else
    STDOUT.print "はずれ\n"
  end
end
STDOUT.print "-"*10, "\n"
result()
```

- **readline("")で、行単位ではなく段落をまるごと読んでくること。**
- **ファイルの終端でgetsはnilを返すが、readlineはEOFError例外を発生させること。この例外はrescueで捕捉される。**

あとは、行末の改行を取り除くchomp!であるとか、“.”を空行に置換するsub!などと、正規表現のマッチした部分より後ろが格納されている変数“$”を押さえることさえ忘れなければ、この手続きを読み解くのは難しくないと思います。

さて、これでファイルから読み込む版の「Rubyクイズ」プログラムが完成しました。しかし、やはりクイズ内容がカレントディレクトリになければならないという仕様は、いくらこれが手抜きプログラムだとはいえあんまりです。そこで、ファイルが見つからなければ、プログラムから読み込むという仕様にしてみましょう。

このためにはRubyの__END__という機能を使います。実は、Rubyインタプリタは__END__という行に出会うと、そこでプログラムが終了したと見なし、ファイルの残りの内容はDATAという定数で参照されるIOオブジェクトから読み出すことができます。そこで、ファイルが見つからなかったときには、DATAから情報を読み出すようにすればよいということになります。変更点は、次の2カ所です。

まず第1点は、ファイルをオープンする部分を「見つからなければDATAから」という処理に変更します。具体的には、プログラムの末尾に__END__という行を置き、その後ろにクイズの内容を追加します。

第2点は、手続きquizのopenのある行を、以下のように書き換えます。

```
begin
  f = open("quiz.dat")
rescue
  f = DATA
end
```

これで、open()でファイルが見つからなかったときに発生する例外を捕まえて、その場合にはDATAから読み込むことになります。たったこれだけの変更で、「ファイルが見つからなければ、プログラムから読み込むという仕様」が実現できてしまいました。さすがはRuby。

さて、ここに示したRubyクイズプログラムは楽しむにはあまりにも手抜きです。実際にRubyクイズで遊ぶには、もっともっと手をいれないといけないでしょう。具体的には、以下のような改造が考えられます。

- **たくさん問題を用意する**
- **乱数で問題を選ぶ**
- **途中であきらめたりした場合も集計する**

これらは読者への宿題にします。とはいっても、「たくさんの問題」はなかなか用意できないでしょうから、いくつかの問題を含んだデータファイルをCD-ROMに収録しておきます（といっても8問しかありませんが）ので、そちらを利用してください。

データ保存

ただ単に、データをファイルから読み込むだけなら、話はそれほど難しくはありません。永続化の最初の難関は、情報の保存と共有にあります。

外部にデータを保存する場合、特に複数のプログラムから共有する場合に、まず考えなければならないことは「排

▼リスト2　ファイル読み込み手続きquiz

```
def quiz
  f = open("quiz.dat")
  data = []
  begin
    loop do
      entry = []
      question = f.readline("")
      question.sub!(/^\.$/, '')
      question.chomp!
      entry.push question
      answers = f.readline("")
      for line in answers
         line.chomp!
         if /^o / =~ line
           entry.push [$', true]
         elsif /^x / =~ line
           entry.push [$', false]
         end
      end
      data.push entry
    end
  rescue EOFError
  end
  data
end
```

他制御」です。あまり聞きなれない言葉ですから、少々説明しましょう。複数のプログラムが同時にファイルに書き込みにいくと、いろいろとまずいことが起きる可能性があります。まず考えられるのは、出力が混じってしまうことです。これにより、書き込み結果が「ノイズ」のようになってしまう可能性があります（**図2**）。

もうひとつの問題は、操作の順番が狂うことにより、一見正しそうに見えますが、実は間違った結果になってしまうことです。たとえば、銀行口座の操作を考えてみましょう（**図3**）。ちょっと現実的でない例ですが、あくまでも説明のためなので我慢してください。

操作Aは口座の残高を取得し、引き出した分の2000円を引いた額を新しい残高として設定しています。一方、操作Bも同様に、口座の残高を取得し、引き出した分の3000円を引いた新しい残高を設定しています。結局、1万円の口座から計5000円を引き出したわけですから、残高は5000円になっているはずです。しかし、**図3**に示すようにこれらの操作が同時に行われると、場合によっては正しくない結果が得られる場合があります。この場合には、計5000円引き出しているのに、残高は7000円も残ってます。引き出すほうにとってはありがたいことですが、銀行にとっては死活問題です。こんなプログラムを作ったら、すぐにクビになってしまいます。

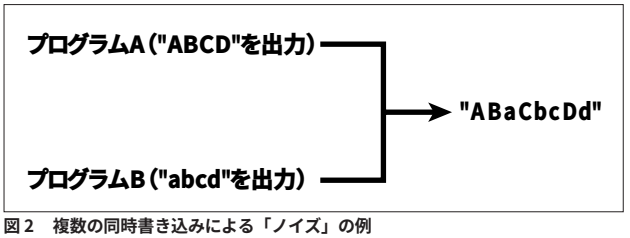


図2　複数の同時書き込みによる「ノイズ」の例

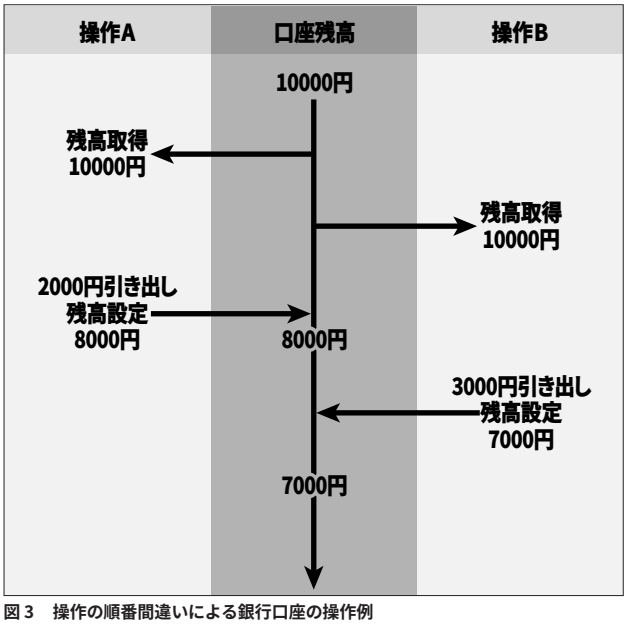


図3　操作の順番間違いによる銀行口座の操作例

このような変なことが起きないためには、同時にファイルの変更が起きないようにする必要があります。これを「排他制御」と呼びます。排他制御とは、同時に行われる可能性のあるそれぞれの操作が、あたかも別々の順番に実行されたと同じ結果を保証することです。このような制御を「直列化」あるいは「シリアライズ」と呼ぶこともあります。

ややこしいのは、永続化には「シリアライズ」という単語を使うもうひとつ別の局面があることです。もうひとつの「シリアライズ」については後述します。

例題であるRubyクイズに戻りましょう。情報保存の例として、過去の成績の統計をとってみましょう。とりあえず統計は /tmp/quiz.stat というファイルに格納することにしましょう。そのためにはquiz.rbの手続きresultを**リスト3**のように修正します。

このバージョンの手続きresultでは、通常の成績表示の後、データファイルである/tmp/quiz.statから読み込みを行い、今回の成績を追加したうえで、新しい成績を書き込みます。排他制御のために、今回はflockメソッドを使っています。flockメソッドによってファイルをクローズするまで、そのファイルをロックすることができます。

最初の実行では、/tmp/quiz.statファイルは存在しないので、エラーになります。「rescue Errno::ENOENT」の

▼リスト3　過去の統計を取る手続きresult

```
def result
  STDOUT.print "解答終了\n"
  STDOUT.printf "成績: %d問中%d問正解\n", $total, $yes
  STDOUT.printf "正答率: %d%\n", 100*$yes/$total
  begin
    f = open("/tmp/quiz.stat", "r+")
    f.flock(File::LOCK_EX)
    yes, total = f.readline.split
    $yes += yes.to_i
    $total += total.to_i
    f.rewind
    f.printf "%d %d\n", $yes, $total
    f.close
  rescue Errno::ENOENT
    f = open("/tmp/quiz.stat", "w")
    f.printf "%d %d\n", $yes, $total
    f.close
  end
  STDOUT.printf "累計成績: %d問中%d問正解\n", $total,
$yes
  STDOUT.printf "累計正答率: %d%\n", 100*$yes/$total
end
```

部分でそのエラー（例外）を捕捉して、新たにファイルを作っています。複数ユーザーから使うためには、このファイルのパーミッションについても考える必要があるでしょう。

共有情報の保存の方法には、いま使った通常ファイルのほかにも、さまざまな方法があります。そのうちの主なものをメリット・デメリットとともに紹介しましょう。

■ファイルによるデータ保存

通常、ファイルはデータ保存の基本で、最もよく用いられる手段でもあります。一方、ある意味で原始的であるため、毎回データ保存のために「車輪の再発明」を強制されるような気分になることもあります。メリットは単純なこと、デメリットは明示的な排他制御が必要なことと、保存したいデータが単純なテキストデータでない場合、特に構造を持つデータの場合にはやや手間がかかります。

ファイルの排他制御のためには、以下の方法が使えます。

・flock

flockメソッドによってロックをかけると、ロックが解除されるまで、ほかのflockは待たされます。ファイルをクローズすると自動的にロックが解放されます。

・ロックファイル

flockメソッドを使う代わりに、あるファイルがロックされていることを示すためのファイルを作成する手法です。あるファイルにアクセスする前に、そのファイルに対応するロックファイルがあるかどうかを確認します。ロックファイルがあれば、なくなるまで待ちます。ロックファイルのチェックと作成の間にほんのわずかでも時間差があると、タイミングによっては先ほどの銀行口座の例と同様の問題が発生する可能性があるので、ロックファイルの作成には工夫が必要です。UNIXでは、一般的にディレクトリとシンボリックリンクがチェックと作成が同時に行えるというロックファイルに向いた性質を持っています。

flockのほうが少々高級で使いやすいのですが、クローズされた瞬間にロックが解放されてしまうので、場合によっては使えないことがあります。なお、ロックファイルもflockも、いずれも強制力はありません。ロックをチェックしなければ、ロックされているかどうか分からないため、意図的であるにしろ、ないにしろ、ロックを無視してデータを破壊することは可能です。これらのロックは、いわば紳士協定というところでしょうか。

■DBM

DBMは、ほとんどのUNIXで使える一種のデータベースで、プログラム同士で文字列を受け渡すことができます。Rubyからは、dbm（またはgdbm）拡張ライブラリによって利用することができます。使い方は以下のようになります。

```
require 'dbm'

f = DBM.open("/tmp/quiz.stat")
f["yes"] = $yes.to_s
f["total"] = $total.to_s
f.close
```

前後にopenとcloseがあるほかは、ほぼハッシュと同じように使うことができます。文字列から文字列へのハッシュで実現できる構造の表現には、非常に適しています。

DBMのデメリットは以下のとおりです。

・ロックを行わないため、排他制御が必要。しかし、flockは使えない
・キー、バリューともに文字列のみという制限がある
・要素となる文字列に大きさの制限がある（実装ごとに違うが、多くのdbmでは4Kバイト）。要素数の制限はゆるい

これらの問題（2番目を除く）を改善したBerkeley DBライブラリというものも存在します。しかし、残念なことに、このライブラリに対するRuby用拡張ライブラリはまだ存在していません（開発中との噂も聞きますが）。

■関係データベース

「関係代数」理論に基づいたデータベースです。というと難しそうですが、PostgreSQL、Oracle、InterBase、MySQLなどのいわゆるデータベースシステムは、ほとんどこの範疇に入ります。これらのデータベースシステムに対するインターフェイスを実現するRuby用拡張ライブラリが開発されています。

関係データベースは、もともと情報の保存と共有を目的として開発されているので、排他制御や検索などの機能を含んでいます。データタイプとしては、文字列以外にも数値や日付などの豊富な型を持っていますが、オブジェクト同士の参照やリンクなどの情報もテーブルの形で表現しな

ければならないという点は、少々面倒に感じることもあります。

オブジェクト永続化

さて、ファイル、DBMあるいはデータベースなどを使って情報を保存するにしても、保存できるデータタイプは限られたものです。Rubyで定義したオブジェクトなどをそのまま保存することはできません。特に、オブジェクトによる情報は、複数のオブジェクト間で相互にリンクされることによって表現されているものも多く、ファイルやデータベースへの保存には、なんらかの変換が必要になります。前に示した手続きresultのような単純なものでも、数値を文字列に変換していましたね。

このようにオブジェクトと保存に適した形式（多くの場合には文字列）の間で、相互に変換することも「シリアライズ」と呼ぶことがあります。また、「マーシャリング」と呼ぶこともあります。シリアライズのほうは、前に登場した別のシリアライズと混同しそうな気がするので、ここではマーシャリングのほうを用います。

多くの場合には、保存するデータに合わせて相互変換するマーシャリング手続きを用意する必要があります。手続きresultにあった、

```
f.printf "%d %d\n", $yes, $total
```

のようなものも、そのようなマーシャリングの一種です（極端に簡単なものですが）。

しかし、楽をしたいのが人間というものです。「お気楽プログラミング」をめざすRubyなら、なおさらです。さいわいなことに、Rubyにはそのようなニーズをサポートする仕掛けが組み込まれています。「Marshal」というモジュールがそれです。

Marshalモジュールには、ほぼ任意のRubyオブジェクトをマーシャリングするメソッドdumpと、逆にマーシャリングされたデータから元のオブジェクト（のコピー）を復元するメソッドloadが提供されます。下記を参照してください。

- **dump(object[,io][,limit])**

objectをマーシャリングします。IOオブジェクトが指定されたときには、そのIOに対してマーシャリングの結果を書き出します。そうでないときには、マーシャリングした文字列を返します。整数limitが指定されたときには、limit段以上深くリンクしたオブジェクトは、マーシャリングしません。

- **load(string_or_io[, proc])**

マーシャリングされた文字列、またはマーシャリングされたデータを保存したIOオブジェクトを引数に取り、元のオブジェクトと同じ状態を持つオブジェクトを返します。手続きオブジェクトprocが指定されたときには、復元されたそれぞれのオブジェクトに対してprocが呼び出されます。

Marshalを使えば、複雑な構造を持つデータも簡単にファイルに書き出すことができます。Marshalはリンクが**図4**のように複雑になっても対応できます。

しかし、Marshalも万能ではありません。なかなか完璧なものはないものです。Marshalの欠点とは、ダンプできないオブジェクトがあることです。具体的には、以下のオブジェクトは保存することができません。

- **特異メソッドを定義したオブジェクト**
- **IOやソケットなど基本的にプロセスを越えて保存できないオブジェクト**
- **拡張ライブラリが定義したような、Rubyが保存方法を知らないオブジェクト**

もっとも、このようなオブジェクトを保存したいと思うことはまれなので、さほど問題にはならないようです。

しかし、人間の楽したい気持ちには限りがありません。いっそ、なにも考えずにオブジェクトがそのまま保存できたらどんなによいでしょう。普通だったら、そうそう簡単

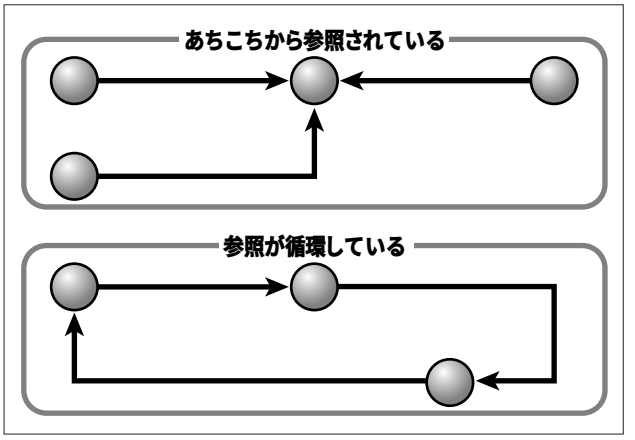


図4　Marshalが対応する複雑な構造を持つデータ

にはいきません。

ところが、そんなうまい話があるんです。それが「PStore」です。PStoreは、Persistent Store（永続的保存）の略のようです。

PStoreは、Ruby本の「6.2 pstore.rb」（251ページ）に解説してあります。しかし、読者の皆さん全員が、あの本を持っているわけではないでしょうから、もう一度説明します。ただ、Ruby本を持っていて損はありませんよ。持ってない人は購入を検討してみてください。

PStoreクラスの機能の抜粋を以下に示します。

・ps = PStore::new(path)

データベースオブジェクトを生成します。データは、pathで指定されたファイルに格納されます。

・ps.transaction { }

トランザクション（一連のデータベース操作）を開始します。このtransactionメソッドの範囲内でだけ、データベースの内容にアクセスできます。

・ps[name]

データベースにnameという名前をキーとして格納されているオブジェクトを取り出します。

・ps[name]=obj

データベースにnameという名前をキーにしてオブジェクトobjを格納します。実際にファイルに保存されるのは、そのオブジェクトから（Marshalを使って）再帰的にアクセスされるすべてのオブジェクトで、トランザクション終了時の状態が保存されます。

・ps.root?(name)

データベースにnameという名前のキーが登録されているかどうか調べます。

・ps.commit

トランザクションを終了します。このメソッドを呼び出した時点で、transactionメソッドの終了にジャンプすることになります。

・ps.abort

トランザクションを中断します。ps.abortの場合も、このメソッドを呼び出した時点で、transactionメソッドの終了にジャンプします。abortでは、トランザクション中にデータベースに対して行った変更はすべて無効になります。

PStoreは、自動的にマーシャリングをしてくれるだけでなく、flockによる排他制御まで行ってくれるので、非常に使いやすいライブラリになっています。

しかし、わずか143行（コメント含む）で実現されているだけあって、あまり大量のデータの取り扱いには向きません。トランザクション開始時に全ファイルを読み込んできて、終了時には全部を書き出すのですから、当然と言えば当然ですが……。PStoreは、ごく小規模のデータ保存にだけ使うのが賢そうです。

しかし、大規模データにも対応できるPStoreのようなものがあればよいのに、といつも思います。難しそうな気もしますが、原理的には不可能ではないと思うので、いつかだれかが作らないでしょうかねえ。

ここでRubyクイズに戻って、手続きresultをPStoreを使って実現してみましょう（**リスト4**）。手続きの中身がずいぶん簡略化されているのが分かると思います。仮に、保存するデータが整数2つではなく、もっと複雑なものであれば、簡略化の効果はもっともっと大きなものになったでしょう。

あ、そうそう。PStore版のクイズプログラムも情報を保存するファイル名が共通です。したがって、まず/tmp/quiz.statファイルを削除してから実行してください。

▼リスト4　PStoreを使った手続きresult

```
require 'pstore'

def result
  STDOUT.print "解答終了\n"
  STDOUT.printf "成績: %d問中%d問正解\n", $total, $yes
  STDOUT.printf "正答率: %d%\n", 100*$yes/$total
  ps = PStore::new("/tmp/quiz.stat")
  ps.transaction do
    unless ps.root?("stat")
      stat = [0, 0]
    end
    stat[0] += $yes        # ただ代入するだけ
    stat[1] += $total      # transaction終了時に保存される
    $yes = stat[0]         # 後で参照するため
    $total = stat[1]       # グローバル変数に代入しておく
  end
  STDOUT.printf "累計成績: %d問中%d問正解\n", $total, $yes
  STDOUT.printf "累計正答率: %d%\n", 100*$yes/$total
end
```

プログラムについて

今回、開発した「Rubyクイズ」プログラムを本誌付録CD-ROMに収録してあります。CD-ROMに収録されているファイルは、以下のとおりです。

quiz0.rb　**リスト1**に示された版
quiz1.rb　quiz.datを読み込む版
quiz2.rb　DATAからも読み込む版
quiz3.rb　成績を保存する版（テキスト）
quiz4.rb　成績を保存する版（PStore）
quiz.dat　クイズデータファイル

リクエストをお待ちしています

今回はRubyクイズプログラムを中心にデータの永続化について学びました。

ただ単に、データを保存するだけでも、いろいろ考えねばならないことがあるものですね。

個人的にはPStoreは非常に使いやすいので愛用しています。たいていの場合は、必要なデータをそのまま放り込めばよいので、これほど便利なことはありません。速度の問題が解決されて、もっと大規模な場合にも心配なしに使える日が来ないかなあと願っています。実現可能かどうかは知りませんが。

今回は説明しませんでしたが、CGIのようなプログラムは、しばしばこの永続化を必要とします。たとえば、Web掲示板の書き込みなどはまさにそれです。CGIは、Rubyに向いた分野ですから、今後ぜひ説明したいと思っています。皆さんも知りたいことがあれば、linuxmag@ascii.co.jpまでメールでリクエストしてください。

『オブジェクト指向スクリプト言語Ruby』
まつもとゆきひろ／石塚 圭樹 共著
アスキー発行
本体価格4000円
ISBN4-7561-3254-5

Rubyの開発者であるまつもとゆきひろ氏と、オブジェクト指向の専門家である石塚圭樹氏によるRubyの公式解説書。Rubyとはどんな用途に向いている言語であるかといった基礎から、Rubyの言語仕様、Rubyによるオブジェクト指向プログラミングなど、Rubyのすべてを紹介している。また、Rubyの応用プログラミングや内部構造、Rubyの拡張についても深く解説されている。そのため、総ページ数は600ページ近くにもなっており、さらに、付録として、Rubyの言語仕様や関数、クラスライブラリや、Ruby関連用語集なども収録されている。本連載でも、しばしば「Ruby本」という呼び方で登場している。Ruby使いの必携書である。（編集部）

Column

今月のRuby 1.5

■新正規表現

Perl5正規表現のうち、Rubyに欠けていたものの一部が実装されました。実装されたのは以下のものです。

・\G
前回のマッチの末尾にマッチします。gsubやscanにおいて有効です。

・(?>...)
独立正規表現です。この範囲内でマッチしたパターンはバックトラックしません。ですから、(?>a＊)abは決してマッチしません。というのも、a＊がすべてのaの並びを「飲み込んで」しまうので、abにマッチできないからです。(?>)で囲まれていなければ、バックトラックにより、aを1つ戻った地点でマッチできるのですが。

これらをどう使うかというと……。うーん、正規表現は難しいなあ（笑）。

■正規表現オプション

Ruby1.5では、改行に対する特別扱いを中止するpオプションがなくなりました。これに対応するのは、「複数行オプション」を意味するmオプションです。

ただし、pオプションとmオプションでは少々挙動に違いがあります。そのために名前が変わったわけですが。

それは、mオプションでは、正規表現の^や$が改行を無視しないという点です。以前のpオプションでは、^は文字列の先頭に、$は文字列の末尾にマッチしていました。mオプションでは、これらは文字列中の改行ともマッチします。Perlになぞらえると、pオプションは「単独行モード」に、mオプションは「純粋複数行モード」に対応するようです（『詳説 正規表現』オライリー・ジャパン発行の256ページを参照）。

1.5ではpオプションを使うと警告が出力されます。将来pオプションは使えなくなるかもしれませんから、早期のmオプションへの移行をお勧めします。

Linuxファイルシステムの現在と未来

動画やMP3データなどのマルチメディアデータはファイルのサイズが数Mバイトから数十Mバイトもある。また、最近ではPCエミュレータであるVMwareが流行っており、VMware上で動かすOS用の仮想ディスクを作成するために、大きなパーティションを用意する人もいることだろう。今回は、その大容量パーティションを使う時に問題となるfsckにかかる時間を解決するジャーナリング機能を解説する。

第2回　fsckとジャーナリング

文：長岡モイチ
Text :Moichi Nagaoka

大容量ディスクの憂鬱

近頃のIDEハードディスクの大容量化、低価格化は、ちょっと前では考えられないほどの水準になっている。筆者も25Gバイトのハードディスクを先月約2万円で購入した。まったくありがたい世の中になったものである。

さて、この大容量ハードディスク、パーティションを切ってext2ファイルシステムとして使用する場合、root（/）や/usrなどのシステムパーティションに5Gバイトを割り当て、残り10Gバイト～20Gバイトにユーザーデータを格納するための専用パーティションを割り当てることが一般的な構成だと思われる。

ディスクが大容量になると、それまで遭遇することのなかった問題に悩むことになる。それはfsckにかかる時間だ。ちゃんとshutdownされていなかったファイルシステムは、次回のブート時にfsckの対象となる。また、ext2ではファイルシステムのスーパーブロックにmount回数を保持する領域があり、決められた回数に達すると安全のために無条件にfsckの対象となる。デフォルトでは20回mountするとブート時にfsckが有無を言わさず実行されてしまう。

fsck

ではこのfsckは一体何をしているのだろう。簡単にいうと、ファイルシステムの整合性をチェックし、不整合があった場合はそれを修復するという処理を行っている。

たとえばシステムクラッシュなど不測の事態が発生すると、更新されたファイルシステムの管理領域の情報がディスクに書き込まれる前に強制終了してしまうため、OSが実行すべきファイルシステムの更新処理が途中のままで、スーパーブロックやブロックグループ内に保持されている空きブロック数や空きi-node数の値と、実際の空き状態を表したビットマップ領域の内容が一致していないなど、ファイルシステムの実際の内容と管理領域の内容に食い違いが生じている状態である。

具体的には、すでに解放されているはずのブロックがブロックグループのビットマップ上では使用中のままになっていたり、反対に割り当てたはずのブロックがブロックグループのビットマップ上では空き状態のままだったりする。i-nodeの管理も同じようにビットマップで管理しているので、同様の状態が起こり得る。

fsckはこのような管理領域の情報と実際の内容が合っていないファイルシステムの状態を検出し、ファイルシステムの管理情報を正しい内容に更新する役割を担う。整合性の取れていない（cleanでない）ファイルシステムを、fsckを実行しないでmountして使用した場合、知らず知らずのうちにファイルシステムの内容が壊れていくので、cleanでないファイルシステムはmount前に必ずfsckを実行する必要があるのだ。

さて、以上の説明でfsckの必要性は理解できたと思うが、同時に現在のfsckによる方法では、ファイルシステ

ムの容量が大きくなりチェックすべきブロック数が多くなってくると、fsckの実行時間も相対的に長くなってしまうことが理解できるだろう。fsckによるファイルシステムのチェックはブート時の早い段階で行われるから、fsckの時間はそのままサービスのダウンタイムに直結することになる。

もしもNFSサーバやSamba、FTPサーバなど、一般に大容量ファイルシステムを必要とされるサーバが不意の電源断などでクラッシュした場合、リブート後にサービスが再開されるまで30分以上もかかってしまったら、クリティカルな状況においては大問題になるはずだ。たとえば、Eコマース、ネットオークション、ネットトレーディング関係のWebサーバなどが挙げられるだろう。

ジャーナリングファイルシステムとは

ジャーナリングは、fsckで見られるクラッシュリカバリ時の長い待ち時間を大幅に短縮できる技術である。もともとはデータベースのトランザクション処理に使用されていた技術をファイルシステムに対して応用した技術であり、今日のミッションクリティカルな企業サーバで使用されているファイルシステムのほとんどにはジャーナリング機能が備わっている。

ジャーナリングはファイルシステムに対する変更をディスク上のログ（ジャーナルとも呼ばれる）に記録することで、ファイルシステムのチェックを変更された箇所に限定して行うことができるようにしている。そのため、クラッシュリカバリの時間を大幅に短縮できるのだ。

たとえば、ext2ファイルシステムなどジャーナリング機能を持たないファイルシステムでは、システムクラッシュした直前にそのファイルシステムに対してどのような変更が行われていたのかわからないので、fsckでは管理領域（スーパーブロック、ビットマップ、i-node領域、間接ファイルブロックなど）のすべてをチェックして整合性を確認する必要がある。そのために大容量ファイルシステムではどうしてもfsckの時間が長くなってしまう。

対するジャーナリングファイルシステムでは、ファイルシステムを実際に変更する前に、まずログにその変更内容を記録する。実際のディスク上のデータの変更はログに記録されるまで行われない。システムクラッシュ後にファイルシステムの整合性を調べるには、ログに記録された変更内容をチェックする。ログには記録されているが実際のファイルシステムには反映されていない処理に対しては、ログの内容に基づき作業を行い処理を完了させる。これはログのリプレイとも呼ばれる。ログに記録されている内容が途中で切れている場合は処理を捨てて、ファイルシステムが不安定な状態にならないようにする。

このように、ジャーナリングファイルシステムとは、システムクラッシュでファイルシステムにどのような変更を行おうとしていたか忘れないように、ファイルシステムに変更を加える前にその内容をログとして記録する機能を持ったファイルシステムということができるであろう。このことからロギングファイルシステムと呼ばれることもある。

誤解を招きやすいが、巷で用いられているジャーナリングファイルシステムとはジャーナリング機能を持つファイルシステムという意味にすぎず、ファイルシステムの内部構造や内部動作まで決めるものではないことに注意して欲しい。

ジャーナリングファイルシステムの構造と動作

さて、ジャーナリングファイルシステムの概要については以上で理解できたと思う。要は「ファイルシステムに変更を反映させる前にログに記録する」ことがミソなわけだ。しかしながら、具体的にどんな内容を記録するのか、どのタイミングで記録するのかなどの方法に関しては、ファイルシステムごとに差別化を図っており、それぞれに特徴がある。

そこで今回は、単純なジャーナリングファイルシステムの構造を説明した後、基本的な動作について例を挙げて解説していきたいと思う。ここで述べるのはあくまで基本的な話であり、実際

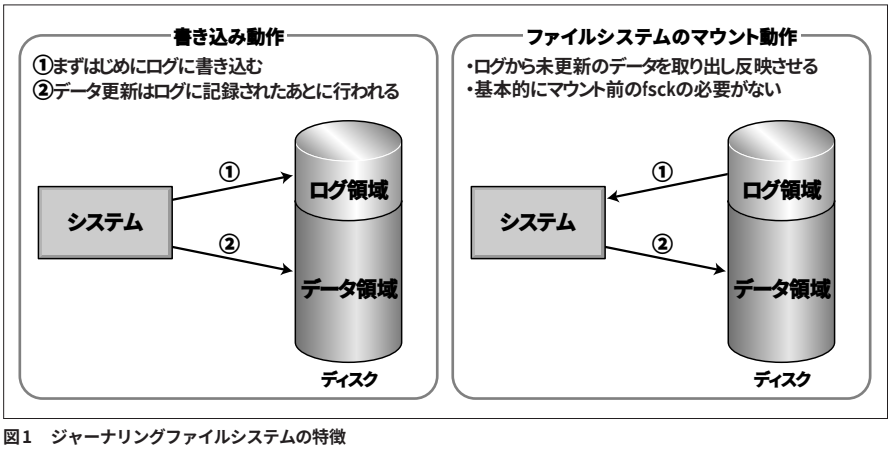


図1　ジャーナリングファイルシステムの特徴

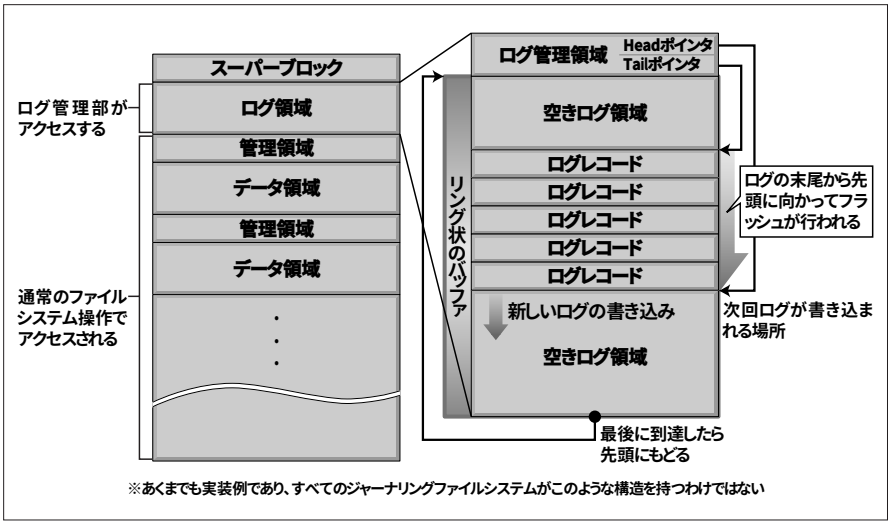


図2　ジャーナリングファイルシステムの内部構造

のジャーナリングファイルシステムはキャッシュやバッファ管理など他の要素が絡み合う複雑な実装になっている。

ジャーナリングファイルシステムの内部構造

ジャーナリングファイルシステムの内部構造は、**図2**に示すようにログ領域と通常のファイルシステム領域の2つに大別することができる。ファイルシステム領域はファイルシステムを実装するために必要な管理領域とデータ領域の構成になっている。ログ領域はファイルシステムに対する変更内容を表すログデータを格納するための領域である。ログの場所は実装によって異なり、専用パーティションに作成するものもあれば、ファイルシステムの先頭や末尾の領域を予約してログ領域として使用するものもある。また、Windowsのスワップファイルのように特殊なファイルとして実装しているファイルシステムもある。

このように、ディスク上の構造だけを見てみると、ジャーナリングファイルシステムとは通常のファイルシステム領域にログを追加しただけの単純な構成であることがわかるだろう。実際、ジャーナリングファイルシステムと呼ばれているファイルシステムの多くはジャーナリングをoffにすることができ、ジャーナリングなしでもファイルの読み書きなどの普通の操作は使用可能になっている。ジャーナリングファイルシステムはファイルシステムに対して変更内容の保存場所と記録する処理を追加することにより、可用性を向上させているのだ。

■ログの構造

ログの構造は一般に環状バッファになっており、ログの末尾まで来ると先頭に戻って書き込まれる。有効なログレコードが含まれている領域の先頭と末尾を指すポインタが存在し、ログが書き込まれると先頭のポインタが移動する。一方、ログの内容がフラッシュされると末尾のポインタが移動する。

ログにはファイルシステムの変更内容を表したログレコードが書き込まれる。このレコードは、ファイルシステムへの操作を表した値とそのデータから構成される。ログの領域は追加書き込みされていくだけであり、通常のファイルシステムの運用ではログの内容を読み込んだり、すでに書かれている内容を変更したりすることはない。

ログにi-nodeに代表されるファイルを管理するためのメタデータとユーザーデータの両方を記録する方式を「フルジャーナリング」と呼ぶ。通常のジャーナリングでは、システムクラッシュ後のファイルシステムの整合性を確保するという目的から、メタデータの変更のみ記録される。したがって、システムクラッシュが発生した場合、ユーザーデータの部分の復旧はできない。重要なデータの場合は同期書き込みを行う必要がある。

一方、フルジャーナリングの場合はユーザーデータもログとファイルシステムの両方の領域に書き込まれることになるので、システムクラッシュ後もログに残っているユーザーデータを復旧することができる。しかしながら、ユーザーデータをログ領域とファイルシステム領域に二重に書き込むオーバーヘッドのため、通常のジャーナリングに比べパフォーマンスは低下する。

■トランザクション

ジャーナリングファイルシステムでは、ログに記録する内容をトランザクション単位で行う。トランザクションとは、ユーザーがファイルシステムに対してある操作を行うときに必要な、ファイルシステム内部で行う一連の操作をいう。

たとえば、ディレクトリ以下にファイルを作成する場合を考えてみよう。ユーザーモードで動作しているアプリケーションでは、ファイルを作成するにはcreatシステムコールを発行する。アプリケーション側からはこのシステムコール1つだけでファイルが作成されるので、単一の処理に見えるかもしれないが、カーネルモードで動作するファイルシステムのコードは、指定され

たファイルシステム上にファイルを作成するために、**図3**のように①～⑤の処理をすべて完了させなければならない。

これらの一連の処理、つまりファイルを作成するトランザクションが無事完了すると、ファイルシステムの状態は新しいファイルが追加された整合性の取れた状態となる。

ジャーナリングファイルシステムの場合、ディスク上の空きブロックビットマップやi-nodeが含まれたブロックの更新を行う前に、必ずログにトランザクションを記録する。ログに書かれる内容はジャーナリングの実装によって異なるが、一例として挙げると、①から⑤の操作を、同一トランザクションIDが付いた個々のログレコードとして書き込む。システムクラッシュ時のログのリプレイ用にメタデータがログに書き込まれるので、空きブロックビットマップやi-nodeが含まれる更新されたイメージのブロックがログに含まれる。XFSなどはブロック全体ではなく、ブロック内の変更のあった部分だけがログに書き込まれるようだ。

さて、ファイルを作成する話に戻ろう。ファイルを作成するトランザクションがどのようにログに書き込まれていくかこれから説明したいと思う。

Step1 トランザクションの割り当て

まず、①の処理を行う前に、新しいトランザクションの割り当てを行い、これから行うファイル作成のトランザクションに対するIDを取得する。

Step2 ファイル操作のトランザクションへの追加

トランザクションIDの割り当てが成功したら、このIDを用いてトランザクション内で行う①～⑤の操作をトランザクションに加えて行く。ジャーナリングでない普通のファイルシステムの場合、①から⑤で行うメタデータの変更はディスクに同期的に書き込まれることが多いが、ジャーナリングの場合はディスク上のブロックを変更する前に必ずログに記録する必要があるので、この段階では書き込みを行わず、トランザクションごとに割り当てられている管理データとリンクされメモリに保存されていくような処理になる。ちなみに、この段階ではログにはまだ記録されず、内容はトランザクションにローカルなまま保持されている。

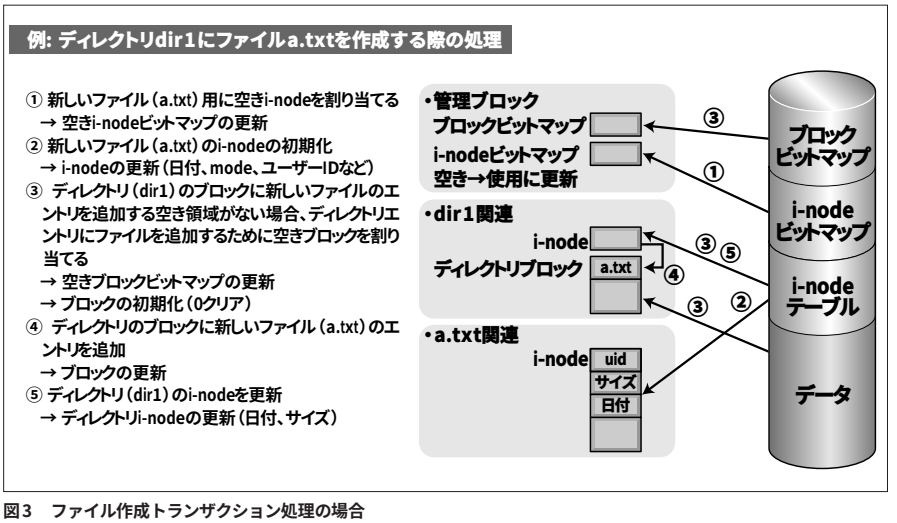


図3　ファイル作成トランザクション処理の場合

Step3 トランザクションのコミットまたはロールバック

上記の名前でデータベースの操作を思い出した人もいるかと思うが、内容も同じである。トランザクションのコミットは、Step2で構築したトランザクションの内容をログに記録する処理を行う。ロールバックは、トランザクション中に行った変更を無効にしたい場合の処理であり、エラーが発生した場合などに呼び出される。

■ログの管理

各トランザクションがコミットすることによってログに記録されていくと、やがてログの環状バッファはいっぱいになる。ログ領域の末尾にある古いログレコードが新しいログレコードに上書きされないように、ジャーナリングファイルシステムはログ領域がいっぱいに近づいてくると古いログレコードを削除するための処理が必要になる。

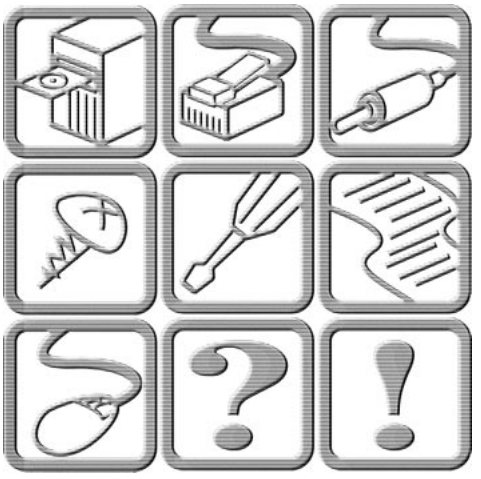
ログから削除可能なレコードは、レコードの内容がすでにファイルシステムに反映されているレコードである。実際にファイルシステムに反映されるタイミングは実装依存だが、トランザクションがコミットされてログ領域に記録されたあとは、システムクラッシュしてもログのリプレイによって復旧できるので、更新は非同期で行われることもある。この場合、非同期書き込みの完了時点で書き込みの該当トランザクションを占めるログ領域を開放することができる。

一般にログレコードの削除を専用のスレッドが行う実装が多い。ログがいっぱいにならないように気をつけないと、ログの空き領域を待ってI/Oが滞ってしまい、パフォーマンスが大幅に低下することにもつながるのだ。

Try & Try

[trái]

[trái]



NFS設定のそれから

文:政久忠由
Text : Tadayoshi Masahisa

今回はLinuxカーネル2.4のネットワーク機能を見ていこうと思いフィルタ機能を中心に調べていたのだけれど、部屋の片付けの際中にふと何かを見つけてしまいそちらに気が移ってしまった時のように、別のことを行ってしまっていた。

前々回くらいにノートPCにLinuxをインストールしたものの利用できるハードディスク容量が少ないので、ローカルネットワークに接続している場合はNFSを使用してディスク容量に余裕のあるLinuxコンピュータ（特にサーバ的な仕事をさせているわけではないし、複数のOSをインストールしてさまざまなテストに使用しているのだけれど、これといった用事のない時は、リモートから効率よく利用できるLinuxでとりあえず稼動させている）に作業領域を確保したという話をした。

その際、スーパーユーザーにスイッチしなくてもマウントできて、ファイルの実行パーミッションも有効にして、さらにrootユーザーでも作業が行えるような設定を施したわけなんだけど、これらに加えて転送パフォーマンスを向上させるためにmountコマンドのオプションとしてrsize／wsizeというNFSレベルの読み書きサイズを紹介した。

実は僕自身、この時点ではrsize／wsizeオプションに関しては調べてみたこともなく、いにしえの言い伝えをそのまま伝承しただけで、これといった確信があるわけではなかった。けれども我が家のネットワークインフラを整備し直すことになったので、今回はこのことについて少し調べてみることにする。

ネットワークを100BASEのスイッチングハブ構成に

今まで我が家のネットワークインフラは、10BASE-Tベースの特別な機能など何も付いていないハブを利用していたのだけれど、NFSを利用するようになって、そのローカルディスクとのレスポンスのギャップは予想を越えてかなりのストレスを感じていた。

僕の所有する形を成し稼動するコンピュータは計5台。このうち3台はノーマルPentiumの200MHz以下、あと2台はCeleronの400MHz台、とまぁお粗末なもので、Linuxをインストールして使用できるようになっているコンピュータでは、最近購入したノートPCのCPUが一番高速だったりする。こういった事情もあって、Linux関連はできればノートPCのCPUを利用して作業を行いたいわけなんだけど、先に述べたディスク容量の問題を解決すべく導入したNFSが低速なネットワークインフラに足を引っ張られてしまい、思惑通りにすんなりとはいかない状況であった。

ここで誤解しないでほしいのだけれど、10BASEネットワークでNFSが実用に耐えないと言っているわけではない。/usrや/usr/local以下など、実行プログラムを共有したり、/homeを共有したりすることに関して、10BASEネットワークでもそれなりに実用的だし、実際に利用され

てきた。実行プログラムのサイズは、100Kバイト以下がほとんどで1Mバイトを超えるものは希だし、一般ユーザーの扱うデータサイズもそれほど大きなものはないからだ。しかもそれぞれの作業過程では数個のファイルのネットワーク転送が発生するだけだ。けれども、僕が求めているワークステーション的な利用というか、数十Mバイトのソースコードを展開してそれをコンパイルしたりするのに使用するとなると、実質600Kバイト／秒程度のネットワーク転送速度では、最たるボトルネックとして著しく表面化してしまうのだ。

現状、10万円もあればそこそこの性能（僕の所有するものよりはるかに高速な）のパソコン一式を入手できるご時世である。少し前までは決してメーカー製PCを購入することはないと思っていたんだけど、ここ最近はハードウェアをいじるのも面倒だし、最速、最新ということにもトンと興味が薄れてしまっていて、とりあえずご機嫌に動いてくれさえすればいいや、というデカダンス傾向（不景気と言われているのはこのせいか？）にあるので、10万円を切る完成品でも買ってLinuxの作業マシンにしちゃおうかなとも考えていた。でも、そもそもの目的を考えるとネットワークを高速にしてみるという手段が今の僕の環境では将来的にももっとも効果的でコストパフォーマンスもよいような気がしたので、とりあえずはネットワークインフラを100BASEにリプレースしてみることにしたのだ（無線LANにもそそられたのだが、やはり速度優先）。

ずいぶん前から100BASEが当たり前だよ、と思う人も多いであろう。でも僕は100BASEにしたとしても期待ほどには実質転送速度が向上しないケースをあまりにも数多く見てきてしまったのだ。だから今まで導入を躊躇してきたのだけれど、現状、100BASEのスイッチングハブ（10BASEにも対応）が7000円程度で入手できるんだなぁ、これが。この金額だとリスクはほとんどないといっていい。幸い我が家のネットワークカードのほとんどは100BASEにも対応しているので買い足す必要はないし、ケーブルもカテゴリー5を使用しているので、単にスイッチングハブを買ってきてそのポートにケーブルを差し替えるだけで済む。

というわけで、我が家のネットワークインフラは100BASEのスイッチング環境となった。5台のPCのケーブルを接続し直しただけで特に何の問題もなく利用できるようになり、移行はあっさりと完了した。5人のユーザーがいてそのそれぞれがネットワークを利用している環境と違い、1人のユーザーが5台のPCを使用しているだけなので、各ポート間のパケット処理といった本来気になるスイッチング能力はそれほど重要ではない。だからそのテストはここでは見送り、とりあえず各PC間のFTPの転送速度を見てみることにした。

結果は予想通りまちまちで、8Mバイト／秒を越える転送速度をたたき出す組み合わせもあれば、なぜか500Kバイト／秒程度にとどまるケースもあった。特に同一のコネクション間でもputとgetで結果が大きく異なる場合もある。

一応、スイッチングハブでもあり、全二重通信もサポートしているので、パケットの衝突を気にせずがんがんに送信してくれてもそれなりに処理されるわけなんだけど、サーバ／クライアントソフトウェアレベルの処理のタイミングやネットワークデバイスドライバ／チップセットのパケットの送出タイミングなどさまざまな部分が影響しているので、一概に原因を特定することは非常に難しく、ものすごく面倒くさい。この問題についてはおいおい気が向いたときに調べることにする。

NFSでの性能

さて、本題のノートPCとLinuxサーバPC間のNFS接続の性能について見ていくことにしよう。それぞれのPCではLinux 2.3系のカーネルを使用している。カーネルのバージョンによってはNFSサーバ側でストールというかリセットが発生してしまうケースがしばらく続いて、テストにならないかなと危惧していたが、現在使用している2.3.99-pre8では今のところそのような問題は発生していないので、ここではこのバージョンを使用してテストを行っている。またネットワークのチップセットはIntel 82557/82558 OEMとVIA Tec. VT86C100Aで、それぞれカーネルに含まれるeepro100とvia-rhineのデバイスドライバモジュールを使用している。

環境としては、クライアント／サーバのPCともNFSの設定を次のようにしている。NFSv3の実装に関しては多少不安定という話も聞くが、僕のところではこれといった障害もないので有効にしている。

```
File systems  --->
Network File Systems  --->
<M> NFS file system support
[*] Provide NFSv3 client support (EXPERIMENTAL)
<M> NFS server support
[*] Provide NFSv3 server support
```

参考値としてFTPのテスト結果を紹介しておくと、ノートPC側からのgetで8.5Mバイト／秒、putで8.5Mバイト／秒（いずれも標準ftpコマンドを使用、サーバはwu-ftpd 2.6.0）である。当然多少のばらつきがあるものの、安定してこの速度を維持している。ただ希に3Mバイト／秒程度に低下してしまうことがあり、ncftpを使用すると300Kバイト／秒程度の転送速度しか出ないこともある。ではncftpだと性能が出ないのかというとそういうことでもなく、他のサーバからの転送では8Mバイト／秒程度のパフォーマンスを発揮する（うむ〜結構厄介）。一応参考値として最初に示した8.5Mバイト／秒を目安にすることにしよう。

NFSのテストは、rsize／wsizeをそれぞれ1024、2048、4096、8192、16384、32768、65536にしてファイルの転送に要する時間を計測することにした。またその際のようすをtcpdumpで取得している。

●rsize／wsizeが1024の場合

```
$ mount linux01:/mnt/1 /mnt/1 -o rsize=1024,wsize=1024
linux01:/mnt/1 on /mnt/1 type nfs (rw,rsize=1024,wsize=1024)
$ time dd if=/dev/zero of=src/test.dat bs=1024 count=10000
real 0m6.144s  user 0m0.030s  sys 0m1.220s
$ time dd if=/dev/zero of=src/test.dat bs=2048 count=5000
real 0m6.061s  user 0m0.050s  sys 0m1.270s
$ time dd if=/dev/zero of=src/test.dat bs=4096 count=2500
real 0m6.041s  user 0m0.020s  sys 0m1.190s
$ time dd if=/dev/zero of=src/test.dat bs=8192 count=1250
real    0m6.039s  user    0m0.010s  sys     0m1.050s
$ time dd if=/dev/zero of=src/test.dat bs=65536 count=153
real    0m5.934s  user    0m0.000s  sys     0m1.080s
$ time dd if=/dev/zero of=test.dat bs=4096 count=2500
real    0m0.094s  user    0m0.000s  sys     0m0.100s
$ time cp test.dat src/test.dat
real    0m6.122s  user    0m0.020s  sys     0m1.210s
```

#tcpdumpのログの抜粋

```
notepc.4046049708 > linux01.nfs: 1160 write [|nfs] (DF)
linux01.nfs > notepc.4046049708: reply ok 136 write [|nfs] (DF)
notepc.4062826924 > linux01.nfs: 1160 write [|nfs] (DF)
linux01.nfs > notepc.4062826924: reply ok 136 write [|nfs] (DF)
notepc.4079604140 > linux01.nfs: 1160 write [|nfs] (DF)
linux01.nfs > notepc.4079604140: reply ok 136 write [|nfs] (DF)
```

コマンド実行のオーバーヘッドも含めての時間を計測しているので正確なネットワーク転送速度ではないが、約10Mバイトの転送に6秒ちょっとということで1.6Mバイト／秒という値が導き出せる。なおbsのサイズを大きくすれば処理のオーバーヘッドが減るために多少時間が短くなる傾向にあることは付け加えておく。

テストの最後の2つはローカルディスクに書き込んだものと、それをcpコマンドで転送した結果である。ローカルディスクへの書き込みではバッファキャッシュへの書き込みなので100Mバイト／秒という値になってしまうが、実質的な物理ディスクの性能は12Mバイト／秒程度のものである。

tcpdumpのログでは、1160というwriteサイズに注目してほしい。このパケットサイズは、NFSのwsizeで指定した1024というデータサイズにIPヘッダとUDPヘッダ、そしてRPC要求コマンドとNFS操作コマンド（合計136バイト分）が付加されたものだ。またbsの指定サイズを大きくしてもNFSレベルの書き込みアライメントは1024のまま変わらない（ここが重要）。まぁそのためのwsize指定なんだけど。

なお、計測ではbs=512の場合も行ったのだが、10Mバイト分の転送では送信の途中でお休みが多発してしまい意味のある値が取れなかった。ただし、この場合writeパケットサイズは648バイトとなる。つまり512バイトのデータサイズに先ほどのヘッダ（136バイト分）が付加されたものであることがわかる。当然パケットが小さい分だけ転送速度が低下することは容易に想像できると思う。テストの最後に行っているcpコマンドの転送でパケットサイズが648になることはないので、あまり心配することはないだろう（たぶんだけど）。なお、cpコマンドでは4096バイト単位（通常Writeシステムコールはメモリページサイズが基準）での書き込み要求を行っているので、bsの指定サイズ4096の場合とほぼ同じ処理内容と考えていい。

ちなみにtcpdumpのログからは、NFSはUDPパケットなのでTCPのような余計なハンドシェイクが行われていないということと、それぞれの書き込みパケットに対する返答はサーバ側のアプリケーションレベルから行われていることが見て取れると思う。

●rsize／wsizeが2048の場合

```
$ mount linux01:/mnt/1 /mnt/1 -o rsize=2048,wsize=2048
linux01:/mnt/1 on /mnt/1 type nfs (rw,rsize=2048,wsize=2048)
$ time dd if=/dev/zero of=src/test.dat bs=1024 count=10000
real 0m6.149s  user 0m0.030s  sys 0m1.290s
```

```
$ time dd if=/dev/zero of=src/test.dat bs=2048 count=5000
real 0m4.455s  user 0m0.040s  sys 0m0.860s
$ time dd if=/dev/zero of=src/test.dat bs=4096 count=2500
real 0m4.309s  user 0m0.020s  sys 0m0.710s
$ time dd if=/dev/zero of=src/test.dat bs=8192 count=1250
real 0m4.308s  user 0m0.000s  sys 0m0.720s
$ time dd if=/dev/zero of=src/test.dat bs=65536 count=153
real 0m4.287s  user 0m0.000s  sys 0m0.810s
$ time cp test.dat src/test.dat
real 0m4.455s  user 0m0.000s  sys 0m0.820s
```

＃tcpdumpのログの抜粋（書き込みサイズ2048バイト以上の時）

```
linux01.nfs > notepc.3701332396: reply ok 144 setattr [|nfs]
(DF)
notepc > linux01: (frag 18100:712@1480)
notepc.3718109612 > linux01.nfs: 1472 write [|nfs]
                                        (frag 18100:1480@0+)
linux01.nfs > notepc.3718109612: reply ok 136 write [|nfs] (DF)
notepc > linux01: (frag 18356:712@1480)
notepc.3734886828 > linux01.nfs: 1472 write [|nfs]
                                        (frag 18356:1480@0+)
linux01.nfs > notepc.3734886828: reply ok 136 write [|nfs] (DF)
```

wsize=2048では、bsの指定サイズが1024の場合は、wsize=1024のケースと同じ1.6Mバイト／秒、bsの指定サイズが2048以上の場合は、2.3Mバイト／秒となった。この違いはtcpdumpのログを見ると一目瞭然である。bsの指定サイズが1024の場合はネットワークの書き込みも1024バイト単位で行われることになるが、bsの指定サイズが2048以上の場合は、それとは異なり、fragというフラグメント化パケットが使用されている。

フラグメント化とはパケットを分割して送っていることを表している。これはどういうことかというと、IPパケットレベルでは1つのパケットとして送信するものの、イーサネットレベルでの1パケット（普通フレームと呼ぶが）のサイズ制限（MTU）があるため、ネットワーク上では分割されて送信されているということなのだ。ちなみにMTU（メッセージ転送ユニット）はifconfigコマンドなどで次のように確認できる。通常イーサネットのMTUは1500バイトである。

```
$ /sbin/ifconfig
eth0 Link encap:Ethernet  HWaddr 00:80:45:10:AB:34
 inet addr:192.168.0.12  Bcast:192.168.0.255
                                        Mask:255.255.255.0
 UP BROADCAST RUNNING  MTU:1500  Metric:1
 RX packets:1068993 errors:0 dropped:0 overruns:0 frame:0
 TX packets:1302162 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:100
```

このMTUの制限のために、IPレベルで1つのパケットは2つに分割され、次のようなバケットとなっている。

```
notepc > linux01: (frag 18356:712@1480)
notepc.3734886828 > linux01.nfs: 1472 write [|nfs]
(frag 18356:1480@0+)
```

最初のパケットではデータサイズがそのまま表示されており、712バイトである（このパケットの実サイズは20バイトのIPヘッダが付加され732バイト）。続くパケットではヘッダサイズ136バイトを引くとデータサイズが求められ、1336バイトであることがわかる。これらを足すと2048バイト、すなわちNFSのデータサイズ2048バイトのパケットとして転送先では展開されることになる。

フラグメント化されたパケットは番号で管理されていて、上記の場合は18356がその対応番号である。またこれに続く数字はパケットを組み立てる際の順番を表している。

ネットワーク上では、各パケットが到着する順番は不確定なため、この番号を頼りに受け取り側ではパケットを再構築（アセンブル）するのだ。このことを悪用したネットワークの攻撃を耳にしたことがあると思う。フラグメントの指定がなされたパケットを受け取った場合、受け取り側はそれを再構成するためにネットワークリソースの一部分を使って待ちの状態になる。が、攻撃はフラグメントに対応するパケットをはじめから送信していないため、待ち受け側は待ちぼうけを食らうというわけだ。当然ネットワークリソースは有限なため、それを大量に送りつけられると待ちぼうけ処理でシステムは手いっぱいとなり、正常なパケットの処理まで滞ってしまうのである。

さて話は戻るが、フラグメント化を利用するとなぜ転送効率が向上するかというと、それは送信側ではIPパケット生成処理回数が減り、それにともない受け取り側では返答が少なくて済むからだ。wsizeが2048バイトの場合は、ネットワーク上ではパケットは2つに分割されているが、それに対する返答は1回で済むのである。このことが1.4倍程度の転送効率の向上につながっているのだ。

●rsize／wsizeが4096の場合

```
$ mount linux01:/mnt/1 /mnt/1 -o rsize=4096,wsize=4096
linux01:/mnt/1 on /mnt/1 type nfs (rw,rsize=4096,wsize=4096)
$ time dd if=/dev/zero of=src/test.dat bs=1024 count=10000
real    0m3.212s  user    0m0.010s  sys     0m0.180s
$ time dd if=/dev/zero of=src/test.dat bs=2048 count=5000
real    0m3.187s  user    0m0.000s  sys     0m0.170s
$ time dd if=/dev/zero of=src/test.dat bs=4096 count=2500
real    0m3.188s  user    0m0.000s  sys     0m0.130s
$ time dd if=/dev/zero of=src/test.dat bs=8192 count=1250
real    0m3.175s  user    0m0.000s  sys     0m0.140s
$ time dd if=/dev/zero of=src/test.dat bs=65536 count=135
real    0m2.738s  user    0m0.000s  sys     0m0.090s
$ time cp test.dat src/test.dat
real    0m3.197s  user    0m0.010s  sys     0m0.230s
```

＃tcpdumpのログの抜粋

```
notepc.4156939180 > linux01.nfs: 152 setattr [|nfs] (DF)
linux01.nfs > notepc.4156939180: reply ok 144 setattr
                                                [|nfs] (DF)
notepc > linux01: (frag 23134:1280@2960)
notepc > linux01: (frag 23134:1480@1480+)
notepc.4173716396 > linux01.nfs: 1472 write [|nfs]
                                        (frag 23134:1480@0+)
notepc > linux01: (frag 23390:1280@2960)
notepc > linux01: (frag 23390:1480@1480+)
notepc.4190493612 > linux01.nfs: 1472 write [|nfs] (frag
23390:1480@0+)
linux01.nfs > notepc.4173716396: reply ok 136 write [|nfs] (DF)
```

wsizeに4096を指定した場合、転送速度は3Mバイト／秒を超えた。また僕の予想に反し、bsの指定サイズが4096未満の場合（512も含めて）でもネットワークパケットを4096バイトでアライメントして転送を行っている。

tcpdumpのログからパケットのフラグメント化の状態は、4096バイトが1280バイト、1480バイト、1336バイトの3つに分割されていることがわかる。特に1480バイトのデータサイズのパケットは、MTUの制限1500バイトのうち、最低限のヘッダ情報20バイトを付加しただけのもっとも転送効率のよいパケットとなっている。さらにリプライが多少遅れて記録されていることからもわかるように、パケットの送出は前のパケットの成功、不成功に関わらず次々と行われていることも見てとれる。

●rsize／wsizeが8192の場合

```
$ mount linux01:/mnt/1 /mnt/1 -o rsize=8192,wsize=8192
linux01:/mnt/1 on /mnt/1 type nfs (rw,rsize=8192,wsize=8192)
$ time dd if=/dev/zero of=src/test.dat bs=1024 count=10000
real    0m2.553s  user    0m0.020s  sys     0m0.150s
$ time dd if=/dev/zero of=src/test.dat bs=2048 count=5000
real    0m2.512s  user    0m0.000s  sys     0m0.120s
$ time dd if=/dev/zero of=src/test.dat bs=4096 count=2500
real    0m2.501s  user    0m0.010s  sys     0m0.140s
$ time dd if=/dev/zero of=src/test.dat bs=8192 count=1250
real    0m2.436s  user    0m0.000s  sys     0m0.140s
$ time dd if=/dev/zero of=src/test.dat bs=65536 count=135
real    0m2.161s  user    0m0.000s  sys     0m0.090s
$ time cp test.dat src/test.dat
real    0m2.574s  user    0m0.000s  sys     0m0.200s
```

＃tcpdumpのログの抜粋

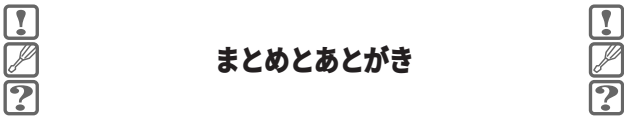
```
linux01.2348998060: reply ok 144 setattr [|nfs] (DF)
notepc > linux01: (frag 64062:936@7400)
notepc > linux01: (frag 64062:1480@5920+)
notepc > linux01: (frag 64062:1480@4440+)
notepc > linux01: (frag 64062:1480@2960+)
notepc > linux01: (frag 64062:1480@1480+)
notepc.2365775276 > linux01.nfs: 1472 write [|nfs]
                                        (frag 64062:1480@0+)
notepc > linux01: (frag 64318:936@7400)
notepc > linux01: (frag 64318:1480@5920+)
notepc > linux01: (frag 64318:1480@4440+)
notepc > linux01: (frag 64318:1480@2960+)
notepc > linux01: (frag 64318:1480@1480+)
notepc.2382552492 > linux01.nfs: 1472 write [|nfs]
                                        (frag 64318:1480@0+)
linux01.2365775276: reply ok 136 write [|nfs] (DF)
```

wsizeに8192を指定した場合、転送速度は4Mバイト／秒を超え、プログラムによっては5Mバイト／秒程度まで向上することがわかる。また4096を指定した場合と同様、bsの指定サイズがどのような場合でもネットワークパケットは8192バイトでアライメント、そして送信されている。

tcpdumpのログの内容からは、最大限にデータパケットを納めたパケット4つと936バイト、1336バイトを納めるパケットの6つで1つのパケットが構成されており、リプライの回数がかなり軽減されていることがわかる。

●rsize／wsizeが8192より大きい場合

rsize／wsizeに8192より大きな値、16384、32768、65536を指定した場合、残念ながら現時点では8192が設定されるようになっている。すなわち結果は8192の時と同じであった。



まとめとあとがき

NFSv2まではパケットサイズの最大値に影響を与えるFSINFOという構造体のrtmaxとwtmaxはそれぞれ8192が最大となっていたのだけれど、NFSv3では32768まで拡張されている。そこでNFSv3の設定を有効にしてカーネルを作成してテストを行ってきたのだが、マウントの段階でのネゴシエーションはNFSv3プロトコルで行われているもののNFSサーバからの返答はrtmax、wtmaxともに8192のままであった。そのためrsize／wsizeで8192より大きい値を設定しても8192の設定にしかならないのだ。

ちなみにLinuxカーネルに含まれるNFSの実装では、NFSv2でrsize／wsizeのオプションを指定しないときのデフォルト値は1024、NFSv3で8192となっている。つまるところNFSv3を有効にしている場合は、何も設定しなくても現時点でもっともパフォーマンスのよい設定になっているということだ。たぶん近いうちにNFSv3の実装は変更されると思うので注意しておくことにしよう。

あと32768バイトではなく65536バイトを指定できたほうがさらに効率が上がるはずだが（システムの最大I/Oサイズによくある数値）、ネットワークのパケットを扱う処理部のオブジェクトメモリサイズ制限、rmem_max、wmem_maxの設定が65536バイト（Linuxはデフォルト値もこのサイズ）となっているので、IPヘッダ情報などを含めるとNFSのデータサイズは当然これらより小さくなくてはならない（先を越されたわけだ）。そのためにこの条件でのメモリアクセスの処理効率を考慮したアライメントとなると、32768バイトというサイズになってしまうのだ。

そういえばフラグメント化パケットで隙間がもったいないと感じた人も多いと思う。たとえばアライメントが8192の場合、データパケットサイズが936バイトのものにはあと544バイトも追加できる。しかし、NFSはファイルシステムの中間に位置し、アプリケーションからはシームレスにファイルシステムとして利用されていることを考えてほしい。アプリケーションからの要求の多くはセクタサイズ（512バイト）を意識しており、cpコマンドの例のようにCランタイムやシステムコールレベルではメモリページサイズ（Intel系は4096バイト）が処理単位として一般に使用されている。もしネットワークのパフォーマンスをよくしようとして8192ではなく、544バイトを追加して8736バイトでアライメント処理を行ったとすると今度はネットワークの出入り口で余計な中間バッファリング処理が発生してしまい全体のパフォーマンスに悪影響を与えてしまうことになる（バッファキャッシュの割り当て単位はページメモリと同じだし、最終的なディスクI/Oだってクラスタやセクタ単位なのだ）。これらを考慮するとどうすべきかが自ずと見えてくるだろう。パケットサイズのアライメントも512バイトの偶数倍、できればページメモリサイズの倍数であることが一番スムーズに流れるのだ。実際、mountコマンドのrsize／wsizeオプションで5000といった中途半端な値を指定したとしても1024の偶数倍に丸められるようになっている。

今回のテストで明らかなようにIPレベルのパケットサイズを大きくすればするほど生成の手間、そして返答すべき回数が減り（送出側も確認の手間が減る）、ネットワークの転送パフォーマンスは向上する。ただこれにはやはり注意点がある。それはミスしたときのペナルティが大きいことだ。フラグメント化パケットは確かに効率はよい。しかし、関連するパケットのうちどれかひとつでもロスしてしまうとその全体を再送する必要がある。さらにパケットの衝突の多い（コリジョンが頻繁に発生する）ネットワークではこのまとめ送りが影響し、輻湊状態となる危険性も非常に高くなる。よって一般的には最高の速度性能を求めるだけではだめで、ネットワークの特性に応じて適切な設定を行うことが大切になってくるのだ。まぁ普通ローカルエリアネットワークで、しかもスイッチングハブを導入して、全二重通信が可能であれば、パケットのロスや衝突はほとんど心配しなくてもよいのだけれど（テスト中パケットがロスしたこともないし）。

以前使用していたダムハブを用いた10BASEネットワークではさすがに無茶な要求だなと思っていたのだけれど、スイッチングハブを用いた100BASEネットワークにリプレースすることで予想以上に使えるじゃんというのが率直な気持ちだ。NFSでコンスタントに4Mバイト／秒、ときには6Mバイト／秒の転送速度が得られたという事実は、8Mバイト／秒程度のローカルディスクがあるのに匹敵すると個人的にほくそ笑む次第である。実際使用していてストレスを感じない。今後NFSv3がきちんと実装されて、NFSのパケットサイズをさらに大きくできればFTPの転送レートに肉薄するだろうから、言うことなしだね。

ツールを組み合わせることの妙

Emacsはじめました

第5回　続・メールを使う

Mewでひとわたり操作ができるようになったところで、日々の活用にうれしいヒントの数々を開陳。世論調査によれば、インストールはLinuxを使う大きな喜びだといいます（近隣の友人・知人調べ）。今回は、Mewをとりまくお助けツールも各種登場し、そういった点も満喫いただけるはず。

文：佐々木太良
Text : Taroh Sasaki

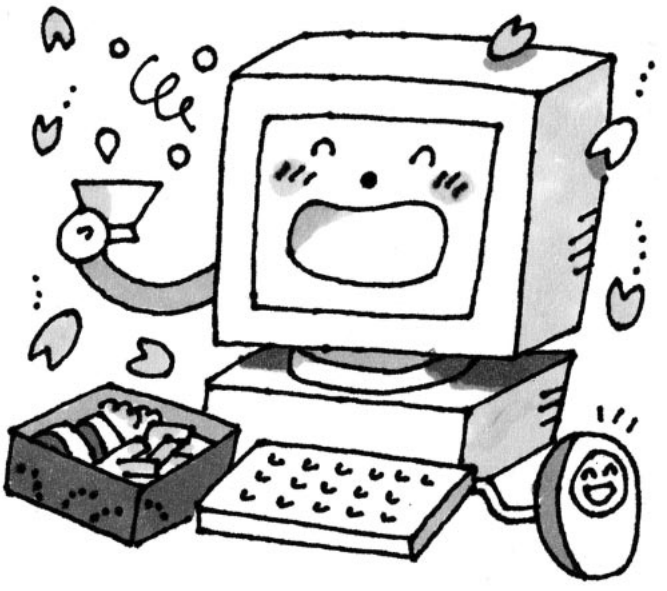


Illustration : Manami Kato

はじめに

じめじめと蒸し暑い季節ですが、皆さん元気にEmacsしてますか。出歩くのも億劫なので家でコンピュータばかりいじっていると、指の間がカビたりしますが、Emacsならコントロールキーをよく使うので指が開いてとっても健康的ですね。などと、わけのわからない出だしになってしまいましたが、今月も引き続き「Emacsでメールをやりとりしよう」というテーマでお送りしましょう。Mewの高度な操作を紹介し、ほかのツールとも組み合わせてより実用的に使ってみます。これで家に閉じこもっていても大丈夫……?

より高度なMewの使い方

Windowsだと、ブラウザ付属のメーラだとか、愛を全世界にばらまいてしまうメーラで、ちょちょっと画像ファイルをつまんで入れて「添付ファイル」（正確には「MIMEマルチパートメッセージ」といいます）なんてことができるのに、Linuxではどうしてこんなに苦労しなければならないのじゃあ、というあなた。Mewを使えば、MIMEなメッセージも簡単に表示したり作成したりできるのですよ。

■MIMEなメッセージを読む

さて、サマリ表示の画面をもういちど見てみましょう。メッセージ番号の次に“M”がついているものがあったら、それはMIME形式でエンコードされたマルチパートからなるメールです（**画面1**）。画像やプログラムなどのファイルをメールに添付すると、添付したファイルはメールの本文テキストとは別のパートとして送られます（**図1**）。だれかからのメールを転送するときも、もとのメッセージは画像と同様に添付ファイルとして本文とは別のパートで送られます。封筒の中に、書状のほかさらに現金入りの封筒が入っている状態を想像してください（ちなみに、普通郵便で現金を送ってはいけませんよ）。

Mewは〈SPC〉キーを押していくだけでメールが読める、楽ちんなツールですが、これはマルチパートメッセージについても例外ではありません。本文を表示しているときにそのまま〈SPC〉キーを押していけば、通常は次のパートが表示されます。メッセージがあまりにも巨大だと、MIMEの解析はせずに、〈.〉で解析が始まります（とミニバッファに表示されます）。また、添付した内容が転送したメールなどのテキストならそのまま表示されますが、バイナリなら**画面2**のようになります。**C-c C-e**でMIMEタイプに対応しているアプリケーションが起動され、〈y〉でセーブすることもできます。**画面2**の例では、パート内のデータ形式を示すタイプが“images／x-MS-bmp”で、アプリケーションとしてxvが指定されています。

キー操作	動　作
C-c C-a	末尾に新しい空のパートを用意する
c	ファイルをコピーして添付
B	Base64エンコード（バイナリファイルの場合）
T	MIMEタイプ（テキスト、画像、音声……）を変更
C-b C-p	（先頭パートから）本文に戻る

表1　MIMEメッセージを書くのに使用するコマンド

タイプが“text/html”でも、C-c C-eでNetscapeが立ち上がるのがうっとうしいときは、**C-c 〈TAB〉**でパートの内容をEmacsのバッファに表示できます。

■MIMEマルチパートメッセージを作成する

作成中のメールにMIMEパートを付加したい場合は、まず**C-c C-a**を押します。この時点で、本文末尾に添付領域が用意され、空の新しいパートが付いて、そこにカーソルが移動します（**画面3**）。

この状態から、〈c〉でファイルを新しいパートにコピーすることができます（このとき、展開されたあとのファイル名を聞いてきます）。

取り込まれたファイルには内容が何かを表すデータの型（MIMEタイプ）やファイル名が自動的に付けられ、画像やプログラムなどのバイナリファイルであれば、通信規約どおり7ビットデータで送れるように、自動的にASCII文字データに変換されます（**画面4**）。もしもバイナリデータがテキストと判断されてしまっている場合は、〈B〉でBase64エンコーディングして文字化します。いったん文字化したファイルは、MIME対応のメーラで復元できるはず。さらに、ファイルに付けられたMIMEタイプが違っていると思われる場合は、〈T〉でタイプを変えることができます。どんなタイプがあるかは、補完を試してみてください。

さて、このようにして添付領域にいくらでも新しいパートを付け加えていくことができますが、本文の編集に戻りたくなったら、一番上のパートまでカーソルを移動させてから**C-b**、さらに**C-p**で本文の領域に戻ることができます。

このほかにも豊富な操作があるので、詳細はメニューの［Mew/draft］－［Attatchment commands］などを一

画面1　サマリバッファに表示されているMIMEメッセージ

画面2　マルチパートメッセージ

画面3　空のパートを追加する

画面4　ファイルを添付する

ヘッダ

```
From kuma@ascii.co.jp Wed May 24 07:34:51 2000
From: kuma@ascii.co.jp
To: taroh@ascii.co.jp
Subject: Happy Birthday!
Date: Wed, 24 May 2000 16:34:51 +0900
Received: from localhost by mail.pb.ascii.co.jp (8.8.8/3.6W) id QAA22150 for <taroh@ascii.co.jp>; Wed, 24 May 2000 16:38:28 +0900 (JST)
X-Mailer: Mew version 1.94.1 on Emacs 20.5 / Mule 4.1(葵)
Message-Id: <20000524163451N.kuma@ascii.co.jp>
X-Dispatcher: imput version 990905(IM130)
Lines: 56
X-UIDL: fcadd5c4e620323632faaa2ffbf28228
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="V2VkLCAyNCBNYXkgMjAwMCAxNzozNjo1NyArMDkwMA=="
Content-Transfer-Encoding: 7bit
```

パートの区切り文字を定義

```
--V2VkLCAyNCBNYXkgMjAwMCAxNzozNjo1NyArMDkwMA==
```

区切り

本文のパート

```
Content-Type: text/plain

お誕生日おめでとう。

ぼくが作った詩と、好物(の絵)をあげよう。

  くま
```

```
--V2VkLCAyNCBNYXkgMjAwMCAxNzozNjo1NyArMDkwMA==
```

区切り

添付ファイルのパート（gif）

```
Content-Type: image/gif; name="honey.gif"
Content-Disposition: attachment;
 filename="honey.gif"
Content-Transfer-Encoding: base64
```


```
R0lGODlhlgBkAPcAAAAAAP//////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////ywAAAAlgBkAAAI/gADCBxI
sKDBgwgTKlzIsKHDhxAjSpxIsaLFixgzatzIsaPHjyBDihxJsqTJkyhTqlzJsqXLlzBjypxJs6bN
mzhz6tzJs6fPn0CDCh1KtKjRo0UBKF3KtGlTpCCdSp1KtapTqB2tapUKcSnWjE9PWv060SvJrUzJ
PkTLtq1Zg2/VJnRLt6rDuHIL4qW5N6/AvjOV+oUrWGfhwQMP51SMmLFNwHIdB06LODEAnGkhf5Xs
8moAyoNBv/Rs+XJj0StRE9RstK7brqr1cj6a2XVbhGEX5obK2mJd2LODxv44lmJvnsN/Jvd5HHlw
56Z5PzccXXp1oM2VX2c+XWj3x9ul/nv/Ljz8TvLlxzdEH5M9zOPuR5tfPD27zPid56/Wj5k/+PX+
ySdberrhJ9Z1BrJUX4D5IcjgZAwlyNFejkmo0oImcbWfg9tZmBKGvvlnV2kkEkagQhKyFhdeXgHm
IUoguqiaYityKFtyLx4Y4WXD7faXabsdFpuP+504l4YbftYhkKBRhZtmOWbIX1iMmVXhU0IiSZh+
9rU3JY3VvXWllWEGCGWUUnIJZpI/bhndmjsW+OBks1VZZpE34vnjl18SBZmdeippYqBT3ajmnDXJ
OKBlgbbZqJa9oXmhZICWKOiijLqZJ4qSftjlnkSuNaac1iEKaoKUBvcpfauKdKapedAlWmer53Uq
UaighkarRrguV2qD4fm62a4XzSjssLZWdKxauE6aLHTEFhstUlqmCauu08aZbWTbHqnktZVl6iqP
4IbraEgtlmtut1uay6tWIarrbmm/kTpvSbaxe++tW+3r778AByzwwAQXbPDBCCes8MIMN9xTQAAA
Ow==
```


```
--V2VkLCAyNCBNYXkgMjAwMCAxNzozNjo1NyArMDkwMA==
```

区切り

添付ファイルのパート（日本語テキスト）

```
Content-Type: application/octet-stream; name="sonet.txt"
Content-Disposition: attachment;
 filename="sonet.txt"
Content-Transfer-Encoding: base64
```


```
gq2C3ILMjl2JzA0KDQqCvYLngsGBQYK9gueCwYFBgr2C54LBgUENCpCwguqCvYvzgUGCzYK/gt2C
woKigsGCz4KigUkNCoK9gueCwYFBgr2C54LBgUGCvYLngsGBQQ0Kl3qLQ4LIjEaQtg0K
```


```
--V2VkLCAyNCBNYXkgMjAwMCAxNzozNjo1NyArMDkwMA==--
```

区切り

図1　マルチパートメッセージ

Column

マルチパートのタイプとアプリケーション

　マルチパートメッセージの添付領域では、各パートの中身がどういう性質であるかを示すために、「タイプ／サブタイプ」の形式のMIMEタイプ（Content Type）がヘッダに含まれています。これは、RFCで規定されたものです。たとえば、JPEGの画像であれば、Image/Jpegとなります。タイプには以下のようなものがあります。

`Text` ……テキストファイル
`Image` ……静止画像
`Video` ……動画像
`Audio` ……音声
`Application` ……そのほかのバイナリファイル
`Message` ……ほかのメールを丸ごと
`Multipart` ……さらに他の`MIME`形式を含んでいる場合

　どのファイルのときどのアプリケーションにどうやってデータを渡すか、という記述は、Mewがインストールされているディレクトリ下のmew-unix.elファイルにあります。立ち上げるアプリケーションを変更したいときは、このファイルを見てカスタマイズするとよいでしょう。

覧してみるとよいでしょう。

シグネチャを付ける

メールの末尾には、署名がついているのがマナーとされています。もちろん、本文をすべて書き終わったあとに、あらかじめ用意しておいたシグネチャファイルを挿入してもかまわないのですが（ごく普通のEmacsのコマンドとして**C-x i**で挿入できることは覚えていますか？）、Mewでは文章を書いている最中でも**C-c C-i**でシグネチャを末尾に付加できます。

このとき付加されるのは、~/.signatureというファイルの内容です。あらかじめ凝ったものを（いや凝ってなくても別にいいんですが）用意しておきましょう。

日本語圏以外の友人やメーリングリストに投稿することが多い場合、複数のシグネチャファイルを切り替えたいことがあります。ISO-2022-JPコード（通称JISコード）を使っているかぎり、本文に日本語を書くことはネットワークの規約には違反しませんが、英語圏の人が使っているメーラだとメール全体が化けて表示されたりと、トラブルもあるようです（メーリングリストにうっかり日本語のシグネチャがついた記事を投稿して怒られたことがあります）。

こんなときのために、c-sigというシグネチャを選ぶツールがあります。http://www.threeweb.ad.jp/~kshibata/c-sig/ を参照してみてください。このページからダウンロードしたファイルをEmacs Lisp（elisp）のディレクトリに置きます。Emacs Lispのディレクトリはディストリビューションによって違うのですが、**C-h v** のあとに“load-path”と入力してごちゃごちゃ表示されるディレクトリ……/usr/local/emacs/site-lispなどです。その後.emacsに**リスト1**を書いておくと、対話的にいろいろなシグネチャを切り替えられるようになります。

いろいろ試してみた結果、

●日本語のシグネチャは、~/.signatureをC-c C-iで付加する。
●英語（英数記号のみ）のシグネチャは、~/.ascii.signatureをC-x i . a s〈TAB〉で付加する。
（~/の状況によって手間は違う）

で使い分けるのが一番楽に感じます。

書きだしも入れてしまおう

ところで余談になりますが、メールの書きだしは、「佐々木です。」などとすることが多いようです。もちろん、ヘッダのFrom:（差出人）行を見れば誰のメールかわかるのですが、Mewの快適メール環境だとスペースを叩いていくだけでぽんぽんメッセージを斜め読みできるので、本文の最初にも入れておくと注目度200%アップ、というわけです（文章の書きだしで名乗らないとその後の筆が進まない、という人もいるようです）。

こんな人は、**リスト2**を.emacsに入れてみましょう。~/Mail/.draftに用意されたファイルが自動的に挿入され

▼リスト1　.emacsにc-sigの設定を付け加える

```
(autoload 'add-signature "c-sig" "c-sig" t)
(autoload 'delete-signature "c-sig" "c-sig" t)
(autoload 'insert-signature-eref "c-sig" "c-sig" t) ……シグネチャを対話的に選択する
(add-hook 'mew-draft-mode-hook                      ……メッセージの作成中、C-c C-iをシグネチャの挿入に使用する
  (function (lambda () (define-key mew-draft-mode-map
   "\C-c\C-i" 'insert-signature-eref)))
)
```

▼リスト2　.emacsに書きだしの自動化を付け加える

```
(add-hook 'mew-draft-mode-newdraft-hook    ……メッセージの新規作成時の処理を追加
  (function (lambda() (let
    ((p (point)))
    (goto-char (point-max))
    (insert-file "~/Mail/.draft")          ……書きだしの文句を挿入
    (goto-char p)))
))
```

るようになります。

メッセージを振り分ける

メーラとして専用に作成されたアプリケーションのなかには、差出人やSubjectを見て到着したメールをフォルダにほうりこんでくれる、自動振り分け機能がついていることがあります。まぁ、かなりの達人でも使っていない人は使っていない機能なので、ついているから良いメーラだというわけでもありません。ではMewはというと……先月号で見てきたように手動でメールを整理することはできます（当たり前ですね）が……自動でメールを振り分ける機能はないんですよ（しくしく）。

しかしそこは、単体で完結してしまっているスタンドアロンのメーラと違うところ、他のツールと組み合わせたりEmacsのマクロを作ったりで、いくらでも機能を付加することができるのです。この点、どんなにプログラミング能力があっても、自分でツールを改良できる可能性がゼロのソースコード非公開の製品とはわけが違います。多くの人が改良に加われることによって、時間の経過とともに、ソフトウェアの到達点もずいぶん違ってくるでしょう。筆者がMewなどのEmacs上のメーラを使い「続ける」理由のひとつは、そういうところにあるのです。

半自動振り分け――その1

Mewでのメッセージの整理――手動振り分けを復習してみましょう。たとえば、+inboxなどのサマリの一覧にカーソルがあるとき、〈o〉を押すと、

```
Folder name: (default +hoge) : +
```

と表示されます。デフォルトのフォルダ名でよければそのまま〈RET〉キーを押し、別のフォルダにしたいときは“＋”の後ろに入力する（補完もききます）のでした。一番簡単な振り分け方法は、これを「繰り返し実行」してしまうことです。たとえば、出したメールが相手のMTAのデーモンさんから戻ってきたときは、たいていSubject:の先頭が“Returned mail:”となっています。そこで、これをあらかじめ作成しておいた+daemonフォルダに振り分けるには、まずサマリバッファでマクロを定義する次のシーケンスを1回実行します。

```
M-x goto-line 1〈RET〉   ……1行目に移動する
C-x (                   ……キーボードマクロの始め
C-s r e t u r n e ...   ……“Returned...”を検索
C-a                     ……検索終了（行頭に移動する）
o d a e m o n〈RET〉     ……+daemonフォルダに整理ⓐ
C-x )                   ……マクロ終了
```

このあとで、

```
C-u 1 0 0 0 0 C-x C-e
```

とすると、マクロが1万回実行され、サマリ末尾まで走査してすべてのReturned mail:の行に整理マーク（o）を付けてくれます。あとは〈x〉で整理するだけですね。

なお、マクロの手順中ⓐの行を**C-u 1 d**とすると、条件にあてはまるメッセージを削除してくれます。どうして単純に〈d〉にしないかというと……試しに不要なメッセージに対してこの2つを手で実行してみてください。後者はメッセージウィンドウを開いてメッセージを表示してしまいますので、キーボードマクロから1万回（やらないと思いますが）繰り返し実行した場合は、数倍の時間がかかってしまいます。

話を元に戻しましょう。削除ならこのように時間を節約することもできるのですが、今回の主目的である〈o〉で整理する場合は、どうやってもメッセージウィンドウが開いて、上記の繰り返し実行がスローに実行されてしまいます。遅いマシンだと整理しているのが目で見えるほどです（泣）。

そんなわけで、ここで述べた方法はかなり原始的ですが、後述のprocmailを用いたかなり先進的な方法を日常的に使っている場合でも、わざわざ整理ルールを書くほどではないときや、すでに+inboxに取り込まれてしまったメールを整理したいときなど（たとえば、だれかのパーティーに関するメールが2～3日飛び交ってそれでおしまいのとき）、筆者はいまだに常用しています。

どうですか、検索と繰り返しという、Mewのためではない機能を活用して整理ができることからも、Emacsの懐の深さがうかがい知れるでしょう。

半自動振り分け――その2

ここで紹介するものも基本的には「その1」と同じ方法ですが、Mewの機能をもう少し活用してみましょう。「そ

の1」では、サマリが表示されているバッファをテキストファイルとして検索した結果、合致するものを選び出していました。したがって、サマリバッファに表示されないほど長く“Re: Re:……”（返信に返信を繰り返すと、しばしばこうなります）がついたときは、Subject: のパターンでは検索できず、そもそもサマリに表示されないCc: のパターンは条件として設定できないことになります。

先月号で説明したとおり、こうしたパターンはMewのコマンド〈/〉で検索できます。したがって、先ほどと同じことをするには、サマリ表示バッファで、

```
/〈RET〉
```

として検索を開始します。“Folder: （+inbox）”と聞いてくるので、+inbox（現在いるフォルダ）でよければそのまま〈RET〉を押します。次に“pick pattern: ”と聞いてくるので、条件として、

```
subject=Returned mail:
〈RET〉
```

を入力します。subject以外の条件がわからなければ、〈SPC〉を押してみましょう。しばらく待つと、条件に当てはまるものだけが+inboxに再表示されます。こうしておいてから、次のシーケンスを1回実行します。

```
M-x goto-line 1〈RET〉   ……1行目に移る
C-x (                   ……キーボードマクロの始め
o d a e m o n〈RET〉     ……+daemonフォルダに整理
C-x )                   ……マクロ終了
```

このあとで、

```
C-u 1 0 0 0 0 C-x C-e
```

のようにマクロを1万回実行します。後半の手順は、先ほどと同じですね。これで表示されているものがすべて整理されて（削除されて）+inboxが空になったように見えますが、別になくなってしまったわけではありません。整理が終わったら再度、

```
s all〈RET〉             ……スキャンしなおし
```

で+inboxの全サマリを表示しておきましょう。

▼リスト3 .procmailrcの例

```
MAILDIR=$HOME/Mail
LOCKFILE=$HOME/Mail/.lock
LOGFILE=$HOME/Mail/from-log

:0                                  ←レシピ
* ^Subject: testa   ……パターン
testa/.             ……アクション

:0
* ^Subject: testb
testb/.

:0
* ^From: taroh@taroh.org
from/taroh/.

:0
inbox/.
```

自動で振り分け

次はいよいよ、メールを自動振り分けしてみましょう。これは、Emacsとはまったく関係ない外部コマンドによる振り分けです。シンプルな道具を組み合わせて、何でもついた巨大なアプリケーションより強力なことができるというのがUNIXの精神です。実際、この方法ではほかの専用のメーラでは不可能なほど変な振り分けが可能です。

ここで使う外部コマンドは、procmailというものです。Red Hat系のLinuxの場合、RPMを使って簡単に導入することができます。procmailの機能というのは、

- 入力からメッセージを1つ読む
- 特定のフォーマットに当てはまっていたら
  - あるディレクトリに（番号を適当につけて）保存する
  - またはメールとして送りなおす
  - または……（省略）

というシンプルなものです。ここで想定されている使用環境というのは、常にオンラインにあるワークステーションです。MTAは1通のメールを受信したとき、個人のシステムメールボックスそれぞれに保存するのではなく、個人別の設定によってprocmailを起動し、しかるべきフォルダに整理したり別のアドレスに転送します。MTAにsendmailを使っているマシンであれば、procmailは~/.forwardファイルから起動されます。

先月紹介したように、ダイヤルアップした際にPOP3／IMAPを使ってメールを取りに行くだけのパーソナルLinuxマシンでは、MTAはプロバイダのマシンで動いていてメールを取りに行く過程はMewが全部やってくれるわけですが、procmailの出る幕はないのかというと、そんなことはありません。メールを取りに行くMewの機能を使わず、オフラインでメールを読む機能だけを使えばよいのです。すでに先月号でMewのインストールと設定に成功している人は、ネットワークに接続していないときでもMewを起動すれば（パスワードを聞かれて、〈RET〉などで無視するとコネクションエラーが出るでしょうが）メールボックスにあるメールを読めることをご存じでしょう。そこで、メールを取りに行くのに、Mewではなく専用のfetchmailというツールを導入して使うことにします。

これで自動振り分けは達成できるのですが、いきなり到着したメールが振り分けられてしまっていると、Mew（Emacs）を起動したときにどれが新着メールか見落とすおそれがあります。それを回避できるのがprom-mewという、Emacsのマクロで書かれたMewお助けツールです。

な〜んか面倒だな〜、ですって？　たしかに次から次へとツールをインストールして、振り分けするだけで1個大隊ですが、カスタマイズが自由で機能は超強力、Mewと関係ないところでも使えるというおまけまでついているので、勘弁してお付き合いください。

procmailの設定と動作確認

さてさて、次から次へと新しいツールを紹介しましたが、順を追って設定していかないと、動かなかったときにどこが原因かわからなくなります。まずは、procmail本来の使い方で動作を確かめてから、プロバイダにある自分のシステムメールボックスに取りに行くことにしましょう。

RPMからの導入が終わったら、**リスト3**のようにprocmailの設定ファイル~/.procmailrcを作ります。

先頭の3行は、~/Mailというディレクトリに自分のMew用メールフォルダを作ることを指定します。その次の“:0”と書かれた行ごとに、振り分けルール（procmailのレシピ）を書いていきます。

はじめの2つのレシピは、テスト用に作ったものです。これは、このあとのテストに使います。“:0”の次の“＊”で始まる行は、単純にUNIX系のegrepコマンドと同様の働きをしますが、大文字と小文字を区別しません。この行に指定したものがメッセージのヘッダに含まれるとき、最後の行のアクションが実行されます。マッチングはegrepと同様ですので、たんなるパターンの一致だけではなく、正規表現による超強力な条件を与えることができるのです。

パターンは省略したり、複数書くこともできます。パターンを省略すると（**リスト3**の例では、最後のレシピがそうです）すべてのメッセージに合致します。複数書くとすべてのパターンに合致したときにアクションが起こります。

そのほかprocmailは、

●パターンと合致しないものを処理する
●1通のメッセージに対して複数の処理をする
●ボディ（ヘッダではなく本文！）に含まれるパターンと合致するとき処理をする

など、いろいろなことができるので、興味のあるかたは“man procmailrc”を読んで研究してみてください。

さて、procmailをテストしましょう。自分のLinuxマシンでsendmailが動いていれば、実際にテストメールを受信してみて害のないフォルダに振り分けられるかどうか調べるのが一番です。シェルから、

```
% ps -aux | grep sendmail
```

としたときになにか表示されれば、sendmailが動いているので、自分の~/.forwardに、

```
| procmail
```

と書いておいてください。こうすると、自分のLinuxマシンのMTA（sendmail）が受信した自分あてのメールは、procmailくんが振り分けてくれます。先月号で紹介したimsetupを使った設定では、Mewを使ってメールを書くと、ローカル（自分の）マシンのMTAに渡さずにプロバイダのMTAに直接渡してしまってテストになりません。そこで、ここは原始的にシェルから、

```
% mail taroh
Subject: testa  ……振り分け条件に設定したSubject:
this is a test.
^D              ……行頭でCtrl-Dをタイプ
```

のようにメールを送ってみてください。~/Mail/testa/の中に、なにかファイルができていれば成功です。

```
% ls ~/Mail/testa/
/home/taroh/Mail/testa:
1
%
```

“1”というファイル名は、自動的にやってきたメールにつけられた番号です。このあと同じテストを繰り返すと、ファイルが2、3、……のように増えていくはずです。

sendmailがローカルのマシンで動いていない場合は、仕方がないのでprocmailに直接メールを渡してテストしましょう。まず、通常の手順で自分あてに“Subject: testa”というメールを出します。受信したら、このメールの+inboxでの番号を覚えておきます。たとえば+inbox/1234は、~/Mail/1234というファイルと対応しているので、

```
% cat ~/Mail/inbox/1234 | procmail
% ls ~/Mail/testa/
/home/taroh/Mail/testa:
1
%
```

としてみて、これまでなかったはずの+testa/1というメッセージにコピーされていればテストは成功です。どちらの場合でも、同様に“Subject: testb”というメールもテストしてみましょう。さらに+testb/1というメッセージができていることを確認してください。

上記のテストが終わったら、

- **~/.procmailrcの中のテスト用レシピ**
- **~/.forward**
- **~/Mail/testa/、~/Mail/testb/ディレクトリと中のファイル**

は、念のため消しておいてくださいね。

fetchmailの導入と設定

POP3またはIMAPによってメールを取りに行くのは、fetchmailというこれまた外部のツールに任せます。fetchmailもRPMから導入できます。

fetchmailの設定は、~/.fetchmailrcというファイルにします。RPMにはサンプルがついてきますが、いつも1カ所のプロバイダに取りに行くだけだったら、**リスト4**の2行だけ書いておけばOKです。

パーソナルなマシンではおおむね問題ないと思いますが、上記ファイルにはプロバイダに接続するときのパスワードが生で書いてありますので、PPPの設定ファイルなどと同様に安全のため、次のようにしておきましょう。

```
% chmod 0600 ~/.fetchmailrc
```

fetchmailの設定が終わったらPPPでダイヤルアップ接続し、シェルからfetchmailを実行してみます。成功すれば、さまざまなメッセージとともにメールが取り込まれていくはずです。これまでの方法でprocmailがうまく設定できていれば、振り分け先のフォルダにはいつのまにかメッセージが増えているはずです。

このツールはEmacsの起動とは関係ないので、PPPツールなどから、プロバイダに接続したときに“fetchmail -d”でデーモンとして起動し、プロバイダから切断するときに“fetchmail -q”で終了すると便利でしょう。

なお、POP3ではなくIMAP4を使ったり、読み出したメールをサーバから消さずに残しておくなど、細かい設定は“man fetchmail”してみてください。

新着メッセージを追いかけよう

自動的に振り分けられたメッセージは「いつのまにか増えている」と書きましたが、実はこれは偽らざる感想です。

▼リスト4　.fetchmailrcの設定

```
poll mail.taroh.org with protocol POP3:
    username sasaki.taroh, password hogehoge, mda "procmail -d taroh";
```

プロバイダのPOPサーバ名 / POPサーバでのアカウント名 / POPサーバでのパスワード / ローカルマシンでのアカウント名

一見、便利そうな機能であっても、メッセージの見落としにつながるとすれば、使うのをためらう人もいるでしょう（達人でも自動振り分けをしない人がいる、と書いたのはこのような理由からです）。

prom-mewというマクロは、procmailのログ（動作記録）を解析して、「新しいメッセージが、フォルダ○と×にあるよ」と教えてくれるものです。

これまたRPMから導入したことを前提に、テストをスタートしてみましょう。導入時に注意しなければならないのは、prom-mewはMewのEmacs Lisp変数を活用しているため、Mewのバージョンがちょっとでも違うと動きません。また残念なことに、prom-mewはEmacs19に対応したMuleに対してのみ開発されているようです。したがって、Emacs20またはXEmacs21を利用している場合は、それに対応するMewがあるからといってprom-mewを使えるわけではありません。

このような環境で、どうしても自動振り分けがないと嫌いやイヤという人は、原始的な振り分け方法に頼るか、次回紹介予定のWanderlustを使うしかありません。

さて、procmailのテストの結果として、**リスト5**のようなログファイルが ~/Mail/from-log に残っているはずです。ここでおもむろにM-x mewで起動するのではなく、**M-x prom-mew**として起動してみましょう。表示は画面5のようになるはずです。

1行目の+inboxにカーソルを移して〈SPC〉キーを押すと、prom-mewが勝手にMewを呼び出して+inboxフォルダが開き、新着メッセージの1番目にカーソルが位置づけられて内容が表示されます。

あとは、〈SPC〉キーで次々と読み進めることができます。また、返事を書いたり転送したり……という操作も、Mew同様に（いや、ここでの操作は完璧にMewの中なので）できます。

+inboxフォルダの最後までメッセージを読み進むと、「これ以上のメッセージはない」と言われて次に“（Type SPC to +mailnews）”という表示が出てきますね。そう、

画面5　prom-mew起動直後のようす

ここで〈SPC〉キーを押すと、次のフォルダに自動的に移れます。最後のフォルダの最後のメッセージを読み終わると、最初のProm-Mewというバッファに戻ってきます。ここで〈q〉をタイプすると、prom-mewから脱出できます。

どうです、いやに簡単ではありませんか。あまり簡単なのでポンポン読み飛ばして、つい相手に礼を欠いたり（返事を出さなかったり）しないようにご用心。

さて次回は……

メール話が続いて恐縮ですが、次回はEmacs上で動くもうひとつのメールツール、Wanderlustを取り上げましょう。なんか、実用ということになると、どうしてもメーラの説明には力が入ってしまいますね……。これにかぎらず、この連載で取り上げてほしいEmacs上のアプリケーションなども、taroh@taroh.orgまでリクエストください。

WanderlustはIMAP4メールサーバとお話しすることもできるし、ネットニュースの読み書きもできます。またメールの保存方式が同じなので、Mewの環境をそのまま移行したり、Mewと併用することもできるため、先月、今月と学んだことはけっして無駄にはならないはずです。

引き続いてHappy Hacking!!

▼リスト5　procmailのログ

```
>From taroh  Sun May 14 17:38:17 2000
 Subject: testa
  Folder: testa/1                                    1234

>From taroh  Sun May 14 17:41:22 2000
 Subject: testb
  Folder: testb/1                                    1236
```

Linux日記

第10回　名前解決（3）

名前解決の話ももう3回目。今回はちょっと特殊な名前解決の説明に続き、いよいよUNIX標準のDNSサーバ、BINDの解説に入ります。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

さて、わが家のOCN接続もほぼ一段落した。最初は、開通から1週間もあればどうにかなるだろうと思っていたのであるが、風邪で寝込んだり、そのおかげでほかの仕事が切羽詰まったりということで、なかなか時間が取れず、結局、本運用開始まで2カ月近くかかってしまった。まぁ、運用といっても、namedとsendmailとnatd（編注：Linuxでいうところのipマスカレード）を動かし、ルータのフィルタ設定を行った程度なのだが。

今度は新しいメールアドレスをまわりに周知させなければならない。メーリングリストのアドレス登録を変更したり、仕事や遊びで個人的にメールをやり取りする相手にも通知しなければならない（やっと新しい名刺ができたところだ）。というわけで、古いUUCP接続のメール環境は、まだ当分稼働させておかなければならない。まじめにやるのであれば、OCN側のSMTP接続のsendmailにUUCP用の設定も統合することになるのだが、おいおいUUCPを止めるつもりなので、今のところは別々のメールサーバ、別々のメールクライアントを動かし、2つのメール環境を使い分けている。ネットワーク型ダイヤルアップの契約は解約した。とはいっても、解約までの時間があるので、もうしばらく基本料金だけは払い続けなければならない。

さて、やっとインターネット接続環境の移行と、次の仕事のための環境のセットアップが終わったというところで、またもや試練が降りかかってきた。仕事部屋の模様替えである。筆者は、今の家に引っ越してきて8年ほどになるが、その間に、いわゆる大掃除というのは数回しかやっていない。最初の頃は物が少なかったので、単に搬入して並べるだけでよかったのだが、その後、それでは済まなくなった。徐々に物が増え、新たに大型機材を搬入するためには、設置場所をまず確保し、そして搬入のための通路を整備しなければならなくなったのである。そして、これを大々的にやる時に、ついでに大掃除をするようになった。最後にこれをやったのは、ビデオ編集機材を搬入した時である。部屋に高さ180cmのEIAラックを搬入するために大掃除をした。このラックは、中身が空の状態で約50kg、搬入時の状態でも30kg以上ある代物である。そこで、移動のためにキャスターを付けた。しかしキャスターがあると、その高さの分、ドアを通らなくなる。結局、寝かせた状態で部屋に搬入し、室内で立てて移動することになった。そんなこんなの作業をするために、大掃除になってしまったのである。

今回の大掃除の直接のきっかけは、ディスプレイの追加導入である。去年から今年にかけて、マシン台数がかなり増えたので、ディスプレイが足りなくなった。そのため、新しいディスプレイを購入することにしたのである。せっかくなので、作業性のよい21インチクラスのもの、また、平面ブラウン管というのを使ってみたかったので、SONYのG500というのを選んだ。しか

し新しいディスプレイを置く場所がない。そこで、現在のテーブルベースの環境（家具屋で見つけた立食パーティ用テーブルというのをずっと使っていた）を棚と長テーブルベースにすることにした。そこで倉庫などで使う幅180cm、高さ180cm、奥行60cmの物品棚と、多少おしゃれな幅120cm、高さ180cm、奥行45cmというスチール棚を新たに購入した。いままで物でいっぱいだった部屋にとても入るとは思えなかったのだが、図面をひいてみると、理論上は置けることがわかった。実際に入れてみると収まるから不思議だ。これらの棚を組み立てるためには、室内の物をいったん外に出さなければならない。当然大掃除である。

掃除の間、OCN環境のネームサーバ／メールサーバのダウンタイムを極力短くしなければならない。そのため、適当な場所に仮設置し、その状態で運用した。とてもドメインを代表するサーバとは思えないような状況で運転していた。

機材の搬入が一段落し、とりあえずコンピュータ類をまず稼働させた。パソコンとはいえ、10台あまりもあり、それらがネットワークでつながっているとなると、結構な量のケーブル類を接続しなければならない。だが、それよりすごいのがビデオ編集機材まわりのケーブルだった。コンピュータ用のケーブルや小物が引っ越し用段ボール2箱、ビデオ用ケーブルもそれ以上あったのだ。

模様替えや大掃除の楽しみは、とんでもないガラクタ（我楽多?!）を発見できることだ。今回は部屋と物置にあったすべての箱を確認したので、結構いろいろ発見された。まず、シャープのX68000が掘り出された。本体、ディスプレイ、外付けハードディスク、イメージユニット、サイバースティックとアスキースティック、各種ゲームソフト、開発環境、完全なドキュメントまで揃ったフルシステムである。つい懐かしくなってしまい、組み立ててしまった。難点はLANカードがないことである。そのうちフリーソフトを拾ってきて、適当にシリアルでつなげようと思っている。

もうひとつ出てきたのが、紙テープリワインダである（**写真1**）。これは、紙テープリーダで読み込み、とっ散らかった紙テープをロール状に巻き直すための道具である。紙テープリーダから吐き出されたテープは、カゴの中にたまる。読み終わったテープをこの機械のリールに巻きつけるのだ。ハンドルを回すと、このリールがぐるぐる回ってテープを巻き取り、紙テープのロールができあがるという非常に便利な道具である（現在では使うことはないが）。ほかにも、ASR-33テレタイプ用の新品インクリボン、昔自作したZ-80Aシステム、HITAC-10用のライブラリの紙テープなども発掘された。

写真1　紙テープとリワインダ
付録CD-ROM No.4にカラー画像データを収録。

■少し特殊な名前解決

前置きはこれぐらいにして、ネームサーバの話の続きを始めよう。前回は、ドメイン名からIPアドレスを求めるもっとも基本的な仕組みを紹介した。今回はその続きとして、少し特殊な名前

Column

ラック

ここでいっているラックは、世間でいうオーディオラックのようなものではない。産業用／業務用のものである。電話局とか放送局にずらっと並んでいるような奴だ。家庭で使うにはかなり高価で重いものだが（キャスターで移動すると床がミシミシいう。機器を装着した状態だと、たぶん200kg弱くらいだろう）、多数の機材を収容する際のスペースファクターがありがたい。また収納機器はスライドレールマウントできる（あるいはレール付きの棚板に置ける）ので、メンテナンス時や配線時には手前に引き出すことができ、とても便利だ。

X68000

日本のパソコンがNECのPC-9801一色だった1980年代後半に、シャープが発売した意欲的なパソコンである（シャープではパーソナルワークステーションと呼んでいた）。CPUは68000で、MS-DOSと同等の機能を持つ独自のDOSが動作した。また、簡単なウィンドウシステムも付いていた。だがこのマシンの最大の特徴は、（当時としては）ゲーム機能が充実していたことである（なんといっても、グラディウスというゲームがシステムに標準で付属してたくらいだ）。

発掘後、動作確認と称して当時のゲームをいくつか動かしてみた。今見ると妙にスローペースな感じがする（だからといってステージをクリアできるわけではない）。マシンの動作チェックをしているところを子供に見つかってしまい、ゲームをやらせろとうるさいのでちょっと困っている。

解決の話から始める。

アスキーから離れてちょっと違うところを見てみる。膨大なトラフィックが集中するサーバのIPアドレスを調べてみよう。

試しに、日頃お世話になっている検索サービスのIPアドレスを調べてみた。その結果が**リスト1**である。Addresses:フィールドに、複数のIPアドレスが現れている。トラフィックが集中するサイトでは、複数のWebサーバを用意し、負荷分散を行う必要がある。Yahoo!では、6台のWebサーバを用意しているようだ。

ここでDNSの重要な点に気付いてほしい。インターネットドメイン名とIPアドレスは1対1対応とは限らないのである。ある名前を指定すれば、常に特定の1台のホスト（アドレス）が選ばれるわけではないということだ。組織や国を識別するドメイン名に対して複数のネームサーバ情報が返されるのと同じように、あるサービスを提供するホストを識別するドメイン名に対して、複数のIPアドレス（すなわち複数のホストコンピュータ）が返されることがある。これは、負荷分散という面においては有利なことだ。1つの名前で複数のホストを指定できることにより、簡単に負荷分散を図れる（リゾルバは、複数の情報が返された場合、通信コンディションが同等であれば相手をランダムに選択するということを思い出そう）。複数の電話回線に同じ電話番号を振る代表番号と同じような仕組みである（各回線がIPアドレス、代表電話番号がドメイン名に相当する）。

▼リスト1　Yahoo! JapanのWebサーバをnslookupで調べる

```
% nslookup                              引数を指定しないと対話モードになる
> www.yahoo.co.jp.                     Webサーバのアドレスを求める
Name:    www.yahoo.co.jp
Addresses:  210.140.200.60, 210.140.200.61, 210.140.200.62
          210.140.200.9, 210.140.200.15, 210.140.200.16
```

ドメイン名	IPアドレス	その実体
www	192.1.1.1	ホストA
www	192.1.1.2	ホストB
www	192.1.1.3	ホストC
www	192.1.1.4	ホストD
www1	192.1.1.1	ホストA
www2	192.1.1.2	ホストB
www3	192.1.1.3	ホストC
www4	192.1.1.4	ホストD

表1　複数の名前とIPアドレスの対応付けの例

もっとも、これだけでは十分ではない。組織内でマシンの管理や保守を行う際に、ホストを識別するドメイン名を指定して特定のホストに接続できないのでは話にならない。そのため、「代表番号」ドメイン名とは別に、きちんと1対1対応するドメイン名も用意する必要がある。たとえばWebサーバが4台あるのであれば、それら4台を示すwwwという名前とは別に、各ホストにwww1、www2、www3、www4というドメイン名を割り当てることができる（これは、前述した別名とは違うものだ）。もちろん、これらの名前にはIPアドレスを1つずつ定義する。このような名前を指定することにより、www1につなぎたい時にwww2につながってしまうということはなくなる。これをホストの側から見ると、同じ名前に複数のホストを割り当てるのと対になる処理であることがわかる。www1という1台のホストは、www、www1という複数のドメイン名を持つのである。先の代表電話番号に対して、1回線に複数の番号を割り当てるダイヤルインサービスのようなことも行えるのである（**表1**）。

実際の運用では、まず、各マシンを一意に識別するドメイン名（www1など）を定め、それに1つずつIPアドレスを割り当てる。そしてサービスの形態に応じて、複数の異なるマシンに同じドメイン名（www）を割り当てるという形になるだろう。また、ルータやファイアウォールとして設定し、ネットワークインターフェイスを複数持つ（複数のIPアドレスを持つ）マシンの場合は、各ネットワークインターフェイスごとに別々の名前を割り当てるといったこともできる。

DNSでは、1つのIPアドレスに複数のドメイン名を割り当てることと、1つのドメイン名に複数のIPアドレスを割り当てることのどちらも完全に合法である。整合性の取れていないデータベースのように見えるかもしれないが、名前解決とサービスの提供という両面で見た場合、これは合理的な構造なのである。

■BIND

インターネットドメイン名からIPアドレスを求める際に、ルートサーバから始めて多数のサーバに順に問い合わせを行い、最終的に、目的のIPアドレスを知っているネームサーバにたどり着くという仕組みはだいたいわかっただろうか。

そろそろUNIXで使われているネームサーバの解説に入ろう。UNIXで使われている標準的なインターネットドメイン名のネームサービスは、BIND（Berkeley Internet Name Domain）というパッケージで提供される。DNS（ドメインネームシステム）は、インターネットで使われている名前解決システムの名称である。そして、BINDは、DNSをUNIXプラットフォームに実装したパッケージのひとつである。しか

しBINDは、インターネットの中核をなしているUNIXでもっとも広く使われてきたシステムなので、結局のところ、もっとも標準的なDNSの実装なのである。Linux、FreeBSDもこのパッケージを使っている。また、Windows用のDNSパッケージにも、これをベースにしているものがある（Windows用のBINDもある）。

当然、今回からの記事もBINDについて説明する。BINDには、大きく分けて4.Xと8.Xという2種類のバージョンがある。4.Xに比べて8.Xは、セキュリティ対策などが充実している（4.Xでも、最新のバージョンはかなりセキュリティが強化されている）。またBIND 8.Xは、DHCPを使った動的なIPアドレス割り当てにも対応しているので、現在の環境で使うなら8.Xを選ぶことになるだろう。この記事も、8.Xについて解説する。

■named

BINDをインストールし（最近のLinuxパッケージでは、いわゆる「サーバモード」のインストールを行えば、BINDがインストールされるようだ）、適当な初期化スクリプトを記述すると、システム上でnamedというデーモンが稼動する。これがネームサーバのプロセスである。ネームサーバはバックグラウンドでデーモンとして動作するので、管理するための対話は、基本的にシグナルで行う。適当なシグナルをnamedに送ることで、サーバの停止、データの再ロード、現在の状態をファイルにダンプ（出力）するといったことができる。よく使うシグナルを**表2**に示しておく。

ネームサーバが出力したファイルを調べることで、ネームサーバが正しく動作しているかといったことを調べることができる。また、スレーブサーバ（後述）の場合、リブート時にマスタサーバと通信できなくても、停止前に出力したデータベースファイルをロードすることで、（データは最新のものではないかもしれないが）サービスを続けることができる。

ネームサーバに登録するデータベースは、いくつかのテキストファイルの形で記述することになる。また、ネームサーバの動作モードなどを指定するパラメータも、テキスト形式の設定ファイルに記述する。namedは起動時に（あるいは再ロードを指示された時に）これらのファイルを読み込み、動作を始める。

ネームサーバにはいくつかの動作モードがあり、モードごとに動作が変ってくる。まず、動作モードについて簡単に説明しよう。

■マスタサーバとスレーブサーバ

マスタサーバとは、あるゾーンを管轄するネームサーバで、そのゾーンデータベースをローカルディスク上に持っているものである。前に述べたように、分散データベースの一部を担っているネームサーバは、（複数レベルのドメインも含めて）ドメイン名空間の一定の部分について、責任を持って名前解決（IPアドレスを返す、ネームサーバを紹介するなど）を行う。この責任範囲のことを、そのネームサーバのゾーンという。このゾーン中の正式なDNS情報は、データベースファイルという形で、そのネームサーバの管理者が手作業で記述しなければならない。管理者が保守するデータベースファイルを保持しているネームサーバがマスタサーバである（プライマリサーバとも呼ばれる）。

SIGHUP	設定ファイル、データベースを再ロードする。設定ファイルなどを変更した場合、その内容は再ロード後に有効になる。
SIGINT	現在のデータベースの内容、キャッシュの内容をファイルに出力する。
SIGTERM	データベースファイルを出力し、終了する。

表2　namedが受理するシグナル

Column

ドメイン名とホスト名

ドメイン名（ネームサーバに登録されている名前）とIPアドレスは1対1対応とは限らないわけだが、UNIXマシンに設定するホスト名（hostnameコマンドで設定、参照される名前）は、各ホストに1つずつである（複数のネットワークインターフェイスを持っていても、ホスト名は1つである）。通常の使用形態であれば、このホスト名とIPアドレスを1対1で対応付けるドメイン名をネームサーバに登録することになる。これにより、そのホストに対して、ホスト名を使ってネットワーク接続することが可能になる。しかし、ドメイン名はホスト名と同じでなければならないわけではない。クライアントはネームサーバにドメイン名を問い合わせ、IPアドレスを受け取る。そして、そのIPアドレスを使って目的のホストと接続する。名前解決と接続の段階では、相手のマシン上で定義されているホスト名はまったく関与しないのである。もっとも、ドメイン名とホスト名が食い違っていると、接続後に問題が生じることがあるので、組織内で使用する場合は、ホスト名とドメイン名を同じにしておくというのが基本である。そして、何らかの理由で別のドメイン名も割り当てたいのであれば、改めてその名前を登録すればよい。

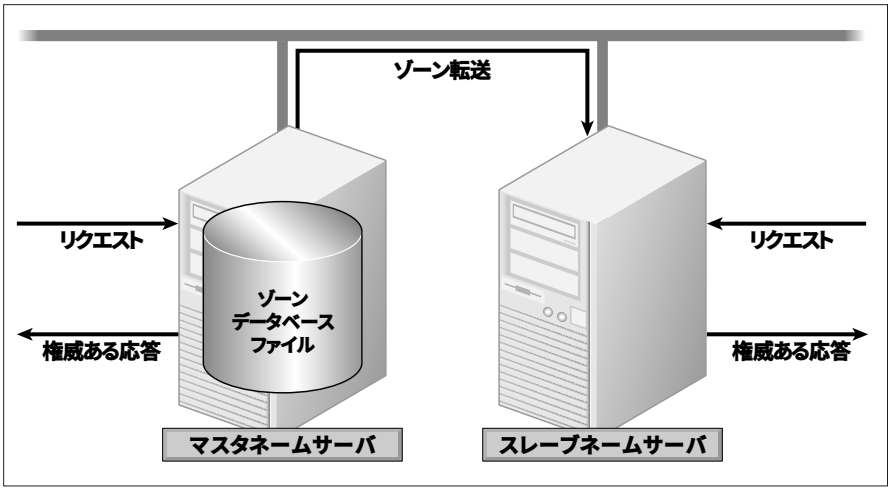


図1　マスタサーバとスレーブサーバ
ゾーンデータベースファイルはシステム管理者がメンテナンスする。

そのゾーンで何らかの情報の変更が必要な場合は、マスタサーバ上のデータベースを更新する。これにより、このサーバに問い合わせを行う別のネームサーバやクライアントは、新しい情報を得られるようになる（DNS情報はキャッシュされるので、更新が反映されるには、多少の時間がかかる）。

スレーブサーバは、マスタサーバからネットワーク経由でゾーン情報をコピーし、その内容をほかのネームサーバやクライアントに提供するネームサーバである。情報はマスタサーバからもらうので、管理者が手作業で管理するデータベースは持っていない（再起動にプリロードできるキャッシュファイルは指定できる）。スレーブサーバは、セカンダリサーバとも呼ばれる。1つのゾーンについて、マスタサーバは1台だが、スレーブサーバは（0台も含めて）任意の数だけ設置できる。スレーブサーバが存在することにより負荷分散が図れ、また、システムダウン時のバックアップとなる（**図1**）。

マスタサーバからスレーブサーバにデータベースを転送する処理を、ゾーン転送という。マスタサーバとスレーブサーバは定期的にゾーン転送を行い、データの同期を行う。ゾーン転送を行う頻度は、マスタサーバの持つデータベース中に記述する。どれだけの頻度で行うかを決めるのは管理者である。頻度が高ければ、更新が速やかに伝播されるが、ゾーン転送のトラフィックが増えることになる。大規模なネームサーバでは、莫大な量の情報を転送しなければならないので、そんなに頻繁に行うことはできない。しかし、間隔が長すぎると、いつまでも新しい情報がスレーブサーバに送られず、マスタサーバとスレーブサーバの情報が食い違っている時間が長くなってしまう（システム管理者はデータ更新時に、明示的にゾーン転送を行うように指示することもできる）。

マスタサーバとスレーブサーバは、どちらも、そのゾーンについて権威のあるサーバとなる。スレーブサーバの場合、ゾーン転送遅れにより最新の情報を返せないことがあるが、それでも権威のあるサーバなのだ。クライアント側は、これらの情報を正しいものとして受け取る。システム管理者は、情報更新の伝播遅れまで考慮して、ネットワーク情報の更新を行わなければならない。つまり、データベースを更新してすぐにマシンを撤去したりすると、トラブルが発生する可能性があるということだ。賢明な管理者は、新しい情報が伝播されるまで待ってから撤去するだろう。

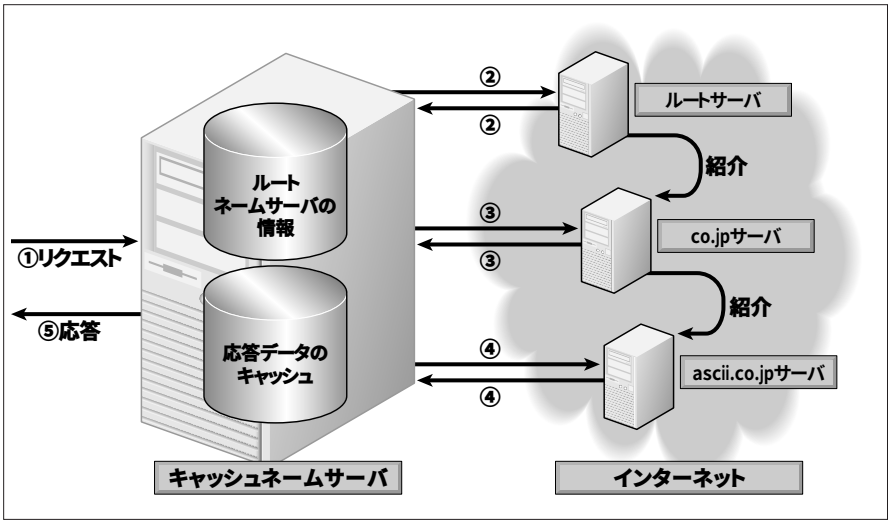


図2　キャッシュサーバ

■キャッシュサーバ

あるネームサーバがクライアントから名前解決リクエストを受けた時、そのネームサーバが検索対象ドメインについてマスタサーバかスレーブサーバであれば、権威のある情報を返すことができる。そうでなかった場合は、前に解説したような手順で目的のドメイ

ン名を検索し、別のネームサーバとやり取りし、最終的な結果を得て、それをクライアントに返すことになる。

この時ネームサーバは、ほかのネームサーバから受け取った応答をキャッシュし、以後同じリクエストを受け取った時は、キャッシュされたデータをクライアントに返すことができる。また、ネームサーバをマスタサーバとしてもスレーブサーバとしても設定しなかった場合、そのネームサーバは権威のある応答をクライアントや別のネームサーバに返すことはできないが、ほかのネームサーバに問い合わせを送り、名前解決を行うことはできる。未知の名前を解決し、その結果をキャッシュするという動作モードを、キャッシュサーバという（**図2**）。

マスタでもスレーブでもないキャッシュサーバは、負荷分散のために使用される。またマスタサーバやスレーブサーバも、自身のゾーン以外のドメイン名については、キャッシュサーバとしてクライアントにサービスを提供することになる。

■フォワーダの指定

キャッシュサーバは、名前解決を行うために、ルートサーバから始めて順に多数のネームサーバに問い合わせを行う。つまり、あるリクエストを解決するために、ほかのネームサーバに複数回の問い合わせを行うのである。これに対して、このような複数回の問い合わせを行わないという動作モードもある。これがフォワーダの指定である。（**図3**）

この動作モードは、クライアントからリクエストを受けると、自身では何も解決を行わず、その名前解決リクエストを別のネームサーバ（フォワーダとして指定したサーバ）に丸投げするのである。それを受け取った別のネームサーバが複数回の問い合わせを行い、名前を解決し、その結果を元のネームサーバに返す。そしてクライアントは、最初にリクエストを送ったネームサーバから応答を受け取るのである。もちろん、この応答はネームサーバ上でキャッシュされ、以後の同じリクエストに備える。

別に、フォワーダなど指定しなくても、ただのキャッシュサーバで十分だろうと思う読者もいるだろう。しかし、サイトとプロバイダが低速回線で接続されている時などは、フォワーダを指定すると有利なのである。たとえば、プロバイダとサイトが64kbps回線で繋がれているとしよう。そのサイト用のネームサーバはプロバイダ側にあるとする。

図4を見てほしい。サイト側にネームサーバがまったくない場合（Aのパターン）、クライアントからの名前解決リクエストはすべて、64kbps回線を経由してプロバイダ側に送られる。クライアントは基本的に応答をキャッシュしないので、非常に多数のリクエストと応答が64kbpsの回線を通ることになる。サイト側にキャッシュサーバを用意すれば（Bのパターン）、クライアント―ネームサーバ間のトラフィックはサイト内のLANだけで済み、また、ネームサーバがデータをキャッシュする

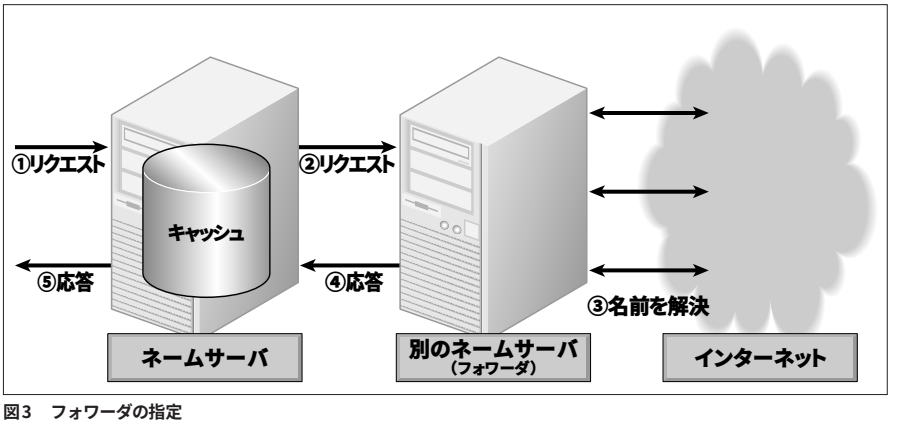


図3　フォワーダの指定

Column

再帰的な問い合わせと非再帰的な問い合わせ

ネームサーバの問い合わせには、再帰的なものと非再帰的なものがある。再帰的な問い合わせとは、それを受け取ったネームサーバがさらに複数回の問い合わせを行い、最終結果を返すというモードである。クライアントがネームサーバに送る問い合わせは、再帰的な問い合わせである。

非再帰的な問い合わせを行うと、ネームサーバは再帰的な問い合わせを行わず、自身が持つデータだけを返す。末端のネームサーバであれば、本来の対象であるIPアドレスなどを返すことができるが、ルートサーバや組織名を解決する中間のネームサーバは、より詳しく知っているネームサーバを紹介する情報を返すことになる。

クライアントから名前解決リクエスト（これは再帰的な問い合わせである）を受けたネームサーバ（キャッシュサーバ）は、ほかのネームサーバに非再帰的な問い合わせを何回か送り、名前を解決する。フォワーダ指定を行ったネームサーバは、クライアントから問い合わせを受けると、その内容を「再帰的な問い合わせ」として別のネームサーバに送るのである。

ので、クライアントに対する応答性が向上するだろう。しかしキャッシュサーバは、1回の名前解決を行うために、多数のネームサーバとやり取りを行う。この多数のやり取りは相変わらず64kbps回線を経由して行われる。このキャッシュサーバの動作モードを変更してフォワーダを指定し、問い合わせをすべてプロバイダのネームサーバで解決してもらうようにするとどうなるか（Cのパターン）。多数の問い合わせを行い、名前解決を行うのは、高速な回線に繋がっているプロバイダのネームサーバになる。64kbpsの回線を通るのは、サイト内のネームサーバからプロバイダのネームサーバに送られるリクエストが1つ、その応答が1つだけになる。DNSのトラフィックが減った分、その帯域をほかのサービスに回すことが可能になるのだ。

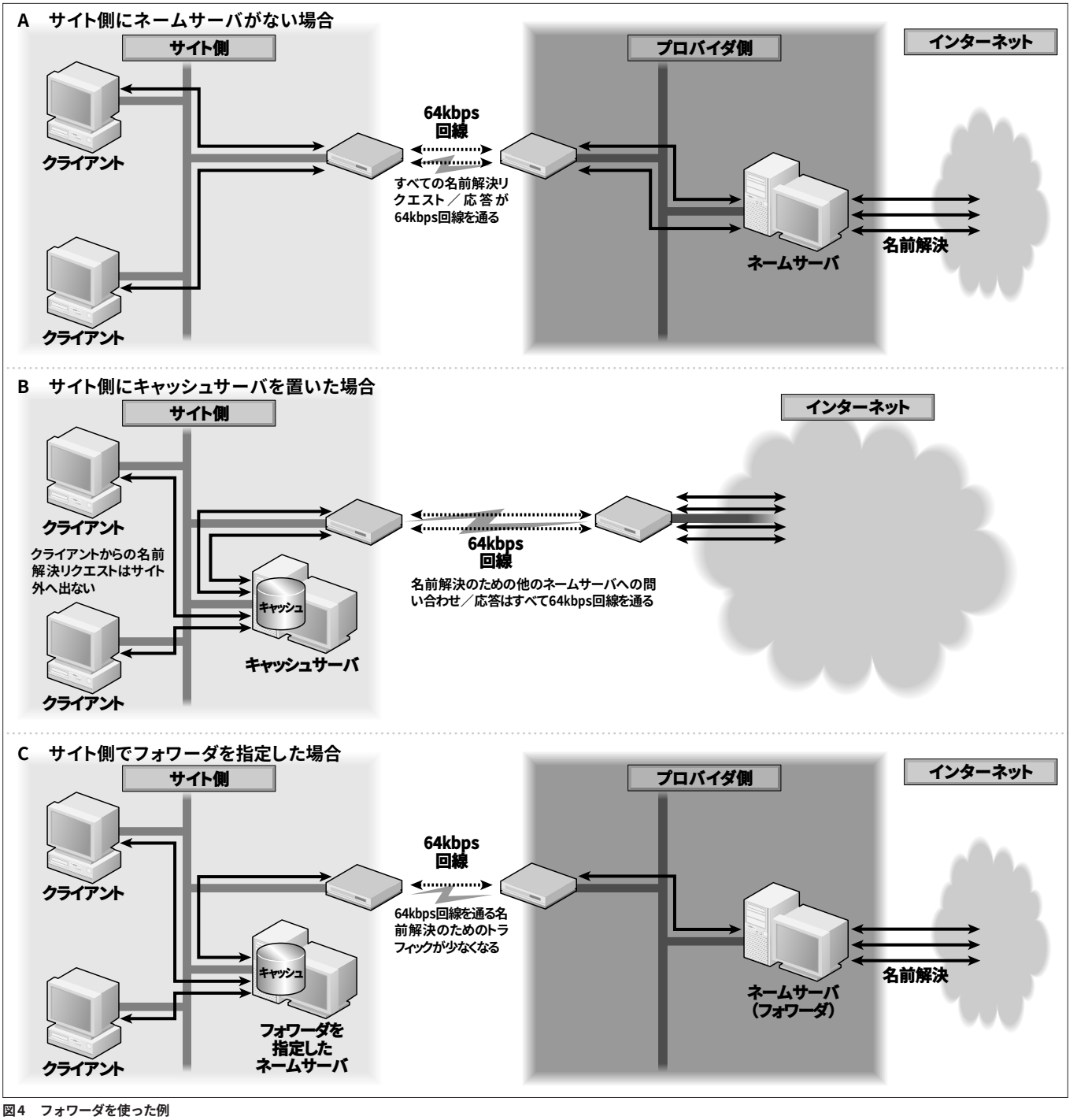


図4　フォワーダを使った例

Linux ガベコレ

Linux Garbage Collection

目からウロコの用語辞典

文: しのはらひろあき
Text : Hiroaki Shinohara

第3回
【スクリプト言語】（すくりぷとげんご）
【AWK】（おーく／えーだぶりゅけー）
【Perl】（ぱーる）
【シェルスクリプト】（しぇるすくりぷと）
【Ruby】（るびー）
【Tcl】（てぃくる）

スクリプト言語

【すくりぷとげんご】


スクリプトられてる人

C言語習得時、ポインタどころか文字列操作で挫折した編者のようなへっぽこプログラマーを救うもの。特定の作業を簡単な記述で実現できるように作られたコンピュータ言語のこと。非生産的だったプログラミング現場に登場したスクリプト言語のおかげで、多くのプログラマーがメモリリークの発生を心配をすることなくバグの生産に熱中できるようになった。

スクリプト言語の多くは、作業の手順を記した“スクリプトファイル”を逐次読み込みして動作するインタプリタ型言語となっており、これがその名の由来ともなっている。“スクリプト”とは、もともと英語の「台本（script）」を意味し、プログラマーが作った台本どおりにインタプリタが動作するというイメージでこう呼ばれているというのが俗説だ。しかし、実際にスクリプト言語を使用してみればわかるとおり、これは誤りである。現実には、スクリプトを書いた人間のほうが、言語の仕様、バグ、予期しない動作に振り回され、あやつり人形のように動かされている。編者は「スクリプト言語」に代わり、より受け身性を強調した「スクリプトられ言語」という呼称を提唱したい。

AWK

【おーく／えーだぶりゅけー】

Aho、Weinberger、Kernighanの3人が作ったスクリプト言語。それぞれの名の頭文字をとって“AWK”と名付けられた。Kernighanは、不純にもCにふけったプログラマーとして有名。Weinbergerは開発当時の米国防長官。残るAhoとは、もちろん吉本興業所属のアーティスト、Aho坂田氏を指す。つまり、核戦争などの有事に備えて強力なスクリプト言語を作りたかった米国防省、メディアミックス展開を見越した吉本興業がKernighan氏に出資してAWKが作られたのだ。開発は困難を極め、納期が翌朝に迫った夜、ストレスに耐えかねた坂田氏が「アホちゃいまんねん。パーでんねん」と語ったという話は、今もプログラマー達の語り草だ。

──で「Hello, World!」を表示する：

```
#!/bin/awk -f
BEGIN {
    h = "Hello, World!";
}
{
    if ( NR > length(h) ) {
        exit;
    }
    printf substr(h, NR, 1);
}
END {
        printf "\n";
}
```

※helloとして作成後、./hello hello として実行する

Perl
【ぱーる】

インターネットで掲示板を運営したり、アクセスカウンタを作ったりするのに利用されるスクリプト言語。非常に豊富な機能をもつため、多機能な掲示板を実現したり、多機能なアクセスカウンタを作成することもできる。データベースへのアクセスも容易で、活用すればバックエンドにリレーショナルデータベースを置いた掲示板や、よりスケーラブルなアクセスカウンタを動かすことも夢ではない。

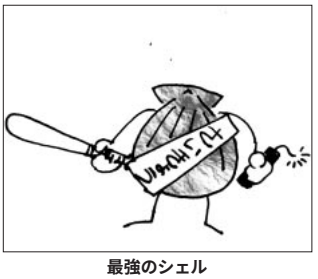
Perlの特徴として、豊富な表現力と略記バリエーションの多様さが挙げられる。このPerlの精神を表したことばが「There's More Than One Way to Go the Wrong Way」（間違えるには何通りもやりかたがある）だが、このことは同時に、Perlスクリプトの保守を難しくさせている。2000年元日、インターネット中の掲示板で「19100年1月1日」の表示が見られたのは好例といえよう。

――で「Hello, World!」を表示する：

```
#!/usr/bin/perl
$H = '48 65 6c 6c 6f 2c 20 57 6f 72 6c 64 21';
foreach ( split(/ /, $H) ) {
    system("echo -n \'" . pack("H2", $_) . "\'");
}
print "\n";
```

シェルスクリプト
【しぇるすくりぷと】

sh、csh、bashなど、Linuxのコマンドシェルはスクリプト言語として使用することができる。この際のスクリプトを“シェルスクリプト”と呼ぶ。シェル上で使用できるコマンドを列記するだけなので、習熟が楽なのが特徴。



最強のシェル

シェルの中でも、sh（/bin/sh）はスクリプト言語としての用途にもっとも適しているといわれており、なおかつ「最強のスクリプト言語」とも呼ばれている。これは、shで書かれたスクリプトが、なんでもできる柔軟さを有しているためである。shスクリプトを使えば、フィルタツールはもちろん、CGIスクリプトや高度なシステム管理、通信さえもが可能となる。たとえば、指定URLからWebページの内容を取得するshスクリプトは、実質わずか6行で記述できる。

```
#!/bin/sh
# * requires perl 5, LWP
while [ "$1" ] ;
do
    perl -e 'use LWP::Simple;
        print get($ARGV[0]) . "\n";' $1
    shift
done
```

――で「Hello, World!」を表示する：

```
#!/bin/sh
#|Hello, World!
grep '^#|' $0 | sed 's/^#|//;'
```

Ruby
【るびー】

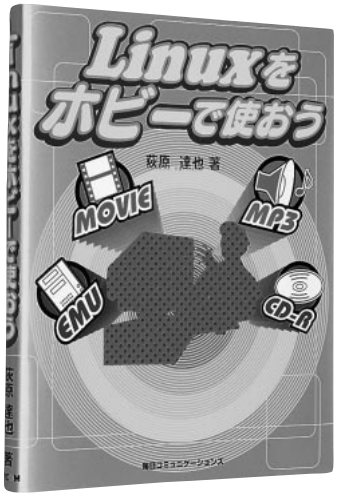
最近流行のきざしを見せはじめている純国産のスクリプト言語。その最大の特徴は「オブジェクト志向」。とにかくオブジェクト＝モノを重要視する物神崇拝志向の言語なのである。名前がPerl（pearl：真珠）より高価な「ルビー」になっている点からも、モノへの執着ぶりがうかがえる。まったく資本主義（編注：以下、Rubyへのいわれのない中傷が続くので削除）だ。私見だが、今のプログラミング業界はモノにこだわりすぎる。このあたりで精神世界への回帰を問うべきではないか。「Iyashi」や「Astral」といったスクリプト言語の登場が望まれる。

オブジェクト志向をのぞけば優秀な言語であり、「スレッド対応」により紡績業界での活躍に期待が寄せられている。――で「Hello, World!」を表示する：本誌連載「Rubyで行こう」、アスキー刊「オブジェクト指向スクリプト言語Ruby」で勉強しよう。

Tcl
【てぃくる】

どこへ行ったんだろう……。

Books



Linuxをホビーで使おう

荻原達也 著
毎日コミュニケーションズ
B5変形判／192ページ　本体価格 1800円

みなさんはLinuxをどのような目的でお使いだろうか？　よく耳にするのは、サーバとして、あるいはテキスト編集やインターネットアプリケーションの利用といったところだ。ムービー再生やMP3などはWindowsでなければできないと思っている方も多いのではなかろうか。ああ、なんともったいない。Linuxは遊びの分野でも活躍できるOSなのだ。本書は、Linuxユーザー向けの遊び方入門だ。具体的には、CD-Rを焼く、MP3を作成／再生する、VMwareで他のOSを動かす、ムービーを再生する方法を取り上げている。それぞれ、必要なソフトウェアのインストール方法と使い方を丁寧に解説しているので、Linux初級者でも安心だ。

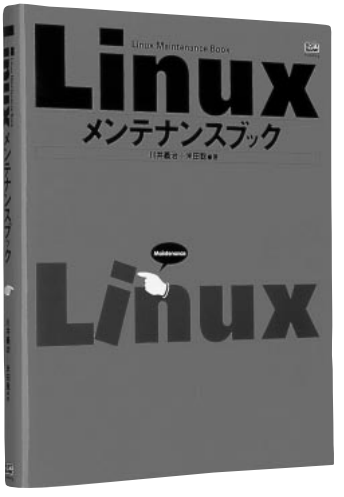
Linuxの入門書は数多くあるが、ホビー用途に主眼を置き、使い道を提案する書籍は数少ない。インストールが済み、設定方法もあらかた修得した。さて、次は何をしようかというユーザーにお勧めしたい。もっとLinuxで遊ぼうではないか。

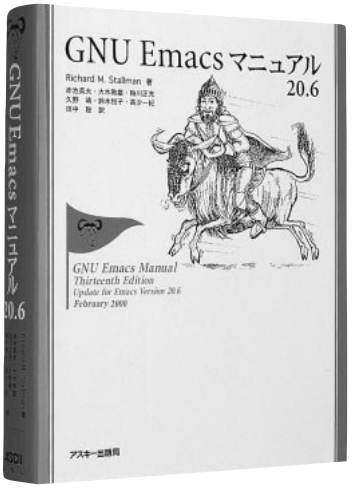
Linuxメンテナンスブック

川井義治／米田 聡 著
ソフトバンク パブリッシング
B5変形判／350ページ　本体価格 2500円

「Linuxをインストールしたら、あなたはもうシステム管理者なのだ。」という言葉をよく聞く。しかし、WindowsだろうがMacintoshだろうが、システムの管理はユーザー自身が行わなければならないのは同じだ。では、Linux（やBSD）の管理者とは何が特別なのだろうか？　最近のLinuxディストリビューションには、数々の設定ツールが付属するが、どれも帯に短したすきに長しといった具合で、ちょっと複雑な設定をしたい場合は、各種のコマンドラインツールを駆使したり、エディタで設定ファイルを書き換えることになる。ツールの使い方や設定ファイルの書式など、システム管理者が知っておくべきことは膨大な量になる。

本書は、Linuxのシステム管理を行ううえで必須の知識を凝縮した虎の巻だ。基本的なコマンドの使い方から、日常的な管理作業に必要な知識までを、平易な文章で詳しく解説している。自分のLinux Boxをキッチリとメンテナンスしたい方に広くお勧めする。



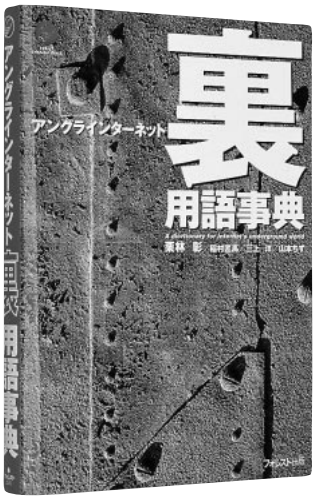


GNU Emacs マニュアル 20.6

Richard M. Stallman 著　赤池英夫／大木敦雄／粕川正充／久野 靖／鈴木悦子／高汐一紀／田中 聡 訳
アスキー
B5変形判／680ページ　本体価格 3800円

Emacsといえば、viとともにUNIX系OSの代表的なエディタであることはいうまでもない。Emacsは、その中からメールを読み書きしたり、ファイルを操作したり、プログラムをコンパイルするなど、さまざまなことができるため、単なるエディタではなく、ひとつの環境なのだという意見もある。一度Emacsを起動したら、すべての作業はそこで行うというユーザーもいるくらいだ。このように、使い方を覚えたら手放すことができないEmacsだが、初心者が利用するには、独特のキーバインドなどがハードルとなるようだ。

本書は、GNU Emacsの最新版、Ver.20.6の公式マニュアルを日本語訳したものだ。Emacsの使い方を、画面構成、起動／終了の方法、テキストの編集はもちろん、メールの読み書き、カスタマイズにいたるまで詳しく説明している。圧倒的な情報量にも関わらず、良心的な価格設定には驚かされる。Emacsを利用するなら、常備しておきたい1冊だ。



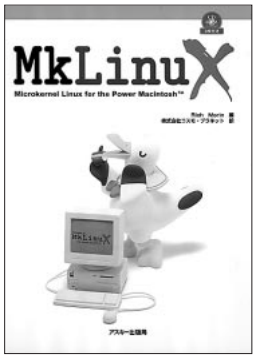
アングラインターネット　裏用語辞典

栗林 彰／稲村匠馬／三上 洋／山本ちず 著
フォレスト出版
A5判／320ページ　　本体価格1600円

インターネット利用者が増えれば、数々のコミュニティが生まれるのは必然だ。そのなかには、少々あやしげな情報がやりとりされる場もある。それがUG（アンダーグラウンド）だ。UGの世界では、日夜、隠語によるコミュニケーションが行われている。そのため、UGとは無縁の人がアングラサイトを覗いてもさっぱり意味がわからないだろう。たとえば、「厨房」という単語。これは決して台所のことではない。厨房→中坊→中学生という連想により、まるで子供のようだという罵倒語なのだ。

本書は、このようなUG用語を解説する辞典だ。辞典といっても、眉間にシワを寄せて言葉の意味を調べるようなものではない。用語解説だけでなく、正しい使い方、間違った使い方や、周辺事情なども面白おかしく書かれているので、ちょっとおバカな読み物として楽しめる。このような奇書を世に送り出した著者陣と出版社には拍手を送りたい。

MkLinux



Rich Morin 編
コスモ・プラネット 訳

アスキー

B5判／352ページ／CD-ROM 3枚付き

本体価格 3600円

Linuxで作る ネットワーク実践構築ガイド

笠野英松 著

技術評論社

B5変形判／396ページ／CD-ROM付き

本体価格 3480円

MN128-SOHOで作るホームオフィス Slotin・SL11版

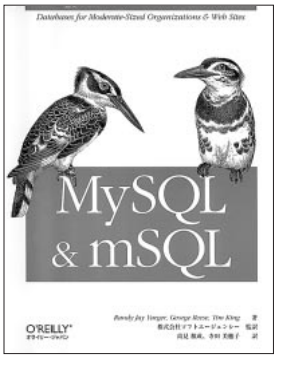
塩田紳二／北川達也／清水隆夫／諫山研一 著

インプレス

B5変形判／324ページ

本体価格 2200円

MySQL & mSQL



Randy Jay Yarger／George Reese／Tim King 著
ソフトエージェンシー 監訳
高見禎成／寺田 美穂子 訳

オライリー・ジャパン

B5変形判／534ページ
本体価格 1380円

読者の声

俺にもいわせろ！

帰宅途中の乗換駅で、偶然にもLinux Japanの風穴編集長にお会いしました。いろいろとお話をうかがい、いつもは裏寂しい終電の車中も楽しく過ごすことができました。編集部にもぜひお立ち寄りください＞風穴編集長。

6月号のアンケートハガキへのおハガキ

裏のマークシート間違ってません？

（岡山県　越智成人さん）

裏の回答欄は滅茶苦茶だ（質問と選択肢の数が合っていない）。

（新潟県　八重倉 孝さん）

同様のご意見をたくさんいただいております。

編集部のチェックミスです。ご迷惑をおかけして大変に申しわけありません。以後、このようなことがないよう、十分に注意いたしますのでご容赦ください。

6月号、特集1へのおハガキ

「Linuxマシンを作る！」は参考になりました。Intelのi810チップセットでも問題なくインストールできるようなので、予算5万円以内でもう1台組もうかなと考えています。

（東京都　平川 信さん）

3万円の中古パソコンでPC-UNIXを始めました。Pentium 133MHzで32Mバイトだけど、結構いけます。6月号の5万円自作機より安く、インストールも一発でOKでした。

（秋田県　唐牛 豊さん）

自作マシンが動く瞬間は嬉しいものですね。シリアル、パラレルポートだけではなく、最近ではグラフィックスやサウンドもオンボードになった低価格マザーボードが増えてきました。このようなマザーを選べば、あっという間にマシンを作ることができます。また、利用目的によっては、唐牛さんのように中古マシンを狙うのも堅実な選択です。

6月号、特集2へのおハガキ

6月号の「ゲーム中毒になる！」は最高です。特にglTronは、映画『TRON』を生で見たオジサンにはウレシイ限りです（TRONがディズニー映画だって知っている人は少ないんですよね）。

（東京都　綾部孝志さん）

こんなに表現力豊かなゲームがLinuxで動作するようになるとは思いませんでした。特に3Dゲームの完成度はゲーム専用機に肉薄するほどです。3Dモノを快適にプレイする場合は、それなりのマシン環境も必要ですが……。ちなみに、担当のお気に入りはTux Racerです。編集部のPentium III 733MHz＋GeForce256マシンで暴走ペンギン化しています。

6月号、ハードウェア特集へのおハガキ

「100BASEイーサネットカード＆ハブ」は大変参考になりました。ちょうど、現在使用中の10BASEイーサネットカードが、新しくインストールしたときに認識されなくて困っていたところでした。これを機に、100BASEイーサネットカードに買い換えようと思い、どのカードが良いか迷っていました。

（静岡県　森田康正さん）

NICの記事は大変参考になりました。ビンボな私にとっては“低価格”という文字が一番嬉しいです。

（静岡県　中尾典孝さん）

イーサネット機器は本当に安くなりました。メルコからも、6800円の10／100Mbpsスイッチングハブ「LSW10/100-8D」が限定1万台発売としてアナウンスされたところです。編集部内もほぼ100BASE化が済んでいます（編集長のEPSON PC-386を除く）。

女性読者急増中！＼(^o^)／てへへ

Linux magazineを毎月買い、歩くコマーシャルとしてお友達にこの雑誌を紹介し続けて約1年が過ぎました。パソコンオタッキーと呼ばれながらも、日々情報を欠かさずチェック。お友達に負けぬよう、これからも読み続けるぞー。これからも、歩く宣伝Prettyウ

ーマンとして仲間を増やしていきます。

（和歌山県　上田なおさん）

6月号付録のRed Hat Linux 6.2（英語版）、ありがとうございました。せっかくパソコンもらったのに（ソー○ックのミドルタワー）、ビデオカードが新しめなせいか、少し古いディストリビューションだと、つっかえて、ぜーーーーんぜんインストールできなかったの……。

この前、着メロのダウンローダを、小学生がキッチリいぢっているのを見た。たしか、あれはLinuxベースですよね。URLも知らない子が、インターネットを使いこなしているなんて、スッゴイ時代……。

（熊本県　佐伯ひとみさん）

ご支援、ありがとうございます＞上田さん。Linux magazineの制作・デザインチームにはゲームオタッキーの女性がいます。担当はゲームに明るくないので、いろいろと教えてもらうのですが、オタッキーと呼ばれる人って、話がとても面白い気がします。

着メロ販売機もLinuxな時代なんですね。通常のデスクトップ画面が見えないようになっているので、わかりにくいのですが、Linuxに限らず、意外な機械が身近なOSで動いていたりします。消費者金融の自動契約機がOS/2で動いていたり、コンビニの情報端末がWindowsで動いていたり……。担当が家で使っている無線LANのベース機もLinuxで動作しているのだそうです。

Linux magazineも、着実に女性読者が増えてきました。ありがたいことです。今後ともよろしくお願いします。

さらに広がる読者層

小学生にもLinux。

（長野県　飛田 空さん）

VMwareの話は聞いておりましたが、「VMware for Linux 2.0を使おう！」の内容が大変参考になりました。日本に販売代理店があって、容易に入手できればなおよいのですが……。

2年前にCaldera Open Linuxでパソコンを始めて、Windowsをまったく知らない昭和2ケタ直後生まれですが、Linuxに夢中です。カミさんのWindowsノートマシンにデジカメの画像を入れてあげられず、「それでもパソコン知ってるの？」とオコられました。

（東京都　三井 洵さん）

飛田さんは小学2年生、三井さんは60代半ばの方です。幅広い年齢の方に読んでいただいております。Linuxを楽しむのに年齢や性別は関係ないのだなあと、Linuxの奥深さを再認識しました。

PCエミュレータのVMwareはすごいソフトウェアです。LinuxのデスクトップでWindows 98が動いているのを見たときにはビックリしました。動作内容を考えれば仕方がないのですが、結構なマシンパワーを必要とするのが玉にきずでしょうか。

表紙

毎号フルーツの表紙を楽しみにしています。他の雑誌と比べてセンスもいいし、書店でもとても目立っています。ところで、前から気になっていたのですが、4月号の果物って何ですか？つぶれて変形した、少し腐ったオレンジなわけないですよね。

（兵庫県　笠井隆敏さん）

う、腐ったオレンジ……。表紙も含めたデザインを監督しているアートディレクターのスガノが泣いております。あれはマンゴーなんです。5月号からは目次ページに果物の名前を入れるようにしました。あわせてご覧ください。

月刊 遠藤さん 7月号

どーも山形の遠藤ッス。とうとう常連の仲間入りッス。今月の小話は、なんと自作機を作ってしまったッス。CPUは、Pentium 600MHzですが、サクサク動くッス。ところで、まだ本題のLinuxをインストールしていないッス。6月号は自作の特集ッス。ちょっと気になるのでしっかり読んでみるべー。なんと5万円で1台完成ッス。またもう1台私も挑戦ッス。禁煙して早1年半、吸ったつもりで妻にナイショの禁煙貯金。結構貯まるッス。自作機で使ってしまったのでまた1からッス。あっ、このビデオカード動作確認が……。あれっ？（・・;）

（山形県　遠藤浩二さん）

今月も、お便りをありがとうございます。禁煙貯金でマシン増設……。担当もやろうかしら。でも、締め切り前にタバコがないと辛いッス（^^;）。ところで、マシンが増えることに奥様は疑問を持たないのでしょうか？　担当は未だ独身ですが、マシンが増えると家族がうるさいので、先月会社に1台持ってきてしまいました。私財を投じて、仕事環境を改善するのだと主張していますが、まわりの編集部員からは会社をゴミ捨て場にするなと非難されています。バレてたか……。

来月お目にかかるころにはすっかり暑くなっていそうです。それまでは、梅雨のジトジトに耐えながら、楽しくLinuxしましょう。では。

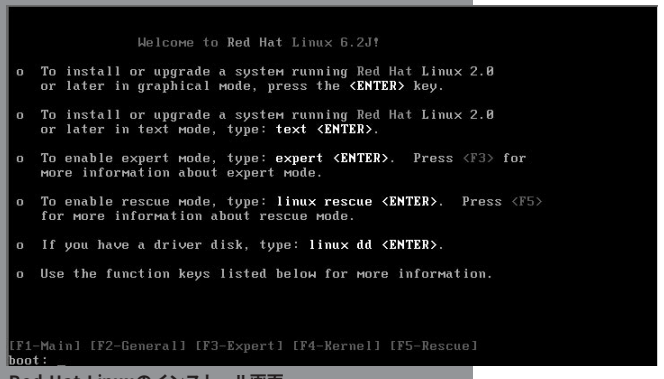
付録CD-ROMに収録した
Red Hat Linux 6.2J
Vine Linux 2.0
Kondara MNU/Linux 1.1
のインストール

本誌付録CD-ROM収録のディストリビューションはFTP版です。非商用ソフトだけが含まれています。また、製品版を販売している各社からサポートを受けることはできません。

ブートフロッピーの作成方法

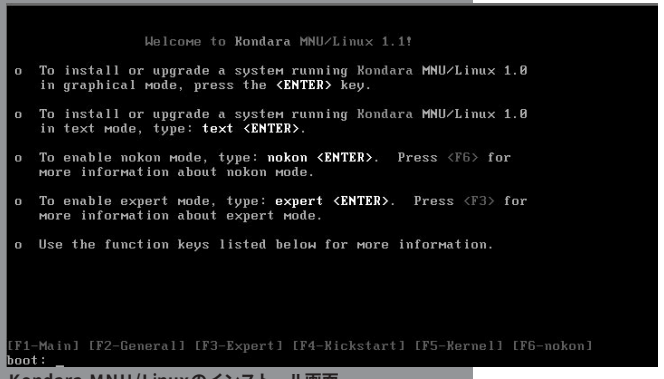
使用するPCがCD-ROMからの起動に対応していない場合は、インストーラ起動用のフロッピーディスクを作成します。Windows環境での作成手順は以下のとおりです。

（1）CD-ROMと空のフロッピーディスクを各ドライブへセット
（2）DOS窓を開きD: [Enter]と入力。CD-ROMドライブが「D:」以外の場合は、そのドライブ名をタイプします。
（3）cd \dosutils［Enter］
（4）rawrite -f \images\boot.img -d a［Enter］
（5）「Please insert a formatted diskette into drive A: and press --ENTER-- :」と表示されたら、再度［Enter］をタイプしてフロッピー作成を開始します。



Red Hat Linuxのインストール画面

Vine Linuxのインストール画面



Kondara MNU/Linuxのインストール画面

インストーラの起動

CD-ROM、もしくは作成したフロッピーをマシンにセットしてインストーラを起動します。インストーラが起動すると、「boot:」というプロンプトが表示されるので、［Enter］を押します。GUIインストーラがうまく起動しないときは、「boot:」の箇所でtextと入力して［Enter］を押し、テキスト画面でインストールを行います。

使用言語とキーボードの設定

使用言語とキーボードの設定は、ディストリビューションによって若干異なります。日本語106キーボードを使用するものとして、ディストリビューションごとに説明します。

3

■Red Hat Linux 6.2J

3-1

まず「Language Selection」で「Japanese」を選択します。
次の「キーボードの設定」では、

「モデル」→「Japanese 106-key」
「レイアウト」→「Japanese」
「デッドキー」→「デッドキーを無効にする」

を選択します。

Red Hat

■Vine Linux 2.0

3-2

キーボードの設定で「形式」→「Japanese 106-key」、「レイアウト」→「Japanese」、「バリエーション」→「None」を選択します。しかし、Vine Linuxのインストーラでは、ここでの設定が反映されないので、インストール後にスーパーユーザーでログインして次の設定を行います。viやEmacsなどのエディタで/etc/sysconfig/keyboardを開き、

```
# KEYBOARDTYPE="pc"
KEYTABLE="jp106"
```

と記述します。これで次回Linux起動時から日本語キーボードが使用できます。

Vine

■Kondara MNU/Linux 1.1

3-3

使用言語とキーボードの設定のみテキスト画面で操作します。「Choose a Language」で「Japanese」を、次の「キーボード タイプ」では「jp106」を選択してOKを押します。


Kondara

マウスの設定

2ボタンのPS/2マウスを使用する場合は、「2 Button Mouse(PS/S)」を選択します。「3ボタンマウスのエミュレーションを設定する（3 ボタンエミュレーション）」にチェックを入れると、2ボタンマウスのボタンを2つ同時に押すことで、3ボタンマウスの真ん中のボタンを押す効果が得られて便利です。

4

各種設定の開始

ディストリビューションによって画面が異なりますが、「次」を押すと各種設定が始まります。

5

インストールタイプの選択

「GNOMEワークステーション」、「KDEワークステーション」、「サーバー（サーバ）」を選択してインストールすると、ブートマネージャLILOをMBRへインストールします。LILO以外のブートマネージャを使うときは、この3タイプを選択してはいけません。また、サーバを選ぶとハードディスクをすべて初期化するので、すでにあるWindowsやLinuxパーティションをそのままにする場合は選択してはいけません。このあとは、自由度の高い「カスタム」を選択したものとして解説します。

6

スワップパーティションの作成

Disk Druidを用いて、Linux用のパーティションを作成します。Linuxには最低限スワップパーティションと、システム用パーティションの2つが必要です。「追加」を押すと、画面のようなダイアログが表示されます。搭載している物理メモリの1～2倍程度のサイズを指定し、パーティションタイプはLinux swapを選択します。「OK」を押すと作成したパーティション情報が表示されます。

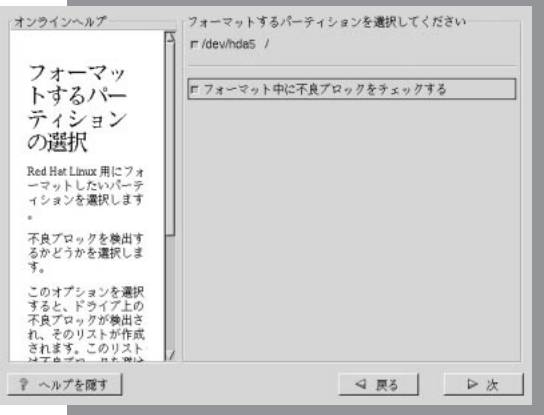
7

8 システム用パーティションの作成

Linuxシステム用パーティションは、マウントポイントに/を選択し、サイズは使用するディスク容量に合わせて指定します。ここで解説している3種類のディストリビューションは、フルインストール時に1.3～1.5Gバイト程度必要です。ハードディスクに余裕があれば、2Gバイト確保しておくとよいでしょう。

9 フォーマットするパーティションの選択

Linuxで使用するパーティションをフォーマットします。「フォーマット中に不良ブロックをチェックする（フォーマット中に不良セクタを検出する）」にチェックを入れると、フォーマット中に、ディスクに異常がないかどうかをチェックしてくれます。一度フォーマットすると、その領域にある情報はすべて削除されます。「次へ」をクリックする前に、誤ったパーティションをフォーマットしようとしていないかどうか、もう一度確認してください。



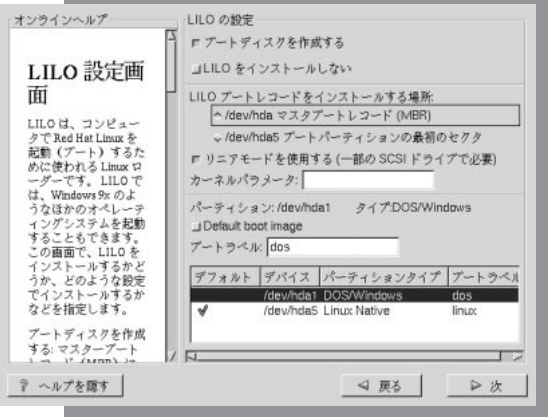
10 LILOの設定

「LILOブートレコードをインストールする場所（LILOを入れるレコード）」は場合に応じて選択します。

/dev/hdaマスターブートレコード（MBR）
- LILOを使用してWindowsとLinuxを起動時に選択する
- ディスクにはLinuxのみをインストールする

/dev/hda3ブートパーティションの最初の（第1）セクタ
- System Commanderなど、LILO以外のブートマネージャを使用する

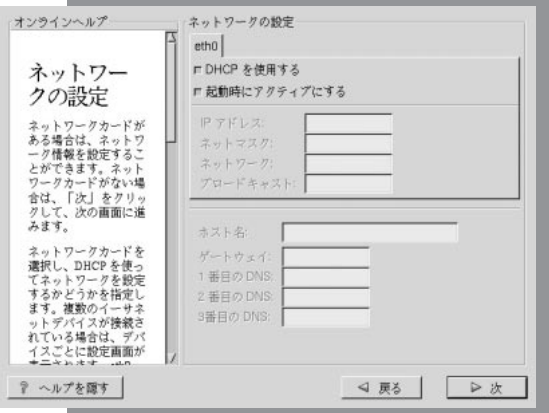
緊急時のために「ブートディスクを作成する（ブートディスクの作成）」をチェックしておくのもよいでしょう。



11 ネットワークの設定

イーサネットを使用するときの設定です。DHCPサーバからネットワークアドレスを取得する場合は「DHCPを使用する（DHCP使用時の設定）」をチェックし、それ以外のときは、各欄にIPアドレスやDNS情報を入力します。「起動時にアクティブにする（ブート時に有効にする）」にもチェックを入れます。

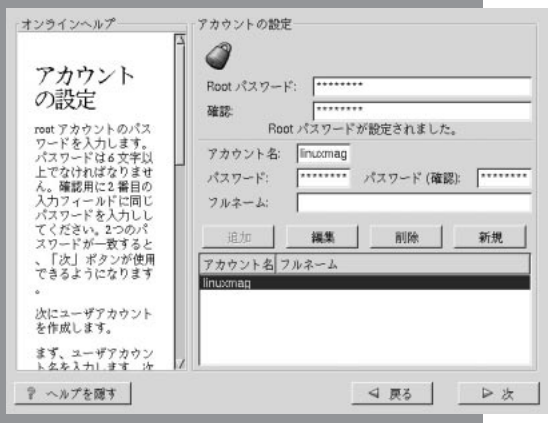
ネットワークカードが装着されていない場合には、この画面は表示されません。ダイヤルアップでインターネットを使用する場合は、インストール終了後にPPP接続ツール（PPxP、netcfg、kpppなど）を使ってネットワークの設定を行います。



タイムゾーンの選択

日本でマシンを使うときは、世界地図上の日本をマウスで指し、下の欄で「Asia/Tokyo」が有効になるようにします。

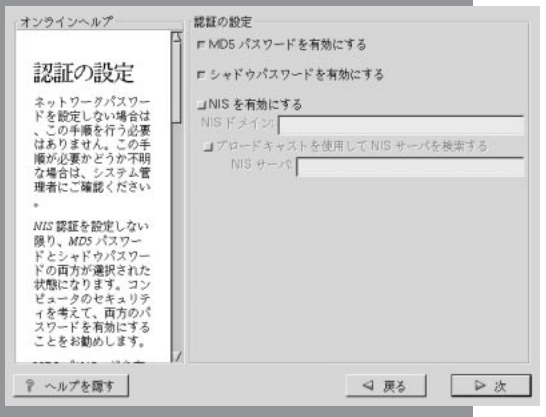
12



アカウントの作成

Linuxで使用するユーザーを作成します。まず、「Rootパスワード（管理者のパスワード）」と「確認」に同じパスワードを入力します。次に一般ユーザーを「アカウント名」、「パスワード」、「パスワード（確認）」、「フルネーム」に入力し、「追加」を押して作成します。この手順を繰り返すことで、複数のユーザーを作成できます。設定を終えたら「次」を押します。

13



ユーザーの認証設定

標準で「MD5パスワードを有効にする（使用）」と「シャドウパスワードを有効にする（使用）」にチェックが入っています。通常はこのままで構いませんが、複数のサーバのユーザーアカウントを一元管理するNISを用いて、認証するネットワークへ接続する場合は、NISの設定も行います。NISの情報に関しては、ネットワーク管理者へ問い合わせてください。

14

パッケージグループの選択

インストールするパッケージグループを選択します。ハードディスクに余裕があったり、パッケージグループの違いがわからないときは、一番下の「Everything」（全て）を選択するとよいでしょう。

15

Xの設定

16 16-1

■Red Hat Linux 6.2J

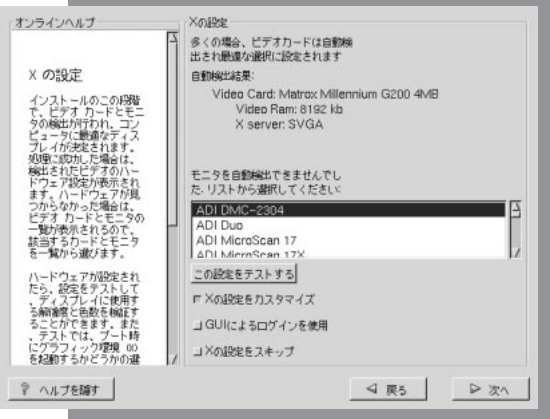
使用するモニタが自動認識されない場合は、リスト中からモニタを探します。リスト中にない場合は、「Generic Monitor」を選択します。次にビデオカードの設定ですが、一般的なビデオカードは自動で認識されます。自動で認識されたビデオカードの情報に誤りある場合は、リスト中からビデオカードが使用するチップと、ビデオカードに搭載されているメモリ量を選択して「次」を押します。

16-2

■Vine Linux 2.0とKondara MNU/Linux 1.1

ほとんどのビデオカードが自動で認識されます。デスクトップ環境を使用するなら「GUIによるログイン」にチェックを入れて「次」を押します。

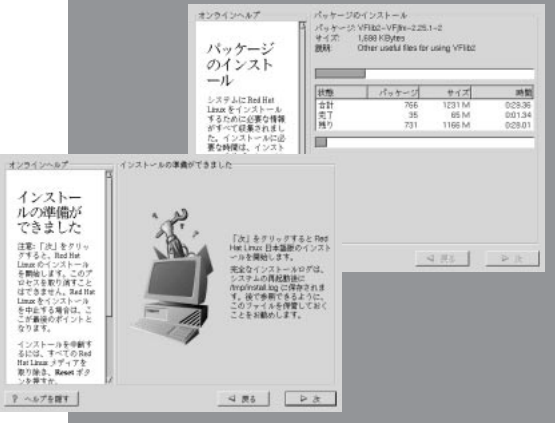
この箇所でXの設定がうまくいかないときは、Xの設定をスキップして、Linuxのインストールが終了したあとで、Xconfiguratorや、XF86Setupといったツールで Xを設定することもできます。Xがうまく設定できなくても焦らないでください。



パッケージインストールの開始

「次」を押すとパッケージのインストールが始まります。終了するまでしばらく待ちましょう。

17



インストール終了

Linuxのインストールが終了しました。「終了（閉じる）」を押すと、インストールプログラムが終了して、マシンが再起動します。再起動中にCD-ROMやフロッピーディスクを抜いてください。お疲れさまでした。

18

付録CD-ROMに収録した

Plamo Linux 2.0のインストール

(1) Plamo Linuxのインストーラは、ビデオカードを自動認識しません。また、ネットワークカードも自動認識しない場合があるので、インストール前に、あらかじめマシンのハードウェア情報を調べておく必要があります。
(2) 以下のインストール解説は、PC/AT互換機用です。NEC PC98x1へインストールする場合は、起動用フロッピーディスクの作成法などが若干異なります。

インストーラ起動用フロッピーの作成

CD-ROMブート可能なマシンでは、CD-ROMからインストーラを起動します。CD-ROMブートができないマシンの場合は、インストーラ起動用のフロッピーディスクを作成します。手順は以下のとおりです。まずCD-ROMと、空のフロッピーディスクをマシンにセットします。

次に、コマンドプロンプトを開き、以下のように操作します（ここではCD-ROMドライブ名をD:やQ:としましたが、マシン環境に合わせて入力してください）。

■PC/AT互換機の場合

(1) d:[Enter]
(2) cd at\install[Enter]
(3) rawrite \boot\bootdsk a[Enter]

■NEC PC98x1の場合

(1) q:[Enter]
(2) cd 98\144[Enter]
(3) makeplfd[Enter]
(4) デスクトップ用だと「3」を選択

インストーラの起動

作成したフロッピーディスクか、CD-ROMをセットして、マシンを起動します。インストーラが起動して、「boot:」が表示されたら[Enter]を押します。しばらくすると「plamo login:」と表示されるので、「root」と入力して[Enter]を押します。さらに「setup」と入力したあとで、[Enter]を押せば、Plamo Linuxのインストールが始まります。

キーボードの設定

画面に設定メニューが表示されます。まず「KEYMAP」を選択して、キーボードの設定を行います。メニュー画面の操作方法は、矢印キーや[Tab]で選択し、[Enter]で次へ進みます。「KEYMAP」を選択すると奥の画面が表示されます。日本語106キーボードを使用する場合は、デフォルトのまま<OK>でよいですが、CTRLキーとCapsLockを入れ替えたり、[全角／半角]キーを[ESC]に割り当てるときは、必要な項目に[Space]キーで[X]とチェックを入れて設定します。

4

パーティションの作成

Plamo Linuxではパーティション作成時にfdiskというツールを使います。ここでは、ハードディスクが、プライマリマスタに接続されているものとします。画面のように「fdisk」を選択して<OK>とすると、Linuxをインストールするディスクの選択画面になります。Linuxでは、プライマリマスタが/dev/hdaとなるので、/dev/hdaを選択して<OK>を押します。

5

fdiskの操作

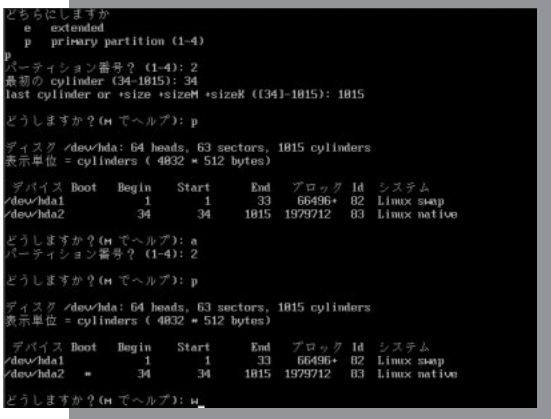
Linuxをインストールするには、システム用とSwap用の2つのパーティションが必要です。fdiskというツールを使い、パーティションを作成します。

■Linuxシステム用パーティションの作成

nコマンドを用いて、Linuxシステム用パーティションを作成します。お勧めパッケージをインストールする場合は、500Mバイト程度確保すればよいでしょう。

■Swap用パーティションの作成

nコマンドで物理メモリの1～2倍程度のパーティションを作成します。次にtコマンドを使い、パーティションIDを82にセットします。pコマンドを使い、システム用とSwap用の2つのパーティションが作成されたことを確認します。最後にwコマンドでパーティション情報を書き込みます。



6

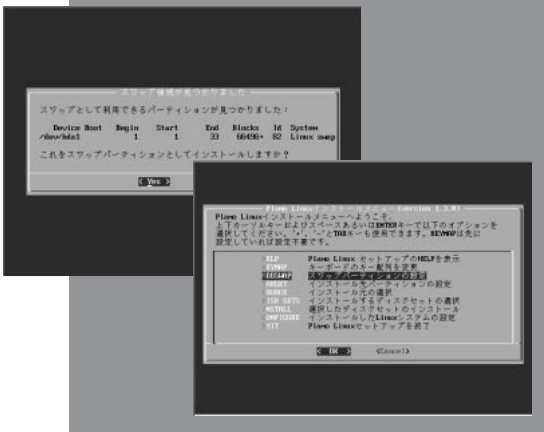
インストール開始

それではPlamo Linuxのインストールを始めます。メニューから「install」を選択して<OK>を押します。

7

Swapパーティションの設定

すでにキーボードの設定は終わっているので、インストール後にSwapパーティションが使えるように、上から3番目の「ADDSWAP」を選択して<OK>を押します。インストーラがSwap領域を表示するので、<Yes>を押します。このあとで、いくつかの質問が続きますが、「Linuxをインストールするパーティションの選択」まで[Enter]を叩いていけばよいでしょう。



システムパーティションの作成（その1）

今度はLinuxシステムをインストールするパーティションの設定です。Linuxシステム用として使用できるパーティションが表示されるので、リストの中から1つ選択して<OK>を押します。有効なパーティションが1つしかない場合は、そのまま<OK>を押します。

次にLinuxシステム用のパーティションをフォーマットします。選択するのは「Format」か「Check」のどちらかです。ハードディスクのフォーマットのみ行う場合は「Format」を、ハードディスクに異常がないか確認しながらフォーマットする場合は「Check」を選択して<OK>を押します。

システムパーティションの作成（その2）

ファイルシステムの単位であるi-nodeは、デフォルトのまま「4096」を選択して<OK>を押せばよいでしょう。i-nodeという言葉に興味がある読者は、先月号から始まった、連載「Linuxファイルシステムの現在と未来」を参考にしてください。Linuxシステム用パーティションのフォーマット終了まで、しばらく待ちます。

インストール元の選択

インストールするメディアを選択します。ここでは本誌付録のCD-ROMを使っているので、「CD-ROMからインストール」を選択して<OK>を押します。次にマシンに接続されたCD-ROMドライブを探すため、「scan」を選択して<OK>を押します。

インストール方法の選択

インストーラがCD-ROMドライブを探し出すと、画面のように表示されます。次の「インストール方法の選択」では「plamo」を選んで<OK>を押します。

パッケージの選択

12

インストールパッケージの選択です。デフォルトで「お勧めパッケージセット」がチェックされているので、そのまま<OK>を押します。次の画面では、インストールするマシンがノートPCかデスクトップか、使用する日本語入力プログラムはCannaかWnnかで、メニューが分かれます。デスクトップマシン上でCannaを使う場合は、右の画面のように「deskc」を選択して<OK>を押します。

パッケージインストールの開始

13

<Yes>を押して、選択したパッケージのインストールを始めます。インストール状況が表示されたら、終了するまでしばらく待ちます。

Xサーバのインストール

14

Plamo Linuxはビデオカードを自動認識しません。マシンを開けてビデオカードのチップを調べるか、Windowsでビデオカード情報を調べておきます。デフォルトではどんなビデオカードでも使用できる「16色用 Xサーバ」がチェックされています。これに加えて、Xがきれいに表示できるように、ビデオカードに対応したXサーバにチェックを入れて<OK>を押します。このあと、しばらくの間パッケージインストールが続きます。

カーネルのインストール

15

「お勧めパッケージセット」を選択していれば、Linuxカーネルは自動でインストールされているので、「skip」を選択して<OK>を押します。

起動用フロッピーディスクの作成

パッケージのインストールがひと通りおわったので、各種設定に入ります。「システム設定」で<Yes>を押して次へ進みます。緊急時に使う、Plamo Linuxの起動ディスクを作成します。まず、空のフロッピーをセットし、「format」を選択して<OK>を押します。フロッピーのフォーマットが終了したら、「simple」か「lilo」を選択して<OK>を押します。ここでは「simple」を選択して、Linuxカーネルのみをフロッピーへ書き込みます。起動用フロッピー作成が終わったら、「continue」を選択して<OK>を押します。

モデムの設定

ダイヤルアップ用にモデムを使う場合は、「モデムの設定」で<Yes>を押します。次の画面では、モデムをどのポートに接続しているかを選択して<OK>を押します。多くのマシンでは「ttyS0」か「ttyS1」のどちらかでしょう。

マウスの設定

PS/2タイプのマウスを使う場合は、画面のように、上から2番目を選択して<OK>を押します。

LILOのインストール（その1）

複数のOSをマシン起動時に選択するブートマネージャLILOをインストールします。ここではデフォルトのまま「simple」を選択して<OK>を押します。

20

LILOのインストール（その2）

LILOのインストール場所を選択します。System Commanderなど、LILO以外のブートマネージャを使うときは「Root」を選択し、それ以外の場合は「MBR」を選択します。「Floppy」を選ぶと、Linuxの起動は遅くなりますが、OSの起動で問題が出ないので安全です。

21

ネットワークの設定

この場面では、DHCPを利用するための設定が行えないので、IPアドレスなどの値を直接入力します。モデムやTAを用いてネットワークを使用する場合は、<No>を押してスキップします。ネットワークカードを使用する場合は、<Yes>を押して以下の項目を入力していきます。

- ホスト名
- ドメイン名
- IPアドレス
- ゲートウェイ
- ネットマスク
- DNS

22

タイムゾーンの設定

日本でPlamo Linuxを使うときは、デフォルトのまま「Japan」を選択して<OK>を押します。

23

終了

インストールが終了しました。[Ctrl]、[Alt]、[Delete]を同時に押して、マシンを再起動します。お疲れさまでした。