News Express

Distribution Software Hardware Headline Event

Linuxビジネス本格化に向かって Linux技術者増員中

Linuxはコミュニティの人々が開発し育ててきた。その開発に携わっている人たちはもちろんLinuxのプロといって差し支えないだろう。そして少し前までは、Linuxを使う＝そういう人たちの中に参加していくことだった。

しかし、Red HatやTurboLinuxなどの商用ディストリビューションが成熟し、Linux搭載PCが市販されるようになってきて、Linuxを利用したビジネス市場が形成されつつある。そこで、ユーザーをサポートするために別の意味でのプロ集団が必要になってきた。

昨年暮れから今年にかけて、ターボリナックスがTurbo-CE技術者認定試験を、レッドハットがRHCE認定試験をスタートしている。両社ともトレーニングもあわせて実施し、Linux技術を習得したエンジニアを世に送り出している。大塚商会では6月までに、Turbo-CE、RHCEの資格を持った認定エンジニアを200名に増員し、Linuxサポートを強化するという。

そして、特定のベンダーやディストリビュータに依存しないLinux技術者の認定機関、LPI（Linux Professional Institute）の日本法人（LPI-Japan）が発足し、7月よりLinux技術者認定試験を開始することになった。

こういう資格試験は主にSIベンダー向けといえるのだが、Linuxをサポートする体制が整うことで、いよいよWindows NT／2000からLinuxへシフトする条件が揃うことになる。本当の意味でのLinuxのブレイク近し、である。

Hardware　発売日　2000年5月

最高速Alphaプロセッサ搭載ワークステーション「AlphaStation DS20E／ES40」

URL　http://www.compaq.co.jp/

発売　コンパックコンピュータ株式会社
TEL　0120-018589
価格　322万円〜

コンパックは、最高速のAlphaプロセッサ21264A 667MHzを搭載したハイエンドワークステーション、Compaq AlphaStation DS20E／ES40を発表した。Tru64 UNIX、OpenVMS、Linuxの各OSに対応し、DS20Eは2 CPUまで、ES40は4 CPUまでのSMP構成をとることが可能。1 CPUあたり8Mバイトのオンボード・キャッシュや、5.2Gバイト／秒の帯域を持つメモリバスを採用し、SPECint2000／SPECfp2000テストで世界最高速を達成したという。またES40は、16Gバイト（32Gバイトまで拡張予定）までのメモリを実装可能で、大規模な科学技術演算にも対応している。

価格は、Alpha 21264A 667MHz、メモリ256Mバイトのメモリ、18Gバイトのハードディスクを搭載したAlphaStation DS20Eが322万円から、Alpha 21264A 667MHz、メモリ256Mバイト、18Gバイトのハードディスクを搭載したES40が536万7000円から（いずれもTru64 UNIX版）。

Software　発売日　2000年4月

PowerPC版Linuxに対応した開発ツール「Absoft ProFortran 6.2 for PowerPC/Linux」

URL　http://www.hulinks.co.jp/

発売　株式会社ヒューリンクス
TEL　03-5363-9041
価格　19万5000円

PowerPC版Linuxに対応したFortran開発ツール、Absoft ProFortran 6.2 for PowerPC/Linuxがヒューリンクスから発売された。

Fortran90／95、Fortran77コンパイラ、多言語対応Fxデバッガ（Fortran90／95、Fortran77、C、C++、アセンブラに対応）からなり、カーネル2.2以降のLinuxで動作する。メモリ64Mバイト、ハードディスクの空き容量50Mバイト以上が必要。またgcc、g77、f2cで作成したオブジェクトファイルとのリンクが可能。

BLASおよびLAPACK90ライブラリが標準で付属し、オプションでIMSL Math & Stat Librariesを追加できる。

価格は19万5000円。オプションのIMSLライブラリを付きは25万円。なお5月31日までAbsoft製品ユーザー向けの特価販売キャンペーンを実施しており、店頭でAbsoft製品のシリアル番号を提示すると、12万5000円で購入できる。

Hardware　発売日　2000年3月31日

超小型IPルータ「LAMB-RT-01SP（子羊ルータ）」

URL　http://www.wildlab.com/

発売　有限会社ワイルドラボ
TEL　ー
価格　5万2800円

ワイルドラボは、CATV、xDSLなどの高速インターネット接続利用者向けの超小型IPルータ、LAMB-RT-01SPの出荷を3月31日から開始した。

昨年発売されたLAMB-RT-01に（1）RS-232Cポートを追加、（2）パラレルポートの利用（要改造）、（3）MicroDrive（IBM社340Mバイト 1インチハードディスク）に対応といった機能を強化したもの。アーキテクチャ的にはLinuxを用いたAT互換機であり、IPフィルタリング、IPマスカレード機能を持ったルータとして使用する以外に、RS-232Cポートを利用することで、FAXサーバ／リモートアクセスサーバ、パラレルポートを利用したプリントサーバ、MicroDriveを利用したメールサーバの構築などの応用が可能。

これらの追加機能はサポートの対象外だが、ワイルドラボではユーザー交流のためのBBSを運営しており、同機の活用事例が得られるとしている。

価格は5万2800円。なお従来のモデルも4万9800円で併売する。

Hardware　発売日　2000年4月

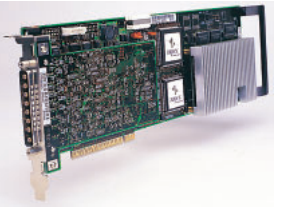
Linux対応MPEG2エンコーダボード「ZL-330」

URL　http://www.zapex.com/

発売　ブラザー工業株式会社
TEL　052-824-2072
価格　オープン価格

ブラザー工業は、同社の子会社である米Zapex Technologies社が開発した、Linux対応MPEG2エンコーダボードZL-330を、米国で4月に発売開始した。価格はオープン（予想価格は約1万ドル、日本円で約105万円）。なお日本では今年夏に発売予定。

ZL-330はPCIバス対応のカード。Red Hat Linux 6.1に対応し、ビデオサーバなどの各種放送／通信システムに利用可能。Zapex社が独自開発した圧縮アルゴリズムにより、高い圧縮率でも画質の劣化が少ないという。また用途に合わせて各種の通信プロトコルに対応したネットワークカードと組み合わせて利用できる。



Hardware

LinuxバンドルPC「Prosignia Server／ProLiant Linuxモデル」

発売日　2000年4月21日

発売　コンパックコンピュータ株式会社
TEL　03-5953-9589
価格　16万6800円〜

URL　http://www.directplus.compaq.co.jp/

コンパックは、Red Hat Linux 6.2J DeluxeをバンドルしたProsignia Server 740/720 Linuxモデル、およびProLiant ML350 Linuxモデルを発売した。同社のProsignia／ProLiantシリーズに、レッドハットから発売されているRed Hat Linux 6.2J Deluxeをバンドルしたもの。

これらの製品は、電話・インターネットによるPC直販事業「コンパック ダイレクトプラス」などを通じて販売される。価格は、Pentium III 500MHz、ECC付きメモリ64Mバイト、9.1Gバイトのハードディスク（Ultra2 SCSI）を搭載したProsignia Server 720 Red Hat Linux 6.2J Deluxeモデルが16万6800円から、Pentium III 600EB MHz、133MHz ECC付きメモリ128Mバイト、9.1Gバイトのハードディスク（Ultra2 SCSI）を搭載したProLiant ML350 Red Hat Linux 6.2J Deluxeモデルが29万8800円から。

また同社では、ProLiant ML350シリーズにターボリナックス ジャパンのTurboLinux Server日本語版、またはレーザーファイブのLASER5 Linux Server Editionをバンドルしたモデルも設定し、販売を開始する。

Hardware

ラックマウント式薄型サーバ「SGI 1200サーバ」

発売日　2000年4月18日

発売　日本SGI株式会社
TEL　0120-161086
価格　69万円〜

URL　http://www.sgi.co.jp/

日本SGIは、ラックマウント可能な薄型サーバであるSGI 1200サーバの発売を開始し、同機を用いたクラスタソリューション、サービスプロバイダ向けのサービスの提供を発表した。

SGI 1200サーバは、SGI 1000サーバファミリーのエントリーモデル。高さ2Uの筐体にPentium III 550MHzまたは700MHzを2個まで搭載可能。メモリは2Gバイト、ハードディスクは72Gバイトまで拡張できる。一般的な19インチラックに19台まで実装でき、効率良く大量のサーバを導入できる。

同社はSGI 1200サーバ向けのソリューションサービスとして、SGI ProPack 1.2を提供する。同サービスは、カーネル2.2.13に対応し、3.8Gバイトまでの物理メモリ対応、I／Oパフォーマンスの向上などを実現するアドインパッケージとして提供され、Red Hat Linux、TurboLinux、SuSEといったディストリビューションに対応する。

価格はPentium III 550MHz、メモリ128Mバイト、ハードディスク9.1Gバイトの最少構成で69万円から。同社では年間1000台の販売を見込んでいる。

Software

論理検証ツール「Verilog-XL」など

発売日　2000年第2四半期

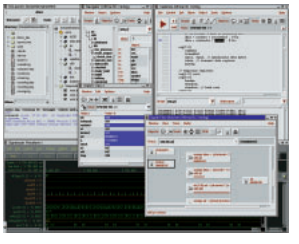
発売　日本ケイデンス・デザイン・システムズ社
TEL　045-475-2311
価格　—

URL　http://www.cadence.co.jp/

日本ケイデンスは、同社の論理検証ソフトウェアツール群をx86 PC版のLinuxに対応させ、2000年第2四半期から出荷すると発表した。

新たにLinuxでサポートされる製品は、Verilog-XLシミュレータ、Verifault-XL故障シミュレータ、言語混在シミュレーション向けのNC-Sim、NC-VHDLシミュレータ、NC-Verilogシミュレータなど、ケイデンスの論理検証ツールの全製品におよぶ。各ツール群のGUIや解析環境も、Linux上で実行できるようになる。

論理検証ソフトウェアツールは、回路やシステムなどを記述する言語であるHDL（Hardware Description Language）で書かれたLSIの機能やタイミングを検証するもの。



サイボウズ ピヨピヨキャンペーン

http://cybozu.co.jp/bep/

サイボウズは、春から新しい生活を始める人々への応援として「ピヨピヨキャンペーン」を実施中だ。アンケートに答えると、巷で人気のノートPCが当たるという企画だ。

応募資格は、サイボウズOffice3などの同社製品をダウンロード、試用してアンケートに答えるというもの。賞品はSONY VAIO PCG-N505A/BP、IBM ThinkPad i Series2621/464、Apple Computer iBookの計3台。

応募期間は5月31日（水）まで。当選者は抽選で決定し、6月上旬に同社のWebサイトで発表される予定。申し込みは上記のURLから。

News Headline from 日刊アスキーLinux

www.Linux24.com

アクアリウムコンピューター「silver neon」を発表

2000年4月23日

アクアリウムコンピューターは、マイクロ・サーバの新製品「silver neon」を発表した。silver neonは、デザイン的には現行機種「blue grass」の試作機「white neon」を継承し、機能については blue grassユーザーの意見を取り入れたもの。主な改良点は、(1) 前面のランプを増やし、サービスの稼動状態の確認ができるようになった、(2) 電源断の際に自動シャットダウンされるようになった、(3) PCとしてのスペックアップなど。silver neonは Celeron 533MHz、上限256Mバイトのメモリ、内蔵20.4Gバイトハードディスクとblue grassから大幅にスペックを上げている。

TurboLinuxおよびRed Hat Linuxをプリインストールする予定だが、バージョンは現段階では未定。管理ツールは、ホライズン・デジタル・エンタープライズの「HDE Linux Controller」を標準搭載し、Oracle8i、ロータスドミノといったサーバアプリケーションへの対応を図っていく。

出荷は6月14日の予定で、5月から先行受注を開始する。価格は未定だが、メモリを128Mバイト搭載したモデルで40万円以下になるという。

アクアリウムコンピューター
(http://www.aqua-computer.com/)

Oracleへの接続を高速化する「FastConnector」発表

2000年4月23日

デジタルデザインは、Oracleへの接続を高速化するミドルウェア「FastConnector for Oracle8i」(以下FC)、「FCReplicator for Oracle8i」(以下FCR) を発表した。

FCサーバ・クライアント間の複数のデータ送受信を1回にまとめ、独自の圧縮転送プロトコルを用いることで高速化を実現している。同社によれば、32kbpsから64kbpsの速度でも、LANと同等の高速性を実現可能としている。FCサーバが対応するOSは、Linux、Windows NT、HP-UX、Solaris (インテル版のみ) で、FCクライアントはWindows 9x／NTに対応する。現バージョンは、Oracle8i R8.0.5に対応し、Oracle8i R8.1.5には今後対応予定。

FCRは、FCと同等の技術をサーバ同士の接続に用い、レプリケーションなどを高速化する。

価格は、5クライアントのFastConnector for Oracle8iのLinux版が5万円、Windows NT版が7万5000円、商用UNIX版が10万円。2サーバ用のFCReplicator for Oracle8iのLinux版が20万円、Windows NT版が30万円、商用UNIX版が40万円。

デジタルデザイン
(http://www.d-d.co.jp/)

Webブラウザ「Mozilla Milestone 15」リリース

2000年4月23日

Mozilla Milestone 15 (以下M15) がリリースされた。M15は初めて「β版」となったMozillaである。

4月23日現在、The Mozilla OrganizationのFTPサイトには、Linux-i386版、Win32版、MacOS版 (MacOS 8.5以降が必要、PPC版のみ) など、各種プラットフォーム用のバイナリが用意されている。Linux-i386版のファイルサイズは約6.3Mバイト、ソースコードのファイルサイズは約22.3Mバイトだ。また、最新のソースコードは、開発用のCVSツリーからも取得できる。CVSによる取得については、www.mozilla.org 内の「source code via cvs」 ページを参照のこと。

Linux版の動作には、glibc 2.1.x環境が必要。

今回のリリースの主な変更点は、J2SE (Java 2 Standard Edition) のランタイム環境が含まれるようになったことや、マウスホイールによるスクロールが可能になったことなど。

Mozilla M15 Release Notes
(http://www.mozilla.org/projects/seamonkey/release-notes/M15.html)
FTPサイト
(ftp://ftp.mozilla.org/pub/mozilla/releases/m15/)

「デ変研TFライブラリ Ver1.26」発売

2000年4月18日

データ変換研究所は、Wordファイルなどからテキストを抽出するライブラリ「デ変研TFライブラリ Ver1.26」を発売した。同社の販売する「デ変研TEXT」、「デ変研DocCat」が持つ、WordやExcelなどのファイルからテキストを抽出する機能を、外部のアプリケーションから利用できるようにライブラリ化したもの。

同ライブラリは、拡張子ではなく内容からファイルの種類を自動判別、テキスト部分をUnicode (UCS-2) のテキストデータとして抽出する。また、Unicode (UCS-2) からEUC／SJISに変換するライブラリも用意されている。動作プラットフォームは、Linux (x86)、FreeBSD (x86)、Solaris (SPARC)、Windows NT (x86) となっている。

変換可能なWindowsソフトウェアは、Microsoft Word 95／97／98／2000、Microsoft Excel 95／97／2000、Microsoft PowerPoint 97／2000、一太郎9／10、OASYS V6／V7、ロータス ワードプロの各製品。

Linux版はTurboLinux 日本語版 4.2、Red Hat Linux 5.2など多くのディストリビューションで動作が確認されている。価格は、各プラットフォーム用の「デ変研TFライブラリ 開発キット」(1ユーザーライセンス) が40万円。

同社では、2000年5月末日まですべての機能が利用できる体験版を用意しており、同社のWebサイトからダウンロード

可能。ファイルサイズは485Kバイト。

データ変換研究所
(http://www.dehenken.co.jp/)

「Filesystem Hierarchy Standard 2.1」リリース

2000年4月15日

「Filesystem Hierarchy Standard（以下、FHS）2.1」がリリースされた。FHSは、Linuxの標準仕様を定める「Linux Standard Base（LSB）」の一部で、Linuxディストリビューションの、ファイルやディレクトリを配置する際の標準を定めている。各Linuxディストリビューション間の、ファイルやディレクトリ構造の違いによる混乱を防ぎ、互換性を保つために策定された。同じ目的で策定された、「File Systems Standard（FSSTND）」の後継にあたるが、「FHS」はLinuxに限らず、一般的なUNIX系OSにも適用可能な規格となっている。

「FHS 2.0」からの主な変更は、（1）ディレクトリ「/var/state」ではなく「/var/lib」を使用。（2）ディストリビューションは、ディレクトリ「/opt」を使用してもいいが、システム管理者の元でインストールされたソフトウェアであっても、「/opt」にインストールされたソフトウェアを変更、削除してはならない。（3）ディレクトリ「/var/share」は使用しない。（4）ディレクトリ「/usr/X386」は存在してもしなくてもよい。（5）ディレクトリ「/var/mail」は、今後もアプリケーションがメールスプールとしてアクセスするために利用されるが、実際のメールスプールへのシンボリックリンクでもよい。（6）ディレクトリ「/var/spool/smail」は存在しても、しなくてもよいなど。

PDF版またはgroff版のFHS 2.1が「Filesystem Hierarchy Standard」のWebページからダウンロード可能

Filesystem Hierarchy Standard
(http://www.pathname.com/fhs/)

「TrustedBSD」が発表される

2000年4月14日

TrustedBSD Projectは4月9日、FreeBSDをよりセキュアにするための拡張セット、「TrustedBSD」をアナウンスした。TrustedBSDはユーザープログラムとカーネル両方の拡張から構成され、NSA (National Security Agency)のセキュリティ評価基準「Orange Book」のB1レベルを目標としている。B1レベル達成のために、以下の機能が開発中とのこと。（1）サードパーティの認証モジュールを統合するための、拡張可能な認証フレームワーク。（2）最小限の権限を与えるために、各リソースごとに認証を必要にすること。（3）マルチレベルのセキュリティモデルを提供する「Mandatory Access Control」。（4）UNIXのセキュリティモデルより細かいファイルアクセス制御を提供する「Access Control List」。（5）セキュリティを侵害するイベント監視のサポート。

TrustedBSDの開発成果の大半は、FreeBSDにフィードバックする予定。

TrustedBSD Project
(http://www.trustedbsd.org/)

「Canvas 7 Linux Edition」のベータ版公開

2000年4月15日

米Deneba Softwareは、Canvas 7 Linux Editionのベータ版を公開した。

Canvas 7は、Windows、Macintoshのプラットフォームで有名なグラフィックソフトウェア。画像編集、ページレイアウト、Webページ編集などの機能を持っている。Canvas 7 Linux Editionは、Windowsのエミュレーションを行うソフトウェア「Wine」上で動作する。「Wine」は、配布物の中に含まれている。また、glibc 2.1以上の環境が必要。

公開されたのは、x86上のLinux用のもののみで、RPMパッケージ、debパッケージ、tarボールが用意されている。ファイルサイズはrpm版が15.2Mバイト、deb版が24Mバイト、tarボール版が15.2Mバイト。

米Deneba Software
(http://www.deneba.com/)

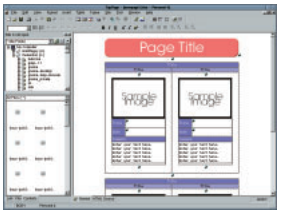
米IBM、Webページ作成ソフトのベータ版を公開

2000年4月14日

米IBMは、Webページ作成ソフト「IBM NetObjects TopPage for Linux」（以下TopPage）のベータ版を公開した。

TopPageは、日本では「ホームページ・ビルダー」として知られており、「IBM NetObjects TopPage」 Windows版は59.95USドルで販売されている。また、2000年の2月にもベータ版が公開されており、今回で2度目の公開となる。同社のWebサイトからダウンロード可能。ベータ版の使用期限は、2000年12月31日までとなっている。ファイルサイズは、本体のバイナリが16.7Mバイト、ドキュメントが9.4Mバイト、チュートリアルが22.5Mバイト。また、TopPageを動作させるには、Windowsのエミュレーションを行なうソフトウェア「Wine」が必要となるが、TopPage用にカスタマイズされた「Wine for TopPage」は、下記のURLからダウンロード可能。ファイルサイズは16.5Mバイト。

ダウンロードページ
(http://www.jp.ibm.com/esbu/E/toppage/down_linux_beta.html)



教育用OS「MINIX」がオープンソースに

2000年4月12日

Andy Tanenbaum氏が開発した、UNIX風の教育用OS「MINIX」のライセンスが、より自由度の高い「BSD License」と同様の内容に変更された。

これまでの「MINIX License」では、変更したソースコードを配布することはできないなどの制約があったが、新たなライセンスでは、ソースコードを自由に変

Linux CONFERENCE 2000 Spring開催

4月18日からの3日間、東京ファッションタウンにてLinux CONFERENCE 2000 Spring（以下LC2000）が開催された。Linux CONFERENCEは、1998年に日本Linuxユーザ会の主催のもと開催され、翌年以降は日本Linux協会が引き継いで開催しているイベントだ。当初は技術者向けのカンファレンス中心だったが、徐々にビジネスユーザー向けの内容も増えてきている。

LC2000では、テクニカルセッション、ビジネスセッションなどのカンファレンスと、BOF（Birds Of a Feather）が合わせて行われた。多数のセッションからいくつか紹介しよう。

2日め午後1時から行われた「日曜大工的PCクラスタ構築のすすめ」では、ビジュアルテクノロジーに勤務し、青山学院大学総合研究所客員研究員も勤める中田寿穂氏が、青山学院大学のPCクラスタ並列計算機プロジェクト「Aoyama Plus（Aoyama-gakuin University PC Cluster System）」の一環として構築された、PCクラスタシステム「ARK」について紹介した。続いて（1）CPUの数に比例した高速な演算、（2）非常に大規模な計算が可能、（3）複数の計算機での結果照合による信頼性の向上、（4） レンタルでも、1カ月で数千万〜1億円以上かかるスーパーコンピュータに比べると、非常に安価で構築できる、などのPCクラスタの利点を説明した。PCクラスタを実現できたのは、フリーの（1）UNIX互換OSであるLinux、（2）開発環境のGCC、（3）並列処理用ライブラリ「MPI（Message Passing interface）」、「PVM（Parallel Virtual Machine）」などの環境が揃ったためだという。

3日午後1時からは、「オープンソース・ライセンスを知る」と題した、東京大学大学院博士課程、大谷卓史氏によるセッションが行われた。

まず、オープンソースのソフトウェアの基本的な定義、「自由な再配布と修正ができるソフトウェア」を可能にしているのがオープンソースライセンスで、このライセンスを理解することで、オープンソースという思想そのものを理解できると説明があった。

また著作権には、著者の意図どおりに作品を発表する権利の「著作者人格権」と、著者が作品の正当な対価を得る権利の「著作者財産権」があり、さらに著作権を市場的に理解すると、ビジネス上の必要から収益の分配が目的となり、財産権が重視される。一方研究者的理解では情報の共有が目的となり、人格権が重視されるとの説明があった。オープンソースの目的は、ソフトウェアをいかに便利に使うかという関心に基づいており、著作者人格権重視の研究者的理解にきわめて近いことがわかる。

更でき、無償、商用を問わずに再配布可能となった。

今回のライセンスの変更は、4月7日、MINIXの開発者であるTanenbaum氏自身による、ニュースグループ「comp.os.minix」への投稿によって明らかになったもの。

現在、MINIXの最新バージョンは2.0.2となっており、「MINIX INFORMATION SHEET」ページからダウンロードすることができる。以前のバージョンである1.5は、x86のほかに、MacintoshやSun SPARCなどのプラットフォーム用も用意されてはいるが、2.0.2ではx86版のみ。

また、Cコンパイラなどの開発環境は、MINIX以前にTanenbaum氏が開発に関わった「Amsterdam Compiler Kit（以下ACK）」を元に作成したものが用意されており、MINIX発表当時から利用されている。以前のバージョンのCコンパイラは、ANSI Cに準拠していなかったが、現在は準拠している。

x86用のバイナリ配布物は3つに分かれており、それぞれ、ROOT.MNX（490Kバイト）、USR.MNX（740Kバイト）、USR.TAZ（3.7Mバイト）となっている。また、ソースコードの配布物は2つに分かれており、カーネルやライブラリなど、システム関連のソースコードは、SYS.TAZ（2.4Mバイト）、ユーザーコマンドのソースコードがCMD.TAZ（3.0Mバイト）。以前のバージョンのMINIXでは、開発環境は別配布となっていたが、現在は、バイナリ、ソースコード共に開発環境を含んだ形で配布されている。

Linuxの開発者、Linus Torvalds氏もかつてはMINIXを使用しており、「comp.os.minix」での、Tanenbaum氏と交わされたカーネルの性能や開発方針などについての論争は有名。

MINIX INFORMATION SHEET
(http://www.cs.vu.nl/~ast/minix.html)

Intel、セキュリティソフトウェアをオープンソースに

2000年4月12日

米Intelは4月11日、ドイツのミュンヘンで4月10日から13日まで行われていた「RSA conference 2000 Europe」において、セキュリティソフトウェア「Common Data Security Architecture（以下CDSA）3.0」をオープンソースにすることを発表した。

同製品は電子メールの暗号化や、e-Businessで必要となる認証、セキュリティの確保といったサービスを提供するミドルウェアとして機能する。現在、米Apple、米Compaq、米Hewlett-Packard、米IBMなどの企業で採用されている。

同社はCDSAをオープンソースとすることで、e-Businessの成長を加速したいという。

同製品は、Windows版、Linux版がリリースされ、リリーススケジュールはWindows版が2000年5月、Linux版が2000年8月となっている。また、同製品のLinux版はIA-32版以外に、IA-64版もリリースされる予定。IA-64版は、同社の支援するLinuxのIA-64移植プロジェクト「Trillian Project」の成果を元にItaniumプロセッサに最適化される予定。

プレスリリース
(http://www.intel.com/pressroom/archive/releases/cn041100.htm)

Distribution ▶▶▶

LinuxPPC 2000日本語版発売

市川充商店（アミュレット）は、PowerPC搭載機向けLinuxディストリビューション「LinuxPPC 2000日本語版」を4月20日に発売した。米LinuxPPC社がRed Hat Linuxをベースに開発した「LinuxPPC 2000」を日本語化したもの。価格は4800円。サポートは提供されない。

製品構成はLinuxPPC 2000のバイナリ、ソース各CD-ROMに加えて、日本語化環境「雪風」を収録したCD-ROMの計3枚。英語版のCD-ROMをインストールしてから、雪風に含まれるパッケージを追加することで、日本語化したLinux環境が得られる。

アミュレットでは、雪風単体での販売は行わないが、同社のFTPサイトからダウンロード可能になるという。

またPowerPC版のLinux上でMacOSを動作させる「Mac-on-Linux」が付属している。対応するMacOSは、日本語版も含めて7.5.2から9まで。

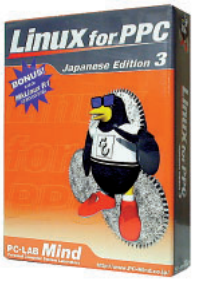
市川充商店（http://www.amulet.co.jp/ppclinux）

Linux for PPC Japanese Edition 3発売

マインドは4月25日から「Linux for PPC Japanese Edition 3」の販売を開始した。パッケージは1種類で、価格は6800円。前バージョンのオフィシャル版からのアップグレード価格は、4000円。メールおよびFAXによるサポートが提供される。

製品構成は、バイナリ、ソースコードの各CD-ROMに加え、「MkLinux R1」CD-ROMの計3枚。現在のLinuxPPCは、NuBus Macintoshなど古い機種では動作しないが、MkLinux R1を同梱することで、これらの機種でもLinuxが利用できる。

カーネル2.2.15、TrueType対応のXFree86 3.3.6、glibc 2.1.3、日本語Netscape Communicator 4.7 PPCなどを採用している。



またLinux上でMacOSを動作させる「Mac-on-Linux」も付属している。

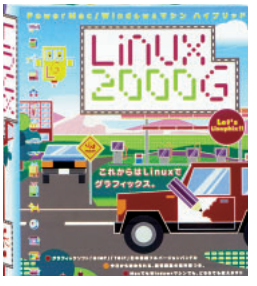
以前から付属していたlinuxppc-jp、linuxppc-jp-develメーリングリストなどの過去ログや日本語manページなどの検索システムが、GNOMEおよびKDEのメニューからアクセスできるようになっている。

マインド（http://www.pc-mind.co.jp/）

Linux2000G発売

グラフィック作成に特化したディストリビューション「Linux2000G」が4月21日にホロンより発売された。x86用バイナリ、PowerPC用バイナリ、ソースの各CD-ROMに加え、エクス・ツールス社の「Shade Preview kit for Linux」の体験版、PartitionMagic機能限定版（いずれもx86用）などを収録している。インストール、Linux2000Gマニュアルのほかに、グラフィックソフトのマニュアルを同梱している。価格は1万1800円。

カーネルは2.2.15pre4にDVD UDF（Universal Disk Format）などのパッチを当てたものを採用している。PowerPC用とx86用を単独のソースツリーからビルドするうえで、このバージョンが最適だったとしている。そのほかglibc 2.1.3、XFree86 3.3.5、GNOME Octoberリリース、KDE 1.1.2などを用いている。90日間の電子メール、電話による件数制限なしのサポートが受けられる。

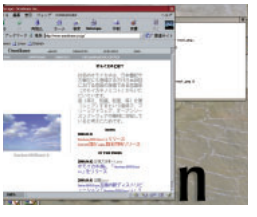


ホロン（http://www.linux2000g.ne.jp/）

「Omoikane GNU/Linux 1.0」リリース

オモイカネは3月31日、Debian GNU/Linux互換の新ディストリビューション「Omoikane GNU/Linux 1.0」（以下Omoikane）をリリースした。

Omoikaneは、3月30日時点のDebian GNU/Linux 2.2 フリーズ版のパッケージを収録しているため、Debian GNU/Linux 2.2との互換性が高く、手を加えることなく全パッケージを利用することができる。ソースも含めCD-ROM 8枚にもなる、potatoのSnapshot全パッケージから必要なものを選択し、バイナリ／ソースをCD-ROM 1枚に収めている。



ハードウェアの自動検出とモジュールロードが可能なスマートインストーラを採用し、インストール時の手間を減らしている。また、いったん行ったインストール作業と設定を再現可能なインストーラディスクを作成できるため、同一構成の多数のマシンへの全自動インストールが容易に行える。

オモイカネ（http://www.omoikane.co.jp/）

Products

100BASE-TX対応でファイル共有ができるネットワークディスク

MaxAttach

LAN環境が整ってくると、みんなでファイルを共有するスペースが欲しくなる。LinuxやWindows NT／2000でサーバマシンを立てればよいのだが、そこまで必要ないという場合には、LANにつなぐだけで簡単に使えるファイルサーバ専用機はいかがだろう。

製品名	MaxAttach
価格	18万9000円（20Gバイト）～44万5000円（80Gバイト）
問い合わせ先	株式会社ニューテック TEL 03-5777-0880 http://www.newtech.co.jp/

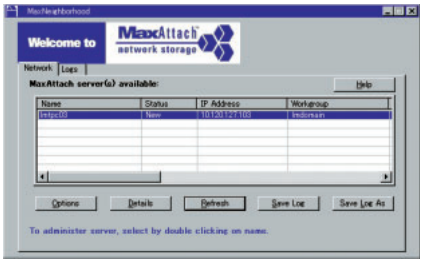
MaxAttachは、ハードディスクメーカーとして知られている米Maxtor社が製造しているFreeBSDベースのファイルサーバである。A4サイズのノートパソコンを厚くした程度の大きさのケースに、3.5インチのハードディスクを最大2台まで内蔵し、20Gバイト、40Gバイト、80Gバイトの3モデルが用意されている。

40Gと80Gモデルは、2台のドライブを内蔵していて、それぞれ別のディスクとして使うこともできるし、合わせて1台の仮想ドライブとして使うスパニングや、2台に同じ内容を書き込むミラーディスクとして使えば、片方にドライブ障害が起こったとしてもデータを消失せずに連続稼働が可能である。

設置と稼働までの時間はわずか

MaxAttachをひとことでいうと、LAN経由の外付けハードディスクドライブというのが、わかりやすいだろう。セットアップは非常に簡単で、ケーブルを接続し、電源スイッチを入れるだけである。

次に、MaxAttachの付属CD-ROMをWindowsマシンに入れ、MaxAttach接続情報管理ソフト「MaxNeighbordhood」をインストールし起動すると、LANに接続されているMaxAttachを自動的に検索し表示する（**画面1**）。検出されたMaxAttachをダブルクリックすれば、Webブラウザによって、MaxAttach



画面1 接続情報管理ソフトMaxNeighbordhood

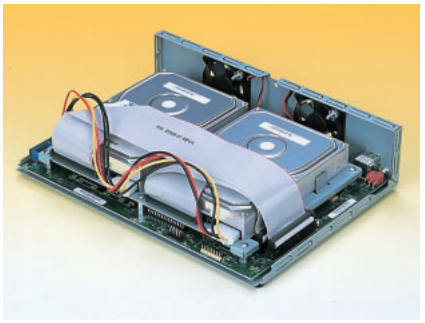


写真1
40Gバイトモデル（A0402AA）は、20Gバイトのハードディスクを2台内蔵している。使用しているDiamondMax 40シリーズの92049U4は、1プラッタあたり10Gバイトの高密度記録のため、データ転送速度は非常に速い。

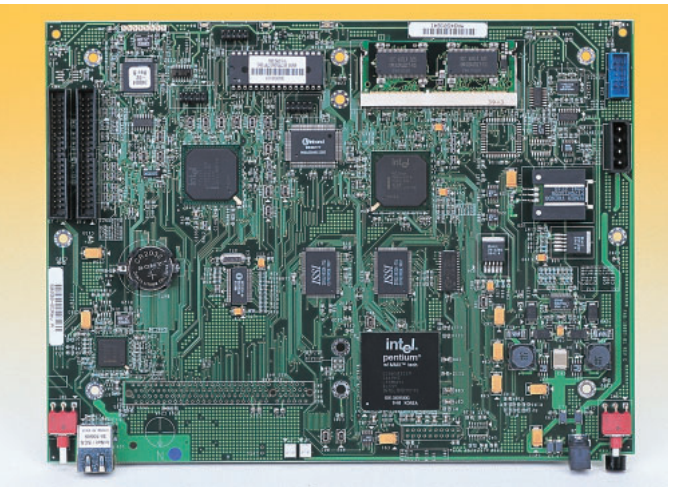


写真2　MaxAttach内部基板
CPUは、組み込み用MMX Pentium 266MHzを使用。2.0V動作のローパワータイプで、基板に直付けするPBGAパッケージだ。メモリは32Mバイトのモジュール、LANコントローラにIntel 82559を搭載。

に接続され、ユーザー管理などの各種設定を行うことができる。

Windowsからはネットワークコンピュータで、接続したMaxAttachが見つけられる。エクスプローラのネットワークドライブの割り当てで、MaxAttachの共有フォルダを、DやEといったドライブ名に設定すれば使いやすいだろう。

FTPサーバ、簡易Webサーバ機能を利用して、ファイルのアップロード／ダウンロードを行うこともできる。

LinuxやNTFSのパーティションに対応

10／100BASE-TXのLANでは速度が遅いと思われるかもしれないが、100BASE-TXのWindowsマシンで計測してみたところ、16Mバイト強のNetscape Communicator 4.7を約5秒で転送できた。メーカーの資料によると、最大6Mバイト／秒の転送速度でファイルの読み書きが可能だ。LinuxマシンからFTPで接続した場合には、約10Mバイト／秒でファイルの転送ができたので、100BASE-TXの能力いっぱいまで使えるようだ。

別売のユーティリティ「Reflect-It」を使用すると、Windows上のデータが更新された場合に、自動的にMaxAttachへも保存してくれるので、いざというときに役立つだろう。

MaxAttach自体にはデータバックアップの機能がないので、ネットワーク経由でデータを転送し、PC用のバックアップソフト（BackupExecなど）で定期的に行っておくようにする。

気になるのは故障したときだが、メールサーバ（SMTP）と送付先を設定しておけば、ハードディスクドライブに障害が発生した場合に、管理者にメールで知らせてくれる機能を備えている。ハードウェアの保証期間は3年間で、メーカーに連絡すると翌営業日には代替え品が発送され、そのあとで故障したMaxAttachを送り返すことになっている。

試用したMaxAttachは英語版だったため、Windowsからは特定の日本語ファイル名を含んだファイルの転送が行えなかった。

本誌が発売になる頃には、Webブラウザによる管理画面やヘルプメッセージが日本語に、そして日本語ファイル名やNFSに対応したバージョンアップが予定されている。なお、MaxAttachのOSのバージョンアップは、Webサイトから配布されるアップデートプログラムで簡単に行うことができる。

MaxAttachは、常に接続してグループ内の共有ファイル置き場として使うのもよいが、セットアップの簡便性を生かして、使用しているマシンのOSをバージョンアップする際や、マシンを入れ換えたりする場合に、ハードディスクのデータをMaxAttachに一時的にバックアップしておくといった用途にも向いているだろう。

型番	A0201AA	A0402AA	A0802AA
容量	20Gバイト	40Gバイト（20G×2）	80Gバイト（40G×2）
ドライブコンフィグレーション	なし	ミラーリング、スパニングが可能	
対応ネットワーククライアント	Windows 95／98／NT／2000		
対応ブラウザ	Netscape 4.07以降、Internet Explorer 4.0以降		
対応ネットワークサービス	DHCP、WINS、DNS		
セキュリティ	共有アクセス、ユーザーレベル認証、NTドメインを利用した認証		
管理機能	ユーザー／グループ管理、リモートシャットダウン／再起動 E-Mailによる障害通知		
ネットワーク接続	10/100Base-T Ethernet（RJ45）		
外形寸法（mm）	280（W）×230（D）×65（H）		
重量	3.5kg		
最大消費電力	40W	40W	60W
標準添付品	10/100Base-Tケーブル、ACアダプタ、 管理ソフトMaxNeighborhood（CD-ROM）		

表1　MaxAttachの主な仕様

iOfficeSSS（サザン）

iOfficeSSS（サザン）は、日々の商談内容、顧客情報、売上管理などを行うグループウェアだ。営業の業務の中で最も重要な商談から受注に至るまでの情報を共有することで、営業方法の分析や営業の育成を行い、営業・販売力の強化を図ることができる。

製品名	iOfficeSSS（サザン）
価格	198,000円～（10ユーザー・1サーバ。追加10ユーザー：10万円）
問い合わせ先	株式会社ネオジャパン TEL 045-912-5971 http://www.neo.co.jp/

株式会社ネオジャパンから、インtラネットWeb対応営業販売支援システム「iOfficeSSS（サザン）」（以下、サザン）が発売された。サザンは、既に発売されているiOffice2000の考え方をベースに、販売管理や売上管理に特化したイントラネットWeb対応グループウェアだ。

簡単なインターフェイス

グループウェアで最も問題になるのが、操作方法が複雑なので導入したにもかかわらず使われないという点だろう。管理職が使用できないということで結局使われないということが多い。その点、サザンのフロントエンドはWebブラウザなので特別な操作方法を覚える必要はない。通常のWebページを見る感覚と変わらないので、だれでも直感的に操作することができる。Webブラウザは Microsoft Internet Explorer 3.0以上か、Netscape Navigator 3.0以上があればよい。サーバ側にはApacheなどのWebサーバが必要になる。

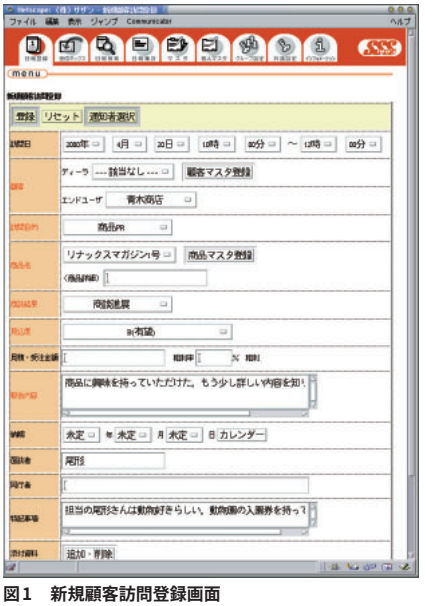


図1　新規顧客訪問登録画面
ここでは新規の顧客訪問に関する日報を入力する。商談が進展した場合は、既存顧客訪問登録で日報を入力することになる。

すべては日報から

サザンの最大の特徴は、日報入力を行うことですべての情報が管理されるという点だ。営業マンが作成する日報の情報を基に、商談管理、顧客管理、売上管理が行われる。日報の入力も簡略化されており、多くの項目はプルダウンメニューから選択できるようになっている。日報の作成は、新規顧客訪問登録と既存顧客訪問登録の2種類がある。新規顧客訪問登録では新規の客先に訪問した際の日報の入力を行い、商談が進展した場合は既存顧客訪問登録で日報を入力することになる（**図1**）。新規顧客訪問登録で入力した顧客情報は、顧客マスタに自動的に登録されるのでいちいち顧客情報を登録する必要はない。

図2　達成グラフ（部署別）
部署ごとの達成率や個人の達成率などを表示できる。計算はすべて日報に書かれたデータによって行われる。

作成した日報は上司や同僚に送信して、アドバイスや指示を受けるようになっている。送信された日報は「受信ボックス」に表示される。

日報は上司や同僚がいつでも閲覧することができるようになっており、訪問先や売上などさまざまな項目で検索できるので非常に便利だ。さらに、入力した日報の情報はさまざまな方法で集計することができる。訪問先一覧や顧客情報、売上などが簡単に集計でき、売上やあらかじめ設定された目標額への達成率もグラフで表示させることができる（**図2**）。売上見込みや商談の進行状況も一目でわかるので管理職の負担を減らす効果も大きい（**図3**）。日報の集計機能は、誰でも使うことができるので、

成績の良い営業マンの売り方を分析することで、どのようにしたら売れるのかを学ぶことができる。

また、営業担当が代わった場合でも、過去の販売実績や情報などを参照することでスムーズな引き継ぎを行うことができる（**図4**）。

スケジュール機能

グループウェアの便利な機能のひとつに、スケジュール管理機能がある。自分のスケジュールはもちろん、同行者などのスケジュールを予約することができる機能だ。サザンにもスケジュールを管理する機能がある。スケジュールは、自分、グループ（部署）、同僚の3タイプで予約することができる。しかし、特定の3人などというような場合、1人ずつ予約しなければならないので面倒だ。もう少し細かく選べるといいだろう。

予約したスケジュールはメールで通知することもできる。また、定期的に訪問するような場合も毎日、毎週、毎月、あるいは日にちを指定など細かく設定することができる（**図5**）。

ただし、スケジュールの重複はチェックされないので、スケジュールが重複しているかどうかは自分で確認しなければならない。同僚がいつのまにか予約したスケジュールと重複していたという問題が発生する可能性がある。また、スケジュールを予約したことを通知するためのメールアドレスも、その都度入力しなければならない（**図6**）。多くの入力項目がプルダウンメニューから選択できるようになっている分、利用頻度が高いと思われるこの機能にプルダウンが付いていないのは残念だ。次期バージョンで対応することを望む。

図3　日報集計（部署別検索画面）
商談の進行状況が一覧できる。選択すると個々の商談状況について詳しく調べることができる。

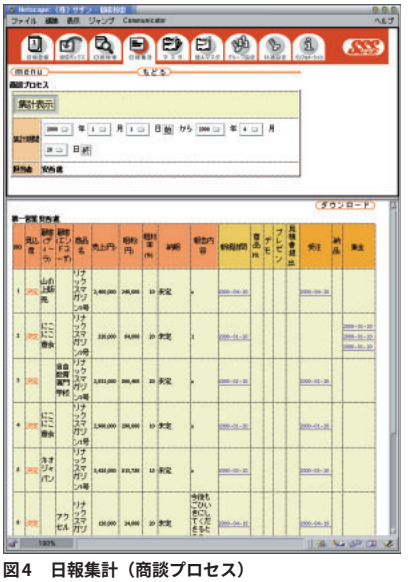


図4　日報集計（商談プロセス）
過去の商談に関する情報を一覧表示できる。引き継ぎや同僚の売り方について調べることができる。

動作環境

Webサーバ上で動作するサザンは、Linuxをはじめ、Windows、Solaris、HP、FreeBSDなど多くのプラットフォームに対応している。Webサーバも標準的なものはほとんどサポートされている。インストールも簡単なので、本誌付属のCD-ROMに収録されている体験版（使用期限40日）をインストールしてみるといいだろう。

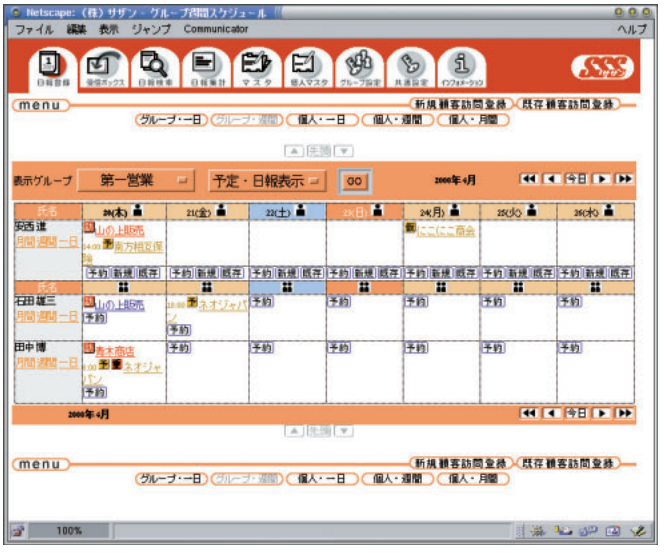


図5　スケジュール画面
この画面でスケジュールの予約と確認ができる。日報に入力した情報も自動的に反映される。

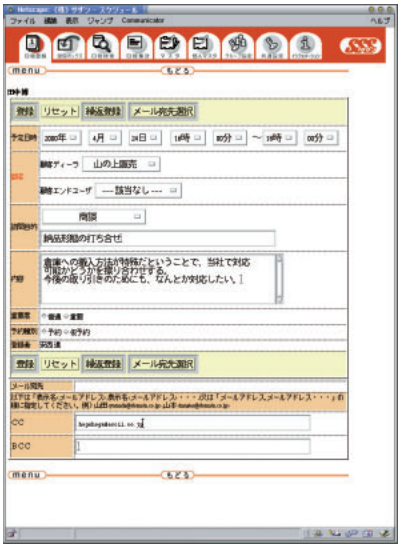


図6　スケジュール（予約）
同僚のスケジュールを予約することができる。予約通知としてメールを送信できるが入力が面倒だ。

Distribution

新着ディストリビューション

Vine Linux 2.0 CR Official製品版

日本語ディストリビューションの雄であるVine Linux。待ちに待ったその最新版がリリースされた。カーネルやXFree86も大幅にバージョンアップし、KDEなども新たに収録された。Vine Linux 1.1との比較を中心に、その人気の秘密を探ってみる。

Corel LINUX OSエンハンスメント

目指すはWindowsリプレースか。Corel LINUXの超簡単インストーラは、キーボードやマウスの設定はおろか、Xの設定すら不要だ。またCorel LINUX独自のファイルマネージャから、設定なしでWindows環境へアクセスする光景は驚きすら覚える。ほかのディストリビューションとは一味違う先進機能を見てみよう。

Red Hat Linux 6.2J

Linuxディストリビューションとして、最大のシェアを誇るRed Hat Linux。Red Hat社により日本語化されたRed Hat Linux 6.2Jは、大容量メモリやRaw I/Oをサポートするなど、システム内部が高度にチューニングされている。Red Hat 6.1Jからの変更点に焦点をあてて紹介しよう。

Kondara MNU/Linux 1.1

新しさと安定さの両立。Kondara Linuxは新しいパケージを次々とリリースし、それらを怒濤のようにバグフィックスしていく。プロジェクトのユニークな雰囲気の裏側には、貪欲に追求された最新テクノロジーが隠されている。そのKondara Linuxが目指すものとは、いったいなんだろうか？

Vine Linux 2.0 CR Official製品版

完成度の高いLinuxディストリビューションとして人気のVine Linux。満を持してその最新バージョンがリリースされた。Vine Linuxは、インストール直後から使いやすい日本語環境が整っているのが特徴で、国内でのシェアは高い。前バージョンのVine Linux 1.1（以下Vine 1.1）がリリースされたのが昨年の6月なので、Vineファンには待ちに待ったリリースということになるだろう。Vine Linuxの製品版にはマニュアルやDynaFont 5書体、Wnn6といった商用アプリケーションのほか、試用版が多数バンドルされ、技術評論社から発売されている。製品版についてのサポートは、ユーザー登録後3ヵ月以内に限り、書面、FAX、メールで受けられる。

Vine Linuxとは

Vine Linuxは使いやすい日本語環境を目指すため、以下のような工夫がなされている。

●検索エンジンNamazu

LinuxにはJFやJMANといった、日本のLinuxコミュニティにより整備された日本語ドキュメントが多数存在する。これらの日本語ドキュメントを快適に利用するため、Vine Linuxでは日本語検索エンジンNamazuを用い、デスクトップ上でキーワード検索ができるようになっている。

●独自に追加されたプリントツール

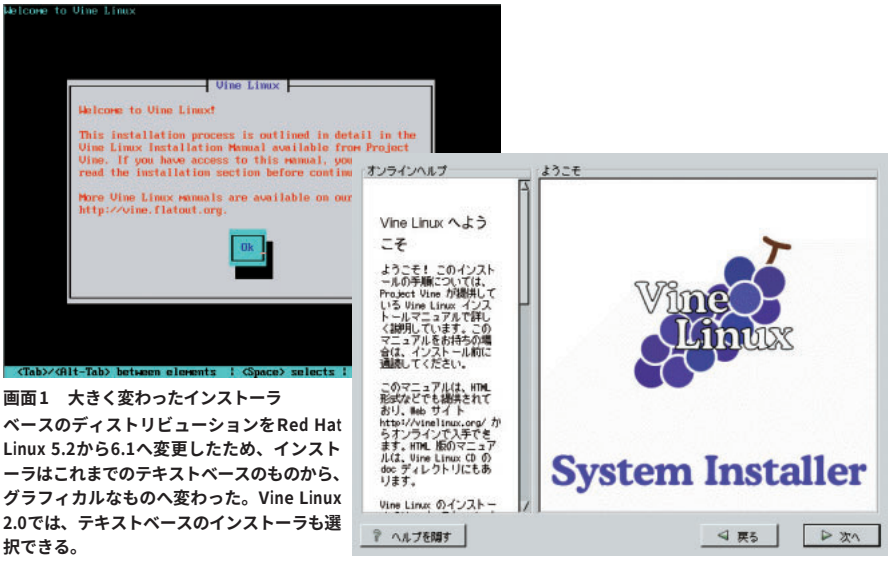
LinuxはWindows環境などと比較して、日本語印刷が弱点となっているが、Vine Linuxでは日本語対応のプリンタを利用するために、プリントフィルタを独自に追加して、美しい日本語ドキュメントの印刷を可能にしている。

●日本語カタログの整理

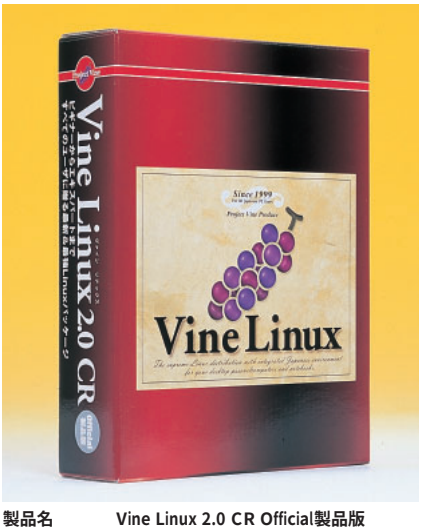
Vine Linuxでは、topやpingなどを実行した際のメッセージも日本語化されている。Vine Linuxをコンソールで使用するときはどうしてもkonを起動する必要が出てくるので、これら日本語カタログに関しては賛否が分かれるところだ。ただこれについては、マニュアルで日本語カタログを英語表示に変更する方法も解説されているので、さほど問題にはならないだろう。

グラフィカルなインストーラ

Vine 1.1はRed Hat Linux 5.2がベースだったが、今回リリースされたVine Linux 2.0（以下Vine 2.0）はRed Hat Linux 6.1をベースにしており、インストーラがテキストベースのものから、Red Hat Linux 6.1で採用されたGUIベースのものに変更されている（**画面1**）。グラフィカルなインターフェイスに慣れたユーザーにとっては直感的な操作ができるという意味で嬉しい変更だ。また従来のテキストベースのインストーラもオプションで選択できるので、好みに合わせたインターフェイスを選択すればよいだろう。



画面1　大きく変わったインストーラ
ベースのディストリビューションをRed Hat Linux 5.2から6.1へ変更したため、インストーラはこれまでのテキストベースのものから、グラフィカルなものへ変わった。Vine Linux 2.0では、テキストベースのインストーラも選択できる。



製品名　Vine Linux 2.0 CR Official製品版
価格　9800円
問い合わせ先　株式会社技術評論社
03-3225-3481
http://www.gihyo.co.jp/

最新のアプリケーション

Vine 2.0はカーネルのバージョンが2.2.14に、XFree86が3.3.6に、ライブラリがglibc2.1.2にと、大幅にバージョンアップされており、サポートされるハードウェアも増えている。また新規ハードウェアを自動で認識するkudzuの採用により、新規ハードウェアの追加も容易になった。Vine 1.1と2.0の機能比較を表1にまとめたのでこちらも参考にして欲しい。

Vine 2.0は使いやすいデスクトップ環境を提供するために豊富なアプリケーションが収録されている。これらを以下に見ていこう。

●豊富なエディタ

エディタはEmacsのほか、XEmacsも使用できるようになっている。Emacs系エディタはelispという言語で設定ファイルを記述するため、elispに慣れないユーザーが一から設定ファイルを作成するのは困難だ。このためVine Linuxでは、以前からこの設定ファイル.emacsの雛型を提供してユーザーの便宜を計っていたが、Vine 2.0ではXEmacsも追加されたため、これらの設定ファイルを.emacs.elと.xemacs.elに分割し、起動時に.emacsで切り替えるようになっている。このほかにも、軽快な動作がウリのEmacsライクなエディタJedも含まれている。

画面3　人気上昇中のメーラWanderlust
Emacs系エディタで動作するメーラとして人気上昇中のWanderlust。画面はXEmacs上で起動したところ。

画面2　選べるウィンドウマネージャ
デフォルトのウィンドウマネージャはこれまでと変わらずWindow Makerだが、デスクトップ環境では最近のLinuxディストリビューションでは標準となりつつある、GNOMEやKDEも選択できる。

●選択幅が広がったウィンドウマネージャ

ウィンドウマネージャはこれまでどおりWindow Maker（**画面2**）がデフォルトで起動するようになっているが、ほかのディストリビューションで標準になりつつあるGNOME（**画面2**）やKDEを使いたいユーザーにも配慮して、グラフィカルログイン画面でウィンドウマネージャを選択できるようになっている。

●デスクトップ用途には必須のメーラ

インターネットが一般家庭にも普及した現在、デスクトップユーザーにメーラは必須アイテムだ。Vine 2.0にはEmacs系エディタ上で動作するメーラとして定評のあるMewや、人気上昇中のWanderlust（**画面3**）が採用されている。

国産ディストリビューションの雄

このほか、マニュアルも日本語文書作成のためのツール紹介や、プリンタの設定に多めのページが割かれている。またPPxPを利用したPPP接続や、オリジナルメーラ（vmail）の利用など、Vine Linuxでのインターネット利用方法の雛型も提示しており、デスクトップ用途として使用するため必要な解説がなされている。このように、安定した日本語環境にこだわりを見せるVine Linuxは、初心者にもヘビーユーザーにも満足のいくディストリビューションといえるだろう。

Vine Linuxのバージョン	2.0CR	1.1CR
カーネル	2.2.14	2.0.36
ライブラリ	glibc2.1.2	glibc2.0.7
XFree86	3.3.6	3.3.3.1
デスクトップ環境	Window Maker、KDE、GNOME、Fvwm2	Window Maker、Fvwm2
ベースディストリビューション	Red Hat Linux 6.1	Red Hat Linux 5.2
日本語入力プログラム	Wnn6 ver.3、Canna	Canna

表1　Vine Linux 1.1と2.0の主な違い
新しいVineではシステムの基本となるカーネルやライブラリが大きく変更されている。

Corel LINUX OSエンハンスメント

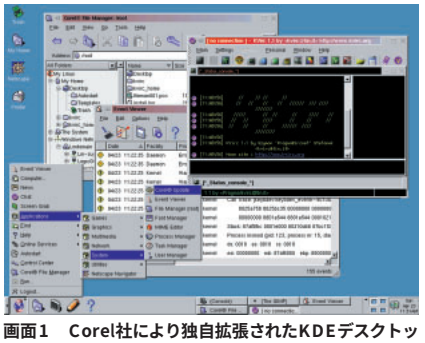
Corel LINUX OSエンハンスメント（以下Corel LINUX）はオフィス用アプリケーションWordPerfectなどでおなじみのCorel社が開発しているディストリビューション。Windows環境との親和性が高く、独自拡張されたKDEデスクトップ環境（**画面1**）が特徴で、Debian GNU/Linuxのパッケージ管理方式を採用している。Corel LINUXは、メディアヴィジョンから発売中のCorel LINUX OS 英語版のカーネルやXFree86をアップデートしたもので、WordPerfect Office 2000 for Linuxの一部として収録されている。このWordPerfect Office 2000には、ワープロソフトのWordPerfect、スプレッドシートのQuattroPro、データベースのParadoxなどが同梱され、Linux上に統合オフィス環境を提供する。Corel LINUXは、単体での販売はされていない。

独自アプリケーション

Corel LINUXにはデスクトップユーザーの使い勝手を考慮した独自アプリケーションが収録されている。

■Corel File Manager

標準のファイルマネージャCorel File Manager（**画面2**）は、Windowsライクなファイルマネージャで、ドラッグ＆ドロップなど直感的に操作できるのが特徴だ。KDE標準のKFM（K File Manager）と比較すると、Windows環境もシームレスに扱える点で優れている。



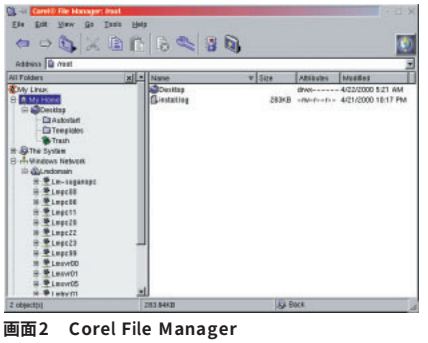
画面1　Corel社により独自拡張されたKDEデスクトップ環境。

■Corel Update

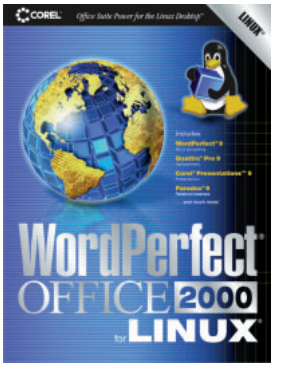
Corel LINUXでは最新の環境を保つため、Corel Update（**画面3**）を採用している。このCorel Updateは、CD-ROMやFTPサイトからパッケージのアップデートを行うツールだ。ネットワークを利用したアップデートでは、FTPプロトコルのほかにHTTPも利用できる。ユーザーはこのCorel Updateを使うことで、Debianのパッケージ管理方式を簡単に利用にできるというわけだ。

超簡単インストール

インストーラは基本的な設定が3画面しかなく、ほかのディストリビューションと比較しても、最も簡単な部類といえる。マウス、キーボード、ビデオカードといったハードウェアも自動で設定してくれるので、数個の質問に答えてインストールを開始すれば、再起動直後からX環境で作業ができる。



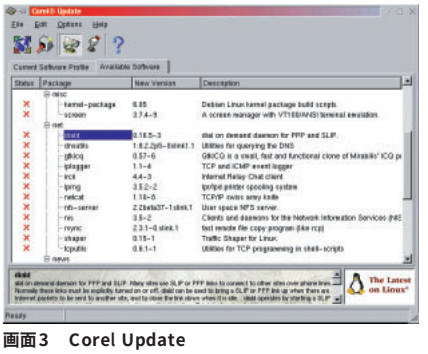
画面2　Corel File Manager
Corel社により独自開発されたファイルマネージャ。画面のように、Windowsのエクスプローラに似たインターフェイスを持ち、Winodws環境とシームレスに接続できる。



製品名　WordPerfect Office 2000 for Linux
問い合わせ先　Corel Corporation
http://linux.corel.com/

このように、デスクトップ用途として、Windowsなど複数のOSが混在する環境でも、ほとんど設定不要で使用できるCorel LINUXだが、現在リリースされているのは英語版のみと、日本のユーザーには少し寂しい。ただ、今夏にリリース予定となっているCorel LINUX日本語版により、日本のユーザーもCorel LINUXの先進性を享受できることになるだろう。今から日本語版のリリースが待ち遠しい。

本誌付録CD-ROMに、Corel LINUX OSエンハンスメント（北米版）のダウンロード版を収録しています。なお、商用ソフトウェアは含まれていません。

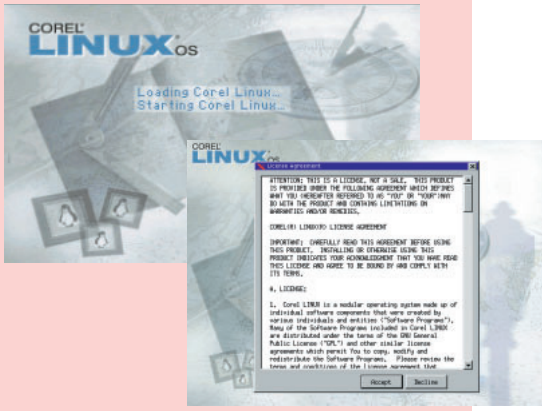


画面3　Corel Update
debパッケージの管理方法にうといユーザーも、このアップデートツールを使えば、簡単に最新の環境が保てる。

Corel LINUX OSエンハンスメントのインストール

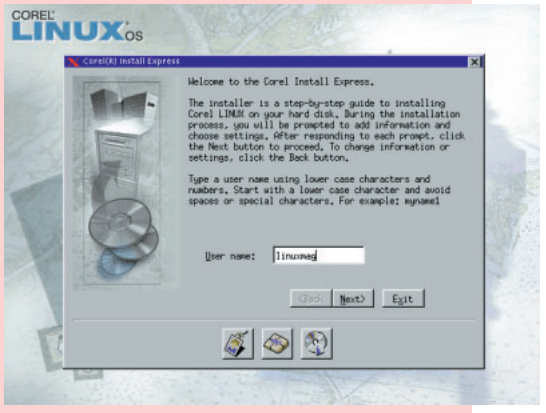
CD-ROMブートが可能なマシンでは、Corel LINUX OSのCD-ROMをドライブに入れて再起動します。CD-ROMブートできない場合は、起動用フロッピーを以下の手順で作ります。

（1）Windows 9x／NTが起動しているPCのドライブに、Corel LINUX OSのCD-ROMを入れます。（2）自動的に起動するCorel Linux Autorunウィザードで起動用のフロッピーを作成します。（3）自動的にAutorunウィザードが起動しない場合は、CD-ROMのルートディレクトリにあるAutorun.exeを、エクスプローラから起動します。



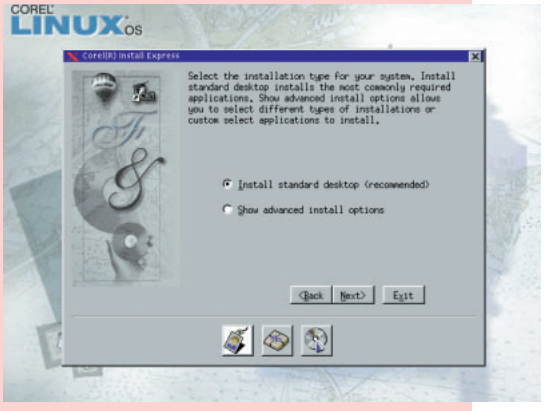
インストーラの起動

CD-ROMから起動可能なマシンならCD-ROMをセットして、そうでないマシンなら上記の手順で作成した起動用のフロッピーディスクでインストーラを起動します。ライセンスへの同意を求められますので、ライセンスに同意する場合は[Accept]を押します。



一般ユーザーの登録

ここでは一般ユーザーを登録します。適当なユーザー名を入力して下さい。

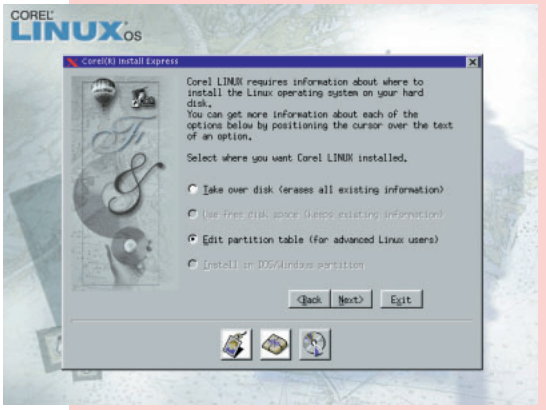


インストールタイプの選択

[Install standard desktop]が推奨されています。これを選択してインストールを行うと、約320Mバイトのディスクスペースを必要とします。グループごとやパッケージごとのインストールをしたいユーザーは、[Show advanced install options]を選択してください。以下では[Install standard desktop]を選択したものとして解説していきます。

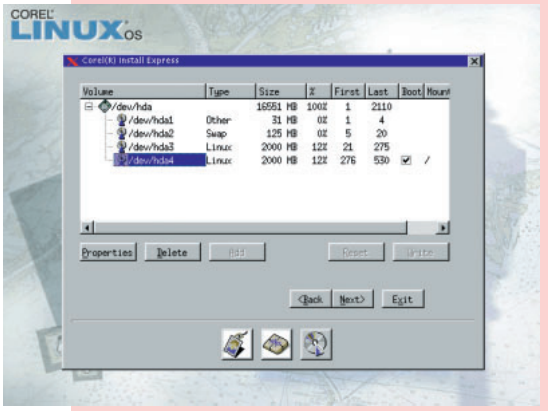
パーティション作成タイプの選択

インストールしようとしているハードディスクをCorel LINUX専用に使うときは[Take over disk]を選択します。これを選択すると、ハードディスクの情報はすべて削除されますので注意が必要です。[Use free disk space]と[Install in DOS/Windows partition]はそれぞれ、使用中のディスクの空き領域にインストールする場合、すでに存在するWindowsやDOSパーティションへインストールする場合に選択します。ここでは[Edit partition table]を選択し、ハードディスクのパーティションを編集していきます。これらのタイプはユーザーの環境によって異なりますので、自分の環境に適したものを選択してください。



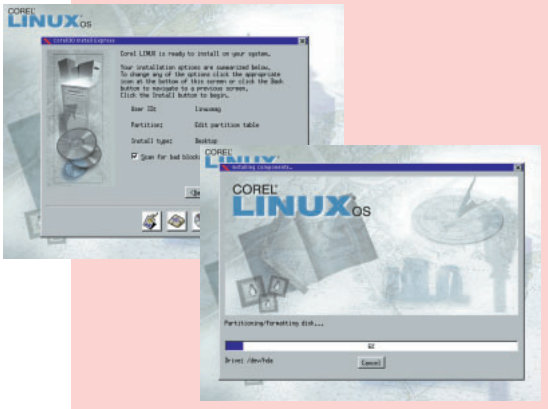
パーティション作成

Corel LINUXをインストールするためには、Swap領域用とLinuxシステム用の最低2つのパーティションが必要です。パーティションを作成するには、画面の場合ならば/dev/hdaをマウスで有効にして、[Add]ボタンを押します。画面に表示されているパーティションを選択して[Properties]ボタンを押すと、ファイルシステムのタイプや、Linux起動時のマウントポイントが指定できます。Swap領域は物理メモリの1～2倍程度作成して下さい。同様にパーティションを選択して[Delete]を押すとそのパーティションが削除されます。Swap用パーティションとシステムパーティションを作成したら[Next]を押します。



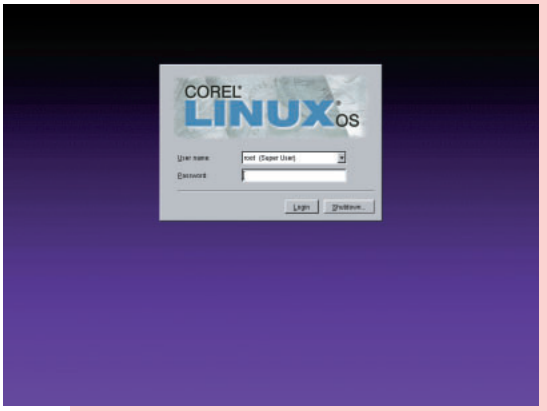
設定内容の確認

これまでの設定で、ユーザーが選択した情報が表示されます。これらの設定を変更したい場合は、画面下部にある3つのボタンが、左から順番に[User id]、[Package]、[Partition]となっているので、変更したい項目をクリックします。画面に表示されている設定でよい場合は[Install]を押して、パッケージのインストールを始めます。すると、画面が切り替わり、棒グラフでディスクのフォーマットやパッケージインストールの進行状況が表示されますので、インストールが終了するまでしばらく待ちます。



インストール終了

無事インストールが終了すると、自動的にCorel LINUXが起動して、シンプルなログイン画面が表示されます。最初はrootでログインします。インストール直後はパスワードが設定されていませんので、そのまま何も入力せずに、Enterキーを押します。ログインできたら、rootのパスワード設定を行う画面になるので、適当なパスワードを入力します。以上でインストールは終了です。



本誌付録CD-ROMに収録されているCorel LINUXは、インストール直後は英語用キーボードの設定となっています。日本語キーボードを使用するには次の手順で設定変更を行います。rootでログイン後、適当なエディタで（/etc/X11/XF86Config）を開きます。このファイルの中に以下のような箇所があります。

```
XkbModel        "pc101"
XkbLayout       "us"
```

上記で「pc101 → jp106」、「us → jp」と変更して保存します。Ctrl+Alt+Backspaceを同時に押してXを再起動します。

Red Hat Linux 6.2J

Red Hat Linux 6.2J（以下Red Hat 6.2J）はレッドハット株式会社が販売するLinuxディストリビューションで、本家Red Hat, Inc.が開発しているRed Hat Linuxを日本語化したものだ。当然のことながら、Linuxディストリビューションの中でも大きなシェアをほこるRed Hat Linuxに向けて開発された、さまざまなパッケージを気兼ねなく利用できる。

Red Hat Linux 6.1日本語改訂版（6.1J）よりカーネルバージョンが2.2.14、XはXFree86 3.3.6が、ライブラリはglibc2.1.3などと基本システムが最新版にアップデートされ、製品タイプを3種類にしてユーザーの便を計るなど、細かい変更が行われている。

選べる3タイプ

Red Hat 6.2Jには3タイプの製品ラインナップが存在する。各タイプの違いを表1とともに見ていこう。まず、デスクトップ用に開発された「スタンダード」と「デラックス」だが、「デラックス」には電話でのサポートが用意されているほか、商用アプリケーションがバンドルされているのが主な違いといえるだろう。この2種類のパッケージについては、収録されている商用アプリケーションが、どの程度自分に必要かを考慮して選択すればよいだろう。このほかにも詳細は未定だが、レッドハット社が開発した高セキュリティWebサーバ「Secure Web Server」を同梱するeコマース用の「プロフェッショナル」も5月にリリース予定となっている。ここではデスクトップ用に開発され、商用アプリケーションをバンドルした「デラックス」を紹介する。

製品名　Red Hat Linux 6.2J
価格　1万2800円（デラックス）
問い合わせ先　レッドハット株式会社
03-3257-0411
http://www.redhat.com/jp/

商用アプリケーション

この「デラックス」は6.1Jの後継にあたる製品で、日本語入力プログラムとして定番のATOK12SEやWnn6のほか、日本語ワープロのdb/NOTE for Linux/BSD Ver.2.02、ブートマネージャのSYSTEM COMMANDER Liteなどがバンドルされる（**表1**）。

また、試用版として機能制限がつくものの、3Dグラフィックソフトとして、各種プラットフォームで人気のShade for Linux Preview Kit（**画面1**）や、データベース開発環境のdbMAGICエンタープライズサーバLinux版なども含まれ、商用アプリケーションを利用して、本格的なLinuxでの開発を考えるユーザーにはありがたい。

本誌付録CD-ROMに収録されているのは、Red Hat Linux 6.2の<FTP英語版>です。この記事で紹介しているRed Hat Linux 6.2Jとは内容面で違いがあります。また、商用ソフトウェアは含まれていません。

先進的なチューニング

この「デラックス」に限らず、Red Hat 6.2Jは3タイプ共通で、4Gバイトの物理メモリ使用、Windows 95/98で使用するVFATファイルシステムへのインストール、データベース使用に欠かせないRaw I/Oなど、随所に先進的なチューニングが施されている。

	スタンダード	デラックス	プロフェッショナル
税抜価格	3980円	1万2800円	2万9800円
発売日	4月21日	4月21日	5月中旬
サポート	メール（90日間）	メール（90日間） 電話（30日間）	未定
商用ソフト	なし	ATOK12SE Wnn6 Ver.3 dp/NOTE for Linux/BSD Ver.2.02 SYSTEM COMMANDER Lite DynaFont 5書体 Adobe Acrobat Reader for Linux EPSON PM770／800 Printer Driver“ PIPS ”	未定

表1　Red Hat Linux 6.2Jの各種ラインナップ比較
デスクトップ用に開発された2タイプ「スタンダード」と「デラックス」の主な違いは商用アプリケーションのバンドルのようだ。このほかにe-コマース用サーバ構築のために開発された「プロフェッショナル」もリリース予定だ。

この中でも、現在Windowsを使用しているマシンに、Linux用のext2ファイルシステム用パーティションを作成したくないユーザーにとっては、VFATへのLinuxインストールのサポートは便利な新機能である。また大容量物理メモリの使用やRaw I/Oのサポートはサーバ用途としての使用にも十分耐えうるだろう。

このほか、1つのLinuxシステムを複数台のマシンで構成するクラスタリング機能（Piranha）もサポートしている。一時にアクセスが集中するWebサーバなどにはクラスタリング機能は必須といえるが、これまではクラスタリングをサポートしているディストリビューションが少なかっただけに、本格的なWebサーバ構築を考えるユーザーには新たな選択肢が増えたといえる。ちなみにこのクラスタリング機能のサポートは、Red Hat 6.2Jのインストール時に選択できる。

Sawmill採用

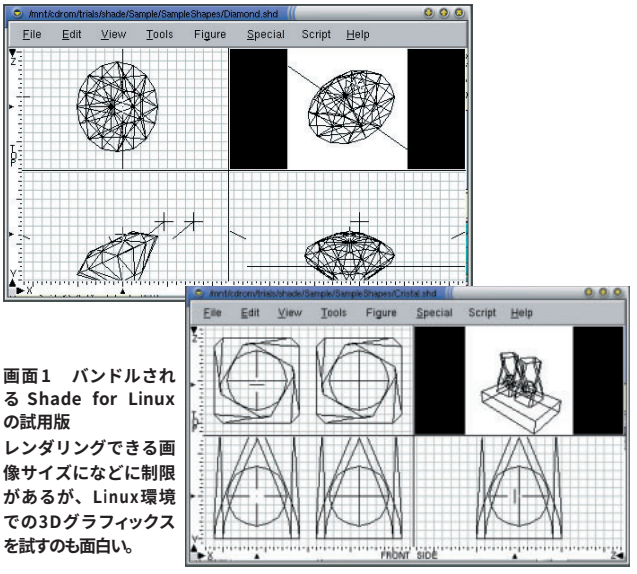
今回からウィンドウマネージャに新しくSawmillを採用している（**画面2**）。このSawmillは機能の多くをLispという言語で実装しており拡張性に優れている。単体でも使えるが、メモリ消費量も小さく、動作も軽快であるため、GNOME環境でEnlightenmentの代わりに使うのも良いだろう。Sawmillを採用しているディストリビューションがまだ少ないせいか、日本でのユーザーは少数のようだ。しかしながら、昨今の重くなりすぎたX環境に辟易しているユーザーにとって、軽快なX環境を取り戻すという意味で朗報といえるだろう。ただ、Red Hat 6.2JではGNOME使用時に、デフォルトでEnlightenmentが起動するようになっている。そのためSawmillを使うためには、GNOME Control Center（**画面2**）から使用するウィンドウマネージャを変更する必要がある。

ドキュメントCD-ROM

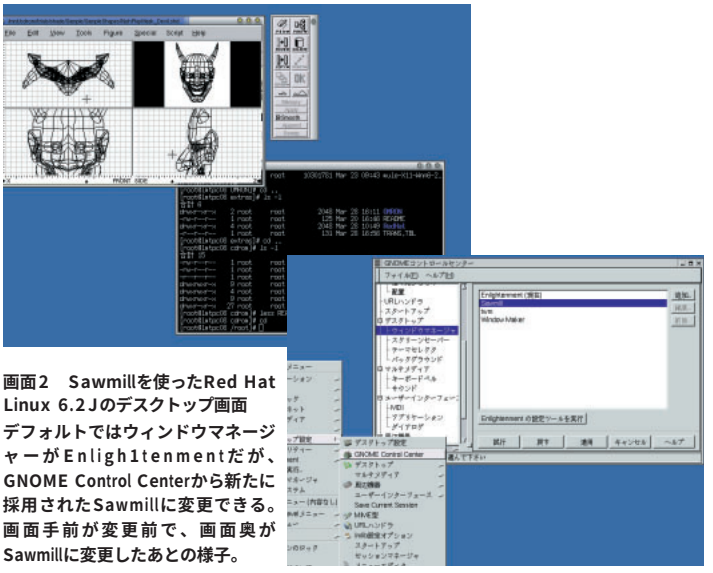
Red Hat 6.2Jにはインストール用CD-ROMのほかにドキュメントCD-ROMが付属する。このCD-ROMには「インストールガイド」や「リファレンスガイド」などが収録され、Webブラウザで閲覧できる。特に「リファレンスガイド」はLinuxconfを用いたLinuxの設定方法や、rpmコマンドの使用法など、基本的なLinuxシステムの管理方法が丁寧に解説されているので、初心者、ベテランユーザーを問わず重宝する。CD-ROMにはこのほかにも、JFをはじめ、翻訳されたLinux関連のドキュメントが収録されており、Windows環境などからも参照できるのが嬉しい。

デスクトップ用途としての課題

とはいえ、デスクトップ用OSとして見てみると、UNIX系OSでは定番のMewや、近頃話題のWanderlustといったメーラがデフォルトでインストールされないのが気になる。PCをデスクトップ用途に使うユーザーには、Webブラウザとともに、使い勝手の良いメーラが必須であると言えるので、なにかしらユーザーに推奨するメーラが欲しいところだ。またCD-ROMに収録されている「リファレンスガイド」もLinuxのシステム管理的な内容に偏っているきらいがあるので、このガイドも含めて、使いやすいデスクトップ環境としてのさらなる拡張が今後の課題といえるだろう。



画面1　バンドルされる Shade for Linux の試用版
レンダリングできる画像サイズになどに制限があるが、Linux環境での3Dグラフィックスを試すのも面白い。



画面2　Sawmillを使ったRed Hat Linux 6.2Jのデスクトップ画面
デフォルトではウィンドウマネージャーがEnlighltenmentだが、GNOME Control Centerから新たに採用されたSawmillに変更できる。画面手前が変更前で、画面奥がSawmillに変更したあとの様子。

Kondara MNU/Linux

新しいアプリケーションの採用や、迅速なバグフィックスなどが特徴の日本語ディストリビューション、Kondara MNU/Linux 1.1（以下Kondara 1.1）がリリースされた。

このKondara 1.1の製品版はデジタルファクトリージャパンから、インストールCD-ROMのほか、マニュアルと商用DynaFont5書体が付属して、7800円で発売されている（アカデミックパックは5200円）。電話、FAX、Web、メールによる24時間365日体制のサポートが新たに受けられるようになった（3件まで）。

1.0との主な違い

Kondara 1.1と1.0の主な違いを表1にまとめた。カーネル、ライブラリ、XFree86といった基本システムがアップデートされ、新たなアプリケーションも追加された。

■GNOME & Sawmill

Kondara 1.0はデフォルトのウィンドウマネージャがEnlightenment（**画面1**）だったが、Kondara 1.1では新たにSawmillというウィンドウマネージャが収録されている。このSawmillは柔軟なカスタマイズが可能なウィンドウマネージャで、軽快な動作とともに、デスクトップ環境GNOMEとの親和性が高いというのが特徴だ（**画面1**）。

■日英バイリンガル

日本語環境のディストリビューションとして、Kondara LinuxとVine Linuxは比較対照となることが多い。Vine Linuxが日本語化にこだわるのに対し、Kondaraは国際化を目指すというスタンスをとっている。たとえばVine Linuxではpingやtopといったアプリケーションのメニューも日本化していたり、各アプリケーションのエラーメッセージなども日本語で表示される。このように、特定の言語環境に特化するプログラミングをKondaraでは推奨しておらず、一度の開発でさまざまな言語環境への対応を目指すというわけだ。そしてKondara 1.1では、1つのバイナリ（コンパイル済みのプログラム）で日本語と英語に対応し、国際化を実現している。Kondara 1.1のグラフィカルなログイン画面（**画面2**）を見ると、Languageメニューから「English」、「English+Japanese」、「Japanese」という3つの使用言語タイプが選択できるようになっており、国際化プログラミングの成果がうかがえる。

■sdr

上記のように国際化されたデスクトップ環境を機能させるため、sdr（simple display reconfigure）を収録している（**画面3**）。このsdrはX上でもコンソール上でも動作し「使用言語の選択」、「ウィンドウマネージャ」、「日本語入力プログラム」の設定変更が行える。

Alphaとソース共有

Kondara Linuxはリリース当初から64ビットのAlphaアーキテクチャをサポートし、今回から各アプリケーションのソースコードを、PCとAlphaという2つのアーキテクチャで共有するようになった。これまでは、最新のアプリケーションがリリースされても、Alphaのサポートは後回しになることが多かった。しかし。ソースコードが両プラットホームで1本化されたことにより、AlphaユーザーもPCユーザーに

画面1　クールなデスクトップ
前バージョンで標準だったEnlightenment（奥）と今回新たに採用されたGNOME+Sawmill（手前）。どちらも非常にクールなデザインだ。このほかにもKDEやWindow Makerが選べる。

Kondaraのバージョン	1.0	1.1
カーネル	2.2.13	2.2.14
glibc	2.1.2	2.1.3
XFree86	3.3.5	3.3.6
Emacs	—	20.5
GNOME	—	1.1.4
sawmill	—	0.24
sdr	—	0.9.4

表1　Kondara 1.0と1.1の主な機能比較
各アプリケーションのマイナーバージョンアップや、新規追加がされている。

遅れをとることなく最新アプリケーションを利用できるようになった。

独自の収録アプリケーション

Kondaraには、ほかのディストリビューションではあまり目にしないアプリケーションも収録されている。

■メーラ

muttはコンソール上で動作するメーラで、POPはもちろん、スレッド表示やPGPもサポートしている。

■ニュースリーダ

slrnは多機能と動作の軽快さを両立させたニュースリーダだ。muttと同様ターミナル上で動作し、マウスでの操作も可能だ。どちらも日本語化されており、kon上での動作も可能だ。これらmuttやslrnはS-Langという画面制御ライブラリを利用しており、elispを使用したEmacsのように高度なカスタマイズが可能だ。

■Ruby

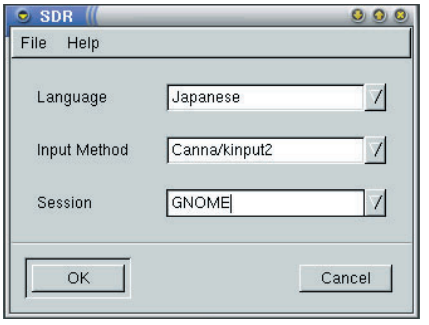
日本で開発されているスクリプト言語で、PerlやPhysonと比較されることが多い。Rubyはオブジェクト指向を前提に開発されているので、シンプルなコーディングが可能だ。また、日本人の手で開発されていることもあり、日本語環境でも安心して利用できる。

■mph

Marvellous Package Hackerを略した名前のこのツールは、インストールされている各種パッケージを簡単にアップデートするものだ。同種のツールとしてはRed Hat社が開発するrpmfindやDebianで次期パッケージ管理ツールに予定されているaptのバックエンドプログラムapt-getがあげられる。このツールを利用すれば、あらかじめ登録しておいたFTPサイトへアップデート情報の確認をとり、インストールまでも行える。これにより常に最新の環境を保つことが可能だ。mphはスタンドアローンな環境でも有効だが、複数マシンのメンテナンスを行う際に、特に威力を発揮するだろう。

ユニークな構成のマニュアル

Kondara 1.1の製品版には約260ページのマニュアルが付属する。このマニュアルには、インストール方法、各ウィンドウマネージャの使用法、PPxPを用いたダイヤルアップの解説をはじめ、先にあげたメーラmuttや、Emacs上で動作するメールクライアントWanderlustの使用方法なども紹介されている。また、ディストリビューション付属のマニュアルとしては、珍しくRPMパッケージの作成方法が解説されている。

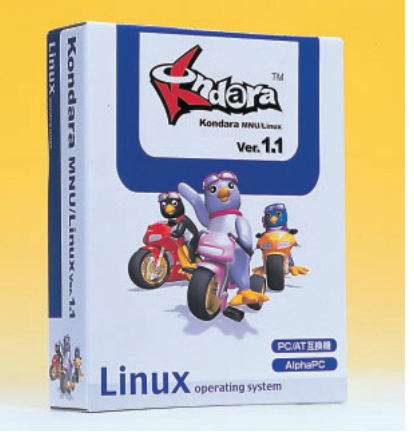


画面3　日英バイリンガルを機能させるsdr
このツールでウィンドウマネージャ、日本語入力プログラム、使用言語環境が選択できる。

ネットワークをフルに活用する

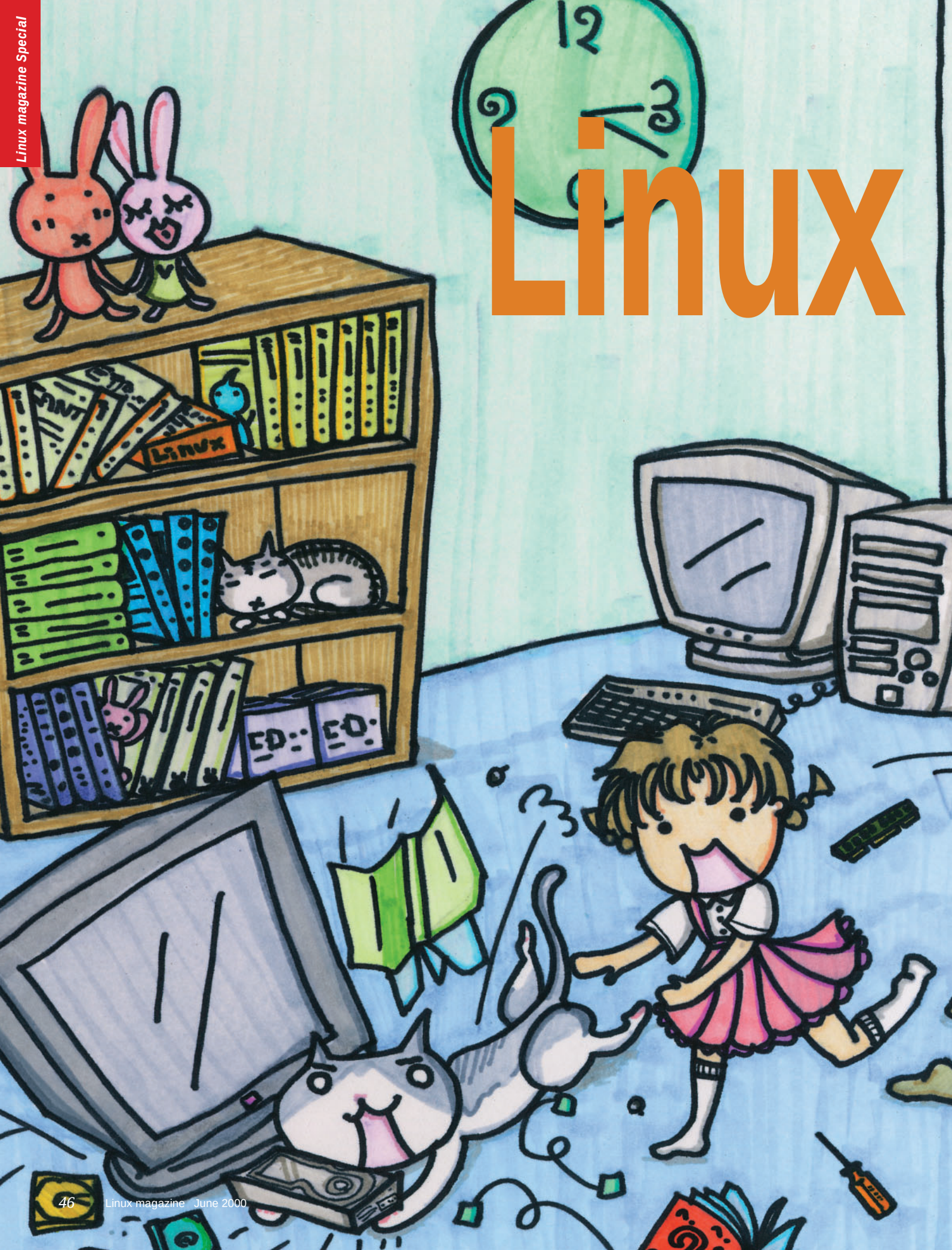
KondaraのWebサイト（http://www.kondara.org/）では「バグ宙太」というバグトラッキングシステムが管理されており、Kondaraのバグフィックスに一役買っている。先にあげたmphとともに、Kondaraは、システム管理のために、ネットワークを最大限に活用するディストリビューションといえよう。このようにプログラムのメンテナンスを念頭においた国際化や、異種プラットホームでのソース共有、そしてネットワークをフルに利用したシステム管理など、最新のテクノロジーを試してみたいユーザーには特にお勧めできるディストリビューションだ。

画面2　グラフィカルなログイン画面
こちらもデスクトップ同様、Kondaraらしい凝ったデザインだ。この画面で、日本語入力プログラムや、使用するウィンドウマネージャを選択する。



製品名　Kondara MNU/Linux1.1
価格　7800円
問い合わせ先　デジタルファクトリジャパン株式会社
06-6882-5850
http://www.digitalfactory.co.jp/

Linux



マシンを作る!

こだわりの完全自作、お手軽なベアボーン、安心のメーカー製パーツキット
手作りマシンでLinuxが動く　!

Linuxマシン自作のススメ～PCの仕組みに強くなる～

文：編集部　Text : Linux magazine

今から10年ぐらい前だろうか、486CPUが出回り始めたときのパソコンは、数十万円するのが当たり前だった。日本ではPC-9801が全盛の頃だ。当時、アメリカでパーツを買って組み立てると、PCが半額以下でできるからと、個人で輸入していた人たちがいた。安いだけではなく、国内では売られていない高速CPUのマシンを手に入れていたのだ。

その頃は、PCのパーツといえばメモリやハードディスクなど周辺機器が中心で、CPUやマザーボード、ケースなどをバラバラに買って組み立てるのは、ごく一部のマニアだけだった。

しかし現在では、自分でパーツを交換したり、増設したりするのは当たり前、PCを一から組み立てるのも普通のことになってきた。

もちろん、自作PCの用途はWindowsが大多数だろうが、事情はLinuxでも変わりはない。Linuxを動かすのに特別なハードウェアは必要なく、Windowsが動作するPCで構わない。Windowsマシンを自作することは、Linux専用機を自作することにもつながるのである。

Linuxで動作する？

ところで、Linuxで使うためにPCパーツを購入する際に、一番気になるのはLinuxで動作するか否かである。ネットワークカードのなかには、パッケージに「Linux OK」とか「Linux Ready」と記載されていて、それがすぐにわかるようなものもある。

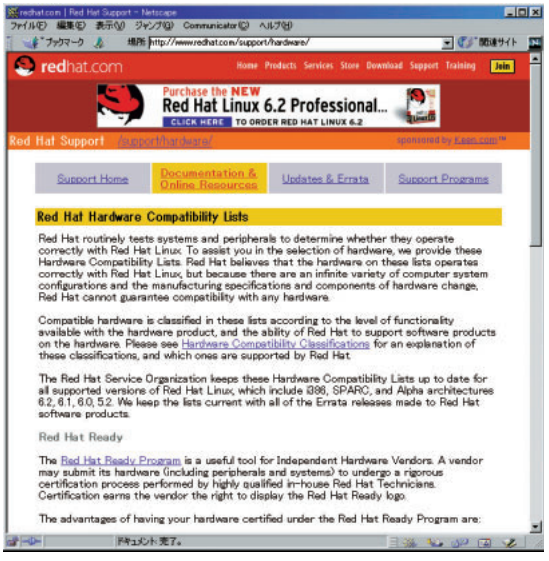
しかし、カタログやパッケージに書かれている動作OSの欄には、たいていWindows 95／98／NTやMacintoshなどは掲載されていても、Linuxの文字が書かれていることは少ない。

なぜかというと、メーカーはWindowsでは動作確認してあっても、Linuxでの動作確認を行っていないため、サポートができないというのが大きな理由だ。単一OSであるWindowsに比べ、Linuxはディストリビューションという形で数多くの種類があり、調べる手間が膨大になるからだ。

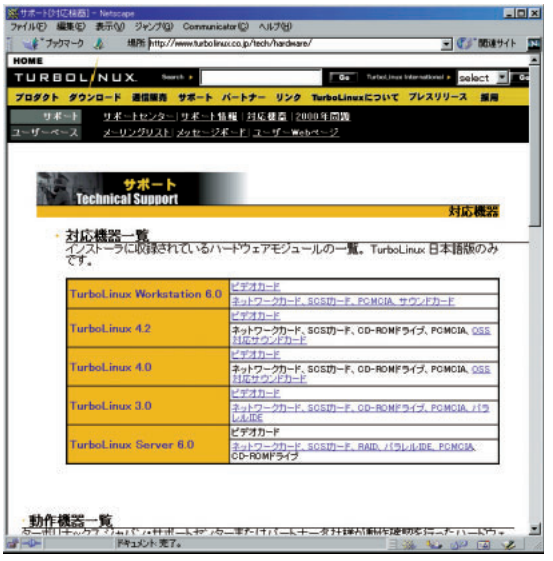
購入した製品が動作しないときに、ユーザーは電話やFAX、電子メールなどでメーカーに問い合わせることになるが、Linuxに関しては受け付けていないところがほとんどだ。結局、Linuxで使う場合には自分の責任でパーツを選択する必要がある。

WebサイトでLinuxへの対応を調べる

Linuxのプログラムは多くがオープンソースとして開発されている。そしてインターネット上で公開し、大勢の有志がテスト、デバッグを繰り返してより良いものになっていく。古い製品はともかく、ハードウェアのドライバ



画面1　Red Hat Hardware Compatibility Lists
http://www.redhat.com/support/hardware/



画面2
TurboLinux対応機器一覧
http://www.turbolinux.co.jp/tech/hardware/

は改良が引き続き行われているため、その時点で不具合があっても、最新バージョンでは修正されているかもしれない。

新しくパーツを購入する前に、Linuxでの動作状況を効率良く調べるには、以下のWebサイトを見ていくとよいだろう。なお、Linuxではカード名ではなくチップの型番が使われていることが多いので、使用しているチップも調べておこう。

■ディストリビューションのWebサイト

レッドハット、ターボリナックス、レーザーファイブといったLinuxディストリビュータは、自社のディストリビューションで動作確認ができているPCパーツのリストを、Web上で公開している（**画面1～3**）。

多くのパーツが一覧で掲載されているため、最初はこれらのWebサイトから見ていくと便利に探せるだろう。

自分が使用している（あるいは利用しようと思っている）ディストリビューションでなくても、そこで動作確認されているようなら、少なくともLinuxで使えることがわかる。そこからドライバのソースをダウンロードして、コンパイルすれば、ほかのディストリビューションであっても使用できる可能性は高い。カーネルやライブラリのバージョンの違いが原因で動作しないかもしれないが、そのときには、READMEなどのドキュメントを見てオリジナルの配布先をあたってみよう。

■Linux情報Webサイト

一般的な情報を含めLinuxに関して調べるなら、「日本のLinux情報」（http://www.linux.or.jp/）、「Linux Online!」（http://www.linux.org/）は見ておいたほうがいい。

「Linux JF (Japanese FAQ) Project」（http://www.linux.or.jp/JF/）に日本語のFAQ情報が掲載されている。Linuxが使える環境ならば、/usr/doc/（あるいは/usr/share/doc/）ディレクトリにそれらのドキュメントがすでに展開されているはずだ。manコマンドで不明な点があれば、詳細を調べるのに利用しよう。

また、全部で1万サイトへのLinux関連サイトへのリンクが掲載されているLinkLinks.com（**画面4**）には、ハードウェアのドライバへのリンクもあるので、調べるのに重宝する。

■PCパーツメーカーのWebサイト

サウンドカードSoundBlaster Live!のCreative社や、グラフィックスカードVoodooシリーズを開発している3dfx社では、Linux用のドライバを自社のWebサイトで公開している。

Windows用のドライバと違ってこういう会社はまだ少ないため、無駄になる可能性もあるが、使用するパーツが決まっているならそのメーカーのWebサイトを探してみよう。Linuxディストリビューションに付属のドライバで動作しても、ハードウェアメーカーがLinux用にチューンしたドライバを作成している可能性もあるので、調べてみる価値はあるだろう。

●

この特集では、完全自作として低予算で作る激安マシンとハイエンドLinuxマシン、省スペース型では組み立てが楽なベアボーンキット、初めて自作する人にも安心なメーカー製パーツキットの4種類のマシンを手作りし、Linuxを動かしてみた。お仕着せのマシンで満足できなくなったら、これを参考にあなたもぜひトライしていただきたい。

画面3
LASER5 Linux 動作確認リスト
http://www.laser5.co.jp/hardware/hardware_index.htm

画面4
LinuxLinks.com
http://www.linuxlinks.com/

秋葉原お買い物レポート～激安Linuxマシンへのかくも険しき道程～

文：にゃー@編集部　Text : Linux magazine

桜の花が見ごろを迎えようとしていたある日の昼下がり、その作戦計画は突然発令された。オペレーションの概略は以下のようなものだ。

「とりあえず秋葉原に行って超安いパーツを見っけてきましょう。でもって、激安PCを作りましょう」っていう感じの企画なワケよ。もちろんLinuxで動かないと意味ないんで、ソコんとこは適当にね。あっ！そうそう予算は5万円以内でヨロシク。いまどきだと、やっぱCeleron？（語尾上げ）」

コンピュータ業界に身を投じて10年。PC自作歴2回（うち1回は、電源投入時にマシンから煙が！）の私が、まさかこのような大役を仰せつかるとは。重大なる使命に燃えた私は任務遂行の決意を胸に秘め、さっそく調査にとりかかった。

激安のためのパーツ選び

とにかく予算が5万円しかないので、それを実現するためにはパーツ構成に頭をヒネらねばならない。とりあえず本誌バックナンバーやら他誌の自作特集やら、とっかえひっかえ目を通してみる。直後の感想は「ワケわから――ん！」であった。ここまでPCハードウェアの世界が変貌していようとは。頭の中では、i810、Socket370、Slot1、PPGA、FC-PGA、DRDRAMなどなど、さまざまな用語がぐるぐる回り状態である。

再度、資料に使った各誌を精読する。その成果が表1である。頭がスッキリまとまったところでパーツ選定に移ろう。低価格にこだわるなら、グラフィックス機能が統合されているチップセットが有利だろう。ビデオカードを別途購入する必要がないからだ。グラフィックス統合タイプ以外のチップセットを搭載したマザーでも、オンボードにグラフィックス機能が実装されているものがあるが、グラフィックス用に別チップが必要であるため割高なケースが多い。入手しやすさなどを加味すると、チップセットはIntel 810（i810）、SiS 530／620／630、VIA MVP4などから選択することになる。また、要不要はともかく、サウンド機能も同様の理由からオンボードであるほうがよい。

続いてCPU。上記のチップセットのうちi810とSiS620／630はIntel CPUに、SiS530とMVP4はAMDのK6シリーズに対応している。対価格性能比を考慮すると、IntelではCeleron、AMDではK6-2ということになる。純粋に価格だけを追求するなら断然K6-2である。ただし今回は、CPU自体や対応マザーボードのマーケットでの動向、CPUのアップグレードなどの将来性といった理由からCeleronを選択することにした。値ごろ感のある433MHzか466MHzが妥当なところだろう。自動的にマザーボードはSocket370（PPGA対応）のものになる。

おおっ！　理路整然と説明できたではないか。昔から「やればできる子なんですよ」と言われていただけはあるな。まとめてみよう。

●CPUはCeleron 433MHzまたは466MHz
●マザーボードはSocket370＋i810（SiS620／630でも可）のオールインワンタイプ
●ほかのパーツはとにかく安いもの

である。ターゲットは決まった（3つめがやや心もとないが）。

CPU（パッケージ*）	ソケット	内部クロック	主な対応チップセット
Intel			
Pentium Ⅲ（SECC、SECC2）	Slot1	450MHz～1GHz	Intel 440BX／810／820、SiS 620／630
Pentium Ⅲ（FC-PGA）	Socket370	500MHz～1GHz	VIA Apollo Pro133／133A
Celeron（SEP）	Slot1	266MHz～433MHz	Intel 400BX／810、SiS 620／630
Cleleron（PPGA）	Socket370	366MHz～533MHz	VIA Apollo Pro133／133A
Celeron（FC-PGA）	Socket370	566MHz、600MHz	
AMD			
Athlon（SECC）	SlotA	500MHz～950MHz	AMD 750、VIA Apollo KX133
K6-2（CPGA）	Socket7	300MHz～550MHz	VIA MVP3／4、SiS530、ALi Aladdin V

表1　CUPと対応するチップセット

＊CPUのパッケージ形状。SEP、SECC、SECC2はカートリッジ型のケースにパーツがパッケージングされており、スロットに装着する。CPGA、PPGA、FC-PGAはチップタイプでソケットに装着する。

とりあえず敵状視察なのだ

次の日、ほかの編集部員2名が秋葉原に買い出しに行くというので、とりあえず憑いて、ではなく付いていって後日の作戦決行に備えることにした。お目付け役として副編集長も同行する。本日のミッションは「各パーツの価格動向を探るとともに、格安なブツを発見した場合にはただちにそれを確保せよ」である。と同時に、秋葉原の雰囲気に慣れておくという目的もある。

実に約2年ぶりの秋葉原だ。あいにくの雨の中、我々はおもむろに表通りへと歩を進めた。ネットワークカードを大量購入するという別任務をおびた2人とは、ここでいったん別れることにする。

「見て回る前にメモリをチェックしよう」ということで、ちっちゃなパーツ屋がひしめく雑居ビル風の建物に潜入、2軒のパーツ屋で価格をチェックした。PC100 SDRAMの64Mバイトが6000円前後、128Mバイトが1万円ちょいといったところだ。金額に余裕があれば128Mバイトにしたいところなので、今は購入しない。メモリは最後の最後だ。2、3日後には、今より値下がりしている可能性もあるのだ。

いよいよ、本格的な索敵行動に移る。店の選択はお目付け役の副編集長にお任せである。指示に従って歩きつつ、キョロキョロとあたりに目を配る。思いがけない掘り出しモノが店頭に置いてあるかもしれない。ところが、気合十分にもかかわらず、これといったものもなく（パソコンショップにはフレッシュマンセールはないのだろうか？）最初の目的ショップに到着した。

平日の午後、しかも雨ということもあり、それほど混んではいない。それでも「お客様達」の放つオーラによって、店内は独特の緊張感に包まれている。店員は何やら冷ややかな雰囲気を漂わせる。負けてはならじ。決意も新たに、索敵を開始した我々であった。

母をたずねて……

まずは、マザーボード（「マザボ」と略さないでほしい）である。意外にも（？）Slot1マザーがずらりと並ぶ。Socket370は少数派のようだ。1万円以下のものと1万円〜1万5000円程度のものを中心にチェック。1万5000円前後で、チップセットにApollo Pro133/133Aを搭載したものが圧倒的に多い。この価格帯ではSlot1＋i810の組み合わせも見られる。1万円以下のものは発見できなかったので（Slot1＋Apollo Pro133で1万800円が最低価格であった）、すばやく退避行動に移ることにした。離脱途中、CPUコーナーにて情報を収集。ターゲットであるCeleron 433MHzのリテール版は9980円であった（466MHzは品切れ）。

このとき副編@お目付け役が「ありゃりゃ！？」と奇声を発した。何事かとその視線の先に目をやると……。「蘇るPentium Pro」の文字が！　一瞬（だけ）「おおっ」と思ったが、今回の企画には関係ないことに気づき適当に調子を合わせておいた（ちなみに、お値段は200MHzが3980円、180MHzで2980円）。

2軒目は家電量販店のコンピュータ館。ここは書籍も置いてあるので、弊

Column

お買い物にWebサイトは常識である

今回の特集でも、事前の価格調査をある程度行った。お買い物が済んだあとにも、秋葉原での価格動向を把握しておくために追跡調査が必要であった。最もお世話になったのが、ここで紹介する「エルミタージュ秋葉原」だ。価格テーブルが見やすいのがいい。

秋葉原に出かけられない人にも、激安への道は開かれている。オンラインショップでは、通常店舗とは異なる独自の価格設定がされていて、これが結構お安いのだ。ソフマップ以外にも、多くのショップがWebサイトでの販売を行っている。

一度チェックしてみては？

エルミタージュ秋葉原
http://www.gdm.or.jp/

ソフマップコムストア
http://www.sofmap.co.jp/

社刊行物の市場調査を兼ねて立ち寄ることにしたのだ。自作用のパーツコーナーはそれほど大きくないためあまり期待していなかったが、富士通の8.4Gバイトハードディスクが9800円で売られていた。しかも「10周年記念セール。50台限り」と書かれているではないか。基本的にこういうの（特価品とか○○台限りとか）には弱いのである。副編@お目付け役にお伺いをたてたところ、「ハードディスクは掘り出しモノが多い。それにあと千数百円出せば10Gバイトのやつが買える」とのこと。ここはグッと堪えることにする。

ハードディスクの値段が気になってきたので、次の店でチェック。4.3Gバイト 5400rpmクラスで9000円から1万円弱、10.2Gバイト 5400rpmクラスで1万1000円前後であった（この後に回ったショップでも相場は同じくらいだった）。副編の言葉が証明されたワケである。さすがはお目付け役、と感心しつつ、今度は金属性のラックに無造作に並べられたケースのチェックにかかる。標準的なATXのミドル／ミニタワーモデルでも価格にかなりの格差がある（iMac以降の傾向なのか、カラフルなものや変わった形状のものも多い）。最低価格帯は6000円～8000円といったところだ。ケースの選択にあたっては、使っている素材の良し悪し（ケースの剛性に関係してくる）や、マザーボードを取り付ける部分が取り外せるかといった作業のし易さを考慮したい。もちろん外観が気に入るかどうかも重要なファクターだ（個人的にはこれが最優先なのだ）。ケースはかさばるうえに結構重たいので、運搬用に他人カー（別働隊の片割れのマイカー。私は免許持ってないのでした）を出してもらっている今日買っておきたいのだが、ハードディスクとともに特売率の高いパーツであることを考慮して（前回の5万円PC企画でもケースは特売品をゲットしていた）この時点での購入は控えた。

続いて、ガラスケースにずらっと並んだマザーボードのコーナーへ。まずは、ざっと値段をチェック。いきなり「9480円」という金額が目に飛び込んできた。コソコソっと（別にコソコソしなくてもいいのだが）そちらに近づいて、ガラスに顔をくっつけてボール紙に手書きされているスペックを確認する。Socket370＋i810Lチップセットで、サウンドもオンボードのオールインワンタイプマザーである。もうひとつほぼ同じスペックでi810（Lなし）のものを発見。こちらは9980円であった。どちらもフォームファクターはMicro ATXである。オールインワンタイプのマザーはMicro ATXのものが多いようだ。PCIスロットを空けることで、コンパクトさと拡張性を両立するためだろう。

ここで堂々とメモをとる。マザーボードは型番しかないので覚えられない自信があるからである（イバるな！）。なぜビデオカードやサウンドカードのように、「スーパークール 810 MarkII」とか「マザボくん 2号」とかいった名前を付けないのだろう？ 不思議である。

ちなみに、このショップで発見したのは、AOpenのMX3W-LとSOYOの

	CPUソケット	チップセット	その他
AOpen MX3W-L	Socket370（PPGA）	Intel 810L＊	Micro ATXフォームファクタ、DIMM×2、PCI×3、AMR×1、USB×2、UltraDMA 33サポート
SOYO SY-7IWM	Socket370（PPGA）	Intel 810	Micro ATXフォームファクタ、DIMM×3、PCI×3、AMR×1、USB×2、UltraDMA 33／66サポート

表2　今回発見したお買い得マザーボードの主なスペック
＊i810とi810Lの違いはUltraDMAのサポートにある。i810はUltraDMA/66に対応しているが、i810LはUltraDMA/33までしかサポートしない。このほかに4MBのディスプレイキャッシュ機構を実装したi810-DC100がある（ディスプレイキャッシュは3Dのレンダリングに使用されるもので、X Windowなどの通常の2Dグラフィックスの処理には影響しない）。

写真1　発掘された激安キーボード＆マウス
安いだけに欠点もある。コードが短いのだ。おまけに、キーボードは本稿執筆中に不調に陥る始末だ。

Column

「カニ」を探せ！

本来なら、Linuxマシンを自作する際には、ビデオカードやサウンドカードなどの互換性に気をつけなければならない。今回はi810ということで、買い物の時点では、実はあまり気にしていなかった。ただ、ネットワークカードについてだけは、本文中にも登場する別働隊からひとつのアドバイスを受けていた。「カニを探せ」である。

なんだかよくわからないが、カードの基板に乗っかっているチップに「カニマーク」がプリントされているものを探せばいいらしい。詳しくは、ハードウェア特集を見ればわかる。

問題が1つ。バルク品でない限り、カードに搭載されているチップなど普通は確認できない（ケースが透明なものもあるが）。ほんの少し度胸があれば、ケースを空けてしまえばいい。度胸がない人は素直になろう。やはり店員さんに聞くのが筋だろう。「カニ」の型番は「RTL8139」である。くれぐも「このネットワークカードはカニですか？」と聞いたりしないように。

SY-7IWMの2機種（主なスペックは**表2**を参照）。価格的にもスペック的にも要望どおりなのだが、なんとなく予感がしたので（その後この予感は裏切られるのであった）購入は見送った。

1ボタン100円です

キーボードとかも見ておこうと2階の売り場に移動しようとしたとき、別働隊から連絡が入った。「990円のキーボードを発見した」というのだ。これは購入せねばなるまいと決意し、店名と売り場を確認しておく。最初の捕獲ターゲットは決まったも同然である。マウスも見たかったので、一応この店の売り場もチェックすることにした。ふと目を留めると、「952円」の値札が付いたキーボードが何気なく（本当に何気なく）積まれているではないか！「（約）40円勝った」その瞬間の正直な気持ちである。

こうなるとほとんど主婦感覚だ。スーパーで野菜や鮮魚の特売品の値段を気にする主婦といっしょである。たて続けに激安マウスも発見された。PS2の3ボタンマウスで、こちらは285円であった。消費税を入れても1ボタンあたり100円ではないか。文字どおり「桁違い」に安い。もちろんこの2品は即座に捕獲・拘束（購入）した。

結局、このあと4店舗ほどを見て回った。いくつかのパーツの価格動向をお知らせしておこう。

激安品をゲットしたキーボードとマウスだが、これは例外としても、デザインや価格にかなりのバラツキがある。安心感のある有名メーカー製品は、キーボード、マウスとも4000円〜8000円ほどだ。CD-ROMドライブは、48倍速、50倍速クラスが多く、価格は6000円前後が標準的。バルク品ではないがケースに「CYBER DRIVE」とある製品（48倍速）が2、3のショップで叩き売り状態だったので、一番安かったショップで購入した。3980円也である。

量販店あなどれず

いったん捕獲品を車に収容してから、ケースの最終探索に向かった。さきほど索敵中に4999円の特売品（ATXミドルタワー。露天積み）を確認していたので、この価格が基準となる。まずは未探索であった（パーツ売り場が8階にありめんどくさかった）秋葉原最大級の売り場面積を誇るショップへ潜入した。エスカレータを延々と登っていく。パーツ売り場に到着してエスカレータを降りたところで、3980円のATXミドルタワーケースを発見。しかも、こちらの弱点を見透かすかのように「限定50台限り」と大書されているではないか。これで、ほぼ決定といった感じだ。電源も230Wとあるので問題ない（あとでわかったのだが、電源は実は250Wであった。なにゆえ230Wと書いてあったのかは不明である）。なんだかあっけないが、見つかるときはこんなものなのだろう。

一応、ほかのケースも見て回る。Micro ATXのおしゃれで安価な（5480円）ケースがあった。電源も200Wで、造りもしっかりしている。1500円しか違わないし、外観はこちらのほうが断然イカしている。これにはかなりココロをひかれたのだが……。「やっぱりサンキュッパでしょ」というお目付け役の一言で限定50台限りのほうに決定したのであった（泣）。でもまあ、ここでMicro ATXのケースを買ってしまうと、マザーボードの選択肢も狭まってしまう。作戦上、合理的な判断であるといえよう。さすが副編@お目付け役兼作戦参謀である（とヨイショしておこう）。

実は途中のショップで展示品限り2000円という掘り出しモノもあったのだが、領収書は出せません（ナゼだ！）とのことで、泣く泣くあきらめたことをご報告しておく。運が良ければ、こういう脱税的限定特価品に遭遇することもあるということだ。

この日の戦果を以下にまとめる。

●日本語112キーボード：952円
●3ボタンマウス：485円
●CD-ROMドライブ：3980円
●ATXミドルタワーケース：3980円

ということで、合計9397円、消費税込みで9867円であった。上々の滑り出しであるといえよう。

写真2　CD-ROMドライブは快調だ
音がかなり大きいのは高回転で動作している証拠なのか？　音以外は特に問題なし。

写真3　裸にしたATXケース
結構ゆとりがあって作業しやすかった。剛性も、値段のわりにしっかりしている。今回捕獲した中でもお買い得の一品であった。

この日の作戦で気づいたのは、「雨の日のお買い物は避よう」ということと、「量販店は結構あなどれないので必ず立ち寄ってみよう」ということの2つである。

第2次秋葉原遠征

翌々日、ふたたび秋葉原に降り立った。今回は単独行動だ。「責任重大である」と自らに言い聞かせながら作戦行動に移ったのであったのだが……。

ここで読者の皆様にお知らせしておかなければならないだろう。結果として目標の5万円を大幅にオーバーしてしまったのである。お目付け役もなく、自由を得たことで、秋葉原中を気ままに歩き回った私は確かに浮かれていた。買い物という行為自体もなにやらヒトのココロをウキウキさせるものだ（ですよね？）。そのうち、脳内にドーパミンやらエンドルフィンやらの快楽物質が分泌されたのであろうか、まさにランナーズハイならぬ「ショッパーズハイ」状態に陥ってしまったのである。それは、うららかな春の日差しのせいだったのかもしれない。言い訳していてもしょうがないので、この日の経過を追っていこう。

とりあえず価格動向を調べようと、前回行っていなかったショップを中心に見て回わることにした。最初はわりと冷静だったのだが、ついついヒートアップしてしまい、気がつくとイッキに7店舗を踏破していた（時間にして2時間弱である）。しかもメモをとってなかったせいで、どの店で何が安かったのかハッキリ覚えていないというありさまだ。さらには1480円のLANカード（100Base-TX対応では最安値圏）と1980円のフロッピーディスクドライブ（2モードのバルク品）をいつの間にか購入していた（ショッパーズハイの初期症状を示しているといえよう）。さいわいにも、これらは値段も安く、この時点では作戦の大きな破綻を意味するものではなかった。

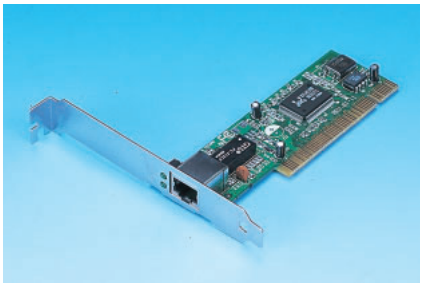


写真4　得体の知れないNIC
もちろん搭載チップは「カニ」である。

物欲という名の病

態勢を立て直すためには、メモをとって価格を正確に比較することが必須であると反省し、作戦を再開。先ほど回った7つのショップを逆順にたどっていく。価格をメモして比較されるのはショップ側としては望ましくないことなのだろうが（実際に禁じている店もある）、「仕事用の記憶域は8ビットしかない」と自覚している身にとってメモは必須だったのである。

一通りメモをとったところで喫茶店にて「ひとり作戦会議」を開く。マザーボードに関してめぼしい発見はなかった。やはりSlot1マザーが主力である。Socket370では、SiS630チップセット、サウンドもオンボードのオールインワンタイプで1万1800円のものを発見したくらいだ。マザーボードは前回発見したAOpenのMX3W-Lに決定する。さっそく喫茶店を出て、くだんのショップに向かい、これを購入した。

CPUおよびメモリについてもメモしまくり、数百円の違いにこだわる主婦感覚で臨んだ（はずだった）。ただ、433MHzのCeleronは扱っていない店も多く、メモリも品切れというケースに多く遭遇した。そのうちの数店で「入荷待ち」とか「品切れ」とか書いて価格表の上に貼ってある紙をペロリンと

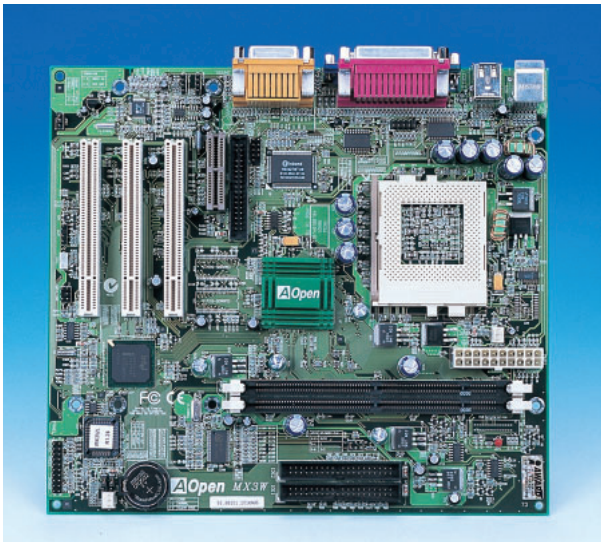


写真5　MX3W-Lの全景
ショップでもらった独自の設定マニュアルによると、PCIはIRQとI/Oポート競合の恐れがあるためスロット3から使用するほうがよいらしい。

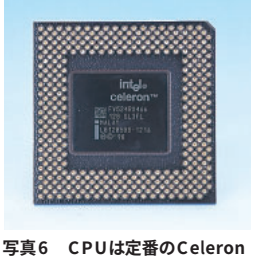


写真6　CPUは定番のCeleron
CPUクーラーが付いてくるので、リテール版がお勧め。価格もバルクものとそれほど変らない。現在なおも値下がり中である。

めくって強引に価格を確認したところ、やはり安かった。安い店から売れていくということなのだろう。

CPUはCeleron 466MHzに決定し、最安値（9499円）だったショップへと急いだ。

思えばこの時点が「ショッパーズハイ」の絶頂だったのだろう、勢いでCPUといっしょに128Mバイトのメモリも購入してしまった。このとき、もとからやや貧弱な私の暗算機能は完全に停止していた。あとの祭り感もあるが、この時点での金額を確認してみよう。

●LANカード：1480円
●FDD：1980円
●マザーボード：9480円
●CPU：9499円
●メモリ：1万799円
●小計：3万3238円

前回購入分とあわせて消費税込みで4万4767円である。作戦完遂のためには、「税込み5233円」のハードディスクしか買えない状態であったのだ。しかし、この時点においても「物欲の暴走状態」は収まっていなかった。当初の作戦目的は意識の彼方へと追いやられていたのだ。

ハードディスクを見て回っているうちに、私はひとつのことに気づいていた。前述したように、10.2Gバイト 5400 rpmクラスで1万1000円前後が相場である。1万円以下だと8.4Gバイト以下になる。千円の差で1.8Gバイトもの増分である。大容量のメモリやハードディスクが贅沢品であった頃を知る者として、この差は大きいと感じざるを得ない。頭の中では「千円で1.8Gバイトのディスク領域がお手元に」というキャッチコピーが浮かんでいた。

まずいことに、「10.2Gバイト、7200rpm、平均シークタイム8.2ms」というスペック値を誇るハードディスクドライブを発見。しかも名前が「BARRACUDA ATA II」(型番はST310210A。Seagate社製)。なんだかスゴそうだ。「特価品」や「台数限定品」と同じくカタログスペックの数値にも弱い私は、即決、即購入した。1万2980円であった。

作戦終了

以上で作戦行動は一応の終結をみた。最終的な購入金額は5万8168円（消費税込み）である。当初の目標金額から8000円のオーバーだ。総括してみよう。

●メモリは64Mバイトで我慢するべきでした（約5000円のマイナス効果）
●ハードディスクは8.4Gバイトで我慢するべきでした（約4000円のマイナス効果）

なーんだ。失敗したのはメモリとハードディスクだけではないか。ここにさえ気をつけていれば作戦は大成功だったのだ。と自分をごまかしつつ、お買い物レポートを終わることにする。最後に教訓を2つ。

●数百円の違いにこだわるのはやめましょう。価格調査に夢中になるあまり、冷静さを失う恐れがあります。
●冷静でいられる同伴者を連れていきましょう。ショッパーズハイに陥りそうなあなたを救ってくれるかもしれません。ただし、人選を誤ると余計に物欲を刺激されて、さらに状況が悪化する可能性があるので注意しましょう。

さあ！組み立てだ！

組み立てに関しては、これといった問題は発生しなかった。撮影用に仮組みしたマシンからハードディスクドライブを外そうとしたときに、なぜだか

写真7　ハイスペックを誇る「BARRACUDA」
このクラスではMaxtorの「Diamond Max Plus 40」と双璧をなす激速ハードディスクだ。

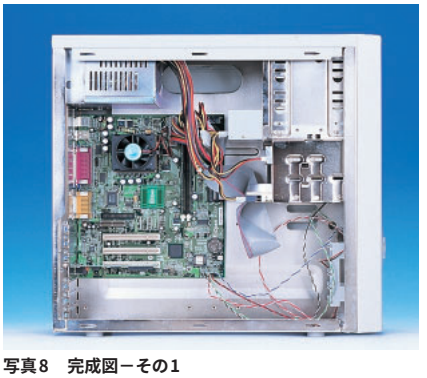


写真8　完成図－その1
実は撮影時の配線では、パワーランプもアクセスランプも点灯しないのであった。

写真9　完成図－その2
背面である。ショップの勧めに従って、NICはPCIスロット3に装着した。

写真10　完成図－その3
前面パネルにある「HUB」の文字が謎めいている。

ネジが抜けなかったことぐらいである。ラジオペンチで強引に引き抜いたのだが、まあこれくらいはよくあることだろう（たぶん）。

ディスプレイを接続して電源オン。BIOSのセットアッププログラムを立ち上げる。IDEドライブの認識などに問題はないようだ。CD-ROMからのブートが有効になっていることを確認して、いったん電源をオフにする。次はいよいよLinuxのインストールである。

インストールが遅い？

今回はテストも兼ねて、Vine Linux 2.0 CR Offical製品版を使用してみた。CD-ROMをセットして再び電源を投入する。インストーラが無事起動。［Xの設定］画面でビデオカードが自動検出されなかった以外は、特に問題なく設定できた（Xの設定はスキップした）。すべての設定を終え、インストールが開始される。まずは、ディスクのフォーマットである。

これが遅いんである。HDDへのアクセスランプも点滅しない（ここでやっと気づいたのだが、電源ランプも点灯していなかった）。なにやら動作していそうな音はしているが、フリーズしているような気配もある。

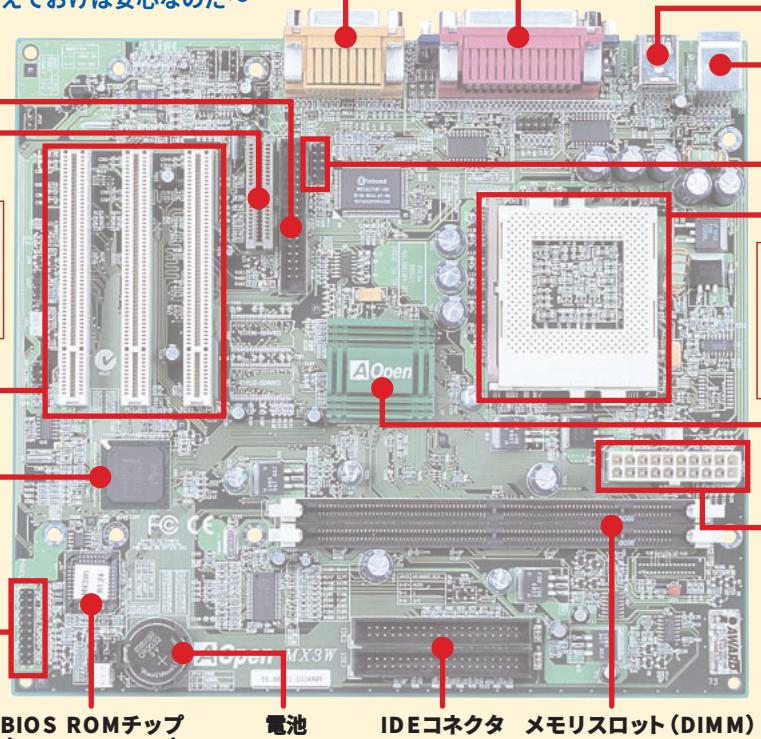
意を決して電源を「プチッ」と切る。マザーボードのマニュアルを再確認して、LEDケーブルの配線を見直す。挿し込む場所は合っていたが、極性が逆になっていた（皆さんも気をつけてネ）。Windows 98の起動ディスクを使ってFDDから起動し、fdiskとformatを試す。OKである。ドライブ自体に問題はないのだ。

ひと手間かけて金かけない

再度インストールに挑戦。今回は電源ランプもHDDへのアクセスランプもちゃんと機能している。気長に待つことにした。25分ほどでフォーマットが完了。「不良ブロックの検査」をオンにしていたので時間がかかっただけっだようだ。さっきはフォーマット中に電源を切ってしまったことになる（無事でよかった）。インストールも順調に終了。それでも50分ほどかかっている。なんだか遅いな。

マザーボードの基本を押さえておこう!

～各部の名称と形、配置を覚えておけば安心なのだ～



ジャンパによる設定はわかりずらくて面倒だが、最近のマザーボードはBIOSでほとんどの設定が可能になっている。完全にジャンパフリーなマザーボードも増えてきている。

基本的にどのマザーボードでも、CPUとメモリの取り付け部分以外は大きな形状の違いはない。ただし、CPUではSlot1とSlotA、メモリではDIMMとRIMMの形状が似ているので注意。

＊写真はAOpen MX3W-L

画面1　インストーラの［Xの設定］画面
i810では、残念ながら“Unable to detect video card”という結果になる。［Xの設定をスキップ］を選んで、インストールを進めよう。

画面2　kbdconfig
1つ下の［jp106_Ctrl_CAPS］を選択すると、左側のCtrlキーとCaps Lockキーが入れ替わったキーバインドが設定される。

i810チップセットでは、GMCH（Graphics Memory Controller Hub）と呼ばれるメインチップ82810にグラフィックス機能が組み込まれている（Direct AGP）。このため、インストーラでビデオカードが自動検出されなかったのだ。X Window System（XFree86）を使うには、ひと手間かけなくてはならない（詳細は58ページからのベアボーンキットの解説を参照）。めんどくさいが、激安マシン実現のためには必要な手間なのである。

/etc/conf.modulesにagpgartモジュールのエイリアスが設定されていることを確認。Vine 2.0ではデフォルトで設定が有効になっているようだ。続いてXconfiguratorでXFree86で使用するグラフィックス環境の設定を行う。起動後の画面で［了解］を選択すると自動検出が行われる。82810 CGC（Chipset Graphics Controller）を検出したことを示すメッセージが表示されるが、なぜか［Xサーバ］の項目は空欄になっている。とりあえず［了解］を選択してみる。悲しいことにXconfiguratorが終了して「Xサーバがない」ことを示すメッセージが表示された。

調査の結果、i810のグラフィックス機能に対応したXサーバはXFree86 3.3.6のSVGAサーバであることが判明。どうやらこれがインストールされていないらしい。rpmコマンドで確認する。

```
# rpm -q XFree86-SVGA
```

やはりインストールされていないというメッセージが表示された。再びrpmコマンドを使って、必要なパッケージをインストール。

```
# rpm -i /mnt/cdrom/Vine/RPMS/XFree86-SVGA-3.3.6-13v13.i386.rpm
```

これがそのコマンドラインである。長いな（実際にはこれで1行）。パッケージのインストールはほんの1、2秒で終わった。なんだかウソくさいが大丈夫か？

もう一度Xconfiguratorを起動する。あいかわらず［Xサーバ］の項目は空のままだが、今度は次の［モニタセットアップ］の画面に切り替わった。

今回使用したモニタはEIZOのFlexScan E57Tである。選択肢の中に該当するものがないので［カスタム］を選択して次に進む。いろいろ試行錯誤した結果、モニタのタイプは［1280x1024 @74Hz 使用可能モニタ］を選択、垂直周波数範囲を［50-90］、ビデオメモリのサイズを4Mバイト、解像度は1024×768ドットの16ビットモードに指定すると、うまく設定できることがわかった。カスタムの設定ではモニタのマニュアルなどを参考に、少し控えめの値を指定するほうがよいようだ。なお、クロックチップは指定せず、適正解像度とクロックの検出はスキップした。

この時点でXFree86を一度立ち上げてみる。成功！　ようやくX Window Systemが使えるようになった（やれやれ）。

このあとXF86Setupを使って、モニタのマニュアルに記載されている周波数範囲を正確に指定し、解像度も1280×1024にして再起動してみたが、問題なく立ち上げることができた。

こまごました設定

めでたくXが使えるようになったのだが、まだ問題が残っていたのである。まずはキーボード。インストール時に［Japanese 106］を指定したはずなのに、アスタリスクをタイプするとダブルクォーテーションが入力されるのだ。明らかに101キーボードのレイアウトになっている。

コンソールのキーボード設定は簡単。次のコマンドを実行し、kbdconfigを起動する。

```
# kbdconfig
```

表示される一覧から［jp-106］または［jp-106-Ctrl-CAPS］を選択すればOK。

なお、VineのWebサイト（http://vinlinux.org/）で確認したところ、この問題はすでに障害として報告されて

いた。すばやい対応である。インストーラの改良を期待しよう。

Xのキーボード設定も変更しておこう。任意のエディタ（viなど）で、/etc/X11/XF86Configの「Section "keyboard"」の該当行を以下のように変更する。

```
XkbModel "jp106"
XkbLayout "jp"
```

あるいは、次の1行を同じセクションに追加してもよい。

```
XkbDisable
```

変更が済んだところで、XFree86を起動して確認する（無事成功）。

マニュアルには、XkbDisableの設定はデフォルトであると記載されている。なにゆえ変更されたのか？　調査したところ、答えはXF86Setupにあることが判明。XF86Setupで設定した際に、デフォルトの設定が上書きされていたのであった。

UltraDMAを有効にせよ

ひととおり設定を終え、気持ち良くLinuxライフを満喫しようとしていた私に（なにしろ、これまでLinuxマシンとして使っていたのは、90MHzの初代Pentium搭載機種だったのだ）、次なる試練が待ちうけていた。

ある筋からの情報によると、hdparmというコマンドのテストモードで、ディスクドライブのお手軽なベンチマークが可能であるという。苦労して組み上げたマシンの性能を知りたかった私は、さっそくテストを実行することにした。コマンドラインは次のとおりである。

```
# hdparm -t /dev/hda
```

結果は4.8Mバイト／秒という惨憺たるものであった。激速ハードディスクであるBARRACUDA ATA Ⅱの実力は、こんなものではないはずだ。納得がいかない。

徹底調査により、初期設定状態ではUltarDMAが有効にならないということがわかった。名前に「ウルトラ」が付くぐらいだから、やはり有効にしたいのが人情。この設定にも、やはりhdparmを使う。

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

これで有効になるはずだ。上記のコマンドラインのうち“d1”の部分は、UltraDMAモードを有効にするオプションである。マイマシンはi810Lマザーボードなので、UltraDMA/33までしかサポートされていないが、ともかくウルトラの仲間入りができたわけだ。

再びhdparmを使ってテストしてみる。結果は良好。5Mバイト／秒弱だった数値が約13Mバイト／秒までアップしている。この設定は、再起動するとリセットされてしまう。それでは、あまりにも悲しいので起動時に確実に実行されるスクリプトである/etc/rc.d/rc.localに上記のコマンドラインを追加しておこう。

いまから考えると、UltraDMA/33しかサポートされていないマザーボードに激速ハードディスクというのは、もったいなかった気もするが（買うと

写真11　82801AB
i810チップセットのI/Oコントローラハブ。問題のUltraDMAとサウンド機能を制御するチップである。

Column

hdparmの甘い罠

hdparmは本文中にもあるように、お手軽にベンチマークを体験できる楽しいコマンドだが、本来はIDEディスクドライブにパラメータを設定するためのコマンドであるらしい。今回紹介したUltraDMAを有効にするオプションもそのひとつなのだ。しかし、便利で賢いこのコマンドには、もうひとつの隠された顔があったんである。

実は調査の過程で、明示的にUltraDMAの動作モードを設定できるオプションも判明していた。

```
# hdparm -d1 -X66 /dev/hda
# hdparm -d1 -X68 /dev/hda
```

これがそのコマンドラインだ。“-X66”と“-X68”がUltraDMAのモード指定であることは、すぐにわかる。では、どちらがUltra DMA/33モードで、どちらがUltraDMA/66モードかというと、“-X68”のほうなのである。66モードなのに、オプションは68とはどういうことか！　実にまぎらわしい！

怒ったフリをしているが、本当の問題はそこにあるのではない。まぎらわしいだけならまだいい。正しく覚えればいいだけだ。ただ、生来の求道者である私は、『“-X33”こそ正しいオプションであるに違いない」と考え、考えただけでなく実行したのである。その結果は……。ハードディスクが固まってしまったんである。

というわけで、「hdparmで変なパラメータを指定しちゃダメよ」というコラムはこれにて終了する。

きに気づくべきである！）、一応ハードディスクに関しては、これで良しとしておこう。

サウンドは結構たいへん

続いてはサウンド。ウィンドウの操作時に妙な効果音が鳴るのはイヤなものだが、MP3やCDで音楽を聞けないのはもっとくやしい。またもや徹底調査が必要だ。

「i810に統合されているサウンド機能は、AC'97と呼ばれるIntelが提唱する規格に基づいて実装されている」ということまでは、すぐにわかったのだが、残念ながらLinuxではサポートされていないらしい。しかし、最新のVine 2.0であれば……と思い、試してみることにした。

編集部にコロがっていたスピーカを取り付けて音楽CDを再生してみたが、音が出ない。CDプレーソフトはCDを認識してきちんと動いている。やはりサウンド機能が有効になっていない。

Vineのマニュアルを確認すると、サウンドの設定はsndconfigで行うとある。実行すると確かに「82810 AC '97 Audio」であると認識してはいるが、「これはサポートされてませんよ」という旨のメッセージが表示される。

いったんはあきらめかけた私ではあったが、ベアボーンキットのパートの執筆をお願いしていた山岸氏から朗報がもたらされた。Advanced Linux Sound Architecture（ALSA）というプロジェクトで、独自のLinuxのサウンドドライバが開発されているというのだ。さっそくALSAのWebサイト（http://alsa.linux.or.jp/）を確認する。トップページのリリース速報に確かに「AC'97」の文字があるではないか。いそいそと最新安定版のドライバ（4月6日付。バージョン0.5.7）をダウンロードした。

基本的に、こういう単独で配布されているドライバは、tarボールを展開して、コンパイルして、設定するという結構面倒なインストール手順を踏まなければならない。bzip2のコマンド構文でいきなり壁にぶつかったが、付属のドキュメントを解読（英語なのだ）しながら、なんとか終えることができた。やればできるものである。

今回確認できたAC'97（i810）用の設定を紹介しておこう。ALSAのドライバはモジュール化されているので、設定は/etc/conf.modulesで行う。まず、**リスト1**に示す行をconf.modulesに追加する。

リスト中の「OSS/Free」以降はLinuxの標準的なサウンドドライバのひとつであるOSS/Freeをエミュレートするための設定である。設定が完了したら、

```
# modprobe snd-card-intel8x0
```

を実行すれば必要なドライバモジュールがロードされる。ロードの確認はlsmodコマンドで行う。ふだんと違って、ずらずらとモジュールが表示されるのでなんだか気持ちいい。

注意点が1つ。ロード直後のデフォルト状態では、ミュートが有効になっているのである。「インストールが完了したはずなのに、音が鳴らない。なぜだ」とあせる（体験談）前に、OSS対応のサウンドツール（GMIXなど）を使ってミュートをオフしよう。

▼リスト1　conf.modulesでのサウンドドライバの設定

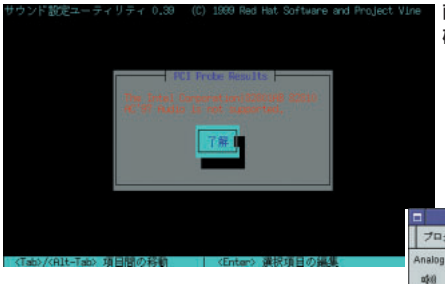
```
# ALSAドライバの設定
alias char-major-116 snd
alias snd-card-0 snd-card-intel8x0
# OSS/Free用の設定
alias char-major-14 soundcore
alias sound-slot-0 snd-card-0
alias sound-service-0-0 snd-mixer-oss
alias sound-service-0-1 snd-seq-oss
alias sound-service-0-3 snd-pcm-oss
alias sound-service-0-8 snd-seq-oss
alias sound-service-0-12 snd-pcm-oss
```

Linuxマシン自作のススメ

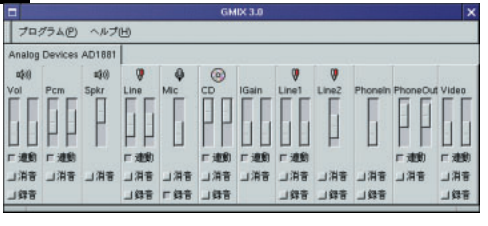
あれこれと設定した結果、どうにかひととおりの機能が使えるようになった。今度こそ、Linuxライフを満喫できそうである。

パーツ選びから始めて、Linux用自作マシンを完成させるためには、かなり苦労した。しかし、必要な作業を続けるうちに得られた知識や情報も多かった。新米Linuxerである私にも「ウデをあげたな」という実感がある。

というワケで、自作経験のない皆さんにもLinuxマシンの自作を激しくお勧めする。



画面3　sndconfigの実行結果
確かに82801ABを認識してはいるが、サポート外だそうだ。実にあっさりしている。



画面4　GMIXのメインパネル
各チャンネルに［消音］とあるのがミュートの設定。これをオフすれば待望のサウンド機能が有効になる。［Analog Device AD1881］とあるのは、AC'97のアナログサウンド機能であることを表している。

簡単！お手軽！ベアボーンキットで作るLinuxマシン

文：山岸典将　*Text : Norimasa Yamagishi*

すべてのパーツを自分で集めて自作する自信はあまりない、または手間はかけたくないが、CPUやハードディスクくらいは自分の要求を満たすものにしたい。そういう方は、ベアボーンという選択肢を検討してみてはどうだろうか。

ベアボーンキットとは

ベアボーンキットとは、一般にCPU、ハードディスク、メモリは別に自分で購入し、追加する形式のキットだ（**図1**）。ケース、電源、FDD、CD-ROMドライブ、ビデオチップ、キーボード、マウスなどのパーツはあらかじめキットの中に入っているものがほとんどだ。サウンドチップやモデム、ネットワークインターフェイスなどが、あらかじめ組み込まれている場合もある。特に、最近は、ビデオ、サウンドといった統合機能を備えたオールインワンタイプのマザーボードが使用されることが多く、低価格でそれなりに自由度を持ってマシンを組めるようになっている。そのため、通常のタワーケースよりも小さく省スペースなケースになっているのも特徴だ。また、追加するパーツを組み込むだけで済むので、簡単に組み上げることもできる。

もっとも、ベアボーンという名で販売されているすべてのマシンが上記のような構成だとは限らない。実際に購入する際には、なにが含まれ、なにが含まれないのかを確認してほしい。

今回は、まずベアボーンキットを選択するポイントについて考え、実際にLinuxをインストールし、使えるようにするまでの過程について見ていこう。

キット選択のポイント

ベアボーンキットを選択するときに注意する点は、主に「Linuxで利用できるか」と「拡張性は必要か」の2つだ。

「Linuxで利用できるか」という点で特に気をつけなくてはいけないのが、ビデオ、サウンド、ネットワーク、モデムである。ハードウェアを使うには、個々のハードウェアに対応した固有のドライバが必要となる。キーボードやハードディスク、マウスといったハードウェアに関しては、あらかじめ用意されたドライバで対応できる場合がほとんどだ。しかし、高解像度のビデオやサウンドといったデバイスに関しては、それぞれのハードウェアごとに命令が異なっているため、専用のドライバが必要になる。

ベアボーンキットは、すべてのパーツを自分で選んで組み上げる場合に比べると、あらかじめ用意されているハードウェアが多いため、選択には制約がある。また、拡張性が低くパーツの変更ができないことも多い。キットの選択は慎重に行いたい。

もっとも、ハードウェアの情報を探し、動作させていくという過程も結構楽しいものだ。そして、その過程で、新たな知識を得ることができるだろう。

利用できるかどうかを確認する一番簡単な方法は、各ディストリビューションの「ハードウェアの動作確認リスト」を参照することだ。といっても、大手メーカー製のマシンやパーツでさえも、すべてについては動作確認されていないというのが実情だ。そこで、実際にはマシンで利用されているハードウェアを調べ、それぞれのハードウ

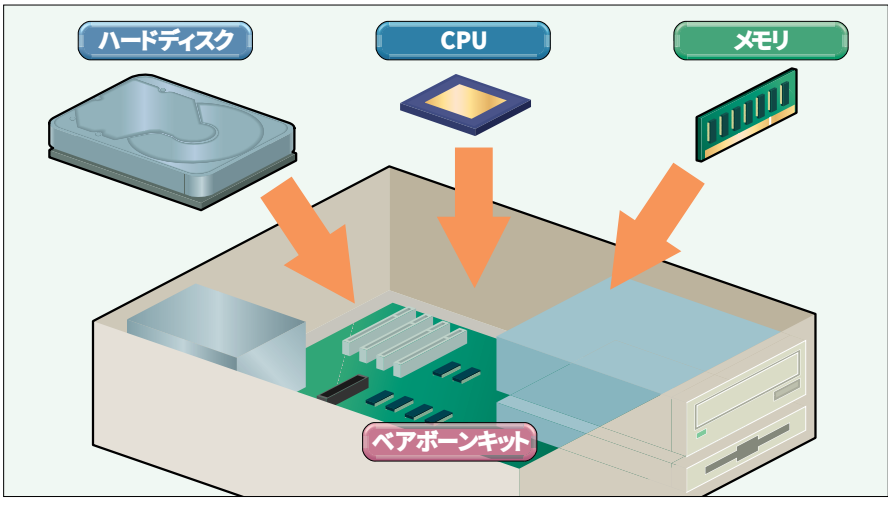


図1
ベアボーンキットにCPU、メモリ、ハードディスクを加えてマシンが完成する。

ェアが対応しているかを調べることになるだろう。

特に、ベアボーンマシンでは、市販のカードを利用せず、それぞれの機能をマザーボードにオンボードで搭載している場合が多いので、カード名でも調べられないことが多い。その場合は利用しているチップの名前を調べる必要が出てくる。

ただ、リストに対応が記載されていなくても、ディストリビューター自身が確認作業を行っていないだけということもある。そのようなときは、さらに調査の範囲を広げてみよう。まず、そのチップセットのメーカー、そして、そのメーカー以外にドライバの開発者がいないかを調べてみるのだ。

ちなみに、以前ドライバがカーネルに組み込まれていた頃は、新たなハードウェアを追加するのにカーネルの再構築が必要だった。しかし、今ではドライバのモジュール化が進んだため、モジュールを追加するだけで簡単に新しいハードウェアを使える場合が多い。この点では、状況は好転しているといえるだろう。

最新のカーネルやモジュールをインストールすることができれば、そのディストリビューションではサポートされていないハードウェアを利用できる場合もある。選択の幅はぐっと広がるはずだ。

●ビデオ

通常のVGAの画面であれば表示には問題ないので、サーバ用途のためX Window Systemは必要ないということであれば、特に気にする必要もないだろう。最新のXFree86 4.0では、逆に古いチップがサポートされていないといったこともあるくらいだ。

もっとも、現実にはXを使いたいという人がほとんどだろう。そこで、XFree86がサポートしているビデオチップを選ぶことが必要になってくる。

以前は、新しいビデオチップや、メーカーが情報を公開していないチップは利用できない、といったことがあった。しかし最近では、最新チップのサポートもかなり速くなり、メーカー自身がLinuxへの対応に力を入れはじめているので、メーカーの情報公開や、ドライバそのものの公開も増えてきている。

ディストリビューション付属のXFree86のバージョンは若干古いこともあるので、自分でXFree86を入れなおさないといけない場合もあるが、ディストリビューションもかなり頻繁にアップデートされるので、それほど問題にはならないと思われる。実際、4月以降にリリースされたディストリビューションはすべて、XFree86 3系列では最新の3.3.6が標準でインストールされるようになっている。

なお、最近ベアボーンキットでよく使われている、インテルのi810チップセットはXFree86-3.3.6でサポートされるようになった。最新のディストリビューション以外を使うのであれば、XFree86-3.3.6を別途インストールする必要がある。

キットが使用しているビデオチップに関しては、Webページやカタログなどに記載されているはずだ。統合機能タイプのマザーボードを使用している場合、記載されているのがマザーボードメーカーのみということもあるが、その場合はマザーボードメーカーのWebサイトなどを見て、どのようなチップを使っているかを調べよう。

XFree86のサポート状況についてはXFree86 ProjectのWebページ（http://www.xfree86.org/。**画面2**）を参照

画面1
ターボリナックスジャパンのWebサイトのサポートページ。ハードウェアとそれに対応するモジュールも掲載されているので、どのモジュールを使えばよいのかわかるので便利だ。

画面2
XFree86 ProjectのWebページでは、最新の情報やXFree 86そのものを入手することができる。

するとよいだろう。また、XFree86でサポートしていなくても、商用のAccelarated-Xなどでサポートしている可能性もある。商用だけあって、インストーラもよくできているので、XFree86のみのバージョンアップに自信がないようであれば検討に値するだろう。

●サウンド

サウンドチップに関しては、使われているチップが、Webページやカタログなどにも記載されていないことがある。このような場合、実機を見てチップを確認するか、メーカーに問い合わせると間違いがない。

ドライバは、やはりメーカー以外で開発されているものが多く、商用のドライバではOSS/Linux、フリーのものではOSS/FreeやALSA（Advanced Linux Sound Architecture）のドライバがメジャーだ。

OSS/Freeは、OSS/Linuxの一部をフリーにしたものが独自に発展していると思えばよいだろう。Linuxのカーネルにも OSS/Freeが組み込まれている。ALSAはOSSとの互換性を維持しながら、さらに高性能・高機能なものを目指している。フリーのドライバが性能的に商用のものに劣るかというと、ほとんどその差異はないといってよいだろう。実際、ALSAのドライバはかなり評価が高い。ただし、商用のものはやはりハードウェアメーカーからの情報供給が受けやすいようで、サポートするハードウェアの数は多い。

OSS/Linux	http://www.opensound.com/
OSS/Free	http://www.linux.org.uk/OSS/
ALSA	http://alsa.linux.or.jp/

表1　各サウンドドライバのWebページ

市販されているディストリビューションのパッケージにはOSS/Linuxが付属しているものもある。また、無償で配布されているものでは、カーネルに含まれるOSS/Freeを利用しているディストリビューションもある（最近ではALSAを利用していることもある）。それぞれのドライバについては、Webページで情報が公開されているので参照してほしい（**表1**）。

●ネットワークインターフェイス

ネットワークに関しては、カタログ等に「10/100BASE-T」などと書かれているだけで、どのようなチップが使われているかはほとんど記載されていない。そのため、やはりチップの確認作業が必要になる。

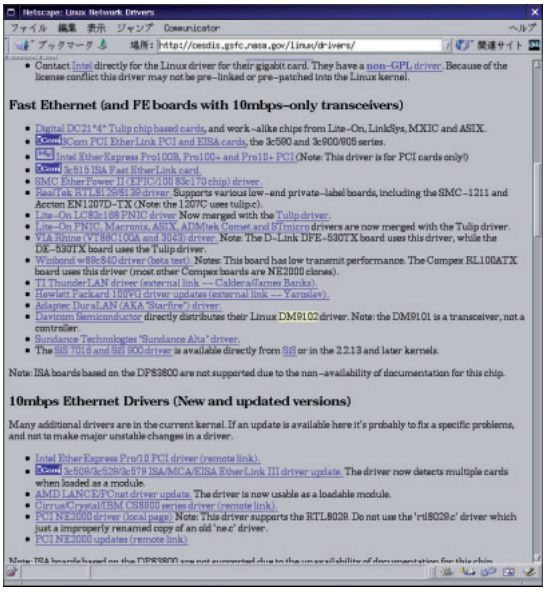
すでに購入してしまっているのならば、Windowsを一度インストールし、コントロールパネルの［システム］―［デバイスマネージャ］で調べるという方法もある。Windowsで使用することを想定している場合がほとんどなので、Windows用のドライバは添付されているはずだ（Linux用のドライバが添付されていることはほとんどないが……）。

Linux用のドライバが用意されているかを調べるには、Linux at CESDIS（http://cesdis.gsfc.nasa.gov/linux/。**画面4**）のWebページを参照するとよいだろう。チップ名と対応したドライバへのリンクがある。

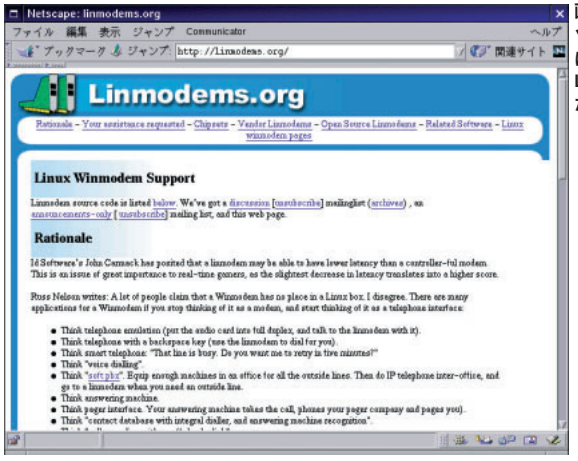
●モデム

モデムはベアボーンキットに限らず

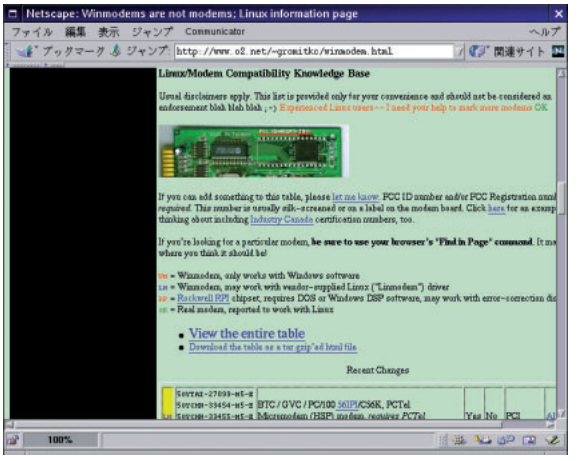
画面3
ALSAの日本語版Webページ。ALSAはOSSとの互換性を持つLinux用のサウンドドライバの開発を推進するプロジェクトである。最新のドライバは、ALSA driver 0.5.7（4月6日リリース）。



画面4
Linux at CESDISのWebページにはさまざまなネットワークハードウェアのドライバの情報が記載されている。



画面5
ソフトウェアモデムに関する情報はLinmodems.orgからたどるとよいだろう。



画面6
「Winmodems are not modems」には各チップごとに、ユーザーによるLinuxでの動作報告がまとめられている。

市販のマシンでも、内蔵している場合には問題になることが多いデバイスだ。内蔵モデムの場合、ソフトウェアモデムを利用しているためLinuxでは利用できないことが多い。さらに、カタログ等にその仕様が記載されていることはほとんどない。はっきり言ってしまえば、現状ではベアボーンキット付属のモデムについては最初から利用しないつもりでいたほうがよいだろう。外付けモデムをシリアルコネクタに接続して使うのが確実で、もし、内蔵モデムが利用できたらラッキーくらいに考えておこう。といっても、モデムを内蔵しているキットにはシリアルコネクタがない機種もあるので、チェックは怠らないようにしたい。

ただし、Linuxからソフトウェアモデムを利用するためのドライバも開発されつつあり、すでにLucentチップを使ったモデムに関しては動作実績がある。また、MotorolaもLinuxへの対応を発表した。ドライバの開発については、Linmodems.org（http://linmodems.org/。**画面5**）が詳しい。それぞれのチップが動作するかどうかについては、Winmodems are not modems（http://www.o2.net/~gromitkc/winmodem.html。**画面6**）というWebページに、ユーザーからの報告が表になってまとまっている。

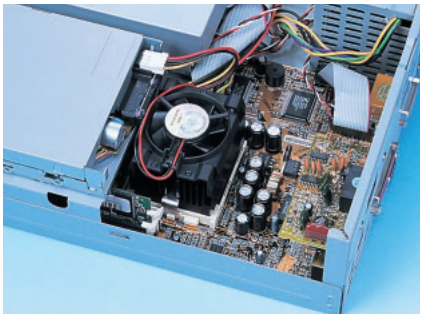


写真1　狭い筐体の中に各パーツが詰め込まれている

●そのほかのパーツ

ベアボーンキットの筐体は小さなものが多い。そのため、筐体の中にFDDやCD-ROMドライブ、電源などがギリギリに詰め込まれているキットもよくみられる（**写真1**）。このコンパクトさは、日本の住宅、オフィス事情においては大きなメリットであるが、反面、拡張性や放熱性においてはデメリットとなる。

通常の使用であればハードウェアを拡張することはあまりないだろうが、ネットワークインターフェイスが用意されていないのに、拡張スロットがひとつもないキットを選ぶと、ネットワークにつなぎたくてもどうしようもなくなることがある。用途がはっきりしていない場合は、1つくらいは拡張スロットのあるものを選んだほうがよいかもしれない。ただ、幸か不幸かWindowsに比べればまだまだ周辺機器のサポートが少ないLinuxでは、拡張スロットもそれほど必要ないと思われるので、やたらと多くの空きスロットを確保しなくてもよいはずだ。

Column

ソフトウェアモデムとは

ソフトウェアモデムとは通常、単体のモデムのハードウェア機能の一部をソフトウェアによるエミュレーションで実現しているモデムのことだ。このようにしてハードウェアパーツの点数を減らすことにより、価格を下げることができる。しかし、このソフトウェアは、Windows用ドライバの一部として供給されている。すなわち、Linux上からこのモデムを利用することはできないのだ。

なお、ソフトウェアモデムはよくWinmodemとも呼ばれるが、実は「Winmodem」は3Comの登録商標だ。つまり、絆創膏に対する「バンドエイド」、ステープラーに対する「ホッチキス」のようなものなので、正確にはソフトウェアで制御するモデムをすべてWinmodemと呼ぶのは誤りであろう。

ハードディスク、メモリに関してもスロットが1つしかないこともある。その場合は、余裕を持った容量のものを選択をしたほうがよいだろう。もっとも、ハードディスクやメモリの価格はどんどん下がってきている。必要になったときに入れ換えるほうが、かえって安くつくということも考えられる。

メモリに関しては、その物理的な大きさ（装着時の基板の高さ）が問題となることがある。コンパクトな筐体はギリギリのスペースで作られていることもあり、背の高いメモリでは、ほかのパーツやケースに接触してしまう可能性があるのだ。意外に気がつかない点だが、ショップの店員さんに聞くなどして、できるだけ確認するようにしよう。

ハードディスクに関しては、熱と音についても考慮したほうがよい。音に関しては、気にしなければよいともいえるが、熱に関しては、それが原因で誤動作する可能性もある（特に小さい筐体の場合）。ハードディスクの発熱量には、ものによってかなり差があるようだ。購入前に、できるだけ情報を集めて選ぶようにしよう。

また、通常は用意されている外部コネクタが用意されていないこともある。そのうち、ディスプレイやキーボードコネクタもなく、あるのはUSBコネクタだけ、というキットが出てくるかもしれない。LinuxでUSBが完全にサポートされるようになれば、それでも問題はないだろうが、現状では外部コネクタの存在も重要なポイントのひとつだろう。

今回選んだパーツ

では、実際にベアボーンキットを組み立てて、Linuxマシンにしてみよう。

選択にあたっては、セカンドマシンとしての使用を考え、小さくて、コストパフォーマンスに優れたものとすることを念頭においた。そのため、オンボードに機能が統合されたマザーボードのものを選ぶことにする。拡張スロットの有無に関しては悩むところだが、最初から必要そうなすべての機能を備えたキットを選ぶことにより、拡張性はなくてもよいことにした。

使用するCPUは、コストパフォーマンス的に優れたCeleron-500MHz（**写真2**）。よほど負荷のかかることをしない限り、十分なパワーが得られるはずだ。

メモリはとりあえず128Mバイトの SDRAMを1枚。64Mバイトではさすがにちょっと少ないし、コストパフォーマンスもあまりよくない。ただし、足りなくなったときのことも考え、念のためスロットは2つは欲しい。

ハードディスク（**写真3**）はIBM製のDPTA-351500。これは編集部にあったもので、15Gバイトの一世代前のものだが、静粛性に定評があり発熱量も少ない。

以上を考え、選択したのがSF-

写真2　Intel Celeron 500MHz
CPUとマザーボードの互換性にも注意が必要だ。ソケットの種類を事前に必ず確認すること。

写真3　IBM DPTA-351500
ハードディスクの選択に際しては発熱にも気をつけたい。

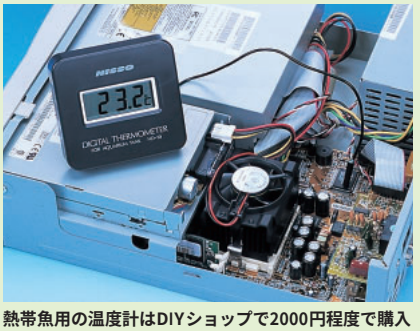
Column

内部温度のチェック

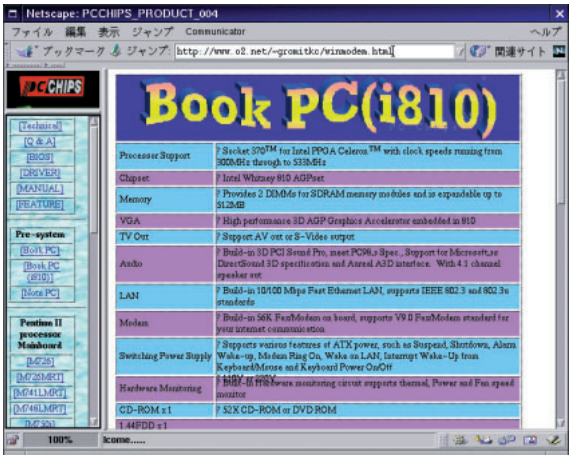
コンパクトなベアボーンの場合、発熱の多いCPUやハードディスクを使用すると、内部温度が上昇して誤動作を起こす危険がないとはいえない。Windowsであれば、マザーボードメーカーが内部温度をチェックするユーティリティソフトウェアを提供している場合もあるが、Linuxに対応したものは数少ない。そのようなときに、内部温度のチェックに筆者が使っているのが、熱帯魚の水層用の温度計である。

液晶表示板から、細いケーブルが出ていてその先にセンサーが付いているので、センサー部分のみを内部に入れて、内部温度を確認することができる。またマザーボード付属の温度計と違い、センサーを自由に移動できるので任意の位置の温度を計測することができる。

内部温度は空気の流れによっても、かなり変化する。たとえばハードディスクやボードの位置を動かしたりするだけで、温度が下がる場合も多い。いろいろな部分の温度を計測して、問題になっている部分をうまく冷却できるようにしよう。



熱帯魚用の温度計はDIYショップで2000円程度で購入したもの。計測できる温度には注意しよう。



画面7
メーカーである「PC CHIPS」のWebページ。細かいハードウェア使用に関しては、掲載されていなかった。

写真5　組み上がったベアボーンマシンと背面パネル
シリアルコネクタは存在しない。Windowsであれば確実に内蔵モデムを使えるが、Linuxではモデムの動作に不安がある。

BK810というキットだ。このキットは、CPU、メモリ、ハードディスクを加えるとひと通りの機能がそろったマシンになる。秋葉原を実際に回ってみたが、いくつかのショップで販売されていたようだ。複数のWebサイトで、通信販売も行われている。購入価格は2万3980円。全パーツを含めても、6万円ほどですべてが揃う。

残念ながら、メーカーのWebページにはハードウェアに使われているチップに関しての詳しい情報は記載されていなかったので、それらについては実際に店頭で見て確認した。

キットを組み立てる

今回のベアボーンキットの組み立ては、機能統合マザーボードを利用しているうえに組み立て説明書も付いているので、とても簡単だった。ただし、同じキットでも販売店によって完全な日本語説明書が付いていたり、英語のものしかなかったりと、いろいろあるようだ。

さらにパーツの入れ替わりが激しいためだろうか。説明書の内容が実機と異なっている場合もある。今回のキットでも組み立てに支障が出るほどではないが、パーツが解説書と異なっているものもあった。

組み立てといっても、CPUとハードディスクとメモリを組み込むだけだ。組み立てに要した時間は1時間ほど。それもすべてきちんと説明書を読みながら進めていってのことだ。組み立てに慣れた人なら、説明書も読まずに、15分程度で組み立てることができるだろう。注意点は、コンパクトなケースを選んだため、気をつけないとケーブルをパーツの間にはさんでしまう恐れがあることくらいだ。

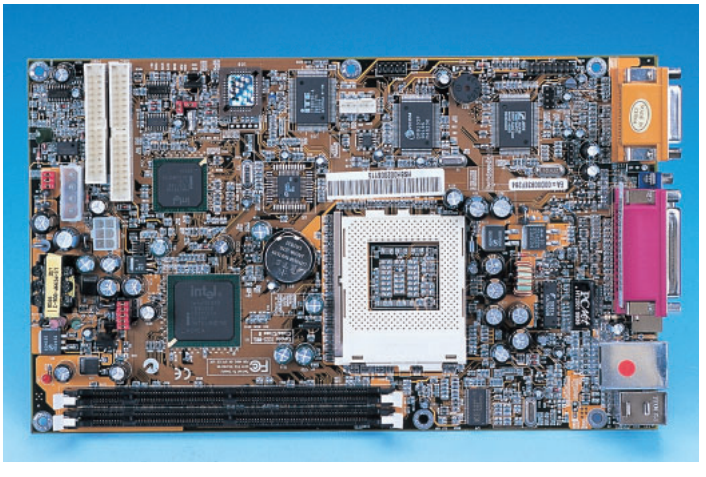


写真4　キットのマザーボード
i810＋Socket370 PPGAという構成。CPUはCeleronということになる。DIMMスロットは2つあり、最大512Mバイトまでメモリを拡張可能。

TurboLinux 6.0のインストール

今回使用するキットは、Intel810（i810）チップセットを使用している。そのため、Xを使用するにはXFree86 3.3.6以降が必要となる。そこで今回は、標準でXFree86 3.3.6を採用しているTurboLinux Workstation 日本語版 6.0を使用することにした。

ただし、XFree86 3.3.6でもi810チップを使うためには別途モジュールの追加が必要になる。よって、インストール段階でのXの設定では、正常に画面を表示することができない。とりあえず、Xの［設定のテスト］で「正しく表示されましたか？」というダイアログに「いいえ」と答え、そのあとに表

写真6　82810チップ
i810チップセットのメインチップ。グラフィックメモリコントロールハブ（GMCH）と呼ばれるとおり、ビデオ機能が統合されている。

示される［設定オプション画面］で［Write Configuration］を選んで（**画面8**）先に進み、一度インストールを完了しよう。

続いて、i810チップを使用するのに必要なagpgartモジュールを追加する。といっても、モジュール自体はすでにインストールされている。必要な作業は、rootでログインし/etc/modules.confというファイルの中にあるagpgartの行のコメントアウトを意味する“＃”を削除し、再起動するだけだ。

```
# vi /etc/modules.conf
```

（viが起動）

```
#alias char-major-10-175 agpgart
```

という行を、

```
alias char-major-10-175 agpgart
```

に変更する。

```
# shutdown -r now
```

（リブート）

再起動したら、再びログインして“startx”と打ち込めばXが起動するはずだ。もし起動しなければ、Xの設定が間違っていることが考えられるので、turboxcfgコマンドを使って再設定しよう。

ディストリビューションがハードウェアをサポートしていても、インストール時に認識されないことがある。その場合は自分で、ドライバを組み込み、ハードウェアを認識させなければならない。

今回のキットに組み込まれているサウンド、ネットワークインターフェイスに関しても、ドライバの組み込みが必要だった。

前述のように、今ではドライバのモジュール化が進んだため、モジュールを追加するだけで簡単に新しいハードウェアを使える場合がほとんどだ。

画面8
Xの設定では、必要項目を入力してそれを書き込んでおけば、あとでモジュールを加えただけでXを使えるようになる。

モジュールの組み込み方法については、それぞれのモジュールによって異なるので、ドキュメントをよく読んで行う必要がある。単体で配布されているモジュールには、たいていドキュメントが付属している（カーネル付属のモジュールの場合はカーネルソースに付属している）。

TurboLinux Workstation 6.0では、インストール時に開発版を選択すれば、自動的にカーネルソースがインストールされる。カーネルソースがインストールされていない場合にも、以下のコマンドラインでインストールできる。

```
# rpm -ihv kernel-source-2.2.13-33.i386.rpm
```

モジュールの設定に関するドキュメントは/usr/src/linux/Documentation/の中にインストールされる。カーネル付属のモジュールはすでにコンパイルされてバイナリになっているものがほとんどだが、多くのドキュメントにはコンパイルの段階からの解説が記載されている。コンパイルの必要がなければ、設定部分からのみ参照すればよい。どのモジュールが現在使える状態になっているかは、以下のコマンドで調べることができる。

```
# modprobe -l
```

なお、モジュールの設定ファイルはスーパーユーザーの権限でしか書き換えられないので、組み込み作業を行うときはrootでログインするか、suコマンドでrootになって作業しなければならない。

単体で配布されているモジュールを利用する際には、コンパイルして使う場合と、すでにコンパイルされたファ

イルが配布されている場合がある。ドキュメントはまず間違いなく英語で書かれているが、ほとんど指示されたコマンドを実行するだけで済むはずなので、英語が苦手な人も挑戦してみてほしい。

●ネットワークインターフェイス

このマシンに付属のネットワークインターフェイスのチップは、Davicom SemiconductorのDM9102だ（**写真7**）。このチップはTurboLinuxで対応していることになっているが、インストール時には自動認識されない。そのため、設定作業が必要である。

DM9120のドライバはモジュールとして用意されているので、モジュールを読み込むように設定すればよい。読み込むモジュールはdfme.oだ。このモジュールをイーサネット用のデバイスに割り当てるために、/etc/modules.confに以下の行を追加し、再起動する。

```
alias eth0 dfme
```

ただし、DM9102はインストール時に自動認識されないため、ネットワークを使えるようにするには、再度設定が必要になる。ネットワークの設定はturbotoolsのturbonetcfgで行う。

●サウンド

ネットワークの設定ができたら次はサウンドの設定に移ろう。サウンドのチップにはC-MediaのCMI8738（**写真8**）が使われている。このチップに関しても、Turbo Linuxではサポートされていることになっているが、やはり自動認識はされないので、自分でドライバを組み込むことになる。

組み込みの方法は、ネットワークドライバとは少し異なり、コマンドライン以下のコマンドを入力する。

写真7　ネットワークのチップDM9102F

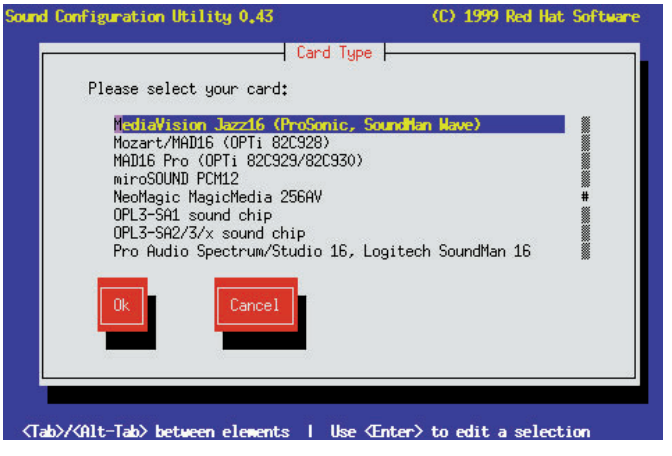
写真8　サウンドチップCMI8738PCI

```
# modprobe cmpci
```

これで、サウンド機能が使えるようになるはずだ。

modprobeコマンドによるモジュールの認識は、起動のたびに行わなければならない。いちいちコマンドを打ち込むのは面倒である。そこで、/etc/rc.d/rc.localにこのコマンド行を追加しておくとよいだろう。rc.localに書かれたコマンドは常に起動時に実行されるからだ。

LASER5 Linux 6.0や Red Hat Linux 6.1には、サウンド機能を簡単に設定できる「sndconfig」というコマンド（**画面9**）が用意されている。ハードウェアによっては自動認識も可能だが、IRQやI/Oアドレスの値を求められる場合もある。念のため事前に調べておくほうがよいだろう。



画面9
sndconfigコマンドでは設定に成功するとRichard Stallmanの声が流れる

●モデム

さて、残ったハードウェアはモデムだ。このキットに搭載されているモデムはソフトウェアモデムであるため、使えるかどうかは実際に試してみないとわからない。結論からいうと、残念ながら今回のキットに付属のモデムを動作させるにはいたらなかった。

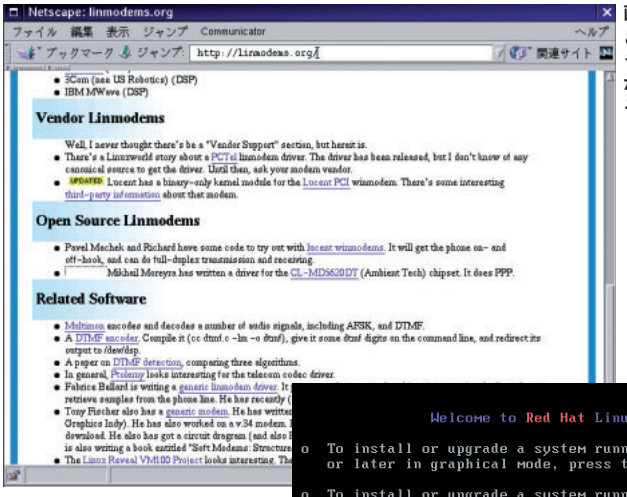
モデムを使うために実際に行うべき作業手順について、参考までに書いておこう。

まず、前述の「Winmodems are not modems」のWebページ（http://www.o2.net/~gromitkc/winmodem.html）を参照し、モデムが動作するかを調べる。そのときモデムの種類を判別するのに使われるのが、「FCC ID番号」だ。この番号はモデムの基板のどこかに印刷されていたり（**写真9**）、シールが貼ってあったりするので、まずそれをメモしよう。その番号をもとにWebページの表を調べ、「WM」という表

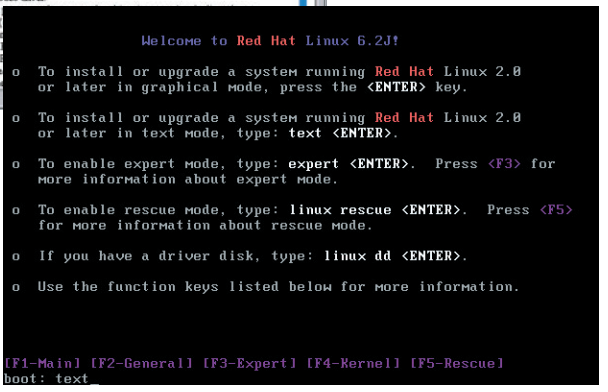
記がされていなければ、そのモデムはLinuxで動作する可能性がある。

動作報告があったら、Linmodems.org（http://linmodems.org/。**画面10**）からベンダー提供のドライバを入手してインストールしよう。Lucent PCIモデム用のドライバであれば、圧縮してあるファイルをunzipコマンドで展開し、rootになって、./ltinstコマンドを実行するだけだ。それ以外のドライバについてはマニュアルを参照してほしい。

動作報告がないからといって、動作する可能性がまったくないわけではない。ドライバは./unltinstコマンドで簡単にアンインストールできるので、試してみる価値は十分にあるだろう。



画面10
ここで紹介したベンダー提供のドライバ以外にもいくつかのドライバがある。試してみる価値はあるだろう。



画面11
インストーラの初期画面では、さまざまなインストール方法を選択できる。

Red Hat Linux 6.2Jのインストール

今回は、さらにRed Hat Linux 6.2Jでもインストールおよび動作テストを行った。

Red Hat Linux 6.2JではXを利用したグラフィカルなインストーラを採用しているが、この原稿の執筆時点では、i810を利用したマシンにおける不具合が報告されており、グラフィカルインストーラを使用することはできなかった。

そのため、インストール用CD-ROMを入れて立ち上げたときに表示されるboot:プロンプトでtextと入力し、テキスト画面によるインストールを行うことにした（**画面11**）。通常はあまり使用されることのないと思われるテキストインストールだが、（そのためかどうかわからないが）残念ながら言語に［Japanese］を選択すると、文字化けが発生して漢字によるメッセージがまったく読めなくなってしまう。

日本語表示はあきらめて［English］を選択し、インストールを進めた。メッセージが英語、画面がテキストモードである以外は、グラフィカルインストーラで行う操作とほとんど変わりはない。グラフィカルインストーラを使用した経験があれば、さほど戸惑うこともないだろう。

テキストインストーラでも、やはりXの設定は行わなければならない。Xの設定のテストではエラーになるのだが、とりあえず設定値をキチンと入力しておき、［Starting X］の画面で［Skip］を選んでインストールを終了しよう。

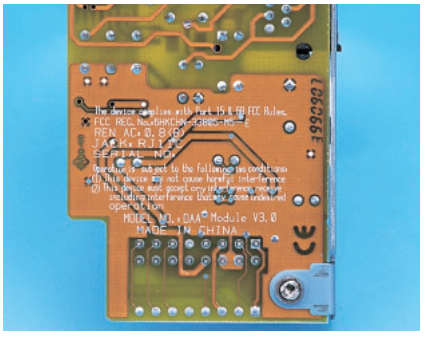


写真9　FCC ID番号
この番号をもとにモデムの種類を判別する。

そして、http://www.redhat.com/errata/から修正された新しいカーネル（kernel-2.2.14-6.1.1.i686.rpm）をダウンロードして、i810が使えるようにアップデートする。このカーネルはRPMになっているので、アップデートはファイルのある場所で以下のようなコマンドを実行するだけだ。

```
# rpm -ivh --force kernel-2.2.14-6.0.1.i686.rpm
```

カーネルパッケージのインストールが完了すると、/bootにvmlinuz-2.2.14-6.0.1という名前のファイルが追加される。起動時にこのカーネルイメージがロードされるように、/etc/lilo.confの「image=」の行を以下のように書き換える。

```
image=/boot/vmlinuz
```

変更を有効にするためには、書き換

え後に、liloコマンドを実行する必要がある。忘れずに実行すること。

ところがこのカーネルのファイル、5Mバイト以上あるのだ。まずネットワークを使えるようにし、ネットワーク経由でこのファイルを転送することにした。そこで、ネットワークカード用のモジュールをベンダーのWebページ（**画面12**）からとってきたのだが、なんとLinuxのドライバなのに、MS-DOSのEXEファイルを実行してドライバのソースとバイナリを展開するようになっている。

しかたがないので、MS-DOS上で展開してから、フロッピーディスクを使ってLinuxマシンにコピーした。MS-DOSのフロッピーディスクをLinuxで読むには、マシンにフロッピーを挿入してmountコマンドを使えばよい。

```
# mount -t vfat /dev/fd0 /mnt/floppy
```

これで、フロッピーディスクの中身が/mnt/floppy以下に見えるようになる。次にモジュール用のディレクトリにdmfe.oという名前でコピーする。

```
# cp /mnt/floppy/dmfe_rh61.o /lib/
module/2.2.14-5.0/net/dmfe.o
```

そして、TurboLinuxの場合と同様、/etc/conf.modulesに以下の行を追加し、再起動する。

```
alias eth0 dmfe
```

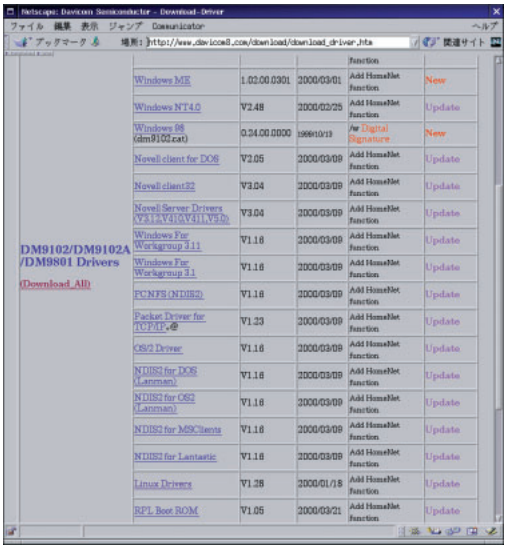
このあと、netconfigコマンドで設定すればネットワークは使えるようになる。実は、このドライバは名前からわかるようにRed Hat 6.1用なのだが、とにかく動くことが先決ということで、使ってみることにしたのだ。もちろん、ドキュメントを参考に、ソースからキチンとコンパイルしてモジュールを作ったほうがよいのはいうまでもない。

その後、先に書いたようにカーネルのアップデートを行えば、Xを使うことができるようになる。ただし、ネットワークモジュールdmfe.oは、カーネルをアップデートした後で、新しいモジュールディレクトリとなる/lib/module/2.2.14-6.0.1/netに、再度コピーする必要がある。

なおサウンドに関しては、自動的に認識されたので特に作業は必要なかったことをお知らせしておく。

おわりに

今回は行わなかったが、カーネルのアップデートによって、新たなハードウェアを使えるようになる場合もある。たとえば、現在の最新の安定版カーネルは2.2.14であり、2.2.13の時にはなかったAthlon用のチップセットAMD-751のサポートが追加されている。また、開発版カーネルではUSB機器に関しても徐々に使えるようになってきており、拡張に弱いというベアボーンの弱点も克服されていくだろう。



画面12 Davicom Semiconductorのダウンロードページ。Linux用のドライバも用意されている。

マザーボード	PC CHIPS BKi-810
対応CPU	Socket370（300MHz-533MHz）
メモリソケット	2
VGA	オンボード（i810）
サウンド	オンボード
ネットワークカード	オンボード（10/100BASE-T）
モデム	オンボード（56K V.90ソフトウェアモデム）
FDD	付属
CD-ROM	付属
拡張スロット	なし
キーボード	付属
マウス	3ボタンマウス（PS2）
スピーカー	2本
USB	2
シリアルインターフェイス	なし
パラレルインターフェイス	付属
サイズ（H×W×D）	82×272×300mm
キット以外のもの	
CPU	Intel Celeron 500MHz
メモリ	128Mバイト（PC100 SDRAM）
ハードディスク	IBM DPTA-351500（15Gバイト）

表2　ベアボーンキットSF-BK810の仕様

Lucent PCI	http://www.linmodems.org/linux568.zip
Cirrus Logic CL-MD5620DT, PCI	http://linmodems.org/CLModem-0.1.0.tar.gz
PC-tel	http://www.o2.net/~gromitkc/pctel/hsp56-linux-1.tar.gz

表3　ソフトウェアモデムドライバ関連のWebサイト

はじめての自作でも安心〜もうパーツ選びには悩まない！〜

文：山岸典将　*Text : Norimasa Yamagishi*

最近のメーカー製マシンはWindowsがインストールされているものがほとんどだ。もちろん、Linuxを使うのであればWindowsは必要ない。そこで、見つけたのが日本ゲートウェイの「JI-SAKU-KI」。マシンを作るのに必要なパーツはすべて揃って、OSは付いていない自作キットだ。OSが付いていないということは、Linuxで使ってくださいと言っているのだろうか。きっとそうに違いない。ということで、さっそく注文してみることにした。

購入するマシンを検討する

注文は、ゲートウェイのWebページ（http://www.gw2k.co.jp/）より簡単に行える。Athlon 650MHzと850MHzの2タイプがあり、CPU以外の仕様は共通となっている。なお、ハードディスクなどのパーツをほかのものに変更することはできないそうだ。

価格は、850MHzのものが16万9800円、650MHzが8万9800円となっている。CPUの価格がそのまま影響しており、コストパフォーマンスが高いのは650MHzのほうだといえるだろう。今回は650MHzのマシンを購入することにした。

組み立て済みのマシンとの価格差が、少し気になるところだろうが、ちょうど同スペックのパーツを使っていると思われるSelect650というマシンが存在する。このマシンはOS抜きの設定はないのだが、Windows 98 Second Editionが付属している以外は同等の構成で、11万9800円となっている。3万円という価格差は、ちょっとした組み立ての手間さえ惜しまなければお買い得であるといえる。特にLinux専用マシンとして考えている場合、十分検討に値する製品だ。

保証期間は90日となっている。パーツにもよるが、バルクのものよりは長く、パッケージのものよりは短い、といったところだろうか。もっとも、バラバラにパーツを買ってくる場合に比べ、すべてのパーツが一度に揃うため、すぐに組み上げて保証期間内にほぼ確実に動作チェックできる。それほど長い保証期間は必要ないのかもしれない。実際問題、現在のハードウェアは初期不良以外の故障の発生率はかなり低いのだ。初期不良さえなければ90日という保証期間はそれほど問題になることはないと思われる。

到着そして組み立て

発注後2週間ほどしてマシンが到着した。まず、添付されてくる組み立てマニュアル（カラーで28ページ）を見る。今まで組み立てを経験したことがない人でも、読みながら作業すれば、戸惑うことなく完成させることができるはずだ。特に普通は絵で済まされてしまいそうな、パーツの構成や実際の組み立て風景などが、写真で解説されているのでわかりやすい（**写真2**）。ビ

写真1　「JI-SAKU-KI」の全キット内容
開封したら、まずはパーツが揃っているかチェックしよう。

CPU	650MHz AMD Athlonプロセッサ
M/B	AMD-750チップセット（Ultra DMA/66対応）
RAM	64MB SDRAM（64MB×1）
HDD	10GB Ultra DMA HDD
FDD	3.5インチ×1
CD-ROM	17x min/40x max
ビデオ	nVIDIA RIVA TNT2 M64 16MB AGP
サウンド	Sound Blaster Live! Value
モデム	56K PCI モデム
キーボード	109 Key 日本語 キーボード
マウス	PS/2マウス、マウスパッド
その他	ネジ、スクリュードライバ、組み立てマニュアル、ドライバCD

表1「JI-SAKU-KI 650」のスペック

デオカード、サウンドカードといったハードウェアに関しては個別のマニュアルは添付されていない。

まず、マニュアルに書いてある添付部品の一覧を確認する。部品についても、ネジの1つ1つまで、写真やていねいなイラストを使って掲載されているので、確認は簡単かつ確実。当然だが必要なパーツはすべてそろっている。プラスドライバーまで付属しているので工具の心配もない。

マザーボードに用意される拡張スロットはPCIのみ5つ。キットで使用するのはそのうちの3つである。メモリスロットも3つ用意されていて拡張性は十分だ。また、IDEインターフェイスはUltraDMA/66にも対応しているので、現在主流の高速なハードディスクを増設したときにも有効に活用することができる。HDDの10Gバイトという大きさは、今となっては小容量と感じられるかもしれないが、3.5インチ用の空きベイが4つあるので、必要ならばいくらでも買い足しは効く。USBコネクタも3つ用意されている。Linuxでも、今後強化されていくであろうUSBのサポートを生かすこともできるわけだ。

付属のAthlon用のヒートシンクとファンはとても大きく、Athlonの発熱量の多さを想像させる（**写真5**）。これだけ大きなヒートシンクとファンがついていれば、CPUに関しては熱の問題が発生することはないだろう。

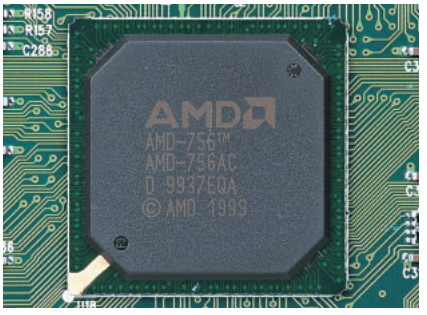


写真4　AMD750チップセットのAMD-756チップ
HDD、PCI、USBなどのI/Oバスを制御する（ADMではペリフェラルバスコントローラと呼んでいる）。最大4ポートのUSB、UltraDMA/66をサポート。


写真5　CPUとケースファン
CPU本体と同じくらい巨大なヒートシンクが目につく。キットに付属のケースファンも大型のものだ。


写真6　メモリ
標準の64Mバイトではややもの足りないところ。アップグレードはまずメモリから。


写真7　キットに付属のマウスとキーボード
マウスと同様、キーボードも少し古いタイプのものだ。

残念ながらマウスは、最近では少なくなってきた2ボタンマウスとなっている（**写真7**）。そのためXを使うためには3ボタンマウスのエミュレーションをしなくてはならない。3ボタンマウスでの操作に慣れた人には、マウスのみ交換したいところかもしれない。

パーツの確認が終われば、いよいよ組み立てだ。写真による解説は、コネクタの位置や向きを間違えることもなく、うれしいものだ（ある程度組み立てに慣れた人でも、結構逆に挿したりするものである）。

組み立てに要した時間は1時間弱。マニュアルにしたがって作業を進めることで、プラモデル感覚で組み上げることができた。使用するパーツがすべて決まっているキットの強みだろう。LEDランプやリセットスイッチなどの

写真2　組み立てマニュアルは写真入りで、とても親切。

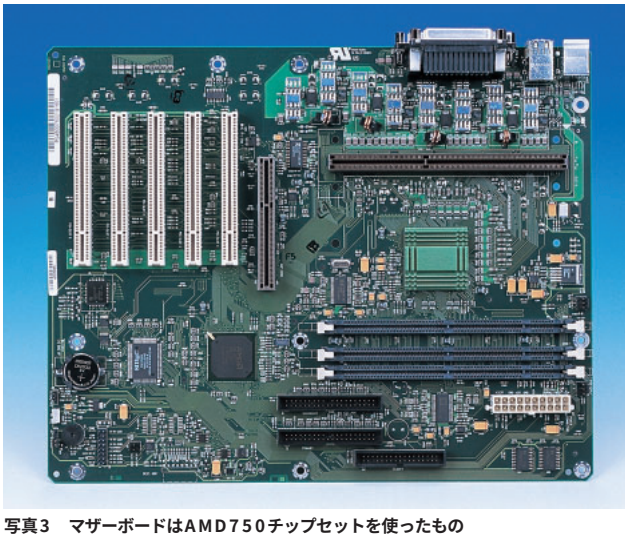


写真3　マザーボードはAMD750チップセットを使ったもの

写真8　組み立て完成後のマシン
写真ではわかりずらいが、前面パネルに「自作機」というエンブレムが付いている。

写真9　ケースの内部
ケースがミドルタワーのため内部に余裕があり、パーツは取り付けやすい。空きベイも多く、今後の拡張もやりやすそうだ。

コネクタ類も1つにまとまっており、間違って接続してしまう可能性が低いのもうれしい。ただ、組み立て完成後にネジが数本余ってしまった。マニュアルを読み直すとネジは予備として多めに入っているとのことだが、注意書きを読み直すまではちょと不安になる。

組み立て完了後の、OSのインストールに関してのマニュアルはないが、Windowsドライバのインストール方法を含むマザーボードの解説書も付属しているので、同時にWindowsを使うという場合に役立つだろう。

Red Hat Linux 6.2J のインストール

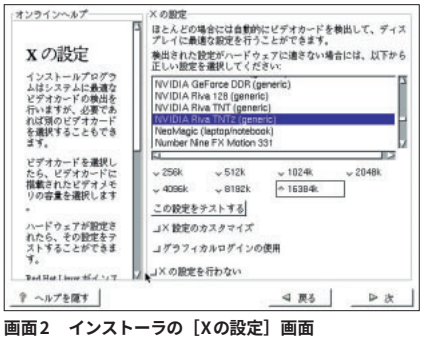
組み上がったマシンにインストールするディストリビューションだが、まず、最新のRed Hat Linux 6.2Jをテストしてみた。

インストールは特にこのマシン固有の問題というのは発生しないので、通常どおり進めていけばよい。付属のビデオカードRIVA TNT2（**写真10**）は自動認識し、サウンドカードのSound Blaster Live Value（**写真11**）も特に設定することもなく認識、動作した。

［Xの設定］でもディスプレイの仕様を自動認識するため、指定するのはビデオカードのRAM容量の［16384K］くらいだ。ただし、ほかの項目について自動認識されたとしても、念のため［この設定をテストする］は行うべきだろう（**画面2**）。もし、テストがうまくいかなかった場合は、［Xの設定のカスタマイズ］を選択して、細かい設定をいろいろ試してほしい。

以上で、インストールは通常どおりに特に問題なく終了した。

ただし、困ったのはモデムだ。詳しくは「ベアボーンキット」のパートを見てほしいが、Linuxで一番問題になる内蔵タイプのモデムを使っているのだ。このモデムは、PCIスロットに装着するタイプのソフトウェアモデムで（**写真12**）、チップはCONEXANTのものを使っている。残念ながら、このチ



画面2　インストーラの［Xの設定］画面
設定のテストを行うべきだろう。もし、ここでXがうまく動作しないまま、［グラフィカルログインの使用］を選択すると、後でログイン画面が表示されなくなるため面倒なことになる。

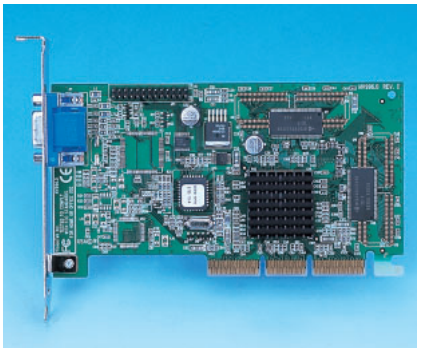


写真10　RIVA TNT2
Linuxでサポートされるビデオカードとしては優れた性能を誇る。

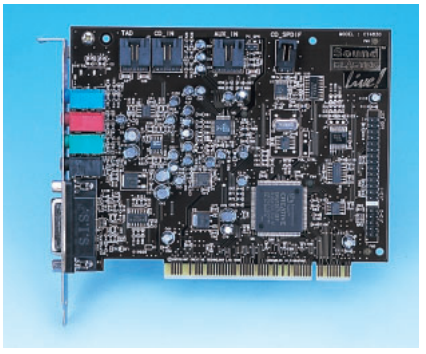


写真11　Sound Blaster Live! Value
メーカー製PCでも、多く採用されているSB Live!の廉価バージョン。デジタルI/O部分が除かれている。

写真12　内蔵型56Kモデム
やはり内蔵モデムが問題になってくる。現状では一部のものを除いてほとんど使用できない。

ップに対応したドライバは、調べた限りではまだ存在しないため、利用することができなかった（**画面3**）。

TurboLinux Workstation 6.0のインストール

次にTurboLinux Workstation 日本語版6.0をインストールしてみる。TurboLinuxの場合は、自分で設定しなければならない点が多い。

大きく異なるのが、Xの設定だろう。TurboLinuxでは、ディスプレイの「画面サイズ」、「水平同期」、「垂直同期」、「デフォルトの色数」、「使いたいビデオモード」といった項目の入力を求められる。画面サイズ、水平同期、垂直同期に関しては、使用するディスプレイのマニュアルを確認して入力しよう。デフォルトの色数とビデオモードはディスプレイの性能にも関係するが、1600×1280ドットの24bpp以下なら問題ないだろう。

また、「起動サービスの設定」といった項目の設定も必要になる。もっとも、これに関しては、必要なサービスがほぼデフォルトで有効になっているので、意識して設定したい項目がなければ、そのまま終了してもよい。

そして、サウンドカードのSound Blaster Live Valueだが、これもTurboLinux自体は対応しているものの自動認識はしない（**画面4**）。そのため、若干の設定が必要となった。といっても、スーパーユーザになって/etc/modules.confというファイルに1行加えるだけだ。

```
# su -
(パスワードの入力)
# vi /etc/modules.conf

alias sound emu10k1
(この行を最後に加える)
```

ここで再起動すれば、サウンドが使えるようになる。TurboLinuxの場合もキットに含まれている内蔵モデムは使用できない。

Red Hat LinuxとTurboLinuxを比較すると、Red Hat Linuxのほうが設定が必要な項目は少ない。これは、今回のマシンに限ったものではなく、2つのディストリビューションの仕様上の違いだ。ただし、設定項目が少ないほうが一概に良いとはいえない面もある。ある程度Linuxに関する知識がある場合は、自分できめ細かい設定を行えるほうが望ましいという場合も多々あるからだ。

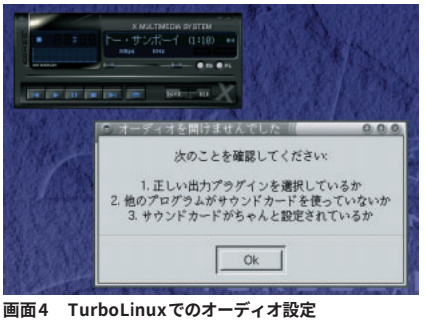
今回は、2つのディストリビューションで簡単なインストールテストを行ったわけだが、このマシンに限らず、Athlonを利用したマシンにおいて注意してほしい点がひとつある。それは、カーネル2.2.12でAthlon対応コードの修正が行われているということだ。不要なトラブルを避けるためにも、Athlonを使う場合、2.2.13以降のカーネルを使用しているディストリビューションを選ぶべきである。古いディストリビューションでは、インストール後に起動しなくなるということもあるようだ。

まとめ

今回の自作パーツキットとRed Hat Linux 6.2J/TurboLinux Workstation 日本語版 6.2の組み合わせは、特に大きなトラブルもなく、モデム以外の機能はすべて利用することができた。モデムに関しては現状では外付けのモデムをシリアルコネクタにつないで利用するしかないだろう。

自分でパーツから選んで自作する場合、たまにうっかり必要なものを買い忘れることもあるが、このキットならまったくそんな心配はない。「自作に挑戦したいと思っているが、いまひとつ自信がない」、「メーカー製マシンの拡張性が不安」といった向きには、お勧めできるキットだろう。あえていえば、ほかのBTO対応完成モデルのように、メモリやハードディスクの容量、ビデオカードの種類など、オプションパーツの構成を選択できるようになると、さらに魅力的なものになるだろう。検討をお願いしたいところだ。

画面3　Red Hatのモデム設定画面
自動認識はもちろんできない。ドライバもないため、残念ながら、今のところ使用はあきらめるしかない。



画面4　TurboLinuxでのオーディオ設定
TurboLinuxではサウンドカードの設定は自分で行う必要がある。

画面5　dmesgの出力
カーネルがAthlonを認識していることがわかる。

ハイエンドLinuxマシンを目指して〜拡張性を重視したい〜

文：徳川 冨士男　*Text : Fujio Tokugawa*

10万円以下の予算でも結構十分な性能をもったLinuxマシンを自作することができるが、もっと予算があるならどこを増強するか、そのあたりを検討しながらハイエンドマシンを作ってみよう。

ハイエンドといっても、目的がサーバかデスクトップかで強化するポイントが違ってくる。サーバ用なら、ハードディスクはRAIDシステムにしてデータの信頼性と応答性を高めるのが常道だ。うーんRAIDか。ちゃんとしたハードウェアRAIDは自作なんかしないでメーカーから買ってくるのが正解だと思うので、今回はハイエンドデスクトップを目指そう。

ところで、予算に上限はつきものだ。それを決めておかないとつい不要なものまで買ってしまうので、今回はだいたい20万円ぐらいまでとした。自作という視点に立つと、一度にすべて最高のものを揃えてしまうよりも、少しずつ買い足して増強することが楽しみといえる。贅沢気分はちょっとだけにとどめておくのがちょうどよいだろう。

1GHzが欲しいけど…

まずはCPUを決めよう。高性能CPUといえばIntelのPentium IIIかAMDのAthlonで決まりだろう。Athlonは1999年8月に発表、PentiumIIIのCoppermineコア版も同10月発表と、まだ新しいため「Athlonなんて知らない！」とハングアップしてしまうディストリビューションも一部あったが、もちろん対応策（パッチ）はあるし、最新バージョンを使用すればどちらでもLinuxは動作するのでご安心を。

4月現在で発表されているのは1GHzクロックのCPUが最高速だが、まだ秋葉原などには流通しておらず買うことはできなかった。どうせ最高速が狙えないのなら、どれを取ってもたいして変わらないということで、手頃な価格のものを選ぶことにした（初物は高いからね）。1GHzのCPUが買いやすい値段になったら、その時点で交換することを検討しよう。

ハイエンドというからには、CeleronやK6-2といった低価格PCで使われている500MHz程度のCPUより、ずっと速くないとつまらない。やはり700MHz以上は欲しい。そういう条件で検討したCPUは、**表1**の通りだ。

Athlonにするか、Pentium IIIにするか迷うところだ。同一クロックならば、いろいろな雑誌で行っているベンチマークを見てもいい勝負をしているし、ほとんどのショップではPentium IIIよりAthlonのほうが安く手に入るようだ。Athlonにも魅力を感じたのだが、やはりPentium IIIのほうがほんの少し速そうだといった、ちょっとした気分で今回はPentium IIIに決めた。

Pentium IIIはベースクロック（FSB）が100MHzと133MHzの2タイプがあるが、これも少しでも速い133MHzのものにしたい。1GHzのPentium IIIのFSBも133MHzだし、今後も高クロックのCPUの新製品が発表されていくなかで、FSBは133MHzが主流になっていくと思われるからだ。

パッケージも2種類あって、カートリッジタイプのSlot 1パッケージと、ソケットタイプのFC-PGAパッケージのどちらにするのか悩むところだ。しかし、Pentium IIで初めて採用されたSlot 1は、2次キャッシュをCPUコアに内蔵することができなかったために

名称（FSBクロック）	パッケージ	価格
Pentium III		
Pentium III 700MHz	S.E.C.C.／FC-PGA	4万3000円〜
Pentium III 733MHz（133MHz）	S.E.C.C.／FC-PGA	4万7000円〜
Pentium III 750MHz	S.E.C.C.／FC-PGA	5万4000円〜
Pentium III 800MHz	S.E.C.C.／FC-PGA	6万9000円〜
Pentium III 800EBMHz（133MHz）	S.E.C.C.／FC-PGA	7万0000円〜
Pentium III 850MHz	S.E.C.C.／FC-PGA	8万5000円〜
Pentium III 866MHz（133MHz）	S.E.C.C.	8万6000円〜
Athlon		
Athlon 700MHz	Slot A	3万2000円〜
Athlon 750MHz	Slot A	4万0000円〜
Athlon 800MHz	Slot A	6万0000円〜
Athlon 850MHz	Slot A	7万5000円〜
Athlon 900MHz	Slot A	8万7000円〜
Athlon 950MHz	Slot A	10万0000円〜

表1　検討したCPU

写真1　Pentium III 733MHzリテール版
FSB 133MHzの FC-PGAパッケージ。リテール版なのでファンも付属する。

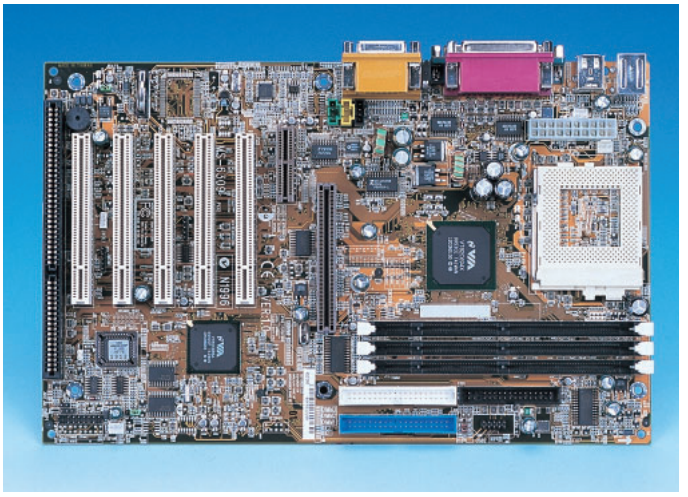


写真2
MSI MS-6309
Slot 1ではなくFC-PGAソケット使用のマザーボード。

作られたという理由があった。最新のCoppermineコアなら2次キャッシュを内蔵しているためにSlot 1にする必然性はない。

Slot 1は消えていく運命との噂も聞こえてくるので、将来性があるFC-PGAタイプを選択することにした。結局、Pentium III 733MHz（FC-PGA、FSB 133MHz）のリテールパッケージ版を5万2800円で購入した（**写真1**）。

デュアルCPU対応のマザーボードも考えたが

次はマザーボードを検討してみる。CPUだけでも決まっているから少しは選択肢が狭まっているのだが、非常に種類が多く、店頭で選ぶのは大変である。いくつかのファクターに分けて考えてみよう。

まず、AthlonやK6に対応しているものと、CPUにPentium IIIのFC-PGAを選んだ関係で使えないCeleron専用Socket370のマザーボードは無視しよう。しかし、FC-PGA対応Socket370とはひと目では区別がつかないのが不便なところ、いちいち説明を読まないといけないのでした。

これで、FC-PGA対応のものとSlot 1対応のものが残る。Slot 1用のS.E.C.C.パッケージじゃないCPUを買ったのに、なぜSlot 1マザーが残るかというと、Socket370変換アダプタにFC-PGAのCPUを装着し、Slot 1に挿すことで対応可能だからだ。しかし、変換アダプタの費用が余計に必要なことに留意しておこう。

明らかに除外できるのは、FSB 133MHzに対応していないマザーだ。少し前までよく使われていた440BXというチップセットは133MHzに対応していない。

最近ローエンドPCでよく使われている、グラフィックス機能を内蔵したIntelの810というチップセットは、型番によって性能（機能）が違うので注意しよう。番号だけで810のあとに何も付かないのが基本モデル。810Lはローコストモデルで PCI バス4本、UltraDMA/33までに制限されている。810DC100はグラフィックスを強化してある。ここまではFSB 66／100MHzに対応する。810EだけがFSB 133MHzに対応しているのだ。

そのほか、SiSやVIAがチップセットを供給している。チップセットにグラフィックス機能を統合している場合、画像用メモリに本体メモリの一部を利用するのと、専用メモリを外付けしているのでは、描画性能が違うのでそのあたりにも注意したい。

サーバ用途ならデュアルCPU対応マザーも候補に入れるべきだろう。PCI 66MHzに対応していたり、SCSIコントローラを搭載したマザーもあったりする。Linuxは、SMP（Symmetric MultiProcessing）に対応しているので、2個のCPUが同時に働くことができる。Webサーバのような目的に適していそうだが、今回はデュアルCPUは除外した。

悩んだ結果Apollo Pro133Aにした

さて、現時点で、FSB 133MHz、UltraDMA/66、AGP 4Xなどの最新技術に対応していて機能的に十分なチップセットは、VIAのAopllo Pro133Aとなった。各マザーボードメーカーがこぞって採用しているので、種類が豊富なところから選びたいというのもある。

結局、VIAのApollo Pro133Aを搭載し、CPUソケットにFC-PGAをそのまま装着できるSocket370タイプの中から、MSI（Micro-Star International）のMS-6309というマザーボード（1万4800円）を選んだ（**写真2**）。

MS-6309は、ATXフォームファクターの4層基板で、BIOSはメーカー製PCに多く使われているAMI BIOSを採用。CPU VoltageをBIOSで設定できるのは、自作派の王道（？）クロック

写真3　Apollo Pro133A（VT82C694X）
FSB 133MHz、AGP 2X／4X、PC133 SDRAMをサポートする。

写真4　VT82C686
スーパーサウスと呼ばれているI/Oコントローラ。UltraDMA/66、AC'97 Digital Audio、USB、PS/2キーボード／マウスをサポートする。

写真5　Pentium III 733MHz
基板上にチップが剥き出しになっているため、従来のCeleron用のファンの流用はできない。

アップへの対応か。メモリはDIMMソケットが3個で、最大1.5Gバイトまで。PCIバスは5本、そのうち1本はISAと兼用になっている。

AMR（Audio/Modem Riser）ソケットが用意されているほか、キーボード／マウス用のPS/2、USBポート、シリアルポート、IrDA赤外線ポート、ゲームポート、オーディオ用（Line-In、Line-Out、Mic-In）などのポートがついている。

オンボードサウンド（Creative社のCT5880相当）を搭載したモデルもあるが、標準ではチップセット内蔵のAC'97 CODEC機能を使った音源だ。ボード上の4つのLEDで状態を表示する機能は、ふだんは必要ないが、最初に組み込んだときやパーツを追加／変更したときなど、電源投入時からOSが起動するまでのあいだのトラブルはこれを見て判断できるだろう。

メモリをケチると遅くなる

メモリは、FSB 133MHzに対応するPC133の128MバイトSDRAMを2枚、2万3000円で購入した（**写真6**）。とりあえず256Mバイトもあれば十分だろう。アプリケーションがメモリを使いきると、ディスク上に一時保存しておくためのスワップが始まり、本来の仕事が遅くなる。Oracleなどのデータベースを動かすときには、メモリは大きいほうがよいのはいうまでもない。

グラフィックスカードには、Matrox の Millennium G400 DH（DualHead対応）32Mバイトメモリ搭載のバルク品（1万8800円）を調達した（**写真7**）。G400 MAXではないので、AGP 4Xには対応していないが、ほとんど大差ない。3D表示を行うときのテクスチャ張り付けの際に、AGPの性能が問われる程度だ。最初はGeForce256にするつもりだったのだが、Linuxで使うならG400も十分に速いと思われたのと、2台のディスプレイに出力できるDualHead対応に魅力を感じてこちらにしたのだった。

Linuxではグラフィックスボード1枚のDualHead対応がまだ行われていないというのは、買って帰った時点で気がついた。ちょっと失敗だったかもしれない。今後ドライバがサポートされるのを待つことにしよう。

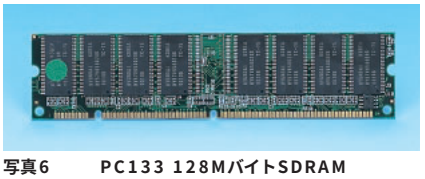


写真6　PC133 128MバイトSDRAM

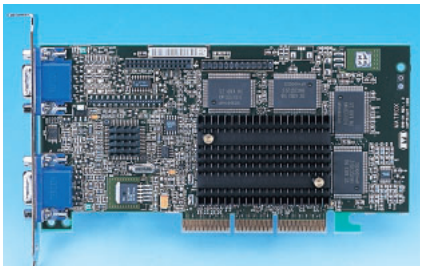


写真7　Matrox Millennium G400 DH（32Mバイト）
デュアルヘッド対応のためかメモリサイズが自動認識されなかった。

Column

SCSIのUltra160対応でも、160Mバイト／秒は無理？

現時点で、高速ハードディスクの転送速度は30Mバイト／秒程度で、160Mバイト／秒ものデータ転送速度を誇るドライブはないが、たとえば本文で使用しているATLAS Vを4台接続し、ソフトウェアRAIDでストライピングに設定すれば、同時アクセスによって合計で100Mバイト／秒以上のスピードで転送が可能になる。

ハードディスクなどの周辺機器とSCSIカードの間が160Mバイト／秒で転送できたとしても、SCSIコントローラカードはPCIバスに挿して使うので、PCIバスの転送速度以上には送れない。そのPCIバスは、クロック33MHz×32ビット＝133Mバイト／秒が最大能力なので、そこがネックとなってしまうのだ。

Adaptec SCSIカード19160と29160Nは、32ビットPCI用だが、64ビットPCIバス用の29160や39160などを使えば、バスの転送レートは2倍の266Mバイト／秒になる。しかし、64ビットPCIバスが付いているマザーボードは、ほとんどがデュアルCPU用で価格が高いのが難点である（トータルのシステム価格が高いから難点とはいえないか）。

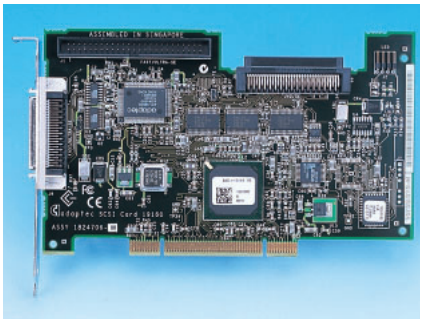


写真8　Adaptec SCSIカード19160
Ultra160 SCSI対応。

写真9　Quantum ATLAS V 18.3S
Ultra160SCSI対応のハードディスク。7200RPM、平均シーク時間6.3m秒、キャッシュ容量4Mバイト。

写真10　Toshiba XM-6401TA
一般的なIDEではなく、Ultra SCSIインターフェイスのCD-ROMを選んだ。

やっぱりSCSIでしょ

UNIXのサーバやワークステーションといえば、ハードディスクはSCSIインターフェイスで接続するのが当たり前だが、IDEのハードディスクもUltra DMA/66転送によって、十分なスピードを備えてきたし、なんといってもドライブの価格が安い。

それでも、ハイエンドなんだから強化するべきところには力を入れて、SCSIを使うことにする。SCSIとIDEのデータ転送速度は、**表2**のようになっている。Ultra160 SCSIでは、最大160Mバイト／秒にまで発展してきている。

そこで、SCSIコントローラは、Ultra160のインターフェイス対応のAdaptec SCSIカード19160（2万7800円）を購入した（**写真8**）。68ピンのLVD（Low Voltage Differential）SCSIコネクタには、Ultra160とUltra2 SCSI対応のハードディスクを接続する。50ピンのUltra SCSIコネクタも用意されているので、低速なCD-ROMなどをつなぐこともできる。19160のコントロールLSIは、AIC-7892Bというチップが使われている。ほぼ同性能の同社の29160Nという製品があるが、カタログを眺めてみる限りではカード自体は同じもの（？）のように思えた。29160NにはNetwareやSCO UnixWareに対応するドライバが付属するのと、付属ケーブルの長さが違っていてドライブ5台まで接続できるといった点がサーバ用ということなのだろう。

ハードディスクはもちろんCD-ROMもSCSI

SCSIハードディスクは、サーバ用途が中心なので高速、大容量のモデルが多い。**表3**に代表的なSCSIハードディスクをまとめておく。IDEのハードディスクと比べると、回転数が速いことと、平均シーク時間が短いことが、SCSIディスクの優位点だ。

購入したのは、4万8500円とまあまあの価格だったQuantumのATLAS V 18.3S（18.3Gバイト）である（**写真9**）。回転数7200RPM、平均シーク時間6.3m秒、バッファ容量4Mバイトで、1プラッタあたり9.2Gバイトと高密度なため、最大29Mバイト／秒のスループットを持つ。

せっかくなのでCD-ROMもSCSI接続の東芝製40倍速XM-6401TA（8800円）を選択した（**写真10**）。価格の面で、IDEのCD-ROMよりSCSI接続のCD-ROMはあまり売れない製品になってしまった。あと少し予算を足せばCD-R／RWやDVD-ROMドライブを検討してみるのもよいだろう。ただし、DVD-VideoをLinuxで観ることは現時点ではあきらめておいたほうがよい。

ネットワークカードは、100BASE対応で、DEC21140チップ採用のプラネックスのENW-9501-F（4750円）にした。DEC21140は古くからあるチップで、これを自動認識しないディストリビューションはないといっても過言で

名称	バス幅	最大データ転送速度
SCSI		
Fast SCSI	8ビット	10Mバイト／秒
Fast Wide SCSI	16ビット	20Mバイト／秒
Ultra SCSI	8ビット	20Mバイト／秒
Ultra Wide SCSI	16ビット	40Mバイト／秒
Ultra2 SCSI	8ビット	40Mバイト／秒
Ultra2 Wide SCSI	16ビット	80Mバイト／秒
Ultra160 SCSI	16ビット	160Mバイト／秒
IDE（ATAPI）		
PIO転送（Mode2）		8.3Mバイト／秒
PIO転送（Mode4）		16.7Mバイト／秒
DMA転送（Mode2）		16.7Mバイト／秒
UltraDMA/33		33.3Mバイト／秒
UltraDMA/66		66.6Mバイト／秒

表2　SCSIとIDEのデータ転送速度

写真11　ケースに組み込んだところ

写真12　完成したハイエンドマシン
ペンギンケースかわいいでしょ。

写真13　Logicool iK-50
USB接続も可能な日本語109キーボード。付属のPS/2キーボード変換アダプタを付ければ、ふつうのPS/2キーボードとして使える。

写真14　マイクロソフト インテリマウス エクスプローラ
マウスの底に光学式センサーを搭載し、ボールを不要にした。USB-PS/2変換アダプタが付属している。

はないくらい標準的なものだ。

なぜかペンギンケースにしちゃった

ケースは、**写真12**をご覧の通りフロントパネル部にペンギンをあしらった「ペンギンケース」（8800円）で、Linuxマシンであることを見た目で主張してみた。大きさはミドルタワー、5インチベイ×2、3.5インチベイ×3で、ペンギンのおなかの部分を押すと前に開いてフロッピーが10枚ほど収納できるようになっている。フロッピーを取り付ける場所がペンギンの頭の上と、口の部分の2カ所あるのはご愛敬か。実際には口の部分にフロッピードライブを取り付けた。

ペンギンケースに付属の電源は、HEC-250GR（Heroichi Electronic）というATX 2.01準拠の250Wだった。Athlonを使う場合、AMDが発表している推奨電源のリストがWebページ（http://www1.amd.com/athlon/power/）に掲載されている。このHEC-250GRもそこに掲載されているので安心して使えるだろう。

しかし、見栄えで決めてしまったが、発熱量の多いSCSIハードディスクを使用するので、長期にわたって運用するのなら、放熱のよいアルミケースやフルタワー型のほうがよかったかもしれない。

キーボード、マウスは、USB対応のものを購入した（**写真13、14**）。どちらも付属のUSB-PS/2変換コネクタを付けることで、PS/2インターフェイスで使用することができる。マウスはボールをなくして光学式で移動を検知するマイクロソフト インテリ エクスプロー

製品名	容量	回転数	平均シーク時間	バッファ容量	インターフェイス
IBM					
UltraStar 36LZX（DDYS36950）	36.7Gバイト	10000RPM	4.9m秒	4Mバイト	Ultra160+ SCSI
UltraStar 18LZX（DMVS18V）	18.3Gバイト	10000RPM	4.9m秒	2Mバイト	Ultra2 Wide SCSI
UltraStar 18ES（DNES318350WLVD）	18.2Gバイト	7200RPM	7.0m秒	2Mバイト	Ultra2 Wide SCSI
Quantum					
ATLAS 10K II	18.4Gバイト	10000RPM	4.7m秒	8Mバイト	Ultra160 SCSI
ATLAS 10K	36.4Gバイト	10000RPM	5.5m秒	2Mバイト	Ultra160 SCSI
ATLAS V	36.7Gバイト	7200RPM	6.3m秒	4Mバイト	Ultra160 SCSI
Seagate					
Cheetah 36LP（ST336704LW）	36.7Gバイト	10000RPM	5.2m秒	4Mバイト	Ultra160 SCSI
Cheetah 18XL（ST318404LW）	18.4Gバイト	10000RPM	5.2m秒	4Mバイト	Ultra160 SCSI
Barracuda 18XL（ST318436LWV）	18.4Gバイト	7200RPM	5.9m秒	2Mバイト	Ultra160 SCSI

表3　代表的なSCSIハードディスクドライブ

ラだ。

結局、全部で22万2700円と予算を大幅にオーバーしてしまった。

さっそく組み立てる

マシンの組み立ての一般的な知識は前パートまでを参考にしてもらうとして、ここでは特別なパーツについて説明しておこう。

まず、ハードディスクとCD-ROMをケースに取り付ける前に、SCSI ID（0〜15までの機器固有のID）を設定する。SCSIインターフェイスのドライブならば、ID0〜ID3の4つのジャンパがあるはずだ。ID0が下位ビット、ID3が上位ビットで、解放状態なら0、ジャンパを付けると1になり、2進法で計算して番号をセットする。詳しくはそれぞれのマニュアルやWebサイトから資料を見つけて、間違えないように設定していただきたい。

このシステムでは、CD-ROMを1に、ハードディスクを2に設定した。なお、SCSIカード自身にもID番号があり、通常は7になっている。また、付属のUltra160ケーブルは終端にターミネータが付いているので、ドライブ自身のターミネータはオフにすること。SCSIを使ったシステムで、ときどき調子が悪いといったトラブルの原因は、ターミネータ周りであることが多いので気をつけよう。

また、複数のハードディスクを接続する場合、IDの小さいドライブから順番にドライブ名が割り当てられるので、ブート用のハードディスクを小さいID番号にしておく。

とりあえずLinuxをインストール

組み上がったところで、さっそくLinuxのインストールに取りかかる。TurboLinux Workstation日本語版6.0（以下Turbo 6.0）のCD-ROMを、CD-ROMドライブに挿入して、そこから起動する。

Turbo 6.0のインストーラは、SCSIコントローラとして、aic7xxxが見つかったと報告してくるので、そのままインストールを続ければよい。このSCSIカードにはAIC-7892Bというチップが使われていて、正しく認識されている。

IDEのハードディスクと違う点は、デバイス名が/dev/hdaから/dev/sdaと変わることくらいで、あとはいつもと同じ手順で行えばよい。

起動時に出力されるメッセージは、dmesgコマンドで見ることができる。**リスト1**はその一部だが、SCSIカードと接続されているデバイスが正しく認識されていることがわかる。

今回のパーツの組み合わせでは、X

CPU	Pentium III 733MHz（FSB 133MHz）
メモリ	256MバイトSDRAM PC133
ハードディスク	18.3Gバイト（Ultra160 SCSI）
CD-ROM	40倍速SCSI CD-ROM
フロッピードライブ	3.5インチ（1.44Mバイト）
グラフィックスカード	Matrox Millennium G400 DH（32Mバイト）
ネットワーク	10/100BASE-TX（DEC 21140AF）
SCSIコントローラ	Adaptec SCSIカード19160（Ultra160対応）
キーボード	Logicool iK-50日本語109（USB対応）
マウス	マイクロソフト インテリマウス エクスプローラ（USB対応）
ケース	ペンギンケース（ATX2.01、250W）
OS	Linux各種ディストリビューション

表4　完成したハイエンドマシンのスペック

▼リスト1　起動時のメッセージ（SCSIデバイスを自動認識）

```
(scsi0) <Adaptec AIC-7892 Ultra 160/m SCSI host adapter> found at PCI 0/15/0
(scsi0) Wide Channel, SCSI ID=7, 32/255 SCBs
(scsi0) Downloading sequencer code... 392 instructions downloaded
scsi0 : Adaptec AHA274x/284x/294x (EISA/VLB/PCI-Fast SCSI) 5.2.0/3.2.4
       <Adaptec AIC-7892 Ultra 160/m SCSI host adapter>
scsi : 1 host.
(scsi0:0:1:0) Synchronous at 20.0 Mbyte/sec, offset 16.
  Vendor: TOSHIBA   Model: CD-ROM XM-6401TA  Rev: 1001
  Type:   CD-ROM                             ANSI SCSI revision: 02
(scsi0:0:2:0) Synchronous at 160.0 Mbyte/sec, offset 63.
  Vendor: QUANTUM   Model: ATLAS_V_18_WLS    Rev: 0200
  Type:   Direct-Access                      ANSI SCSI revision: 03
Detected scsi disk sda at scsi0, channel 0, id 2, lun 0
scsi : detected 1 SCSI disks total.
SCSI device sda: hdwr sector= 512 bytes. Sectors= 35861388 [17510 MB] [17.5 GB]
 sda: sda1 sda2 sda3 sda4 < sda5 sda6 sda7 >
```

Window Systemまで問題なく起動された。さすがにCPU、ハードディスクが高速なため、インストールもすいすいと進む。占有ディスク容量の少ないディストリビューションなら、前準備を除いたファイルのコピーが行われている時間は10分ほどだった。

今回の一番の強化点であるSCSIハードディスクの性能を、hdparmコマンドで計ってみたところ約28Mバイト／秒の転送スピードだった。ちなみに編集部にあったSCSIハードディスク（IBM Ultrastar 18LZX 9.1Gバイト）をちょっと借用し、ソフトウェアRAIDでストライピングを組んでみたところ、約52Mバイト／秒の好成績が得られたことを報告しておく。

画面1　カーネル設定画面

USBキーボード、マウスは使える？

現在のLinuxディストリビューションは、安定版カーネルである2.2系を使用しているものがほとんどだ。その2.2系カーネルは、標準ではUSBをサポートしていないので、普通はUSBを使えない。

しかし、開発版カーネル2.3系でUSBドライバの実験が行われており、もうじきリリースされる新カーネル2.4で正式サポートの予定だ。それを待つのもよいのだが、今回は最新ディストリビューションで実験してみた。

今月号の付録CD-ROMに収録されている「Corel LINUX OSエンハンスメント」には、USBドライバが入っているようで、インストール時点からあっさりと自動認識され、問題なく動作した。

では、ほかのディストリビューションでもうまくいかないものかと試してみたのだが、Red Hat Linux 6.2JやTurbo 6.0などはダメだった。何も入力できないのではどうしようもない。うーん残念。PS/2変換コネクタを使うしかないか。

VIA Apollo Pro133AでUltraDMA/66を試す

今回は、IDEを使用していないため必要ないが、現在市販されているIDEハードディスクは、ほとんどUltraDMA/66対応になっている。しかし、UltraDMA/66の転送モードを使うには、PC側のインターフェイス、すなわちマザーボード上のチップセットが、UltraDMA/66に対応している必要があるうえ、Linuxのドライバがそのコントローラに対応していなくてはならない。未対応ならUltraDMA/33やCPUによるPIO転送になる。

Intelの440BXというチップセットは、UltraDMA/33までしか対応していないため、それを使っているマザーボードの中には、UltraDMA/66対応のコントロールLSIを別に搭載しているものもある。また、UltraDMA/66対応のIDEコントローラカードも販売されているので、必要ならばそれを増設するという手もある。

購入したMS-6309マザーボードは、チップセットにVIA Apollo Pro133Aを採用していて、UltraDMA/66をサポートしている。そこで、実際にUltraDMA/66対応のハードディスクを接続して、Turbo 6.0で試してみたのだが……。

▼リスト2　標準カーネルのIDEデバイスの認識

```
RAM disk driver initialized:  16 RAM disks of 4096K size
Uniform Multi-Platform E-IDE driver Revision: 6.20
VP_IDE: IDE controller on PCI bus 00 dev 39
VP_IDE: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0xffa0-0xffa7, BIOS settings: hda:DMA, hdb:pio
    ide1: BM-DMA at 0xffa8-0xffaf, BIOS settings: hdc:pio, hdd:pio
hda: IBM-DPTA-351500, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: IBM-DPTA-351500, 14324MB w/425kB Cache, CHS=1826/255/63
Floppy drive(s): fd0 is 1.44M
```

▼リスト3　VIA82CXXXを有効にしたカーネル2.3.99-pre5の起動画面

```
Uniform Multi-Platform E-IDE driver Revision: 6.30
ide: Assuming 33MHz system bus speed for PIO modes
VP_IDE: IDE controller on PCI bus 00 dev 39
VP_IDE: not 100% native mode: will probe irqs later
VT 82C691 Apollo Pro
Split FIFO Configuration:  8 Primary buffers, threshold = 1/2
                           8 Second. buffers, threshold = 1/2
    ide0: BM-DMA at 0xffa0-0xffa7, BIOS settings: hda:DMA, hdb:pio
ide0: VIA Bus-Master (U)DMA Timing Config Success
    ide1: BM-DMA at 0xffa8-0xffaf, BIOS settings: hdc:pio, hdd:pio
ide1: VIA Bus-Master (U)DMA Timing Config Success
hda: IBM-DPTA-351500, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: 29336832 sectors (15020 MB) w/425KiB Cache, CHS=1826/255/63
```

ブート時に出力されるメッセージのうち、IDEコントローラの部分を取り出したのが**リスト2**である。3、4行目の先頭に「VP_IDE」とあり、VIAのコントローラと認識されていることがわかる。しかし、4行目のそのすぐ横に「not 100% native mode:」と書かれているのを見ると、きちんと動作しているのか不安になるところだ。

しかたがないので、カーネルを書き換えて強制的にドライバを組み込んでみた。X上のktermから、

```
# cd /usr/src/linux
# make xconfig
```

として、カーネル設定画面を起動する。「Block Device」セクションで、スクロールしていくと真ん中あたりに、「VIA 82CXXX chipset support (EXPERIMENTAL)」が見つかるので、「y」をチェックし、カーネルイメージに直接組み込むことにする。ここでEXPERIMENTALと書かれている通り、実験段階ということなので、不具合が発生する可能性もある。

あとは、カーネルをコンパイルし、lilo.confを新しいカーネルからブートするように書き換える。そして、そのカーネルで起動したところ「VT 82C691 Apollo Pro Chipset Core ATA-33」と表示され、Turbo 6.0標準のカーネル2.2.13に入っているVIAのIDEドライバは、UltraDMA/33対応のものだったことが判明したのだった。

わざわざこのUltraDMA/33対応ドライバを使っても、標準のドライバに比べて性能差はないと思われる。標準ドライバで、ディスクの読み書きに不自由したり、不具合があったりしなければ、そのまま使っていて差し支えないだろう。

カーネル2.4を一歩先取りして味わう

開発版カーネルの2.3系はいよいよ最終コーナーに差し掛かってきたようだ。The Linux Kernel Archives (http://www.kernel.org/) を見ると、2.3.99-pre5が最新だった。ミラーサイトからFTPでダウンロードし、それを組み込んで最新カーネルの新機能をちょっとだけ味わってみた。

先ほどと同様に、make xconfigを実行し、今度は「IDE, ATA and ATAPI Block devices」セクションで、同じく「VIA82CXXX chipset support (EXPERIMENTAL)」を**画面1**のように「y」にしてから、カーネルの作成し直しだ。新しい2.3.99-pre5カーネルで起動したときの結果が**リスト3**である。今度は「VT 82C691 Apollo Pro」の表示が出力された。

こういうテストをするときに実感するのが、ハードディスクのスピードだ。カーネルのmakeが目にも止まらぬ速さで進んでいくのを見ていると、高いお金を出してSCSIハードディスクを買った甲斐があるってものだ。

Column

メモリが正しく認識されない?

実は、このマシンにLinuxをインストールしたあと、X用のソフトウェアをいくつか動かしていたら、なぜかすごく遅いことに気がついた。よくよく調べてみると、メモリを256Mバイトも積んでいるのに、64Mバイトしかないと認識されていた(リスト参照)。

おかしいと思って、最新ディストリビューションであるTurboLinux Workstation日本語版6.0やRed Hat Linux 6.2Jもインストールしてみたのだが、どれも同じ結果だった。

確かカーネル2.2系ではこんなことは起こらないはずでは……と、つぶやいていても解決しない。/etc/lilo.confに、

```
append="mem=256M"
```

を書き加えて、Linuxカーネルにメモリサイズを明示するようにした。なお、書き換えたあとで、

```
# /sbin/lilo
```

をし忘れることがよくあるのでご注意を。原因不明だが、そういうときにもあわてず騒がずに済むように、こういう手順も覚えておこう。

▼リスト　メモリサイズを指定する。

```
Linux version 2.2.13-33 (product@build.turbolinux.co.jp) (gcc driver
version 2.95.2 19991024 (release) executing gcc version 2.7.2.3) #1
Sun Mar 19 15:01:16 JST 2000
Detected 733345051 Hz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 730.73 BogoMIPS
Memory: 63588k/66556k available (1064k kernel code, 416k reserved,
1052k data, 64k init)
```

自動認識では64Mバイトと表示される

lilo.confに「append="mem=256M"」を追加して、256Mバイトと認識させた

```
Memory: 257240k/262144k available (1068k kernel code, 412k reserved,
2988k data, 64k init)
```

これも自作PC!?

卓状コンピュータ「オオタク01」

～コンピュータになった机の王様～

「卓上」ではなく「卓状」である。ファニーな外観そのままの素直にして絶妙な形容だ。家具デザイナーによって設計されたこの物体は、あくまで「コンピュータになったテーブル」なのだろうか。そして「オオタク」とはやはり……。

制作者の原ななえさんに、まずはネーミングについてうかがった。

日本中のオタクに敬意をはらってオタクにちなんだ名前にしたかった、というのもあります。

卓状コンピュータというのは、卓の形をしたものの中で一番進化しているという意味で「卓の王様＝王卓」でもあり、すばらしい技術者達の集まった大田区のように、技術の結晶であってほしいという意味もある。どの意味にもとれるようカタカナにしました。

では、卓状コンピュータというコンセプトはどのようにして生まれたのだろうか。

もともとオオタク01は、旭川のエキスパンドというコンピュータパーツショップの横山康也さんから、「コンピュータが組み込まれている机がほしい」と持ちかけられたのが、はじまりなんです。横山さんには開発も手伝っていただきました。

予想に反して出発点はコンピュータだったのだ。テーブル然とした雰囲気は原さんの「主張」なのだろう。

オオタク01は、コンピュータとしてちゃんと動作する。撮影のために、お借りした際の構成もCeleron 500Mhzを搭載し、あなどれないスペックを持ったマシンという一面を見せていた。前面上部に取り付けられたカメラもダテではない。

背面パネルを開けると、その裏にマザーボードが装着されている。コード類の取りまわしに、制作時の苦労がうかがえる。

じゃあ、とりあえず作ってみようということになって、市販のコンピュータケースをばらす、モニタをすっぽり

キーボードは引き出しに収納してしまっておける。

ややモニタから遠くなるが、このままでも操作可能。

上部中央にあるのが内蔵のカメラ。

包める箱を作るときて、その次に5インチのケース。これが、どこにも売ってなくって。

さんざん探したあげく、長野県の第一工業さんとカワベ工業さんにご協力いただいて、電話とファクスだけで5インチケースをムリ言って作ってもらいました。

マザーボードからの電源や、電源スイッチなど長さが足りなかったため延長しまくりました。あとから気づいてアースをとって……。モニタも大きくて納まりが悪くて。結構キーボードやマウスなどもじゃまで、とにかくいろんなものの納まりに苦労しました。

話の中に登場する特注の5インチケースは、前面パネルに並んだドライブベイとして使用されている。1つのケースに2つの5インチドライブが収まるようになっているが、上側のドライブが乗るレールがないため完全には固定できない。このあたりは改善の余地ありである。「今回は値段を優先しました」ということなので、いたしかたない面もあるのだろう。

その値段であるが、Celeron 500MHzのWindows 98モデルで、26万8000円となっている。先頃、札幌のマルサデパートにオープンした期間限定（夏まで）のLimitShop「AGROOM」にて、5月中旬より展示販売されるそうだ。ちなみに現在の原さんの活動拠点は旭川である。

01と付くからには、シリーズ化の展望があるのだろうか。このあたりについてもうかがってみた。

オオタクはシリーズ化します。もう「オオタク02」設計中です。小さく作るつもりです。02が出た時に01の良さがわかってくる。そんな02になる予定です。02とともに01のマイナーチェンジも行っていきたいですね。

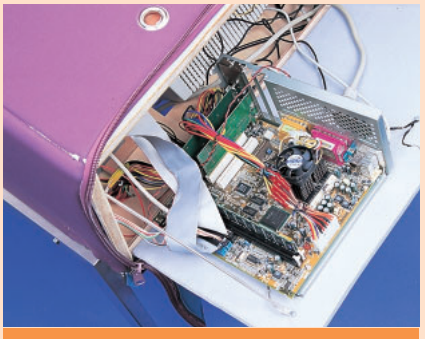
02、03、04、...0Aと16進法でシリーズ展開していく予定になっています。いずれもコンピュータケースですので、大型も小型もモバイルにもチャレンジしようかと思っています。Morphy Oneにもチャレンジしたいです。

実は、年に2回ほど毎年、東京で展示会をしているんですが、次は10月頃に行われる「HAPPENING」というイベントに参加できれば、その場でオオタク02を発表したいと思っています。まだ日程や場所などは決まっていないんですけど。

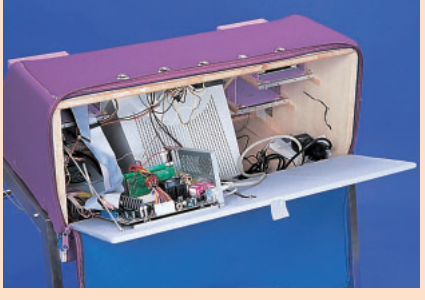
最後に、ふだんのコンピュータとの接し方、そしてLinuxに対するイメージをうかがった。

主にメールのやりとり、顧客管理、ハガキやカタログなど印刷物の製作に使ってます。いまのところ家具の設計には使ってませんが、プレゼンなどに使えるようにと考えているところです。

Linuxは具体的にいうとフォントなどが充実していないので、まだ「使える！！」といった感じではないのですが、目指すところが公表されているぶん未来を感じます。



組み立ては、このように背面パネルを開けて行う。



＜profile＞
原ななえ（あぐら家具企画代表）
東京生まれ
武蔵野美術大学　油絵学科　抽象絵画コースに学び、立体作品の製作にとり組む。卒業後、北海道旭川市の家具メーカー工場勤務。1993年には、2級木工技能士資格を取得。1995年、「あぐら家具企画」（北海道旭川）設立。オリジナルの家具を、デザイン、製作、販売している。最近は、舞台家具なども手がける。

シープ（左）トラベル（右）、どちらも原さんの作品。原さんによるとトラベルは「移動可能な椅子」であるとのこと。別売りのショルダーベルトを装着すれば、その名のとおり旅行にも連れていける。

自作PC完成！～ベンチマークするということ～

文：編集部　Text : Linux magazine

はじめて自作したPCに電源を入れて、無事に立ちあがったときの感覚というのは独特のものだ。電源が入り、フロッピー、CD-ROM、ハードディスクの各ドライブが音を立てはじめる。やがてBIOSの「ピー」とか「ピピッ」とかいうチェック音が聞こえる。OSは入ってないので、起動できるシステムがないことを通知するメッセージが出て止まってしまう。でもなんだか、ホッとしたような、おもはゆいような不思議な感じに包まれる。

「自作」から生まれるもの

考えてみれば、パーツを選び、それを組み立て、OSをインストールするいう作業は、布地を選んで洋服を作ったり、木材を買ってきて本棚を作ったりすることと少しも変わるものではない。素材選びから、作る過程の楽しさ、そして作り上げた「もの」への愛着は、どの行為にもあてはまるだろう。つまりは、コンピュータを組み立てるという行為も、洋裁や日曜大工と同じレベルで趣味として成り立ち得るのだ。

プログラム開発やWebサーバ構築といったハッキリした目的を持っている場合を別として、コンピュータそのものやOSに興味を持っているとか、とりあえずメールを使っているが、あとはなにをすればよいのか思い浮かばないない、というのであれば、ぜひ一度、PCの自作に挑戦してみてほしい。なにも、すべてのパーツを揃えて一から組み上げる必要はない。ハードディスクやメモリを増設するといったことからはじめてみよう。

たとえ買ってきたパーツが動作しなかったとしても、設定を見直したり、OSを再インストールしてみたり、とやっているうちに自然とコンピュータの仕組みやOSの動作に対する理解も深まっていくはずだ。理解が深まれば、コンピュータを何のためにどう使たいのか、ということへの回答が得られるかもしれない。同時に、苦労したぶんだけ完成したマシンへの愛着も強くなるだろう。

ベンチマークへの欲求

PCの自作がそれほど特殊なことではなくなりつつある現在でも、「自作マニア」と呼ばれる人達がいる。その中に、「ベンチマーク族」とも呼べるような一群が確実に存在している（しかもかなりの多数派である）。さらにその中に、CPUをオーバークロックでブンまわすことに魂を奪われた「クロックアップ派」、金に糸目をつけず、とにかく最新最強のパーツを求めてやまない「最新技術集約派」、ガラクタかと思えるようなパーツを使って最大限の性能を生み出そうとする「錬金術派」、己の英知のすべてをそそいでベンチマークの結果のみを追求する「純粋ベンチマーク派」など、さまざまなタイプが存在する。

タイプは異なっていても、ベンチマークテストのため「だけ」にマシンを組み上げるという点でベンチマーク愛好家たちの目的は一致している。作り上げたマシンの性能を知りたいというのは、彼らにとって当然の心理なのだ。というより、知らずしてどうするというのが彼らの主張だ。

ここまでいかなくても、自分のマシンの性能を知ってみたいというのは誰もが持っている潜在的な願望なのかもしれない。ベンチマークはその欲求を満たしてくれるメソッドなのだ。検索サイトへ行って「ベンチマーク」でサーチをかけてみると、かなりのヒット数があるはずだ。彼らの情熱に触れれば、新たな刺激が得られるだろう。

ベンチマーク結果発表

ここで、今回の特集で作成した4台のベンチマーク結果をお知らせしておこう。それぞれのマシンで、CPU、メモリ、ビデオカードなどのハードウェア構成がまったく違うため、これらの数値を単純に比べることにあまり意味はない。ここで紹介するベンチマークの数値を基準にして、自分のマシンの性能レベルを確認してみてほしい。「同じチップセットを使ったマザーボードなのに、うちのマシンはディスクが遅いな」とか「適当に寄せ集めて作ったのに結構いい結果が出たな」といったふうに思っていただければよい。

意外に性能が低いと思ったら、改めてハードウェア構成とOSの設定を見直してみよう。バランスの悪い構成になっているのかもしれないし、自分のマシンのポテンシャルを十分に引き出し

ていないことがあるかもしれない。

特に、ある程度の最適化が自動的に行われるWindowsと違って、Linuxではユーザーにまかされている。特集の中でも触れたように、通常にインストールしただけの状態では、動作していない機能も多い。面倒といえば面倒なのだが、設定を変えることで、どのような効果が得られるかを知ることができる。UltraDMAの有効化のようにベンチマークの結果にあきらかに反映される機能もある。最適化の効果が、目に見えるかたちでわかるのは結構うれしいものだ（今回のテストでも、ハードディスクのベンチマーク計測に際してUltraDMAを有効化している）。

ベンチマークツール

今回、ベンチマークテストに使ったツールを紹介しておこういずれも本誌のバックナンバーに一度は登場したおなじみのベンチマークツールである。

	配布元URL	テストオプション
午後のこ～だ ver 2.30Final	http://homepage1.nifty.com/herumi/	-test -nopsy
Bonnie	http://www.textuality.com/bonnie/	-s 384
HDBENCH clone Ver 0.14.1	http://www.enjoy.ne.jp/~gm/program/hdbench/	デフォルト設定

表1　ベンチマークツールの配布元と設定オプション

使用する前には、必ず各配布元の注意事項などに目を通しておくこと（配布元と今回のテストで使用したオションについては**表1**を参照）。

なおテスト環境には、Red Hat Linux 6.2Jを使用した。

■午後のこ～だ ver 2.30Final

MP3のエンコーダ。専用のベンチマークツールではないが、オプションとしてベンチマーク機能を持っている。MP3のエンコード処理速度を計測することで、CPUの処理能力を知ることができる。Pentium IIIのSSE、AthronのEnhanced 3D Now!にも対応する。

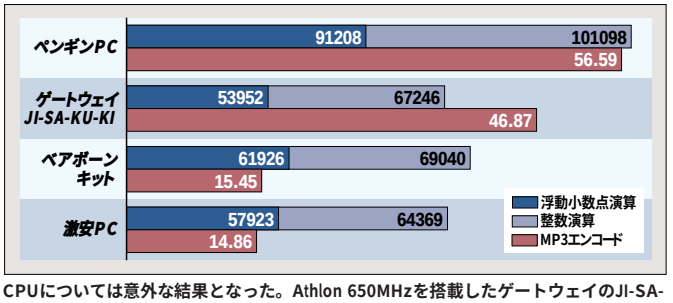
■Bonnie

ハードディスク専用ベンチマークツール。ディスクに対して連続的な読み込み／書き込みを行って、その速度を計測する。数多くのオプションがあり、さまざまな条件下でのテストが可能。

■HDBENCH clone Ver 0.14.1

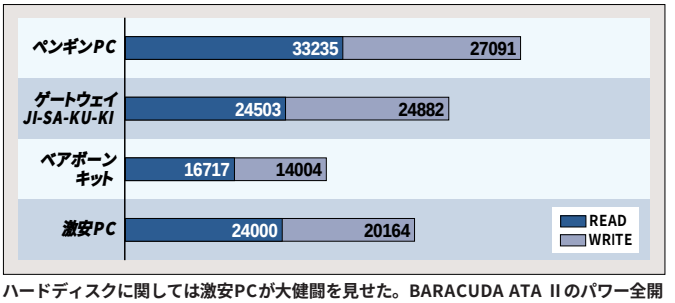
総合ベンチマークソフト。クローンの名は、Windows版の同名ソフトにインターフェイスがそっくりであることからきている。CPU、ビデオ、ディスクの各パートごとに、浮動小数点演算、スクロール、読み取りなど、いくつかのテスト項目があり、総合的なマシンの性能を把握できる。

グラフ1



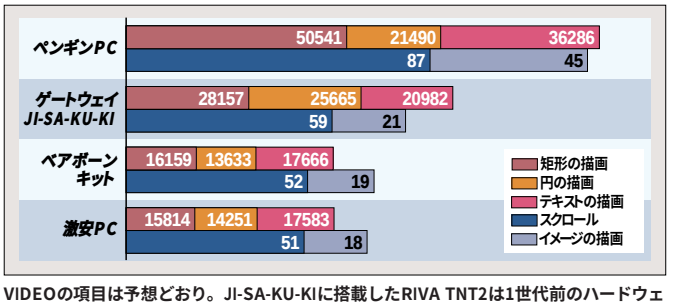
CPUについては意外な結果となった。Athlon 650MHzを搭載したゲートウェイのJI-SA-KU-KIがHDBENCH Cloneの浮動小数点演算の項目で最下位、整数演算の項目でも3位に低迷したのだ。残念ながら原因は究明できなかった。MP3のエンコードは、ほぼ予想どおりの結果が出ている。

グラフ2



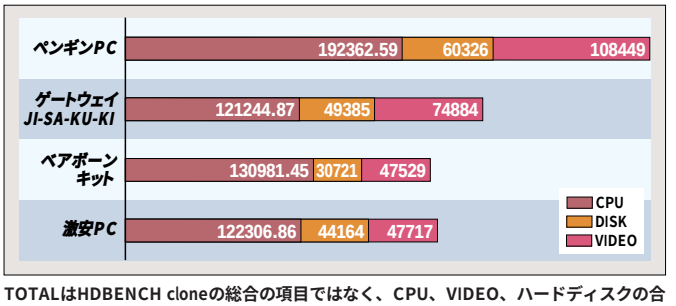
ハードディスクに関しては激安PCが大健闘を見せた。BARACUDA ATA IIのパワー全開と言いたいところだが、これは複数回行ったテストの中で時折桁外れに高い数値を示したため。何故か数値のバラツキが大きかった。低いほうでは、ベアボーンキットとそれほど変わらない値もあった。

グラフ3



VIDEOの項目は予想どおり。JI-SA-KU-KIに搭載したRIVA TNT2は1世代前のハードウェアであることを考えると、健闘したといっていいだろう。i810チップセットに統合されたグラフィック機能もそれなりの性能を持っていることを示した。

グラフ4



TOTALはHDBENCH cloneの総合の項目ではなく、CPU、VIDEO、ハードディスクの合計値。こうしてみると、あたりまえではあるが、ペンギンPCの突出ぶりがわかる。JI-SA-KU-KIが伸び悩んだのは、やはりCPUでの低迷が響いた。

あなたのネットワークはまだ10Mですか？

100BASE イーサネットカード&ハブ

アキバで集めたネットワークカード11機種をLinuxで使う 爆安スイッチングハブであなたのLANをパワーアップ！

文：リナックス・マガジン・ラボ
Text : Linux magazine Lab.
Photo : Shuichi Mito (Dee)

イーサネットの基礎知識

Linuxは、比較的古くからTCP/IPをサポートしていたこともあり、他のUNIX系OSと同様にネットワークとの親和性は高い。近年、イーサネット機器の低価格化は著しく、企業や大学のみならず家庭でも100Mbpsの高速イーサネットによるLANを構築することが珍しくなくなった。そこで今回は、イーサネットの概略とあわせ、現在販売されているNIC（Network Interface Card）11機種と、スイッチングハブ3機種を紹介する。LinuxでLANを構築する際に参考にしてほしい。

実際の機器を紹介する前に、イーサネットの概略を簡単に解説しておこう。

イーサネットの歴史

イーサネットは、1973年にXeroxのPARC（Palo Alto Research Center）で発明された。当初の伝送速度は2.94Mbpsだったが、1980年にDEC、Intel、Xeroxの3社によって伝送速度10Mbpsの規格、Ethernet 1.0（DIX 1.0）が発表された。Ethernet 1.0は、1982年にEthernet 2.0（DIX 2.0）となり、翌1983年にIEEE（Institute of Electrical and Electronics Engineers、米国電気電子学会）がEthernet 2.0をもとにIEEE 802.3として標準化した。これが10BASE-5だ。厳密にはEthernet 2.0とIEEE 802.3は異なる規格だが、多くの場合で互換性がとれるようになっている。

のちにIEEE 802.3標準として、10BASE-2、10BASE-Tなどが追加された。さらに、1995年には、伝送速度が100Mbpsの100BASE-Tも標準化されている。

イーサネットの物理メディア

現在よく使われているイーサネット規格は、伝送速度が10Mbpsの10BASE規格と伝送速度が100Mbpsの100BASE規格に大別される。それぞれ、接続方式などの違いによりいくつかの種類があるので、比較的一般的なものについて説明しよう。

10BASE-5は、直径12mm、インピーダンス50Ωの同軸ケーブルを媒体として使う。これは、従来のEthernet規格で規定されているものだ。ケーブルが太いため、伝送特性が良く、1本のケーブルで接続できる長さ（セグメント長）は500mと長いが、ケーブルが高価で、引き回しが面倒だったため、現在ではあまり使われない。

10BASE-2は、直径5mm、インピーダンス50Ωの細い同軸ケーブルを使う。最大セグメント長は185mだ。

10BASE-5と10BASE-2では、接続されているすべてのマシンで1本のケーブルを共有するため、断線が起こるとネットワーク全体がダウンするという欠点がある。

10BASE-Tでは、高価な同軸ケーブルではなく、安価なツイストペアケーブルを媒体とする。ツイストペアケーブルとは、＋と－の極性の異なる2本の線を対にしてよじったもので、線に電流が流れることで発生する電磁ノイズを線をよじることでキャンセルさせるものだ。ツイストペアケーブルには、電気特性により1～5のカテゴリが定められており、数字が大きいものほど高い周波数（すなわち高い伝送速度）に対応できる。

10BASE-Tで利用されるのは、カテゴリ3以上の2対4芯シールドなしツイストペアケーブル（UTP、Unshielded Twisted Pair）で、2対の線をそれぞれ送信と受信に使う。ケーブルの両端は俗にRJ-45と呼ばれる8極のモジュラコネクタで末端処理を行う（**写真1**）。接続方法も10BASE-5や10BASE-2とは違い、ハブを介して各マシンを接続する（**図1**）。最大セグメント長は100mだ。現在、もっとも普及しているのがこの10BASE-Tだろう。

100BASE-T規格の中でも、特に普及が進みつつあるのが100BASE-TXだ。この特集で紹介するのもすべて100BASE-TX対応製品となる。100BASE-TXの接続形態は10BASE-Tと同じだが、ケーブルはカテゴリ5の2対4芯UTPケーブルかシールド付き2対4芯ツイストペアケーブル（STP、Shielded Twisted Pair）を用いる。家庭やオフィスでSTPを使うことは稀

写真1　8極モジュラコネクタ（通称RJ-45）
10BASE-T／100BASE-TXでは、8極のうち4極（2対4芯）だけを利用している。

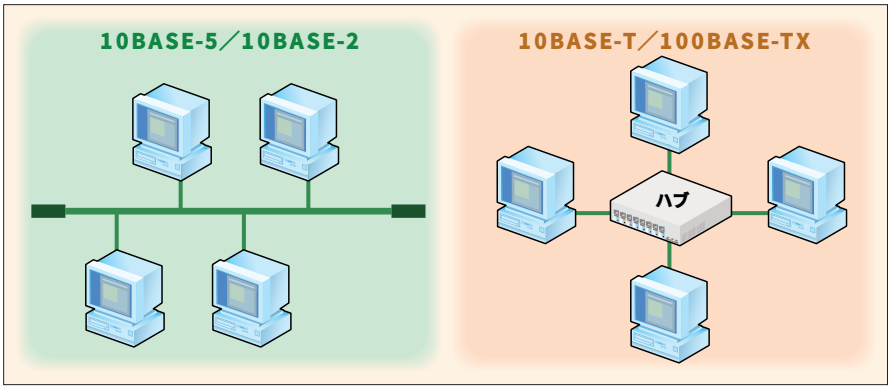


図1　イーサネットによる接続方法の概略
10BASE-5／10BASE-2は、同軸ケーブルを使いバス型配線を行う。10BASE-T／100BASE-TXはハブを介したスター型配線を行う。

だろう。現在、イーサネット用として販売されているUTPケーブルはほぼすべてがカテゴリ5になっている。また、ハブを用いる際には100BASE-TXに対応したものを使う必要がある。

ハブを用いることで、ケーブルの1カ所に断線が起きても、そのケーブルで接続されたマシンがネットワークを使えなくなるだけで、ネットワーク全体がダウンすることはない。

イーサネットのフレーム

イーサネットでは、基本的には複数のマシンで1つの伝送路を共有するバス型のネットワーク構造をとるため、複数のマシンが同時にデータを送信することはできない。このため、各マシンではデータをフレームという小さな単位に区切り、細切れに送受信する。現在のイーサネットでやりとりされるフレームにはEthernet DIXフレームとIEEE 802.3フレームがあり、フレームヘッダに一部違いがあるが、現在の製品では混在使用できる（**図2**）。一般に、TCP/IPでの通信にはEthernet DIXフレームが使われ、その他のプロトコルにはIEEE 802.3フレームが使われることが多い。

フレームの最初にプリアンブルと呼ばれるデータを送信する。これは、データを受信するマシンが同期をとるために使う。Ethernet DIXフレームのプリアンブルは8オクテット（1オクテット＝8ビット）だ。一方、IEEE 802.3フレームでは、7オクテットのプリアンブルに続き、フレームの開始を示すSFD（Start of Frame Delimiter）が1オクテット付く。実は、これらは定義が違うだけで、実際に送られるデータパターンは同一となる。

次に宛先のMACアドレスと送信元のMACアドレスが続く。MACアドレス（Media Access Controlアドレス）とは、あらかじめNICなどの製品の出荷時に焼き込まれているアドレスで、個々の機器でユニークである。したがって、宛先、送信元はMACアドレスで一意に決まる。

Ethernet DIXフレームでは、次の2オクテットはフレームタイプとなる。これは、続くデータ部にどのようなプロトコルのデータが含まれるかを示す。たとえば、フレームタイプが0800Hだとデータ部にはIPが、8137HならIPXが含まれている。一方、IEEE 802.3フレームでは、ここの2オクテットはフレームタイプではなく、データ部の長さを示す。

データ部の長さは46オクテットから1500オクテットの可変長だ。46オクテットに満たない場合はパッド（調整用ビット）と呼ばれる意味のないデータが付加され、46オクテットになるよう調整される。Ethernet DIXフレームのフレームタイプには1500より大きい数を使うことでIEEE 802.3フレームと区別できるようになっている。

最後の4オクテットはFCS（Frame Check Sequence）というデータの誤り検出をするためのデータだ。これは、宛先アドレスからデータの最後までのデータから算出した32ビットのCRC（巡回冗長検査）値で、受信したマシンで同様の計算を行い、その結果とFCSが一致していない場合はそのフレームが壊れたとみなして捨て去る。

通信の仕組み

では実際の通信はどのように行われているのだろうか。イーサネットでは、CSMA/CD（Carrier Sense Multiple Access with Collision Detection）という方式を採用している。

データを送信したいマシンは、伝送路にキャリア（信号）が流れていないかをチェックし（Carrier Sense）、流れていなければ伝送路が空いていると判断してデータを送信する。このデータは接続されているすべてのマシンに受信される。前述のように、フレームには宛先MACアドレスが記入されているので、受信したマシンは自分宛のフレームだけをOSに渡し、自分以外のマシン宛のフレームは破棄する。すべてのマシンがこのような動作をすることで1本の伝送路を共有した多重アクセス（Multiple Access）が可能になるのだ。

ところが、これは理想的に動作している場合の話だ。同時に複数のマシンがデータを送信してしまうこともある。

これを衝突（Collision）といい、衝突したフレームは正しく伝えられない。これでは困るので、データを送信中も伝送路上の信号を調べ、波形が乱れることなくフレームの送信が終わることを確認する（Collision Detection）。衝突を検知した場合は送信を中断し、ジャム信号（停止信号）を送信することで他のマシンに衝突の発生を知らせる。

ジャム信号を送信したあと、すぐにデータの再送を行うと再び衝突が起きる可能性が高いので、一定の時間待ってから送信する。この時間は乱数を加えて決定することで衝突が起きにくくなるよう工夫されている。しかし、連続して16回の再送が行われるとそのフレームは破棄され、その後の処理は上位層（TCP/IPやIPXなど）に任される。

100Mbpsイーサネット

LANの活用範囲が広がると、10BASEイーサネットよりも高速なネットワークが求められるようになった。このため、考案されたのが100Mbpsのイーサネット（Fast Ethernet）だ。これにもいくつかの規格が定められている。

10BASE-Tと高い互換性を保ったまま、信号周波数を6.25倍に、符号化効率を1.6倍に高めることで、伝送速度が10倍の100Mbpsを実現したのが100BASE-TXだ。100BASE-T規格には、このほか4対8芯のカテゴリ3 UTPケーブルを利用する100BASE-T4、光ファイバケーブルを利用する100BASE-FXなどがある。米国に比べネットワークの普及が遅れた日本では、始めからカテゴリ5のUTPケーブルが敷設されることが多かったため、100BASE-T4の需要はあまりない。100BASE-FXは、UTPケーブルでは不可能な遠距離接続や、電磁ノイズの多い工場内での接続に利用されている。

このほかに、データのやりとりを事前に調停し、衝突が発生しないデマンドプライオリティという仕組みを使う100VG-AnyLANという方式もあるが、これもほとんど普及しなかった。100Mbpsイーサネットは、事実上100BASE-TXの独擅場だといえよう。

100BASE-TXは、10BASE-Tとの混在利用も可能であり、ネットワークの中でも特にデータ転送が多い部分の10BASE-T機器を置き換える形で使われ始めた。その後、100BASE-TX機器の導入コストが下がったこともあり、ネットワーク全体を100BASE化することも多くなってきている。

ネットワーク全体が100BASE-TXになると、アクセスの集中するサーバにはより高速な方法で接続したくなるのは当然だろう。IEEEでは、1Gbpsの伝送速度を実現する1000BASE-X、および1000BASE-T（ギガビットイーサネット）も規格化し、対応する製品も販売され始めた。また、10Gbpsイーサネットも標準化に向け作業が進められている。

プラネックスコミュニケーションズの製品に見られるように、100BASE-TXを4本束ね、400Mbpsの論理リンクとする高速化へのアプローチも興味深い。この場合、1本の線に障害が起きても残りの線で通信を継続するフォールトトレランス機能も持つという。

ハブの種類と機能

10BASE-Tや100BASE-TXで3台以上のマシンを接続するときには、ハブと呼ばれる集線装置を用いる。ハブは、大きく分けるとリピータハブとスイッチングハブの2種類があり、最近まではスイッチングハブは高価であったため、リピータハブが広く使われている。しかし、この1年ほどで10BASE-T／100BASE-TXに対応する低価格のスイッチングハブが各社より発売された。1年前には数万円だった8ポートの製品が現在では1万円以下で購入できるようになっている。本特集で紹介するのはこうした低価格のスイッチングハブだ。では、それぞれのハブの機能について見ていこう。

リピータハブはその名の通りリピータ（増幅器）機能を持ち、入力された信号の劣化した部分を整形してすべてのポートに出力する。また、衝突検出機能も持ち、衝突を検知するとすべてのポートにジャム信号を送信する。このようにイーサネットの通信原理に合わせて動作するのがリピータハブということになるが、CSMA/CDには多数

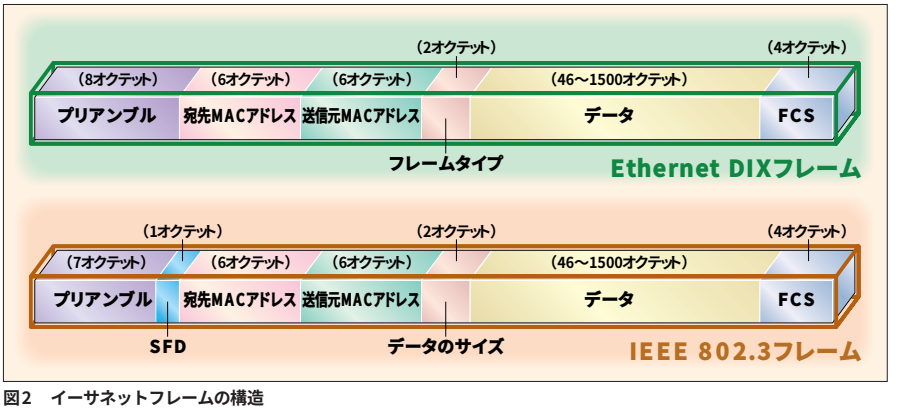


図2　イーサネットフレームの構造
現在のイーサネットでは、プロトコルによってEthenet DIXフレームとIEEE 802.3フレームの2形式が混在利用されることが多い。

のマシンを接続して流れるデータ量が増えると衝突が頻発し、回線の利用効率が悪くなるという欠点がある。この問題は次に説明するスイッチングハブを利用することで緩和できる。

スイッチングハブは、送受信されるフレームから各ポートに接続されているマシンのMACアドレスを学習し、接続ポートが特定できるフレームについては目的のポートにのみ出力する。入力されたフレームをすべてのポートに出力するリピータハブとは違い、関係のないポートには出力しないので、衝突を削減できるうえ、使用されていない別のポート間でも並行して通信を行うことができる（**図3**）。

スイッチングハブにはポート間のフレーム転送方式によっていくつかの種類があるが、最近の10BASE-T／100BASE-TX両対応のスイッチングハブはストア＆フォワード方式を採用している。これは、受信したフレームをいったん内部のバッファメモリに格納し、フレーム長やFCSによるエラーチェックを行う。そしてエラーのなかったフレームのみを目的のポートへ出力する方式だ。信頼性が高く、10Mbps—100Mbpsの速度変換が可能になる。フレーム全体をバッファに格納するための遅延（レイテンシ）が発生するが、運用上問題になることはない。

それに対し、カットスルー方式は、宛先MACアドレスまでを受信するとすぐに目的のポートへ出力を開始する。このためストア＆フォワード方式より遅延は小さいが、エラーチェックが行われないぶん信頼性は低下する。また、速度変換を行うこともできない。このほか、宛先MACアドレス先頭から64オクテットまでをバッファに格納してエラーチェックを行い、目的ポートへの出力を開始することで信頼性と高速性を半分ずつ兼ね備えるモディファイド・カットスルー方式もある。さらに、決して安価ではないが、状況に合わせてストア＆フォワード方式とカットスルー方式を自動的に切り替えるものもある。

その他の付加機能

10BASE-Tや100BASE-TXでは送受信をそれぞれ別のケーブル対で行う。本来なら送信中は受信側の信号を監視し、衝突を検出するのだが、2台のマシンをクロスケーブルで直結した場合は衝突は起きないので相互の送受信を同時に行える。これが全2重通信（Full Duplex）だ。NICとデバイスドライバが全2重通信に対応していれば、送信と受信のそれぞれで10Mbps（10BASE-T）、100Mbps（100BASE-TX）を使えるので、帯域は倍になる。また、スイッチングハブを介して接続した場合にも全2重通信が可能となる。この場合もハブが全2重通信に対応している必要があるが、最近の機種ではほとんどが対応している。

10BASE-Tと100BASE-TX、半2重通信と全2重通信が混在したネットワークでは、接続するマシンやハブ同士は同じ通信モードを使用しなければならない。こうした通信条件を調停する仕組みがNWayオートネゴシエーションだ。接続するマシンやハブがNWayオートネゴシエーションに対応していれば、接続時に通信条件を自動設定することができる。最近のNICやスイッチングハブはほとんどがNWayオートネゴシエーションに対応している。

スイッチングハブの内部には、通常256Kバイト～2Mバイト程度のバッファがあり、未送信のフレームを格納・保持している。しかし、特定のポートへのアクセスが集中すると（特に100Mbpsのポートから10Mbpsのポートへの転送）バッファがオーバーフローを起こすことがある。あふれたデータは捨てられ、その後の処理（再送するかどうかなど）は上位のプロトコルに任されることになるが、データがあふれないよう、フロー制御を行うスイッチングハブも増えてきた。フロー制御を行うハブは、バッファの空き容量が少なくなると、データの送信元に対し送信データを減らすような信号を送り、バッファがあふれるのを防ぐ。半2重通信では架空のジャム信号を送るバックプレッシャー方式、全2重通信ではPauseパケットを送るIEEE 802.3x方式を使う。フロー制御を行うことで、パケットロスを減らし、帯域を効率よく利用できる。

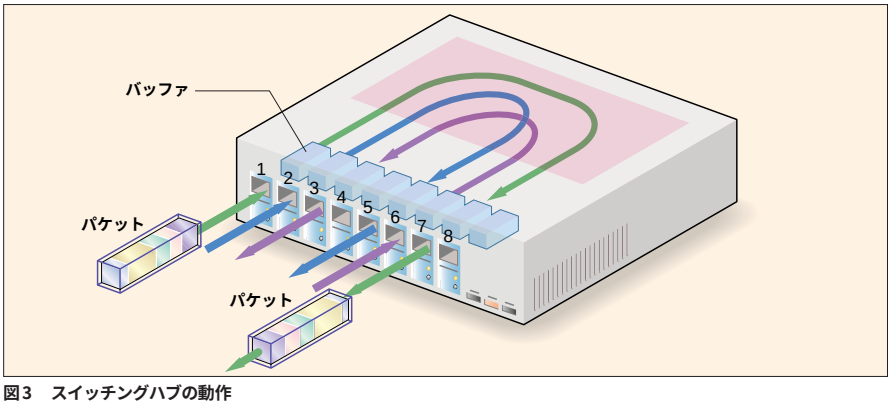


図3　スイッチングハブの動作
スイッチングハブはMACアドレスと接続ポートの対応を管理するアドレステーブルを持ち、独立した通信を同時に処理することができる。これにより、ネットワーク全体のスループットが向上する。

PCI 100BASE-TXネットワークカード

ここで紹介するネットワークカードは、編集部員が実際に秋葉原のパーツショップを回って、買い揃えたものである。また通信販売のWebサイトでも比較的よく見かける製品であり、入手は容易だ。

PCIバスの100BASE-TX製品に限定したのは、このカテゴリーにも一部の人気ブランド以外は低価格化が進んでおり、今や完全に普及価格帯の商品と言えるからだ。

もちろん極限まで出費を抑えたいのなら、10BASE-Tの製品、たとえばNE2000コンパチを選ぶことも可能だが、I／OアドレスやIRQを確認して指定する必要があり、面倒だ。その点PCIバス用のネットワークカードなら、リソースの割り当ては自動で行われるため、基本的にスロットに挿すだけで用いることが可能だ。

価格は、実際に購入した価格、および店頭や通信販売での価格を調査した結果をあわせた実勢値で示している。動作確認は、インストーラによって自動認識のしかたが異なると考えて、テキストベースのRed Hatインストーラを採用したLASER5 Linux 6.0 Rel.2（以下LASER5）、グラフィカルインストーラのアナコンダを採用したRed Hat Linux 6.2J（以下Red Hat）、そしてTurboLinux Workstation 6.0（以下Turbo）の各ディストリビューションで行った。

実際には、インストーラによる自動認識の差はほとんどなかった。ただしカーネルのバージョンによってドライバのバージョンが異なるため、新しくリリースされたディストリビューションほど、自動認識できる製品が増えてくるといった傾向がみられた。

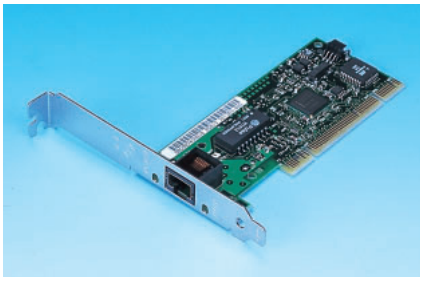
ペンギンマークは、パッケージの外側に「Linux Ready」「Linux OK」といった表示がしてあることを示している。これがついていないからといって、Linuxで使えないわけではない。実際、今回紹介するカードは最終的にはすべてLinuxで使うことができた。

ただしLinuxでの動作を保証していない製品については、メーカーに問い合わせをするのは、遠慮していただきたい。また今回の動作検証については、編集部で保証するものではない。こちらもご了承いただきたい。

インテル

PRO/100＋マネージメント・アダプタ

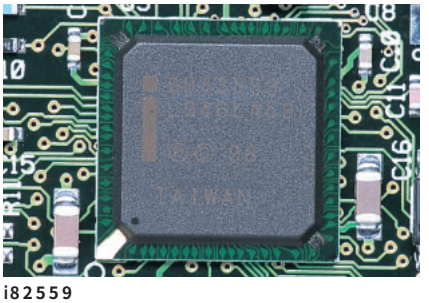
価格：7000〜9500円
ドライバ名：eepro100.o
http://www.intel.com/



インテルのネットワークカードというとEtherExpress PRO/100＋（以下EE）というイメージが強い。PRO/100+マネージメント・アダプタはその後継品である。EEはi82558というコントローラを採用していたが、このカードに採用されているのは、非常に小さなBGAパッケージのi82559である。コントローラが小さいためか、カード上はとても閑散としているように見える。またカードエッジの切り込みが2カ所にあることからわかるように、このカードはPCIバスの信号レベルは5Vと3.3Vの両方に対応している。

現状ではパッケージに記載されている対応OSリストにLinuxは載っていない。だが、このカードは今回確認した3つのディストリビューションのどれでもインストール時にEEと自動認識され、実際の動作にも問題はなかった。おそらくi82558のパッケージをBGAに変更して低コスト化を図ったのがi82559ということなのだろう。カードレベルでは高い互換性が保たれているようだ。

PCI 100BASE-TXという同じスペックの製品が、1500円から買える現状ではこのカードは決して安くはない。価格差を信頼できるブランドへの「安心料」と考える方向きと言えるだろう。



i82559
チップセットでおなじみのBGAパッケージだが、サイズは非常に小さい。

スリーコム ジャパン

Fast EtherLink XL PCI TX

価格：8000〜9000円
ドライバ名：3c59x.o
http://www.3com.com/

Fast EtherLink XL PCI TXは、米3Com社の日本法人であるスリーコムジャパンが販売しているネットワークカードだ。比較的伝統のあるメーカーだけに、ユーザーの信用もあり、Linuxで使うネットワークカードの定番のひとつにあげられよう。

コントローラは、同社の3Com 40-0483-004を採用している。一世代前の製品と比較して、3Com社独自のParallel Taskingアーキテクチャが改良されて、速度の向上やCPU使用率の低減が図られている。Linux用のドライバがParallel Taskingを利用しているかどうかは、不明だ。また、トランシーバ・インターフェイスなどを集積したことで、カード上の部品点数がかなり減っている。

パッケージにLinuxに関する記述はないが、上で書いたように以前からLinuxマシンで用いられており、ドライバの開発も進んでいる。今回テストしたディストリビューション3種は、いずれもカードを自動認識し、ドライバモジュールを適切にインストールした。インストール後はトラブルもなく、ネットワークが利用できた。

インテルのPRO/100＋と並ぶ高価なカードだが、定番ならではの安心感を求めるユーザーには適している。

3com 40-0483-004
3com独自のParallel Taskingを採用しているが、Linuxで利用できるかは不明。

コレガ

FastEther II PCI-TX

価格：1800〜2500円
ドライバ名：via-rhine.o
http://www.corega.co.jp/

FastEther II PCI-TXは、コレガが扱っている100BASE-TXカードのなかで最も低価格の製品である。今となっては、比較的大きめのカードで、ブートROM用のソケットも搭載しているが、現状ではサポートされていない。

コントローラチップにはDL10030と表記されている。製造元はカナダのD-Link社のようだが、同社の製品リストには掲載されていなかった。コレガのWebサイトの動作検証リストには、VIAのVT86C100Aと記載されていることから、セカンドソースによる同等品と思われる。

パッケージには、「Linux OK！」の文字とともに、Tux君が印刷されている。必要な情報はすべて同社のWebサイトから得るようになっており、マニュアルにLinuxに関する記述はない。だが、上述のように搭載されているコントローラはVT86C100Aの同等品であり、LASER5では手動でvia-rhineを指定することで、またRed Hat、Turboでは自動でvia-rhineと認識され、インストール終了後は問題なくネットワークが使用できた。

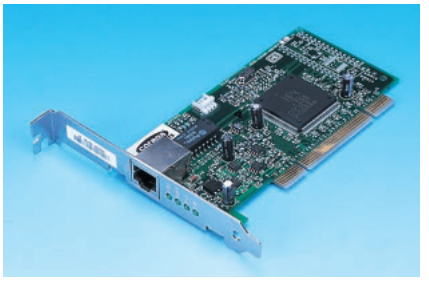
最近の小さなカードにはない、ブートROMソケットを持っているため、これが利用できるようになれば、さらに用途が広がるだろう。

D-Link DL10030
VIA VT86C100Aのセカンドソース品。ドライバも共通。

コレガ

FEtherW PCI-TX

価格：4800〜5500円
ドライバ名：pcnet32.o
http://www.corega.co.jp/

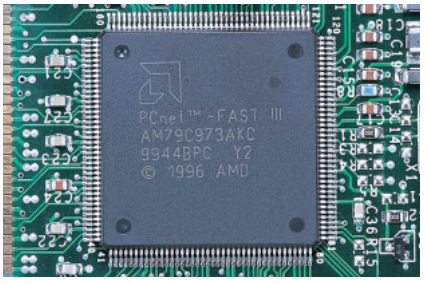


FEtherW PCI-TXは、AMD社製のコントローラを採用した、コレガのネットワークカードである。今回集めたカードのなかでは比較的高価な部類に属する。PCI 2.2に準拠しており、PCIバスの信号レベルは5V／3.3Vの両方に対応している。

最近のネットワークカードは、コストダウンのためか動作モードを表すLEDが少なかったり、まったくついていなかったりするが、このカードには「全二重」「100Mbps」「送受信」「リンク」の各動作状態を示すLEDが4個ついており、設定の確認やトラブルシューティング時には重宝する。

コントローラは、AMD社のAM79C973（PCnet FAST III）が用いられている。それほど人気のあるチップではないが、Linuxでサポートされている。Linux用のネットワークカードドライバといえば、Donald Becker氏が有名だが、このカードのドライバは別の開発者によって作られているので、彼のWebサイトを探してもソースが見つからないはずだ。最新版を探す時には注意しよう。

安定版カーネルの最新版2.2.14に付属するドライバのソースを用いれば、このカードを認識させることができる。

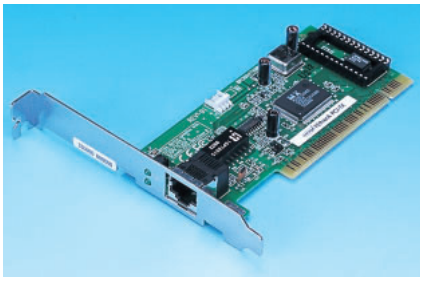


AMD AM79C973
PCnet FAST IIIとも呼ばれる。このチップを用いたカードは、あまり多くない。

コレガ

FEtherA PCI-TX

価格：2300〜3000円
ドライバ名：tulip.o
http://www.corega.co.jp/



FEtherA PCI-TXは、コレガが販売する3種類のPCI 100BASE-TXのなかでは中間の価格帯に属する製品だ。FastEther II PCI-TXと同様に、ブートROM用のソケットも搭載しているが、現状ではサポートされていない。

コントローラは、Macronix社のMX98715Aを採用している。このチップはDECの21140シリーズの互換製品で、21140より安価に提供されているらしく、このチップを採用したカードは、ほとんどがDECのチップを用いたカードよりも安い。

ドライバモジュールは、DEC用のtulipを使用できるが、ドライバのバージョンによっては、うまく動作しない可能性もあるので、注意が必要だ。その場合は後述するように、最新のドライバのソースを手に入れて、自分でコンパイルする必要がある。

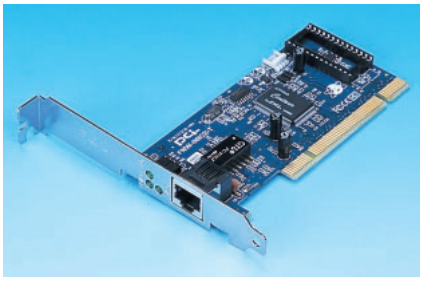
Red HatとTurboでは、最初から含まれているドライバモジュールで、自動的に設定が行われる。LASER5では、インストール時にtulip.oで動作するカードとして認識するが、カーネル2.2.5に付属しているtulipドライバモジュールが古いため、そのままでは利用できなかった。しかし新しいドライバのソースを取得して組み込んだところ、問題なくネットワークを利用できた。

Macronix MX98715A
DEC 21140互換製品。トランシーバ・インターフェイスを統合し、省電力化している。

プラネックスコミュニケーションズ

FNW-9802-T

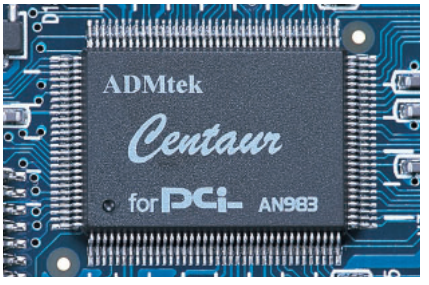
価格：2000〜2500円
ドライバ名：tulip.o
http://www.planex.co.jp/



プラネックスコミュニケーションズは、LinuxやFreeBSDなどオープンソースのシステムに積極的に対応していることで知られている。FNW-9802-Tは、同社の低価格帯に属する製品である。

コントローラには、ADMtek社のAN983（Centaur）が用いられている。このチップは、tulip.oというドライバモジュール名から分かるように、DEC 21140シリーズの互換製品である。トランシーバ・インターフェイスなどを統合し、低価格化を図った製品である。FNW-9802-TもDEC（最近はintelと書かれていることもある！）のチップを用いたカードの一般的な価格よりは、かなり安く販売されているようだ。

プラネックスコミュニケーションズでは、製品のLinux対応度を「Linux Ready」と「Linux OK」の2段階に分けて表示している。Linux Readyとは、同社またはLinuxディストリビューター各社による動作の確認が行われたことを示している。このカードはLinux Readyであり、添付されたフロッピーディスクには必要なドライバモジュールのソースが収められ、インストールの説明はマニュアルに掲載されている。このソースを用いてドライバを作成・インストールして、このカードが利用できることを確認した。

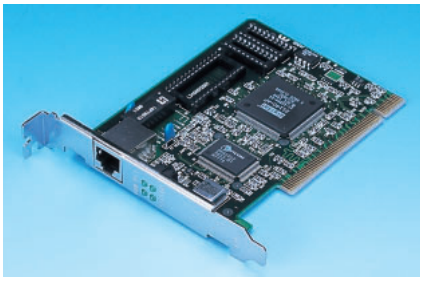


ADMtek AN983
Centaurというコードネームを持つ。「for PCi」の表記から、同社の専用品と思われる。

プラネックスコミュニケーションズ

ENW-9501-F

価格：4300〜5000円
ドライバ名：tulip.o
http://www.planex.co.jp/



ENW-9501-Fは、プラネックスコミュニケーションズのネットワークカードのなかでも比較的高価な製品である。約1年前に発売されたLinux magazine No.1でも紹介されており、製品サイクルが非常に短いコンピュータの部品としては、珍しいほどの長寿製品といえるだろう。

これほど長く売られている理由のひとつは、コントローラに定番中の定番であるDEC 21140AFが用いられているからだろう。tulipドライバモジュールをわざわざ最新のものに入れ替えることもなく、ほとんどのディストリビューションで自動認識され、適切にインストールされるからだ。Linux以外にもFreeBSDでも動くことが確認されている。

同社は自社製品のLinux対応の基準として、動作確認済みを表す「Linux Ready」と動作保証の「Linux OK」を設けている。このカードのパッケージには、Linux OKと記されており、TurboLinux、Red Hat Linux、Slackwareでの動作を保証している。

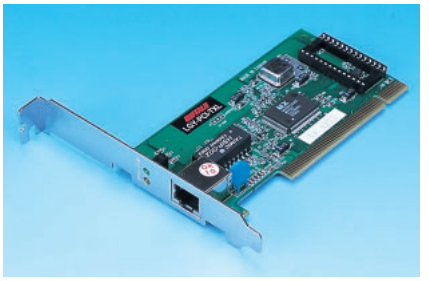
低価格の製品では、省略されていることの多いLEDも「100Mbps」「全二重」「送受信」「リンク」の4つが用意されており、状態の確認や問題の解決に役に立つ。

DEC 21140-AF
幅広いOSに対応しており、マルチブートPCにも適している。

メルコ

LGY-PCI-TXL

価格：1800〜2200円
ドライバ名：tulip.o
http://www.melcoinc.co.jp/



メルコはNEC PC-9801の全盛期から続く周辺機器メーカーであり、ネットワーク接続が当然のことになった現在では、多くのネットワーク関連機器を販売している。LGY-PCI-TXLは、同社のネットワークカードとしてはローコストモデルのグループに属しており、サポートするOSはWindows NT／2000／9xに限定されているが、適切なドライバを用意すればLinuxで使うことができる。またPC-9801時代から続くメーカーの製品らしく、このカードはPC-9821シリーズにも対応している。

コントローラは、Macronix社のMX98715Aを用いている。このチップは、DEC 21140シリーズ互換であり、21140シリーズと同じようにtulipドライバモジュールが利用できる。

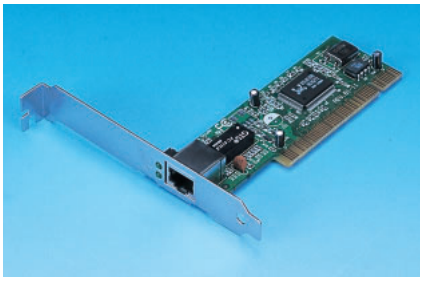
ただしMX98715Aは、互換製品としては比較的新しいため、ドライバのバージョンによってはうまく認識されないことがある。その場合には、最新版のドライバを入手することで、解決できることがある。LGY-PCI-TXLは、Turbo、Red Hatでは自動認識されてそのまま利用できたが、LASER5では新しいドライバを用意する必要があった。

Macronix MX98715A
DEC 21140の互換製品であり、tulipが利用できる。

エレコム

LD-10/100S

価格：1600〜2000円
ドライバ名：rtl8139.o
http://www.elecom-laneed.com/



LD-10/100Sは、エレコム社がLANEEDブランドで発売しているネットワークカードだ。同社の製品のなかでは、最も低価格な部類に入る。

最近の安価な100BASE-TXネットワークカードは、VIA Technologies社のVT86C100Aか、Realtek社のRTL8139Bが用いられることが多い。LD-10/100Sは、RTL8139Bが採用されている。チップに描かれている絵から「カニ」と呼ばれることが多い。

RTL8139Bは、トランシーバ・インターフェイスなどを統合し、少ない部品数でネットワークカードを構成できるようになっている。同チップを用いたカードに安価な製品が多いのはそのためだ。安い代わりに性能もそれなりのチップでは決してない。

このカードは、Turbo、Red Hatのインストーラからは自動認識される。またLASER5ではリストからrtl8139を選択することで、正しく動作させることができた。

なお、エレコム社のWebサイトを見ると、LD-10/100Sはすでに生産が終了している。後継の製品はLD-10/100ALで、VT86C100A互換のコントローラが採用されており、Linuxに対応した安価なネットワークカードという特徴は、引き継がれているようだ。

Realtek RTL8139B
なぜかはわからないが、「カニ」が描かれている。安価なチップだが、性能は悪くない。

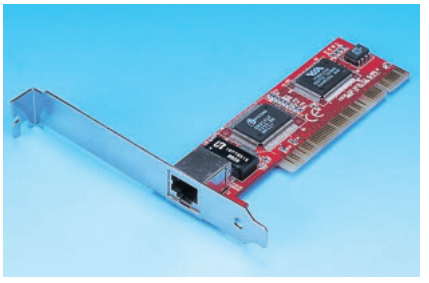
ナカガワメタル

TR-PCI-100

価格：2800〜3500円

ドライバ名：via-rhine.o

http://www.thanrive.co.jp/



TR-PCI-100は、非常に小さな基盤と、ブラケット部分が分離した状態でプラスチックケースに入れられて販売されており、ユーザーは使う前に組み立てる必要がある。コスト削減のアイデアなのだろうが、他社製品の低価格化が進んだため、今では特に安い製品ではなくなっている。

コントローラは、VIA VT86C100Aを用いている。このチップは、比較的安価なネットワークカードに用いられることが多い。だが性能的には、より高価なカードと比べても遜色なく、決して「安かろう悪かろう」な製品ではないことがわかる。

編集部で購入したパッケージには、「Linux動作確認済」と書かれたシールが貼ってあり、添付のフロッピーディスクにドライバのソースが入っていた。もっともテストに用いた3つのディストリビューションでは、いずれもインストール時に自動で認識され、ドライバモジュールも適切に組み込まれたため、添付されたソースは必要なかった。

残念ながらこのカードには、LEDが1つもないため、トラブル時や動作の確認を行いたい際には、苦労するだろう。1000円台の製品でも、「リンク」「全二重」のLEDくらいは付いてくるものだ。ぜひ改善を望みたい。

VIA VT86C100A
RTL8139Bと並ぶ低価格チップ。

ノーブランド

UE1211D-TX

価格：1470円

ドライバ名：rtl8139.o

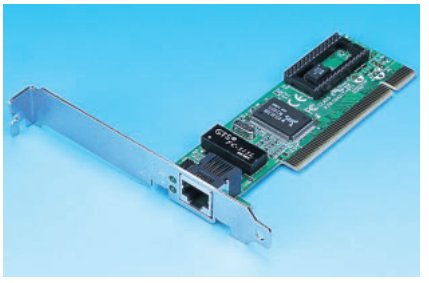
ー

UE1211D-TXは、今回紹介するカードのなかで最も安い製品だ。1500円以下でPCI 100BASE-TXのカードが買えるとは、正直言って驚きだ。白い箱で売られており、箱には一応「Pragmatic」とブランド名らしき表記はあるのだが、編集部内でも誰も知らなかったうえに、付属マニュアルの日本語が少々怪しいものだったので、ノーブランドという扱いにした。

コントローラは「カニ」こと、Realtek社のRTL8139Bを用いている。約1年前のLinux magazine No.1でネットワークカードを紹介した際には、RTL8139Bを搭載したカードは取り上げられていなかった。「カニ」はこの1年で低価格を武器に、急激に普及したようである。

最近のディストリビューションには、ドライバモジュールが含まれている。LASER5の場合は、インストール時にリストからrtl8139を指定することで、TurboやRed Hatはカードを自動認識してネットワークが使用可能になる。

パッケージやマニュアルにLinuxに関する記述はない。添付のフロッピーディスクに、RTL8139B用のソースファイルが収録されており、古いディストリビューションを使用している場合にも対応できる。



Realtek RTL8139B
周辺チップを統合し、低価格化を実現している。

低価格8ポートスイッチングハブ

少し前までは、非常に高価だったスイッチングハブも急激に値下がりが進み、3〜4年前のリピータハブの感覚で買える値段になってきた。今回のテストでは「8ポート10／100 Mbpsスイッチングハブで1万円以下」という条件で集めたところ、コレガ、プラネックスコミュニケーションズ、メルコの3社の製品が該当した。

スイッチングハブの場合、ポートとネットワークカード間を全二重接続できるのが、リピータハブとの違いである。今回紹介する3機種は、ネットワークカードが対応していれば、いずれも自動的に全二重で接続できるようになっていた。詳細は後述するが、どの機種でも全二重で接続できていることを確認した。

接続するコンピュータの条件を変えずに、ハブを交換して性能比較を行ったが、有為な差は得られなかった。

コレガ

Fast SWIII-8P

価格：7190円

http://www.corega.co.jp/

Fast SWIII-8Pは、コレガの低価格スイッチングハブだ。1Mバイトのバッファを持ち、1000個のMACアドレスを記憶できる。スループットは148,810パケット／秒（100Mbps接続時）。IEEE 802.3xに準拠したフローコントロール機能を備える。

全ポートに100／10、全二重／半二重を自動認識するオートネゴシエーション機能を持ち、既存の10Mbps環境を活かしたまま段階的に100Mbps LANに移行できる。

ファンを持たないので動作音は小さい反面、放熱が心配になるが、金属製の筐体を用いることで対処している。

プラネックスコミュニケーションズ

FX-08SMC

価格：7880円

http://www.planex.co.jp/

FX-08SMCは、ほかのプラネックス社製品と同じ青色の筐体を持ったスイッチングハブだ。プラスチック筐体でファンなしなので、放熱面で多少不安が残る。底面にマグネットがあるので、デスクの横などに設置できる。

ハブとしての性能は、1Mバイトのバッファ、1000個までのMACアドレスを記憶、148,810パケット／秒（100Mbps接続時）のスループットを持ち、IEEE 802.3x準拠のフローコントロール機能を備える。

なお4月にFX-08SMCの後継機であるFX-08SMC2が発売されている。バッファが256Kバイト、MACアドレスの記憶数が4000までと変わった以外はほぼ同じスペックを持つ。

メルコ

LSW 10/100-8R

価格：7770円

http://www.melcoinc.co.jp/

LSW 10/100-8Rは、ほかの2機種と違いRJ45コネクタが前面にある。設置条件や好みに合うものを選べばいいだろう。

1Mバイトのバッファ、1024個までのMACアドレスを記憶、スループット148,810/秒、IEEE 802.3x準拠のフローコントロール機能を持つ。

オートネゴシエーション機能により、100／10、全二重／半二重の条件を自動認識できるため、既存の10Mbps環境が混在したまま段階的に100Mbps LANに移行できる。さらにカスケード用ポートを備えているので、LANの拡張に合わせてハブを買い足していくことが可能だ。

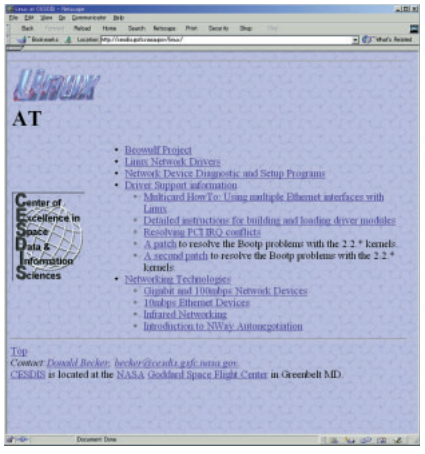
NICがLinuxで動かない？　そんなときは

Linuxのネットワークデバイスサポートはかなり充実しており、現在販売されているNICの多くがLinuxで動作する。しかし、今回紹介したNICも、全機種があっさりと動いたわけではない。イーサネットカードに搭載されたコントローラチップの種類によっては、新しいドライバでないとうまく動かないものもあるのだ。特に互換チップには新しく開発されたものも多いので注意が必要だ。

ここでは、ディストリビューションに含まれるドライバでNICが動作しなかったときの対処法を解説しよう。

まずは情報収集

最初に、自分の使おうとしているNICがどのようなコントローラチップを使っているかを知る必要がある。製品紹介ページの写真を参考に、基板上にある似たような形状のチップを探そう。このような形のチップは1～2個しかないはずなので、苦労はないだろう。見つけたら表面に書かれた型番をメモする。



画面1　NASA CESDISのLinux Webサイト
URL　http://cesdis.gsfc.nasa.gov/linux/

次に、そのコントローラに対応するドライバがあるのかどうかを調べなければならない。最近のNICならば、この特集で紹介した製品と同じコントローラを使っているものも多いだろう。また、Linuxのイーサネットドライバを数多く開発し、公開しているNASA CESDISのWebサイト（**画面1**）が参考になる。ここには、ディストリビューションやカーネルソースに含まれていない新しいバージョンのドライバも多数公開されている。

インターネットが利用できれば、検索エンジンも活用して情報を集めよう。

ドライバの更新

ドライバは、Cのソースコードとして配布されているので、コンパイルして使うことになる。カーネルを再構築する際に直接カーネルに組み込んでしまってもよいのだが、今回はモジュールとして組み込むことにする。

作業内容は次のようになる。入手したソースをコンパイルし、モジュールとして使えるオブジェクトファイル（～.o）を作成する。そして、それをモジュールを置くディレクトリ（/lib/modules/カーネルバージョン/以下）にコピーし、depmod -aを実行してモジュールデータベースを再構築する。modprobeコマンドで組み込み、動作することが確認できたら、起動時に自動組み込みされるよう/etc/conf.modules（ディストリビューションによっては/etc/modules.conf）ファイルに登録する。どのドライバもだいたい同じ手順で作成できるので、未知の製品にも応用できるはずだ。

では、ディストリビューションによってはドライバの更新が必要だった製品について、モジュールの更新方法を個別に説明する。

●FEtherA PCI-TX、LGY-PCI-TXL

コレガのFEtherA PCI-TXとメルコのLGY-PCI-TXLは、MacronixのMC98715というイーサネットコントローラを採用している。これは、DECの21＊4＊ Tulipチップ互換のコントローラなのだが、古めのTulipドライバでは動作しない。このため、先ほど紹介したCESDISのWebサイトより、tulip.c（バージョン0.91g）を入手し、**画面2**のようにして新たなモジュールを作成した。

コンパイル時に、gccに与えるオプションはtulip.cの最後に書かれているので、

```
# tail tulip.c
```

として見ることができる。

動作を確認したら、/etc/conf.modules（/etc/modules.conf）ファイルに、

```
alias eth0 tulip
```

と記述する。

●FNW-9802-T

プラネックスのFNW-9802-Tには、ADMtekのAN983というコントローラが使われている。このチップもTulip互換だが、先ほどのCESDISから入手できるバージョン0.91gのTulipドライバでは未対応だ。でもあわてることはない、製品に付属のドライバディスク、あるいはプラネックスのWebサイト（http://www.planex.co.jp/）から入手できるアーカイブファイルにLinux用のドライバ（tulip.c バージョン0.91u）が含まれている。あとの作業手順は先ほどと同じだ。

●FEtherW PCI-TX

コレガのFEtherW PCI-TXは、コントローラとしてAMDのAM79C973が使われている。このチップに対応するドライバは、pcnet32.oだが、バージョンが1.22以降でないと使えない。CESDISではこのドライバを配布していないが、現時点で最新の安定版カーネル2.2.14にバージョン1.25kfが含まれているのでこれを使おう（カーネルごと再構築してもかまわない）。付録CD-ROM Disk2に、カーネル2.2.14のソースを収録しているのでこれを使う。ソースを展開したら、linux/drivers/net/pcnet32.cを適当な作業ディレクトリにコピーし、コンパイルする。pcnet32.cの末尾にコンパイル方法が書かれているが、そのとおりにしても動かなかったので、次のようにした。

```
# gcc -D__KERNEL__ -DMODULE -Wall
-Wstrict-prototypes -O6 -m486 -c
pcnet32.c
```

こうしてできたモジュールファイルpcnet32.oを、installコマンドを使い、**画面2**と同じ要領で所定のディレクトリにコピーする。depmod -aでモジュールデータベースを再構築したら、

```
# modprobe pcnet32
```

として、組み込めることを確認しよう。最後に、/etc/conf.modules（または/etc/modules.conf）に、

```
alias eth0 pcnet32
```

と記述しておけばブートとともに組み込まれるようになる。

開発版の最新ドライバを使う

最新カーネルや、CESDISからドライバのソースを入手すれば、ほとんどのNICが利用できるはずだが、それでも対応できないときは、開発版カーネル向けの最新ドライバを試してみよう。これは、Scyld Computing CorporationのWebサイト（http://www.scyld.com/）で公開されている（**画面3**）。開発版のため、更新頻度も高く、安定度も未知数なので注意して使おう。

ここからnetdrivers.tgzというファイルをダウンロードし、展開後、makeコマンドを実行すればモジュールファイルが作られる。先ほどと同様に、installコマンドで必要なモジュールとpci-scan.oをコピーする。pci-scan.oは、PCIバスに接続されたNICの検出に使われるもので、従来は各モジュールごとに含まれていたルーチンを分離独立させたものだ。

あとは、depmod -aを実行して、modprobeコマンドでモジュールを組み込むなど、従来と同じ手順でよい。depmod -aがエラーになるときは、一度再起動するとうまくいくかもしれない。

コンパイルしてモジュールを作成する

```
# gcc -DMODULE -D__KERNEL__ -Wall -Wstrict-prototypes -O6 -c tulip.c `[ -f /usr/
include/linux/modversions.h ] && echo -DMODVERSIONS`
```

オリジナルのモジュールをリネームする

```
# mv /lib/modules/`uname -r`/net/tulip.o /lib/modules/`uname -r`/net/tulip.o.orig
```

新しいモジュールをコピーする

```
# install -m 644 tulip.o /lib/modules/`uname -r`/net/
```

モジュールデータベースを再構築する

```
# depmod -a
```

モジュールを組み込む

```
# modprobe tulip
```

画面2　Tulipドライバの作成と更新

画面3　Scyld Computing CorporationのWebサイト
以前はCESDISにあった開発版最新モジュールはここで公開されている。

性能比較

同じ「PCI 100BASE-TXネットワークカード」というカテゴリーの製品でも、ノーブランドの1500円から有名どころの8000円まで値段に開きがある。この5倍以上の差は、性能の差なのだろうか。このことを確かめるために、全11機種のカードを同じ条件で動かして、比較してみた。

実験方法

実験には編集部内で組み立て、利用しているPCを用いた（**表1**）。クライアント側のネットワークカードを交換した以外は、なるべく条件を揃えてファイルの転送速度を測定した。

99ページで紹介したスイッチングハブのなかから、今回はプラネックスのFX-08SMCを用いて実験を行った。もちろん、事前にいくつかテストをして、メルコのLSW 10/100-8R、コレガのFast SWIII-8Pを用いても有意な差が出ないことは確認済みだ。FX-08SMCを使用したのは、マグネットで固定できるのが便利だったからという理由からだ。

使用した2台のPCは、編集部内のLANから切り離し、スイッチングハブに接続して、独立したLANを構成した（**図1**）。

実際の測定は、**リスト1**のようなテキストファイルを作製し、ftpコマンドでファイルの転送（両方向）を行った。その所要時間から転送速度を、CPU使用時間からCPU使用率をそれぞれ算出した。低いCPU使用率で大きな転送速度が得られるカードが優れたカードである。

100BASE-TXは理論的には10Mバイト／秒以上の帯域を持っているため、小さいファイルではほとんど差がつかない。そこで約660MバイトのISOイメージファイルを用いた。このサイズではキャッシュに入りきらないため、ハードディスクから10Mバイト／秒以上の速度でデータを読み出せるようにしなければならない。そこでhdparmコマンドを用いて、DMA転送を有効にした。

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

今回のサーバ／クライアントで用いて

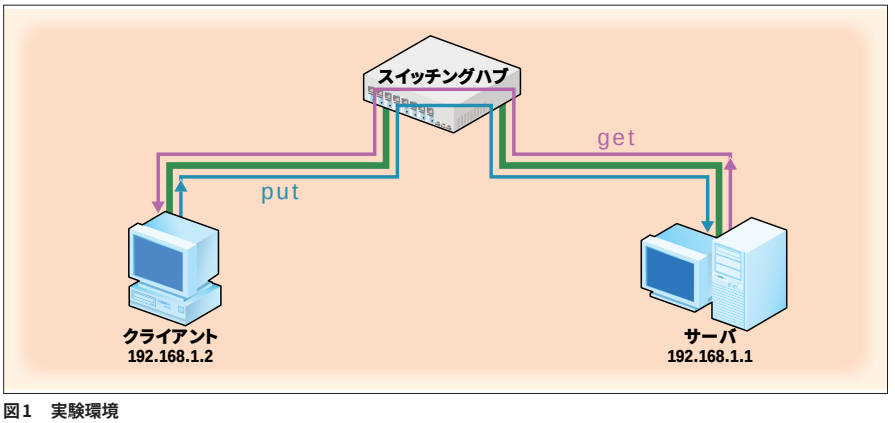


図1　実験環境

テスト用自作PC（サーバ）	
マザーボード	Microstar MS-6167
CPU	AMD Athlon 500MHz
メモリ	PC100 SDRAM 128Mバイト
ハードディスク	IBM DPTA351500（15Gバイト）
ネットワークカード	intel EtherExpress PRO/100＋
OS	TurboLinux Workstation 6.0
テスト用自作PC（クライアント）	
マザーボード	EPOX KP6-BS（dual CPU対応）
CPU	Celeron 300AMHz
メモリ	PC100 SDRAM 128Mバイト
ハードディスク	Quantum FBlct17.3AT（17.3Gバイト）
ネットワークカード	実験ごとに交換
OS	LASER5 Linux 6.0 Rel.2 Red Hat Linux 6.2J TurboLinux Workstation 6.0

表1 テストに使用したPCのスペック

リスト1 実験用スクリプト

```
$ cat get-script
open 192.168.1.1
user foo hogehoge
get test.iso /dev/null
quit
$ time ftp -n <get-script
```

いるハードディスクは、IBMのDPTA351500とQuantumのFBlct17.3ATだ。どちらも上のようにしてDMA転送を有効にすると、18～19Mバイト／秒の速度が得られるため、ディスクからの転送が間に合わないということはない。

結果

上記の設定でファイル転送を行った結果を**グラフ1、2**に示す。getの場合は、どのカードも理論的限界ぎりぎりの11.2～11.4Mバイト／秒が出ている。putの場合も、少しばらつきはあるが、10Mバイト／秒前後に集中している。少なくとも転送速度という点では、各カードにまったく優劣はつけられないようだ。

一方ファイル転送時のCPU使用率については、多少ばらつきがあるようだ。getの場合にはAMD AM79C973を用いたコレガのFEtherW PCI-TXが群を抜いて小さい値を示している。一方Parallel Taskingなど独自技術で優れた性能を主張する3com 3C905B-J-TXはまったくふるわない結果に終わっている。またputでは30%前後のものと40%前後の2グループに分類できるようだ。

以上のようにCPU使用率で無理に順位をつけることもできるが、その差はとても小さく、実際にPCを使っているときには体感できるとは思えない程度だ。よってクライアントマシン用のネットワークカードとしては、今回紹介したどれを使っても満足できる結果が得られるだろう。一方、FTP／Webサーバなど、データを送り出す作業が集中するマシンでは、put時のCPU使用率が小さいものを選ぶべきだろう。

全二重の効果

スイッチングハブのポートとネットワークカードの間は、全二重で接続することができる。大量のデータが両方向に行き交う状況では、効率アップが期待できる。そこで**図1**の環境で、全二重で接続することの効果を確認してみた。方法は単純で、クライアント／サーバの両方からお互いにftpでgetするというものだ。クライアント側のNICには、インテルのPRO/100＋を用いた。

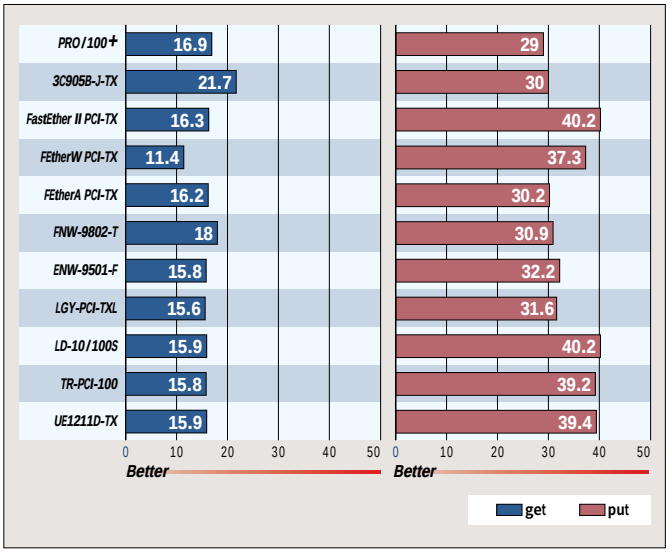
特に何も手を加えてない状況で、何回かトライしてみたが、両方向にデータが流れ始めるとネットワークカードが反応しなくなるという症状に見舞われてしまった。これはディストリビューションに付いているドライバモジュール（eepro100.o）が古いことが原因だった。最新のソースを取得して両方のマシンにインストールしてみたところ、全二重で接続した場合は、100Mbpsの2倍近い帯域が得られていることが確認できた。

また、モジュール組み込み時のオプション指定で、半二重に固定して同じようにデータをやりとりすると、転送速度は半分以下の4Mバイト／秒に低下した（**表2**）。

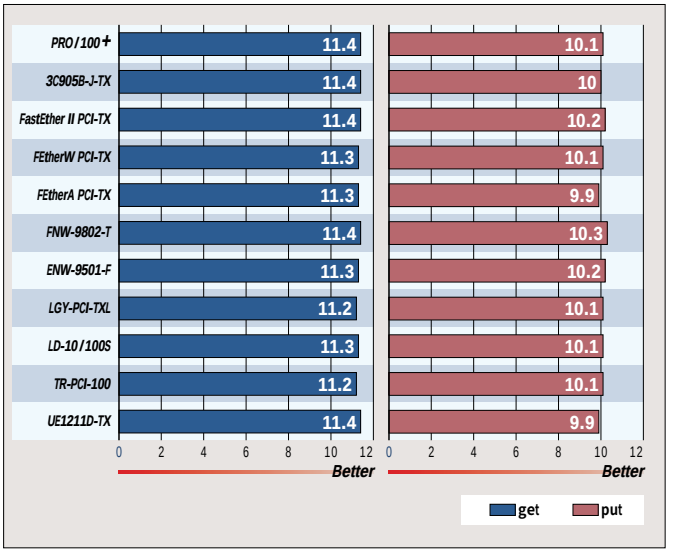
	サーバ→クライアント	クライアント→サーバ
サーバ→クライアントのみ	11.4Mバイト/s	
全二重	11.1Mバイト/s	9.7Mバイト/s
半二重	4Mバイト/s	4Mバイト/s

表2 全二重の効果

グラフ1　CPU使用率（%）



グラフ2　転送速度（Mバイト／秒）



遊び倒せ！ Linux！ ゲーム

中毒になる!

昔からゲームとUNIXは、切っても切り離せない関係であった。
もちろんUNIXの子孫であるLinuxも同様だ。
ホームユーザーならぜひともLinuxゲームを楽しんでもらいたい。
そこで最新の3Dモノから懐ゲーのリメークまで、
厳選した19本のフリーなゲームソフトを紹介する。
ハマりすぎた結果として、学業や仕事に支障が出てしまっても、
編集部では責任がとれないので、あしからず。

さあ、ハマれ!

文:出井 一、編集部
Text : Hajime Dei, Linux magazine

ゲームしてもいいですか？

Linuxは、最新のUNIXだ。UNIXと聞くと、よく分からないけどなんとなく高級な感じがして、ゲームとは無関係だと思う読者もいるだろう。確かにLinux（UNIX）は、サーバやプログラム開発というお堅い用途で使われている事例は多いが、決してそれ「だけ」のOSでないのだ。

■

■ はじめにゲームありき

■

そもそも最初のUNIXは、ゲームのために作られた！　1968年、ベル研究所に勤務していたKen Thompsonは、当時彼がハマっていたSpace Travelというゲーム（宇宙船をいろいろな星に着陸させるゲームだったらしい）を思う存分楽しむために、誰も使っていなかったマシン用にファイルシステムやシェルの原形を作り上げた。これが、UNIXのはじまりだ。

このような出自を持つUNIX、そしてその子孫のLinuxでゲームをやるのは、当然のことなのだ。

今ならどんな安いコンピュータでも、当たり前のように1024×768ドットくらいのグラフィック画面を表示できる。だが一昔前は、何百万円もするUNIXマシンでもテキストしか表示できなかったりした。そんな表現力の乏しい環境で作られたゲームが面白いのか？面白かったのだ。

たとえばRogue。これはテキストだけで画面が構成されたロールプレイングゲームで、自分のキャラクターは「@」、敵モンスターもすべてアルファベット1文字で表されるというものだった。このようにシンプルな画面でもプレイを続けていくうちに、薄暗い洞窟をおそるおそる歩いている自分のイメージが浮かんできたものだ（睡眠時間を削り過ぎたせいかもしれないが）。

その後Rogueライクなゲームが何種類も登場したが、なかでもNetHack（**画面1**）は、多彩なイベントや仕掛けで、多くのプレイヤーから貴重な睡眠時間を奪ったものだ。今回紹介するゲームのなかには、このNetHackのグラフィック版もある。秀逸なアイディアはそのままに、見た目も進化しており、中毒の危険性もいっそう高くなっているといえよう。

このようにLinux（UNIX）上で動くゲームには、表現力に頼らず優れた着想とアイデアで人気を得たゲームが多くある。最近のように高解像度のグラフィック表示や3Dグラフィックスが利用できるようになっても、それは変わらない（**画面2**）。

■

■ ゲームの効能

■

ゲームが含まれているOSは、Linuxだけではない。Windowsだってそうだ。コンシューマー向けのWindows 9xだけでなく、業務用のWindows NT／2000にさえソリテア、フリーセルが入っている。もちろんこれには理由がある。コンピュータにさわったことのない初心者がマウスやキーボードの操作に慣れ親しんだりするために、お手軽なゲームは最適なのだ。

前述のRogueやNetHackにも同じような効能があった。これらのゲームでは、自分のキャラクターを動かすキーがH／J／K／Lに割り当てられていた。言うまでもなく、これはviエディタのキーアサインと同じであり、Rogueをやり込んだプレイヤーにとって、viのカーソル移動は最初から分かっているも同然だ。

Linuxでフリーのゲームをプレイする場合には、このようなマウスやキーボードへの慣れ以外にも、パッケージのインストール練習にもなる。最初はおっかなびっくりでも、何回かやっているうちにコツがつかめてくるはずだ。そうなれば、もう初心者卒業といっていいだろう。

千里の道も一歩から。Linuxに慣れ親しむなら、まず使ってみよう。そのためには、ゲームが最適だ。睡眠時間と引き換えに、Linux経験値を上げよう！

画面1　JNetHack（写真はWindows版）。このシンプルな画面がハマりの元だ。

画面2　glTron。フリーのゲームソフトの表現力もここまできた。

一般的なインストール方法

Linux上で動作するソフトの配布形態は、tar＋gzipを利用した昔ながらのソース配布（「tarボール」などと呼ばれる）と、Red Hat系ディストリビューションでおなじみのRPMを使ったパッケージ配布に大別できる。以下では、RPMパッケージのインストール方法とソースからビルドしてインストールする方法について簡単に説明する。

●RPMを利用したインストール

ここでは基本となるrpmコマンドの操作方法を説明する。なお、通常は一般ユーザーのままでもインストール作業を行えるが、一部のパッケージについてはスーパーユーザー（root）になって作業を行う必要がある（suコマンドを使用する）。

バイナリパッケージは、「hoge-1.2.3-4.i386.rpm」のように、ファイル名の末尾に「.i386.rpm」が付く。これをインストールするには、

```
$ rpm -Uvh hoge-1.2.3-4.i386.rpm
```

とする。アンインストールの際は、

```
$ rpm -e hoge
```

とする。アンインストール時には、バージョン番号などは指定しないことに注意しよう。

glibcのバージョン違いなどの理由でバイナリパッケージをインストールできない場合には、ソースパッケージからバイナリパッケージを作成する「リビルド」を行う。

ソースパッケージのファイル名は「hoge-1.2.3-4.src.rpm」のように、ファイル名の末尾に「.src.rpm」が付く。これを、

```
$ rpm --rebuild hoge-1.2.3-4.src.rpm
```

とすると、/usr/src/redhat/RPMS/i386に、対応するバイナリパッケージが作成される（**画面3**）。このバイナリパッケージを、rpmの－Uvhオプションを使ってインストールすればいい。

●ソースから作成する

ソースのtarボールから作成する場合、一般ユーザーのホームディレクトリ（/home/ユーザー名）にsrcというサブディレクトリを作成し、そこでtarボールを展開するとよいだろう。

tarボールは、「hoge-1.2.3.tar.gz」のように、ファイル名の末尾に「.tar.gz」が付く。これを、

```
$ tar xzvf hoge-1.2.3.tar.gz
```

とすると、tarボールに含まれる全ファイルが展開される。

インストール方法は、INSTALLやREADMEといった名前のドキュメントファイルに記載されているので、確認しておこう。

展開されたファイルの中に、configureという名前のものが含まれていれば、

```
$ ./configure
```

とするだけでビルドに必要な情報を取得し、自分の環境に適したMakefileを自動的に作成してくれる。その後、

```
$ make
```

とすると、コンパイルやリンクが行われ、実行ファイルが作られる。続いて、suコマンドを使ってスーパーユーザー（root）になった状態で、

```
# make install
```

とするとインストールが行われ、プログラムやデータファイルなどが適切なディレクトリにコピーされる。また最近では、

```
# make uninstall
```

とするとアンインストールされるようになっているソフトも多いようだ。

このほか、ソースファイルを修正する「パッチ」が別途配布されているソフトもある。パッチを当てる（修正する）には、patchコマンドを使用する。たいていの場合、ソースを展開したディレクトリで、

```
$ patch -p1 < パッチファイル名
```

とすればいい。

画面3　ソースパッケージをリビルドしてバイナリパッケージを作成する。

宇宙船で自由に宇宙を飛び回るゲームシステム

XShipWars

バージョン：1.33f　　ライセンス：GPL

http://fox.mit.edu/xsw/

XShipWarsは、複数のプレイヤーが宇宙船をリアルタイムに操作するサーバ・クライアント方式のゲームシステムだ。宇宙船や宇宙ステーション、惑星、ワームホールなどが美しくデザインされ、効果音も凝ったものが使われている。標準でスタートレックTNG/DS9の舞台となるユニバース（宇宙）が用意されているほか、付属のエディタを使って自分でユニバースをデザインすることも可能だ。

●ビルドとインストール

XShipWarsは、本体のtarボールのほか、基本データ、画像、サウンドのtarボールが配布されている。なお、サウンドやジョイスティックを利用する場合、サウンドドライバ（YIFFかEsounD）やジョイスティックドライバをインストールする必要がある。

本体のtarボールには、クライアントとサーバ、サーバ用のモニタ、ユニバース作成用のエディタが含まれている。インターネット上のサーバに接続して遊ぶ場合はクライアントのみビルドすればいい。

「./configure」として設定を行った後、client/Makefile.Linuxをエディタで修正する。変更の必要があるのは、73行目のCFLAGSと、88行目のLIBの2個所だ。コメントを参考にして必要ない部分を削ろう。たとえば、ジョイスティックを使わないなら、CFLAGSの「-DJS_SUPPORT」とLIBの「-Ljsw」を削除する。

続いて、「make client_linux」とするとクライアントがビルドされる。インストールは、ディレクトリclientに移動してから「make install」とする。実行ファイルは/usr/gamesに、データは/usr/share/games/xshipwarsにそれぞれコピーされる。

最後に、別途配布されている基本データ、画像、サウンドのインストールを行う。/usr/share/games/xshipwarsに移動してから、xswdata1.33d.tgz、stimages1.6.tgz、stsounds1.4.tgzをすべて展開すればいい。

●クライアントの起動

「xsw&」としてクライアントを起動すると、ウィンドウが開いて美しいタイトルが表示される（**画面1**）。

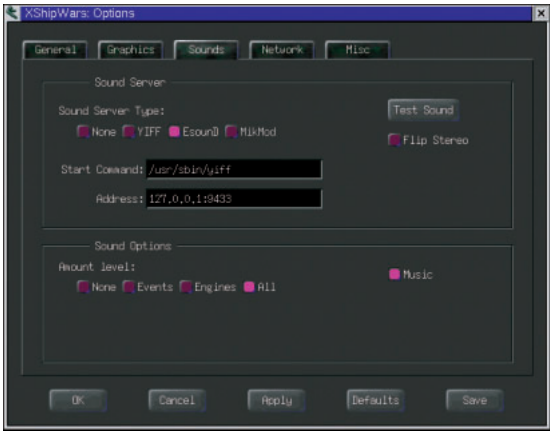
まずは、[OPTIONS]ボタンを押して初期設定を行う。設定ダイアログはいくつかのページに分かれおり、操作に使用するキーや、サウンドドライバの選択、デフォルトで接続するサーバなどの設定が可能だ。

たとえば、EsounDを利用する場合、[Sounds]ページ（**画面2**）の[Sound Server Type]で[EsounD]、[Amount level]で[All]を選択する。[Apply]ボタンで設定を有効にしたら、[Test Sound]ボタンを押して、実際にサウンドが再生されるかどうか確かめよう。正しく設定されたら、[Save]ボタンで設定ファイルに保存する。

設定が終了したら、次に[CONNECT]ボタンを押してユニバースリストウィンドウを開く（**画面3**）。接続可能なユニバースが一覧表示されるので、接続先をダブルクリックで選択してサーバに接続しよう。[TWU]か[Dogstar]がお勧めだ。また、XShipWarsのWebサイトには、接続可能なサーバの情報が随時更新されている。これらを反映するようにプロパティを変更したり、新しいユニバースをリストに追加することも可能だ。

なお、XShipWarsはゲームシステムなので特定の目的は設けられておらず、各ユニバースの設計者が目的を設定する。たとえば、[TWU]では基本的な戦

画面1
美しいタイトル画面が気分を盛り上げる。



画面2
サウンドを楽しむには、サウンドドライバの初期設定が必要だ。

闘方法を学習すること、[Dogstar]ではプレイヤー間のコミュニケーション（チャット）を楽しむことが目的だ。ユニバースの目的に沿った行動をとるようにしよう。

●ユニバース接続後の操作

サーバに接続すると、即座にプレイが始まる。プレイ中のウィンドウは、左側に自船のコンソール、中央に周囲の宇宙を表示するビュースクリーン、右側にスキャナのコンソール、下にメッセージボックスという構成になっている（**画面4**）。

操作にはキーボード（あるいはジョイスティック）を使用する。たとえば、宇宙船は↑／↓キーで加減速し、←／→キーで方向転換する（慣性の影響を受けるため、即座に停止することはできない）。

また、Tabキーで周囲の物体が順番にスキャンされ、右のコンソールに表示される。武器は数字キーで切り替え、スペースキーで発射だ。F1キーでこうしたキーの一部が表示されるほか、設定ダイアログの[General]ページで[Map Keyboard]ボタンを押すと、全キーマップを参照できる。

また、ビュースクリーン上で右クリックすると、スターチャート（**画面5**）や設定ダイアログの表示、サーバの接続・切断処理などを行うメニューがポップアップする。

●サーバを起動する

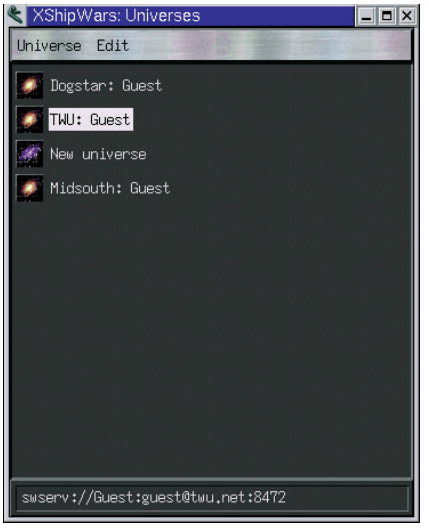
ローカルネットワークでサーバを動かせば、電話料金を気にせずにこころゆくまで宇宙船の操作を練習したり、仲間内だけでプレイを楽しんだり、ユニバースの設定を自由に変更したりできる。

サーバは「make server_linux」、モニタは「make monitor_linux」でそれぞれビルドされる。各サブディレクトリ（serverとmonitor）に移動して「make install」とすると、/home/swserv以下に実行に必要なファイルがインストールされる。

次に、/home/swserv/restartを編集する。とりあえずは、19行目を「set CONF = etc/default.conf」に修正するだけでいい。

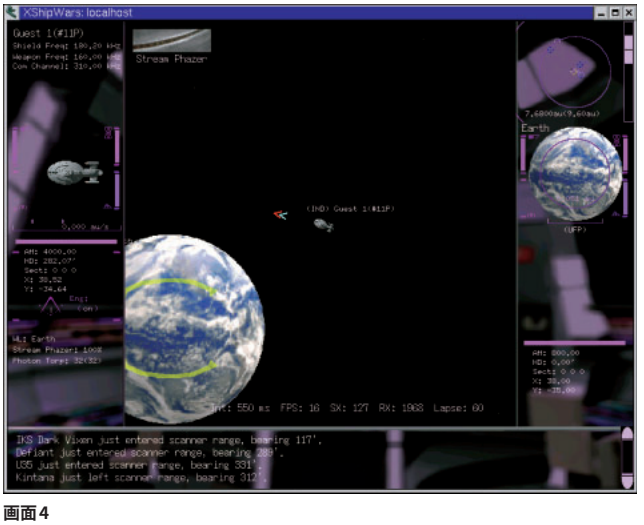
サーバを起動するには、rootになった状態で「/home/swserv/restart&」とする。さらに、「/home/swserv/bin/monitor&」でモニタを起動すると、サーバのモニタリングやシャットダウンが可能だ（**画面6**）。

なお、XShipWarsのWebサイトには、クライアントやサーバ、インストールなどに関する詳細なHTML形式のマニュアルやFAQが用意されている。詳しい操作や設定などについてはこれらを参照されたい。

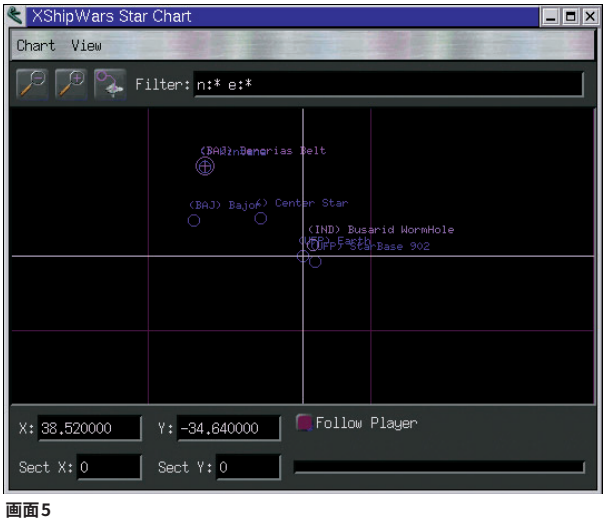


画面3
参加するユニバースを選択して、サーバに接続する。

画面6
サーバのモニタリングやシャットダウンを行うモニタ。



画面4
中央のビュースクリーンをはじめ、さまざまな情報が表示される。



画面5
スターチャートウィンドウで自分の宇宙船の位置を確認する。

ペットを連れて、さあ冒険に出かけよう

Japanese NetHack

バージョン：1.1.5　　ライセンス：GPL

http://www.jnethack.org/jnethack.html

Japanese NetHackは、プレイするたびにマップが変化するRogueタイプのロールプレイングゲームだ。この手のゲームの元祖、Rogueのクローンであるnethackを日本語化したものだ。昔のNetHackは「@」などの文字だけで表示される端末ゲームだったが、現在ではGTK＋を利用したグラフィカルな画面でも楽しめる。日本語化されているため、英語が苦手でも大丈夫だ。

●ビルドとインストール

Japanese NetHackは、tarボールのみ配布されている。Linuxだけでなく、＊BSDなど他のUNIX系OSやWindows 9x用のソースもまとめられているため、ビルドする前にいくつかファイルを書き換える必要がある。

最低限の修正でLinux用のバイナリを作成するには、展開先のディレクトリのsrc/Makefileの178行以降のWINTTYLIBを「WINTTYLIB = -termcap」、185行以降のWINX11LIBを「WINX11LIB = -lXaw -lXmu -lXext -lXt -lXpm -lX11 -lm」に変更すればいい。

ビルドとインストールは、「make」→「make install」という手順で行う。/usr/gamesに実行ファイルjnethackが、/usr/games/lib/jnethackdir以下にデータが格納される。

●まずはキャラクターを決める

コマンドラインで「jnethack&」として起動すると、スプラッシュ画面が表示された後、メインウィンドウが開く。環境変数PATHに/usr/gamesが含まれていない場合は、ホームディレクトリの.bash_profileを編集して/usr/gamesを追加しよう。

まずは、[ゲーム]－[遊ぶ]以下のサブメニューから、プレイするキャラクターを選択する。NetHackのキャラクターは職業と種族が入り混じっており（最新版では分離された）、考古学者、野蛮人、洞窟人、エルフ、戦士、薬師、騎士、僧侶、盗賊、侍、旅行者、ワルキューレ、魔法使いの13種類。キャラクターごとにクセがあり、初期装備や戦い方などが大幅に異なる。最初は、戦闘のしやすい野蛮人や洞窟人、ワルキューレなどがお勧めだ。

キャラクターを選択すると、メインウィンドウにあなたのパラメータやマップが表示されるとともに、キャラクターごとに異なるイントロが表示される（**画面1**）。キャラクターごとの神様（たとえばワルキューレならオーディン）が「イェンダーの魔除け」を欲しているという内容だ。

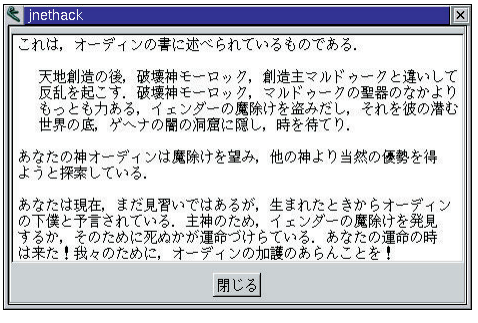
●基本的な操作

操作は基本的に1文字のコマンド入力で行う。上下左右への移動はviライクなhjklで行う。怪物は斜めにも移動するため、yubnキーを使った斜め方向の移動も大切だ。なお、進行方向に怪物がいる場合は、移動するかわりにその怪物への攻撃となる。

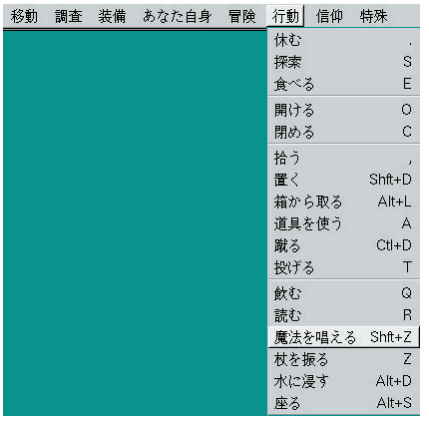
このほか、基本的なコマンドとしては以下のようなものがある。

i	所持品の一覧を表示する
w	武器を持ち替える
W／T	防具類を付ける／脱ぐ
t	武器などを投げる
,	足元のものを拾う
e	食物を食べる
q	薬などを飲む
r	巻物や魔法書を読む
o／c	ドアを開ける/閉じる
＜／＞	階段などを昇る／降りる
s	隠し扉などを探索する
.	休憩して体力を回復する

このほかにも多くのコマンドが用意されおり、ジャンルごとにメニューに分かれている（**画面2**）。各項目には対応するキーも表示されているので、少



画面1
選択したキャラクターごとに異なるイントロが表示される。



画面2
コマンドは1文字入力で行うほか、メニューからの選択も可能。

画面3
メインウィンドウには、ダンジョンのマップなどが表示される。

画面4
ダンジョン内にはアイテムを売買できる店などが存在する。

しずつ覚えていけばいい。

●ペットとともに行動せよ

ダンジョンの階層はだいたい25〜30階の深さで、各階層にはいくつかの部屋が通路で結ばれている（**画面3**）。階層内の部屋を探索しながら、怪物と戦い、落ちているアイテムや金を拾いつつ、下の階層への階段を探す、というのが基本的な行動パターンだ。部屋の中には、アイテムを売買する店（**画面4**）や、玉座、動物園、寺院などの特殊なものもある。

地面に落ちているアイテムのうち、巻物や魔法書、薬、指輪などは、仮の名前や色で表示され（**画面5**）、実際に使ってみるまで効能がわからない（使わずに知る方法もいくつかある）。中には、使用結果をふまえてプレイヤーが名づけなければならないこともある。たとえば、装備している武器が青く輝く巻物は「武器に魔法をかける(enchant weapon)」だ。

ダンジョンで出会うたいていの怪物は攻撃的だが、中立的・友好的なものもいる。特に、プレイ開始直後に側にいる犬（または猫）は、あなたと共に行動するペットだ（名前をつけられる）。ペットは、怪物を攻撃してくれたり、呪われていないアイテムだけを拾ったり、ときには店の売り物をちょろまかしてくれる大切な存在だ。

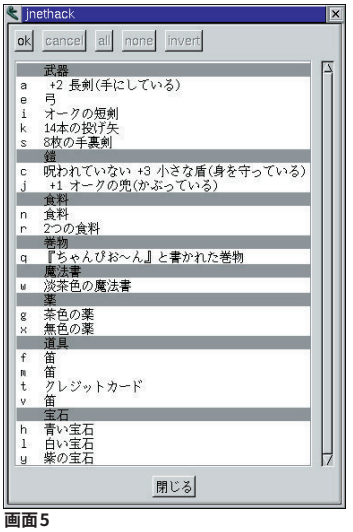
なお、ペットも腹をすかし、あなたと同様に成長する。適当なタイミングでエサとなる干し肉などを投げて与えよう。魔法を使ってペットを別の怪物に変化させたり、他の怪物をペットにすることも可能だ。

●困ったときには攻略本を

ダンジョンには、通常の階層のほかにも、ノームの鉱山（**画面6**）や神託所、城、ゲヘナといった「特別レベル」が存在する。こうした特別レベルも含めた行動については、とてもここでは書ききれない。

NetHackでは、通常のコンピュータRPGに比べて行動の自由度が極めて高く、中にはソースコードを読まないと分からないような仕掛けも存在する。ソースコードを読むのもNetHackの楽しみのひとつだ。

最近は日本語で書かれた攻略用の文書も充実しており、初心者なら「猿でもできるJNetHack」（付属CD-ROMに収録）、さらに深い知識が必要になったら「JNetHack Spoilers」（http://www.ke.ics.saitama-u.ac.jp/matsuda/nethack/jspoiler/）を参照するとよい。あなたが想像しているよりも、はるかに多く用意された仕掛けの数々に驚嘆すること請け合いだ。



画面5
仮の名前が付けられた巻物は、実際に使うまで効能がわからない。

画面6
通常の階層とは異なる外見の「特別レベル」も用意されている。

163種類ものソリティアを集めたカードゲーム

PySol

バージョン：3.40　　ライセンス：GPL

http://wildsau.idv.uni-linz.ac.at/mfx/pysol.html

PySolは、バリエーション豊かなカードゲームだ。バージョンアップされるたびにゲームの種類が増え、おなじみの「クロンダイク」や「フリーセル」をはじめ、最新版の3.40ではなんと163種類にも及ぶトランプや花札の一人遊び（ソリティア）をプレイできる。さらにプラグイン形式で新たなゲームを追加できるほか、カードの大きさやデザインを変更するカードセットも提供されている。

●ビルドとインストール

PySolは、ファイル一式をtar＋gzipしたtarボールで配布されている。インタプリタ言語のPythonで記述されているため、コンパイルの必要はない。なお、Pysolの実行にはPython 1.5.2とTcl/Tk 8.0以降が必要だ。

tarボールを展開したディレクトリで「make install」とすると、/usr/local/binに実行ファイルが、/usr/local/share/pysol/3.40にデータファイルがそれぞれコピーされる。

続いてカードセットを導入しよう。別途配布されているtarボールを展開後、「cp -a data/cardset-＊ /usr/local/share/pysol/3.40」として、Pysolのデータディレクトリにまとめてコピーすればいい。

このほか、サウンドサーバを導入すると、実行時にWAVEやMP3によるBGM再生が可能になる。Linux用として、SDLなど必要なライブラリをスタティックリンクしたpysolsoundservermodule.soが用意されているので、これを/usr/lib/python1.5/site-packagesにコピーしておこう。

●ゲーム中の操作

「pysol&」として起動すると、BGMの演奏とともに、クロンダイクのカードが配られてプレイできる状態になる（**画面1**）。なお、次回からは、終了前にプレイしていたゲームが自動的に開始される。

別のゲームに切り替えるには、[File]－[Select game]以下から選択する。ゲームが163種類もあるため、いくつかのサブメニューに分かれている。最初は、[Popular games]以下のクロンダイク、フリーセル、スパイダーといった代表的なゲームをプレイするとよいだろう。プレイ可能なプレビュー画面を備えたウィンドウでゲームを選ぶことも可能だ（**画面2**）。

ゲーム中の操作にはマウスを使用する。左ボタンのクリックでタロン（置札）から新しいカードを引き、ドラッグでタブロー（一時的なカード置き場）に移動するというのが基本的な操作だ。このほか、中ボタンのクリックで積まれている途中の表向きのカードを確認でき、ダブルクリックか右ボタンクリックでファウンデーション（台札）へカードを移動できる。

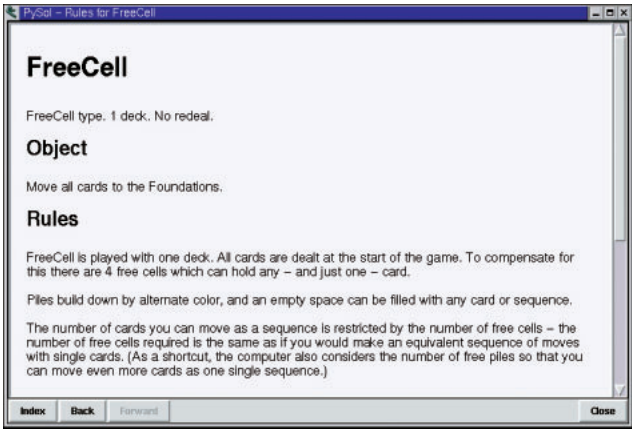
操作の手間を減らすには、[Options]－[Automatic play]以下の項目を選択する。伏せてあるカードを表向きにする、カードを配る、ファウンデーションへカードを移動するという3種類の動作を自動化できる。ゲームの種類によっては自動化しないほうがよいこともあるので気をつけよう。

ツールバーには、ゲームを最初からやり直す[Restart]、現在の状態をファイルに保存して後で読み込む[Save]・[Open]、新しい配置でゲームをやりなおす[New]などのボタンが用意されている。クリア時に表示されるCongratuation画面（**画面3**）を目指し

画面1
クロンダイクをはじめ、163種類ものゲームをプレイできる。

画面2
プレビュー画面でそのままゲームをしてみることも可能だ。

画面3
見事にクリアすると、このCongratuation画面が表示される。



画面4
よく知らないゲームは、まずルールの確認から始めよう。

てがんばってほしい。

●よく知らないゲームをプレイする

ルールがよくわからない場合は、ツールバーの[Rules]ボタンを押してルールの解説を表示する（**画面4**）。説明は簡潔な英語で書かれているので、読むのはそれほど難しくない。なお、ポピュラーなゲームのバリエーションの場合は、基本となるルールとの相違点だけが記述されている。

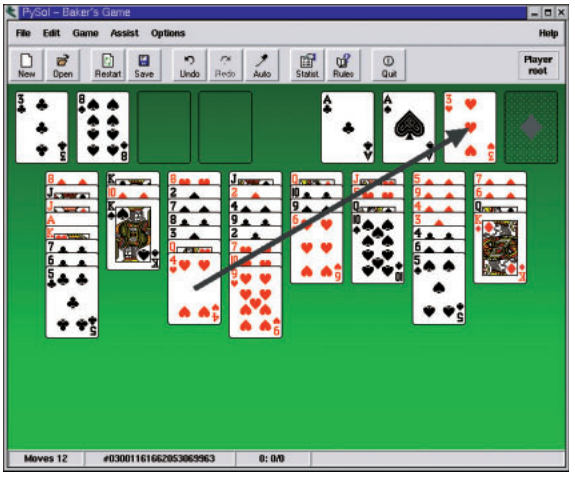
ある程度ルールを把握したら、試しにプレイするのが理解への早道だ。初めのうちは、操作を遡ってやり直せるアンドゥ機能（ツールバーの[Undo]ボタンかZキー）、動かせるカードを教えてくれるヒント機能（[Assist]－[Hint]かHキー）、コンピュータがプレイするデモ機能（[Assist]－[Demo]かCtrl－Dキー）などの機能を使うとよい（**画面5**）。ただし、ヒントやデモで示される手は、単にカードを動かせるというだけで最善の手ではない。特に、難易度の高いゲームでは手詰まりになりやすいので注意されたい。

●豊富なカードセットを楽しもう

[Options]以下の項目により、カードの裏面のデザインやテーブルの色使いを変更できる。たとえば、裏面のデザインを変更するには、[Options]－[Card background]で一覧表示される中から選択すればいい。[Options]－[Save options]で、設定がファイルに保存される。

カードの大きさや表面・裏面のデザインをまとめて変更するには、[Options]－[Cardset]を選択する（**画面6**）。このウィンドウでは、カードセットが大きさや年代などによって分類され、プレビュー画面を眺めながら、好みのカードセットを選択できる。たとえば、画面の狭いノートパソコンでは、「2000」や「tuxedo」を選択して、ひと回り小さなカードを使うと操作性がアップするだろう。

ディスクに余裕があるなら、ぜひとも別途配布のカードセットをインストールしよう。丸いカードの「Get A Round」や、アンティーク調の「Denizens」、花札の「Kintengu」など、バリエーション豊かな51種類のカードセットが追加される。なお、花札のカードセットは、花札のソリティア（「MatsuKiri」など8種類）専用だ。他のゲームで使うと、カードの番号がわからないため正常にプレイできない。



画面5
デモ機能を利用してゲームを自動的に進めることも可能だ。

画面6
プレビュー画面でデザインを確認しながらカードセットを選択。

多人数対戦も可能な海戦シミュレーションゲーム

Batalla Naval

バージョン：0.74.0　　ライセンス：GPL

http://batnav.sourceforge.net/

Batalla Navalは、スペイン生まれのマルチプレイヤー海戦シミュレーションゲームだ。懐かしのボードゲーム「バトルシップ」のように、相手がボード上に配置した船の位置を探りつつ、ミサイルを発射して破壊する。1対1の対戦だけでなく、最大8名までの同時対戦に対応しているのが特徴で、人間だけでなくロボットを相手にプレイすることもできる。動作にはGNOME／GTK＋が必要だ。

●ビルドとインストール

Batalla Navalは、サーバ／クライアント／ロボットのプログラムが、1つのtarボールにまとめられて配布されている。今回、日本語カタログ（ja.po）を含む日本語対応パッチ（gbatnav-ja.patch）を用意した。ビルドする前に、「patch -p1 < gbatnav-ja.patch」とすると、日本語環境ではメニューなどが日本語で表示される。

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。/usr/local/gamesに、gbnserver（サーバ）、gbnclient（クライアント）、gbnrobot（ロボット）の各ソフトがインストールされる。

●サーバとクライアントの起動

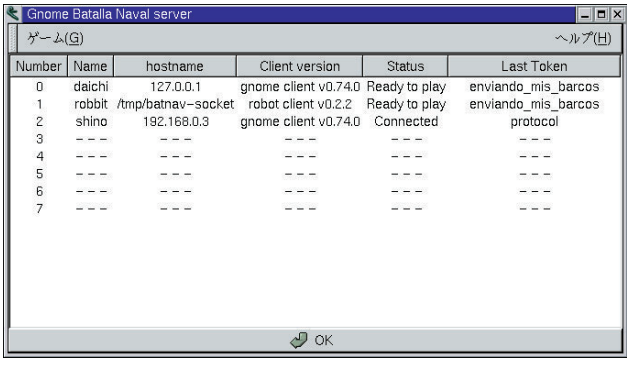
まずは、ネットワーク接続されたマシンのひとつでサーバを起動する。コマンドラインで「gbnserver&」とすればいい。サーバのウィンドウには、最大8名までのプレイヤーのリストが表示される（**画面1**）。

続いて、ゲームに参加する各マシンで、クライアントを「gbnclient&」として起動する（**画面2**）。サーバと同じマシンでクライアントを起動した場合は、自動的にサーバに接続した状態になる。

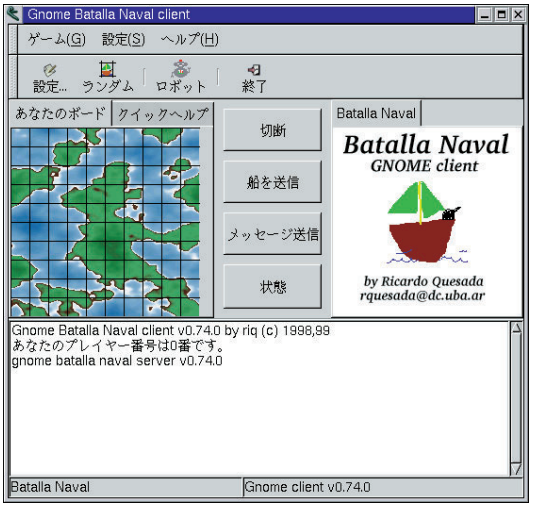
サーバとは別のマシンでクライアントを起動したときは、ツールバーの[設定]ボタンを押してダイアログを開き、[Server name]にサーバのホスト名を設定する（**画面3**）。このほか、ユーザー名の変更も可能だ。

なお、インターネット経由での接続も可能だが、サーバの一覧を表示する機能は用意されていないので、あらかじめサーバのドメイン名やポート番号を調べておく必要がある。

ダイアログを閉じて、中央の[接続]ボタンを押すと、設定したサーバに接続され、中央の[船を送信]ボタンなどが使用可能になる。



画面1
1つのサーバには、プレイヤーが8名まで参加できる。

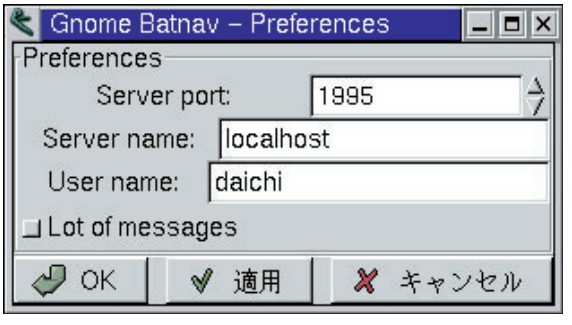


画面2
プレイヤーごとに起動するクライアントのウィンドウ。

●船をボード上に配置する

ゲームを開始する前に、あなたの船をボード上に配置する必要がある。船は全部で10隻で、

4マスを占める船…1隻
3マスを占める船…2隻



画面3
サーバを動かしているマシンのホスト名を設定する。

2マスを占める船…3隻
1マスを占める船…4隻

という構成だ。これらを、互いに隣接しないようにボード上にクリックで配置する（**画面4**）。なお、いちいち手で配置するのが面倒なら、ツールバーの[ランダム]ボタンで自動的に配置してくれる。配置が完了したら、ウィンドウ中央の[船を送信]ボタンを押し、プレイの準備が整ったことをサーバに伝えよう。

人間の対戦相手がいない場合は、ツールバーの[ロボット]ボタンを押して、サーバを動かしているマシンでロボットを起動する（**画面5**）。複数のロボットを同時に実行することも可能だ。サーバのリストにロボットが追加されたことを確認しよう。クライアントからも、ウィンドウ中央の[状態]ボタンで調べられる。

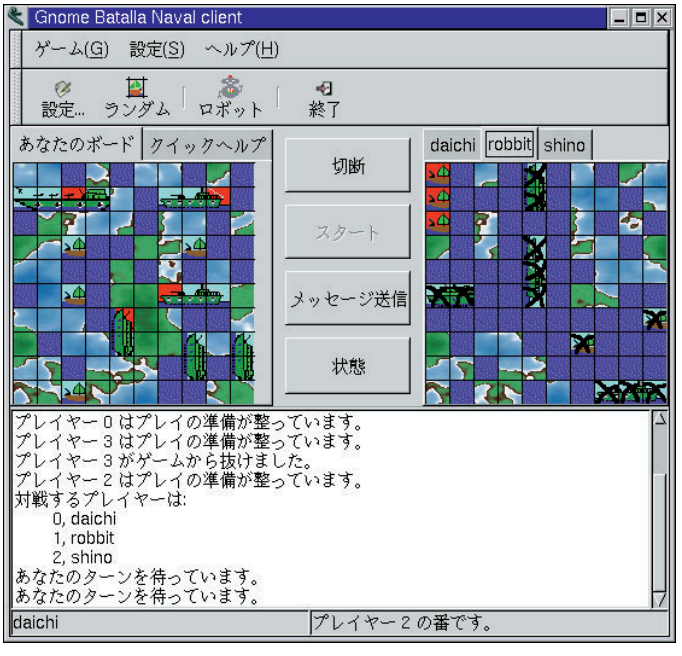
続いて、中央の[スタート]ボタンを押すと、プレイ準備が整っている人間やロボットとのプレイが開始される。1対1の対戦をはじめ、8名までのマルチプレイヤー対戦も可能だ。

●相手の船の位置を読め

プレイが始まると、ウィンドウ右のエリアに各プレイヤーのボードが表示される。ボードはプレイヤーごとに完全に独立している。ゲームはターン制で、各ターンに1回ずつ、自分以外の任意のプレイヤーのボード上のマスをクリックし、ミサイルの発射を指示できる（**画面6**）。

クリックしたマスに相手が船を配置していた場合、1マスだけの小さな船ならば即座に破壊され、×印付きで表示される。一方、2マス以上を占める大きな船の場合は、赤い被弾マークが表示され、残りのマスがすべて被弾した時点で破壊される。

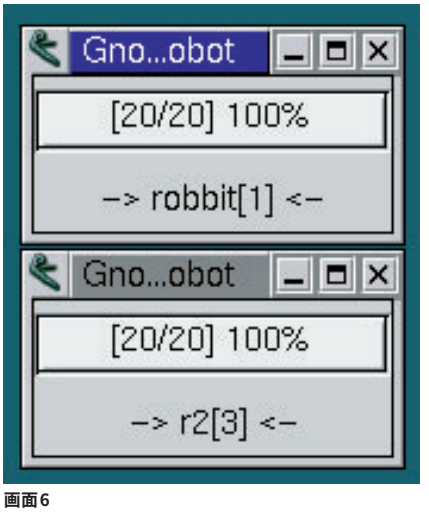
被弾マークが表示されたら、その上下左右のいずれかに必ずその船の残りのマスが存在するので、連続して攻撃しよう。また、船は隣接できないので、被弾したマスの斜め4マスや、破壊した船の周囲のマスには、ほかの船が存在しないことが確定する（自動的に開いた状態になる）。このため、大きなサイズの船を素早く発見することが勝利の秘訣だ。



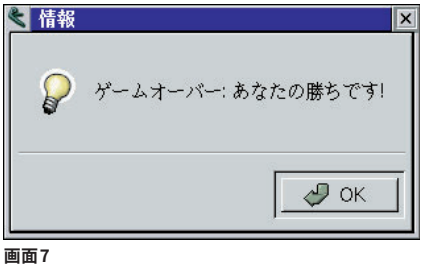
画面5
コンピュータが操作するロボットとの対戦も可能だ。

配置した船がすべて破壊されたプレイヤーから抜けていき、最後に残ったプレイヤーが勝者となる（**画面7**）。なお、ロボットのプレイヤーは自分が負けたり、プレイが終了すると自動的に終了するので、プレイのたびに起動する必要がある。

画面4
ボード上に10隻の船を隣接しないように配置しよう。



画面6
相手の船の位置を読んでミサイルを発射しよう。



画面7
最後まで残ったプレイヤーが勝利者となる。

プログラムしたロボットを戦わせるシミュレーションゲーム

RealTimeBattle

バージョン：1.0.3　　ライセンス：GPL

http://realtimebattle.sourceforge.net/
http://michiko.shiratori.riec.tohoku.ac.jp/rtb/jp/Main.html（日本語）

RealTimeBattleは、自分でプログラムしたロボットをアリーナで戦わせるシミュレーションゲームだ。1対1の対戦から、最大120体のバトルロイヤルまで、さまざまな条件での戦闘が可能だ。ロボットは、実際の物体のように慣性や摩擦の影響を受け、エネルギーを消費する。C／C++／Perl／Java／Schemeなど、一般的なプログラム言語でロボットを作成できるのが最大の特徴だ。

●ビルドとインストール

RealTimeBattleは、tarボールでのみ配布されている（tarボールにSPECファイルが含まれるので、自分でRPMパッケージを作成することは可能）。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、tarボールには日本語カタログが含まれており、環境変数LANGが適切に設定されていれば、ウィンドウの表示が日本語になる。

●トーナメントの条件を設定

コマンドラインで「RealTimeBattle&」として起動すると、各種のボタンが並んだコントロールウィンドウが開く（**画面1**）。

RealTimeBattleでは、同じロボット同士の一連のゲームを「シーケンス」、シーケンスの集合を「トーナメント」と呼ぶ。トーナメントを開始するには、[新規トーナメント]ボタンを押して、トーナメントの条件を設定するウィンドウを開く（**画面2**）。

まず、参加するロボットの種類と、舞台となるアリーナを選択する。また、1シーケンスあたりのゲーム数とロボット数、トーナメント全体でのシーケンス数を設定することで、「1対1の総当り戦」や「全ロボットのバトルロイヤル」などを実現できる。

[スタート]ボタンを押すとトーナメント開始だ。なお、現在の設定をファイルに保存したり、ファイルから読み込むこともできる。

●トーナメントを実行する

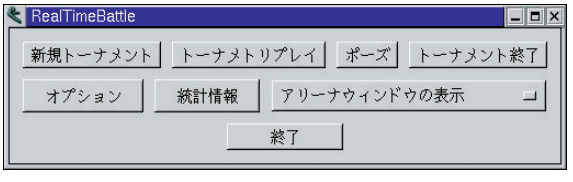
トーナメントが始まると、各シーケンスのゲームが順番に行われ、各ロボットのプログラムが子プロセスで起動されて戦闘が行われる。ゲーム中のようすは、アリーナ／メッセージ／スコアの3ウィンドウにリアルタイム表示される（**画面3**）。

ロボットはタイヤ付きの乗物で、周囲の情報を得るレーダーと、敵のロボットなどを破壊するキャノンを備えている。レーダーとキャノンは、どちらもロボットの向きとは独立して向きを変えられる。

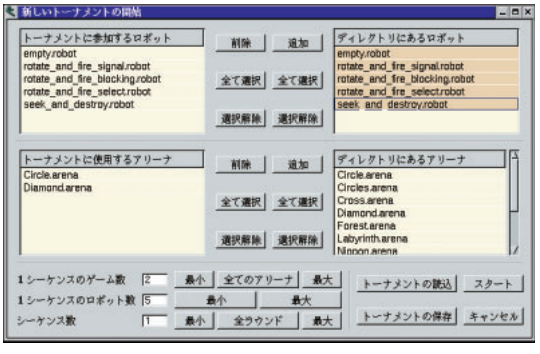
ロボットの状態はエネルギーで表わされる。エネルギーは、キャノンを撃ったり、敵の攻撃を受けたり、壁やロボットに衝突すると減少する。また、ランダムに登場する「クッキー」を取るとエネルギーが回復、「地雷」を踏むと減少する。

敵のロボットを破壊する（エネルギーを0以下にする）と、1ポイントが加算される。アリーナ内のロボットが1台以下になるか、タイムアウトになるとそのゲームは終了だ。

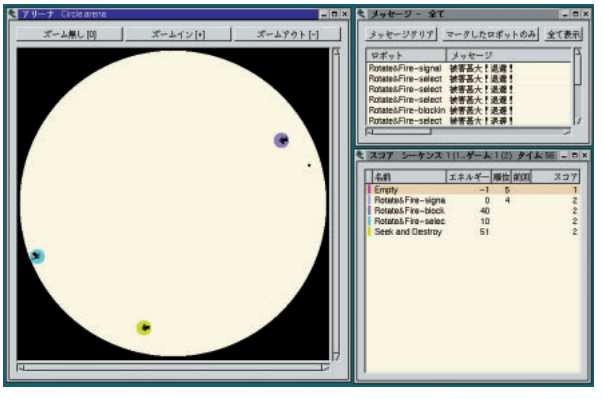
なお、コントロールウィンドウの[統計情報]ボタンを押すと、戦闘結果をさまざまな観点で整理する統計情報ウィンドウが開く（**画面4**）。



画面1
トーナメントの開始はこのコントロールウィンドウから指示する。

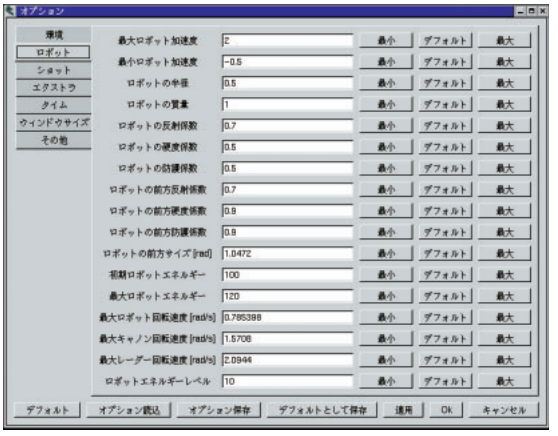


画面2
トーナメントに参加するロボットや舞台となるアリーナを選択。



画面3
アリーナ、メッセージ、スコアがリアルタイムに表示される。

画面4
戦闘結果をさまざまな観点から整理してくれる統計情報ウィンドウ。



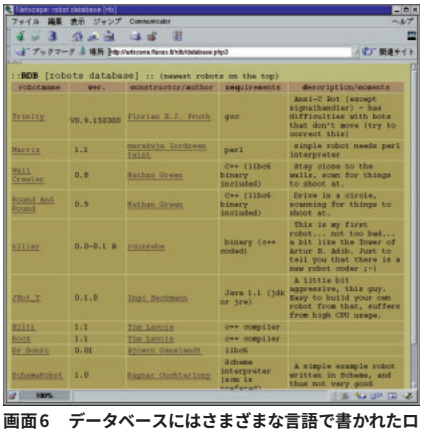
画面5
設定ダイアログでは、ゲームに関する細かな設定を変更できる。

●設定は細かく変更できる

コントロールウィンドウの[オプション]ボタンで開く設定ダイアログには、トーナメントに関するさまざまな設定がジャンルごとにタブ付きページに分けられている（**画面5**）。

たとえば、[環境]ページでは、ロボットの動きに影響を与える重力や空気抵抗、摩擦などのパラメータ、[ロボット]ページではロボットの初期エネルギーや加速度の最大値などを設定するといった具合だ。このほか、クッキーや地雷に関する設定、時間に関する設定、ウィンドウサイズなども細かく変更できる。

自分でロボットのプログラムを作成する場合は、[その他]ページの[ロボット検索パス]に、自分のロボットの実行ファイルを置いたディレクトリをを記述するといい。トーナメントの条件設定ウィンドウ（**画面2**）のロボット一覧に、そのディレクトリのファイルが含まれるようになる。



画面6　データベースにはさまざまな言語で書かれたロボットが登録済みだ。

●ロボットのプログラミング

他人が作ったロボットの戦いを眺めているだけでは、RealTimeBattleの楽しさは半分も理解できない。やはり、自分のロボットをトーナメントに参戦させることに意味がある。

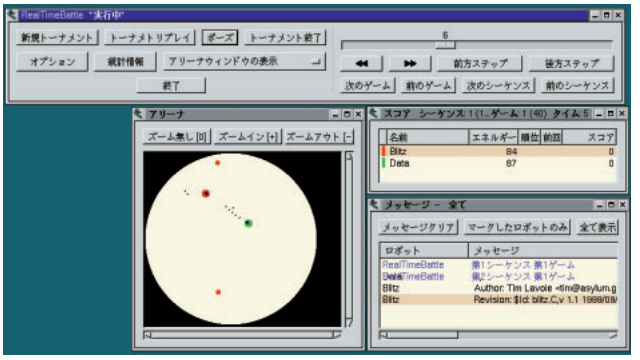
ロボットのプログラムは、RealTime Battle本体から子プロセスとして起動され、標準入出力（stdin/stdout）を利用して本体とメッセージ交換を行う。このため、たいていのプログラム言語でロボットを作成可能だ。

実際、ロボットデータベースのページ「MrFrost」（http://artscene.fluxus.lt/rdb/）には、CやC++をはじめとして、PerlやJava、Schemeなどで書かれたロボットが登録されている（**画面6**）。これらをダウンロードして、あなたのロボットと戦わせることもできるし、あなたのロボットをこのページに登録することも可能だ。

メッセージの種類や受け取り方など、ロボットの作成に関する詳細は、Webサイトのマニュアルを参照されたい（日本語のマニュアルは日本語サイトで読める）。

また、印刷可能なPS／DVI／テキスト形式のマニュアル（英語版）も配布されている。

なお、RealTimeBattleのWebサイトでは、ユーザーから送られてきたロボットによるコンペティションが行われており、順位や勝敗などの結果とともに各ゲームのログファイルも配布されている。これをダウンロードして、コントロールウィンドウの[トーナメントリプレイ]ボタンで読み込むと、あなたのマシンで各ゲームの進行を再現できる（**画面7**）。



画面7
ログファイルを読み込んで、ゲームの進行を再現可能だ。

コミカルな風船割りアクションゲーム

Circus Linux

バージョン：0.0.2　　ライセンス：GPL

http://www.newbreedsoftware.com/circus-linux/
http://www.devolution.com/~slouken/SDL/ （SDL）

Circus Linuxは、昔懐かしいアーケードゲームの「サーカス サーカス」によく似たゲーム。シーソーを利用して2人のピエロを交互にジャンプさせ、画面上方を左右に流れる風船を割っていく。ルールは単純ながら、コミカルなキャラの動きや効果音、BGMが気分を盛り上げてくれる完成度の高いゲームだ。ピエロを跳ね返す障害物を追加したり、2人で交互にプレイすることもできる。

●ビルドとインストール

動作にはクロスプラットホームなマルチメディアライブラリのSimple DirectMedia Layer（SDL）とSDL Mixer Libraryが必要だ。どちらもtarボールとRPMパッケージの両方で配布されている。Circus Linuxをビルドする前にインストールしておこう。

Circus Linuxは、tarボールでのみ配布されている。展開先のディレクトリでプレイする場合は、そのまま「make」でビルドすればいい。「./circuslinux」でプレイ開始だ。

実行ファイルやデータファイルを/usr/local/games以下に置くには、Makefileの12行目を「DATA_PREFIX=/usr/local/games/circuslinux-data/」に変更してからビルドする。インストールは手動で行う。circuslinuxを/usr/local/gamesに、data以下のファイルを/usr/local/games/circuslinux-dataにコピーすればいい。

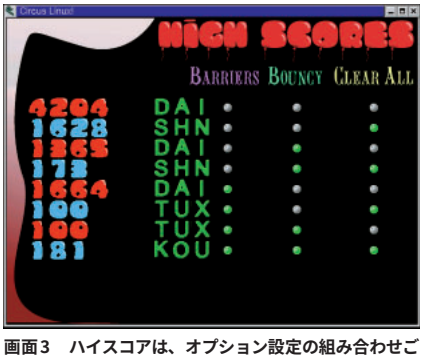
●マウスでシーソーを操作する

「circuslinux」として起動すると、コミカルなBGMとともにタイトル画面が表示される（**画面1**）。ソロプレイなら「ONE PLAYER」、2人でプレイするなら「TWO PLAYERS」（交互に独立した面をプレイ）か「TWO PLAYERS COOP」（相手の続きの面をプレイ）をマウスでクリックすればいい。

オプションとして、「BARRIERS」（障害物が現れる）、「BOUNCY BALLOONS」（風船に当たると跳ね返る）、「CLEAR ALL」（すべての風船を割るまで新しい風船が現れない）を設定できる。複数の組み合わせも可能だ。

シーソーの操作にはマウスを使用する。左クリックでピエロが落ちてくるので、マウスを左右に移動させて受け止めよう（**画面2**）。受け止めた反動でシーソー上のピエロがジャンプする。こうして交互にピエロをジャンプさせて風船を割っていくのだ。高くジャンプさせて一度に多くの風船を割るには、シーソーのできるだけ端のほうでピエロを受け止める必要があるが、それだけミスする危険も増す。

なお、ハイスコアは、オプションの組み合わせ（8通り）のそれぞれについて記録される（**画面3**）。



画面3　ハイスコアは、オプション設定の組み合わせごとに記録される。

画面1　風船が乱れ飛ぶタイトル画面。障害物などの設定も可能だ。

画面2　シーソーを動かし、ピエロをジャンプさせて風船を割れ。

迷路を走ってフラッグを取るレースゲーム

XRally

バージョン：0.9　　ライセンス：GPL

http://www.linuxgames.com/xrally/

XRallyは、昔懐かしいアーケードの「ラリーX」によく似たレースゲームだ。敵の車にぶつからないように、自分の車を迷路の中を走らせ、スモークで敵の車を混乱させながらコース上のフラッグを取っていく。シビアな状況判断が要求されるゲームだ。自分でコースを作成することも可能で、追加用のコースが別途配布されているほか、XRally専用のマップエディタも作られている。

●ビルドとインストール

XRallyはtarボールでのみ配布されている。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。Xのフォント設定の順番によっては、表示される文字が画面からはみ出してしまうことがある。これを解決するには、graphics.cのフォント指定（147〜149行目）を「"-*-*-…-*"」から「-misc-*-…-*」に変更するといい。

XRallyのアーカイブにはTest／Real／Leoの3つのLevel（コースをまとめたファイル）が付属する。さらに、Silver／MiraxというLevelが別途配布されており、付属のLevelと同じディレクトリに展開しておけば、XRallyの起動時に読み込んでくれる。

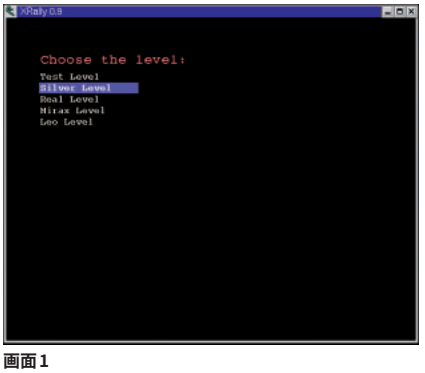
●カーソルキーで自分の車を操作

「xrally&」として起動すると、利用可能なLevelの一覧が表示される（**画面1**）。Testは文字通りテスト用なので、RealとLeo（追加レベルを展開した場合はSilverとMirax）の中からプレイするレベルを選択しよう（Leoは比較的難易度が低い）。

最初のマップが表示され、シグナルがブルーになったらゲーム開始だ（**画面2、3**）。自分の車が自動的に走り始めるので、カーソルキーを使って向きを変えよう。画面には自分の車の周囲しか表示されないので、右下のレーダーを参考にして、敵の車（赤）を避けつつ、フラッグ（黄）に向かう。目的は、すべてのフラッグを取ることだ。

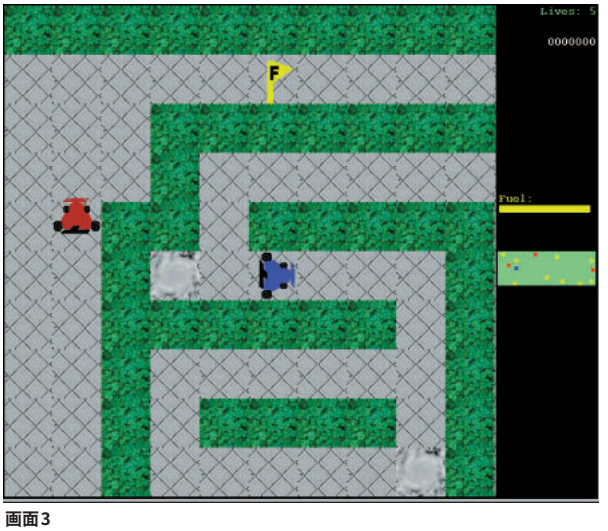
敵の車は自分の車よりもスピードが速いため、そのままでは必ず追いつかれてしまう。スペースキーでスモークを出し、敵の車の動きを止めよう。スモークは同時に2つまで出せる。このほか、敵の車同士が接触した場合もしばらく動きが止まる。敵の車と接触するか燃料切れになると、自分の車が1台失われ、すべての車を失うとゲームオーバーだ。

ところで、XRallyのLevelはテキストファイルなので、Emacsなどのテキストエディタで直接編集可能だ。また、画像からマップを生成するツールや、GUIでマップを編集するエディタも作られている。



画面1
まずは、プレイするレベルをTest以外から選択する。

画面2
敵の車をスモークで混乱させつつ、コース上のフラッグを取れ。



画面3
Leo Levelは、他のLevelよりいくぶん難易度が低い。

オーソドックスな縦スクロールシューティング

xsoldier

バージョン：0.96　　ライセンス：GPL

http://www.surfline.ne.jp/hachi/xsoldier.html

xsoldierは、X上で動作する縦スクロールシューティングゲームだ。画面は少々地味ながら、武器を変更・パワーアップするアイテムや、各ステージの最後に登場するボスキャラなど、シューティングゲームのツボを押さえた作りになっている。キーボードのほか、ゲームポートに接続したジョイスティックによる操作も可能だ。

●ビルドとインストール

xsoldierは、tar＋gzipされたtarボールのほか、JG（Japanese Game and amusements）にRPMバイナリパッケージとして収録されている（Vine用）。tarボールからビルドする場合は、「xmkmf -a」→「make」→「make install」という手順をとる。

どちらの場合も、実行ファイルは/usr/local/gamesにインストールされるので、環境変数PATHにこのディレクトリを含めておくといいだろう。

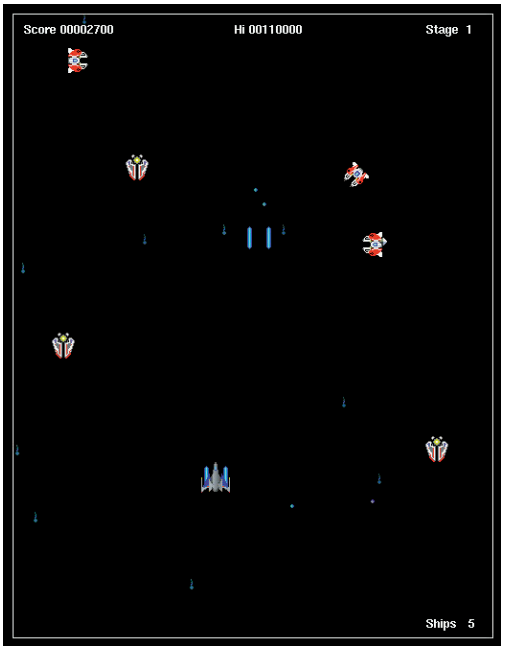
●アイテムでパワーアップ

ktermなどのコマンドラインで「xsoldier&」として起動すると、タイトルとスコアリストが表示される（**画面1**）。スペースキーを押すとプレイ開始だ。画面上部から敵が弾を撃ちつつ、次々と自機に接近してくる（**画面2**）。カーソルキーで上下左右に移動しながら、左Shiftキーで弾を撃って敵を撃破しよう。基本的に左Shiftキーは押しっぱなしにしていればいい。

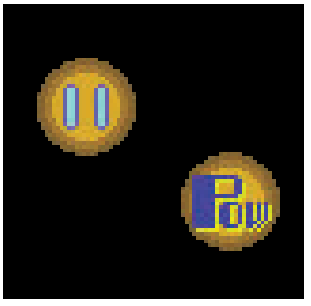
敵の中には、破壊するとアイテムを残していくものがある。アイテムは「武器チェンジ」と「武器パワーアップ」の2種類（**画面3**）。「武器チェンジ」は一定時間で表示が変化し、レーザータイプ（初期装備）・拡散タイプ・爆弾タイプに切り替わる。一方、「武器パワーアップ」は、武器の威力を増大させるもので、弾の数が増えたり、弾の飛ぶ方向が広がったりする。

全部で8ステージ用意されており、各面の終わりにはボスキャラが待っている（**画面4**）。ボスキャラは、攻撃の激しさ、耐久性ともにザコとは段違いなので、気合を入れて対処しよう。ボスを倒すとステージクリアとなり、各種ボーナス点が加算される。

画面1
タイトルとともにスコア上位の10名が表示される。

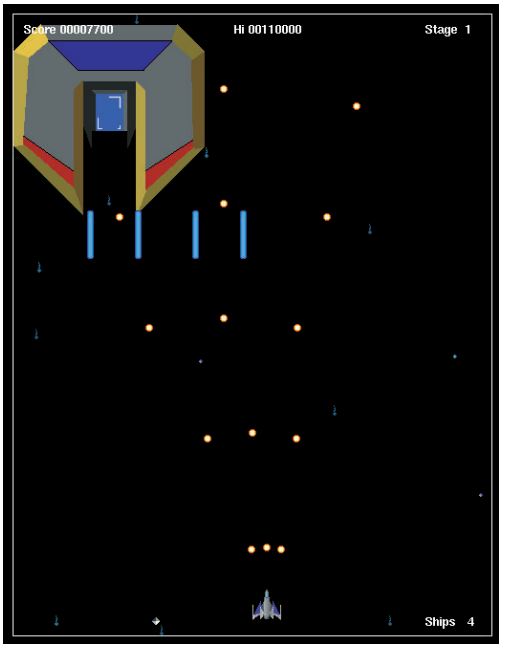


画面2
カーソルキーで移動しつつ、左Shiftキーで敵を破壊せよ。



画面3
アイテムは「武器チェンジ」と「武器パワーアップ」の2種類。

画面4
各ステージの終わりにはお約束のボスキャラが登場。



美しい盤面のピンボールゲーム

Roll'm Up

バージョン：－　　ライセンス：As Is

http://www.medialab.lostboys.nl/

Roll'm Upは、カフェの店内を模した美しい盤面と派手なサウンドが特徴のピンボールゲーム。オランダのLost Boys media labが技術プレビューとして公開しているもので、Linux版のほか、Windows版、BeOS版、Mac版も配布されている。バイナリ配布のため、Linux版の動作にはx86系で動作するglibc2ベースのLinuxが必要だ。

●同時に3個のマルチボールプレイ

Roll'm Upは、ファイル一式をtar＋gzipしたtarボールで配布されている。バイナリ配布なので適当なディレクトリ（/usr/local/gamesなど）にそのまま展開すればいい。展開先のディレクトリで「./Rollemup&」とすると、軽快なサウンドとともにタイトル画面が表示される（**画面1**）。

操作にはキーボードを使用する。Nキーを押すとゲーム開始だ（**画面2**）。Enterキーをしばらく押し、プランジャーでボールを打ち出そう。左右のフリッパーはそれぞれZキーと/キーで操作する。右上の小さなフリッパーも/キーで動く。

ピンボールゲームで必須の台を揺らす操作にはスペースキーを使用する。Roll'm Upでは、盤面の表示自体は揺れず、ボールの動きだけが変化するので注意されたい。もちろん、短時間に揺らしすぎるとティルトをくらってフリッパーが操作できなくなる。

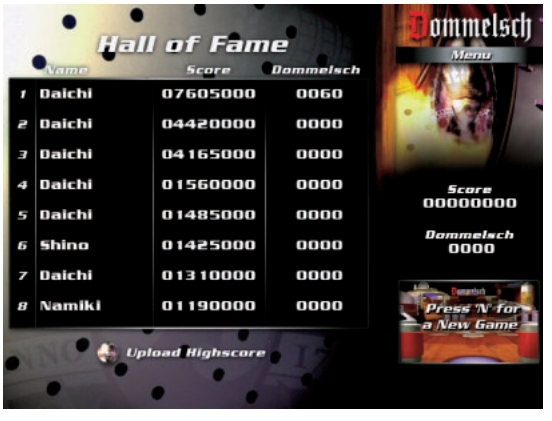
カフェの店内を模した盤面には、ピアノやビール缶の列、トランポリン、バーの椅子などが配置されており、これらを指示された順番で倒していく。次に狙う場所はウィンドウ右側に「Hit the Piano」などと表示されるので参考にしよう。最後に盤面上部に現れる「ボーナスホール」にボールを入れると、Roll'm Upの目玉である3個のボールを使ったマルチボールプレイが始まる（**画面3**）。

6個あるボールを使い果たしてゲームオーバーになると、ハイスコア画面に切り替わる（**画面4**）。インターネットのサーバに得点をアップロードすることも可能だ。1000万点はとらなければ話にもならないが…。

画面1
華麗なタイトル画面。Nキーを押すとプレイ開始だ。

画面2
左右のフリッパーや台を揺らすテクニックでボールを落とすな。

画面3
3個のマルチボールプレイは高得点のチャンスだ。



画面4
ハイスコアを叩き出したら、あなたの名前を登録しよう。

思考力が試される論理型パズルゲーム

KPooka

バージョン：0.3　　ライセンス：GPL

http://www.ucs.co.za/~pgr/kpooka/

KPookaは、赤い「ブロブ」を操作して、緑のブロブを押しのけながら出口を目指すパズルゲームだ。ペンゴや倉庫番といった名作パズルの流れを汲む論理型パズルで、アクション要素がない分、盤面の先を読む技術が要求される。パズルは全部で15レベル用意されており、後半には動かせないブロックや別の場所に転送されるテレポーターが登場して難易度が高くなる。動作にはKDE／Qtが必要だ。

●ビルドとインストール

KPookaは、tarボールとRPMバイナリパッケージの両方で配布されている。なお、バイナリパッケージは/opt/kde以下にファイルを展開するので、KDEが/usr以下にあるディストリビューションではtarボールからビルドしたほうがいい。tarボールからの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。

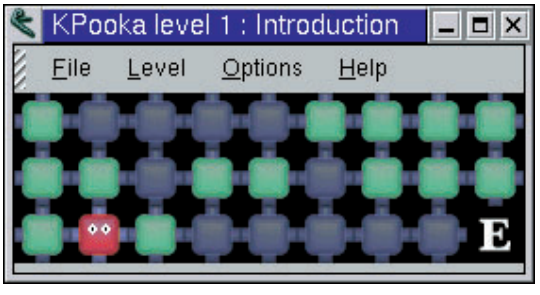
●赤いブロブを出口まで移動

コマンドラインで「kpooka&」とするか、KDEメニューの[ゲーム]－[KPooka]を選択すると、最初のレベル（面）が表示される（**画面1**）。なお、次回からは各ユーザーがまだ解いていないレベルが自動的に表示される。

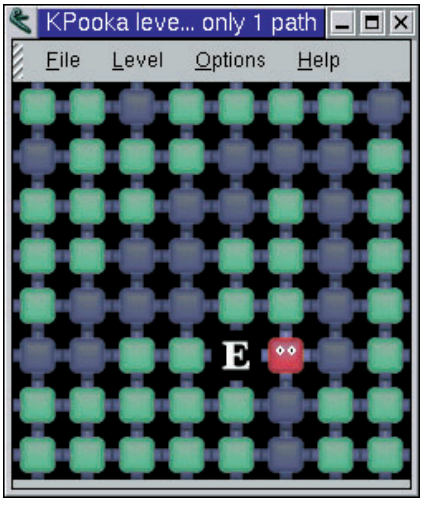
盤面に並ぶ四角いブロックは、上下左右がパイプで接続されている。目玉のある赤いブロブをカーソルキーで操作し、「E」と表示された出口のブロックまで移動させることがゲームの目的だ（**画面2**）。

いくつかのブロックには赤いブロブの邪魔をする緑のブロブが置かれていて、これを押しながら移動する必要がある。ただし、倉庫番などと同様、1つ後ろに空きブロックがある場合しか押すことはできない。押されたブロブは、他のブロブか盤面の端に達するまで自動的に移動する。

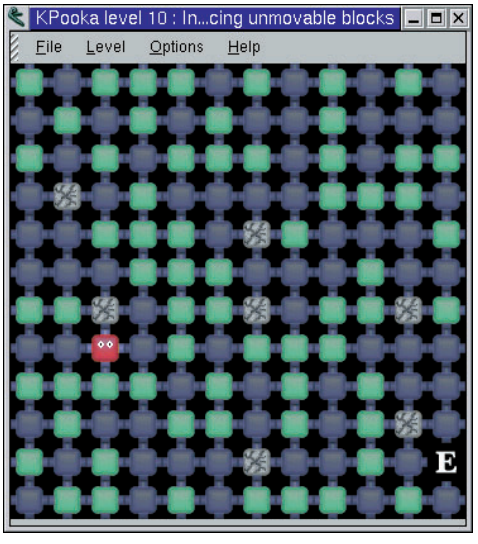
レベルが進むにつれ、押すことのできない灰色のブロック（**画面3**）や、別の場所に移動するテレポーター（**画面4**）といったギミックも登場する。なお、手詰まりになった場合には、Escキーでそのレベルの初期状態に戻り、何回でもやり直し可能だ。



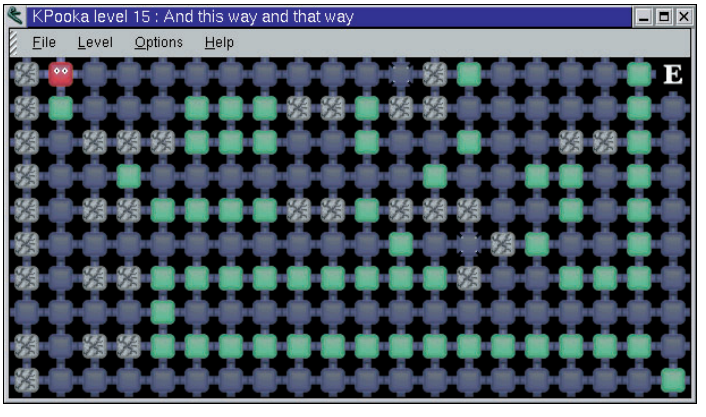
画面1
赤いブロブで緑のブロブを押しながら出口を目指せ。



画面2
出口に到達すれば次のレベルに進むことができる。



画面3
レベルが進むにつれ、動かせない灰色のブロックが登場する。



画面4
最終の15レベル。テレポーターを使って脱出路を確保せよ。

画面を回転させて球をゴールまで運べ

XROT

バージョン：1.3.2　　ライセンス：フリー

http://www.ci.cs.meiji.ac.jp:8150/siraisi/xrot.html

xrotは、画面を回転させて、落下する球をうまくゴールまで運ぶアクションゲームだ。早くゴールするためには、球が壁に衝突して生じるタイムロスを、できるだけなくすようにプレイしなくてはならない。慣れてくると、キー操作により球の速度を上げたり、跳ね返りを小さくすることができるようになる。動作にはPseudo Color（256色）表示をサポートしたXサーバが必要だ。

●ビルドとインストール

XROTは、ファイル一式をtar＋gzipしたtarボールでのみ配布されている。Imakefileを利用しているため、「xmkmf」→「make」→「make install」という手順でビルドとインストールを行う。インストール先の初期設定は/usr/X11R6/binだ。

なお、XROTはPseudo Color環境で動作するため、XFree86 3.xでは256色モードでのみ動作する。ハイカラーやフルカラーでXを動作させている場合は、XF86Configを修正するなどして、256色モードで起動しなおす必要がある。

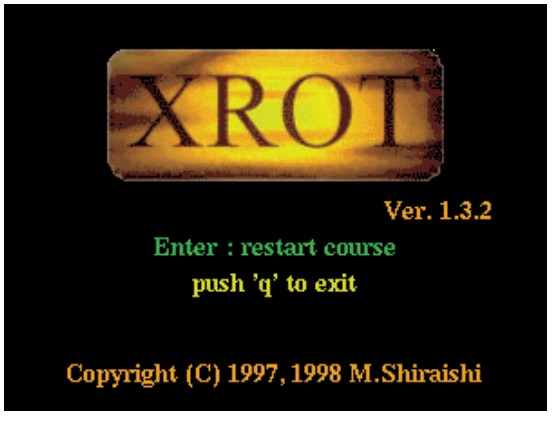
●あなたの挑戦を待つ7つのコース

「xrot」として起動すると、ロゴが回転するタイトル画面が表示される（**画面1**）。スペースキーで7つのコースのいずれかを選択しよう。

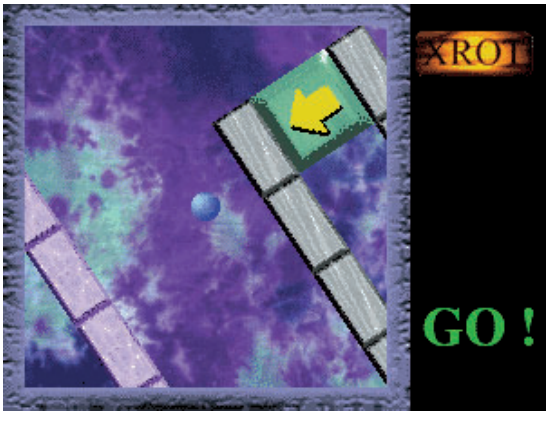
操作の基本は、←／→キーで画面を回転させて、常に下方向に動こうとする球をゴールまで誘導することだ（**画面2**）。壁に衝突すると、球の落下速度が鈍ってしまうので、できるだけ壁に当てないようにする。

このほか、スペースキーで球の速度アップ、↓キーで衝突時の跳ね返りの緩和、球の速度が小さい場合に限り↑キーでジャンプが可能だ。コースの難易度が上がるにつれ、通路の幅は狭くなり、コース上の障害物も増えてくるので、これらのキーをうまく使いこなそう（**画面3**）。

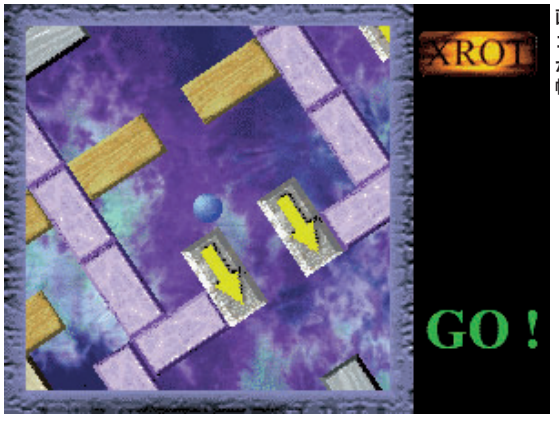
通常コースは一本道で、進行方向を示す矢印が壁に描かれているので迷うことはない。通路が複数に分岐していたり、斜め方向の通路があるなど、特に難易度の高いスペシャルコースも用意されている（**画面4**）。



画面1
ロゴが回転するタイトル画面。256色モードでXを起動しよう。



画面2
落下する球を画面を回転させてゴールまで運べ。



画面3
コースの番号が上がるにつれ通路の幅が狭くなる。

画面4
斜めの通路や迷路など特に難易度が高いスペシャルコース。

36種類の石を配置するパズルゲーム

Kishido

バージョン：1.0.2　　ライセンス：GPL

http://www.Informatik.Uni-Oldenburg.DE/~km/kishido/

Kishidoは、「石道」と呼ばれる日本発のパズルゲームのクローンだ。オリジナルはPC-8801シリーズ用で、1990年4月にアスキーから発売された。縦8×横12のマス目に区切られたボード上に、一定のルールに基づいて72個の石をすべて配置することがゲームの目的だ。上下左右を石で囲まれたマスほど条件はきつくなるが、それだけ高得点のチャンスだ。動作にはKDE／Qtが必要だ。

●ビルドとインストール

Kishidoは、tarボールとRPMパッケージの両方で配布されている。RPMバイナリパッケージはSuSe6.3用なので、他のディストリビューションではRPMソースパッケージをリビルドするとよいだろう。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。

●マークと色に注意して配置

コマンドラインで「kishido」とするか、KDEメニューの[ゲーム]－[kishido]を選択するとゲームが始まる（**画面1**）。石のマークと色はそれぞれ6種類ずつあり、同一の石が2つずつ（計72個）用意されている。ゲーム開始時には、ボードの四隅に1つずつ、中央に2つの石が置かれている。

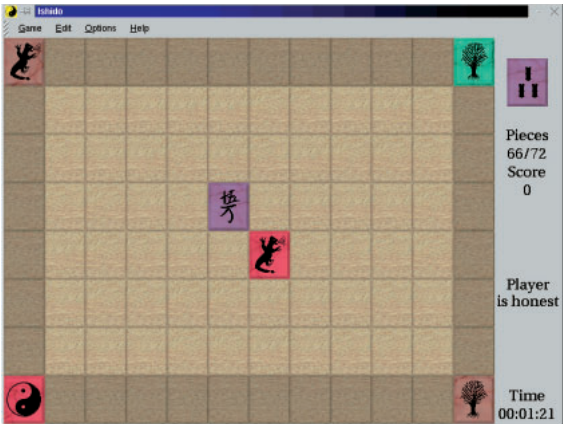
画面右上に次の石が表示されているので、その置き場となるボード上の空きマスをクリックしよう。石を置けるのは、少なくとも1つの石が周囲（上下左右の4マス）に置かれているマスだ。また、新たに置く石のマークと色は、次の条件を満たす必要がある。

- **周囲の石が1つ…マーク・色のいずれかが周囲の石と一致**
- **周囲の石が2つ…マークが周囲の石の一方、色がもう一方と一致**
- **周囲の石が3つ…マークが周囲の石の2つと一致し、色が残りの1つと一致（逆でも可）**
- **周囲の石が4つ…マークが周囲の石の2つと一致、色が残りの2つと一致**

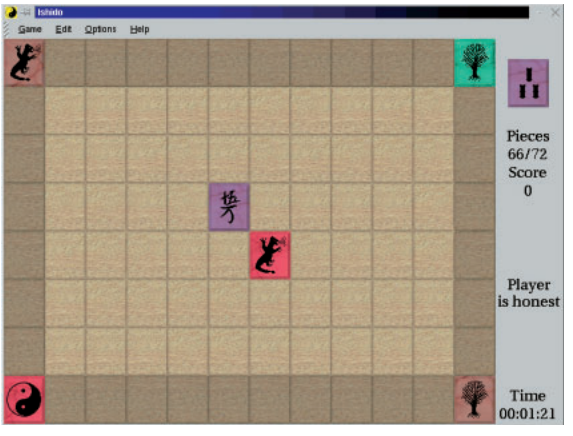
周囲の石が多いほど条件が厳しいが、それだけ得点も高くなる。周囲の石が1つだと1点、2つなら5点、3つなら20点、4つだと100点も入るのだ。

なお、マウスの右ボタンをクリックすると、石を置ける場所が赤く反転する（**画面2**）。ルールを覚えるまではこの機能を活用するとよいだろう（使い過ぎるとハイスコアが記録されない）。すべての石をボードに置くか、石を置ける場所がなくなるとゲームオーバーだ。ハイスコアには、得点と所要時間に加えてプレイヤー名も登録できる（**画面3**）。

画面3
厳しい条件の場所に石を置いてハイスコアを目指せ。



画面1
四隅と中央に石が置かれた状態でゲームが始まる。



画面2
右ボタンクリックで石の置ける場所を確認（使い過ぎに注意）。

コンピュータプレイヤーを追加可能なボードゲーム

Chinese Checkers

バージョン：1.0　　ライセンス：GPL

http://jcatki.dhs.org/CC/

Chinese Checkersは、反射神経を必要としない、思考型のボードゲームだ。ルールはコマを1つずつ動かすか、ほかのコマを飛び越えるかして相手の陣地に進んでいくという、西洋のチェッカーズと同じものだ。自分で作製・改良した、コンピュータの思考ルーチンを追加可能であり、さまざまな特性を持ったコンピュータプレイヤー同士を戦わせることができる。

●ビルドとインストール

Chinese Checkersは、ファイル一式をtar＋gzipしたtarボールでのみ配布されている。ビルドの手順は多少複雑で、まずtarボールを展開してできたディレクトリ内にあるjax.tgzを、別のディレクトリで展開し、本体とは別にビルドする。以下のようにするといいだろう。

```
$ mkdir ../jax
$ cd ../jax
$ tar xzf ../CC/jax.tgz
$ make
$ cd ../CC
$ make
```

できあがった実行ファイル（CC）をパスの通ったディレクトリにコピーし、players/ディレクトリ以下の共有ライブラリを、適当なディレクトリにコピーし、環境変数でそのディレクトリを以下のように指定する。

```
$ mkdir /usr/local/games/CC
$ cp players/*.so /usr/local/games/CC
$ export CC_GAME_BASE_DIR=/usr/local/
games/CC
```

●意外に強いコンピュータプレイヤー

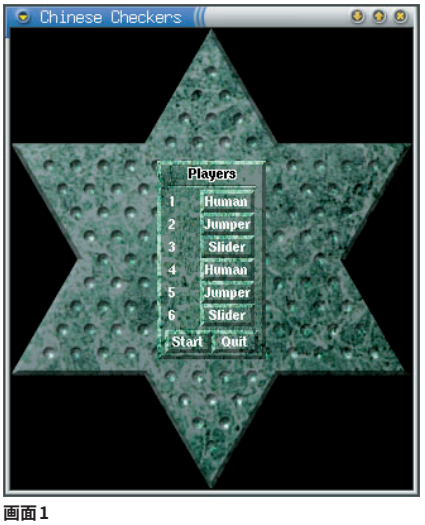
「CC」として起動すると、プレイヤーの選択画面が表示される（**画面1**）。最大6プレイヤーまで参加させられるが、あまり多人数だと盤面が込みすぎてしまうので、2〜4プレイヤー程度が適切だ。現在のバージョンでは「Human」を2人より多く設定すると、うまく動かないようなので、実際にはプレイヤー対コンピュータ（1〜3）で戦うことになる。

コマの動かし方は、1マスずつ進むか、自分の隣にいるほかのコマを飛び越えるかの2通りだ。飛び越えたマスの隣にまたほかのコマがあれば、もちろん連続でジャンプしていける。

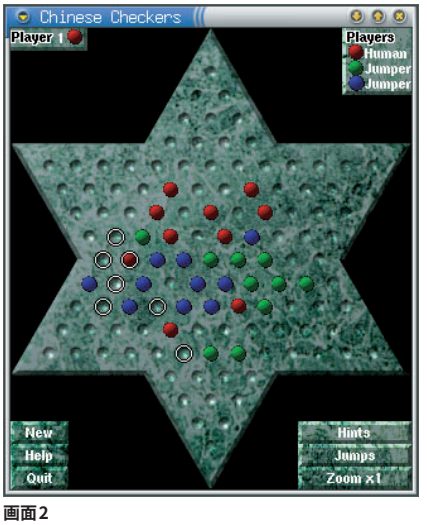
自分のコマをクリックすると、動かせる場所が表示される（**画面2**）。盤上のすべてのコマを利用して、相手より早く向かいの陣地にコマを集めよう（**画面3**）。

コンピュータプレイヤーは、1つずつコマを進めるのが好きな「slider」と、ジャンプするのが好きな「jumper」がいる。どちらも戦法はシンプルだが、うっかりしていると人間が負けたりするので、けっこう侮れない。

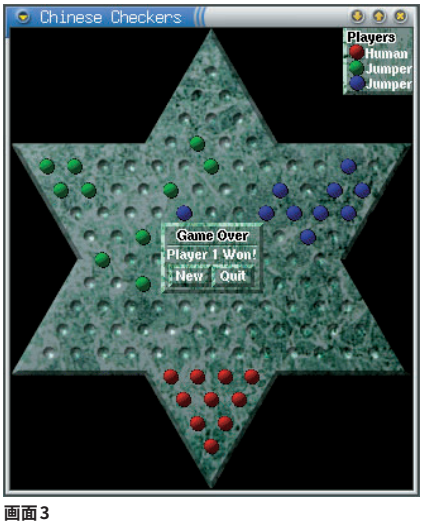
作者のWebサイトによれば、そのほかにも連続ジャンプを狙う「Robo」や、その改良版の「MM」などがいるようだ。sliderやjumperのソースを参照してオリジナルのプレイヤーを作成することも可能だ。



画面1
プレイヤー選択画面。敵は「Slider」「Jumper」の2種類。



画面2
ヒントを参考に、効率良くコマを進めよう。



画面3
コンピュータに負けないように！

Xで3D表示を行うには

X Window Systemで主に使われている3Dグラフィックスライブラリは、Silicon Graphics社のOpenGL、またはフリーの互換ライブラリである「Mesa 3D graphics library」（以下Mesa）である。Mesaを用いれば、XFree86でサポートされているビデオカードで3Dグラフィックスが実現できる。

普通にMesaをインストールした状態では、すべての描画作業はCPUが行っている。そこそこ強力なCPUを使っていれば、アクション性の低いゲームなら、この状態でも十分に3Dグラフィックスを楽しむことが可能だ。だが、フレームレート（画面の書き替え速度。1秒あたりの画面数で表わす）が重要な意味を持つアクションゲームを、快適にプレイすることは困難である。

そのようなゲームをプレイする場合は、3Dアクセラレータを利用する必要がある。

Mesaで使用可能な3Dアクセラレータ

XFree86 4.0の発表以前は、3DアクセラレータをMesaで利用する方法は、それぞれのプロジェクトにより異なっていた。

その中で最もサポートが進んでいたのは、米3dfx社のVoodooシリーズだ。それは3dfx社が同社のハードウェア専用の3DグラフィックスライブラリであるGlideのLinux版をリリースしているため、MesaのドライバはGlideライブラリへのラッパーとして、比較的容易に実現できるからだ。現状では、Linux上の3Dアクションゲームを楽しむためには、Voodooシリーズの3Dアクセラレータ、特に2D／3D統合型のVoodoo 3／Voodoo Bansheeが最も確実だ。

Windows用の3Dアクセラレータとしては一人勝ちを続けている米nVIDIA社も、同社のRIVA 128、RIVA TNT／TNT2、GeForce256の各シリーズに対応したドライバをWebサイトで提供している。現状ではそれほど性能が高いわけではないようだ。

ドライバのインストール

Mesaから Voodoo 3／ Voodoo Bansheeの機能を使うためには、専用のドライバ、Glide 3Dグラフィックスライブラリが必要だ。XFree86 3.3.5以降のXサーバを用いていれば、標準でVoodoo 3／Voodoo Bansheeに対応している。これらはすべて、http://linux.3dfx.com/から得ることができる。いずれもRPMパッケージ形式で提供されている（リスト1）。

最初にXサーバ、XF86Setupをインストールし、正しく動作できるようにする。その後、デバイスドライバ、Glideライブラリ／SDKをインストールしよう。

XFree86 4.0

先月号でも紹介したように、XFree86の最新バージョンである4.0がすでにリリースされている。4.0では、OpenGLの表示をX上で行う仕組みであるGLXと、Mesaの機能をXサーバに統合している。さらに、3Dアクセラレータの機能を直接利用するDirect Rendering Infrastructure（DRI）が付け加えられており、今までよりも多くのグラフィックスカードでハードウェアの機能を活かした3D描画が可能になっていくだろう。

これまでは個別の製品に対応して開発が行われてきたので、開発者の努力の結果を享受できるのは、その製品を持っているユーザーに限られていた。しかしDRIのような枠組みができたことで、この状況も改善されるだろう。現在の4.0で利用できる3Dアクセラレータは、やはり3dfx社のVoodoo3／Voodoo Bansheeだ。DRIを用いたドライバの開発は、米Precision Insight社で行われており、Matrox G400、ATI Rage 128、インテル i810など最新の製品が最初にサポートされるようだ。3Dアクセラレータは、世代間の性能差が大きいので、あまり古い製品への対応は行われないと考えられる。

上記のアクセラレータの中でもi810は、チップセットの一部（メモリコントローラハブ）とグラフィックスコアを統合した製品であり、ローコストPCの多くで採用されているため、快適な3Dグラフィック環境をLinuxで実現するためのコストが一気に下がることになる。「Linuxで3D」が一気に普及するのではないだろうか。

リスト1　Voodoo 3／ Voodoo Banshee用ドライバパッケージ一覧

Xサーバ	XFree86-SVGA-3.3.5-4.i386.rpm
XF86Setup	XFree86-XF86Setup-3.3.5-4.i386.rpm
デバイスドライバ	Device3Dfx-2.3-5.src.rpm
Glideライブラリ	Glide_V3-2.60-16.i386.rpm
Glide SDK	Glide_SDK-2.2-16.i386.rpm

ペンギンが雪山を滑り降りる3Dアクションゲーム

Tux Racer

バージョン：0.12　　ライセンス：GPL

http://tuxracer.sourceforge.net/

Tux Racerは、Linuxのマスコットであるペンギンの Tux君が、雪山を滑り降りる3Dアクションゲームだ。まだ未完成で、ラップタイムを保存したりすることはできないが、雪や氷の上を滑降し、勢いをつけてジャンプするときの迫力はなかなかのもの。変化に富んだコースが9種類用意されており、暗闇の中でレースを行うこともできる。動作にはOpenGL（あるいはMesa）とGLUT、Tclが必要だ。

●ビルドとインストール

Tux Racerは、tarボールでソース一式とデータがそれぞれ配布されている。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。

データのほうは、展開後のディレクトリで「mkdir /usr/local/share/tuxracer ; cp -a * /usr/local/share/tuxracer」としてコピーする。

●コースの先を読んで方向を決めろ

コマンドラインで「tuxracer&」とすると、コース選択画面が表示される（**画面1**）。現在のコースの全容が表示されており、n／pキーで順次、1〜9キーで直接の切り替えが可能。また、mキーで左右反転したミラーコースになる。なお、初めて起動する場合は、「tuxracer hoge&」のように適当な引数を指定しないと、起動に失敗してしまうようだ。

sキーを押すとプレイ開始だ。Tux君が登場して、雪山を滑降し始める（**画面2**）。滑降中は、j／lキーで左右に向きを変更できる。といっても、スピードがついている場合はなかなか向きが変わらないし、地面の状態（雪・氷・土など）にも影響される。

もし、木や山に衝突してスピードが落ちてしまったら、kキーでパドリングして勢いをつけよう。スペースキーでブレーキをかけることも可能だ。

コースの端まで到達するか、Escキーを押すとゲームオーバーだ。コースによっては、空高くジャンプしたり、剣山のような岩の間をすり抜けたり（**画面3**）、暗闇の中を滑り降りたり（**画面4**）と、さまざまなバリエーションを味わうことができる。

画面1
コース選択画面では、コース全体の構造が表示される。

画面2
プレイ開始だ。Tux君が腹ぱいになって滑降し始める。

画面3
コースの先を読みつつ、剣山のような岩の間をすり抜ける。

画面4
先の読めない暗闇の中でレースを行うことも可能だ。

空間把握能力が問われる3D落ちモノパズル

GNOME 3D Tetris

バージョン：1.1.0　　ライセンス：GPL

http://webdat.com/seb/3dtetris.html

GNOME 3D Tetrisは、さまざまな立体ブロックを操作して平面上に敷き詰める3D落ちモノパズルゲームだ。回転軸が3つもあるため、複雑な形のブロックを思い通りに回転させるには優れた空間把握能力が要求される。ブロックはワイヤフレームで表示されるため、3Dアクセラレータは必要ない。オプション設定により、キー設定や盤面サイズ、難易度などを変更可能だ。動作にはGNOME／GTK＋が必要となる。

●ビルドとインストール

GNOME 3D Tetrisは、tarボールとRPMバイナリパッケージの両方で配布されている。バイナリパッケージはRed Hat 6.x系用だ。

tarボールからのビルドとインストールも、「./configure」→「make」→「make install」という一般的な手順なので難しくはない。

●立体のブロックを回転させろ

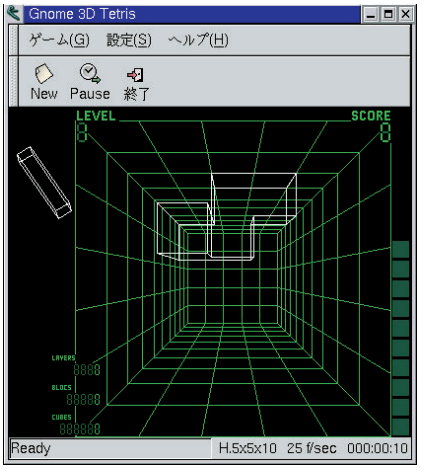
コマンドラインで「gno3dtet&」とするか、GNOMEメニューの[ゲーム]－[3D Tetris]を選択すると、盤面を含むウィンドウが開く。ツールバーの[New]ボタンでプレイ開始だ。

画面手前にワイヤフレームの立体ブロックが登場し、ゆっくりと画面奥に向かって下降する（**画面1**）。ブロックの移動はカーソルキー、各軸ごとの回転はそれぞれq／a、w／s、e／dキーを使用する。思い通りの位置と向きになったら、スペースキーでブロックを一気に底まで落下させよう。

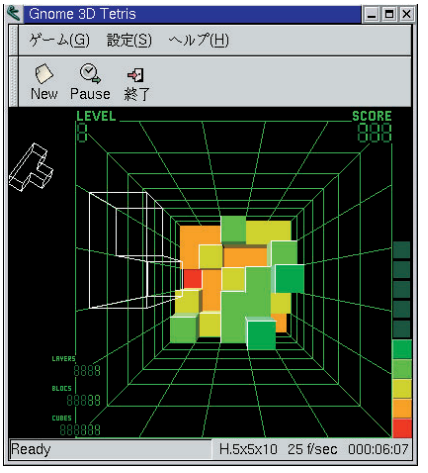
操作に慣れていない最初のうちは、6つのキーを使おうとせずに、各軸ごとに回す方向を固定しておくといいだろう。

積まれたブロックは、高さに応じた色（右側に表示）で塗りつぶされ（**画面2**）、ブロックで隙間なく平面を埋めると、その平面のブロックが消える。ブロックが盤面の上まで達するとゲームオーバーだ。ハイスコアを出せば名前を登録できる（**画面3**）。

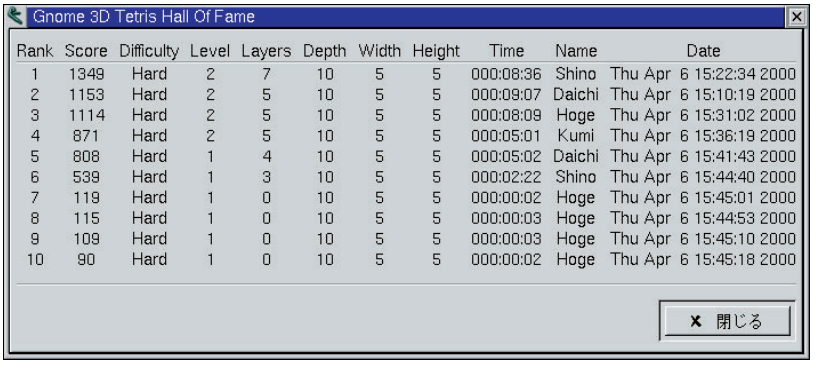
なお、初期設定では難易度が「Hard」になっている。プレイしてみて難しすぎるようなら難易度を変更しよう。[設定]－[設定]でダイアログを開き、[Game difficulty]の設定を[Easy]や[Medium]にすればいい（**画面4**）。このほか、盤面のサイズやキー操作を変更することもできる。



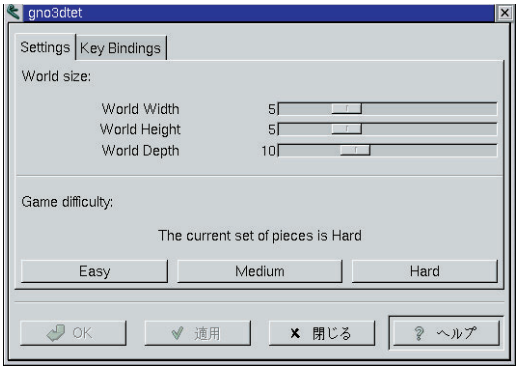
画面1
立体のブロックを思い通りに回転させるのは難しい。



画面2
積まれたブロックは、高さに応じて色分け表示される。



画面3
ハイスコアにはスコアや難易度、プレイヤー名などが表示される。



画面4
ゲームが難しすぎる場合は、難易度を[Easy]か[Medium]にしよう。

ライトサイクルで爆走する3Dアクションゲーム

glTron

バージョン：0.59　　ライセンス：GPL

http://www.gltron.org/

glTronは、世界初のCG映画「TRON」に登場するライトサイクルゲームをX上で実現した3Dアクションゲームだ。四角いアリーナの中で、未来のバイク「ライトサイクル」を互いの航跡に接触しないような走らせるサイバーなゲームだ。なお、3D表示にはOpenGL（またはMesa）を使用するが、オプション設定でテクスチャなどの画面効果をすべてオフにすれば、3Dアクセラレータなしでもプレイ可能だ。

●ビルドとインストール

glTronは、tarボールでのみ配布されている。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。動作には3DライブラリのMesaと、サウンドライブラリのMikModが必要だ。

なお、EsounDサウンドドライバ（esd）を利用している環境では、起動前に「esdctl off」として、機能を一時停止しておく必要がある。

●ライトサイクルを走らせろ

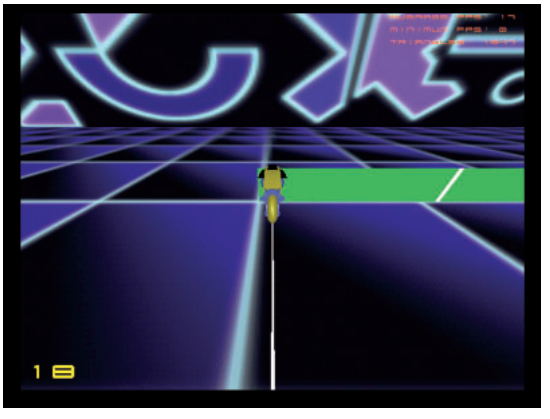
「gltron」として起動すると、タイトル画面が表示される（**画面1**）。「GAME」→「START GAME」と選択するとプレイ画面に切り替わる。最初にF10キーを1回だけ押して、カメラモードを「ライトサイクルの後方」に変更するとプレイしやすいだろう。スペースキーを押すとプレイ開始だ。

ライトサイクルは、aキーで左、sキーで右に方向を変える。アリーナの壁や、ライトサイクルの後に作られる光の壁に衝突しないように走り続けよう（**画面2**）。衝突するとライトサイクルは破壊され、最後まで残ったプレイヤーが勝者となる。プレイヤーは同時に4名まで参加でき、通常はプレイヤー1を人間が受け持つ。

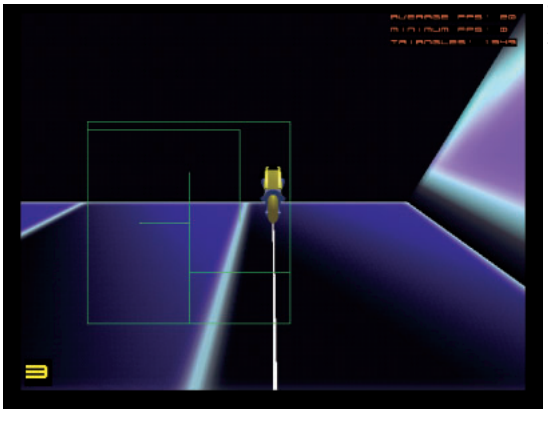
このほか、テクスチャなどの画面効果の設定をはじめ、BGMや効果音、操作に使用するキーや、プレイ中の2Dマップの表示（**画面3**）など、各種の設定を変更可能だ。

なお、3Dアクセラレータのない環境でプレイする場合、「gltron -gwtbmx1」として起動すると、あらかじめテクスチャなどの画面効果をオフにした状態になる（**画面4**）。

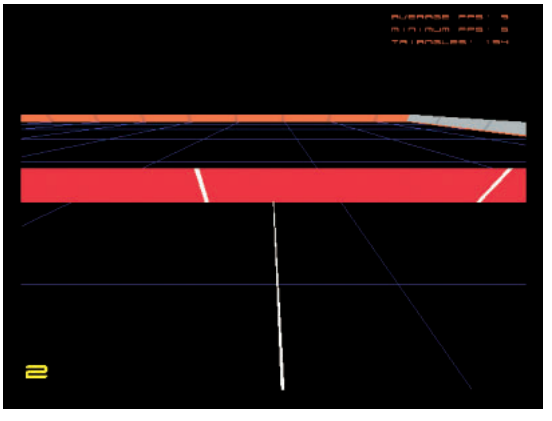
画面1
タイトル画面。画面表示やサウンドの設定変更も可能だ。



画面2
ライトサイクルを壁にぶつからないように走らせる。



画面3
2Dマップ表示をオンにすると、ぐっと走りやすくなる。



画面4
この画面でよければ3Dアクセラレータがなくてもプレイ可能。

ネットワーク3Dタンクバトルゲーム

BZFlag

バージョン：1.7d　　ライセンス：GPL

http://bzflag.sourceforge.net/

BZFlagは、障害物の置かれたフィールド内で戦闘を繰り広げるマルチプレイヤー3Dタンクバトルゲームだ。ローカルなネットワーク上でサーバを起動して対戦するほか、インターネット上のサーバに接続してプレイすることも可能だ。実行にはMesaが必要だが、3Dアクセラレータなしでも十分プレイを楽しめる。Linux版以外にWindows版なども配布されており、異なるOSのマシン同士で対戦することも可能だ。

●ビルドとインストール

BZFlagはサーバソフトとクライアントソフトがまとめて配布されており、tarボールおよびRPMバイナリパッケージ（Red Hat 6.x系用）が用意されている。このほか、実行には3DライブラリのMesaが必要だ。

ソースからビルドするには、展開先のディレクトリで「make linux-i386」としてプラットホームを指定し、「make all」でビルドする。RPMが使える環境では、「make package」でdistディレクトリにRPMバイナリパッケージが作成されるので、これをインストールすればいい。

それ以外の環境では、展開されたbin/bzflag（クライアント）とbin/bzfs（サーバ）に対するシンボリックリンクを、/usr/local/binなどに作っておこう。展開先のディレクトリから、以下のようにする。

```
$ ln -s bin/bzflag /usr/local/bin/bzflag
$ ln -s bin/bzfs /usr/local/bin/bzfs
```

●クライアントの画質を設定

コマンドラインで「bzflag」とすると、タイトルとメニューが表示される（**画面1**）。メニューの移動は↑↓キー、選択はEnterキー、キャンセルはEscキーで行う。

まずは、[Options]を選択してオプション設定を行う。ここでは、画面表示やサウンド、キー割り当てに関する設定を変更できる（**画面2**）。

最初の8項目が画面表示に関する設定で、初期値は「On／On／Off／Off／Off／Low／Off／On」とかなり抑え目になっている（**画面3**）。CPUがPentium II／IIIやCeleronであれば、3Dアクセラレータなしでも「On／On／On／On／Linear／Low／On／On」程度は軽くこなせるはずだ（**画面4**）。なお、画面の解像度は[Change Video Format]で変更できる。

さらに、Mesaが対応している3Dアクセラレータカード（Voodoo系など）があれば、「On／On／On／On／Linear Mipmap Linear／ High／On／On」と最高の画質に設定した画面（**画面5**）でも、快適な速度でプレイ可能だ。

●サーバの起動

ローカルなネットワーク内で対戦する場合、マシンのどれかひとつでサーバソフト（bzfs）を起動する必要がある。[Join Game]を選択して画面を切り替え、[Start Server]を選択しよう。ここでは、ゲームの種類などの条件を設定する（**画面6**）。

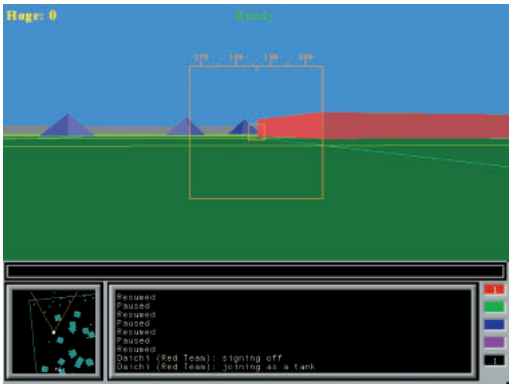
参加人数が少ない場合は、自分以外は全て敵となる「Free for all」、ある程度人数がいる場合は、チームに分かれて相手チームのフラッグを取り合う「Capture the Flag」を選択するとよいだろう。

このほか、接続可能なプレイヤー数や連続発射が可能な弾数を制限したり、別の場所に瞬時に移動するゲート「テレポータ」やTabキーでの「ジャンプ」、さまざまな効果をもたらす「スペシャルフラッグ」などの有無もこの画面で設定できる。

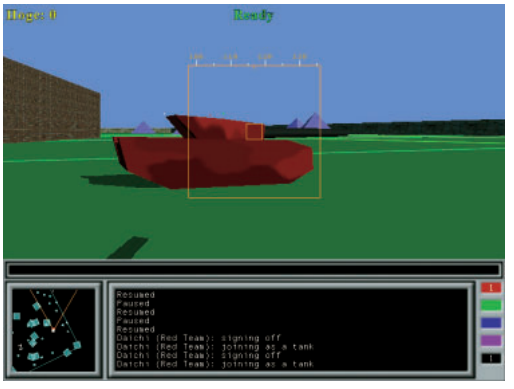
最後に、最下行の[Start]を選択する

画面1
初期設定ではウィンドウに4つのビューが表示される。

画面2
オプション設定で、プレイしやすい範囲で綺麗な画面を探す。



画面3
初期設定による画面は、ソフトウェアによる描画でもスムーズ。



画面4
テクスチャや影などを有効にする。CPUが速ければまだ余裕。

とサーバソフトが起動する。Escキーでこの画面から抜け、[Server]にサーバを起動したマシンのホスト名、[Callsign]に自分のコールサインを設定して、[Connect]がお勧めだ。

●ゲーム中の操作

マウスの右ボタンをクリックするとあなたのタンクがフィールドに現れる。画面にはフィールド内の様子が3D表示され、左下にはレーダー、その右にはシステムメッセージや会話のログ、右下には各チーム（色別）のプレイヤー数が表示される。

タンクの操作には主にマウスを使用する。カーソルを画面の上下に動かすと前進・後退、左右に動かすと旋回だ。移動速度はカーソルが画面中央からどれだけ離れているかに比例する。停止するには中央部の小さな矩形にカーソルを置けばいい。

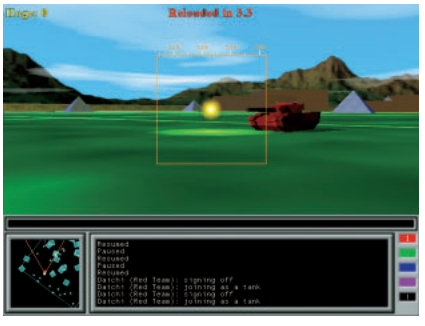
まずはレーダーを使って敵を見つけよう。レーダーのレンジは1～3キーで変更可能だ。敵を視界に捕らえたら、画面中央に来るように移動しつつ左ボタンで弾を発射しよう（**画面7**）。

なお、連射できる弾数（初期設定では1発）を超えると、再充填には3秒を要する。このほか、プレイヤー全員やチームの仲間に対してメッセージを送ることも可能だ。

●ソロプレイとインターネット対戦

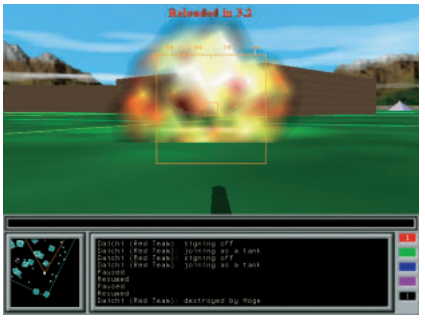
近くにBZFlagを遊んでくれる仲間がいない場合、2つの解決方法がある。ひとつは、自動操縦のロボット戦車を導入することだ。起動時に「-solo 台数」とオプションを付けると、指定した台数のロボット戦車がゲームに搭乗する。

もうひとつの方法は、インターネット上で運用されているサーバに接続することだ。Join Game画面で、[Find Server]を選択してしばらく待つと、誰でも接続できるサーバの一覧が表示される（**画面8**）。現在のプレイヤー数なども表示されるので、ゲームが行われているサーバに接続しよう。



画面5
品質を最高にする。さすがに3Dカードなしではゲームにならない。

画面6
サーバを起動する。ゲーム全体の各種条件はここで設定。



画面7
見事に敵の戦車を撃破。相手プレイヤーの顔が目に浮かぶ。

画面8
インターネットのサーバに接続してプレイすることも可能。

リアルな挙動の3Dビリヤードゲーム

Gtulpas

バージョン：1.0.0　　ライセンス：GPL

http://www.suse.cz/gtulpas/
http://www.mesa3d.org/ （Mesa）
http://www.student.oulu.fi/~jlof/gtkglarea/ （GtkGLArea）
http://www.gnu.org/software/guile/ （guile）

Gtulpasは、9ボール、スヌーカー、キャロムをプレイできる3Dビリヤードゲームだ。大きなテクスチャデータを用いていないので、3Dアクセラレータなしでも十分に楽しむことができる。優秀な物理エンジンを搭載しており、ボールに捻りや押し引きを加えた際の挙動はなかなかリアルだ。ウィンドウを分割して複数の視点からテーブルを眺めたり、自分でゲームのルールを追加してプレイすることもできる。

●ビルドとインストール

Gtulpasは、ファイル一式をtar＋gzipしたtarボールで配布されている。ビルドの際は、「./configure --prefix=/usr」→「make」→「make install」としよう。/usr以下にインストールしないと、起動時に初期化ファイルなどの読み込みに失敗して異常終了するので注意されたい。

このほか、テンキーのないキーボードで視線を切り替えられない、ヘルプブラウザでヘルプが表示されないという不具合がある。これを解決するパッチ（gtulpas-lm.patch）を用意したので、ビルド前にGtulpasの展開先で「patch -p1 < gtulpas-lm.patch」として修正されたい。

動作にはGNOME、Mesa、GtkGLArea、guileが必要だ。これらのライブラリは、ビルド前にあらかじめインストールしておく必要がある。特に、3DライブラリのMesaは、最新の3.1（beta3以上）が必要なので、3.0や3.1beta1などを使っている場合は入れ換えよう。以下では、Mesa 3.1、guile 1.3.4、GtkGLArea 1.2.2を使用している。

●画面表示とゲームの切り替え

起動すると、スプラッシュ画面が表示された後でウィンドウが開く。なお、端末画面にボールの座標などが大量に表示されるので、「gtulpas > /dev/null&」と/dev/nullにリダイレクトして起動するとよい。

ウィンドウには、中央のメインビューのほか、左側に3つのミニビューが用意され、さまざまな角度からテーブルを眺められる（**画面1**）。ミニビューを消したり、ウィンドウのサイズを変更することも可能だ。

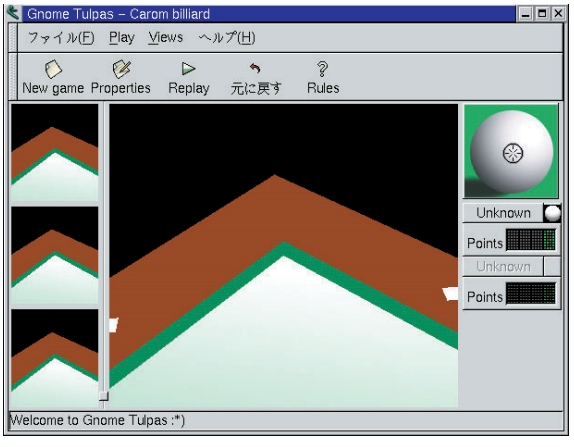
用意されているゲームは、ポケット（穴）のないテーブルで行う三つ球のキャロム（**画面2**）、ポケット台で行うおなじみの9ボール（**画面3**）、少し大きめのポケット台と小さな球で行うスヌーカー（**画面4**）の3種類だ。ゲームの切り替えは、[File]－[Select game]で行う。

ゲームが始まったら、[ファイル]－[Players]でプレイヤー名を登録しよう。現時点では、人間同士の対戦のみ可能だ。プレイヤー名や得点、現在の順番といった情報は、メインビューの右側に表示される。

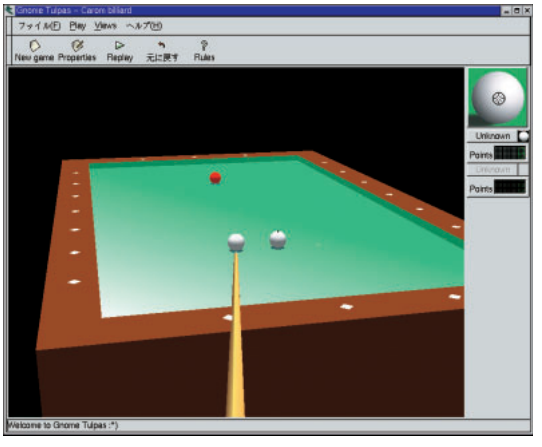
●英字キー＋左ドラッグで操作

ゲーム中の操作はマウスとキーで行う（**リスト1**）。基本的な操作は、「英字キーを押しながらマウスを左ドラッグする」というもの。たとえば、Cキーを押しながらマウスを上下にドラッグすると、キューのストローク（手球を撞く動作）になる。

プレイヤーの視線はマウスの左・中ドラッグで変更できるが、これだけでは思い通りのアングルにすることが難しいため、0～9キーで代表的な視線に切り替えるといい。お勧めは、キューの向きに視線を連動させる0キーだ。左ドラッグだけでキューの向きを変えられ、手球が常にビューの中心に位置するので狙いをつけやすい。Ctrlキー



画面1
初期設定ではウィンドウに4つのビューが表示される。



画面2
三つ球のキャロム。手球以外の2つのボールに当てれば得点だ。

による微調整も可能だ。

右上に表示された手球の撞点を変更すると、手球を撞く際に捻りや押し引きのアクションを加えられる。キューの向きと撞点が決まったら、C＋左ドラッグ（または右ドラッグ）で手球を撞こう。直前のショットを取り消したり、リプレイすることも可能だ。

●各ゲームのルール

(1) 9ボール

9番のボールをポケットしたプレイヤーが勝ちとなるルールは御存じの方が多いと思われるので簡潔に述べる。手球は、テーブル上の最小番号の的球に当てる必要がある。テーブル上のボールには番号が付いていないが、プレイヤー名の右に表示される色の球を狙えばいい。

的球を1つ以上ポケットすると続けてプレイでき、それ以外の場合は相手の順番となる。スクラッチ（手球がポケットすること）などのファウルを犯すと、相手プレイヤーは手球フリーで、M＋左ドラッグにより手球の位置を自由に変更できる。

(2) キャロム

日本ではキャロムといえば四つ球だが、Gtulpasに用意されているのは三つ球のキャロムだ。手球を2つの的球に当てると1点加算されて続けてプレイでき、それ以外の場合は相手の順番となる。先に、[ファイル]－[Game options]で設定した点数（初期値は20点）に達したほうが勝ちだ。

なお、Gtulpasでは、各ゲームのルールや、テーブルの大きさなどの設定を、/usr/share/gtulpas/rulesおよびschemeディレクトリに置かれたguileスクリプトから読み込む。ルールや設定を追加して、新しいゲームを組み込むことも可能なので、三つ球のルールを修正すれば、四つ球のルールを作成できるはずだ。

(3) スヌーカー

イギリスで盛んなスヌーカーは、15個の赤球と、黄・緑・茶・青・ピンク・黒のカラーボールを使用するちょっと複雑なゲームだ。ここでは概略だけを述べる。

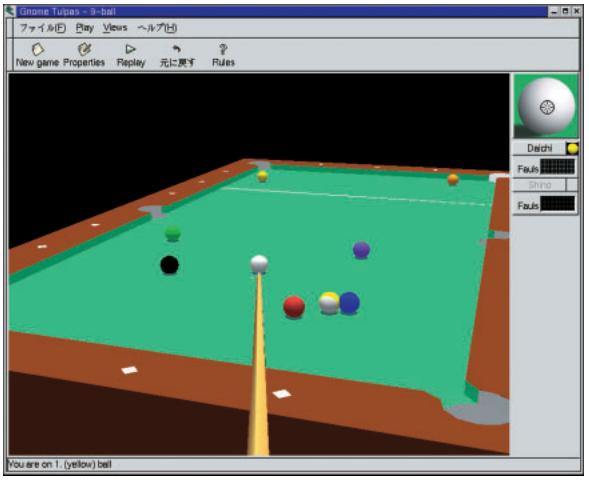
プレイの基本は、「赤球をポット（ポケットのこと）するとカラーボールをポットする権利を得る」というもの。つまり、赤球とカラーボールを順番にポットしていく。赤球が残っている場合、ポットしたカラーボールはテーブルの定位置に戻される。

赤球をポットすると1点、カラーボールは上記の黄～黒の順にそれぞれ2～7点が加算される。なお、どのカラーボールを狙うかは自由だが、狙うカラーボールをショットする前にクリックで指定する必要がある。

テーブル上に赤球がなくなると、点数の低い順にカラーボールをポットする。この場合、ポットしたカラーボールはテーブルに戻さない。最終的な得点が高いプレイヤーが勝ちだ。このほか、相手のファウルによっても点数が加算される。スヌーカーについての詳細は「First Snooker Experience」（http://www.st.rim.or.jp/~tomio/Snooker/）などを参照されたい。

C＋左ドラッグ	キューをストローク
右ドラッグ	（同上）
V＋左ドラッグ	キューを回転
M＋左ドラッグ	ボールを移動（フリーボール時）
0	キューの向きに連動した視線に切り替え
1～9	テーブルを八方から見た視線に切り替え
Z＋左ドラッグ	ズーム
＋／－	ズームイン/アウト
P＋左ドラッグ	平行移動
中ドラッグ	（同上）
R＋左ドラッグ	回転
左ドラッグ	（同上）
S＋左ドラッグ	自動回転
Ctrl－S	自動回転の停止
H	ヒント表示（トグル）

リスト1　マウス、キー操作一覧



画面3
おなじみ9ボール。ファウル後は手球がフリーボールになる。

画面4
赤球とカラーボールを交互にポットするスヌーカー。

美しい3D画面表示の俯瞰型3Dシューティング

GLAGLAXIAN

バージョン：0.2　　ライセンス：GPL

http://www.multimania.com/hosxe/glaglaxian.html

GLAGLAXIANは、美しい3D画面表示が特長の俯瞰型シューティングゲームだ。まだ未完成で巨大ボスやボーナスステージは登場しないものの、通常弾とボムの使い分けや、特徴ある敵キャラ、クールなBGMなど、十分楽しめるレベルに達している。キー操作により、テクスチャやフォグの表示をオフにすることもできるが、実用的な速度でプレイするには、やはり3Dアクセラレータが必須だ。

●ビルドとインストール

GLAGLAXIANは、tarボールでのみ配布されている。動作には3Dライブラリの Mesaが必要だ。また、BGMや効果音を楽しむには、サウンドライブラリのMikModも必要になる。

そのまま「make」とすると、サウンド機能付きでビルドされる。一方、サウンド機能を利用しない場合は、「make SOUND=no」とする。インストールは手動で行う。プログラムのglaglaxianとpictures、samplesディレクトリを、適当なディレクトリ（/usr/local/gamesなど）にコピーすればいい。

なお、EsounDサウンドドライバ（esd）を利用している環境では、起動前に「esdctl off」として、機能を一時停止しておく必要がある。

●通常弾とボムを使い分けよう

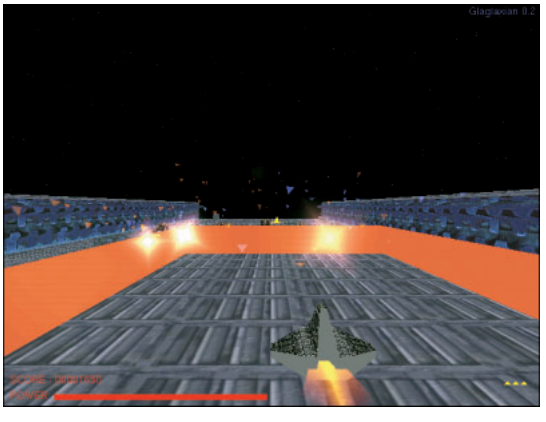
インストール先のディレクトリに移動して「./glaglaxian&」とすると、タイトル画面が表示される（**画面1**）。別のディレクトリから起動すると、データの読み込みに失敗するので注意されたい。yキーでプレイ開始だ。

画面上方から敵が襲来するので、カーソルキーで自機を前後左右に移動させ、wキーで通常弾を発射しよう（**画面2**）。通常弾は連射可能だ。敵の数が多いときには、xキーのボムで敵を一掃する（**画面3**）。ボムは数に限りがあるので乱用は控えよう（画面右下の▲が残りボム数を示す）。

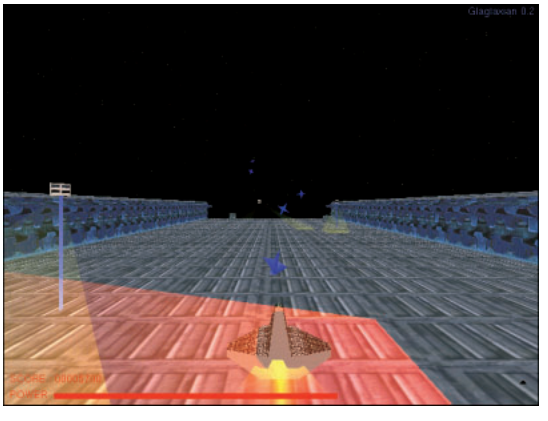
面が先に進むにつれ、光の矢を放つスライム状の敵や、こちらをサーチライトに捉えるとミサイルを連射する敵など、バリエーションに富んだ敵が登場する（**画面4**）。画面下のパワーグラフが0になるとゲームオーバーだ。

画面1
タイトル画面。yキーを押すとプレイ開始だ。

画面2
カーソルキーで自機を操作し、上方から飛来する敵を破壊せよ。



画面3
敵を一掃してくれるボム。数に限りがあるので大事に使おう。



画面4
サーチライトに捕らえられると、ミサイルが雨あられのように…。

Webサイト一覧表

今回紹介したゲームのWebサイトをまとめておく。本誌で説明したこと以外にも、有用な情報が見つかるかもしれない。また、頻繁にアップデートされる作品もあり、本誌が発売するころには、より新しいバージョンが出ているものもあるだろう。最新の情報はこちらから得てほしい。

XShipWars	http://fox.mit.edu/xsw/
Japanese NetHack	http://www.jnethack.org/jnethack.html
PySol	http://wildsau.idv.uni-linz.ac.at/mfx/pysol.html
Batalla Naval	http://batnav.sourceforge.net/
RealTimeBattle	http://realtimebattle.sourceforge.net/ http://michiko.shiratori.riec.tohoku.ac.jp/rtb/jp/Main.html
Circus Linux	http://www.newbreedsoftware.com/circus-linux/
XRally	http://www.linuxgames.com/xrally/
xsoldier	http://www.surfline.ne.jp/hachi/xsoldier.html
Roll'm Up	http://www.medialab.lostboys.nl/
KPooka	http://www.ucs.co.za/~pgr/kpooka/
XROT	http://www.ci.cs.meiji.ac.jp:8150/siraisi/xrot.html
Kishido	http://www.Informatik.Uni-Oldenburg.DE/~km/kishido/
Chinese Checkers	http://jcatki.dhs.org/CC/
Tux Racer	http://tuxracer.sourceforge.net/
GNOME 3D Tetris	http://webdat.com/seb/3dtetris.html
glTron	http://www.gltron.org/
BZFlag	http://bzflag.sourceforge.net/
Gtulpas	http://www.suse.cz/gtulpas/
GLAGLAXIAN	http://www.multimania.com/hosxe/glaglaxian.html

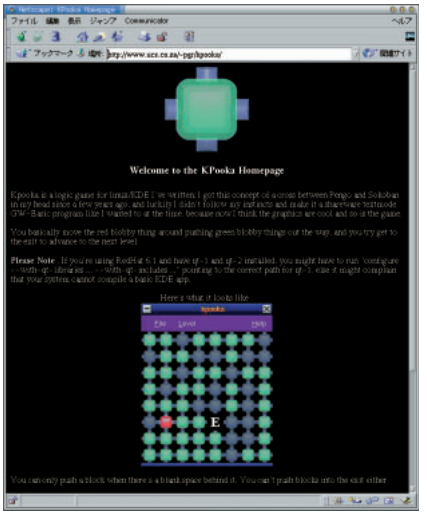
リスト1 掲載したゲームのWebサイト

画面1　XShipWars

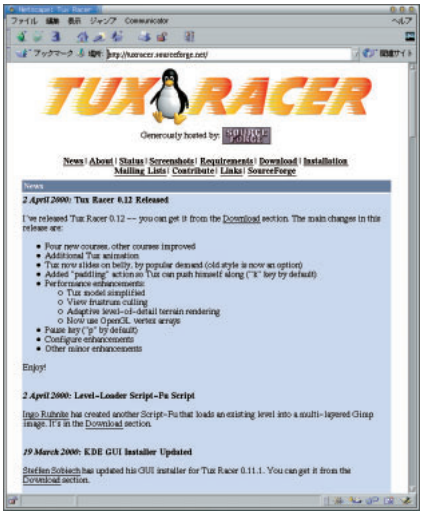
画面2　BZFlag

画面3　XRally

画面　4glTron



画面5　KPooka



画面6　Tux Racer

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1回＞

文：インプライズ株式会社　加藤大受
Text：Daiju Kato

昨年12月に米国でリリースされたJBuilder Foundationは、Linux上でのJavaプログラミングをビジュアルに実現する開発ツールとして話題になりました。ようやく日本語版が登場することになります。そこで、Linuxでのアプリケーション開発を促進するJBuilder 3.5 Foundationのインストールと使い方についてご紹介します。

JBuilder 3.5 Foundationについて

JBuilder 3.5 Foundationはインプライズが開発・販売しているJava開発ツールであるJBuilderのエントリー版です。前バージョンのJBuilder 3まではStandard版がエントリー版でしたが、昨年12月に米国インプライズがJBuilder Foundationを同社のWebサイトにて無償提供しました。JBuilder Foundationは10日間で10万ダウンロードを超えるほどの人気があり、特にLinux版はKDE、GNOME上で動作する最初のJavaビジュアル開発ツールとして数多くの開発者に支持されることになりました。今年3月、JBuilder FoundationもJBuilder 3.5 Foundationとしてバージョンアップしました。そして、今年4月6日、JBuilder 3.5日本語版がインプライズより発表され、待望のJBuilder 3.5 Foundationの日本語版が登場いたしました。そこで、いち早く読者のみなさんにお届けするとともに、インストールから簡単な使い方までを数回に分けて、詳細に説明していきます。

なお、本記事ではβ版を使用しているため、一部製品とは異なるところが存在するかもしれませんが、その点についてはご了承ください。

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1　JBuilder 3.5 Foundationの動作環境

マシン	IBM PC-350(6587-JBV)
CPU	AMD K6-2 400MHz
メモリ	160Mバイト
ハードディスク	6Gバイト
OS	LASER 5 Linux 6.0 Rel.2

表2　筆者の使用環境

動作環境

JBuilder 3.5 Foundation（以下、JBuilder 3.5）のLinux版の動作環境は**表1**の通りです。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はLASER 5 Linux 6.0とRed Hat Linux 6.1日本語版の2種類だけですが、JDKがサポートする環境であるカーネル2.2.5以降、glibc 2.1以降に該当するTurboLinux 6.0 WorkstationやKondara/MNU Linuxなどでも動作するはずです。参考までに筆者の使用している環境を**表2**に示します。

JDKのセットアップ

JBuilder 3.5のLinux版を使用するには、Sun MicrosystemsのJava 2 SDK Standard Edition version 1.2.2

Linux版（JDK）が必要となります。これは、本誌付属のCD-ROMに収録されています。JDKlicense.txtを参照してください。同社のWebサイトからダウンロードすることもできます。

まず、JDKをインストールします。

もし、rootでログインしていないときはsuコマンドを使用してスーパーユーザーとしてログインしてください。

```
$ su
```

続いて、本誌付録のCD-ROMをドライブにセットし、マウントします。

```
# mount /mnt/cdrom
```

次に、JDKの収録されているディレクトリに移動します。

```
# cd /mnt/cdrom/Linuxmag/JDK
```

次のように/usr/local以下にJDKをインストールします。

```
# tar zxvf jdk1_2_2-linux-i386.tar.gz
-C /usr/local
```

続いて、JBuilder 3.5のインストールを行うため、JDKにパスを設定します。

Java 2 JDKの入手先	http://java.sun.com/products/jdk/1.2/ja/download-linux.html
Jbuilder 3.5インストールキーの入手先	http://www.inprise.co.jp/jbuilder/foundation/download/

表3　URL一覧

```
# java -version
java version "1.2.2"
Classic VM (build 1.2.2-L, green threads, nojit)
```

画面1　インストールの確認

```
# export PATH=/usr/local/jdk1.2.2/
bin:$PATH
```

パスを設定したら正しくインストールされているかどうかを確認してみましょう。**画面1**のようにバージョンが表示されれば正しく設定されています。

ライセンスキーの入手

JBuilder 3.5を使用するためには、ライセンスキーが必要となります。このライセンスキーはインプライズのWebサイトで登録することで入手できます。Webサイトにて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のインストール

準備が整いましたので、JBuilder 3.5をインストールします。もし、X Window Systemを起動していないのであれば、Xを起動してください。

まず、JBuilder 3.5のインストールディレクトリに移動します。

```
# cd /mnt/cdrom/Linuxmag/JBuilder/
sol_linux/foundation/no_vm
```

続いて製品のインストーラを起動します。

```
# sh install.bin
```

JBuilder 3.5のインストーラはすべてJavaで書かれているので、まず使用するJava VMを選択します（**画面2**）。さきほどインストールしたJDKにパスを設定しましたので、こちらのVMを

```
# sh install.bin
InstallAnywhere is preparing to install...

Please choose a Java virtual machine to run
this program.
(These virtual machines were found on your PATH)
----------------------------------------
 1.     /usr/local/jdk1.2.2/bin/java
 2.     Exit.

Please enter your selection (number):1
```

画面2　Java VMの選択

画面3　言語の選択画面

画面4　インストーラの開始

選択します。

Java VMを選択すると、GUIのインストーラが起動し、**画面3**のような画面が表示されます。こちらではインストーラが使用する言語を選択することができます。すでに日本語が選択されているので、このまま［OK］ボタンを押します。

画面4のようなインストールプログラムが開始されます。［次へ］ボタンを押すと、**画面5**のような製品のライセンス情報が表示されます。

ライセンス情報の画面には製品を使用する際のライセンスについて記載されています。ライセンス契約に同意する場合は、ボタンをクリックし、［次へ］ボタンを押します。

画面6のようなインストールディレクトリを指定する画面が表示されます。デフォルトでは、/usr/local/jbuilder35にインストールされます。インストール先を変更したいときは、［選択...］ボタンを押して希望するディレクトリを選択してください。

［インストール］ボタンを押すと、製品のインストールが開始されます。製品のインストールが終了すると**画面7**のようにインストール終了を告げる画面が表示されます。

ドキュメントとサンプルプログラムのインストール

先ほど起動したインストールプログラムではプログラムしかインストールされません。ドキュメントおよびサンプルプログラムはそれぞれ別のインストールプログラムで提供されています。プログラムのインストールと同じように、インストールプログラムを起動してインストールを行います。このとき、プログラムをインストールしたディレクトリと同じディレクトリにドキュメント、サンプルプログラムともインストールしてください。

まず、ドキュメントのインストールプログラムのディレクトリに移動します。

```
# cd /mnt/cdrom/Linuxmag/JBuilder/
sol_linux/docs
```

次に、ドキュメントのインストールプログラムを起動します。

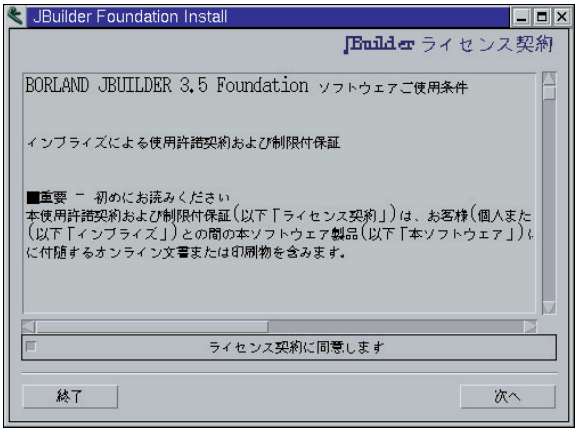
```
# sh install.bin
```

続いて、サンプルプログラムをインストールします。まず、サンプルプログラムのインストールプログラムのディレクトリに移動します。

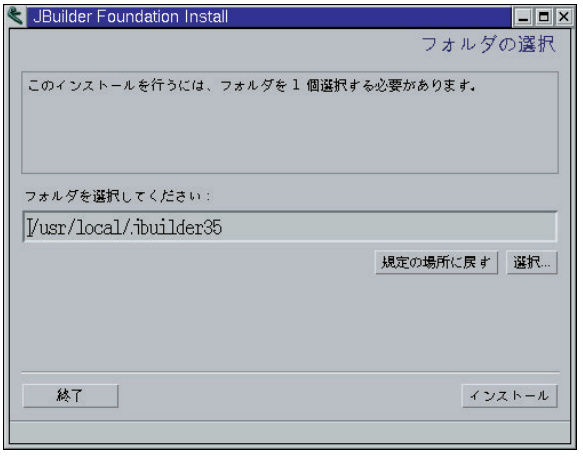
```
# cd /mnt/cdrom/Linuxmag/JBuilder/
sol_linux/samples
```

次に、サンプルプログラムのインストールプログラムを起動します。

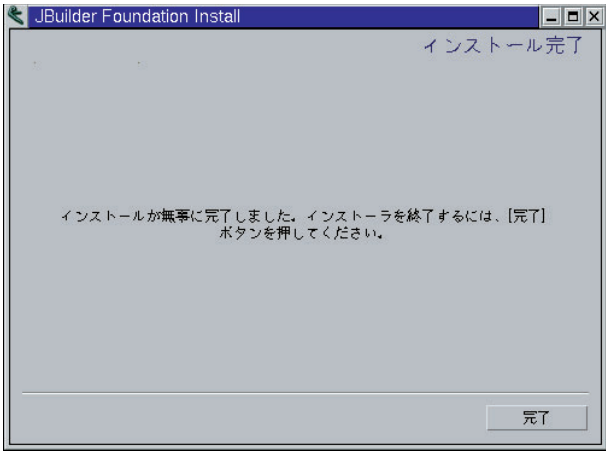
```
# sh install.bin
```



画面5　ライセンス情報



画面6　インストール先の指定



画面7　インストール終了

JBuilder JIT for Linuxのインストール

続いて、JBuilder JIT for Linuxのインストールを行います。JITコンパイラはJavaアプリケーションのパフォーマンスを向上させるツールです。

まず、JBuilder JIT for Linuxのディレクトリに移動します。

```
# cd /mnt/cdrom/Linuxmag/JBuilder/
sol_linux/LinuxJIT/
```

次に、/tmp以下にファイルを解凍します。

```
# tar zxvf javacomp-1.2.15.tar.gz
-C /tmp
```

解凍したファイルを/usr/local/jdk1.2.2/jre/lib/i386にコピーします。

```
# cp /tmp/javacomp-1.2.15/*.* /usr/
local/jdk1.2.2/jre/lib/i386
```

最後にJITの環境設定を行います。

```
# export JAVA_COMPILER=javacomp
```

ランチャのセットアップ

KDEを使用している場合は、デスクトップ上にJBuilder 3.5を起動するためのアイコンが作成されますが、GNOMEの場合は自分でランチャのセットアップが必要です。足跡のマークのアイコンがあるGNOMEパネルで右クリックして、［新規ランチャの追加］を選択します（**画面8**）。**画面9**のようなランチャアプレットの設定画面が表示されるので、次のように入力します。

名前:JBuilder 3.5 Foundation
命令:/usr/local/jbuilder35/bin/jbuilder

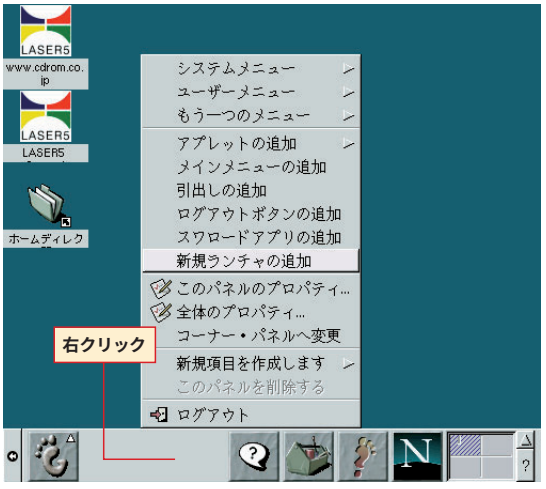
画面9　新規ランチャの設定

続いて、アイコンを設定します。アイコンのボタンを押すと、**画面10**のようなアイコンの選択画面が表示されますので、パスを/usr/local/jbuilder35に変更し、JBuilderのアイコンを選択します。ランチャアプレットの設定を行うと、GNOMEパネルにJBuilderのアイコンが追加されます（**画面11**）。

JBuilder 3.5の起動

GNOMEパネルからJBuilderのアイコンをクリックすると、初回だけ**画面12**のようなライセンスキーの入力画面が表示されます。ここで、［追加］ボタンを押して、入手したJBuilder 3.5

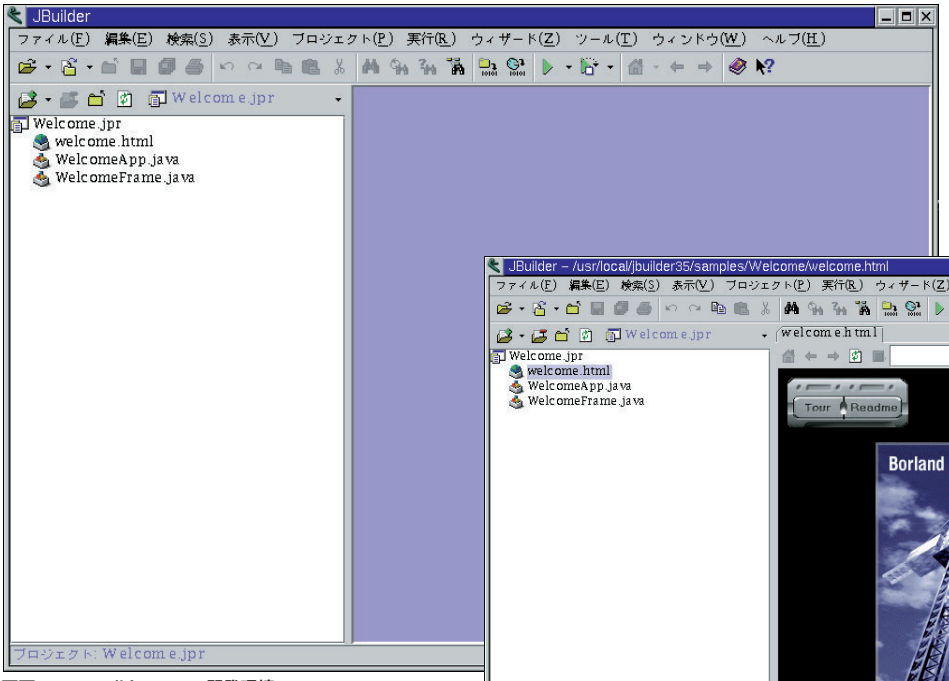
画面10　アイコンの選択



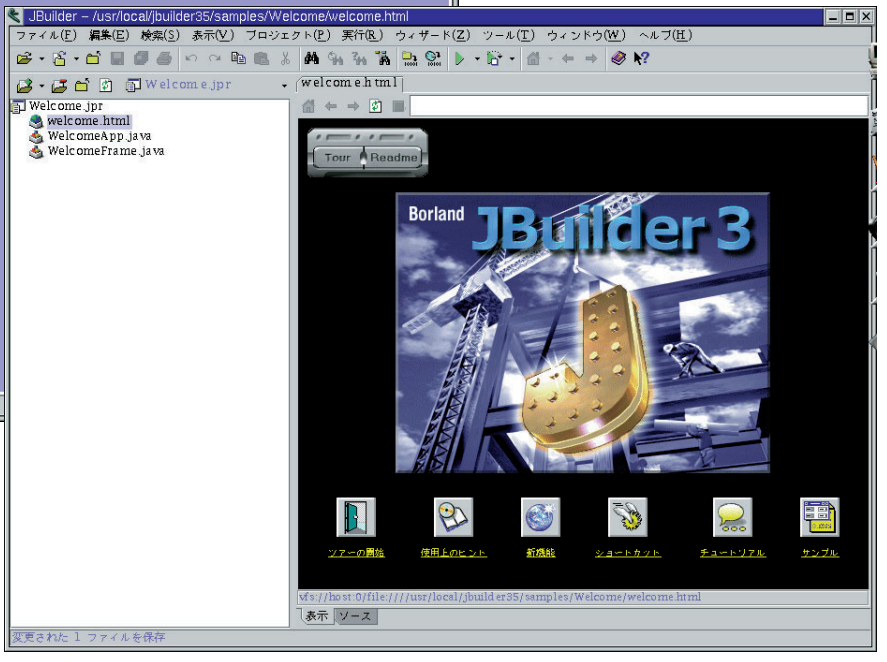
画面9　新規ランチャの設定


画面11　GNOMEパネルに追加されたJBuilderアイコン

画面12　ライセンスキーの設定画面



画面13　JBuilder 3.5の開発環境



画面14　Welcomeプロジェクト

のライセンスキーを入力し、［OK］ボタンを押すと、**画面13**のようにJBuilder 3.5の開発環境が起動します。

最初に起動したときは、Welcomeプロジェクトが開かれています。welcome.htmlをダブルクリックすると、**画面14**のように開発環境でhtmlファイルが実行されます。このプロジェクトファイルはJBuilderの使い方について簡単に説明しています。ツアーを参照すると、開発環境の使い方を簡単に覚えることができますので、ぜひツアーを最後まで進めてください。

JBuilder 3.5を使ってみよう

それでは、JBuilder 3.5を使って簡単なプログラムの作り方を説明しながら、製品の概要を説明していきます。

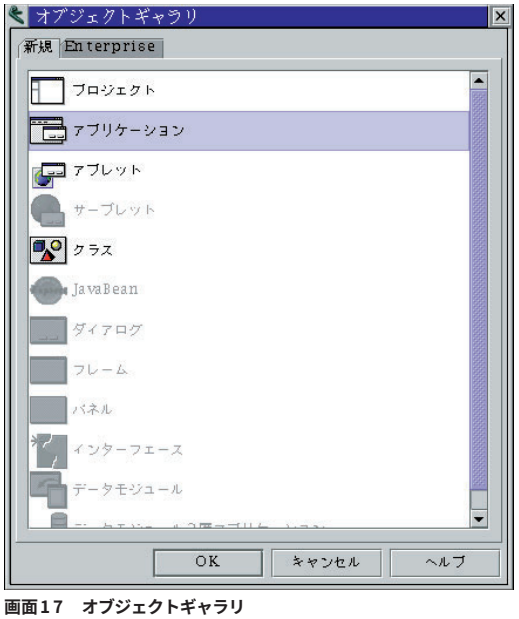
まず、［ファイル］－［プロジェクト"Welcome.jpr"を閉じる］を選択して、Welcomeプロジェクトを閉じます。続いて、［ファイル］－［新規プロジェクト作成］を選択し、**画面15・16**のようなプロジェクトウィザードを起動します。プロジェクトは作成するアプリケーションを管理する機能です。コマンドラインのJDKを使用している場合、アプリケーションのソースコードを管理するのに苦労しますが、JBuilderの提供するプロジェクト管理機能を使用するとすっきりとソースコードを管理することができます。また、開発環境内で実行する場合に最初に実行されるプログラムの指定、VMに渡すパラメータ、コンパイルオプションなどをプロジェクトに登録することができます。

プロジェクトウィザードを進め、作成するアプリケーションの概略を入力し、ウィザードを閉じるとプロジェクトファイルとプロジェクトの概要が書かれたHTMLファイルが作成されます。

続いて、［ファイル］－［新規］を選択します。ここではオブジェクトギャラリ（**画面17**）と呼ばれるウィンドウが表示され、こちらのウィンドウから作成したプログラムの種類を選択します。今回は、「アプリケーション」を選択します。選択すると、**画面18**のようなアプリケーションウィザードが起動されます。アプリケーションウィザードはJavaアプリケーションを作成するときの雛型を作ってくれる支援機能です。ここでは、メニュー、ステータスバー、バージョン情報ダイアログボックスなどの雛型についても生成してくれます（**画面19**）。

画面15　プロジェクトウィザード（1/2）

画面16　プロジェクトウィザード（2/2）



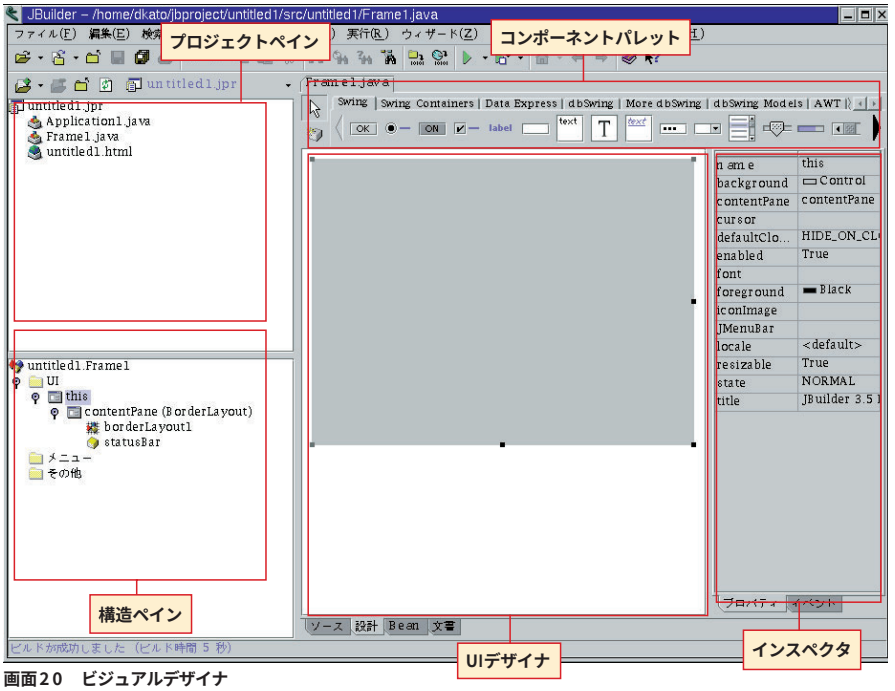
画面17　オブジェクトギャラリ

画面18　アプリケーションウィザード（1/2）

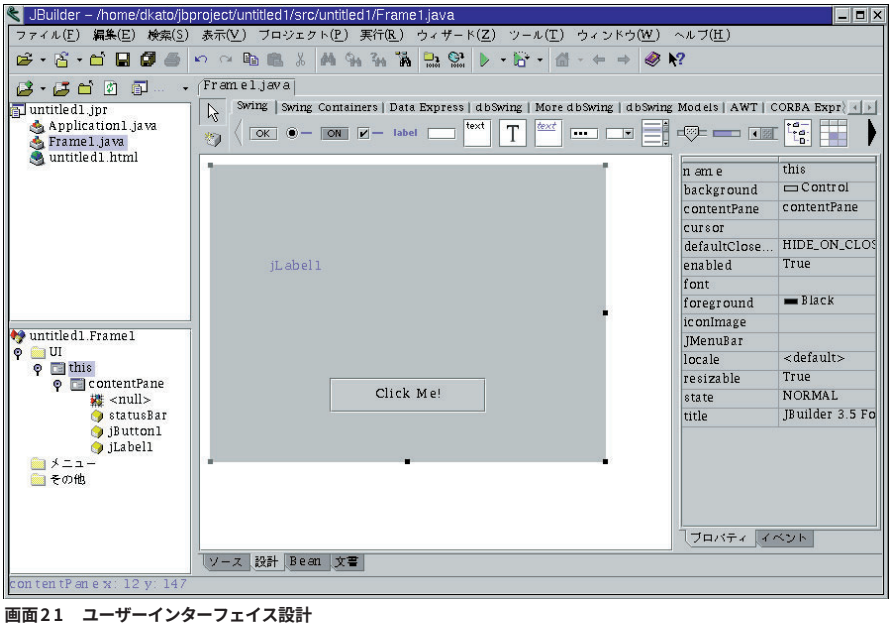
ユーザーインターフェイスの設計

今回は「ステータスバーの生成」と「フレームを画面中央に配置」のみをチェックしてアプリケーションウィザードを終了します。アプリケーションウィザードによって、Application1.javaとFrame1.javaが生成されました。現在選択されているFrame1.javaはJavaアプリケーションのユーザーインターフェイスを作成するクラスです。右下の「設計」のタブをクリックすると、

画面19　アプリケーションウィザード（2/2）



画面20　ビジュアルデザイナ



画面21　ユーザーインターフェイス設計

▼リスト1

```
//カウンターの初期化
int counter = 0;
//ボタンを押したときに発生するイベント
void jButton1_actionPerformed(ActionEvent e) {
   //クリックしたら増加
   counter++;
   //ラベルにクリック数を表示
   jLabel1.setText("Counter="+counter);
}
```

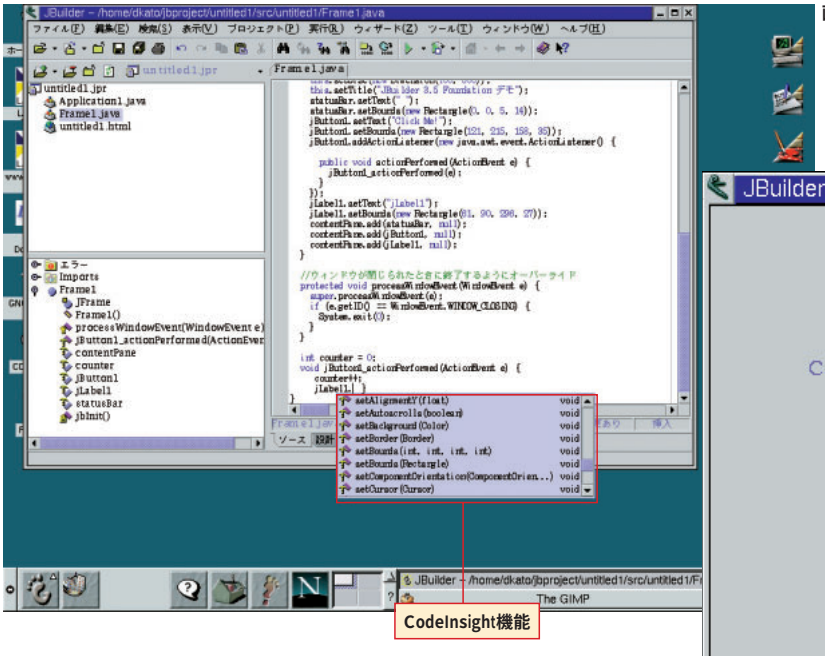
画面20のようなビジュアルデザイナが起動し、ユーザーインターフェイスの設計を行うことができます。

ここでは、パレットに並んでいるJFC/Swing、AWTなどのコンポーネントを配置していくだけでユーザーインターフェイスを作成することができます。コンポーネントのプロパティはインスペクタウィンドウと呼ばれるウィンドウに表示されているプロパティで設定することができます。また、イベントのページでは選択したコンポーネントのアクションについて設定することができます。

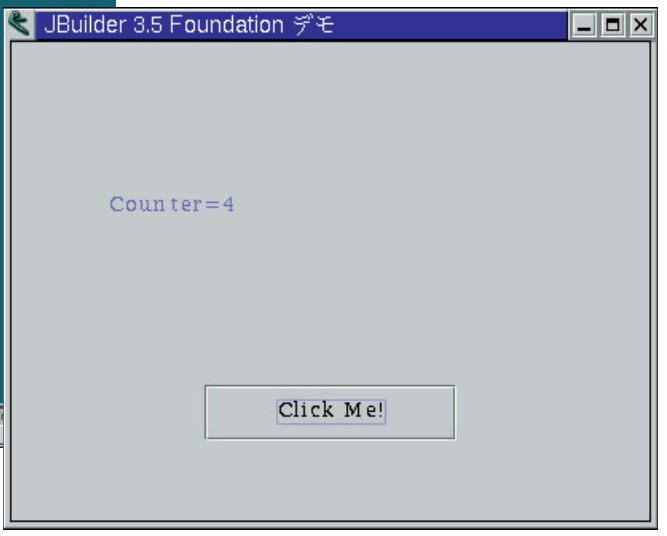
今回は、jLabelコンポーネント、jButtonコンポーネントをフレームに配置し、ボタンをクリックするとそのクリック数がラベルに表示されるようにします（画面21）。まず、構造ペインでcontentPaneを選択し、インスペクタでlayoutプロパティをnullに設定します。次に、これらのコンポーネントを画面上に配置し、jButton1コンポーネントのtextプロパティを「Click Me!」に変更し、jButton1コンポーネントを押したときに発生するactionPerformedをインスペクタウィンドウで選択し、ダブルクリックします。イベントを設定するとソースコードエディタが開きます。ここに、イベントが発生したときに実行したいコードを書きます。

コーディング

JBuilderには豊富なコード支援機能があり、あるオブジェクトが選択可能なメソッドなどを、「.」を入力したときに表示してくれるCodeInsight（画面22）や、コンパイルすることなくソースコードのエラーを左側のオブジェクトツリーに表示してくれるErrorInsight、for文やif文などの構文



画面22　コード入力支援機能



画面23　アプリケーションを実行したところ

を入力してくれる機能などが用意されており、Javaプログラミング初心者でも簡単にプログラミングを行える機能が提供されています。

ここでは、コード入力支援機能を利用してボタンをクリックしたときにカウンタを増加させ、ラベルに表示するコードを記述します。コードは**リスト1**のようになります。

プログラムの作成が終了したら、Application1.javaを選択し、右クリックで表示されるスピードメニューから実行を選択します。ソースコードがコンパイルされ、作成したアプリケーションが実行されます（**画面23**）。

JBuilder 3.5にはこれ以外にも柔軟なブレークポイントの設定や、ソースコードデバッグに対応したビジュアルデバッガ、再利用可能なJavaBeanコンポーネントの作成を支援するBeanExpressなどが用意されており、JDKによるコマンドラインでのプログラミングと比較して、数倍から数十倍の開発生産性を実現することができます。

今回はインストールから簡単な使い方を解説しました。JBuilder 3.5を使えば、GUI上で簡単にプログラムが作成できることが理解できたと思います。次回は、もう少し詳しい使い方、コンポーネントなどの解説をします。

Column

今回使用したのはJBuilder 3.5 Foundationですが、データベースやアプリケーションサーバ構築に適した、Professional版とEnterprise版が用意されています。これらの製品を使えば、データベースやアプリケーションサーバを短期間で開発することができます。

Borland JBuilder 3.5
Professional
価格：68,000円

Borland JBuilder 3.5
Enterprise
価格：360,000円

WINGZによるHyperScriptプログラミング

WINGZ v2.5J for Linux

第2回　実践編　GUI構築とイベント処理

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プログラミングの実践編として、コントロールを使ったダイアログボックスの生成・設定の方法やメニューの作成といった、HyperScriptによるGUI構築の実際と、イベント処理の方法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

前回の復習

HyperScriptプログラミング第1回の基礎編では、HyperScript自体の紹介とスクリプトの記述の仕方を中心に解説しました。第2回の内容に入る前に、前回の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、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

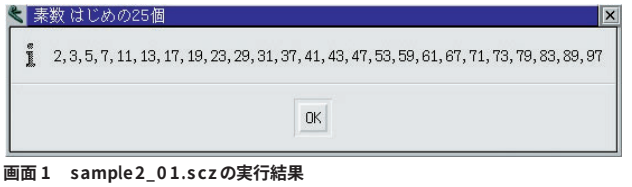
HyperScriptはBASICに似た言語形態のインタープリタです。実行環境としてはWINGZ自身が必要となりますが、インタープリタであるため試作～実行のサイクルが短くてすむなど、一般のツールキットよりも比較的気軽にプログラミングをすることができます。

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行させる、といった方法があります。もちろん、それぞれの記述の中に条件分岐や変数、ユーザー定義関数の定義を行うことができ、他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。

スクリプトの記述例として、第1回で使った素数を求めるサンプルを付録CD-ROMに収録してありますので



画面1　sample2_01.sczの実行結果

(sample2_01.scz)、参考にしてください。このスクリプトでは、ローカル変数とグローバル変数を使い分け、ユーザー関数を条件により呼び出し、メッセージダイアログボックスを表示しています(**画面1**)。

■体験版について

本号の付録CD-ROMに、WINGZ v2.5J for Linuxの体験版と本文で使用するサンプルスクリプトを添付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カ月間の使用が可能です。ただし、最終使用期限は西暦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トをご参照ください。

体験版には、関数のリファレンスマニュアル(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。

GUI構築の実習1:コントロール

GUI構築の実習として、はじめにHyperScriptでダイアログボックス上にコントロールを作成する手順と、コントロールの各種属性を設定する方法について説明します。

■スクリプトによるコントロール生成の実際

後述のダイアログボックスビルダーを利用すれば、マウス操作で簡単にダイアログボックスやコントロールを生成するスクリプトを作ることができますが、ここではまず、コントロールを扱うスクリプトを実際に記述してみましょう。

◇コントロールを生成するスクリプトの例(サイズの自動調整機能)

以下に示す**例02**では、モードレスダイアログボックス上にプッシュボタンコントロールを配置しています(各サンプルのファイル名は付録CD-ROMに収録したスクリプトファイル名です)。

画面2　例02で表示されるダイアログボックス
スクリプトで指定した個数とサイズをもとに、各コントロールアイテムの幅が自動的に決まる。

例02 - sample2_02.scz

```
NEW MODELESS DIALOG BOX "位置指定のサンプル"
AT (-1,-1)(5000,1500)

{1行目}ADD PUSH BUTTON "item 1","item 2", "item 3"
           AT (0,0000)+(5000,500)

{2行目}ADD PUSH BUTTON "item 1","item 2", "item 3",
           "item 4", "item 5" AT
           (0,0500)+(5000,500)

{3行目}ADD PUSH BUTTON "item 1","item 2", "item 3",
           "item 4", "item 5", "item 6", "item 7",
           "item 8" AT (0,1000)+(5000,500)

USE DIALOG BOX
```

プッシュボタンの位置を指定している個所に注目してください。

```
AT (起点X,起点Y)+(5000,500)
```

スクリプトを実行すると、ダイアログの1行目から3行目に並んだプッシュボタンのそれぞれの幅が、下の段にいくほど狭くなっていっています(**画面2**)。HyperScriptでは、指定されているコントロールアイテムの数と、指定されている配置サイズを、できるだけ重ならない位置とサイズになるように自動計算したうえで、コントロールが生成されます。この場合のコントロールの幅は、コントロールアイテムの名前(この場合は「item 1」などのコントロール名)が画面に表示されるべき実際の長さを、このコントロールに指定されているフォント情報をもとに計算して反映しています。

もちろん、コントロールのサイズ指定は任意に行うことも可能ですが、HyperScriptによるコントロールの生成の際には、必要最小限の情報(この場合は各プッシュボタンの名前とだいたいの位置)を与えるだけで、適切な位置とサイズでコントロールが生成されます。

◇コントロールの属性設定の例

WINGZで扱うコントロールには、もちろんコントロールの種類ごとに多数の属性がありますが、ここではすべてのコントロールの制御に共通する基本的な属性の設定例を示します（**例03**）。

例03 - sample2_03.scz

```
NEW MODELESS DIALOG BOX "制御属性のサンプル"
AT (-1,-1)(5000,1000)

ADD PUSH BUTTON "GREY", "DISABLE", "HIDE"
    AT (0,0000)+(5000,500)

ADD PUSH BUTTON "item 1","item 2", "item 3",
    "item 4", "item 5" AT (0,0500)+(5000,500)

SELECT CONTROL 1
SCRIPT
"SELECT CONTROL 4,5,6,7,8","RUN CTSTRING(1,0) & ""
CONTROL""",
"SELECT  CONTROL  1","SHOW  CONTROL  NAME  IF(
CTSTRING(1,0) =""GREY"",""UNGREY"",""GREY"")"

SELECT CONTROL 2
SCRIPT
"SELECT CONTROL 4,5,6,7,8","RUN CTSTRING(2,0) & ""
CONTROL""",
"SELECT  CONTROL  2","SHOW  CONTROL  NAME  IF(
CTSTRING(2,0) =""DISABLE"",""ENABLE"",""DISABLE"")"

SELECT CONTROL 3
SCRIPT
"SELECT CONTROL 4,5,6,7,8","RUN CTSTRING(3,0) & ""
CONTROL""",
"SELECT  CONTROL  3","SHOW  CONTROL  NAME  IF(
CTSTRING(3,0) =""HIDE"",""SHOW"",""HIDE"")"

SELECT CONTROL 4,5,6,7,8 ; SCRIPT "MESSAGE ""dummy""
TITLE ""DUMMY"""

USE DIALOG BOX
```

このスクリプトを実行すると、モードレスダイアログボックス上に、上下2段にプッシュボタンが表示されます（**画面3**の1番上の状態）。下段にあるボタンは制御の対象となるダミーのプッシュボタンですので、表示状態の確認だけに使います。

では、まず上段に表示されている制御用の3つのプッシュボタンのうち、左端の［GREY］ボタンを押してみてください。このボタンを押すと、対象のコントロールに対してGREY CONTROLコマンドが実行され制御対象のプッシュボタンがグレー表示になります（**画面3**のまん中の状態）。グレー表示になったコントロールは、コントロールとしての機能が無効となります。無効になっている下段のプッシュボタンを押そうとしても押せないことを確認してみてください。

次に、先ほど押した左端のプッシュボタンの名前が、［UNGREY］となっていることを確認してから、このボタンを押してみてください。先ほどまでグレー表示だったコントロールが、元の状態に戻ることが確認できるはずです。

このサンプルでは、上段の各制御用プッシュボタンに対して、制御のON/OFFが交互に切り替わるよう設定していますので、このように表示状態も切り替わります。実際のアプリケーションの処理においては、目的に応じて制御のON/OFFを指定していくことになります。

制御コマンドのうち、GREYやHIDEは対象の表示が変わるのでわかりやすいと思いますが、上段中央のボタンに設定したDISABLE CONTROLは、対象の表示状態を変えることなくコントロールの制御を無効化しますので、使用の際に少し注意が必要です。無効化されたコントロールの上にマウスカーソルを移動しても、マウスカーソルの形状がコントロール用に切り替わらないことで、コントロールの無効化を確認することができます。

画面3　例03の実行例
1番上のダイアログボックスが初期状態です。［GREY］ボタンを押すと、中央のダイアログボックスのように、下段のコントロールアイテムがグレー表示される。

これらの制御系コマンドは、それぞれ独立してON/OFFできている点に留意してください。制御系コマンドの代表的な使用例は、WINGZのプリファレンスダイアログボックスです。ここでは、ダイアログボックス左上のポップアップメニューで対象となるエリアを切り替えてみせていますが、実際の処理ではダイアログボックス上に配置されている全コントロールを、選択されたエリアごとに表示／非表示させることでダイアログボックス上のコントロールの切り替えを実現しています。表示されているコントロールについても、状態によってGREY CONTROLやDISABLE CONTORLされるべき状態があることをおわかりいただけると思います。

◇コントロールの生成サイクル

ダイアログボックス上にコントロールを生成するための、作成手順と設定をまとめると以下のようになります。

●ダイアログボックスの定義
●コントロールの生成・処理スクリプト定義
●初期表示状態の設定
●クローズ処理

ダイアログボックスの表示状態を設定するには、ON OPENイベントハンドラを利用すると、生成時に一度だけ行うことができて便利です。**例04**はON OPENイベントハンドラの簡単な使用例です。こういったイベントを処理するイベントハンドラの利用法については後述のイベント処理の実習で詳しく説明します。

例04 - sample2_04.scz

```
NEW MODELESS DIALOG BOX "初期表示のサンプル"
AT (-1,-1)(5000,5000)
SCRIPT
"ON OPEN",
    "CALL " & CURRENTSCRIPTNAME(0) &
        ":UpdateDisplay()",
"END OPEN"

ADD FIELD AT (0,0)(5000,4500)
        CONTROL ALIAS "DayName"
ACCEPT CARRIAGE RETURNS

ADD PUSH BUTTON "LINE NUMBER"
```

4

```
AT (0,4500)+(5000,0500)
SCRIPT
"SELECT CONTROL 1",
"RUN ""FIELD LINE NUMBERS "" &
IF(TEXTFIELDINFO(1,9) = 1, ""OFF"", ""ON"" )"

USE DIALOG BOX

FUNCTION UpdateDisplay()
    DEFINE rt ; rt = CHAR(13)

    SELECT CONTROL "DayName"
    PUT TEXT
        "業務日報" & rt &
        ADATE(NOW(),"元号gg年(西暦yyyy年)
        mm月dd日(") &
        JCWEEKDAY(NOW()) &")" &
        ATIME(NOW(),"hh:mn") & rt
END FUNCTION { UpdateDisplay }
```

4

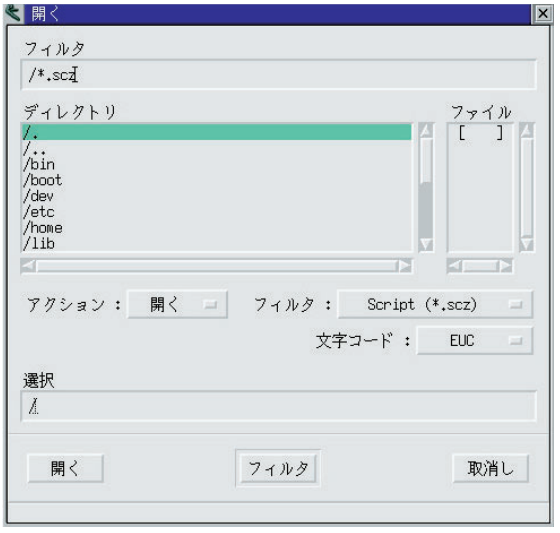
■ダイアログボックスビルダーを使った簡易GUI構築

この項ではマウス操作だけで簡単にダイアログボックスとコントロールを作成できるダイアログボックスビルダーを使った簡易GUI構築の例を紹介します。

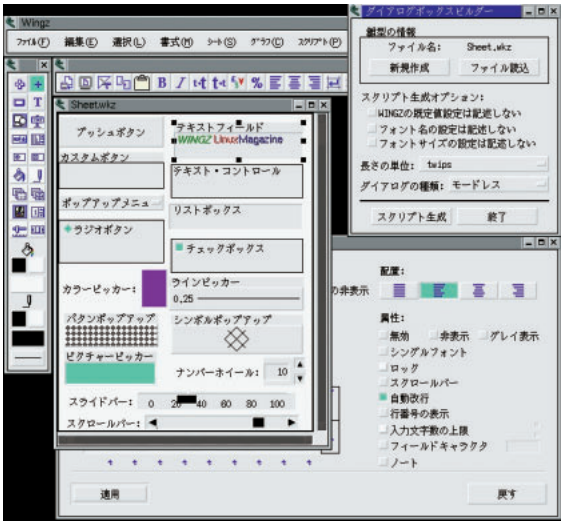
◇ダイアログボックスビルダーの起動と使用方法

ダイアログボックスビルダーの起動用スクリプトファイルは、WINGZ体験版をインストールしたディレクトリの中のwzapps/dbb/dbb.sczです。実行するスクリプトフ

画面4　例04の実行例
ON OPENイベントハンドラを使うことで、ダイアログボックスの初期状態（この例では日時などの文字の入力）を設定している。



画面5　スクリプト・オープンダイアログ



画面6　ダイアログボックスビルダー Sheet.wkz に配置されているのがWINGZで使用可能なコントロール。画面左にあるツールパレットからマウス操作で生成できる。

ァイルの指定は、WINGZの［スクリプト・オープン］ダイアログボックス（**画面5**）で行います。

このダイアログボックスは、WINGZの［ファイル］メニューから［開く］サブメニューの［スクリプト］を選択することで表示されます。このダイアログボックスの［アクション］の項目を「実行」にして起動スクリプトdbb.sczを選択し、［実行］ボタンを押すと、指定したスクリプトファイルが実行されます。

◇雛型ウィンドウの生成

ダイアログボックスビルダーが起動したら、［ダイアログボックスビルダー］ダイアログボックスの［新規作成］ボタンを押して、ダイアログボックスの雛型となるワークシートを作成します。コントロールを配置する前に、作成するダイアログボックスの目標の大きさにワークシートをリサイズしておきましょう。

◇ツールパレットと該当するコントロール

ダイアログボックスビルダーでは、コントロールの配置はツールパレットのコントロールツールを使って行います。作成したいコントロールのツールをマウスでクリックしてから、雛型のワークシート上にマウスをマークしてコントロールを作成します。コントロールの位置やサイズも、このとき指定します。

◇コントロールの属性設定方法

コントロールにハンドル（コントロールの外周に現れる小さな黒い四角）が表示された状態（選択状態）で、コントロールの上にマウスカーソルを移動させてマウスの右ボタンを押すと、選択中のコントロールに対するサブメニューが表示されます。このメニューの［文字］―［カラー／ライン］や［コントロールオプション］アイテムを選択することで、各種属性設定ダイアログボックスが呼び出せます（**画面6**）。適宜、属性を設定してください。雛型上のコントロールに設定した属性は、そのままスクリプトに反映されます。

◇スクリプトの生成と実行

雛型の作成が終了したら、［ダイアログボックスビルダー］の［スクリプト生成］ボタンを押してください。必要に応じてメッセージボックスが表示されますので、その指示に従っていくと、雛型をもとにしたダイアログボックスとコントロールを生成するスクリプトが表示されます。

生成されたスクリプトをアクティブにして、実行してみてください（WINGZの［スクリプト］メニューの［実行］を選ぶか、Ctrl－2を押します）。雛型と同じサイズ、同じ属性でダイアログボックスとコントロールが表示されるはずです。

このようにダイアログボックスビルダーを利用することで、コントロール作成の際のサイズと位置の指定、初期属性の設定といった面倒な工程を省略することができます。実際には、ダイアログボックスビルダー自身もHyperScriptで記述された1つのスクリプトです。体験版においても、スクリプトの内容を公開していますので動作を確認してみてください。

WINGZで扱えるコントロール

種類別にコントロールの概要を解説していきましょう。

◇プッシュボタン、カスタムボタン

プッシュボタンは、これまでにも例にあげたとおり、コントロールの基本となるものです。カスタムボタンはプッシュボタンと異なり、デフォルト設定ではボタン枠や塗りつぶしの指定がありません。後述するON REPAINTイベントハンドラを使って、ボタン上に画像を貼り付けたり描画したりするのに利用することができます。

◇ラジオボタン、チェックボックス、リストボックス、ポップアップメニュー

リストの表示、選択を行う場合は、リストボックスやポップアップメニューを利用します。ポップアップメニューやラジオボタンは、一度に1つのアイテムしか選択させたくない場合などに利用します。

◇テキスト、テキストフィールド

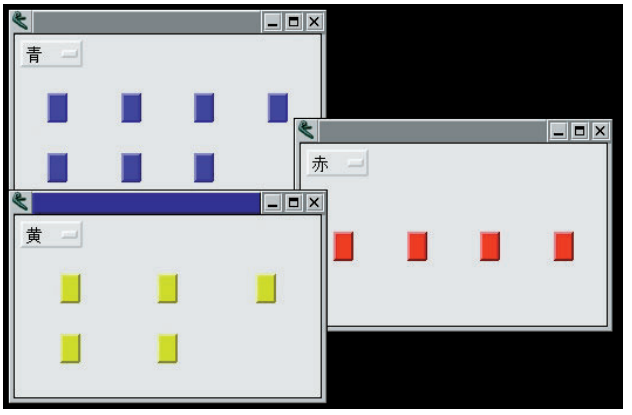
テキストコントロールは、HyperScriptコマンドを発行しない限り、文字列の内容を変更できないコントロールです。これに対して、テキストフィールドは文字列の書き込みが行えるコントロールです。後述のON KEYイベントハンドラなどを使うことで、独自のキー入力処理などを実装することができます。

その他、色／線／パターンの指定や、スケールを使った数値の指定も、ラインポップアップ、パターンポップアップ、カラーポップアップ、シンボルポップアップ、ピクチャーポップアップ、ナンバーホイール、スクロールバー、スライドバーなどのコントロールを使って行うことができます。

■コントロール制御のワンポイント

◇配列を使ってグループ別にコントロールを表示する手法

ダイアログボックスにコントロールを生成していくと、生成した順にコントロールのID番号が割振られます。生成したコントロールを選択する場合には、このコントロールIDや、コントロール名／コントロールタイトル、コントロールエイリアスを使って指定することができます。コントロールを選択するコマンドはSELECT CONTROLですが、コントロールの指定を配列に格納し、SELECT CONTROLにその配列名を指定することで、特定のコントロールのグループをまとめて選択することもできます。次の**例05**のスクリプトでは、ポップアップメニューを選択するたびに、ダイアログボックス上で表示するコントロールの種類を切り替えています。例05の実行例を**画面7**に示します。



画面7　例05の実行例
ダイアログボックス左上にあるポップアップメニューの選択によって、ダイアログボックスのコントロールの表示／非表示が切り替わる。

例05 - sample2_05.scz

```
DEFINE group[3,10]

group[1,10] = 7
group[2,10] = 4
group[3,10] = 5
GET group[1,1..group[1,10]] FROM 2,3,4,5,6,7,8
GET group[2,1..group[2,10]] FROM 9,10,11,12
GET group[3,1..group[3,10]] FROM 13,14,15,16,17,18

NEW MODELESS DIALOG BOX "" AT (-1,-1)(5000,3000)

ADD POPUP MENU "青", "赤", "黄"
AT (100,100)+(1000,500)
    SCRIPT "CALL " & CURRENTSCRIPTNAME(0) &
    ":SelectGroups( CTVALUE(0,0) )"

ADD PUSH BUTTON "","","","","","",""
AT (100,700)+(4800,2000)
ADD PUSH BUTTON "","","",""
AT (100,700)+(4800,2000)
ADD PUSH BUTTON "","","","",""
AT (100,700)+(4800,2000)

SELECT CONTROL group[1,1..group[1,10]]
    FILL BG BLUE()
```

5

```
SELECT CONTROL group[2,1..group[2,10]]
    FILL BG RED()

SELECT CONTROL group[3,1..group[3,10]]
    FILL BG YELLOW()

CALL SelectGroups(1)

USE DIALOG BOX

FUNCTION SelectGroups( val )
    SELECT CONTROL 2,3,4,5,6,7,8,9,10,11,12,13,14,
    15,16,17 HIDE CONTROL
    SELECT CONTROL group[val,1..group[val,10]]
        SHOW CONTROL
END FUNCTION { SelectGroups }
```

5

GUI構築の実習2：メニュー

コントロールに続くGUI構築の実習として、HyperScriptでメニューを作成する手順を説明します。

■WINGZのメニュー構成要素

次のスクリプトを実行してみてください。

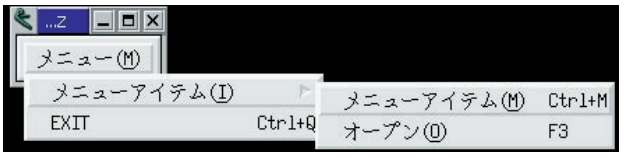
```
SHOW MENUBAR "short"
```

WINGZのメニューの配置が変更されるはずです。このようにWINGZでは、複数のメニューバーを登録しておき、条件によってそれらを切り替えて使用することが可能です（元のメニューバーに戻すにはSHOW MENUBAR "long"を実行してください）。

HyperScriptのメニュー構築では、はじめにメニューバーを定義し、次にメニューバーに所属するサブメニューとメニューを定義します。サブメニューは名前のとおり、メニューの下位に表示される要素ですが、作成順序としては、あらかじめサブメニューを定義しておいた後で定義済みサブメニューをアイテムとして指定する、という手順になります。

■スクリプトによるメニューの実装方法

実際に新しいメニューバーをスクリプトで定義して表示させてみましょう。**例06**のスクリプトでは、ごく簡単な構成のメニューバーを作成しています（**画面8**）。新たにメニューバーやメニューを作り直さなくても、現在のメニューバーやメニューに要素を加えていくことも可能です。



画面8　スクリプトで作成した簡単なメニュー
ショートカットなどのキーアサインも、スクリプトで指定可能。

例06 - sample2_06.scz

```
ADD MENUBAR "test"

ADD SUBMENU "サブメニュー" WITH 2 ITEMS
    ADD MENUITEM "メニューアイテム(~M)" KEY "M"
    COMMAND "MESSAGE ""メニューアイテムからコマンド
        を呼びました"""
    ADD MENUITEM "オープン(~O)" KEY 4096 + 3 COMMAND
        "OPEN DIALOG"

ADD MENU "メニュー(~M)" WITH 2 ITEMS
    ADD MENUITEM "メニューアイテム(~I)" SUBMENU
        "サブメニュー"
    ADD MENUITEM "EXIT" KEY "Q" COMMAND
        "SHOW MENUBAR ""long"""

SHOW MENUBAR "test"
```

メニューバーとその要素で定義しているスクリプトも、コントロールスクリプトと同じように記述していることがおわかりいただけると思います。

■メニューバービルダーを使ったメニュー作成

ダイアログボックスビルダーと同様、GUIベースでメニューバーを簡易作成できるツール「メニューバービルダー」を体験版にも添付しています。起動スクリプトのあるディレクトリにドキュメントファイルを収めてありますので、ダイアログボックスビルダーと合わせて試してみてください。メニューバービルダーの起動スクリプトは、wz25demo/wzapps/mbb/mbb.sczです。

画面9　マウスイベントによる制御の例
マウス操作（マウスボタンを押す・放す・マウスカーソルを動かす）によって発生するイベントに応じて、コントロール名の表示が変わる処理を設定している。

イベント処理の実習

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。GUIベースのアプリケーションにおいて、イベントの処理は必須の機能です。ここではHyperScriptにおけるイベント処理について概要を解説します。

◇イベントとイベントハンドラ

マウスのクリックや、ウィンドウの開閉、エラー状況といった動作や状態をスクリプトに伝えるためのメカニズムを、HyperScriptではイベントと呼び、そのときに呼び出されるスクリプトのサブルーチンをイベントハンドラと呼称します。たとえば、あるコントロールに対してマウスをクリックしたり、ウィンドウのサイズが変わったりした状態はそれぞれイベントが発生したことになり、対象となるコントロールやウィンドウのスクリプトにイベントハンドラが記述されていれば、それぞれのイベントに対する処理が実行されます。

■HyperScriptによるイベント処理の実際

イベントハンドラは、コントロールやウィンドウにアタッチしたスクリプトに記述しておくことで、対象に対して所定のイベントが行われた際に呼び出されます。1つのコントロールに対して複数のイベントハンドラを記述することもできますし、逆にイベントハンドラをいっさい記述しないでおくことも可能です。

■マウス関連のイベントハンドラの例

次のサンプル（**例07**）では、モードレスダイアログボックス上のプッシュボタンコントロールに対して、ON MOUSEDOWN、ON MOUSEUP、ON MOUSEMOVEの3つのイベントハンドラを定義しています。スクリプトを実行し、表示されるプッシュボタンに対して、マウスをクリックしたりコントロール上でマウスをクリックしたままマウスカーソルを動かしてみてください。それぞれの動作（イベント）に対して、イベントハンドラであらかじめ定義しておいた処理（この場合は単純にコントロール名を変更すること）が、逐次行われていることが確認できるはずです（**画面9**）。このように、ON XXX～END XXXで記述した個所がイベントハンドラです。イベントハンドラ中に記述したスクリプトが、それぞれのイベント発生時に自動的に呼び出されます。

例07 - sample2_07.scz

```
NEW MODELESS dIALOG BOX "" AT (-1,-1)(3000,3000)

ADD PUSH BUTTON "" AT (0,0)(3000,3000)
SCRIPT
"ON MOUSEDOWN",
    "SELECT CONTROL 1",
    "SHOW CONTROL NAME ""MOUSE DOWN""",
"END MOUSEDOWN",
"ON MOUSEUP",
    "SELECT CONTROL 1",
    "SHOW CONTROL NAME ""MOUSE UP""",
"END MOUSEUP",
"ON MOUSEMOVE",
    "SELECT CONTROL 1",
    "SHOW CONTROL NAME ""MOUSE MOVE""",
"END MOUSEMOVE"

USE DIALOG BOX
```

◇ウィンドウの動作に関連するイベントハンドラの例

ワークシートやダイアログボックスに対しては、次のイベントハンドラを定義することができます。

```
ON OPEN, ON CLOSE, ON SAVE, ON RESIZE, ON ACTIVATE,
ON REACTIVATE
```

ON OPEN、ON CLOSE、ON SAVEイベントハンドラは、それぞれウィンドウが「開く際」、「閉じる際」、「保存される際」に、各イベントが実行される前に呼び出されます。したがって、イベントハンドラに条件を設定し、途中

で中断（ABORT OPENなど）させることもできます。たとえば、オペレータが所定の処理を終えるまでは途中でウィンドウを閉じないようにするなどといった処理に有効です。**例08**にON CLOSEイベントハンドラの例を示します。

例08 - sample2_08.scz

```
NEW MODELESS DIALOG BOX "" AT (-1,-1)(6000,2000)
SCRIPT
"ON CLOSE",
    "IF CTVALUE(1,1) * CTVALUE(1,2) * CTVALUE(1,3)
* CTVALUE(1,4) <> 1",
        "ABORT CLOSE",
    "END IF",
"END CLOSE"

ADD CHECK BOX "item 1", "item 2", "item 3","item 4"
        AT (0,0)(6000,1000)
    SHOW CONTROL TITLE "全てチェックしてください"

ADD PUSH BUTTON "CLOSE" AT (0,1000)+(6000,1000)
    SCRIPT "CLOSE NOW"

USE DIALOG BOX
```

その他、ON RESIZEイベントハンドラは、ウィンドウのサイズが変更された際に実行されるイベントハンドラで、ON ACTIVATE、ON DEACTIVATEは、それぞれ「ウィンドウがアクティブになった際」、「他のウィンドウへアクティブ状態が遷移する際」に実行されるイベントハンドラです。

◇入力に関連するイベントハンドラの例

テキストフィールドやナンバーホイールといった、キーボードから文字や数字を入力することができるコントロールに対しては、次のイベントハンドラを定義することができます。

```
ON KEY, ON MOVEMENTKEY, ON ENTER, ON EXIT
```

ON KEYイベントハンドラは、入力コントロールにキャラクタキーを使った入力が行われる際に実行されます。ON MOVEKEYイベントハンドラはON KEYと異なり、キャラクタキーではなくスクロールアローキーなどの特殊

画面10　ON CLOSE イベントハンドラによるウィンドウの制御
ウィンドウを閉じようとしたときにチェックボックスのすべてのアイテムがチェックされていないと、操作は中断されてしまう。

画面11　ON KYEイベントハンドラの例
下段のテキストフィールドへのキー入力によって発生するイベントに反応して、入力したキャラクタとその文字コードが上段のテキストコントロールに表示される。

キーに対するイベント処理を行うための窓口になります。

ON ENTER、ON EXITはそれぞれ、入力コントロールに対して入力フォーカスがあたった際に実行されるイベントハンドラと、入力フォーカスが外れた際に実行されるイベントハンドラです。

例09では、テキストフィールドに文字が入力されると、文字とそのキャラクタコードをテキストコントロールに表示する（**画面11**）スクリプトを記述しています。

ON KEYイベントハンドラの中では、 KEY( )関数を使って入力した文字のキャラクタコードを取得することができます。この例ではキャラクタコードが32未満の文字制御コードを排除していますが、実際には文字制御コードが入力された場合の処理（たとえばBackSpaceキーやTABキーなどの処理）も、ON KEYイベントハンドラの中で定義できます。サンプルでは、入力文字を処理した後、ABORT KEYを発行することで、入力した文字自体のテキストフィールドへの書き出しを抑制しています。このように、ON KEYイベントハンドラでは、入力した文字の処理を独自に行わせることも可能です。

例09 - sample2_09.scz

```
NEW MODELESS DIALOG BOX "" AT (-1,-1)(3000,2000)

ADD TEXT "" AT (0,0)+(3000,1000)
NO LINE BORDER ; ALIGN CENTER

ADD FIELD AT (0,1000)+(3000,1000)
ALIGN CENTER
```

```
SCRIPT                                                    9
"DEFINE current_key",
"ON KEY",
    "current_key = KEY()",
    "SELECT CONTROL 1",
        "IF current_key > 31",
            "SHOW TEXT CHAR( current_key ) & "" -
""
            & current_key", "ABORT KEY",
        "ELSE",
            "SHOW TEXT """""",
        "END IF",
"END KEY"

USE DIALOG BOX
```

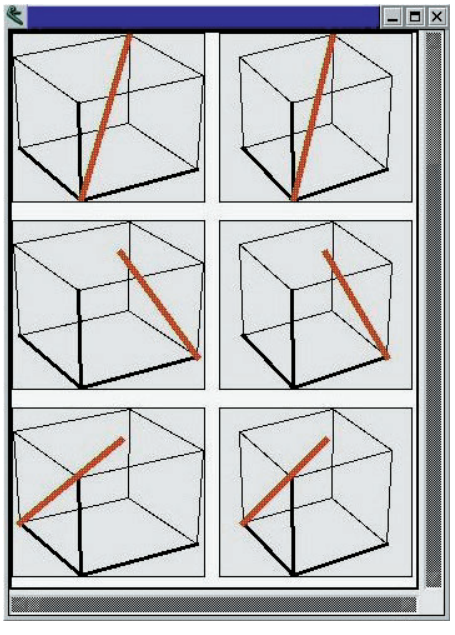
■その他のイベントハンドラ

ON ERRORイベントハンドラは、それ自身が記述されているスクリプトの実行時にエラーが起きた場合に呼び出されるイベントハンドラです。スクリプトのデバッグを行う際に、既知のエラーを無視して実行を続けたり、あるいはその逆にエラー発生個所の究明を行うために利用することができます。

ON IDLEイベントハンドラは、WINGZに対してユーザーが操作（マウス操作、キー入力等）をいっさい行っておらず、他のスクリプトも実行されていない場合に、定期的に発生する「アイドルイベント」によって呼び出されます。このイベントハンドラはWINGZの従来のバージョンとの互換性を保つために残されている意味合いが強いため、現在のバージョンにおいてはON IDLEイベントハンドラよりも、所定の時間（実行時刻または実行間隔の指定）がくると、指定しておいたスクリプトが実行されるユーザーイベント機能を利用するほうがよいでしょう。

最後にON REPAINTイベントハンドラですが、これは対象となるコントロールに対して再描画イベントが起きた際に実行されるイベントハンドラです。再描画とは、具体的には他のウィンドウの背面になって隠れていたコントロールやウィンドウが、前面に表示（リアライズ）される際に描画が行われた場合などが対象となります。カスタムボタンコントロールに対してON REPAINTイベントハンドラを使うことで、画像データを表示させたり、カスタムボタン上に定義した2次元または3次元の座標において、ユーザーが定義したグラフや図形を作画することが可能になります（**画面12**）

体験版に添付している試験飛行（wz25demo/sample/testfl2j.wkz）の起動画面（**画面13**）の飛行機の描画も、カスタムボタンに対するON REPAINTイベントハンドラを利用して描画させています。このように、HyperScriptを使ってシーケンシャルに図形が描画できることと、ワークシート上のカスタムボタンについては画像をJPEG形式などのピクチャーデータに保存できることを利用して、幾何学図形や、繰り返し描画させた図形、カスタムグラフなどをカスタムボタンに描画し、その後画像ファイルに保存するといった処理を行うことも可能です。



画面12　カスタムボタン上の描画例
カスタムボタン上に図形を描画する際には、座標系のスケール・方向なども指定できる。

今回はGUIの構築とイベント処理を中心に解説しました。次回は、DataLink機能を使ったリレーショナルデータベースとの接続、問い合わせの実際と、HyperScriptとC言語とのリンクについて解説します。

画面13　ON REPAINTイベントハンドラによる描画の例
画面上の飛行機は、マウスのドラッグ操作によって発生するイベントにより視点を変更する処理が行われ、マウスボタンを放したタイミングで再描画される。

隠喩としてのコンピュータ

インターフェイス的主体の錯誤

文：豊福 剛
Text : Tsuyoshi Toyofuku

東 浩紀は『サイバースペースは何故そう呼ばれるか』（『InterCommunication』、1997年秋号～2000年春号、NTT出版）という連載で、興味深いハイパーメディア論を展開している。この論考の問題設定の中心には、インターネットがサイバースペースという空間（スペース）の隠喩によって語られることへの疑いがあり、そこに潜むイデオロギーを考察している。また、シェリー・タークルの「インターフェイス・バリュー」論をふまえて、GUI以前と以後でのコンピュータに対する認識論的な断絶の意味を考察し、GUI以後のコンピュータについて、イメージとシンボルが分割されない、新しい記号的様態と解釈している。

東の考察は、論点が多岐にわたり、錯綜しているため、決して読みやすくない。ある対談では「大幅に書き直すつもりです」（『広告』1999年11・12号、博報堂）と発言している。ただし、イメージとシンボルが区別されない記号を成立させるものとしてGUIを捉える着想は一貫している。そこで、ここでは、東の論点を整理しつつ、批判的に検討してみたい。

映画の構造とGUIの構造

東は、映画というメディアの構造とGUI以後のコンピュータのメディアとしての構造を、次のように対比させている。

映画を見るとき、見る側と見られる側は峻別されない。映画を「見る」とは、スクリーンの上を流れるイメージを眺めるだけでなく、同時にそのイメージを構成する視線そのもの、つまりカメラの位置と構図の時間的推移にも付き従う経験である。見る側（主体）と見られる側（客体）、カメラとスクリーンとをたえず往復する。映画を見る主体は、スクリーン（見えるもの）とその背後（見えないもの）の往復、すなわち、想像的同一化（見えるもの）と象徴的同一化（見えないもの）のあいだの往復によって組織される。

これに対して、GUIのスクリーンを見る主体（インターフェイス的主体）は、スクリーンをたえずイメージとシンボルとに二重化することで組織される。インターフェイス的主体は、スクリーンの背後を認めない。そこには、スクリーンとその背後という二重構造はなく、スクリーンしか存在しない。しかし、スクリーンにおいてはイメージしか存在しないのではなく、「デスクトップ」や「ゴミ箱」といったイメージはシンボルとして解釈される。イメージを一方で虚構と知りつつ、他方でそれをシンボルとして受け

入れるが、イメージとシンボルはともにスクリーンの上に（二重化されて）見出される。

整理すると、想像的同一化と象徴的同一化の対が、映画においては、見えるもの（スクリーン）と見えないもの（カメラ）の対に、GUIにおいては、見えるもの（イメージ）と見えるもの（シンボル）の対に、それぞれ対応することが前提になっている。

イメージとシンボルの二重化という錯誤

この図式は、混乱しているように思える。映画における象徴的同一化は、他者（カメラ）の視線への同一化であるが、GUIにおける象徴的同一化は、シンボルの意味を理解することである。したがって、映画とGUIとでは、象徴的同一化の内実に違いがある。

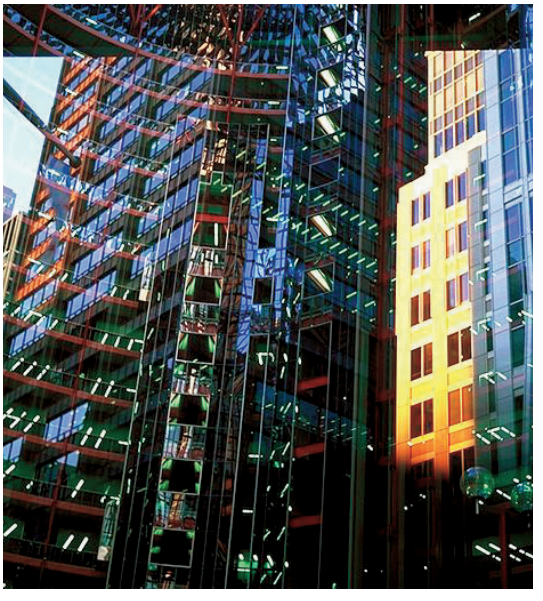
私は、GUIにおいては、想像的同一化と象徴的同一化の対は、見えるもの（スクリーン）と見えないもの（プログラム）の対に対応すると考えている。そして、スクリーンの背後にあるプログラムは、プログラミング言語によって記述されたものである。映画において、カメラが捉えた映像がスクリーンに投影されるのと同じように、コンピュータにおいては、プログラムが生成する映像がスクリーンに投影される。

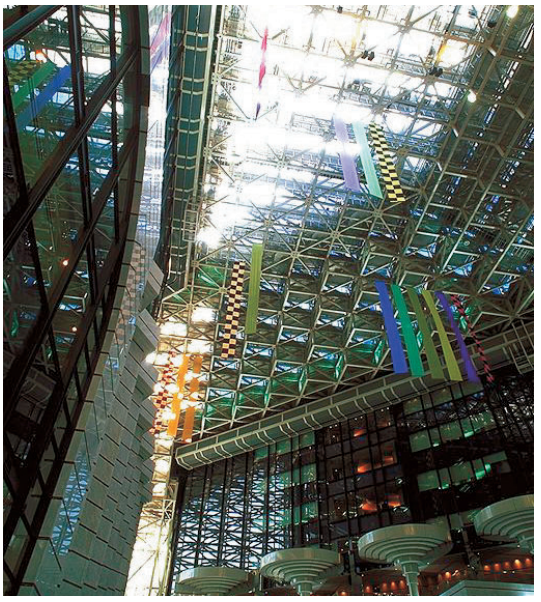
したがって、GUIにおける象徴的同一化とは、たとえばアイコンを操作することによって、それに対応して実行されるプログラムの処理内容を理解することであるはずだ。映画において、スクリーンとその背後という二重構造があるのと同じように、GUIにおいても、同じ二重構造がある。

インターフェイス的主体を肯定的に評価する東は、GUIにおけるこの二重構造を認めない。このため、スクリーン上にはイメージとシンボルが二重化されている、という奇妙な言説になってしまう。その原因は、GUIにおいて何がシンボルなのかの理解が混乱している点にある。

ある箇所では、イメージをスクリーン上の絵に、シンボルをスクリーン上の文字に対応づけている。しかし、絵と文字がひとつの平面に配置されるということであれば、マンガや絵本、あるいは字幕つきの映画もそうなのであって、なにもGUIにかぎった話ではない。さらにいえば、漢字のような表意文字は、文字そのものにおいてイメージとシンボルが二重化されている、とすらいえる。

あえてGUIのメディアとしての新しさを強調するとすれば、それはスクリーン上に絵と文字が配置される点にではなく、スクリーン上のオブジェクトに対して何らかの操作





を行うと、それに対応してオブジェクトがアクションを実行する点にある。このインタラクティブ性にメディアとしてのGUIの新しさがある。しかし、このインタラクティブ性は、スクリーンの背後にあるプログラムの効果にすぎない。

エクリチュールとしてのプログラムの排除

もうひとつ批判的に検討しておきたい論点がある。それは、スクリーンにおいてイメージとシンボルが二重化されるGUIを、ジャック・デリダの「エクリチュール」という概念と関連づけている点である。

東によれば、デリダが論じるエクリチュールとは、「見えるものと見えないもの、目の記号（絵）と耳の記号（音声）のあいだにある審級」を指す。それは「視覚的分割を撹乱する記号、イメージとシンボルを相互陥入させ、『目と耳のあいだの空間』を開く」。しかし、このエクリチュールが排除されることで、見えるものと見えないもの、文字記号と音声記号という二項対立が現れる。デリダは、これをエクリチュールの排除による転倒である、と考えた。GUIにおいては、イメージとシンボルは対立するものではなく、スクリーン上に二重化されている。このことから、GUIはデリダのいうエクリチュールを体現している、と東は考える。

しかし、スクリーンの背後にあるプログラムは、プログラミング言語で記述された記号である。私は、プログラムは「見えるものと見えないもののあいだにある審級」であると考える。エクリチュールとしてのプログラムが排除されることで、スクリーンとその背後の対立、見えるものと見えないものの分断が現われ、インターフェイス的主体が構成されるのだ。

スクリーンの背後にあるプログラムは、ソースコードであるかぎりでは、見えるものである。しかし、バイナリコードに変換され、ソースコードへのアクセスが分断されるとき、それは見えないものに転化してしまう。したがって、エクリチュールとしてのプログラムの排除は、超越論的にではなく、歴史的・政治的に問われなければならない。

サイバースペースとオリエンタリズム

東のGUI論には、以上のような異論があるものの、連載のタイトルが端的に示す問題設定、すなわち、インターネットが「空間」の比喩で語られることへの疑い、について

は評価している。

「サイバースペース」という語は、1984年に発表されたウィリアム・ギブソンの『ニューロマンサー』というSFからはじまる。私は、熱心なSF読者ではないのだが、それでも当時『ニューロマンサー』に代表されるサイバーパンクものを読み漁ったのは、そこにパンク的なものが書き込まれていたら嬉しいな、という期待があったからだ。サイバーパンクSFのアンソロジーである『ミラーシェード』に収録されていたブルース・スターリングによるサイバーパンク宣言には、サイバーパンク運動の文化的背景のひとつとして、パンク〜ニューウェイブ系のロックの重要性が指摘してあった。とはいえ、実際のサイバーパンク小説は、サイバー的なもののイメージを打ち出しているだけで、SFとしては退屈な内容だった。サイバーパンクにおけるパンク的なものについても同様の印象をもった。

サイバーパンク的な感性は、小説よりもむしろ映画『ブレードランナー』によって確立されたように思う。この映画の音楽は、バンゲリスというプログレバンドが担当していて、全然パンク的ではないのだが、その映像はインパクトがあった。酸性雨の降り注ぐ中、巨大な電光掲示モニタには芸者の顔アップが映されるし、レプリカントを追跡するハリソン・フォードは、ダウンタウンの屋台でうどんをすすっていたりする。『ニューロマンサー』の舞台が千葉であるのと同じ理由で、当時のサイバーパンク的美学においては、オリエンタルな要素が重要なアイテムになっていた。

80年代前半、SFや映画で表現された「サイバースペース」のイメージには、なぜオリエンタルな要素が混在しているのか？ それは「悪魔祓い」の結果であると東は説明している。

情報メディア技術の浸透によって、「ひとりの人間がひとつの場所でひとつのことをする」という前提が脅かされる。この「不気味さ」から免れるためには、「不気味なもの」を外部に疎外しなければならない。そのため「不気味なもの」は、（西洋にとって）地理的遠方であるオリエントに局所化され押さえ込まれるのだ、と。

このような「悪魔祓い」を設定せずに、情報メディアの「不気味さ」に対応した作家として、東はフィリップ・K・ディックを取り上げている。興味深いのは、ディックが、「不気味なもの」に擬似生物性を見出し、それを感情移入できる「可愛いもの」に逆転させた点だ。このことは、Linuxのカーネルはなぜペンギンにシンボライズされるのか、という問題とつながる気がしている。

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

知的所有権をめぐるエピソードたち

文：安田幸弘
Text : Yukihiro Yasuda

アジアの大都市には、必ずと言っていいほど、海賊ソフトのマーケットがある。いつだったか、ジャカルタで非営利サイトのポストマスターが集まったとき、みんなでゾロゾロとジャカルタの海賊ソフトマーケットにでかけたことがある。ぼくはWindowsのソフトは買わない・使わない主義なので、見るだけだったのだけれど、みんなここぞとばかりに高価なWindows用ソフトを買い込んでいた。スリランカの某氏なんぞは、「メーカー小売希望価格」で計算したら1千万円ぐらいの買い物をしていたようだ。でもジャカルタ価格ならこれが数千円。知的所有権もヘッタクレもあるもんか、である。あるトルコ人は「オレたちの国では、NT ServerだろうがPhotoshopだろうが、トマトやキュウリと一緒に道端の屋台で売ってるんだぜ」なんて、妙な自慢をしていた。ゲイツ氏がキュウリと一緒に売られているNT Serverを見たら、激怒するだろうなあ。

苛烈なるアメリカ流ビジネス

ゲイツ氏の功績は、技術的にはマイコンにBASICを実装したこと、ビジネス的にはDOSをIBMに売ったことなどが思い出されるけれど、本当の「功績」は、マイコン時代にソフトウェアと知的所有権という概念を「正しく」結びつけたことだろう。もっとも、知的所有権という考え方は、決して新しいものじゃない。知的所有権を武器にライバルを（ときとして少々エゲツない方法で）蹴落としてきた企業は、決してゲイツ氏の会社ばかりではなかったようだ。先日、「トウガラシの文化誌（晶文社）」という痛快な本を読んだのだが、その中にホットソース界のマイクロソフトともいうべきマキルヘニー社に関するエピソードが書かれていた。マキルヘニー社は商標権を武器とした法廷闘争でライバルを蹴散らし、あるいは買収して、「タバスコ」という固有名詞を持つありふれた唐辛子のソースを世界のブランドに育て上げたという。あの国で弁護士が繁盛するワケである。

まあ、ホットソースだろうがソフトウェアだろうが、あるいは名前も知らない雑草だって、知的所有権を正しく（？）使いさえすれば、巨万の富を築けるのがアメリカなのだろう。昨今は、アメリカ流ビジネス全盛で、バブルだの高過ぎだのと言われながらも米国企業の株価は上がる一方、ビジネスをやるんだったらアメリカ流しかない、みたいな雰囲気だし、アメリカはアメリカで「オレたちのやり方が一番。オレたちのやり方でやらないのはフェアじゃない」と、やたら鼻息が荒い。

海を隔てて、そんなアメリカ流ビジネスを見ていると、確かに彼らのやり方は悪くないんじゃないかという気もする。たぶん、金儲けをしたい人にとっては、良いやり方なんだろう。だけど、そもそもアメリカは、先住民が暮らしていた広大な土地を「ここまではオレのもの」と宣言して建てた国であり、アフリカから「こいつとこいつはオレのもの」と宣言して拉致してきた労働力で経済の基盤を作ったことを忘れちゃいけない。

カレー粉も知的所有権

ぼくは最近、アメリカ人の所有権に関する認識は、根本的に歪みまくっているんじゃないか、と思っているのだ。昨今の欧米系多国籍企業が知的所有権を盾に、第三世界で猛威をふるっているのを見るとなおさらである。

身近なところでは、GNUが米国のソフト関連の知的所有権に異議を唱えているのはご存じの通りだが、最近は第三世界の国々で、こうした欧米の知的所有権の主張ははおかしいぞという声が高まっている。有名な例に、インドのターメリックの特許がある。アメリカの特許制度では、とにかくありとあらゆるものごとについて、それまで誰も知らなかったことに特許が許可されるらしい。アルゴリズムだろうがモノだろうがおかまいなし

である。で、特許でひと山当てようとする連中は、誰も知らないモノを見つけようと、未知の国々を放浪して歩く。ミシシッピ大学の研究者が、インドを放浪して見つけたのがターメリック―日本では鬱金（ウコン）と呼ぶ香辛料で、カレーの黄色は、こいつの粉だ―に含まれる外傷に効く薬の成分だった。どの文献を見ても、ターメリックが外傷に効くとは書かれていなかったので、特許を申請、受理されたのだが……インド政府からこれにクレームがついた。インドの人々は、伝統的にターメリックを外傷治療に使ってきたし、外傷に対する治療効果に関する文献もインドにはちゃんとあった。アメリカ人が知らなかっただけだ。

というわけで、この特許は結局取り消されたのだが、これは舞台が特許制度に否定的なインドで、しかも広く知られた効果に関する内容だったのが幸運だった。たとえば、もしこれが文字を持たない第三世界のある部族にだけ知られた事実だったら、その特許は取り消されることもなく、彼らが昔から使ってきた伝統薬を利用したとたん、それは違法行為になってしまう。「もし」と書いたけれど、これは実際に発生していて、「生物学的海賊行為」と呼ばれている。アマゾンの密林に自生する薬草などは、これでかなりやられているそうだ。アメリカの言い分は「論文になっていない」ということなのだが、アマゾンの密林に暮らす人々に論文を書けっていうんだろうか。やっぱり、知的所有権の概念は相当歪んでるとしか思えないのだが、アメリカは自分たちの価値観をWTOなどを通して世界のルールにしようとしている。昨年のWTOシアトル会議が大荒れに荒れたのも当然だろう。

オープンソースは異議申し立てか

知的所有権制度のすべてが不要だとは思わない。しかし医薬品に関する特許を認めないインドでは、薬の価格が特許を認めている国よりはるかに安いこと、海賊ソフトが蔓延する韓国では、技術的な条件がよく似た日本よりソフトの価格水準はずっと低い、といったことの意味を考え直してみてもいい。社会に有益なものの創造を促し、社会を豊かにするために必要な知的所有権もあるが、知的所有権が私企業の巨大な権益を保護する道具でしかないとしたら、それは社会にとって有害な概念だ。

若かりしゲイツ君がBASICをインテルチップに移植していたソフト産業の黎明期に、知的所有権によるソフトの保護には意味があったのかもしれない。しかし、今ではソフトウェア産業は知的所有権の保護がなくても、十分に成立することは、GNUやオープンソースで実証された。ソフトメーカーから猛反発を食らうかもしれないが、保護されなきゃつぶれるような技術なんかいらないと思う人は少なくないと思う。

先日、アメリカでマイクロソフト有罪の判決が出たという。だがアメリカの知的所有権制度の歪みこそ、マイクロソフトをあれほど巨大な企業にした元凶ではないだろうか。アメリカの裁判所は、その前に歪んだ自国の知的所有権制度を地球的な正義と常識に従って修正するべきだった。そうすれば、わざわざマイクロソフトに有罪の汚名を着せる必要もなかったし、それは世界中のプログラマーとPCユーザーに大いに歓迎されたに違いない。

地球的な規模で、歪んだ知的所有権意識への異議申し立てが始まっている。昨今のオープンソースソフトのうねりは、そうした異議申し立ての無意識的な表れなんじゃないだろうか。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ドクターShiodaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shioda

4・6 Netscpae 6.0発表
4・5 Sun StarOffice 5.3β配布
4・1 今年のApril Fool RFCはRFC2795
3・29 シーゲートが買収される
3・28 BeOSの無償ダウンロード開始
3・24 Corel WordPerfect Office 2000 for Linux出荷
3・14 Lineoが日本のユナイテッド・システム・エンジニアを買収

そろそろ日本のボーナス商戦や米国のホリデーシーズンに向けていろいろな新製品が登場する時期ではあるが、ここのところあんまり、購入欲をかき立てるような製品にはとんとお目にかからない。いくらはやりだとはいえ、並んでまでは物を買いたくないしねぇ。だってそのうち山積みだよ。並ばなくても買えるようになるしねぇ。某社は初期ロットは避けるのが鉄則だというし……。

Microsoft和解交渉決裂

注目のMicrosoft対司法省の裁判だが、どうも和解交渉は決裂したようである。となると、昨年すでに、裁判官の事実認定で、「シャーマン法（米国の独禁法のひとつ）に違反している」との判断が行われているため、この裁判では、Microsoftは有罪となり、このあと、具体的な是正措置が言い渡されることになる。もっとも、Microsoftには控訴する権利があるので、おそらく裁判終了後に、控訴するはず。

具体的にどういうことが要求されるのかは、まだ分からず、Microsoftの企業分割から、ビジネスに対する制限（特定分野への参入を禁止するなど）まで、さまざま考えられる。交渉が決裂したのは、司法省側からの要求をMicrosoftが受け入れることができなかったからであり、この是正措置については、司法省などを含めて聴聞会が開かれ、裁判官が決定する。

とりあえず、是正措置が出たとしても、控訴したら、しばらくは裁判が続くことになるので、すぐにMicrosoftのビジネスに影響が出ることはないと思われる。ただ、この訴訟に関連した集団訴訟が115件あり、こちらも問題といえば問題。つまり、最高裁でも有罪となれば、これを根拠とした集団訴訟にも負けることになるからである。このあと、集団訴訟はかなり増えてくることになるかもしれない。

そんなMicrosoftだが、ニューヨークタイムズやUSAトゥデイといった新聞に全面広告を打ち、イメージアップを図る作戦を展開中なのだとか。また最近は、ビルゲイツの言動もおとなしくなったともいう。そういえば、X-Boxのときも、あまり過激な発言は見られなかったようだし、少しは裁判の影響が出ているのかも。Microsoftの株価は75ドルぐらいまで下がり、ビル・ゲイツの持ち分7億8500万株は、10ドル下がれば、70億ドルつまり7000億円（1ドル100円換算）も損しちゃうのである、なんでも、判決当日の損は、1兆円を超えていたのだとか。まあ、これだけ損するなら、謙虚にもなろうというもの。

ブラウザ戦争すでに終結

さて、この裁判の原因ともなったNetscapeのブラウザ、ようやくGekkoエンジンを使ったβ版が登場した。こんどは見た目もずいぶんと違っていて、CSSの表示なんかがまともになった。これで、linuxでもCSSを使ったページがまともに見られるというもの。だが、時期的にはちょっと遅すぎという感じ。ブラウザ戦争自体は、いつの間にか終わっちゃった雰囲気だからである。

同じブラウザでも、Mosaicのライセンスを持ち、IE1や2のベースを提供したスパイグラス（実は、いまのIE5のバージョン情報を見るとまだスパイグラスのライセンスについての記述がある）は、オープンTVに買収されたという。オープンTVは、セットトップボックス向けのソフトウェアの会社で、

AOLやSunなどから出資を受けているところ。もともとは、トムソンとSunの合弁会社としてスタートしたが、トムソンはMicrosoftと提携したために、出資を取りやめている。

まあ、ブラウザを巡る悲喜こもごもといった感じではある。

Sunも提訴を検討中

そのSunだが、反トラスト法違反でMicrosoftに対する民事訴訟を起こすかどうかを検討中だという。まあ、Microsoftのやることはなんでも気にくわないSunなので、司法省との裁判が有罪と決まり、Java関連での恨みを晴らしたいところなのかも。せいぜい、Microsoftから賠償金を搾り取れる程度で、特にJavaコミュニティにとって有利なものになるとは思えないが、Sun、いやマクネリにしてみれば、Microsoftからカネでもとって、日頃のウップンを晴らしたいのだろう。

だが、そのSunにしても、Apache Software FoundationへのXML技術の提供が遅れており、何も出していないのに、パンフレットにApacheに技術提供を行っていると宣伝しまくっていた。これでASF側はやっぱり怒っているらしい。この前のBlackDownとのLinux Java一件といい、どうもSunはあちこちの人を怒らせるのが得意らしい。まさか、SolarisよりLinuxが人気あるので、ワザとやっているなんてことはないだろうねぇ。なにせ、ウップン晴らしに訴訟を考えるような会社だからねぇ。

とりあえず、Javaについては、JCP（Java Community Process）を修正し、IBMやHPを含めたSunとは独立した委員会が、新しい仕様の管理権を持つようにしたみたいである。ただ、一部の会社は、まだライセンスなどについて不十分だと思っているよう。多少は良くなったみたいだけど、まだまだなのかしらね。まあ、Java考案者としての権利を主張したいのはわかるけれども、ここまで広まっちゃうと、ちょっとした言動が大きな波紋となっちゃうので、Microsoftみたいに多少は謙虚になったほうがいいのかも。

世の中不足ばっかり？

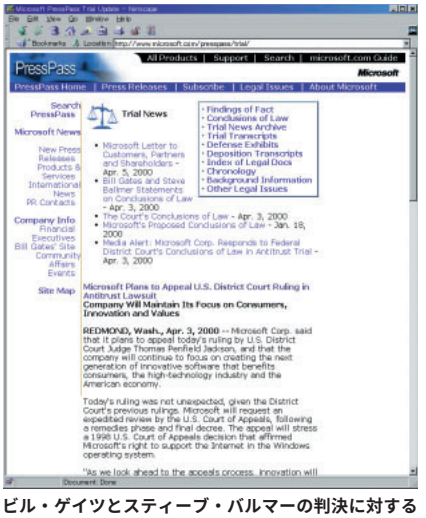
なんでも、任天堂のゲームボーイ・アドバンスの発売が延期になったのだとか。原因は部品不足だという。なんだか昔のスペースインベーダーの頃を思い出す話。あの当時、急にタイトーのスペースインベーダーが流行ったため、TTLなどの部品が不足して、各社苦労したという。筆者も当時、自作のマシンの組み立てで部品入手に苦労した記憶がある。

この部品不足は、携帯電話などの人気が原因だとか。そういえば、携帯やPHSがNTTの加入電話を抜き、なんでも2人に1人は携帯かPHSをもっているのだとか。これじゃあ不足するかもねぇ。

それに、PlayStation2がアッというまに100万台以上も売れるなんてことを考えると、最近の商品は、一度に大量に動くなんてことがザラだからねぇ。もっとも、大量に売れると、ちょっとしたミスが致命的になったりするので、要注意ってこともあるしねぇ。

そういえば、インテルのモバイルPentium IIIなどを搭載した東芝のマシンに不具合が出たのだとか。原因は、インテルの仕様変更がまずかったようだ。AppleのiBookやPowerBookでは、ソフトウェアの不具合により、スリープからの復帰でエラーが出るという障害が発見され、Dellのノートパソコンでもメモリの欠陥により、やはりリジュームから復帰できなくなったという。まあ、これだけノートパソコンが増えてくれば、不具合も増えようというもの。なんでも、イギリスでは、秘密情報部員が機密文書の入ったノートパソコンを盗まれたのだとか。こういうノートパソコンだったら、安全だったのかも。

では、今月はこのへんで。



ビル・ゲイツとスティーブ・バルマーの判決に対する声明
http://www.microsoft.com/presspass/trial/

Corel WordPerfect Office 2000 for Linux。日本語が使えるのはいつの日か?
http://linux.corel.com/products/wpo2000_linux/index.htm

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台
～Mozillaパーティ～

4月の大きな話題といえば、やはりNetscape 6の登場だろう。日刊アスキーLinuxでも、Netscape 6の発表にあわせ、関連記事を多数公開した。そして、このNetscape 6登場の翌日、都内でMozillaに関わる人たちが集まったパーティが催された。今回は、このパーティの報告をお届けする。

米国からも国際電話で挨拶

Netscape 6が発表された翌日の4月8日、日本のMozillaコミュニティが集まり、「mozilla.party.jp 1.0」と題されたパーティが行われた。

パーティのオープニングは、日本におけるNetscapeの顔ともいえるNetscape Communications社・瀧田氏の「(Netscape 6) PR1もみなさんのおかげということで……」という挨拶から始まった。滝田氏はNetscape Communicator日本語版を実際に作ってきた人でもある。

そのあとはNetscape国際化部門マネジャー・Bob Jung氏が国際電話で日本の会場に挨拶。どんどんMozillaを勉強して、アイデアを出していってほしい、Mozillaの国際化についても、意見をいってほしいという旨の話をされた。Bob Jung氏は、Mozillaに関する「夢」を持っているのだそうだ。その夢とは、たとえばルビのサポートや、フォント設定ダイアログボックスでのフォント見本のサポートなどだ。これらの夢は、http://www.gimlay.org/~andoh/mozilla/bobj.htmlで見ることができる。

開発者の苦労

続いてセッション1が始まった。MozillaZineという、Mozilla関連ニュースサイトの日本語訳を行っている神部氏によるMozillaの概要説明では、Mozillaの歴史からはじまり、Gecko搭載によるメリット(軽量/XUL)などの紹介が行われた。神部氏の発表中、特に会場が沸いたのが、電卓のデモンストレーションである。Mozilla上でCSSやXMLを使えば、比較的簡単にアプリケーションを作れることを実証するデモンストレーションだ。これは、Windowsの電卓とまったく同じものを、XULで作ってしまったというもの。作製者は“いけも”さんである(**画面1**)。

次に紹介されたのがBugzilla。BugzillaはMozillaのBTS(Bug Tracking System)で、Mozillaのバグを管理するシステムだ。バグの検索や対応状況、報告などを行うことができる。これは本家(英語サイト)で動作しているわけだが、このほど登場したのが日本語版Bugzillaだ。現在はまだ試験運用中だが、ここではMozillaそのもののバグだけではなく、日本語パックに関するバグも扱うことになっている。

続いてMozilla日本語パックの山本和彦氏、古川良一氏による発表があった。

写真1　MozillaZineの神部氏

写真2　当日、入場券代わりに配られたステッカー

このセッションでは、Mozillaの国際化の仕組みや、日本語化の苦労、ツール「Mozilla Translator」の紹介、そして今後の計画などが語られた。特に日本語化の苦労では、言語に関するリソースがあちこちのディレクトリにまたがっており、それらを別々に管理していたら大変であること、そして、そうした管理を助け、ローカライズを簡単にするためのツール「Mozilla Translator」が登場し、作業が楽になったことなどが語られた（このツールがなければとてもではないが作業ができなかったそうだ）。

そして、会場を沸かせたのがMozillaのスキン機能による外観の変更だ。Netscape 6の登場時も大きく報道されたが、Mozillaも同様にその外観を取り替えることができる（MP3プレーヤのスキンと同じ）。この機能はNetscape 6では「テーマ」と呼ばれている。当日、古川氏がMozilla用のスキンを持参。これがまたStar Trekライクなもので、画面左上には「U.S.S. Mozilla Navigator」というロゴ（？）が入っている（**画面2**）。今後の予定としては、せっかくリソースを自由に変更できるのだから、「赤ちゃん言葉バージョン」や、「大阪弁バージョン」も、作ってみたいとのことだ。

セッション1最後の発表は、TAM猫氏、money氏、DJ MOJI氏によるmozilla.orgの和訳についてだ。2000年1月から始めて、現在までに65件の文書を公開している。参加しているのは18名で、すべてボランティアベースでやっていて、現在翻訳者を募集中である。

ソースコードにがっかり？

このあとはセッション2に突入。まずは実際にMozillaのコーディングをしている加藤氏が登場し、Mozillaの内部構造を解説。なぜ、どこで文字化けが起こるのか、といった話題をやさしく説明してくれた。公開に踏み切った当初のNetscape Communicatorのソースコードを見てがっかりした……というドキッとする逸話や、コーディングしている人ならではの「この部分がくせ者」的な話題が飛び出し、パーティ中もっとも濃い内容の発表だったのではないだろうか。

加藤氏のあとは、サン・マイクロシステムズでMozillaのXIMを修正している片貝氏が登場。片貝氏もMozillaのコーディングをされていて、日本語入力の方法などの改善をしている。片貝氏の要望としては、UNIXの場合はプラットフォームが多く、自分がテストできる環境も限られているので、いろいろな人にMozillaを使ってもらいたい、とのことである。

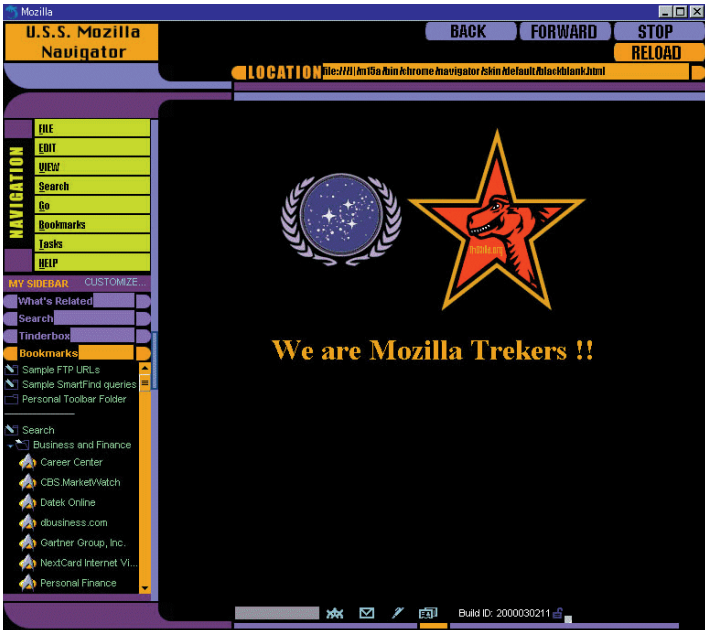
●

発表のあとはパネルディスカッションからピザパーティへと流れたわけだが、非常にフレンドリーかつ内容豊富な手作りイベントであったといえよう。なお、ここで紹介した各プロジェクトは、安藤幸央氏のパーティ報告 Webページ（http://www.gimlay.org/~andoh/mozilla/）から参照できる。

この号が出る5月上旬には、Mozilla Milestone 15（M15）が公開され、次いで日本語パックも公開されている予定である。

（日刊アスキー Linux　吉川大郎）

画面1　Windowsに標準で付属する電卓（左）とXUL電卓（右）
笑えるほど似ている。XUL電卓は、いけもさんのWebページ（http://www.wakaba.toyonaka.osaka.jp/~ikemo/）で手に入れることができる。



画面2　Star Trek風Mozilla
画面左、「My Sidebar」の中の、アイコンの形状にも注目だ。

正しきタコへの道

初級Linuxer養成講座

第9回　ひとり管理者の心得（4）～アクセス権とは

マルチユーザー環境を前提に作られたLinuxでは、複数のユーザーの間でファイルを共有したり、故意に内容を隠したりできるように、ファイルの「アクセス権」というものが存在する。前回説明したファイルの所有権（UIDとGID）と同様に、個人的に使うマシンではあまり意味のないものだが、避けて通ることはできないのが現状だ。

文：竹田善太郎
Text : Zentaro Takeda

この原稿が雑誌の形になるころには、全国どこに行っても渋滞だらけの、**悪夢のようなゴールデンウィーク**も終わって、「五月病」という言葉がぴったりの時期になっていることだろうが、原稿を書いているのは4月の初め、いわゆる「新学期」シーズンである。この時期になると、書店の一等地に山積みになるのが、語学関連の参考書や放送講座のテキストなどだ。筆者も、中学生のころから、NHKの語学講座を聴取しようと**試みた**ことが何度となくあるのだが、恥ずかしながら1年間（あるいは半年間）欠かさず聴き続けられたことはほとんどない。毎朝（あるいは毎晩）決まった時間にラジオを聴くというのは、**とんでもない苦行**であるし、留守録音するとしても、テープのセットを忘れたり、ラジカセのタイマー機能がうまく動かなかったりして録音に失敗し、そのまま**いやになって中断**してしまう、ということが多かった。

Linux magazineでこの連載を始める前に、某雑誌でも似たようなテーマの連載を書かせてもらっていたのだが、その中で**Linuxを使った語学番組自動録音システム**のアイデアに触れたことがある。しばらくは忙しくて放っておいたのだが、ちょっと空いた時間があったので、フリーで配布されている音声関係のソフトウェアを試して回って、このたび**全自動家庭内語学講座聴取システム**を動かすことに成功した。

こう書くとなんとも大げさだが、仕組みは他愛のないもので、Linuxマシンのサウンドカードの Line入力端子につないだAMラジオの音声を、/dev/dspデバイス経由でWAVファイルに保存し、さらにそれをMP3形式にエンコードしてハードディスクに蓄えるというものだ。番組の長さに応じて一定時間で録音を中断できるアプリケーションを見つけるのに苦労したが、Andrew L. Sandoval氏の mpegrec 1.0（wavrec）というフリーソフトがこれを解決してくれた。あとは、簡単なフロントエンドプログラムをPerlで書いて、これをcronで自動的に起動するだけである。ntpdを使ってLinuxマシンのシステムクロックを標準時に合わせておけば、**時報ぴったり**の気持ちよい録音を残すことができる。

さらに、CGI用のスクリプトを書いて、録音済みのファイルにWebページからアクセスできるようにすれば、家庭内LAN経由で家族の誰もが、自分の好きな時間に語学番組を再生できるようになる（**画面1**）。20分番組1回分の録音データは、64kbpsのモノラルデータで10Mバイトに満たないので、3Gバイト程度のハードディスクがあれば、**1年分の番組もまるごと保存**しておける。仮に、カセットテープで録音を保存したとすると、120分テープを使ったとして1週間（6回）で1本、年間だと50本ものテープが必要になる計算だ。場所はとるし、テープ代も馬鹿にならない（1本100円としても5000円。ちなみにHDD 3Gバイト分のお値段は、20GバイトクラスのHDDを使ったとすると2000円くらい）。なにより、50本もあるテープから目的の日時の番組を見つけるのは、ものすごく面倒くさい。HDD上のファイルなら、検索の手間もかからないし、思い立ったときに即座に再生できる。

また、ポータブルのMP3プレイヤーにダウンロードすれば、家事や通勤通学の最中に聴くのも自由自在だし、自宅のサーバに外出先からダイヤルアップ接続できるようにしておけば、長期の旅行中でも聞き逃すことはない。

WAVファイルからMP3ファイルに変換するときに、ダウンロード用の**軽いデータ**も一緒につくっておけば、転送の時間も節約できる。

ここでは詳しく説明する余裕はないので、近いうちに別の記事で紹介したいが、2週間ほど使ってみたところ、予想以上に使い勝手がよく、手放せなくなってしまった。このシステム専用にするために、新しいLinuxマシンをもう1台組もうかとまで考えている。語学雑誌などの広告を見ると、ハードディスクを使った自動録音機能つきのラジオも販売されているようだが、とても高価（9万円近い）である。これなら、PCをもう1台買ってもたいした違いはないし、すでにLinuxを動かしているマシンがあるのならそれに兼用させることもできる。

無人の自動留守録音は、WindowsやMacintoshがもっとも苦手とする処理なので、Linuxマシンを**超多機能全自動ラジカセ**として使うのも悪くない。

面倒なアクセス権

さて、話を無理やり本題に戻すが、前述の自動留守録システムで、録音済みのMP3形式データをWebページからアクセスできるようにしたとき、いちばんやっかいだったのが**ファイルのアクセス権**の設定だった。外部からは接続不可能な家庭内LANでしか使わないものなので、ファイルのアクセス権などどうでもよいと軽く考えていたのだが、最近のLinuxに付属するWebサーバは、いわゆる**セキュリティ設定**がかなり厳しくなっていて、公開しようとしているデータのあるディレクトリが**だれでも自由にいじれる設定**になっていると、逆にWebブラウザでは**だれも見ることができない**状態になってしまうのだ。

もちろん、Apache、あるいはディレクトリのアクセス権のどちらかの設定を変えれば問題は解決するのだが、**アクセス権とはどんなものか**を知っていないと、トラブルの原因がどこにあるか判断することすら難しいだろう。

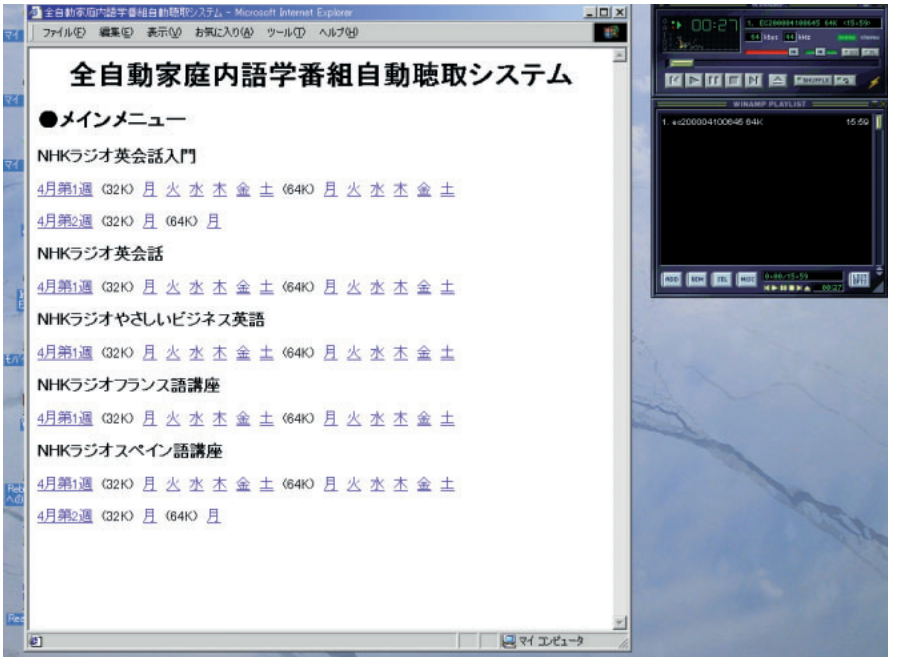
個人が勝手きままに使っているマシンでこのようなアクセス権というものがあると、自由度が低くて使いづらいし、さまざまな**トラブルの原因**になってしまうこともあるのだが、逆に、アクセス権があるおかげで、OSを構成する**重要なファイルを間違って削除**してしまう危険性も（まったくないというわけではないが）少なくなっているという事実もある。

前回、UIDについての話題の中で、いろいろな制限が面倒なので日常的にrootユーザーの権限でLinuxを使っているユーザーも多いのでは、という趣旨の話をしたが、後述するように、rootユーザーの権限では、アクセス権によるトラブルの防止機能がまったく**無効**になってしまうので、できればそのようなことはしないほうがよいということなのだ。

現在、LinuxのユーザーIDやファイルのアクセス権設定などをもう少し柔軟にして、たとえば、rootユーザーでなくてもシステムやアプリケーションの設定を変更できるようにする方法が開発されているようだが、現状のLinuxでは、アクセス権との付き合いを避けることはできない。

3＋1種類の「だれ」

Linuxのアクセス権を理解するとき、「**だれ**」が「**なにを**」しようとしているのかということを常に意識するとよい。まるで**国語の教師**のようなことをいっている



画面1　「全自動家庭内語学番組自動聴取システム」の実行例
世間では、MPEG-1やMPEG-2形式でテレビ放送を録画することが流行っているようだが、現状のパソコンやLANの性能では、このような「ラジオ・オン・デマンド」システムのほうが身の丈に合っている気がする。調子に乗って、フランス語やスペイン語の講座までメニューに加えてみた。MP3形式だと、再生の手段を問わないのがありがたい。

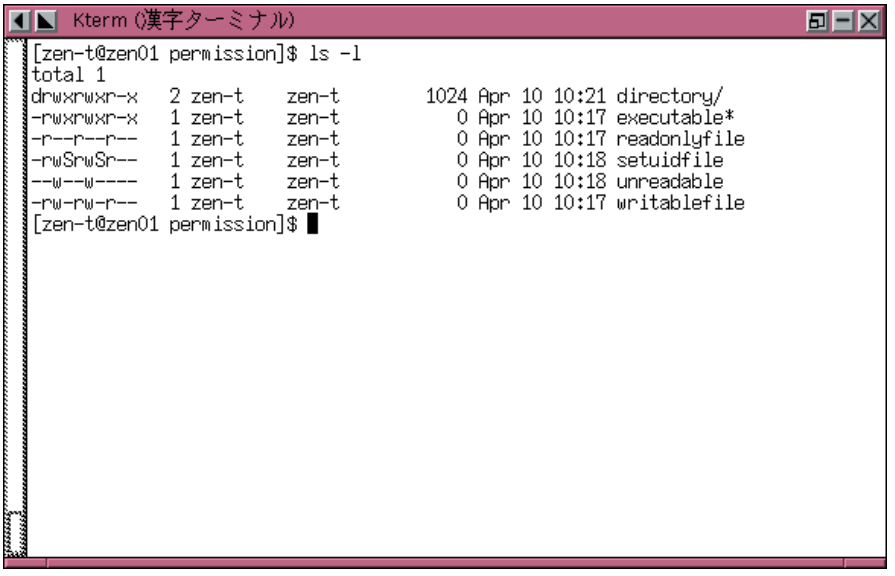
が、ファイルのアクセス権に限らず、コンピュータのセキュリティ技術一般を理解しようとするのなら、このことを常に意識においていないと、とんでもない間違いをすることもある。もっと難しい言葉のほうがしっくりくるという変わった人なら、「だれ」は**人格**、「なにを」は**行為**というように置き換えてもよいだろう。

アクセス権における「だれ」（人格）は3種類ある。まず最初は、アクセスの対象となるファイルを作成したユーザー、すなわち、そのファイルの**所有権をもつユーザー**である。前回説明したように、ファイルにはそのファイルの所有者を表わすUID情報が名札のようにつけられていて、Linuxはそのファイルにアクセスしようとしたユーザー（あるいはプロセス）のUIDと、ファイルにつけれらている名札のUIDを比較して、それが同一だったら「所有者本人がアクセスしようとしている」と判断する。

2番目の人格は、ユーザーIDではなく**グループID（GID）が同一**のユーザーである。各ファイルの「名札」には、UIDと一緒にGIDも記述されていて、そのファイルにアクセスしようとしたユーザー（プロセス）のGIDがそれと同一だったら、「同一グループに属するユーザーからのアクセス」と判断する。

最後の3番目の人格は、UIDもGIDも違う**その他大勢**に分類されるユーザーである。

もうひとつ、これら3種類のどれにも当てはまらないのが、絶対権力をもつroot（管理者）ユーザーである。rootユーザーはいわば**神様**のようなもので、ファイルの所有者のUIDやアクセス権がどのようになっていようとも、それに関係なく、なんでもできることになっている。rootユーザーとしてログインした状態では、ファイルの所有権やアクセス権は**なんの効果もない**ということになる。たとえば、**書き込み不可**に設定されていたファイルを書き換えてしまったり、削除してしまうような操作も、**なんの障害もなく**できてしまうのだ。rootの権限で作業するときは細心の注意が必要だといわれるのは、このような理由からだ。



画面2　ls -lコマンドの実行例
ここでは、いろいろなアクセス権設定がされているファイルを表示してみた。各ファイルの行の左端がアクセス権設定の内容である。「rwx」あるいは「rwS」という3文字が1組になって3組並んでいて、左から、「所有者」、「同一グループユーザー」、「その他大勢」に対するアクセス権を表している。「unreadable」ファイルは、書き込みはできるが読み出しはできない設定（-w-）になっているが、このような設定はあまり見かけることはないだろう。

3種類の「できること」

Linuxの**ファイル**の実体は、ハードディスク（あるいはその他の記憶装置）上に記録されたデータの固まりであるが、それに**アクセスする**という動作、すなわち**「できること」**（行為）は、3つの種類に分けることができる。**ファイルの読み出し**と**ファイルの内容の書き換え**と**ファイルの内容をプログラムとみなして実行する**という3種類の動作である。ファイルのアクセス権では、これらの3種類の動作それぞれについて、その操作を認めるか否かを設定できるようになっている。

ファイルの読み出し権限については、英語の“ read ”という単語の頭をとってr権限、書き換えは“ write ”のW権限、実行は“ execute ”の2番目の文字X権限と表記される。この3つの文字の意味は覚えておいたほうが便利だろう。

■ディレクトリを「実行」するとは？

Linuxの「アクセス権」は個々のファイルだけでなく、**ディレクトリそのもの**にもそれぞれ設定できるようになっている。実際には、Linuxではディレクトリも単なるファイルの一種とみなされるのだが、たとえば、ディレクトリの読み出しができないような設定になっていれば、そのディレクトリ中のファイル一覧をlsコマンドなどで表示させることは不可能になる。ただし、ファイルそのものの読み出し

権限が“可”になっていれば、アプリケーションなどでのファイルの読み出しはできる。書き込み権限については、書き込み不可になっている場合、そのディレクトリへの新しいファイルの作成ができなくなる。

ところで、もうひとつの権限である「実行」についても、ディレクトリに設定することができる。しかし、**ディレクトリを実行する**といっても、まったく**意味不明**だろう。相変わらず世間に根強く存在するコンピュータを敵視する人たちが、**これだからコンピュータ用語は困る**などと、格好の攻撃目標にしそうな表現である。

難しくいうと、Linuxにおけるディレクトリの「実行」権限は、「そのディレクトリをパス名に含んだファイルの参照を許可するかどうか」を設定するためにある。これでもやはり意味不明だと思うので、もっと平たく言い換えれば、そのディレクトリより下にあるファイルやディレクトリにアクセスできるようにするかどうか、ということになる。つまり、ディレクトリの実行権限が「不可」になっていると、そのディレクトリの中、およびさらに下のサブディレクトリ中にあるファイルは**使えなくなる**と思っていればよい。また、ディレクトリの実行権限が不可の場合、**cdコマンド**でそのディレクトリに**移動できなくなる**。

特別な場合を除いて、ディレクトリの「実行」権限を「不可」にすることはない。逆に、後述するchmodコマンドを使う場合、まちがってディレクトリの実行権限を不可に設定してしまうと、そのディレクトリの下のファイルにアクセスできなくなって困ったことになる場合がある。**ディレクトリの実行権限は常に「可」**と覚えておいたほうがよいだろう。

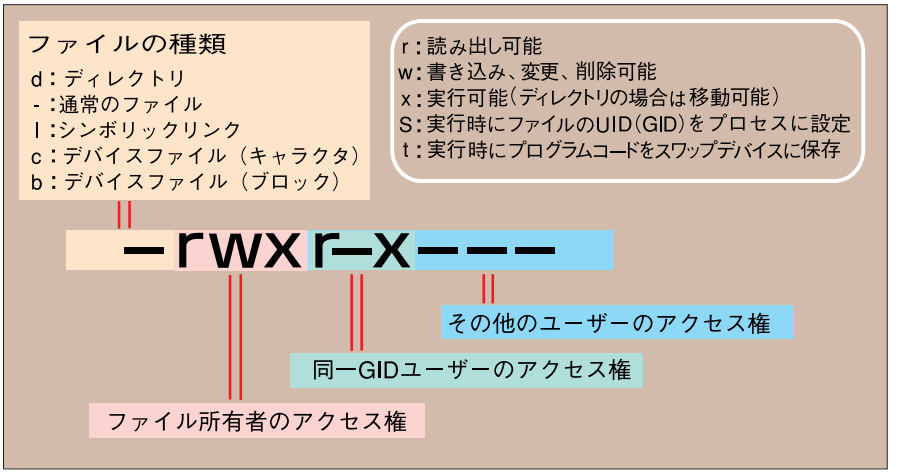


図1　アクセス権の意味
ls -lコマンドで表示されるアクセス権は、図のように「所有者」、「同一GIDをもつユーザー」、「その他大勢」ごとに表示されている。chmodコマンドによるアクセス権の設定では、これとは異なる形式で記述する必要があるが、「r」「w」「x」「s」「t」などの文字の意味は同じである。

アクセス権の表現方法

ファイルやディレクトリのアクセス権がどのような設定になっているかを知るには、ファイルの一覧を表示する**lsコマンド**を使えばよい。lsコマンドに**-lオプション**をつけて実行すれば、ファイルのアクセス権、UID、GID、サイズ、更新日時などの情報がすべて表示される（**画面2**）。

ls -lコマンドの出力は、左からアクセス権、リンク数（これについては、いずれ説明したい）、UID（ユーザー名）、GID（グループ名）、サイズ、更新日時、ファイル（ディレクトリ）名の順になっている。ここでは、一番左側に表示されるアクセス権の部分にだけ注目してみよう。ファイルのアクセス権は、大体次のように表示されているはずだ。

```
-rw-r--r--
```

また、ディレクトリについては、次のようになっているものが多いだろう。

```
drwxr-xr-x
```

あるいは、

```
drwxrwxrwx
```

となっていることもあるかもしれない。

最初の「d」の部分は「ディレクトリである」ことを示しているのだが、その他のr、w、xなどの文字は何を意味しているのだろうか。これは、先ほど述べた「read」、「write」、「execute」の権限があることをそれぞれ示しているのだ。「-」（ハイフン）になっている部分は、その権限が「ない」ことを示している。

たとえば、「rwx」という表現は、読み出し、書き込み、実行のすべての権限が許可されていることを示している。「r-x」だと、読み出しと実行はできるが、書き込み（変更）はできないことを示す。「---」だと、読み出しも書き込みも実行も一切できない、つまりそのファイルを**使うことが認められていない**ことを表わしている。

「rwx」の3文字の組み合わせが3組並んでいるのは、それぞれ左から、「所有者」、「同一グループ」、「その他大勢」に対して、どのような権限が設定されているかを示しているからだ（**図1**）。ここで、最初に述べたような「**だれ**」が「**何をできるか**」という考え方が役に立つのだ。

たとえば、「`rw-r-----`」というアクセス権設定の場合、そのファイルの所有者（UIDがファイルのUIDと同一のユーザー）は、ファイルの読み出しと書き込みが可能、グループIDが同一のユーザーについては、読み出しのみが可能、その他大勢（UIDもGIDも、ファイルに設定されているそれとはまったく異なるユーザー）については、一切の使用ができないということになる。

ファイルの操作をしていて、

```
Permission denied
```

というエラーメッセージ（「アクセスが許可されない」という意味）が表示されたら、とりあえず「ls -l」コマンドを実行して、アクセスしようとしたファイルのアクセス権がどのようになっているかを調べる。まず、ファイルのUIDとGIDの部分を見て、自分のUIDやGIDと同じかどうかを調べ、次にアクセス権の部分を見て、自分のUIDやGIDでどのようなアクセスが認められているのかを調べる。たとえば、そのファイルに書き込みをしようとして怒られたのであれば、自分がそのファイルに書き込める権限があるかどうかを調べるのだ（**画面3**）。

自分が作成したファイル（ファイルのUIDが自分のUIDと同一）で、書き込みを拒否されたのであれば、ファイルのアクセス権の左端は、

```
-r--
```

となっているはずだ。それが、内容を書き換えてはいけないファイルなら、自分の操作が間違っていたということになるのだが、書き換えてしまってもかまわないファイルなら、アクセス権の設定を変更して、そのファイルへの書き込みができるようにしなければならない。このときに使うのが、最後に説明するchmodコマンドだ。

画面3　「Permission denied」エラーに対処する
書き込み不可（「r--r--r--」など）に設定されているファイルに書き込みをしようとすると、Permission deniedというエラーメッセージが表示されることがある。このような場合は、lsコマンドでアクセス権設定を調べてから、後述するようなchmodコマンドでアクセス権を変更する。ただし、chmodコマンドでアクセス権を変更できるのは、自分が所有権をもっているファイルだけである。他人の所有権が設定されているファイルの場合は、いったんrootユーザーになってアクセス権（あるいは所有権）を変更しなければならない。

chmodコマンド

chmodコマンドは、ファイルやディレクトリのアクセス権の設定を変えるコマンドである。前回説明したchownコマンドと組み合わせて使われることの多いコマンドなのだが、使い方はちょっと難しい。chmodコマンドでは、「**だれ**」に「**なんの権限を**」を「**与えるか奪うか**」という形式でアクセス権を設定をする。

chmodコマンドの詳しい使い方については、本誌の先月号の特集（79ページ）でも解説されているが、簡単にいえば、

`chmod [設定するアクセス権の内容] [ファイルあるいはディレクトリ名]`

という形式でファイルやディレクトリのアクセス権を変更する。

chmodコマンドがやっかいなのは、アクセス権の指定方法がわかりづらい点だ。たとえば、

```
# chmod rwxXstugo file1.txt
```

などというコマンドラインを見ても、大半のユーザーはちんぷんかんぷんだろう。このコマンド例は、chmodのオンラインマニュアルにも出てくる例だが、「rwxXstugo」の部分が設定する実行権限の内容を表わしている。このアクセス権設定の意味は、「ファイルの

所有者、同一グループユーザー、その他のユーザーに対して、read、write、executeの権限を与えるが、execute権限についてはいずれかのユーザーに対して、すでにexecute権限が与えられている場合か、ディレクトリファイルに対してだけ設定し、ファイルの実行時にはプロセスのUIDとGIDをファイルのUIDとGIDに設定し、プログラムコードをスワップファイルに書き込む」ということになる。chmodコマンドは、**かくも詳細な設定ができる**ということを誇示するための見本なのだが、初心者のユーザーはここまで複雑な設定は必要ないだろう。

個人的に使っている Linux マシンなら、ファイルの所有者、すなわち**ユーザー自身がそのファイルにアクセスできるかどうか**を設定できれば、当面は十分間に合う。このような設定を行うなら、

```
chmod +rwx ファイルやディレクトリ名
```

あるいは、

```
chmod -rwx ファイルやディレクトリ名
```

という表現だけ覚えておけば十分だ。＋はそれ以下のアクセス権を**追加**するという意味、－はアクセス権を**削除**するという意味になる。**rwx**の部分は、**必要なアクセス権だけを残す**ようにする。たとえば、「foo.txt」というファイルを、書き込み不可（書き込み可能権を削除）にするには、

```
chmod -w foo.txt
```

と入力する。

再び書き込みできるように変更したければ、

```
chmod +w foo.txt
```

とすればよい。mycommandという名前のシェルスクリプトファイルを作成して、それをコマンドラインから直接実行できるようにしたい（実行権限を追加したい）場合には、

```
chmod +x mycommand
```

とすればよい。ちなみに、ディストリビューションや個人の設定によって異なる場合もあるが、このような方法でchmodコマンドを実行すると、**ファイルの所有者と同一グループユーザーについてのアクセス権のみ**が変更される。「その他大勢」のユーザーの権限は元のまま変更されない。その他大勢のユーザーのアクセス権を変更したい場合は、「others」という意味の**o**をアクセス権設定の前につける。たとえば、foo.txtを「その他大勢のユーザー」も書き込みできるようにするには、次のようにすればよい。

```
chmod o+w foo.txt
```

同様に、ユーザー自身に対する設定を変えたければ「user」の意味の**u**を、同一グループユーザーに対する設定については、「group」の意味の**g**を、すべてのユーザーに対する設定については、「all」の意味の**a**をそれぞれ追加すればよい。

正しいアクセス権の設定は？

chmodコマンドの使い方は理解できたとしても、ファイルのアクセス権はどのようにしておけばよいのか考え込んでしまう読者もいるだろう。これについては、前回のUIDやGIDでも述べたように、個人が自分専用に使っているLinuxマシンに関しては、あまり難しく考える必要はない。

アプリケーションをインストールする場合には、付属するドキュメントの指示に従って適切な設定をしなければならないこともあるが、それ以外のファイルについては、**間違って消したり壊してしまって困るものは、書き込み不可（-w）に、ディレクトリやスクリプトファイルなどは、実行可能（+x）**にすることだけ覚えておけばよい。読み出し不可（-r）の設定については、複数のユーザーで使っているマシンでは重要になるが、個人専用のマシンで使う意味はあまりないだろう。

ところで、筆者が以前勤務していたある雑誌の編集部では、編集作業用のUNIXワークステーションを複数の編集者で共用していたのだが、そこでは、原稿のテキストファイルを作成したら、

```
chmod 777 genko.txt
```

を実行して、すべてのユーザーに対して「読み書き実行」すべてを可能な状態にする、という決まりがあった。ずいぶん乱暴な方法のようにも思えるが、ファイルのアクセス権のせいで、ほかのメンバーが必要なときにファイルを読み出せなくなったりすると困るという配慮から、このようなことをしていたようだ。ちなみに、「777」というのはchmod コマンドのアクセス権設定方法の一種で、「a+rwx」と同じ意味である。

かくして、ファイルのアクセス権というものは、不特定多数の人間がアクセスするようなマシンでは重要なのだが、個人や限られたユーザーだけが使うマシンにおいては、ちょっと厄介な存在なのである。

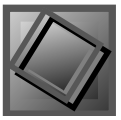
VMware for Linux 2.0を使おう！

Linuxの上でWindowsが動作するPCエミュレータがバージョンアップ

文：塩田紳二
Text :Shinji Shioda

LinuxがパワフルなOSであることはみなさんもよくご存じだろう。しかし、Linuxだけですべてを済ませるのが難しいことも知っているに違いない。たとえば、Microsoft WordやExcelのデータファイルを受け取ってしまったときなど、多くのLinuxerが途方にくれているに違いない。そんなときに威力を発揮するのがVMwareだ。VMwareを使えば、Linuxの上にWindowsを始めとするいろいろなOSをインストールできてしまう。

本稿では3月にバージョンアップして、よりパワフルになったVMware 2.0を紹介する。



VMwareとは

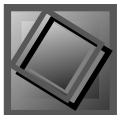
VMware（**画面1**）とは、VMware社の開発した仮想マシンを実現するソフトウェアである。LinuxやWindows NT／2000の上で動かし、そこに仮想的なPCを実現、その中で、WindowsやLinuxなどのOSを動かすもの。こういう説明を聞くとなんだかよくわからないもののように思えるが、早い話、OSの中でもうひとつOSを動かすことができるわけで、たとえば、Linuxの上でWindows 98を動かして、Windows用のアプリケーションを使うなんてこともできる。あるいは逆に、Linuxの上で、複数のLinuxディストリビューションを動かすなんてことも可能。この方式がいいのは、仮想マシンは元の環境（VMwareではホストOSという）から見れば、1つのプロセスにしか過ぎないので、環境を切り替える手間がいらないことだ。その反面、仮想マシンを実現するためのオーバーヘッドで、多少実行効率が悪くなるが、昨今のCPUパワーが余り気味のマシンでは「どおって」ことはないだろう。

VMwareには、現在、Linuxの上で動くVMware for Linuxと、Windows NT／2000の上で動く、VMware for Windows NT and Windows 2000の2つがある。製品形態としては、パッケージ版と、バイナリのみをダウンロードし、ライセンスを購入する電子配布版の2種類があり、さらに、コマーシャル版と、ホビイスト、学生向けのノンコマーシャル版がある。どちらも、VMware社のサイトから購入することが可能。なお、電子配布版では、購入後、電子メールでライセンスファイルが送られてくる。また、ゲストOSとしてSuSE LinuxやTurboLinuxのディスクイメージが最初から付属（このディスクイメージを使うと、インストール

画面1　VMware for Linux 2.0の画面
Linuxの中で別のLinuxディストリビューションが動作している。

作業なしにすぐ利用できる）した版も用意されている。

また、VMwareは、30日間の評価利用が可能である。これは、VMwareのWebサイトからバイナリをダウンロードして、評価用のキーをもらう。こちらは、メールアドレスなどを登録するだけで、すぐ、評価用ライセンスファイルが送られてくる。



VMwareによるPCエミュレーション

VMwareに似たものとして、LinuxではWindowsのエミュレータであるwineや、DOSのエミュレータDOSEMUなどがあるが、これらと違うのは、VMwareが、PCというハードウェアを仮想化している点。つまり、VMwareを動かすと、BIOSの起動からPC/ATというハードウェアがエミュレーションされるのである。

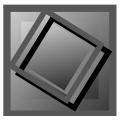
ただし、ハードウェアのエミュレーションという点では、VMwareは完全に動作中のマシンそのものをエミュレーションするわけではない。たとえば、USBについては、それが装備されていて、しかもホストOSがサポートしていたとしても、VMwareの中で動くOS（ゲストOS）からは、いまのところ扱うことができない。現在のバージョンではハードディスク、CD-ROM、フロッピーディスクなどのデバイスとインターフェイス、シリアル、パラレルポート、サウンドカード、ビデオカード、ネットワークカードなどがエミュレーションされる。このうち、ビデオカードとネットワークカードは、仮想的なデバイスが用意され、動作しているマシンのハードウェアをエミュレーションしたり、仮想化するのではなく、まったく架空のデバイスが作られ、それを使う。というのは、ビデオカードでは、低レベルまでエミュレーションを行うと非常に効率が悪いため、各種OS用にデバイスドライバを作り、これに対するコマンドをVMwareで解釈して描画を行わせているのだ。こうすることで、実用上問題のない速度での表示が可能になる。

ネットワークの場合、ゲストOSとホストOSで通信を行うことができなければ、環境として非常に使いにくいものになる。そのためには、それぞれが個別のネットワークアドレスを持たねばならない。そこで、VMwareでは、各仮想マシンが仮想的なイーサネットデバイスを持ち、独自のMACアドレスを持って動作し、個別にIPアドレスなどを持つようになっている。

なお、ハードディスク、CD-ROM、フロッピーディスクやシリアル、パラレルポート、サウンドカードは、VMwareがホストOSの機能を使って、実デバイスへのアクセスを行う。ただし、ハードディスクに関しては、別パーティションをゲストOSから直接操作させるように設定することが可能で、こうすることで、ハードディスクのエミュレーションを行う必要がなくなるため、動作効率が良くなるようである。考えようによっては、サウンドカードも仮想化したほうがいいかもしれないが、逆に、サウンドカードにはさまざまな種類があり、低レベルでは仮想化した機能のマッピングがかなり大変になる。しかも、それをDMAなどの低いレベルで行いつつ、かつリアルタイムに行うのは困難である。このため、単なる実デバイスの仮想化にしているのだと思われる。実デバイスの仮想化は、コンテキストに合わせて、レジスタの待避や復帰などで実現可能である。

なお、このVMwareは、専用のBIOSイメージを持っており、BIOS自体も仮想化してある。というのは、BIOSは、通常、リエントラントになっていないため、リアルモードの実アドレスで作業用メモリを参照していること、また、過去との互換性を取るために、実際のBIOSはスタックサイズの調整を行っているからである。だいたい、いまでもDOS全盛期のソフトウェアがそのまま動くのがPC/ATであり、中には、ファンクションコールをフックしているものや、公式には公開されていない内部ルーチンを使うものなどもある。そうしたソフトウェアから、ハードウェアの差違を見せないようにしたうえで、互換性を持たせているのだから、中身がぐちゃぐちゃなのである。それをそのまま仮想化しているのが、Windows 9x系であり、このために専用の仮想マシンを内部に持っている。VMwareでは、この部分を仮想化するのは困難として、専用のBIOSイメージを持っている。このため、起動時にはマシンの種類に関わらず、PhoenixのBIOS起動メッセージが表示される。

では、実際にVMwareをインストールし、Linuxの中でWindows 98 Second Editonを動かしてみることにしよう。



VMwareのインストール

とりあえず、ここでは、評価版をベースに入手からインストールまでの解説を行う。ただし、ライセンスキーをVMware社のWebサイトにあるVMware STOREから購入すれば、そのまま製品版となり、基本的にはほとんど差がない。また、実際上も、評価版は試用期間が設定されているだけで、機能的には違いがない。

まずは、VMwareをダウンロードする（**画面2**）。URLは、http://www.

vmware.com/download/download.htmlである。ここからLinux版、Windows NT／2000版を選んでファイルを取得する。Linux上で動くVMware for Linuxには、tar＋gzip版（tar.gz）と、Red Hat系で使われるRPM形式の2種類がある。RPMが利用できるディストリビューションを使っているなら、RPM版のほうが作業がちょっとラクである。VMware for Linuxの最新版は今年の3月に出たVMware 2.0。ファイルサイズは約6Mバイトである。

必要なファイルをダウンロードしたら、rpmコマンドもしくはtarコマンドを使って、配布イメージをディスクにインストールする。なお、VMwareでは、ホストOSのファイルシステム上のファイルを使って、仮想マシンのハードディスクをエミュレートするため、すくなくとも、ゲストOSをインストールできるだけの空き領域が必要になる。Linux版では、この領域は各ユーザーのホームディレクトリ以下に作られるため、/home以下にかなりの空きが必要となる。デフォルトでは、2Gバイトのハードディスクをエミュレートするようになっている（ただし、ゲストOSごとの設定で変更は可能）。たとえば、Windows 98あたりでは、インストールを行うのに400Mバイト前後は必要になり、これに加えて、スワップ領域やアプリケーションのインストール領域などを考えると、最低でも1Gバイト程度は必要と思われる。

rpmコマンドやtarコマンドを使ったインストールは、rootにて行う。rpmコマンドの場合、VMwareの実行ファイルは、/usr/bin以下にインストールされるほか、/usr/lib/vmwareにもファイルが置かれる。ドキュメントなどは、/usr/doc/vmwareに、ヘルプファイルは、/usr/lib/vmware/helpにある。

画面2　VMwareのダウンロードページ
VMware for Windows NT and 2000、VMware for Linuxがダウンロード可能だ。

画面3　評価用ライセンスキーを申し込む
e-mailアドレスなどを登録すれば30日間、無料で試用ができる。

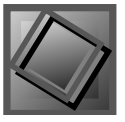
ファイルのインストールが終了したら、次に、/usr/bin/vmware-config.plを起動する（これは、Perlのスクリプトファイル）。ここで、機種、ディストリビューション依存のコードをコンパイルする。途中でいくつか質問が表示されるが、基本的には、リターンキーでデフォルトの答えを選択すればよい。

スクリプトが終了すると、インストールと設定は終了である。次に、評価用のライセンスキーを入手する。このライセンスキーがないと、VMwareは動作しない。

評価用のライセンスキーを取得するには、VMwareのWebサイトへ行き、http://www.vmware.com/forms/Download.cfmにあるフォームに記入する（**画面3**）。最後に送信ボタンを押せば、登録したメールアドレスにVMwareのライセンスキーが送られてくる。VMware for Linuxの場合、添付ファイルとして送られてくるのはテキストファイルで、これをlicenseという名前にして、ユーザーのホームディレクトリにある“.vmware”ディレクトリに置く。ライセンスキーは、ユーザー1人に1つ必要である。

ちなみに、VMware for Windows NT and 2000では、ライセンスキーはレジストリ登録用のテキストファイル（～.regファイル）で、こちらは、Windows NT／2000で、ファイルにセーブしたあと、ダブルクリックで実行して、レジストリに登録する。

ライセンスキーファイルを登録したユーザーで/usr/bin/vmwareを実行（X Window System環境から）すれば、VMwareのウィンドウが表示されるはずである。これでゲストOSをインストールする準備が整った。



VMwareの画面構成

VMwareのウィンドウは、メニューバー、ツールバーおよびステータスバーと、ゲストOSから見えるVGA画面であるウィンドウ部分から構成されている。ステータスバーの右側には、ハードディスクやCD-ROMのアクセスランプに相当するアイコンがあり、アクセスに応じて点滅する。ただし、シリアルポートやパラレルポートに対応するアイコンは、仮想マシンの環境設定で利用するように設定しない場合には表示されない。

VMwareでは、ゲストOSに合わせて、コンフィギュレーションファイルを用意する。これは、Configuration

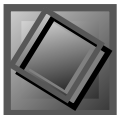
Wizardもしくは、Configuration Editorで作成する。ゲストOSを動かすには、これらを使ってコンフィギュレーションファイルを作成し、それをロードしたのち、仮想マシンの電源を入れ、ゲストOSのインストールを行う。

仮想マシンの全状態は、ディスクに保存することができ、動作中の仮想マシンの状態を保存しておけば、いつでもセーブした時点からゲストOSを再開することができる。

マウスとキーボードは、ゲストOSのウィンドウを一度クリックすると、それらからのデータがゲストOSへ振り向けられ、そのまま使えるようになる。標準状態では、この状態から抜けるには、Ctrl＋Alt＋Escというキーの組み合わせを押す。しかし、ゲストOSインストール後に、VMware Toolsというプログラム（Windows 95／98／NT／2000およびLinux用）を使うことで、VMwareの上にマウスカーソルがある場合には、ゲストOSへマウスやキーボードからのイベントが向かい、VMwareのウィンドウを離れると、ホストOSへ向かうように変更される。

なお、VMwareは、複数起動することが可能だが、個々の仮想マシンが使う実デバイスは共有ができない。たとえば、ある仮想マシンがフロッピーディスクを使っている場合、他の仮想マシンはその間フロッピーディスクを使うことができなくなる。このため、VMwareでは、コンフィギュレーションファイル中で、利用するデバイスを選択できるほか、一時的に仮想マシン内の仮想デバイスと実デバイスの結びつきを切断、再接続することができるようになっている。これを使うことで、たとえば、ホストマシンの2つのシリアルポートをどちらもCOM1として2つの仮想マシンに割り振ったり、必要なときだけフロッピーディスクドライブを仮想マシンに接続して利用することができる。

また、シリアルポートやパラレルポート、フロッピーディスクなどは、実デバイスの代わりにファイルを割り当てることが可能だ。たとえば、シリアルポートに出力するだけといった使い方であれば、その出力をファイルへ切り替えることもできる。



ゲストOSのインストール

VMwareでは、仮想マシン内で動くOSをゲストOSと呼んでいる。Linuxの中でLinuxを動かしても、それはそれでおもしろいのだが、ここでは、異種OSとしてMicrosoftのWindows 98 Second Edition（以下Win98SE）を動かしてみることにした。なお、このためにはWin98SEのインストールCD-ROMが必要となる。Win98SEのCD-ROMは2種類あり、パソコン本体などに付属してくる一部のCD-ROMは、ブート可能CDになっているが、市販のパッケージに入っているCD-ROMは、ブートができないタイプのものである。VMwareは、ハードウェアのエミュレーションを行うため、ゲストOSをインストールするためには、パーティションの作成やディスクのフォーマットといった作業から開始する必要がある。ブート可能なCD-ROMでは、CD-ROMだけで一連の作業を行えるようになっているが、ここでは通常の市販パッケージ版を使うことにする。この場合、パッケージに同梱されている起動フロッピーから起動する。なお、Win98SEのライセンスは、1台のマシンに1コピーだけしか利用を許可していないので、ライセンス違反にならないよう注意してほしい。

まず、VMwareを起動する。ここで、最初にConfiguration Wizardを使い、Win98SEをインストールする環境を作る（**画面4**）。作ったら、これをセーブし、ドライブに起動フロッピーとWin98SE CD-ROMをセットして、VMwareのPowerONボタンを押す（フロッピーディスクを使う設定にしておくこと）。これで、起動フロッピーが読み込まれて、インストーラが起動する。WindowsをインストールするためにVMwareで作成した（仮想の）ハードディスクは、まったくのサラなので、区画を作成し、フォーマットする必要がある。これはインストーラの指示にしたがえばよい。

これで、環境設定に問題がなければ、Windows 98のインストールが開始される。インストールプログラム自体には特に難しいことはないが、筆者の環境では途中で止まってしまった。これは、Windows 98のインストール過程の2度目の再起動時に起こる。ファイルのコピーが終わって、最初の再起動があり、そのあとで、デバイスの設定が行われたのちにもう一度、再起動がかかる。このときに、EMM386でエラーが発生する。このエラーが出ると、このあと、何度リセットしてもハングアップする。これには、仮想ドライブ中にあるファイルを手動で書き換える

画面4　Configuration Wizardの画面
ゲストOSを選び、コンフィギュレーションファイルを作成する。

ことで対応できる。

前記のようになったら、再度、起動フロッピーを入れて、VMwareをリセットする。最初に表示されるメニューで“Start computer without CD-ROM support.”を選ぶと、インストーラではなくDOSプロンプトになる。ここで、Cドライブのルートに移動する。ルートにあるconfig.sysを書き換えるのだが、この時点で、すでに、Cドライブの\windows\commandにテキストエディタがインストールされているのでそれを利用しよう。

```
c:
cd ¥
¥windows¥command¥edit ¥config.sys
```

とするとテキストエディタが起動する。矢印キーでカーソルの移動ができる。

```
device=C:¥WINDOWS¥EMM386.EXE RAM
```

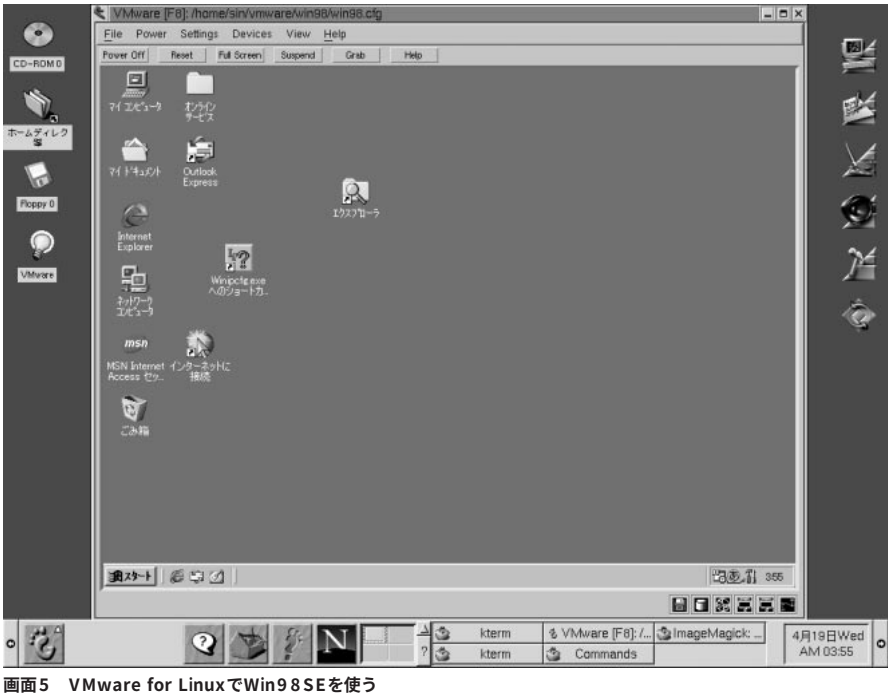
という行があるので、これを、

```
device=C:¥WINDOWS¥EMM386.EXE NOEMS
```

に変更する。どうも、VMwareでは、EMS（DOSからバンク切り替えのメモリを扱うための仕様）のエミュレーションがうまく動いていないようである。書き換えたら、Alt＋F、X、Enterのようにキーを押す。これで、ファイルが上書きセーブされる。ここで、起動フロッピーを抜いて、再度VMwareをリセットする。これで、インストールが先に進むはずである。

なお、インストール中、かなり長い間止まって見えるようなところがあるが、画面にエラー表示が出なければ、待っていれば先に進む（筆者の環境ではインストールに1時間30分以上かかった）。後述するように、VMware環境は、それほど速くない。

Windowsのインストールが終わったら、VMware Toolsのインストールを行う（こちらについては、VMwareのWebサイトにあるドキュメントなどを参照してほしい）。



画面5　VMware for LinuxでWin98SEを使う
Linuxの上でWindowsが動く。もうこれでWordやExcelのファイルも恐くない。

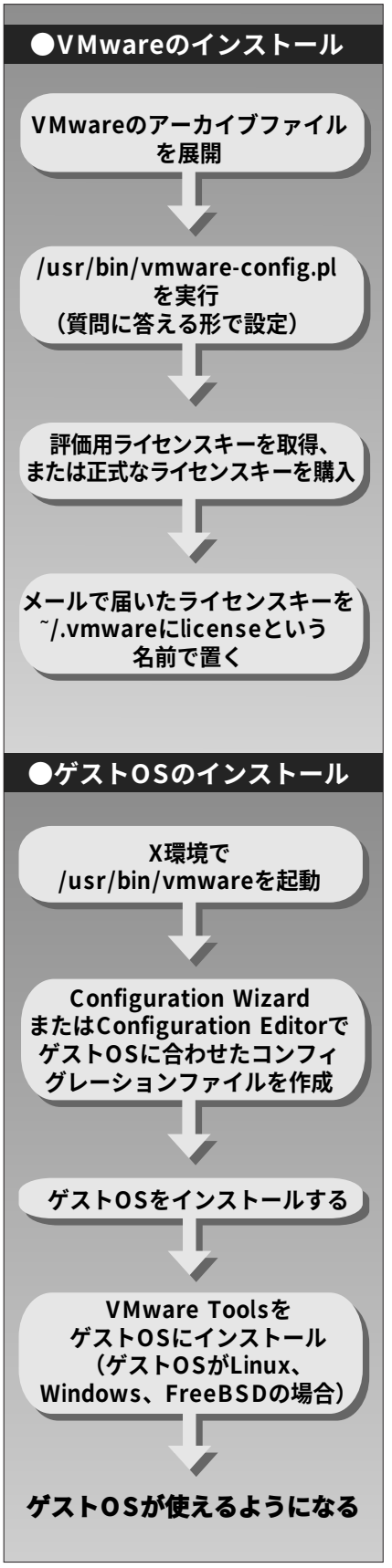
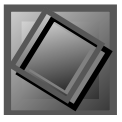


図1　VMware利用までの手順

このほかにWindows 98の設定で問題があったのは、実ネットワーク側にあるDHCPサーバからアドレスのリースが受けられなかったことぐらいで、こちらは、Windows 98のネットワークプロパティで、直接IPアドレスを指定することで、解決した。これでWindows on Linux環境（？）のできあがりだ（**画面5**）。

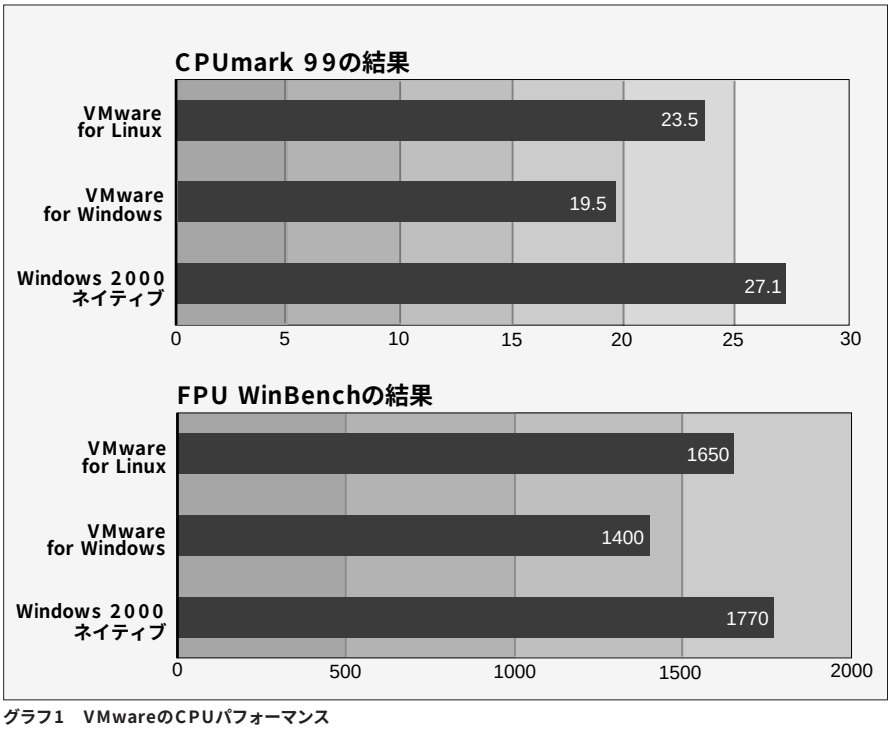


VMware環境での実行効率

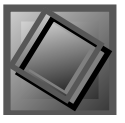
VMware内で動作するゲストOSではどの程度の機能があるか、ベンチマークなどを使って調べてみることにした。ベンチマークには、ZDnetのWinBench99を使ってみたのだが、VMware for Linux環境では、ほとんどの項目でエラーが発生し、ちゃんとベンチマークを取ることができなかった。唯一CPUの速度だけは測ることができたので、これを元に実行効率を考察してみる。比較対照として、同一のマシン上でハードディスクのみを交換して、Windows 2000を動作させ、その中でVMware for Windows NT and 2000を動かし、ゲストOSとして同じWin98SEをインストールした。そこでの比較だが、**グラフ1**のようなものになった。なお、ホストマシンは、333MHzのCeleronを使っている。

これによると、VMwareはWindowsで動かしたより、Linuxで動かしたほうが効率が良さそうである。Windows 2000上での測定値を100%とすれば、Linux版VMwareのCPUmark99は87%、Windows版のVMwareでは72%となった。Windows版では、仮想マシンのプライオリティを変更することができ、テストではこれをHighにして行った。Linux版では9割近いスピードになっているのに対して、Windows版は7割。これは、おそらく環境の違いによるものとだと推定される。LinuxがWindows 2000よりも軽いぶん、VMwareの動作効率が上がったのであろう。

ただ、体感速度的に見ると、Windows版、Linux版のどちらのVMwareもそれほどの違いは感じない。また、ホストOS側と比べると、やはりゲストOSはかなり遅い印象を受ける。これは、ホストOSがLinuxでゲストOSがLinuxの場合にも感じた。特にどのゲストOSでもディスクアクセス関係はかなり遅い感じで、アプリケーションを使う場合の体感速度は、ホストOSとゲストOSではかなり違うのではないかと思われる。逆にこれから推測すると、頻繁にディスクをアクセスしないようなアプリケーション、使い方（たとえば、メモリ占有率を上げるなどの方法がある）であれば、7割から9割程度の速度で、複数環境を同時に使える、いつでもゲストOSを中断できる、などのメリットはある程度生きてくると思われる。



グラフ1　VMwareのCPUパフォーマンス



VMwareを使う？

全体として、複数のOSを1台のマシンで同時に使ったり、利用頻度は高くないものの、ときどき動かさねばならない環境などというのがあるなら、VMwareで複数環境を同時に使うというのはかなり便利だと思う。たとえば、Linuxで作業している最中に、Windows 98を動かして、ちょっとExcelで処理するとか、簡単なドキュメントをWordで作るとかである。

そういえば、異種OSでネットワーク越しにGUI操作を可能にするVNC（Virtual Network Computing）というソフトウェアがある。これを使ってLinuxマシンから別のWindowsマシンを使うなんてことが行われてきたが、VMwareを使うと、見た目には似たような感じになる。

CPU速度が向上し、ディスクなども大容量になってきた昨今、複数の環境を1つのマシンに入れてしまってもいいのかもしれない。

Linux日記

第9回　名前解決（2）

今回は、インターネットドメイン名の基本的な構造を示し、そしてドメイン名を解決するために、分散データベースを使うネームサーバというサービスが必要であるということを説明しよう。

文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

とりあえずすぐに原稿が書けるネタが思い付かなかった筆者は、編集者にいわれた通りDNSの記事を書き始めた。ちょうど我が家でOCNエコノミーの設定をやっていたからだ。ところが、ネームサーバに割り当てたマシンのRTC（リアルタイムクロック）が2000年問題にひっかかり、マシンを別のものにリプレイスしなければならなくなった。結局、マシンを1台新調し、SETIのためのCPUパワーがさらに強化されたのであった。

前回は、ここまでの話をした。このような仕事（ネットワークの解説など）をしていると、どうしても自分の仕事場（すなわち自宅）に、サーバまであるネットワークを敷設することになる。筆者はWindows関係の仕事もやっているので（というか、そちらのほうが多い）、さまざまな環境を用意するために、かなり多数のマシンを用意している。今回、OCNエコノミーの開通とほぼ同時にWindows 2000が登場した。プレリリース版を試験的にインストールしたマシンもあったのだが、正式発売を機に、製品版のServerとProfessionalをインストールしてまじめに動かすことにした（そういう仕事もあるのだ）。Windows 2000導入に伴うマシンのやりくりと、インターネットサーバに伴うやりくりと、加えてタコなRTCを持つマシンのやりくりとで、我が家のマシン（12台くらいある）のうち、6台のOSを入れ換え、ディスクやネットワークアダプタなどをとっかえひっかえした。くらくらするような作業だった。

すべてのマシンを止めて、じっくりやればいいのであるが、既存のインターネット環境（特にメール）を止めるわけにはいかず、また、高速マシンを長時間止めるというのはSETIの成績的にも好ましくないので、いかに各マシンのダウンタイムを短くしながら新しい環境に移行するかという点で結構悩まされた。

ちなみに、SETI@HOMEのほうは、暮れのPentium III 500MHz×2マシンとついこの間のPentium III 700MHzマシンの導入により（あと、出先のサーバの拝借により）かなりペースが上がった。少ない日でも15程度、多い日なら20ユニットまでいっている。個人成績では、185万人中の上位0.5%に入った。

■インターネット接続環境の整備

前回の原稿を書き終えてから、OCNの常時接続の設定作業を再開した。前回述べたように、我が家のネームサーバにはFreeBSDを使うことにした。タコなRTCのマシンから移行するために、まずは、Pentium 120MHzのシステムにFreeBSDをインストールするという作業である。これはかなり古いマシンなので、ATAPI接続のCD-ROMドライブではCD-ROMブートができない。別のマシンでブートフロッピーを作って、フロッピーブートしてみようと思ったのだが、うまく動かない。こんなことで悩んでいるより、CD-ROMブートできるようにしたほうが早いと思い、マシン構成のやりくりで浮いていたSCSIホストアダプタ、Adaptec

AHA-2940を装着し、手元にあった遅いSCSIのCD-ROMドライブをつなぐことにした。ここで問題が発生した。このCD-ROMドライブにターミネータが付いていなかったのである（今時のハードディスクと異なり、集合抵抗をソケットに挿すタイプだったのだ）。ターミネータがないことでエラーが起こってもつまらないので、やはりマシンのやりくりで浮いていた9GバイトのSCSIディスクもつないだ（こいつの役割は、純粋にターミネータである）。で、どうにかCD-ROMブートし、FreeBSDがインストールできた。もちろん、インストール後にこれらは外した。

■pingが返ってこないぞ

次の作業は、このマシンの設定であるが、DNSやsendmailは、RTCがタコなPentium 133MHzマシンでやっていたので、単にファイルをコピーするだけである。この時、興味深い現象でトラブったので、その事例をちょっと紹介しておこう。普通は起こらないことなのだが、現在我が家が2系統の経路でインターネットに接続しているがために発生した現象である。

連載の最初の頃に述べたように、我が家のメインのインターネット接続環境は、IIJのネットワーク型ダイヤルアップ接続だった。この接続形態では、契約者にグローバルアドレスが割り当てられる。これに、OCNの常時接続環境を追加した。これもグローバルアドレスが割り当てられる。これらは異なるネットワークアドレスなので、異なるLANセグメントを用意した。そして、ルータとして動作するマシンで、この2つのネットワークを接続した。

ルータマシンをひと通り設定し、IIJのアドレス側のセグメントのマシンのルーティング設定を変え、OCN経由でインターネットに繋がるようにした。ネームサーバによる名前解決は、OCN経由で問題なく動作した（ネームサーバは、OCN側のセグメントにつながっている）。ではpingは通るかなと思ったら、これが通らない。OCNのアドレスを持つ我が家のLAN上のホストには届くのだが、外部へのネットワークコネクションが確立しないのである。デフォルトルート設定は間違いない。ネームサーバも問題なく動いている。なんで繋がらないの？　数分考えた末、結論が出た。こちらが送るパケットは、（たぶん）相手に届いているのである。だが、送信元アドレスはIIJ側のアドレスだ。すなわち応答は、IIJ側のルータに返されるのだ。しかし、我が家のLANは、IIJに常時接続しているわけではない。繋がっていなければ、IIJから送信側ホストにはパケットは届かないのである（**図1**）。

IIJのダイヤルアップ型接続は、OCNが問題なく動作するようになったら解約するつもりである。解約により、このLANセグメントはグローバルIPアドレスが使えなくなるし（このアドレスは、IIJからの借り物なのだ）、OCNのグローバルアドレスは4ビットしかないし、各種マシンをすべてグローバルアドレスのセグメントに置くのも何なので、メインのLANをプライベートアドレスにすることにした。プライベートアドレスセグメントからインターネッ

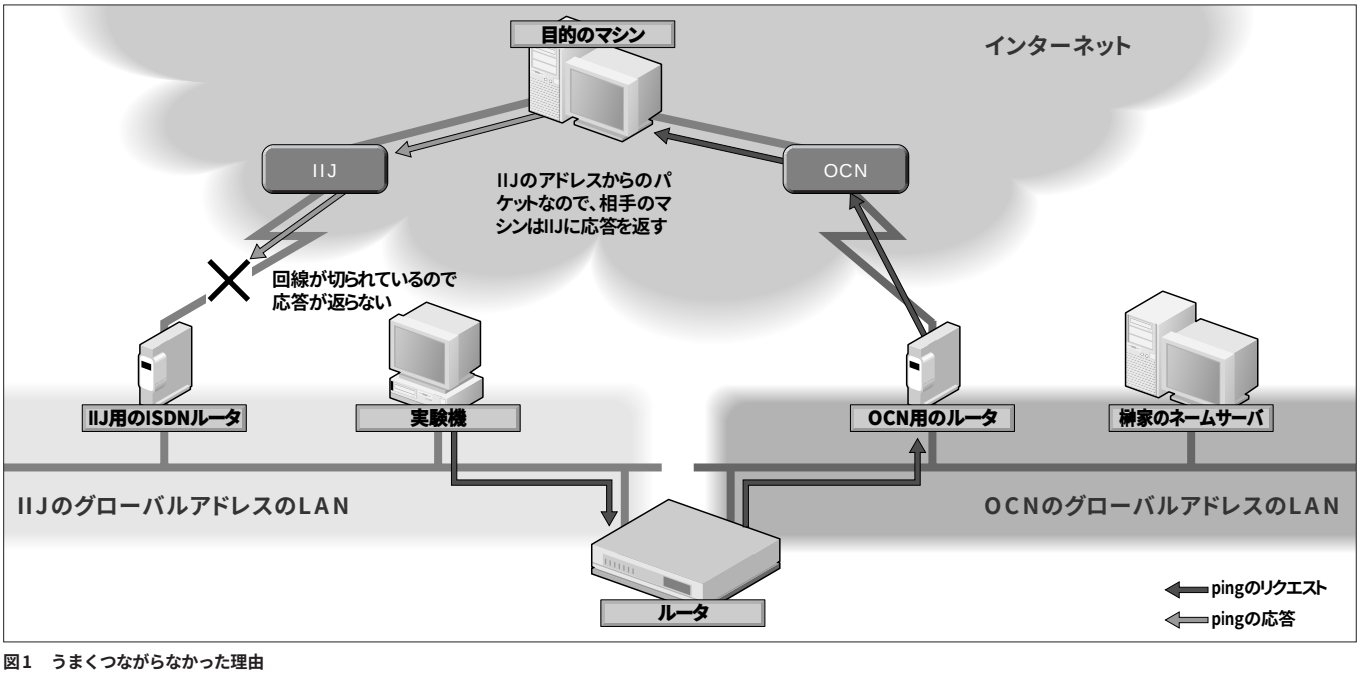


図1　うまくつながらなかった理由

トに接続するためには、NATやIPマスカレードを使わなければならない。プライベートIPアドレスは、インターネット上に送るべきではないし、送られても応答が返ってこないからだ。

というわけで、ルータマシンでIPマスカレードが使えるようにした。そして、IIJ側セグメントのマシンに、試験的にプライベートアドレスを割り当ててみた。めでたしめでたし、問題なく動作するようになった。

教訓は、IPパケットは、経路情報に制御されて正しい宛先に送られるということだ。この経路情報は、パケットが実際にどのような経路で送られてきたかには関係ない。本来であれば、IIJから送り出されるはずのパケットがOCNから送られた。しかし中継では宛先アドレスしか見ないので、相手には届く。相手は、送信元アドレスを宛先として応答を返す。応答は、経路制御情報に従って（OCNではなく）IIJに送られる。頭ではわかっていたことではあるが、身を持って経験することになったのであった。

また、このような時に動的な経路制御（routedデーモンを動かすなど）を行っていると、ネットワーク上でトラブルが発生する可能性がある。我が家の場合でいえば、IIJ側のアドレスが、OCN側のアドレスの下に繋がっているという情報をデーモンが送り出してしまうのである。もしこれをOCN側のルータが認識してしまうと、誤った経路制御情報が登録されてしまう。本来、基幹回線を通ってOCNからIIJに送られるべきパケットが、誤って我が家に送られてくる可能性があるのだ。このようなトラブルを防ぐために、動的な経路制御情報は、信頼できる相手としかやり取りしないというのが原則である。そしてプロバイダ側も、このようなトラブルを防ぐために、末端の契約者からの経路制御パケットは受け取らないようにしている（と思う）。このようなトラブルは、大規模な組織内ネットワークでも起こるので気を付けよう。

さて、前置きが長くなったが、前回の続きでDNSの話だ。

■DNSを使った基本的な名前解決

名前が階層化されているということにより、インターネットドメイン名を1つのデータベースで集中管理する必要がなくなり、名前とIPアドレスなどの情報を分散データベースに収めることが可能になった。と書くと難しく聞こえるが、階層化ファイルシステムを考えてみればそんなに難しいことではないとわかるだろう。ここでいうデータベースはディレクトリに相当するものだ。階層化ファイルシステムでは、全体で莫大な数のファイルがあったとしても、各ディレクトリに記録されているファイルエントリの数は、把握し切れる程度のものだ。ディレクトリの数も膨大になるが、ツリー構造のおかげで、それぞれのディレクトリの位置を識別し、検索するのは容易である。階層化ファイルシステムは、膨大な数のファイルを、階層的に配置されたディレクトリという分散データベースで管理しているので、（ファイルの数の割には）管理が容易になっているのである。

インターネットドメイン名も同様である。ルートドメインの下にあるcomやjpといったトップレベルドメイン名から伸びる枝ごとに、各サブドメインを管理するデータベースを別々のネームサーバ上に分散できる。さらに下位のサブドメインも同様である（**図2**）。

ファイルシステムの場合は、管理データベースは各ディレクトリである。つまり、ツリー構造のノードの部分に必ずデータベースが存在していることになる。DNSの場合、この分散データベースは、階層化ファイルシステムの

Column

グローバルアドレスとプライベートアドレス

インターネットにおけるIPアドレスは限られた資源である。今や、インターネットに接続する組織のすべてのコンピュータに正式なIPアドレスを割り振ることは困難になった。そのため、割り当てられたアドレス数では不足する組織は、組織内の各コンピュータに非公式なIPアドレスを振る。そして、インターネットに接続するためのルータやゲートウェイ、ファイアウォール上で、これらのコンピュータからのIPパケットの送信側アドレスを正式なIPアドレスに変換するのである。

組織に割り当てられた正式なIPアドレスを、グローバルIPアドレスという。これは、世界中で一意に識別できるIPアドレスである。これに対して、組織内で使うための非公式なIPアドレスが定められている。これがプライベートIPアドレスである。プライベートIPアドレスは、インターネット上ではどこにも割り当てられていない。それゆえ、もしこのようなプライベートIPアドレスがインターネット上に送出されても、それはどこかの組織が間違って送り出している無効なパケットであることがわかる。もちろん、送信元のアドレスがプライベートアドレスだと、応答は返ってこない。

現在、以下のアドレスがプライベートIPアドレス用に定義されている（かっこの中はネットマスク）。

10.0.0.0〜10.255.255.255（8ビット）
172.16.0.0〜172.31.255.255（16ビット）
192.168.0.0〜192.168.255.255（24ビット）

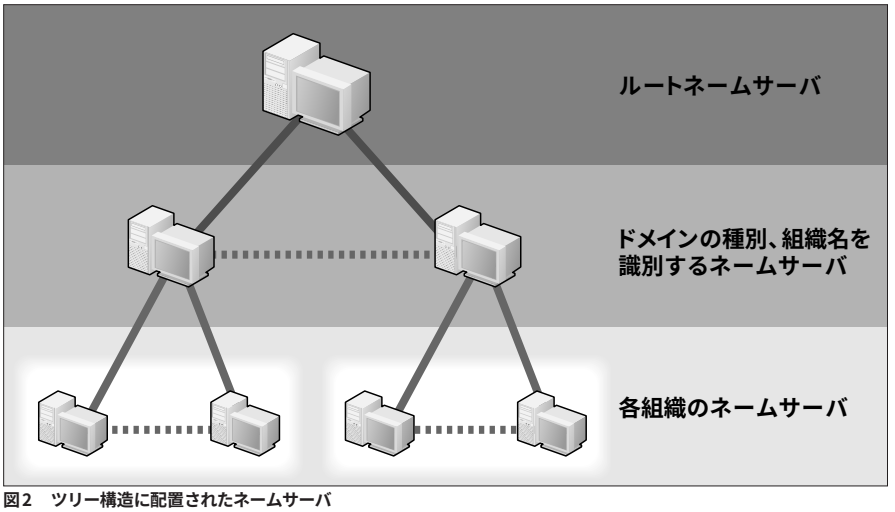


図2　ツリー構造に配置されたネームサーバ

ディレクトリと似ている部分もあるし、違う部分もある。これからDNSを使った名前解決を実際に行い、その類似点と相違点を見ていく。ここでは例として、www.ascii.co.jpというドメイン名からIPアドレスを求めてみよう。

階層化された名前空間で検索を行うには、まずはルートにたどり着かなければならない。ルートドメインは、ルートディレクトリと同じで名前はない。ルートドメインを明示的に指定する場合は、“.”という形で表記する（ルートディレクトリを“/”と表記するのと同じことだ）。ルートドメインの下位には、com、net、jp、us、ukといったトップレベルドメインがある。さて、DNSでは、どのようにしてルートにたどり着くのか？　UNIXのファイルシステムでは、ルートディレクトリはi-node番号が2という約束があり、それに基づいて探すことができた。DNSの場合は、トップレベルドメイン名の情報を解決するためのネームサーバ（ルートネームサーバ）の情報を、各ネームサーバに静的に登録することになっている。つまりネームサーバは、ルートネームサーバのIPアドレスだけは最初から知っているのである。ルートネームサーバは随時変更されるが、最新情報はFTPで持ってくることができる（国内なら、たとえばftp.nic.ad.jpの/internic/rs/domainなどから入手できる）。ネームサーバを運用する場合は、この情報を適宜最新のものに更新しなければならない。

■nslookupでドメイン名を調べる

ここで実際にネームサーバが返す情報を見るために、nslookupというコマンドを使ってみよう。ネットワークアプリケーションがドメイン名からIPアドレスを求める場合、アプリケーションが直接ネームサーバと対話することはない。各システムの構成に応じて、オペレーティングシステムやネットワークライブラリがこの処理を行ってくれるからだ。このクライアント側のモジュールをリゾルバという。通常のネットワークソフトウェアはリゾルバを介して名前解決を行うが、nslookupはリゾルバを介さず、直接ネームサーバと対話するためのツールで、いろいろな入力や検索モードに応じて、ネームサーバがどのような情報を返すかを生の形で見ることができる。詳細な使用方法はmanを参照のこと。

ここでは、nslookupを対話モードで使い、ネームサーバが返す情報を見ていく。ここで示したnslookupの実行例は、FreeBSD 3.3のものであるが、nslookupはネームサーバのパッケージ（BIND）に含まれるものなので、Linuxでもほとんど同じ結果が得られるはずだ。また、実行例中で示されているドメイン名やIPアドレスはすべて実在のものである。これらには、ユーザーに対して前向きに公開していない情報も含まれているが、誰でも合法的に得られる情報であるし、実際にシステム内部で使われている情報である。こういった情報を悪用することもできるが、実際に悪用したいと思っているクラッカーならこういう情報を求める方法は常識中の常識なので、そのまま掲載している。

nslookupを使う場合、FQDNをきちんと使うこと。つまり、フルパスのドメイン名を指定する場合は、最後にピリオドを付けるということだ。ピリオドがない場合、相対指定（指定した名前のあとにデフォルトドメインを付加する）として解釈される。以後の解説も、基本的にFQDNで表記していく（ピリオドで終わらない名前は相対指定ということだ）。

ここで紹介するnslookupを使った実験は、各コンピュータの設定がきちんと行われており、そして何らかの方法でインターネットに接続していれば実際にやってみることができる。そのマシンでネームサーバが動作している必要はない。nslookupはデフォルトで、そのシステムがDNSリクエストを送るネームサーバに接続する。また、問い合わせを送るネームサーバを明示的に指定することもできる。nslookupはWindows NT、Windows 2000にも移植されている（コマンドラインツールである）ので、UNIXユーザーでなくても確かめてみることができる。

まずは、ルートサーバを見てみよう（**リスト1**）。これは前に説明したように、ネームサーバ上に静的に登録されているので、IPアドレスは明らかである。

ルートサーバはトップレベルドメイン名を管理しているのだが、別にcom.とかjp.というホストがあるわけでもなく（もちろん、名前のないホストがルートにあるわけでもない）、そのIPアドレスを求めるということもない。トップレベルドメインは、まさに組織種別や国といったグループの分類のために存在しているドメインだからだ。では、ルートサーバはトップレベルドメインについて何を教えてくれるのか？

このようなホストコンピュータを表さないドメイン名については、そのドメインを管轄しているネームサーバを教えてくれるのである。これは、ルートサーバに限ったことではない。ネームサーバは、指定されたドメイン名がホスト（あるいは指定した特定の情報）を特定するうえで十分なものでない場合は、その情報をより詳しく知っているであろう別のネームサーバを教えてくれるのだ。たとえばルートネームサーバに、jp.というドメイン名を問い合わせると、jp.ドメインを管轄しているネームサーバを教えてくれる。そのため、クエリータイプにNS（ネームサーバ）を指定しているのだ。

ルートサーバを確認したら、トップレベルドメイン名について調べてみよう。ここではjp.ドメインについて調べてみる。ネームサーバは、jp.ドメインを管理するネームサーバを教えてくれるはずだ（**リスト2**）。

返された情報を見ると、Authoritativeな（権威がある）応答と、Non-authoritativeな（権威のない）応答があることがわかる。権威があるというのは、そのサーバが正式なデータベースを持っているということである。権威がないというのは、正式なデータベースに基づいていないということだ。たとえば、正式なデータベースを持たないネームサーバ上でキャッシュされているデータに基づいて得られた応答は、権威のない応答である。この例は、問い合わせはローカルホスト上のネームサーバを介して行っているので、そこにキャッシュされている情報がNon-authoritativeな応答として返されるのである。一般に、権威がある応答と権威がない応答は同じ内容であるが、データが変更された時には、キャッシュ情報と正式な情報が異なる場合がある。もちろん、権威がある応答のほうが正しい情報である（はずだ）。

さて、jp.ドメインを管轄するネームサーバがわかった。次にjp.のサブドメインであるcoを調べてみよう。co.jp.ドメインの問い合わせを送ってみる。coもグループを識別するドメイン名なので、返される情報はネームサーバ情報

Column

同じ情報を返すネームサーバは複数ある

ネームサーバが返す別のネームサーバ情報を見ると、あるドメインを管轄するネームサーバが複数あることがわかる。これは、信頼性の向上と負荷分散のためである。リゾルバは、次に問い合わせを送るネームサーバをランダムに（あるいは応答時間などを評価して）選択する。インターネットは、信頼性の低いネットワークである。回線が切れたり、マシンがダウンしていることもある。そういう時でも、極力サービスが停止しないようになっているのだ。

▼リスト1　ルートサーバを調べる

```
% nslookup          引数を指定しないと対話モードになる
> set type=NS       クエリータイプにNS（ネームサーバ）を指定
> .          ルートドメインをFQDNで指定
             ルートサーバー情報が得られる。
Non-authoritative answer:
(root)  nameserver = J.ROOT-SERVERS.NET
(root)  nameserver = K.ROOT-SERVERS.NET
(root)  nameserver = L.ROOT-SERVERS.NET
       :
Authoritative answers can be found from:
J.ROOT-SERVERS.NET      internet address = 198.41.0.10
K.ROOT-SERVERS.NET      internet address = 193.0.14.129
L.ROOT-SERVERS.NET      internet address = 198.32.64.12
       :
```

▼リスト2　jp.ドメインについて調べる

```
> jp.     jpドメインをFQDNで指定
          jpドメインのネームサーバ情報が得られる
Non-authoritative answer:
jp      nameserver = DNS0.SPIN.AD.jp
jp      nameserver = NS-JP.SINET.AD.jp
jp      nameserver = NS.WIDE.AD.jp
jp      nameserver = NS0.IIJ.AD.jp
jp      nameserver = NS0.NIC.AD.jp
jp      nameserver = NS-JP.NTT.NET

Authoritative answers can be found from:
DNS0.SPIN.AD.jp internet address = 165.76.0.98
NS-JP.SINET.AD.jp       internet address = 150.100.2.3
NS.WIDE.AD.jp   internet address = 203.178.136.63
NS0.IIJ.AD.jp   internet address = 202.232.2.34
NS0.NIC.AD.jp   internet address = 202.12.30.131
NS-JP.NTT.NET   internet address = 210.175.162.226
```

である（**リスト3**）。

返されたネームサーバ情報は、jp.ドメインの時と同じである。これがDNSと階層化ファイルシステムの違うところだ。co.jp.とjp.は明らかに違うレベルである。階層化ファイルシステムなら、jpを管理するディレクトリとcoを管理するディレクトリは別のものになる。しかし、DNSの場合は、1台のネームサーバで、複数のレベルの解決を行えるのである。あるネームサーバが管轄するサブドメイン部分のことを、そのサーバが管轄する「ゾーン」という。ゾーンについては、またあとで解説しよう。

さて、次はascii.co.jp.の解決である。これもアスキーという会社組織のグループを識別するドメイン名なので、得られる情報はネームサーバである。coを専門に解決するサーバはないので、この情報を返すサーバはjp.ドメインを管轄するものだ。

■アスキーのドメインを検索する

さて、アスキーについて調べてみよう（**リスト4**）。結果を見ると、ascii.co.jp.を管理しているネームサーバが3台あることがわかる。2台はおそらくアスキー社内にあるもの（ascns1とascns2）、もう1台は、プロバイダであるIIJのネームサーバである。プロバイダにもネームサーバを用意することで、回線切断時でも名前解決が可能になり、信頼性を向上できる。

さて、いよいよ目的のwww.ascii.co.jp.の検索である。このドメイン名は、組織やグループを識別するものではなく、特定のホストコンピュータを指定するもののはずだ（そうでなければ、ブラウザが接続できない）。そこで、クエリーのタイプをネームサーバからホストに変更する（**リスト5**）。

Name:というフィールドに示されているのが目的のホストのドメイン名、Address:はそのIPアドレスである。目的のホストのドメイン名は、www.ascii.co.jp.ではなく、at2.ascii.co.jp.である。そして、Aliases:というフィールドに、指定したwww.ascii.co.jp.が現れている。Webサーバの名前が2種類現れた。試しに、http://at2.ascii.co.jp/というURLを指定して、ブラウザで接続してみよう。http://www.ascii.co.jp/と指定した時と同じ画面が表示されるのがわかる。つまり、アスキーのWebサーバは、at2.ascii.co.jp.というドメイン名のホストなのである。それに対して、www.ascii.co.jp.という別名（Alias）が割り当てられている。ネームサーバに対して別名を指定した場合も、本来の名前と同じようにIPアドレスが返される。Webサーバの名称は、一般にwwwなので、別の名前を持つホストにwwwという別名を割り当てているのだろう。このようにすることで、組織内でマシンを入れ換えたりした時でも、別名の割り当てだけ変更すれば、ユーザーが名前指定を変更することなく、サービスを継続できる。もちろん、ホストのドメイン名をwwwにし、別名を割り当てないという使い方も可能だ。

■さて次回は

次回は、少し特殊な名前解決について説明し、そしていよいよBINDの解説に入る。BINDというのは、UNIXにおいてもっとも広く使われているドメインネームサーバのパッケージだ。

▼リスト3　co.jp.ドメインについて調べる

```
> co.jp. co.jp.ドメインを調べる
         co.jp.ドメインのネームサーバ情報
co.jp   nameserver = ns0.iij.ad.jp
co.jp   nameserver = dns0.spin.ad.jp
co.jp   nameserver = ns-jp.sinet.ad.jp
co.jp   nameserver = ns-jp.ntt.net
co.jp   nameserver = ns0.nic.ad.jp
co.jp   nameserver = ns.wide.ad.jp
ns0.iij.ad.jp   internet address = 202.232.2.34
dns0.spin.ad.jp internet address = 165.76.0.98
ns-jp.sinet.ad.jp       internet address = 150.100.2.3
ns-jp.ntt.net   internet address = 210.175.162.226
ns0.nic.ad.jp   internet address = 202.12.30.131
ns.wide.ad.jp   internet address = 203.178.136.63
```

▼リスト4　ascii.co.jp.について調べる

```
> ascii.co.jp.    ascii.co.jp.ドメインのネームサーバを調べる

Non-authoritative answer:
ascii.co.jp  nameserver = ascns1.ascii.co.jp
ascii.co.jp  nameserver = ascns2.ascii.co.jp
ascii.co.jp  nameserver = ns1.iij.ad.jp

Authoritative answers can be found from:
ascns1.ascii.co.jp  internet address = 202.32.48.44
ascns2.ascii.co.jp  internet address = 133.152.0.130
ns1.iij.ad.jp   internet address = 202.232.2.35
```

▼リスト5　www.ascii.co.jp.のIPアドレスを調べる

```
> set type=A      ホストを示す情報を指定
> www.ascii.co.jp.         Webサーバのドメイン名を調べる

Name:    at2.ascii.co.jp
Address:  210.140.231.23
Aliases:  www.ascii.co.jp
```

Webサーバ構築術(第10回)

Perlで記述したためか、スクリプトの数が増えすぎ、後半部分が今回へと流れ込んだ。今回は、ApacheとDBの連動利用の第2回目だ。前回のソースの続きと、Perl以外でのスクリプトについてふれる。

Services

ApacheとPostgreSQLの連動（その2）

文：中島昌彦
Text : Masahiko Nakajima

表示部分、抽出処理が格段に楽になる

Webサーバを運営している以上、自分のサイトにはなにかしら検索の仕組みが欲しいものだ。

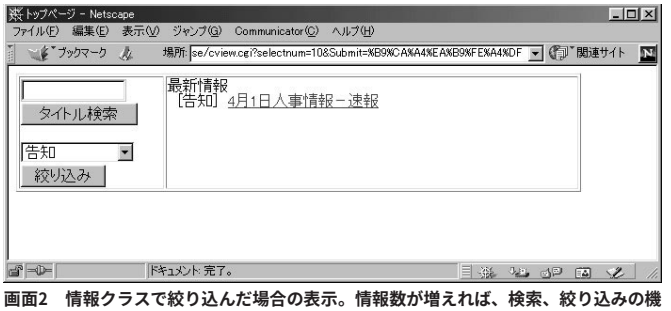
PostgreSQLのようなDBでデータを管理することで、タイトル検索、絞り込みの作業の処理が手軽に装備できる（**画面1**）。この例の機能としては、タイトルに対する検索機能と、クラス別に分けた絞り込みの機能を装備した（**画面2**）。数個の情報であれば、わざわざ検索機能や絞り込み機能を加える必要はないが、公開情報が増えていけば、タイトルの検索や絞り込みの機能が必須である。

そのための仕組みが、index.cgi、cview.cgi、sview.cgiと、テンプレートとなるindex.htmlの4ファイルだ。また、1つのデータの内容を表示するdisp.cgiとテンプレートファイルdisp.htmlの合計6ファイルで、一般ユーザーが使うアクセス部分が構成されている（**リスト1**）。

スクリプト自体はごく単純で、指定したデータベースに接続（Pg::connectdb("データベース名")）したあとに、SQLクエリーを発行する（$conn－>exec("クエリー")）。$conn－>execはオブジェクトとして戻り値を返すので、格納変数として、$resultを使っている。$conn－>execで得られた結果は、$result（行数、pid）として取得できるので、それを元に処理をする。

巨大なDBを扱うようになると、$conn－>execで実行した内容に対して、正しくDBが動作しているかどうかのチェックが必要になる。本来であれば、$result－>resultStatusを使ってexecしたSQLが正しく実行できているかどうかのエラーチェックが必要となる。PostgreSQLを使えるようにするためのPerlモジュール「Pg.pm」の細かなドキュメントは、

画面1　利用者が目にする部分。DBの検索機能を利用することで、タイトル部分の検索機能、クラスによる絞り込み機能が手軽に実装できる



画面2　情報クラスで絞り込んだ場合の表示。情報数が増えれば、検索、絞り込みの機能が必須となる

▼リスト1-1 index.cgi

```
#!/usr/bin/perl

use Pg;
require "psqldb.pl";

&query;
&opendb ('testrel');
$max=&sqlout ('select * from classmaster order by cnum');
if (!$max) {
        $select='<OPTION VALUE="">--ありません--';
} else {
        $select='<SELECT NAME="selectnum">\n';
        $select .='<OPTION VALUE="">ジャンル選択';
        for ($i=0;$i<$max;++$i) {
                $select .='<OPTION VALUE="'.$result-
>getvalue($i,0).'">'.$result->getvalue($i,1)."\n";
        }
        $select .="</SELECT>\n";
        $select =~ s/("$QUERY{selectnum}")/\1 SELECTED/;
}

$max=&sqlout ("select
subject.count,subject.relsub,classmaster.cnam from
subject,classmaster where subject.cnum = classmaster.cnum
and subject.relopen='True' order by subject.count");

if (!$max) {
        $getlist='--ありません--<BR>';
} else {
        for ($i=0;$i<$max;++$i) {
                $subject=$result->getvalue($i,1);
                $cnam=$result->getvalue($i,2);
                $subject=~ s/ +$//g;
                $cnam=~ s/ +$//g;
                $getlist .='。。['.$cnam.']。。<A
HREF="disp.cgi?value='.$result-
>getvalue($i,0)."\">$subject</A><BR>\n";
        }
}

print "Content-type: text/html\n\n";
open (READ,"index.html");

while (<READ>) {
        s/--getlist--/$getlist/g;
        s/--search--/$QUERY{search}/g;
        s/--select--/$select/g;
        s/--selectnum--/$QUERY{selectnum}/g;
        print;
}
exit;
```

▼リスト1-2 cview.cgi

```
#!/usr/bin/perl

use Pg;
require "psqldb.pl";

&query;
&opendb ('testrel');
$max=&sqlout ('select * from classmaster order by cnum');
if (!$max) {
        $select='<OPTION VALUE="">--ありません--';
} else {
        $select='<SELECT NAME="selectnum">\n';
        $select .='<OPTION VALUE="">ジャンル選択';
        for ($i=0;$i<$max;++$i) {
                $select .='<OPTION VALUE="'.$result-
>getvalue($i,0).'">'.$result->getvalue($i,1)."\n";
        }
        $select .="</SELECT>\n";
        $select =~ s/("$QUERY{selectnum}")/\1 SELECTED/;
}
$option='';
if ($QUERY{selectnum}) {
        $option="and subject.cnum='$QUERY{selectnum}'";
}

$max=&sqlout ("select
subject.count,subject.relsub,classmaster.cnam from
subject,classmaster where subject.cnum = classmaster.cnum and
subject.relopen='True' $option order by subject.count");

if (!$max) {
        $getlist='--ありません--<BR>';
} else {
        for ($i=0;$i<$max;++$i) {
                $subject=$result->getvalue($i,1);
                $cnam=$result->getvalue($i,2);
                $subject=~ s/ +$//g;
                $cnam=~ s/ +$//g;
                $getlist .='。。['.$cnam.']。。<A
HREF="disp.cgi?value='.$result-
>getvalue($i,0)."\">$subject</A><BR>\n";
        }
}

print "Content-type: text/html\n\n";
open (READ,"index.html");

while (<READ>) {
        s/--getlist--/$getlist/g;
        s/--search--/$QUERY{search}/g;
        s/--select--/$select/g;
        s/--selectnum--/$QUERY{selectnum}/g;
        print;
}
exit;
```

▼リスト1-3 sview.cgi

```
#!/usr/bin/perl

use Pg;
require "psqldb.pl";

&query;
&opendb ('testrel');
$max=&sqlout ('select * from classmaster order by cnum');
if (!$max) {
        $select='<OPTION VALUE="">--ありません--';
} else {
        $select='<SELECT NAME="selectnum">\n';
        $select .='<OPTION VALUE="">ジャンル選択';
        for ($i=0;$i<$max;++$i) {
                $select .='<OPTION VALUE="'.$result-
>getvalue($i,0).'">'.$result->getvalue($i,1)."\n";
        }
        $select .="</SELECT>\n";
        $select =~ s/("$QUERY{selectnum}")/\1 SELECTED/;
}

$max=&sqlout ("select
subject.count,subject.relsub,classmaster.cnam from
subject,classmaster where subject.cnum = classmaster.cnum
and subject.relopen='True' and subject.relsub like
'%$QUERY{search}%' order by subject.count");

if (!$max) {
        $getlist='--ありません--<BR>';
} else {
        for ($i=0;$i<$max;++$i) {
                $subject=$result->getvalue($i,1);
                $cnam=$result->getvalue($i,2);
                $subject=~ s/ +$//g;
                $cnam=~ s/ +$//g;
                $getlist .='｡｡['.$cnam.']｡｡<A
HREF="disp.cgi?value='.$result-
>getvalue($i,0)."\">$subject</A><BR>\n";
        }
}

print "Content-type: text/html\n\n";
open (READ,"index.html");

while (<READ>) {
        s/--getlist--/$getlist/g;
        s/--search--/$QUERY{search}/g;
        s/--select--/$select/g;
        s/--selectnum--/$QUERY{selectnum}/g;
        print;
}
exit;
```

▼リスト1-4 index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>

<HEAD>
        <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>トップページ</TITLE>
</HEAD>

<BODY BGCOLOR="white">

<P>
<TABLE BORDER="1" WIDTH="600">
        <TR>
                <TD WIDTH="150" VALIGN="TOP">
                        <FORM ACTION="sview.cgi?"
METHOD="GET" ENCTYPE="application/x-www-form-urlencoded">
                        <INPUT TYPE="TEXT"
NAME="search" SIZE="12" VALUE="--search--"><BR>
                        <INPUT TYPE="SUBMIT"
NAME="Submit" VALUE="タイトル検索"><BR>
                        </FORM>
                        <FORM ACTION="cview.cgi?"
METHOD="GET" ENCTYPE="application/x-www-form-urlencoded">
                        --select--
                        <INPUT TYPE="SUBMIT"
NAME="Submit" VALUE="絞り込み"></TD>
                        </FORM>
                <TD VALIGN="TOP" WIDTH="434">最新情報<BR>
                        --getlist--
                </TD>
        </TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

▼リスト1-5 disp.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>

<HEAD>
        <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>--subject--</TITLE>
</HEAD>

<BODY BGCOLOR="white">

<P>
<TABLE BORDER="0" CELLPADDING="5" CELLSPACING="4"
WIDTH="600">
        <TR>
                <TD WIDTH="600"><H3>[--cnam--]｡｡--
subject--</H3></TD>
        </TR>
        <TR>
                <TD WIDTH="600"><PRE>--content--
</PRE></TD>
        </TR>
</TABLE>


</BODY>

</HTML>
```

```
$ perldoc Pg
```

で閲覧できるので、そちらを参照してほしい。

PHP3を使うとHTMLが管理しやすくなる

ここまでで、前回から続いてきた一連のApache＋DBの連携利用が一段落する。

ところで、Perlを使った問題点がご露呈する。あくまでも処理系として作られたPerlをCGIとして使おうとすると、HTML表示のためのテンプレートと処理部分のペアという使い方が管理しやすい。処理スクリプト中にテンプレートを入れ込んでしまっても悪くはないが、それをしてしまうとテンプレート部分の修正が面倒だ。かといって、HTMLとスクリプトをペアで使うと、今回のケースのように、管理するファイルの数も増える。

そんなことから、最近のはやりは、HTML中にスクリプトを書き込む方法だ。HTML中にスクリプトを書くといっても、あくまでもサーバサイドでの処理である。サーバ側でスクリプトを処理した結果がクライアント側に渡されるため、スクリプトの具体的な処理内容が見られることはない。サーバサイド処理の仕組みとして、Javaなら JSP、Perl系なら比較的違和感なくシフトできるPHP3が使われるが、JSPにしてもPHP3にしても構築したサーバに自分でインストールしない限り、組み込まれない機能だ。今回はインラインスクリプトの実例としてPHP3を使った例を示そう。

▼リスト1-6 disp.cgi

```
#!/usr/bin/perl

use Pg;
require "psqldb.pl";

&query;
&opendb ('testrel');

$max=&sqlout ("select * from subject,classmaster,content where
subject.cnum=classmaster.cnum and subject.count=$QUERY{value} and
content.count=$QUERY{value} and subject.relopen='True'");

if (!$max) {
        $cnam='ありません';
} else {
        $subject=$result->getvalue(0,4);
        $subject=~ s/ +$//g;
        $content=$result->getvalue(0,8);
        $cnam=$result->getvalue(0,6);
}


print "Content-type: text/html\n\n";
open (READ,"disp.html");

while (<READ>) {
        s/--subject--/$subject/g;
        s/--content--/$content/g;
        s/--cnam--/$cnam/g;
        print;
}
exit;
```

PHP3の入手とビルド、httpd.confの修正

新たな処理系を加えるというと、大変な作業のように思いがちだ。ところが、PHP3はそれほど面倒なものでもない。特に、新バージョン（PHP4）のβテストが始まっているため、PHP3はかなり完成度の高い処理系である。

まず、http://jp.php.net/からPHP3を入手する。執筆時点でのPHP3の最新バージョンは3.0.16である。入手したものを、

```
# tar xvfz php-3.0.16.tar.gz
# cd php-3.0.16
# ./configure --with-pgsql --with-
apxs=/usr/local/apache/bin/apxs
# make
# make install
```

の手順でインストールしていく。configureのときの――with－apxsが重要で、Apacheのダイナミックシェアードオブジェクトを使いモジュール登録する。必ずしも/usr/local/apache/bin/apxsにapxsがあるとは限らないので、それぞれの環境に合わせて――with－apxs＝でapxsのある場所を指定する。Red Hat系の場合、RPMパッケージ「apache-devel」をイ

ンストールすると、/usr/sbinディレクトリにapxsがあるはずだ。

make installをした段階で、/usr/local/bin/phpにPHP3の処理系がインストールされる。次に、Apacheの設定変更が必要だ。httpd.confにある、

```
#LoadModule php3_module modules/libphp3.so
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source .phps
```

の行の先頭にある#を外す。PHP3をmake installした段階で、httpd.confの、

```
AddModule mod_php3.c
```

という行はコメントアウトが外れているはずだが、該当行も一応確認しておこう。

▼リスト2-1 list.php3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
        <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>subject一覧</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<BR>

<TABLE BORDER="1">
        <TR>
                <TH>#</TH>
                <TH>reldate</TH>
                <TH>count</TH>
                <TH>relopen</TH>
                <TH>cnum</TH>
                <TH>relsub</TH>
        </TR>

<?
        pg_connect("","","testrel");
        $result=pg_exec ("select * from subject");
        $max=pg_numrows ($result);
        for ($i=0;$i<$max;$i++) {
                $reldate=pg_result($result,$i,0);
                $count=pg_result($result,$i,1);
                $relopen=pg_result($result,$i,2);
                $cnum=pg_result($result,$i,3);
                $relsub=chop (pg_result($result,$i,4));
                print "  <TR>
                        <TD>$i/$max</TD>
                        <TD>$reldate</TD>
                        <TD>$count</TD>
                        <TD>$relopen</TD>
                        <TD>$cnum</TD>
                        <TD>$relsub</TD>
        </TR>\n";
        }
?>
</TABLE>

</BODY>
</HTML>
```

▼リスト2-2 list.php3を実行したWebブラウザのソース表示結果

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
        <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>subject一覧</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<BR>

<TABLE BORDER="1">
        <TR>
                <TH>#</TH>
                <TH>reldate</TH>
                <TH>count</TH>
                <TH>relopen</TH>
                <TH>cnum</TH>
                <TH>relsub</TH>
        </TR>
        <TR>
                <TD>0/2</TD>
                <TD>Wed Mar 15 00:21:41 2000 JST</TD>
                <TD>1</TD>
                <TD>t</TD>
                <TD>20</TD>
                <TD>4月1日人事情報－速報</TD>
        </TR>
        <TR>
                <TD>1/2</TD>
                <TD>Wed Mar 15 00:43:10 2000 JST</TD>
                <TD>2</TD>
                <TD>t</TD>
                <TD>30</TD>
                <TD>速報の修正</TD>
        </TR>
</TABLE>

</BODY>
</HTML>
```

ここまで作業をしてから、Apacheの再起動をすると、PHP3が使えるようになる。

ディストリビューションによる違いや、Apacheのバージョンの違いなどが原因でうまくインストールできない場合には、phpを展開したディレクトリにあるINSTALL、INSTALL.DSO、INSTALL.REDHATなどのドキュメントを読んでいただきたい。

PHP3とApacheは親密性が高い

DBとの連係でPHP3が着目されている理由というのは、PHP3自体がApacheのモジュールとして動作したり、DBへアクセスする機能を持っているというだけにとどまらず、その処理構造にあるともいえそうだ。

PHP3では、スクリプト部分を“＜？　？＞”で囲むという、独特のスタイルをもっている。それ以外は、純粋に出力されるだけなので、HTML中にスクリプトを記述すれば、HTML単体を管理するだけでスクリプト処理ができることになる。私自身がPHP3を使い込んでいるわけではないので、あまり細かいところまでは説明できないが、PerlからPHP3へのシフトは、手軽にできそうだ。

こうしたPHP3の特徴は、いろいろ記述するよりもサンプルを見たほうが早い。list.php3を参照してほしい（**リスト2-1**）。

動作内容は、subjectテーブルにアクセスして、すべてを表示するというものだ（**画面3**）。リストを見ると、ふつうのHTML中にPHP3のスクリプトが記述されている。スクリプトは“＜？？＞”で囲まれた部分で、この部分はPHP3が処理した内容に合わせて置き換えられる。あくまでもサーバサイドでの処理であって、クライアントが見たときには、ごく普通のHTMLでしかない（**リスト2-2**）。

Perl＋Pg.pmの組み合わせで作ったCGIと使い勝手は同様だ。PHP3では、pg_connectでDBに接続し、pg_execでSQLクエリーをPostgreSQLに引き渡す。pg_resultで得られた結果を引き出せるので、そのままタグを組み込んで表示すればいい。なお、PHP3の場合、chopの扱いが行末のスペースを削除するという動作をする。まさにDBのフィールドに合わせた機能になっている。得られた値をそのままchopすれば、行末スペースの削除ができる。文字数指定のcharフィールドのフィルタリングには最適な仕組みである。

#	reldate	count	relopen	cnum	relsub
0/2	Wed Mar 15 00:21:41 2000 JST	1	t	20	4月1日人事情報-速報
1/2	Wed Mar 15 00:43:10 2000 JST	2	t	30	速報の修正

画面3　list.php3を実行した

PostgreSQLのインターフェイスをWebにする

ここまでややこしい仕組みを作らなくても、もっとも簡単にDBをWebから利用したいという場合がほとんどだろう。起動プロセスが増えるとか、速度が遅いとか、いろいろな問題が必ず出てくるものの、手軽にDBを使うなら、スクリプト中からSQLクエリーをpsqlに渡すCGIという奥の手がある。しかも、psqlはHTMLとしてテーブル出力をする機能があるので、速度を求めないならば、psqlをそのまま使うという方法もないわけではない。

HTMLのFORM機能を使って手抜きともいえるpsqlをそのまま利用したのが、psql.htmlとpostquery.cgiだ（**リスト3**）。psql.htmlを表示させたのが**画面4**で、CONNECT DBに接続するDBテーブルの名前を、SQL QUERYにSQL文を入力し、OUTボタンを押すと、postquery.cgiを実行する。potquery.cgiでは、その入力されたデータを利用して/usr/bin/psqlを呼び出している。そして、Content-typeを送り出し、psqlを呼び出して得られた結果を出力するというものだ（**画面5**）。

HTMLのタグは、psqlを呼び出すときに、-Hと-Tオプションを付けることで、psqlが処理をする。-Hを付けて起動すると、戻す形式がTABLEタグとなる。これに、-T borderを付けて、ボーダ指定で呼び出せば、得られた値を＜TD＞内に組み入れてというややこしいことを考えずにTABLE表示される。得られた出力をそのままHTML内に組み込めば、それなりのものができあがる。

ただし、単純にSQLをそのまま渡してしまうと、得られる結果が悲惨なものになる。たとえば、フィールド数が多いものの場合、単純にpsqlで渡してしまうと、表示まで時間がかかる。そこで、例では最低限の追加として、selectクエリーの検索結果の件数を制限するlimitを加えている。psqlを呼び出すとき、limitが数値以外であれば、SQLクエリーの最後にlimit指定数が加わる。

psql.htmlではデータベース名と、SQLクエリーを入れて、ボタンを押せば、そのままpsqlを呼び出して表形式で出力する。単純なSQLを使うだけならば、telnetをする必要がなくなり、手軽にデータベースにアクセスして、その出力をHTMLで得られる。

DBに貯えた全データをHTML形式に変換する。

手作業でHTMLを作っていては、パブリッシングまでの時間がかかりすぎる。しかし、ニュースサイトのように最新のページにアクセスが集中する場合、DBを毎回アクセスしていては効率が悪い。そこで、実際のニュースサイトでは、DBでデータを貯えて管理するものの、データ登録と同時にHTMLを吐き出し、HTMLだけをWebサーバにミラーするという仕組みになっている。

このことによるメリットは、WebサーバでCGIを動かさないようにでき、サーバのパフォーマンスを確保できることと、バックエンドでDBが管理をしているので、修正やデータの追加が手軽にできるという2つだ。たとえページデザインを変更するとしても、全ファイルをサーチして、すべてのHTMLを出力し直すというスクリプトを書けば済む。エンドユーザーとつながるところまでDBを使う必要はないので、気がつきにくいものであるが、最新情報を追っているWebサーバでは、コストパフォーマンスを上げるためにこうした使い方をしていることが多い。さらに、コンテンツをピュアなテキストで管理できることで、以後のデータの流用も手軽にできる。

また、GooやInfoseekといったWeb検索サイトのロボットプログラムは、CGIページ（.cgiで終わるURL）を検索対象にしないようなので、ニュースリリースなどの公開したい情報はHTMLファイルに変換しておいたほうがよいだろう。

▼リスト3-1 psql.html

```
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>

<HEAD>
        <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>QUERY OUT</TITLE>
</HEAD>

<BODY>

<FORM ACTION="postquery.cgi" METHOD="POST"
ENCTYPE="application/x-www-form-urlencoded">
<P>CONNECT DB<BR>
<INPUT TYPE="TEXT" NAME="dbname" SIZE="25"><BR>
<BR>
SQL QUERY<BR>
<TEXTAREA NAME="query" ROWS="4" COLS="62"></TEXTAREA><BR>
<BR>
LIMIT  <INPUT TYPE="TEXT" NAME="limit" VALUE="10"
SIZE="2"><BR>

<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="OUT">
</FORM>

</BODY>

</HTML>
```

▼リスト3-2 postquery.cgi

```
#!/usr/bin/perl

require "../jcode.pl-2.11";

read(STDIN, $buf, $ENV{'CONTENT_LENGTH'});
foreach $pair (split(/&/, $buf)) {
        ($name, $value) = split(/=/, $pair);
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",
hex($1))/eg;
        &jcode'convert(*value,'euc');
        $value =~ s/\+/ /g;
        $value =~ s/[\r\n]//g;
        $QUERY{$name}=$value;
}

if ($QUERY{limit}) {
        $QUERY{query} .=" limit $QUERY{limit}";
}
$querystr="/usr/bin/psql -H -T border $QUERY{dbname} -c
\"$QUERY{query}\"";
$getquery=`$querystr`;

print "Content-type: text/html\n\n$getquery";
```

通常はDBを別サーバに立てる

ひとまず、1台のマシンでPostgreSQLとApacheを共存させる構成を取ったが、これが商用サーバとなるとそういうわけにはいかない。ある程度のアクセス数が見込まれるならば、WebサーバとDBを分割し、Apache、DB共に最適のパフォーマンスで動くような設計にする。また、DBサーバを分離することで、DBのバックアップ処理やメンテナンスをWebサーバとは違ったタイミングで処理できることも大きい。Webサーバが停止したときにはどうにもならないが、DBが万一停止した場合でも、Webサーバは独立して動かしていられる。

また、絶対に避けられないメンテナンスとして、VACUUM作業がある。PostgreSQLでは、VACUUMによってDBの不要レコードの解放や、DBの最適化が行われる。そのため、

```
$ psql -c 'vacuum' DB名
```

という作業をときどき行うことになるが、VACUUM実行中には、VACUUMが作業しているテーブルへのアクセスができなくなる。閲覧者側のWebサーバにDBと完全に密接させたCGIが動いていると、DBのメンテナンスと同時にApacheも必然的に停止することになる。

デイリーで10万ページビューを越えるようなサイトならば、DBとWebサーバを分割し、DBは独立してメンテナンスができるような状態が必須だ。しかもバックアップサーバに常にデータをバックアップしておくような構成を取っておいたほうがいい。PostgreSQLならば、pg_dumpやpg_dumpallというバックアップツールがあるため、cronで、

```
pg_dumpall -o >testrel.out
```

といった形で定時バックアップを取り、バックアップサーバにtestrel.outを移してから、バックアップサーバ側でインポートし直すという形で同一のDBを持つように動作させるとよいだろう。バックアップサーバ側で、

```
$ destroydb testrel
$ createdb testrel
$ psql -f testrel.out testrel
```

とすることで、バックアップDBサーバ側でも主DBサーバと同じDB構成を構築できる。こうしておけば、万一、主DBサーバに障害が起きたとしても、バックアップDBサーバと主DBサーバを入れ替えて復旧できる。ただし、pg_dumpでバックアップしたデータをインポートし直す場合、安定性が多少落ちる。バックアップは確実に取れているようだが、インポート時にDBを破壊するケースがあるようだ。

もし復旧できなかったとしても、エクスポートしたデータには問題がないようなので、initdbで初期化し直してインポートすれば、たいていは問題が解消する。cronによる自動化処理では落とし穴があるので、このあたりには注意したほうがいい。

これ以外にも、主DBにデータを書き込むと同時にバックアップDBサーバにも同時にデータを書き込むという二重化もあるが、どれぐらいクリティカルな情報であるかという点から、バックアップ方法を検討すべきだろう。

HTMLベースから、DBベースでのWebパブリッシング手法に切り替えると、個々のHTMLファイルを作る必要がなくなる。ローカルにはHTMLファイルがほとんど存在しないというケースが出てくるので、DBの管理に注意しておかないと、万一の障害時に復旧できなくなることだけは覚えておこう。

画面4 SQL文をWebブラウザから入力できる

画面5 SQL検索の実行結果

Linuxファイルシステムの現在と未来

Linuxでは長らくext2がネイティブファイルシステムとして使用されてきたが、最近ではext3やReiserFSなど次世代ファイルシステムの開発も進んできた。また、SGIのXFSやIBMのJFSなど、企業が自社ファイルシステムをLinuxへ移植するニュースも話題になり、Linuxのファイルシステムが面白いことになってきている。そこで、Linuxの各ファイルシステムと他のOSとの比較などを解説していきたいと思う。

第1回　次世代ファイルシステムとext2

文：長岡モイチ
Text :Moichi Nagaoka

Linuxをめぐる次世代ファイルシステムの争い

まずは、最近のLinux界隈で話題になっている次世代ファイルシステムについて整理しておこう。ここでは、以下のファイルシステムを取り上げる。

- **ReiserFS**
- **Ext3**
- **SGI XFS**
- **IBM JFS**

■ReiserFS

ReiserFSはext2などが採用している伝統的な固定ブロックサイズのUNIXファイルシステムの構造とは異なる、バランス木アルゴリズムを用いたまったく新しいファイルシステムである。ちなみに、ReiserFSの名前は開発者のHans Reiser氏から来ている。ReiserFSはSuSEがサポートしていることから、SuSE Linuxを使っている人はすでにReiserFSを使っているかもしれない。ReiserFSは当初ジャーナリング機能を持たなかったが、ジャーナリング機能を持つようになって注目を浴びるようになった。ジャーナリング機能とは、ファイルシステムに対する変更をトランザクションログとして記録し、障害時に復旧の手助けをするためのものである。現在活発に開発が進められており、安定度も増し次期開発版カーネルのツリーに統合される可能性が高い。現在はカーネルに対するパッチとして配布されている。

■Ext3

Ext3はその名前の通りext2ファイルシステムをもとにジャーナリング機能を追加したファイルシステムであり、ファイルシステムの機能や内部構造はext2とほぼ同じものとなっている。ファイルシステムの内部構造がext2と同一のため、既存のext2パーティションを再フォーマットせずにext3として運用することが可能だ。よくも悪くもext2の性格を受け継いでいるといえよう。最新バージョンは0.0.2dという数字からもわかる通りジャーナリング機能は開発途上である。こちらもReiserFS同様、カーネルに対するパッチとして配布されている。

■SGI XFS

XFSはもともとSGIのUNIX互換OSであるIRIX 5.3から採用されているIRIXのネイティブファイルシステムであり、パフォーマンス、スケーラビリティともに優れた次世代のファイルシステムとして知られている。

XFSがオープンソース化されてLinuxに移植されることになったので、特に企業ユースとしてLinuxを使いたい人から大きな期待が寄せられている。SGIはつい最近、開発版カーネルであるカーネル2.3をターゲットにしたLinux版XFSを公開した。

■IBM JFS

IBMもSGIと同様Linuxをターゲットとしたオープンソースプロジェクトに力を入れており、IBMのUNIX互換OSであるAIXのネイティブファイルシステムであるJFSを現在移植中である。

JFSはAIXのネイティブファイルシステムとして実績のあるファイルシステムであり、ジャーナリング機能を搭載している。JFSは現在、カーネルに対するパッチとして配布されており頻繁にバージョンが上がってはいるものの、まだファイルシステム操作の基本機能のみ動作している状態のようだ。

次世代ファイルシステムの必要性

ext2ファイルシステムがLinuxのネイティブファイルシステムに採用されてからすでに7年以上経過していると思われるが、その間ext2は小さな改良を続けてきたものの基本設計は変わっていない。これまで一般のLinuxユーザーにとってext2に取って代わるネイティブファイルシステムの必要性がなかったともいえる。

ところがLinuxが企業ユーザーにも受け入れられるようになり、Linuxが搭載されたサーバがエンタープライズレベルの大規模な用途にも使用されるようになってから、現在のext2ファイルシステムのパフォーマンス、信頼性、スケーラビリティが気になるようになってきた。SGIのXFSやIBMのJFSの移植は、「Linuxを将来のUNIXビジネスの中核に持ってくるためにはよりハイパフォーマンスなファイルシステムが不可欠である」という企業の戦略的判断が当然あったと思われる。

また、IDEハードディスクの大容量化が加速していく最近では、個人ユーザーが手軽に入手できるディスク容量も20Gバイトを超え、この現象は今後も続いていくものと思える。そうなると、1Gバイト前後のパーティションではたいして気にならなかった問題も、10Gバイトを超えると重大になってくる。次世代ファイルシステムは企業ユーザーのみならず、我々個人ユーザーにも深く関係する問題なのだ。

しかしながら、ほとんどの個人ユーザーは現在のext2を使っていて不満に思うことはほとんどないはずだ。また、次世代ファイルシステムに共通のキーワードであるジャーナリングとはいったい何なのか、ext2とどこがどう違うのか疑問に思う読者もいるかと思う。そこで、現在のLinuxにおいてネイティブファイルシステムであるext2の実装に関して取り上げ、次回以降では次世代ファイルシステムの実装について紹介していきたいと思う。

ext2ファイルシステムの概要

ext2ファイルシステムは、Linuxの初期に採用されていたMINIXファイルシステムの制限から脱出すべく開発されたextファイルシステムをさらに改良したものであり、長い間ネイティブファイルシステムとして使われている。ext2ファイルシステムは当時最も一般的であったBSDのFFS（Fast File System）の実装に強い影響を受けている一方、完全互換というわけではなく、FFSとは異なるポリシーや、FFSにはないext2独自の機能もある。

ext2ファイルシステムは可変ブロックサイズ、255文字までのロングファイルネーム、シンボリックリンクサポートなど、UNIXファイルシステムの基本的な機能はすべて実装されている。FreeBSDなどのBSD系OSはBSDで開発されたFFSのコードをベースに拡張されてきた、いわば伝統の血が流れたファイルシステムであるが、LinuxはLinus氏がMINIXを参考にスクラッチからつくりあげたOSであり、ext2ファイルシステムはその祖先にバークレイではなくMINIXの血が流れている。ただし実際はMINIXファイルシステムは構造的にも機能的にも非常に貧弱であり、ext2の内部を見る限りではext2の設計と実装のモデルとなったのはFFSであると思われる。

表1にext2とFFSの比較を示す。

ext2の内部構造

ext2のファイルシステムのレイアウトを**図1**に示す。ext2ファイルシステムの内部はブートブロックに続き、ブロックグループと呼ばれる領域が繰り返すような構造になっている。ブロックグループはデフォルトではブロックサイズ（4,096バイト）の8倍をブロック数とみなした大きさであり、中身は管理領域とデータ領域で構成されている。ext2ではファイルシステムの領域をブロックグループ単位で区切り、それぞれのブロックグループに管理領域を分散させている。

ブロックグループの管理領域は、スーパーブロックの冗長なコピー、ブロックグループ内の空きブロック数や空きi-node数を管理する領域、空きブロックや空きi-nodeをビットマップで管理するビットマップ領域、データを格納するデータ領域から構成されている。次に、これらの領域を順に説明する。

仕様	ext2	FFS
最大ファイルシステムサイズ	4Tバイト	1Tバイト
最大ファイルサイズ	2Gバイト	2Gバイト
最大ファイル名	255	255
ブロック	1、2、4Kバイト	4、8Kバイト
フラグメント	なし	0.5、1、2、4、8Kバイト

表1　ext2とFFSの比較

■スーパーブロックとブロックグループディスクリプタ

スーパーブロックはファイルシステムの基本的なパラメータを保持する領域である（**図2**）。スーパーブロックは主にファイルシステムを一度構築したら変更されない静的な情報が格納される。ファイルシステムの総ブロック数や総i-node数、ブロックサイズやフラグメントサイズ、シリンダグループやブロックグループあたりのブロック数などである。ただし、すべてが静的というわけではなく、たとえばファイルシステム全体の空きブロック数や空きi-node数はスーパーブロックに管理されているので、それらのパラメータに変化があった場合はディスク上のスーパーブロックの内容も更新される。

また、各ブロックグループに存在するブロックグループディスクリプタは、ブロックグループ固有の情報が格納されている領域である（**図3**）。ブロックグループディスクリプタにはi-nodeビットマップ、ブロックビットマップ、i-nodeテーブルといった後続する管理領域に対するポインタ（ブロック番号）の静的な情報と、ブロックグループ内の空きブロック数、空きi-node数、作成されたディレクトリ数といった動的な情報が含められている。

■ i-nodeビットマップとブロックビットマップ

ブロックグループディスクリプタのあとに続く管理領域として、ブロックグループ内の全i-nodeと全ブロックの使用状態を保持しているi-nodeビットマップとブロックビットマップが存在する。「ビットマップ」という言葉通り、ビットマップ領域の各ビットがi-nodeやブロックの1つ1つを表している。ビットが0の場合は使用可能で、1の場合はすでに使用済みであることを表す（**図4**）。dumpe2fsコマンドを使えばファイルシステムの管理領域の内容を出力することができ、現在のビットマップ状態も表示することができる。ファイルシステムの作成直後はビットマップは0ばかりのきれいな状態だが、ファイルシステムを使用していくに従いファイルシ

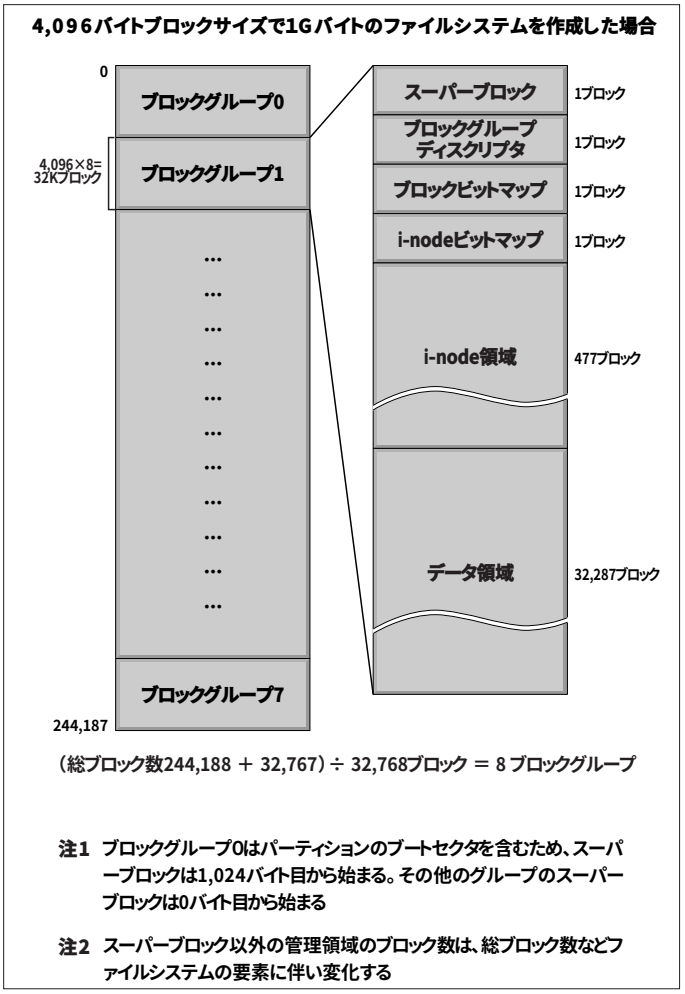


図1 ext2ファイルシステムのレイアウト
4,096バイトブロックサイズで1Gバイトのファイルシステムを作成した場合

__u32 s_inodes_count	総i-node数
__u32 s_blocks_count	総ブロック数
__u32 s_r_blocks_count	予約ブロック数
__u32 s_free_blocks_count	総空きブロック数
__u32 s_free_inodes_count	総空きi-node数
__u32 s_first_data_block	最初のデータブロックNo
__u32 s_log_block_size	ブロックサイズ
__s32 s_log_frag_size	フラグメントサイズ（未使用）
__u32 s_blocks_per_group	ブロックグループあたりのブロック数
__u32 s_inodes_per_group	ブロックグループあたりのi-node数
__u32 s_mtime	ファイルシステムマウント時刻
__u32 s_wtime	ファイルシステム更新時刻
__u16 s_mnt_count	マウント回数
__s16 s_max_mnt_count	fsckまでの最大マウント回数
⋮	

図2 スーパーブロックのレイアウト

__u32 bg_block_bitmap	ブロックビットマップのブロックNo
__u32 bg_inode_bitmap	i-nodeビットマップのブロックNo
__u32 bg_inode_table	i-node領域の先頭ブロックNo
__u16 bg_free_blocks_count	空きブロック数
__u16 bg_free_inodes_count	空きi-node数
__u16 bg_used_dirs_count	ディレクトリ数
__u16 bg_pad	余白
__u32 bg_reserved[3]	予約

図3 ブロックグループディスクリプタのレイアウト

ステムがフラグメント化してしまい、ビットマップの状態も0と1が混在した状態になる。

もしも使用しているファイルシステムが遅いと感じるならば、dumpe2fsコマンドで出力される各ブロックグループごとのFree blocksやFree inodesの部分をチェックしてみるとよい。

■ i-node領域とデータ領域

i-node領域はi-nodeエントリが格納されている領域で、ファイル、ディレクトリ、リンクなどのファイルシステム上のオブジェクトを管理する領域である。データ領域はブロックサイズごとに区切られたファイルなどの中身が格納されている。データ領域には完全にデータの断片しか存在せず、データ領域のそれぞれのブロックがどのような内容であるのかはi-node領域にあるi-nodeエントリで管理されている。

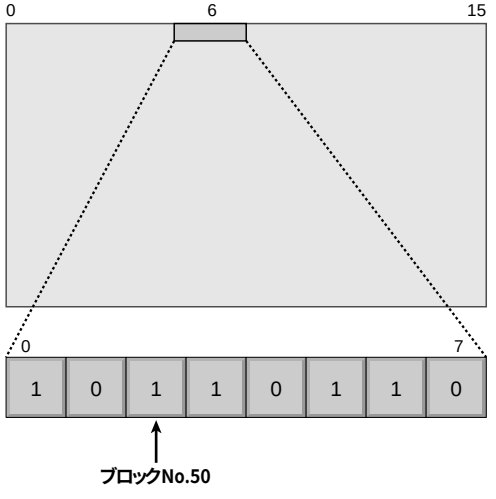
ファイルシステム中のオブジェクトについて

それでは、実際にファイルやディレクトリがどのように保存されているのか詳しく解説していこう。

■ i-node

i-nodeとは、ファイルシステム中に存在するファイルやディレクトリなどのオブジェクトの実体といえる。ここで注意してほしいのは、i-nodeの中にはファイル名を格納する場所がないということだ。このことは、たとえばあるファイルをアクセスしたい場合にユーザーはファイル名を指定するが、OSの内部ではファイル名をいったんi-node番号に変換し、i-node番号でそのファイルを識別する。

i-nodeには**図5**のような内容が含まれている。ls -lコマンドを実行すると出てくるようなファイルの属性が格納されていることがわかる。ちなみに、ファイル名しか出てこないlsコマンドとファイルの属性も表示させるls -lコマンドは、実は内部でアクセスするデータブロックの範囲が違うことがわかると思う。lsでは表示対象のディレクトリをたどって、そのディレクトリに含まれているファイル名を表示するだけでよいが、ls -lで属性まで表示させようとすると、ディレクトリに含まれているファイル名とi-node番号のデータ



例：ブロックNo.50の状態を求める場合

1）ビットマップ領域からブロックNo.50の状態を含むバイトを求める

50 ÷ 8 ＝ 6（バイト）

よって、第6バイトの内容を読み込む

2）ブロックNo.50のバイト中のオフセットを求める

50 mod 8 ＝ 2（ビット）

よって、ビット2の状態を調べる

1 ＝ 使用
0 ＝ 未使用

3）ビット2の値からブロックNo.50は使用中であることがわかる

図4　ビットマップ領域

i_mode	ファイルモード
i_uid	ユーザーID
i_size	サイズ（バイト）
i_atime	アクセス時刻
i_ctime	属性変更時刻
i_mtime	データ変更時刻
i_dtime	削除時刻
i_gid	グループID
i_links_count	リンク数
i_blocks	ブロック数
i_flags	ファイルフラグ
osd1	OS依存部1
i_block	ブロックのポインタ配列（15エントリ）
i_version	ファイルバージョン
i_file_acl	ファイルアクセス制御
i_dir_acl	ディレクトリアクセス制御
i_faddr	フラグメントアドレス
osd2	OS依存部2

図5　i-nodeの構造

だけでは足りず、各ファイルのi-node番号に対応するi-nodeエントリをさらに取得しなければならない（**図6**）。

i-nodeエントリに話を戻そう。i-nodeエントリにはファイルモード（rwx）、そのファイルのユーザーID、グループID、サイズ、3種類のタイムスタンプなど、ファイルの属性のほかにデータブロックへのポインタを格納するための配列がある。i-nodeにはファイルをはじめとするファイルシステム上のオブジェクトを表すのに必要な情報がすべて入っているといえる。ファイルに対して何か操作したい場合はi-nodeが必要になり、これがi-nodeがファイルの実体であるという意味である。

データブロックへのポインタの配列の先頭の12要素は直接ブロックポインタと呼ばれる。直接ブロックポインタにはデータブロックの論理ブロック番号が格納される。たとえば、ブロックサイズが4,096バイトの場合、配列の先頭の要素にはデータの先頭4,096バイトのデータが含まれるブロックのブロック番号が格納されている（**図7**）。

配列の13番目の要素は第1間接ブロックポインタと呼ばれ、データブロックではなく、間接ブロックポインタと呼ばれるデータブロックを指すポインタだけを含む特別なブロックのブロック番号が格納される。間接ブロックポインタはファイルのデータそのものではなく、データブロックのありかを指すポインタ（ブロック番号）が格納されている特別なブロックである。同様に14番目、15番目の要素はそれぞれ第2間接ブロックポインタ、第3間接ブロックポインタと呼ばれ、データブロックを得るために2段、3段の間接ブロックポインタをたどっていく必要がある。間接ブロックポインタの実体は、4バイト（32ビット）のブロック番号を格納するポインタの羅列である。したがって、ブロックサイズが4,096バイトの場合、4,096÷4＝1024個のポインタが格納されていることになる。

■ディレクトリ

ext2のディレクトリの内部構造を**図8**に示す。ディレクトリはディレクトリエントリから構成されている。ディレクトリエントリはファイル名とi-nodeの対応（リンク）表であり、カーネルは与えられたファイル（パス）名をi-node番号に変換するときにこのディレクトリエントリの内容を使用する。ディレクトリもディレクトリエントリによって管理されており、ファイルとの区別はない。名前にリンクされたi-node番号で示されるi-nodeの中に、ディレクトリであることを示す属性が付いている。

たとえば、カーネルが/var/log/

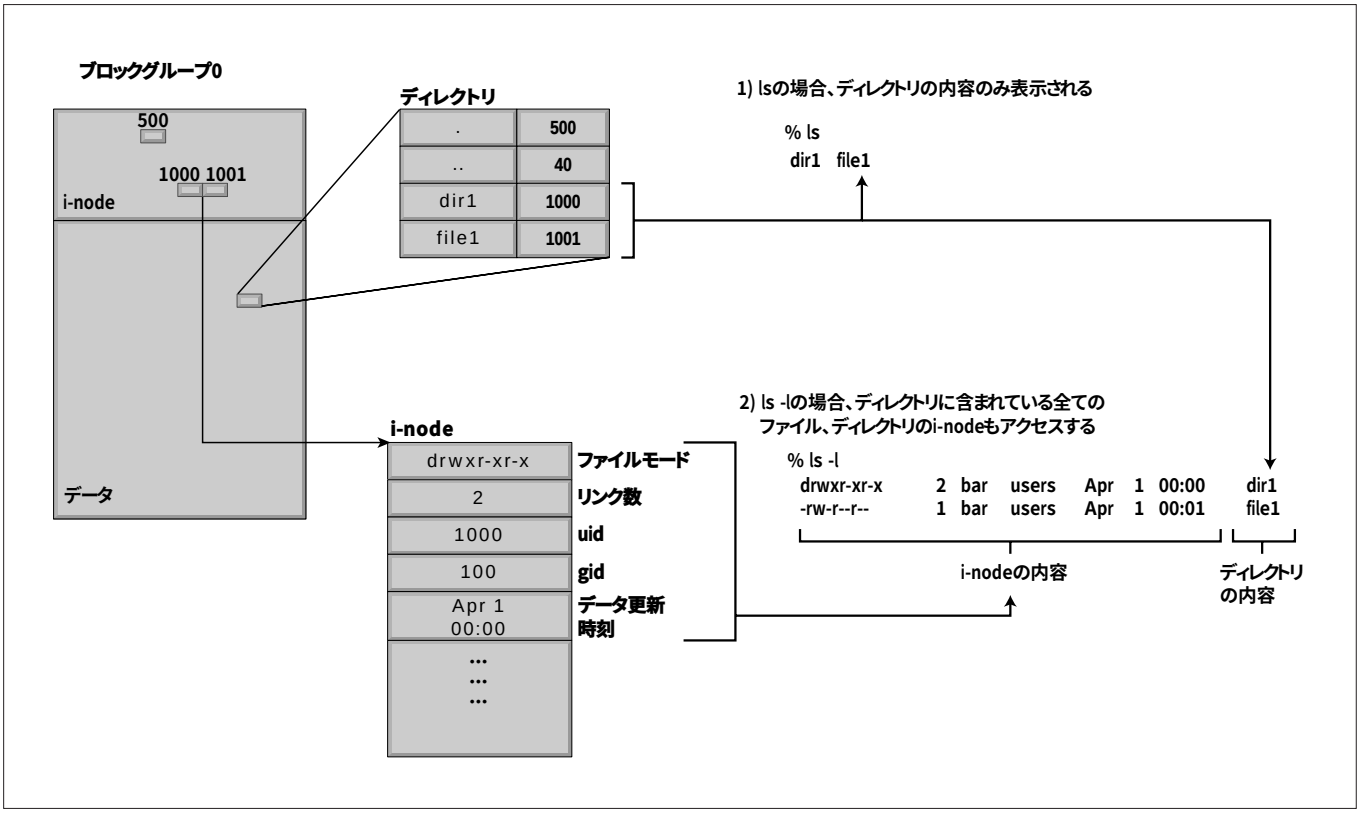


図6　lsコマンドとls -lコマンドによるアクセス内容の違い

messageファイルの中身を読みたいときは、まず最初にrootディレクトリからvarのディレクトリエントリを検索し、対応するi-node番号を使ってi-nodeエントリの物理的位置を求めてi-nodeの内容にアクセスする。i-nodeの情報からvarディレクトリのデータブロックにアクセスし、同じようにvarディレクトリのディレクトリエントリからlogの名前のエントリを検索する。最後にlogディレクトリからmessageという名前のエントリを検索し、対応するi-node番号を使ってファイルmessageの中身にアクセスできるわけだ。

リソース割り当て（アロケーション）について

前節でext2ファイルシステムの内部構造について理解できたものと思う。次は、ファイルの書き込みによってファイルシステム内のi-nodeやブロックがどのように割り当てられるか見てみよう。

ext2におけるファイルデータの格納単位はブロックである。ファイルのデータはブロックサイズに分割され、ファイルシステム上のデータブロック領域に格納される。ブロックサイズごとに分割されたデータの断片は、i-node上のブロックポインタ配列によってつながれている。i-nodeのブロックポインタが破壊されたらデータの断片をつなぐリンクがなくなってしまうので、データを復旧する手だてはほとんどない。

現在のファイルシステムのリソース状況はスーパーブロック、各ブロックグループのブロックグループディスクリプタ、ブロックビットマップ、i-nodeビットマップで管理される。

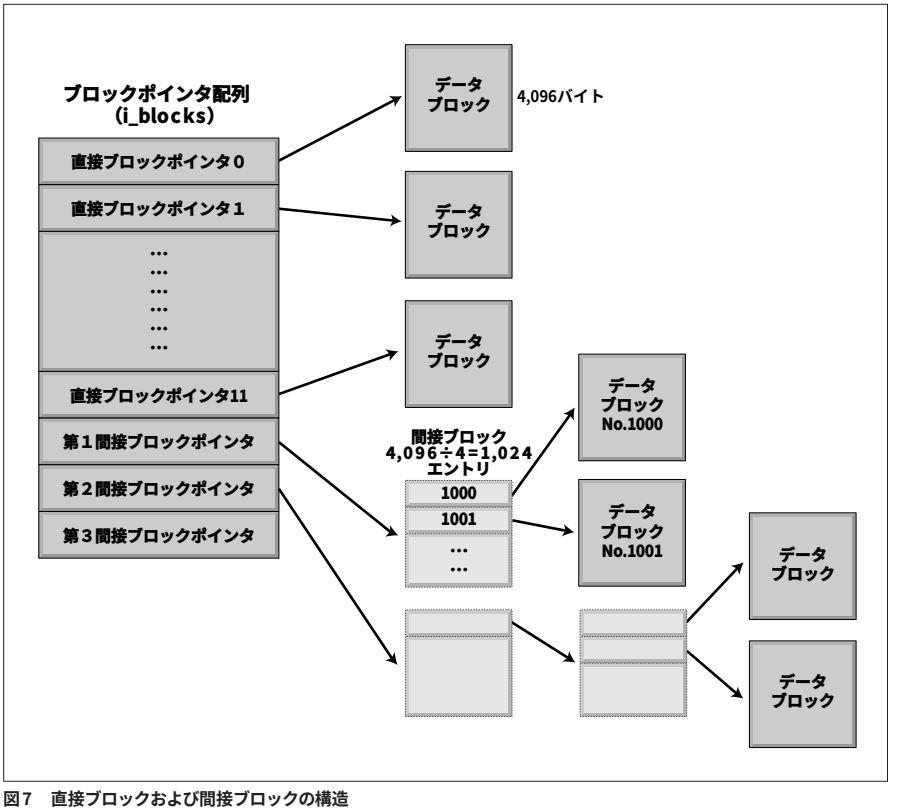


図7　直接ブロックおよび間接ブロックの構造

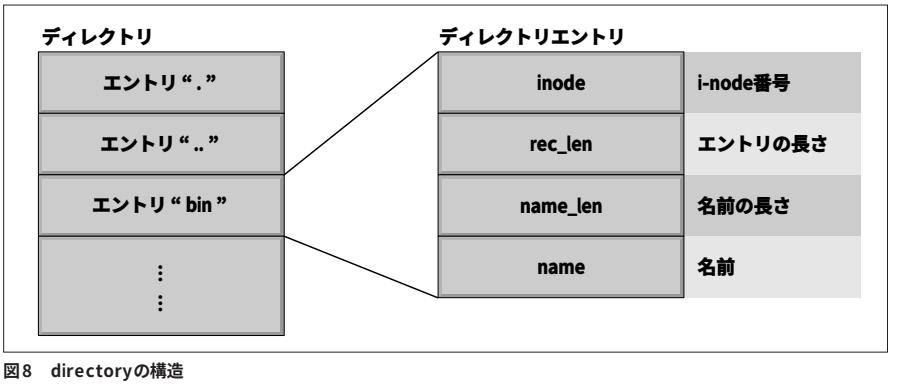


図8　directoryの構造

■ext2におけるリソース割り当ての流れ

あるアプリケーションがファイルを新しく作成し、そのファイルにデータを書き込んだことを想定してみよう。まず、あるディレクトリ内に新しくファイルを作成すると、ファイルシステムから空きi-nodeが割り当てられる。次に、ディレクトリにファイル名とi-nodeを対応させたディレクトリエントリが追加される。

ファイルを作成した直後はサイズが0バイトでありデータブロックは割り当てられていない。この状態でファイルにデータを書き込むと、データブロックが割り当てられ、i-nodeのブロックポインタ配列にセットされる。また、ファイルのサイズやタイムスタンプが変更されるので、i-node内のデータも更新される。

空きi-nodeやデータブロックがどのように割り当てられるかというアルゴリズムについては後述する。

ファイル、シンボリックリンク、ディレクトリなどのファイルシステム内のオブジェクトを作成すると、i-nodeが1つ消費される。また、ファイルにデータを書き込んだりディレクトリにファイルやシンボリックリンクのエントリ

を追加するに従い、データブロックがブロックサイズ単位で消費されていく。

■リソース割り当ての戦略

i-nodeやデータブロックはどのような方法で割り当てられていくのだろうか。まず、データブロックに関しては、ファイルのi-nodeと同じブロックグループから空きブロックを割り当てようとする。i-nodeとそのデータをディスク上のなるべく近い位置に配置することにより、ヘッドのシーク時間を短くしてスループットを上げるためだ。もしも空きブロックがそのブロックグループ内にない場合は、残りのブロックグループをリニアに検索して空きブロックを求めていく。

では、ブロック割り当ての指針となるi-nodeの位置はどのように決定され割り当てられるのだろうか。i-nodeの割り当てアルゴリズムはディレクトリの場合とそれ以外の場合で異なる(**図9**)。普通のファイルやシンボリックリンクの場合はi-nodeの割り当てはそのエントリが属するディレクトリのブロックグループと同じところから割り当てようとする。これは先ほどのi-nodeとデータブロックの関係と同じであり、ディレクトリに属するファイルをなるべくディスク上の近い位置に配置することによりヘッドのシーク時間を短くしようとする目的のためである。

しかしすべてのオブジェクトを1つのブロックグループに割り当てると、すぐにブロックグループのリソースを使いきってしまうことになる。このような状態でブロックグループ内にあるファイルにデータが書き込まれると、他のブロックグループから空きブロックを調達してくることになり、ファイルサイズが増加するに伴いi-nodeとブロックの距離が離れていく結果となる。

したがって、ディレクトリの場合はi-nodeの割り当てにはブロックグループを均一に使用するための異なるアルゴリズムが使用される。ディレクトリの作成をトリガとしてリソースの割り当てをファイルシステム全体に分散させるのである。

その方法は、まず各ブロックグループの空きi-node数の平均を求め、空きi-nodeが平均以上あり、かつ最も空きブロック数の多いブロックグループからi-nodeが割り当てられる。このアルゴリズムにより、ディレクトリを作成するたびに異なるブロックグループが割り当てられるようになり、リソースの割り当てがファイルシステム全体に広がるようになる反面、同じディレクトリ内のファイルは局所化されてシークによってパフォーマンスが低下するという問題を抑えることができる。

■プリアロケーション

あるファイルに対するブロックの割り当て時に、そのファイルがあとで大きくなったときに連続したブロックが割り当てられるよう、あらかじめ連続したブロックを予約するプリアロケーションという手法も用いられている。デフォルトでは割り当て対象ブロックの次のブロックから最大7つの連続したブロックを予約し、そのファイルがopenされている間は保持されている。なるべく連続したブロックを割り当てることにより、読み込み時のスループットを上げようという戦略だ。

ext2ファイルシステムの問題点

以下の視点から、ext2ファイルシステムの問題点について考えてみよう。

■ディスク使用率

固定ブロックサイズによるデータ管理方法は、ファイルサイズがブロックサイズの倍数でない場合は最後のブロックに必ず未使用の領域ができてしま

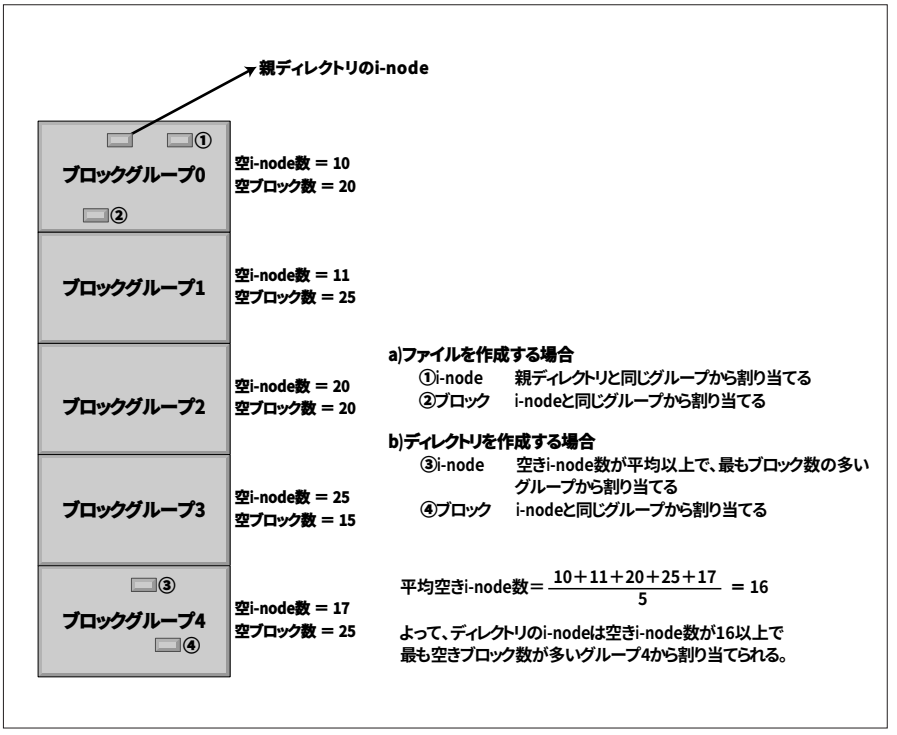


図9 リソース割り当ての戦略

うという問題がある。特に小さなサイズのファイルを大量に作成する場合、未使用部分の割合が多くなり、場合によってはディスクの使用率が50%以下になってしまう。

FFSではフラグメントという概念によりブロックをさらに分割して割り当て単位を小さくすることによりこの問題に対処しているが、ext2ではフラグメントは実装されていないので、ブロックサイズの前後のファイルを大量に作成するような場合はブロックサイズを小さめに調整する必要がある。

■パフォーマンス

ext2やそのモデルのFFSでは空き領域をビットマップで管理しているので、ファイルの作成と削除を繰り返していくうちにファイルシステムの内部は使用領域と空き領域が虫食い上に混在したフラグメント化の状態となる。ファイルのデータブロックがファイルシステム上に分散するとアクセスにたびたびシークが必要となり、パフォーマンスは大幅に低下してしまう。ディスクI/Oのパフォーマンスはいかにシークを少なくしてデータをディスクから連続的に読み書きできるかいうことで決定されるからだ。

フラグメント化はプリアロケーションによりある程度は抑えられるが、ファイルシステムの使用率が上がると連続してブロックを割り当てられなくなり、プリアロケーションの効果も期待できなくなる。

また、最大ブロックサイズが4,096バイトというext2の制限も8,192バイトブロックが標準であるUFSなど、他のファイルシステムに比べてパフォーマンス的に不利となる。

■信頼性

ext2ファイルシステムは信頼性の面でFFSなどに比べて弱いと思われる。その理由は、ext2の書き込み方式によるものだ。ext2では非同期書き込みを前提としており、デフォルトではすべてのデータは非同期でディスクに書き込まれる。すべてのデータとは、スーパーブロックやビットマップなどの管理領域、i-nodeエントリ、ファイルのデータなどすべてを意味する。ext2のモデルとなったFFSやその派生であるUFSは、ビットマップ領域、i-nodeエントリ、ディレクトリエントリなどのメタデータはデフォルトでは同期書き込みで行われる。システムクラッシュが発生した場合にファイルシステムへのダメージを最小限に抑えるためだ。

また、ext2にはファイルシステムへの同期書き込みを指定するため、mountコマンドにsyncオプションが存在するが、このオプションはFFSやUFSなど、他のファイルシステムでの同期書き込みとは意味が異なる。FFSではsyncオプションはデータも含めた同期書き込みの指定を意味するが、ext2ではsyncオプションを指定してもメタデータの同期書き込みだけで、データブロックに対しては同期書き込みが行われない。ext2においてデータブロックに対しても同期書き込みを行うには、明示的に毎回アプリケーションからsync操作を実行する必要がある。

ext2のこの仕様は信頼性よりもパフォーマンスを重視した結果だが、企業サーバのファイルシステムとしてext2を使用する場合は注意が必要だ。

■クラッシュリカバリ

ext2に限らずビットマップによるリソース管理を行っているファイルシステム全般に当てはまることだが、ビットマップによる管理方式はディスク上にデータの整合性をチェックする情報が存在しないので、クラッシュ後のファイルシステムチェックではすべての管理領域を走査してファイルシステムの整合性をチェックする必要がある。最近のハードディスクの大容量化に伴い、大容量ファイルシステムに対するfsckの時間が30分以上かかるケースも出てきた。このfsckによるダウンタイムの増加は運用性を重視するサービスにおいては問題となる。この問題に対処するには、ジャーナリング機能を備えたファイルシステムを使用する必要がある。

次回は

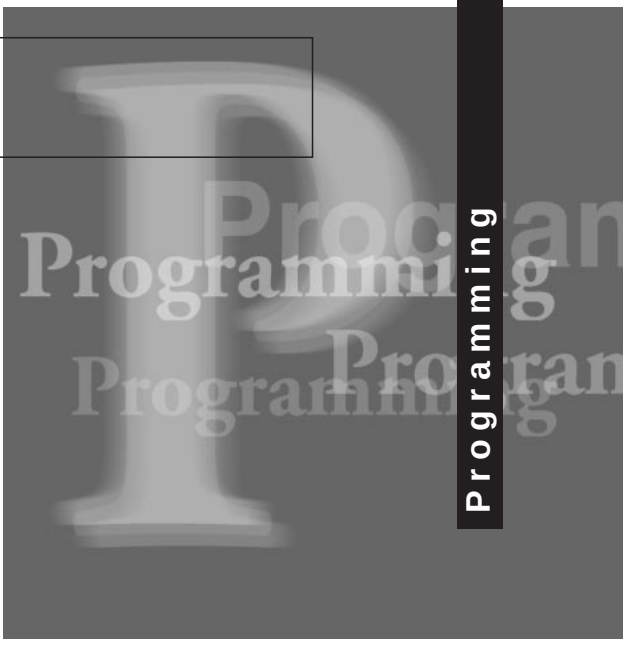
以上、今回は次世代ファイルシステムの紹介と、ext2ファイルシステムの構造と問題点について取り上げてみた。次回は、次世代ファイルシステムの実装について紹介する予定である。

Column

Linuxファイルシステムの歴史

記事内にあるように、ext2ファイルシステムは、Linuxネイティブファイルシステムだが、Linuxの初期のバージョンでは、MINIXのファイルシステムが採用されていた。しかし、扱えるディスク容量やファイルサイズ、ファイル名の長さの制限などさまざまな問題があったため、extファイルシステムが採用されるようになった。さらに、このextファイルシステムを拡張したものが現在のext2ファイルシステムである。255文字までのロングファイルネーム、シンボリックリンクサポートに加え、最大2Gバイトまでのファイルを扱うことができるようになった。ディスクの最大容量は4Tバイトとなっている。ちなみにext2はSecond Extended File Systemを省略したものである。

プログラミング工房



Xには、ネットワーク透過、オープンな環境による高パフォーマンス、変更可能なユーザーインターフェイスなどの優れた特徴がある。今回は、実際のプログラミングを通して、Xがどのようにネットワークトラフィックを節約しているかについて見ていく。

第7回　Xのプログラミング（2）

文：藤沢敏喜
Text : Toshiki Fujisawa

先月号ではXの歴史とその設計思想について解説し、Xでの描画はネットワークを通じてXプロトコルでやり取りされ、オープンな環境でも高いパフォーマンスで動作可能であることを述べた。

今月号では、実際にウィンドウを開くサンプルプログラムを作成してみることにより、これらの概念の理解を深めてみる。

ウィンドウを開き、直線を描画するプログラム

さて、まずは今月号のサンプルプログラムを実行してみよう。root権限で今月号の付録CD-ROMをマウントしてから、一般ユーザーでログインし、本連載のプログラムがあるディレクトリに移動する。そこには、x11test.cとMakefileがある。

```
$ make 1
```

とすると、x11test.cというファイル名のプログラムが自動的にコンパイルされ、**画面1**のようなウィンドウが現れるはずだ。

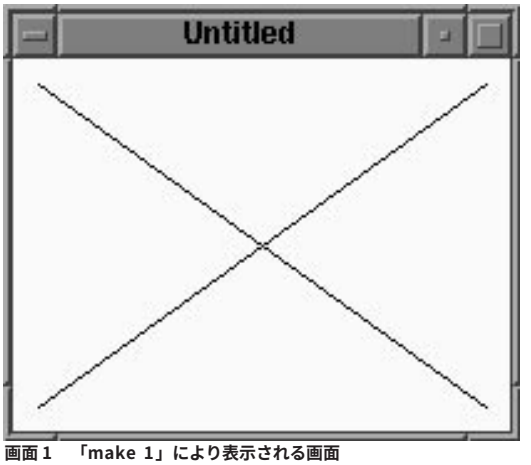
それでは、このプログラムのソースコードを順に追ってみることにしよう。x11test.c はコマンドライン引数により、今回解説する3つのサンプルを実行する。コマンドラインに「1」を指定すると、

```
base_window_open(50, 50, 200, 150 );
line_draw();
XFlush(base_disp);
for(;;){}
```

というプログラムが実行されることになる。

ここでは、筆者が定義したbase_window_openという関数を用いて、ルートウィンドウのx座標が50、y座標が50の位置を左上とした、幅が200ドット、高さが150ドットのウィンドウを開く。

そして、後述のline_draw関数により交差する2本の直線を描画し、XlibのXFlush関数を呼び出してXサーバに描画を依頼したあと無限ループとなる。



画面1　「make 1」により表示される画面

■base_window_open関数

リスト1に示すのがbase_window_open関数だ。この関数では最初に先月号で説明したXlib関数のXOpenDisplayを呼び出し、サーバへの接続を行う。ここで、引数にNULLをすると環境変数DISPLAYで指定されるXサーバに接続することになる。

次に、Xlib関数XCreateSimpleWindowを呼び出し、ベースとなるウィンドウを開く。そして、デフォルトスクリーン画面を調べ、あとで説明するExposeイベント関連の設定を行う。

■line_draw関数

次に実行されるのが、**リスト2**に示すline_draw関数だ。この関数内ではGCと呼ばれるグラフィックコンテキスト（コラム参照）を作成し、2本の直線を描画している。

Xでは、左上を原点とし、X座標は右方向、Y座標は下方向がプラスとなっているため、最初のXlib関数XDrawLineでは、左上から右下へ直線を引き、次のXDrawLineではそれと交差する形で、右上から左下へと直線が引かれる。

2本の直線を引き終わると、使用したGCで確保していたメモリなどの資源を解放する。

▼リスト1　base_window_open関数

```
Display         *base_disp;
int              base_screen;
Window           base_window;
/*--------------------------------------------------------------------------*/
static void
base_window_open(int x0, int y0, int w, int h)
{
        XEvent          e;
        XSizeHints      sh;

        base_disp   = XOpenDisplay(NULL);   /* NULLで環境変数DISPLAYを参照 */
        if( base_disp == NULL ){
                fprintf(stderr, "Can't connect to %s\n", getenv("DISPLAY"));
                exit(0);
        }
        base_window = XCreateSimpleWindow(
                base_disp,                      /* Xサーバの指定            */
                DefaultRootWindow(base_disp),   /* 親ウインドウのID         */
                x0,                             /* ウインドウの左上のX座標 */
                y0,                             /* ウインドウの左上のY座標 */
                w,                              /* ウインドウの幅           */
                h,                              /* ウインドウの高さ         */
                8,                              /* ウインドウボーダの幅     */
                XBlackPixel(base_disp,0),       /* ウインドウボーダの色     */
                XWhitePixel(base_disp,0)        /* 背景の色                 */
        );

        sh.flags = USPosition | USSize;
        XSetWMNormalHints(base_disp, base_window, &sh );
        base_screen = DefaultScreen(base_disp);
        XMapWindow(base_disp, base_window);
        XSelectInput( base_disp, base_window, ExposureMask );

        /* 上で開いたウインドウを描画するため、Exposeイベントを送る */
        e.type = Expose;
        e.xany.window = base_window;
        XSendEvent(base_disp, base_window, True, ExposureMask, &e);
}
```

上で指定した(x0,y0)の位置に無理やり表示
ウインドウマネージャの介入を阻止
screen番号を得る
ウインドウを可視化
Exposeイベントを受け取れるようにする
再描画イベント
イベントの構造体

図3　GPGの全関数のリンクとGPGのmain関数の呼び出し

■ Xlib関数XFlush

Xlib関数XDrawLineでは、直線描画開始命令やその座標などがXプロトコルに変換され、メモリ中にバッファリングされる。ただし、即座にXサーバへ送信されるわけではない。

これらのバッファリングされたXプロトコルをサーバに送るのがXlib関数XFlushであり、この関数を呼ぶことにより実際に2本の直線が描画されることになる。

ここで、もしXDrawLineを呼ぶたびにXサーバとの通信を行うと、通信パケットが何回もネットワークを流れることになる。しかし、Xではバッファリングしてまとめて Xサーバに送ることにより、通信効率を上げている。

再描画の必要性と再描画の実現方法

上で説明したプログラムでは2本の直線を引いたが、このプログラムには問題がある。それは、この2本の直線の上に別のウィンドウが通過した場合、直線が消されてしまうということだ（**図1**）。

多くのウィンドウシステムでは、自分のウィンドウが他のウィンドウによって消されてしまった場合は、自分自身の責任でそれを再描画することになっている。ウィンドウシステム内で各ウィンドウを保存しておくと膨大なメモリが必要となるため、自分のウィンドウの内容を一番よく知っている自分自身が、責任をもって再描画するというのはリーズナブルな考え方だ。これは、Xだけでなく Microsoft Windowsなどでも同様な実装となっている。

この再描画を行うプログラムは、先ほどのディレクトリで、

```
$ make 2
```

と入力することにより実行できる。今度は、先ほどのプログラムと違って、他のウィンドウによって2本の直線が消されても、再びウィンドウが表に現れるときには、2本の線がきちんと引き直されるはずだ。

このプログラムのソースコードは次のとおりだ。

```
XEvent      e;
```

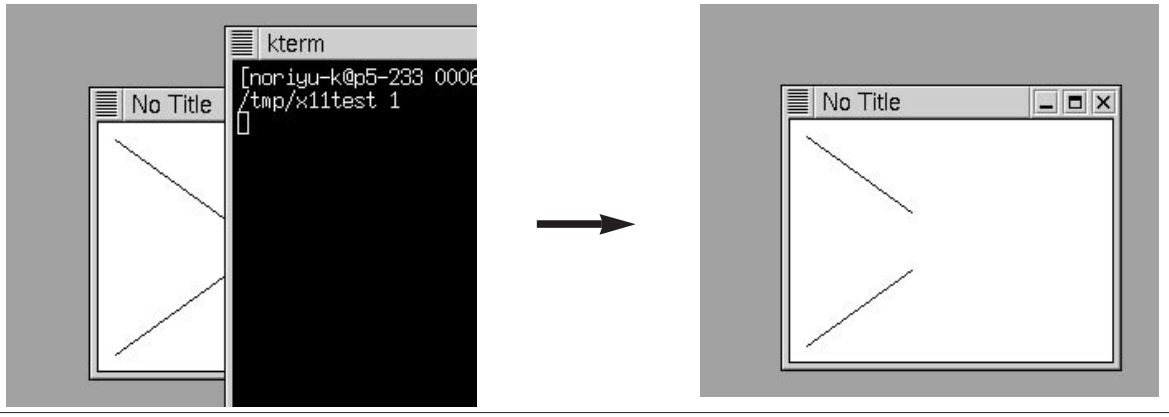


図1　左の画面のように、他のウィンドウ（この場合はkterm）によって直線を引いたウィンドウが隠されてしまった場合、Kterm のウィンドウがなくなっても、直線は消されたままになってしまう。「$ make 2」によって表示されたウィンドウの場合、直線は再描画される

▼リスト2　line_draw関数

```
void
line_draw(void)
{
        GC      gc;

        gc = XCreateGC( base_disp, base_window, 0, NULL);
        XDrawLine(base_disp, base_window, gc,  10, 10,  190, 140 );

        XDrawLine(base_disp, base_window, gc, 190, 10,   10, 140 );

        XFreeGC( base_disp, gc );
}
```

```
base_window_open(50, 50, 200, 150 );
for(;;){
    XNextEvent(base_disp, &e);
    if( e.xany.window == base_window ){
            if( e.type == Expose ){
                    line_draw();
            }
    }
    XFlush(base_disp);
}
```

Xでは、ウィンドウの内容が消された場合、Exposeというイベントが発生することになっている。上記のXEvent型はこのイベントを扱うための型で、Xlib関数のXNextEventによりイベントが発生するまで待ち、イベントが発生したときに、そのイベントをこの型の変数へ格納してリターンする。

上記のe.xany.windowはイベントが発生したウィンドウ識別子を示していて、e.typeはその発生したイベントのタイプを示している。この場合、2本の線が引かれているウィンドウ（base_window）にExposeイベントが発生した場合、line_draw関数が呼ばれることになる。

すなわち、2本の線を表示するウィンドウが他のウィンドウによって隠されたのち、隠された部分が再び露出した瞬間にExposeイベントが発生し、line_draw関数で直線が再描画されるのだ。

ボタンを表示するプログラム

さて、線を引く方法と再描画の概念がわかったところで、次はボタンを作ってみることにする。先ほどのディレクトリで、

▼リスト3　button_open関数

```
#define BUTTON_MAX      3
void func_0(void) { printf("push FreeBSD\n"); }
void func_1(void) { printf("push NetBSD\n");  }
void func_2(void) { printf("push Linux\n");   }
/*-------------------------------------------------------------------------*/
void
button_sample(void)
{
        XEvent          e;
        button_t        *button[BUTTON_MAX];
        int             i;

        base_window_open(50,50, 140, 195);

        button[0] = button_open(20, 20, 100, 35, "FreeBSD", func_0 );
        button[1] = button_open(20, 75, 100, 35, "NetBSD",  func_1 );
        button[2] = button_open(20,135, 100, 35, "Linux",   func_2 );

        for(;;){
                XNextEvent(base_disp, &e);
                for(i=0; i<BUTTON_MAX; i++){
                        if( e.xany.window == button[i]->window ){

                                if( e.type == Expose ){
                                        button_redraw(button[i]);
                                }else
                                if( e.type == ButtonPress ){
                                        (button[i]->callback)();
                                }
                        }
                }
                XFlush(base_disp);
        }
}
```

ボタンの左上の(x,y)座標

ボタンの幅と高さ

ボタンが押されたときに呼び出されるコールバック関数

すべてのボタンについて処理

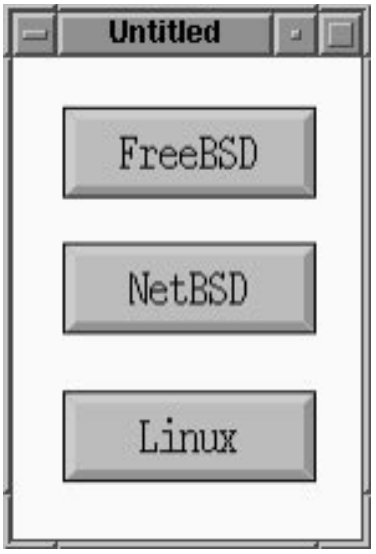
i番目のボタンのウインドウ識別子

再描画イベント

i番目のボタンを再描画

ボタンが押されたというイベント

i番目ボタンのコールバック関数



画面2　「make 3」により表示される画面

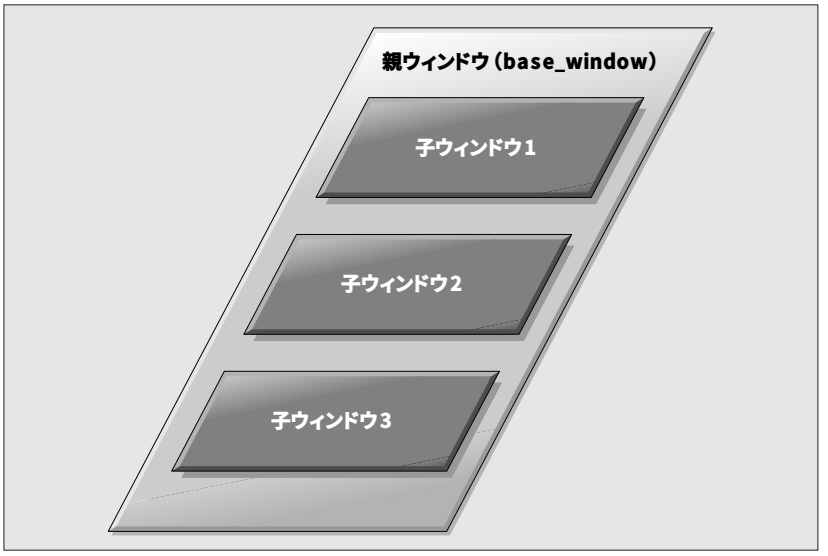


図2　ベースウィンドウと3つの子ウィンドウ

```
$ make 3
```

を実行することにより、**画面2**のようなウィンドウが表示される。

ここで実行されるプログラムは、**リスト3**に示すものだ。まず、先ほどのように、base_window_openを使ってベースウィンドウを開き、その上に3つのボタンを作成する。

このプログラムでは、button_openという関数を作成してボタンを表示している。button_open関数の引数は、ボタンの左上の(x,y)座標、ボタンの幅と高さ、ボタンに表示する文字列、そして、ボタンが押されたときに呼び出される関数へのポインタ（ステップアップC言語参照）だ。

つまり、

```
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3行で縦に3つのボタンが作られ、各ボタンの表面には、「FreeBSD」「NetBSD」「Linux」という文字列が表示される。そして、それぞれのボタンが押されたときには、func0, func1, func2 が呼び出されることになる。こ

Column

グラフィックコンテキスト（GC）

Xは、Xプロトコルを通してさまざまな描画を行うため、通信を効率よく行う必要がある。そのために、「グラフィックコンテキスト（GC）」という概念が導入されている。

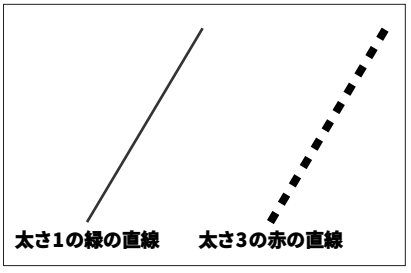


図　描画のサンプル

ここで、GCを理解するために、**左下図**のように線の太さや、色、そして破線などの種類を指定する場合を考えてみる。

ここで同じ種類の2本の直線を引く場合、Xプロトコルとして、

直線を引く	太さ	色	種類	始点終点その1
直線を引く	太さ	色	種類	始点終点その2

というパケットを流すことも考えられるが、何本もの直線を引く場合、

太さ	色	種類

の部分は、共通であることが多く、直線を引くたびにパケットを送ることはトラフィックの無駄だ。このため、Xでは「太さ」「色」「種類」の情報を、GCとしてあらかじめサーバに送っておき、実際に線を描画する場合は、GCの番号として指定するようになっている。

GCの登録	太さ	色	種類
直線を引く	GCの番号	始点終点その1	
直線を引く	GCの番号	始点終点その2	

これにより、ネットワークトラフィックを削減することができる。GCには、上記のような直線の種類のほかに、文字のフォントなどさまざまなものがある。GCの概念はXプロトコルのパフォーマンスを上げるうえで、非常に重要なものとなっているのだ。

れらの関数は、

```
void func_0(void) { printf("push FreeBSD\n"); }
void func_1(void) { printf("push NetBSD\n");  }
void func_2(void) { printf("push Linux\n");   }
```

として定義されているので、たとえばNetBSDのボタンが押されると、このプログラムを起動した端末に、

```
push NetBSD
```

という文字列が表示される。

ボタンの実現方法

ボタンを表示する方法として、2本の直線を引いたときのように、ベースウィンドウに直接ボタンを描画する方法がある。しかしながら、この方法の場合、ボタンが押されたことを検出するためには、ボタンの四隅の座標を覚えておく必要がある。そして、マウスが押されたというイベントが起きるたびに、マウスの座標がその4組の座標内にあるかどうかを判断しなくてはならない。この方法はかなり面倒で、バグを生みやすいプログラムになってしまう。

そこで、今回作成したプログラムでは、button_openを呼び出すと、そのボタンを1つのウィンドウとして作成するようにしてある。例では3つのボタンを表示しているので、ベースウィンドウ（親ウィンドウ）の上に3つの子ウィンドウが表示されていることになる（**図2**）。

このように、各ボタンをウィンドウにすることで、マウスが押された場合そのイベントがどのウィンドウで発生したかを調べるだけで、ボタンが押されたかどうかの判断が可能になる。

ボタンを実現するために必要な構造体

ボタンを実現するためには、いくつかの情報が必要になるが、今回作成したプログラムでは、この情報を保持するために、次のような構造体を定義している。

```
typedef struct {
        Pixmap  pixmap;
        Window  window;
        int     width;
        int     height;
        void    (*callback)(void);
} button_t;
```

今回のプログラムでは、このbutton_t型を使い、

```
button_t *button[3];
```

として、3個分のボタン情報を格納する、buttonという名前のbutton_tへのポインタを保持する配列を定義してある。

前述のように、今回のプログラムでのボタンは、Xの1

Column

Xlibに関する参考文献

LinuxやFreeBSDに関する書籍は多数出版されているが、プログラミングについての書籍は多くない。特にX Window Systemに関しては、最近になってgtk+とQ+に関する書籍が数冊出版されているにすぎず、Xlibに関しての参考図書を探すのには苦労する。

しかし、X Window SystemがUNIXワークステーションで広く使われ始めた10年ほど前には、かなりの数の書籍が発行されている。Xlibの仕様は、当時からほとんど変わらないため過去の書籍でも十分役に立つが、すでに絶版になっているものが多いのは残念だ。

ちなみに、筆者がXlibのプログラミングに関して参考にした書籍のうち、現在でも役に立つと思われるものは下記のとおりだ。現在では入手が難しいものもあるが、大学やソフトウェア会社の図書室には蔵書されていることが多いと思われるし、古本として入手できる機会もあると思う。

また、これらの書籍については、書籍検索（http://www.kinokuniya.co.jp/など）でISBN番号を入力すると目次などを調べることができる。

- Xウィンドウ実践技術講座　松田晃一著　1992年　ソフト・リサーチ・センター　ISBN4-915778-17-7
- X Windowハンドブック　西村亮監修　1990年　アスキー　ISBN4-7561-0032-5
- X-Window Ver.11プログラミング［第2版］　木下凌一・林秀幸著　1993年　日刊工業新聞社　ISBN4-526-033995
- Xlibリファレンスマニュアル　1993年　ソフトバンク　ISBN4-89052-441-X
- Xプロトコル・リファレンス・マニュアル　1991年　ソフトバンク　ISBN4-89052-249-2

つのウィンドウとして実現するので、そのウィンドウを特定するための識別子を保存する必要がある。たとえば、ボタンの識別子は、

```
button[0]->window 1番目のボタンのウィンドウ識別子
button[1]->window 2番目のボタンのウィンドウ識別子
button[2]->window 3番目のボタンのウィンドウ識別子
```

に保存されることになる。

また、ウィンドウシステムでは、ボタンを押されたときに何らかの関数を呼び出す必要があることが多く、この機構を簡単に実現するため、ボタンが押されたときに呼び出すべき関数もこの構造体に保存しておくことにする。関数を呼び出すためにC言語では関数へのポインタを用いるが、これは、

```
void (*callback)(void);
```

として定義されている（特別講座ステップアップC言語参照）。

ところで、最初に解説したように、Xではウィンドウが他のウィンドウに隠されて内容が消され、再び表に現れる場合には再描画を行う必要がある。この再描画時に必要なのが、ボタンの幅と高さのピクセル数を保持するための、width、heightという2つのint型のメンバである。そしてボタンのグラフィックデータを保存するために、

```
Pixmap pixmap;
```

というPixmap型のメンバも定義してある。

ボタンは、立体感を出すために枠を描いたり、文字列を描いたりする必要があるが、これを再描画のたびに描いていては、面倒なだけでなく無駄なネットワークトラフィックが生じてしまう。これを避けるため、ボタンのグラフィックデータをPixmapという「入れ物」に保存しておくのである。このPixmapの概念は、来月詳しく解説してみたい。

イベントの検出

リスト3では、

```
for(i=0; i<BUTTON_MAX; i++){
```

というforループの中で、3つのボタンのウィンドウが、発生したイベントのウィンドウであるかどうかを判断している。

```
if( e.xany.window == button[i]->window ){
```

というのが、その判断部分であり、e.xany.windowの値が発生したイベントのウィンドウ識別子である。この値が、

```
button[i]->window
```

の内容と等しければ、そのi番目のボタンが押されたと判断されるというわけである。

今回の場合再描画イベントであるExposeと、マウスボタンが押されたイベントであるButtonPressの2つのイベントが発生するが、どちらのイベントが発生したかはe.typeによって判断できる。e.typeがExposeの場合は、

```
button_redraw(button[i]);
```

という関数により、i番目のボタンを再描画し、e.typeがButtonPressの場合は、

```
(button[i]->callback)();
```

という部分で、buttonという構造体の、callbackというメンバに格納された関数へのポインタが示すコールバック関数が呼ばれることになる。

来月号の予告

ここまでで誌面がつきてしまったので、来月はPixmapの概念や、ボタンを実現するbutton_open関数などについて、引き続き解説してみたい。

ツールキットを使うプログラムと比べ、Xlibを直接使うプログラミングは多少難解ではあるが、Xの本質を理解するうえでXlibの知識は欠かせない。

Xlibすなわち X プロトコルは、X ウィンドウシステムの基本中の基本であり、この基本概念を理解できればツールキットを使ってプログラミングをする場合にもかならず役に立つはずである。

特別講座

ステップアップC言語

関数へのポインタ

■関数へのポインタの使い方

関数へのポインタという概念は、なかなか理解しにくいかもしれない。しかし、これはC言語では重要な概念で、特にウィンドウシステムでは多用される。したがって、Xのプログラミングには欠かすことができない。

さて、この概念を理解するために、コマンドラインの引数として、関数名と角度（ラジアン）を、

```
$ ./ftest cos 3.14159265
```

というように与えると、その計算結果を表示するプログラムftest.cを考えてみることにする。

関数へのポインタを使うと、このプログラムは**リストc-2**のような形に書き直すことができる。

ここで、

```
double (*func)(double);
```

というのが関数へのポインタの宣言部分であり、funcという変数にはsin関数などの関数の開始アドレスが保持されることになる。

なお、このプログラムをコンパイルするには、libm.aライブラリをリンクする必要がある。先月号で解説したように、

```
$ cc ftest.c -o ftest -lm
```

としなければならない。

■割り込み関数の登録

上記の例のように、呼び出す関数の名前があらかじめ想定できるような場合は、関数へのポインタを使わなくてもプログラムを書くことができる。しかし、周期的に呼び出される関数を登録するライブラリを作る場合などには、関数へのポインタは必須だ。

たとえば、1秒おきにhandlerという関数を呼び出すプログラムは、**リストc-3**のように書くことができる。

ここでは、signal関数でSIGALRMシグナルが起きたときに、handler関数を呼び出すように登録して、alram関数で、1秒後にSIGALRMシグナルを送るように記述している。

このような割り込み関数の登録をする場合、関数のポインタがなければ実現が非常に難しいことがわかる。

■コールバック関数

本文で説明したように、ウィンドウシステムではウィンドウの内容が消されると、再描画が必要になったり、ボタンが押されたときにそのボタンに関連付けられた関数を呼び出す必要がある。

このような関数は、システム側から呼び出されることから「コールバック関数」と呼び、その登録には上記で説明したような関数のポインタが使われることになる。

▼リストc-1　ftest.c

```
#include <stdio.h>
#include <math.h>

int main(int argc, char **argv)
{
        char    *name;
        double   value, ans;

        if( argc != 3 ){ return -1; }
        name = argv[1];
        sscanf(argv[2], "%lf", &value);

        if( strcmp(name, "sin") == 0){
                ans = sin(value);
        }else
        if( strcmp(name, "cos") == 0){
                ans = cos(value);
        }else
        if( strcmp(name, "tan") == 0){
                ans = tan(value);
        }else{
                printf("error\n");
                return -1;
        }
        printf("ans=%f\n", ans);
        return 0;
}
```

▼リストc-2　関数へのポインタを使うように変更したftest.c

```
#include <stdio.h>
#include <math.h>

int main(int argc, char **argv)
{
        char    *name;
        double  value;
        double (*func)(double);

        if( argc != 3 ){ return -1; }
        name = argv[1];
        sscanf(argv[2], "%lf", &value);

        if( strcmp(name, "sin") == 0){
                func = sin;
        }else
        if( strcmp(name, "cos") == 0){
                func = cos;
        }else
        if( strcmp(name, "tan") == 0){
                func = tan;
        }else{
                printf("error\n");
                return -1;
        }
        printf("ans=%f\n", func(value));
        return 0;
}
```

▼リストc-3　1秒おきにhandler関数を呼び出すサンプル

```
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

void handler(int n){
        printf("Hello!!\n");
        alarm(1);
}

int main(int argc, char **argv)
{
        signal(SIGALRM, handler);
        alarm(1);
        for(;;){
                /* */
        }
}
```

PostgreSQLを極める

最近はどこにでもWindowsマシンがある。本誌の読者でも、ふだんはWindowsを使っている（使わざるをえない？）人が多いのではないだろうか。筆者も例外ではなく、日常業務の多くの時間をWindowsで過ごしている。そこで今回は趣向を変えて、PostgreSQLとWindowsの連携について紹介しようと思う。

第8回　Windowsとの連携（1）

文：片岡裕生
Text : Hiroki Kataoka

PostgreSQLは、UNIX系OSを主なターゲットに開発されてきました。そのため、多くのUNIX系OS上で利用できます。しかし、LinuxやFreeBSDなどのPC-UNIXが浸透してきたとはいえ、まだまだWindowsほど普及しているとはいえない状況です。職場のデスクには、Windowsをインストールした PCがある、という読者も多いことでしょう。このような状況では、WindowsでもPostgreSQLを使いたいと思うのは筆者だけではないでしょう。

そんなわけで今回は、WindowsでPostgreSQLを使う方法についていろいろと紹介します。

Windows NT版PostgreSQL

最初に紹介するのは、そのものズバリ、Windows NT上で動作するPostgreSQLです。実は、最近のPostgreSQLはWindows NT上でも動作するのです。ただし、Cygnus Solutions社が無料で提供している、Cygwin（B20.1）というWindows用UNIX互換ライブラリを利用していますので、事前にそれをインストール・設定しておく必要があります（そのほかいくつかのライブラリなども必要）。

PostgreSQLをWindows NT＋Cygwinに移植しているのは、谷田 豊盛（たにだ ゆたか）さんで、Webページではコンパイル済みバイナリやFAQが公開されています。

Something PostgreSQL
http://www.s34.co.jp/~luster/pgsql/

PostgreSQLとWindows NTの間に、UNIX互換ライブラリを挟んでいるということもあり、同じ条件のUNIX系OS上で動作しているPostgreSQLと比べると、100％の性能は出ないようです。しかし、「開発の便宜、テスト、個人の楽しみ」（readmeから引用）などの目的においては、かなり気になる存在ではないでしょうか。

筆者はふだんWindows NTを利用していないため、このWindows NT版PostgreSQLは利用したことがありません。ですから、残念ながら、これ以上詳しい紹介ができないのですが、WindowsとPostgreSQLの連携という今回の題材においては、やはり最初に紹介すべきものでしょう。

クライアントとしてのWindows

ここでは、PostgreSQL自体はあくまでもLinuxなどのサーバ上で動かすとして、WindowsをPostgreSQLのクライアント環境としてのみ利用するパターンを考えてみます（**図1**）。

PostgreSQLはクライアント／サーバによる運用をサポートしていますので、クライアント部分だけをWindows側に持ってくることは可能です。実際に利用可能な形態を、以下にあげます。

1. PostgreSQL用クライアントライブラリを利用
2. JDBCを利用
3. ODBCを利用

それでは、それぞれの利用形態について、もう少し詳しく紹介していきたいと思います。なかでもODBCについては、筆者が多少係わっているということもありますから、少し詳しく解説します。

PostgreSQL用クライアントライブラリ

PostgreSQLには、“libpq”というクライアントライブラリがあり、PostgreSQLに付属する各種ツールなどは多くがこのライブラリを利用しています。つまり、libpqはPostgreSQLの標準クライアントライブラリといえるものです。

実は、libpqにはWindows版があります。32ビットWindows環境で利用でき、Windows 98やWindows NTなどに対応しています。Windows版libpqライブラリはダイナミックリンクライブラリ（DLL）形式で提供されており、英語版ではありますが、コンパイル済みのクライアントライブラリ“libpq.dll”がPostgreSQLのソースパッケージにも含まれています。なお、英語版クライアントライブラリには、日本語を扱ううえで問題があります。

日本語を正しく扱うためには、マルチバイトに対応したクライアントライブラリを用いなければなりません。PostgreSQLのソースファイルから新たに作成することもできますが、筆者のWebページでもコンパイル済みのマルチバイト対応クライアントライブラリを、日本語Windows版libpqという名称で公開しています。

インターウィズ PostgreSQL関連情報
http://www.interwiz.koganei.tokyo.jp/software/PostgreSQL/index.html

上記のページからダウンロードできる日本語Windows版libpqでは、クライアントの文字エンコーディングをデフォルトでシフトJISにするなど、日本語Windows上で利用するうえで便利なように、多少の手を加えてあります。

このクライアントライブラリを利用する最も一般的な方法は、C言語によるアプリケーション開発です。Windows版libpqライブラリの仕様自体は、UNIX系OS上のlibpqライブラリとまったく同じですから、PostgreSQLに付属のドキュメントなどを参考にするといいでしょう。

また、このWindows版クライアントライブラリを利用した別のツールとして、Windows版libpgtclもあります。これは、Tcl/Tkと呼ばれるスクリプト言語から、PostgreSQLデータベースにアクセスするための拡張モジュールです。Tcl/Tkは、GUIアプリケーションも作成できる高度なスクリプト言語で、Windows版もあります。これらを利用すれば、GUIを利用した高度なデータベースアプリケーションを、非常に手軽に作成することができます。

先ほど紹介した筆者のWebページから、日本語Windows版libpgtclが入手可能です。

JDBCの利用

最近では、Javaによるアプリケーション開発の例をよく見かけるようになりました。Javaアプリケーションはもちろんのこと、JavaアプレットやサーブレットなどのJava

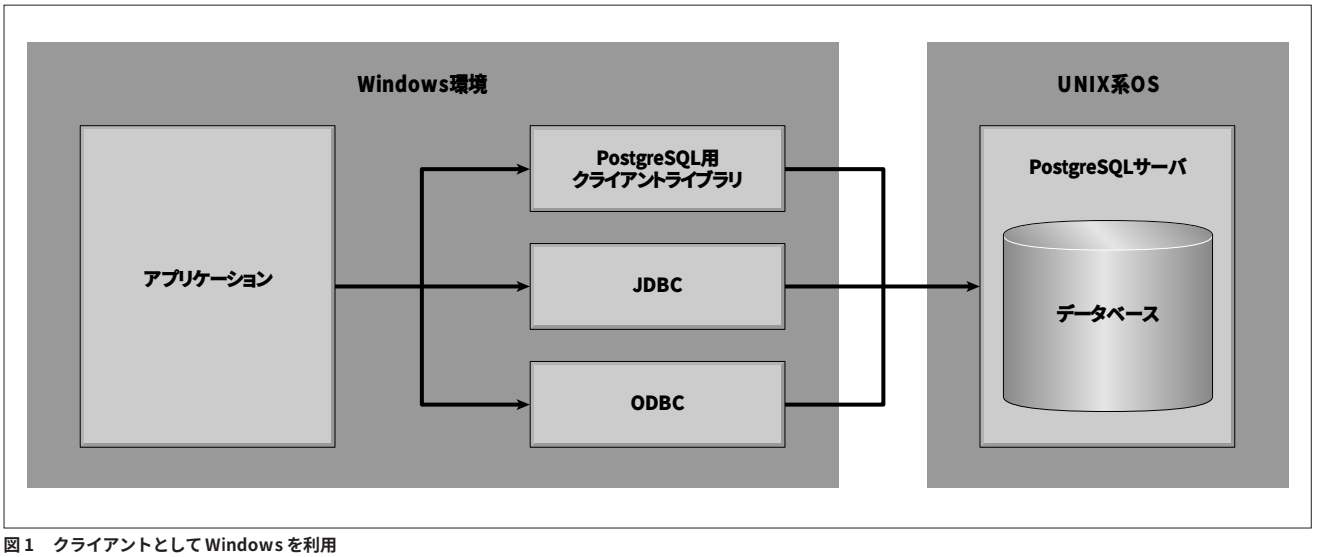


図1　クライアントとしてWindowsを利用

環境からデータベースを利用するしくみとして“JDBC”が有名ですが、PostgreSQLにもJDBCドライバが存在します。現在ではJDBC 1用とJDBC 2用の2種類のJDBCドライバが提供されており、PostgreSQLのソースパッケージにソースファイルの形で含まれています。

PostgreSQLが標準で提供しているJDBCドライバは、特に日本語に対応しているわけではありません。しかし、現在のJava自体は日本語を取り扱うことができるので、PostgreSQLデータベース側とJavaによるクライアント側との間で特に文字エンコーディング変換が必要でない限り、日本語環境でも正しく動作させることは可能です。

ところが、クライアント側がWindows環境の場合はどうでしょうか。Windowsの文字エンコーディングはシフトJISですから、PostgreSQLデータベースがシフトJISをサポートしていない現状では、どうしても文字エンコーディング変換が必要になります。つまり、Windowsクライアントから PostgreSQLデータベースを利用するということが、標準のJDBCドライバではできないのです。

さいわいなことに、持田 修司さんが文字エンコーディング変換を可能にするPostgreSQL JDBCドライバ用のパッチを作成し、下記のWebページで公開されています。

PostgreSQL JDBC Driver Character EncodingPatch
http://www.netside.co.jp/~mochid/comp/postgresql-jdbc/index.html

このパッチを適用したドライバでは文字エンコーディング変換が可能で、PostgreSQLデータベース側とクライアント側との文字エンコーディングが異なっていたとしても（たとえば、EUC_JPとシフトJISであっても）、それぞれの環境で正しく日本語を扱うことができます。

具体的な利用方法などは省略しますが、機会があればぜひ紹介したいと思います。

ODBCの利用

Windows環境からデータベースを利用する方法としては“ODBC”が有名で、多くのWindowsアプリケーションがこれをサポートしています。もちろん、PostgreSQL用のODBCドライバがありますので、これら多くのWindowsアプリケーションからPostgreSQLデータベースを利用することが可能です。

それでは少し詳しく紹介していきます。

■ODBCの構成

ODBC（Open Database Connectivity）とは、PostgreSQLのようなデータベース管理システム（以下DBMS）のための共通インターフェイスです。ODBCによって、異なる仕様のさまざまなDBMSが、統一されたプログラムインターフェイスで利用できます。ODBCの構成は**図2**のようになっています。

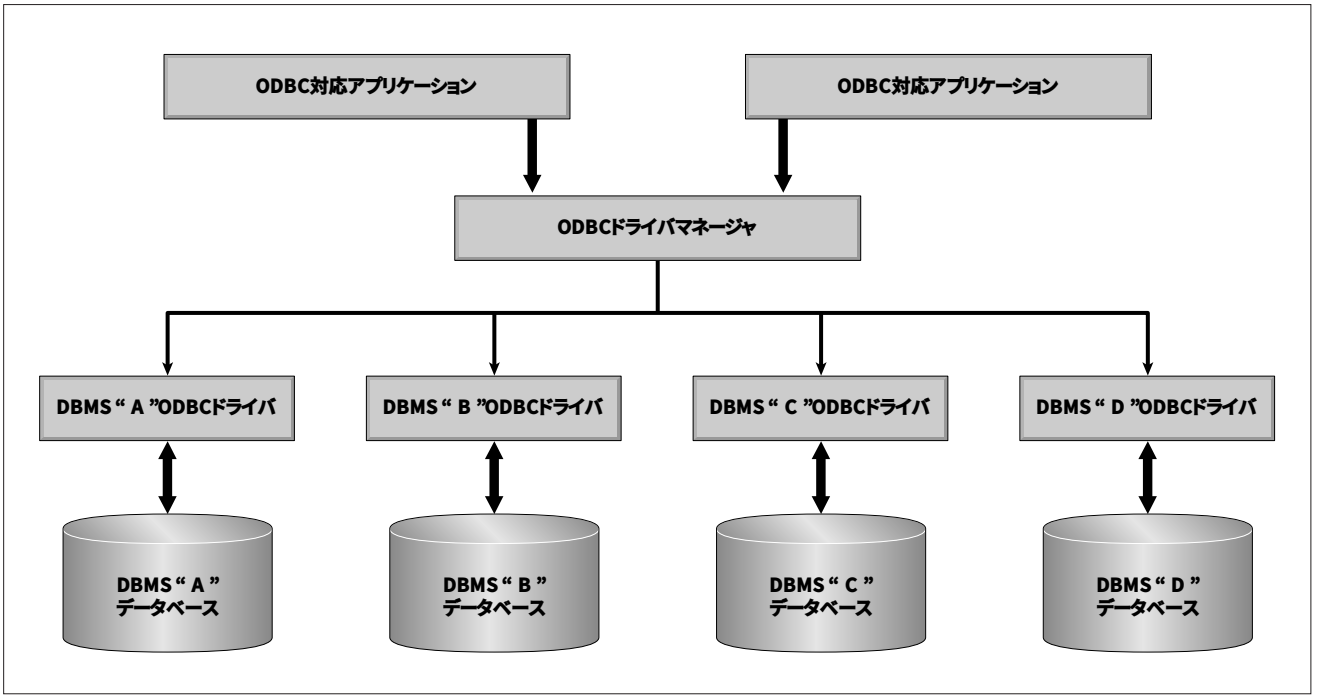


図2　ODBCの構成

各種DBMSごとの違いは、それぞれに用意された“ODBCドライバ”で吸収するようになっており、ODBCドライバマネージャがこれらを管理しています。アプリケーションは、常にODBCドライバマネージャに対してのみアクセスしますので、個々のDBMSごとの違いはあまり意識しないでよいことになるのです。

ODBCを利用してPostgreSQLデータベースにアクセスするためには、PostgreSQL用ODBCドライバがインストールされていればよいことになります。

■PostgreSQL ODBC Driver

PostgreSQL用のODBCドライバでWindowsに対応しているもののうち、最も代表的なものが“PostgreSQL ODBC Driver”です。

PostgreSQL ODBC Driverは、Insight Distribution Systems社によってPostODBC（Postgres95用のODBCドライバ）を元にして作成された、フリーでオープンソースのPostgreSQL用ODBCドライバです。現在では、Byron Nikolaidisさんがメンテナンスを行っています。なお、単に“PsqlODBC”と表記される場合が多いようです。

32ビットWindows環境に対応しており、Windows 98やWindows NTなどで利用可能です。PostgreSQL ODBC Driverの現在の公式Webページは、次のURLになります。

ftp://ftp.postgresql.org/pub/odbc/index.html

また、PostgreSQLのソースパッケージにも、同じPostgreSQL ODBC Driverが含まれていますが、どちらのバージョンでも日本語は扱えません。日本語を扱いたいのであれば、後述する日本語版が利用できます。

対応しているPostgreSQLサーバはバージョン6.2から7.0まで（日本語版は6.5.xまで）で、ホストベース認証とpassword認証に対応しています。残念ながら、crypt認証には対応していません。

PostgreSQL ODBC Driverでサポートしているデータ型を**図3**と**図4**にあげておきます。**図3**は、PostgreSQLのデータ型からODBC定義のデータ型へ変換される場合の対応関係を、**図4**は、ODBC定義のデータ型からPostgreSQLのデータ型へ変換される場合の対応関係を示しています。

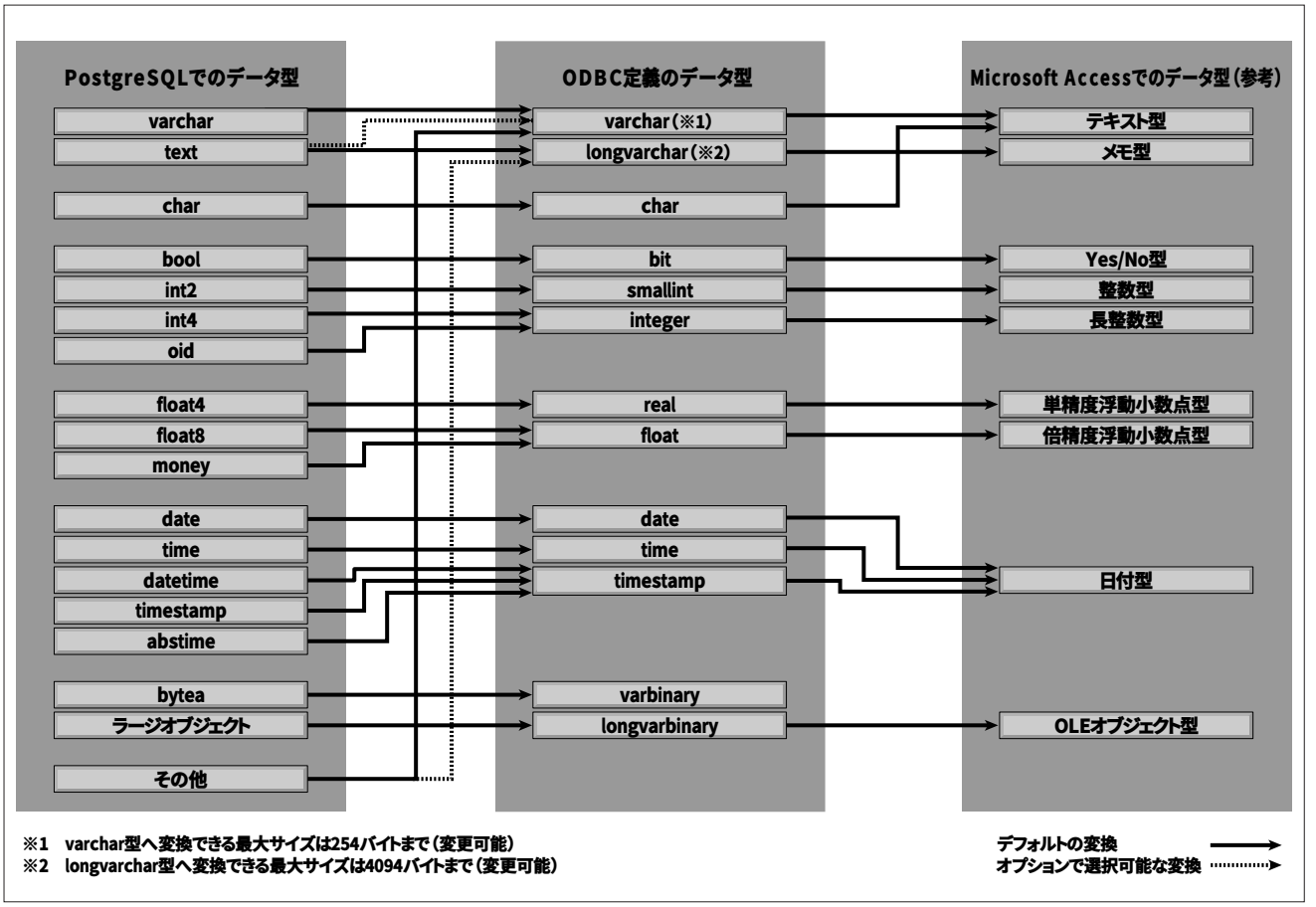


図3 データ型の対応（PostgreSQL→ODBC）

それぞれについて、参考のためにMicrosoft Accessでのデータ型も載せておきました。なお、PostgreSQLのデータ型は基本のものだけを掲載しています。たとえば、PostgreSQLのinteger型は実際にはint4型の別名ですので、図にはinteger型を載せていません。この図にあるとおりにデータ型の変換が行われるのは、あくまでも変換先のデータ型が指定されていない場合だけです。浮動小数と整数の間の変換のような一般的な型変換は、サポートされています。

■PostgreSQL ODBC Driver 日本語版

PostgreSQL ODBC Driver 日本語版は、オリジナルのPostgreSQL ODBC Driverに対し、筆者が文字エンコーディング変換（EUC_JP ⇔ SJIS）機能などを追加したものです。リリース時期によっては、日本語版独自の改良を施している場合もあります。しかし、改良点はできるだけオリジナル側にフィードバックしていますので、必ずしも日本語版のほうが優れているわけではありません。

PostgreSQL ODBC Driver 日本語版は、以下のWebページから入手できます。

インターウィズ PostgreSQL ODBC Driver 日本語版
http://www.interwiz.koganei.tokyo.jp/software/PsqlODBC/index.html

なお、PostgreSQL ODBC Driver 日本語版は、オリジナル版に対する差分という形で提供していますので、日本語版を利用するためには、先にオリジナル版をインストールしておく必要があります。上記ページには、ドライバ本体のほかにFAQも用意してあります。何か問題が起きた場合には、ぜひ参照してください。

PostgreSQL ODBC Driver 日本語版がサポートしている、PostgreSQLデータベース側の文字エンコーディングは、EUC_JPのみです。

PostgreSQL ODBC Driver 日本語版の2000/01/25版からは、ユーザー定義文字（外字）やIBM補助漢字などの、Windows特有の文字のサポートが強化されました。この機能を利用するには、PostgreSQLサーバのほうも対応していなければなりません（sjis.patchを適用済みでなければならない）。もちろん、半角カナも利用できます。なお、EUC_JPの環境では、半角カナが2バイト表現に、ユーザー定義文字やIBM補助漢字の多くは3バイト表現に変換され

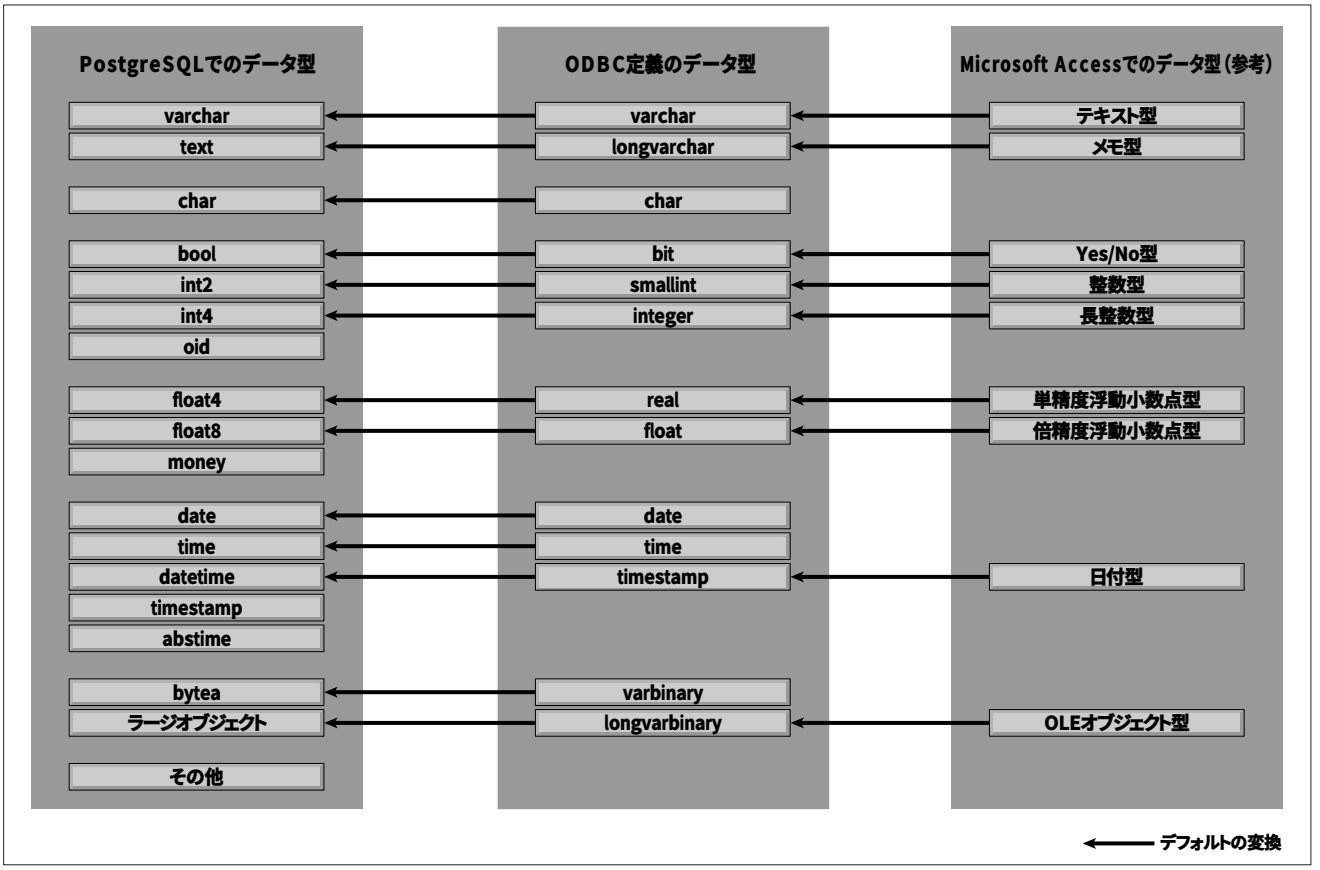


図4　データ型の対応（ODBC→PostgreSQL）

ますので、テーブルのカラムサイズなどには注意が必要です。

インストール

PostgreSQL ODBC Driver 日本語版を使う場合でも、まずはオリジナル版のPostgreSQL ODBC Driverをインストールしなければなりません。

■PostgreSQL ODBC Driverのインストール

先に紹介したオフィシャルサイトから、インストールキット“ postdrv.exe ”を入手します。最終的に日本語版をインストールする予定なら、ここでインストールするオリジナル版のバージョンはあまり関係ありません。気にせず最新版を入手しましょう。そしてこのファイルをWindowsマシン上で実行すれば、インストールが始まります。インストール自体はボタンを押していくだけの簡単なものですが、いくつかポイントがありますので紹介しておきます。

PostgreSQL ODBC Driverのインストールキットには、英語版ODBCドライバマネージャが同包されています。Windowsマシン内にすでに日本語版の古いODBCドライバマネージャがインストールされている場合、これが英語版の新しいODBCドライバマネージャにアップグレードされてしまいます。英語版のODBCドライバマネージャがインストールされると、コントロールパネルのODBCアプレットの画面表示が、すべて英語になってしまいます。覚悟していてください。

もちろん、WindowsマシンにODBCドライバマネージャがインストールされていない（Windowsのコントロールパネルに ODBCアプレットがない）場合には、ありがたく英語版のODBCドライバマネージャをインストールしてもらいましょう。

画面1　ODBCデータソースアドミニストレータ画面で［追加...］ボタンを押す

インストールの途中で、ODBCドライバマネージャのインストールに関する選択肢が表示される場合があります。ここでは、むやみに英語版にされないためにも、以下の優先順位で選ぶのが安全です（それでも英語版にされる場合はあり得る）。

1. Install Driver Manager（with version checking）
2. Install Driver Manager

以上で、オリジナル版PostgreSQL ODBC Driverのインストールは完了です。日本語を扱う予定なら、引き続き日本語版のインストールを行います。

■PostgreSQL ODBC Driver 日本語版のインストール

こちらも、先に紹介したWebページから日本語版への差分ファイルを入手します。このサイトで公開されているファイルは圧縮されていますので、ダウンロード後に展開して“ psqlodbc.dll ”という名前の日本語版差分ファイルを取り出します。

日本語版差分ファイルの準備ができたら、これをインストール済みのオリジナル版に上書きします。具体的には、Windowsのシステムディレクトリ（たとえば C:\WINDOWS\SYSTEMやC:\WINNT\SYSTEM32）にある、同名のファイルに上書きコピーします。これでめでたく日本語版になりました。

なお、すでに一度でもオリジナル版PostgreSQL ODBC Driverをインストール済みであれば、以後は日本語版差分ファイルの上書きだけで最新版に移行できます。つまり、バージョンアップするためだけなら、オリジナル版の再インストールは必要ありません。

以上で、PostgreSQL ODBC Driver 日本語版のインストールは完了です。Windowsのコントロールパネルの ODBC（または、ODBCデータソース）アプレットを起動して、“ ドライバ ”タブを表示してみてください。ドライバ一覧中に“ PostgreSQL ”と表示されていればOKです。

なお、PostgreSQL ODBC Driver 日本語版のインストール方法は、Web ページ（http://www.interwiz.koganei.tokyo.jp/software/PsqlODBC/index.html）にも書いてありますので参照してください。

設定方法

それでは最も一般的なODBCの使い方を説明します。

まずは“データソース”を作成します。データソースとは、DBMSの種類やデータベース名、ユーザー名やパスワードなどの、DBMSへアクセスするために必要な情報の集まりのことです。ODBCは、このデータソースの情報を元に、DBMSにアクセスするのです。このしくみのおかげで、アプリケーションではすでに作成済みのデータソースを指定するだけで、DBMSにアクセスできるようになります（ちなみにデータソースを利用しない方法も存在する）。

データソースを作成するには、Windowsのコントロールパネルの ODBC（またはODBCデータソース）アプレットを起動します。すると、ODBCデータソースアドミニストレータ画面が表示されます（**画面1**）。そして、［ユーザーDSN］、［システムDSN］、［ファイルDSN］のいずれかのタブを表示します。

Windowsのログインユーザーごとに保管されるデータソースを作成したい場合には［ユーザーDSN］タブを、システム全体で共通なデータソースを作成したい場合には［システムDSN］タブを、通常のファイルとしてデータソースを作成したい場合には［ファイルDSN］タブを選択します。ファイルDSNでは、データソースが通常のファイルに保管されますので、たとえばほかのPCにコピーすることも可能です。

新しいデータソースを作成するには［追加...］ボタンを押します。すると、データソースの新規作成画面が表示されますので、一覧の中から使いたいODBCドライバを選択します（**画面2**）。ここでは、PostgreSQLデータベースへアクセスしたいわけですから、“PostgreSQL”を選択します。なお、ODBCデータソースアドミニストレータ画面で、既存のデータソースを選択してから［削除］ボタンを押せばデータソースの削除が、［構成...］ボタンを押せばデータソースの登録内容の変更が可能です。

データソースの新規作成画面でPostgreSQLドライバを選択して、［完了］ボタンを押すと、次にPostgreSQL Driver Setup画面が表示されます（**画面3**）。ここでは、PostgreSQLデータベースにアクセスするのに必要な情報を指定します。各項目の意味は次のとおりです。

- **Data Source**
  データソースの名称です。データソース登録後はここに指定した名称が表示されます。必ず指定します。

- **Description**
  データソースの説明です。どういうデータソースなのかを覚え書きのように記入できます。省略できます。

- **Database**
  PostgreSQLサーバ内のデータベース名を指定します。省略できます。

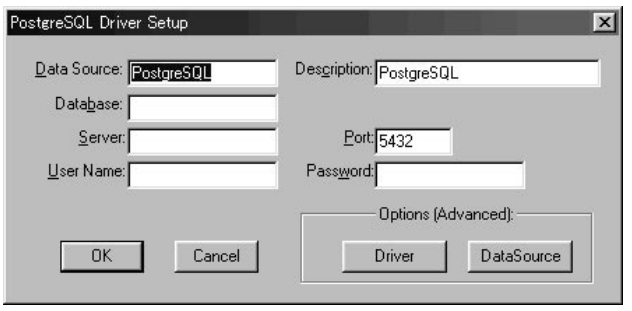
- **Server**
  PostgreSQLサーバのホスト名を指定します。省略できます。

- **Port**
  PostgreSQLサーバのポート番号を指定します。通常は5432です。省略できます。

- **User Name**
  PostgreSQLデータベースにアクセスする際のユーザー名を指定します。もちろん、ここに指定するユーザー名は、PostgreSQLサーバに登録されていなければなりません。省略できます。

- **Password**
  PostgreSQLデータベースにアクセスする際のパスワードを指定します。省略できます。

画面2　データソースの新規作成で“PostgreSQL”を選択する



画面3　PostgreSQL Driver Setup画面で、データベースアクセスに必要な情報を入力する

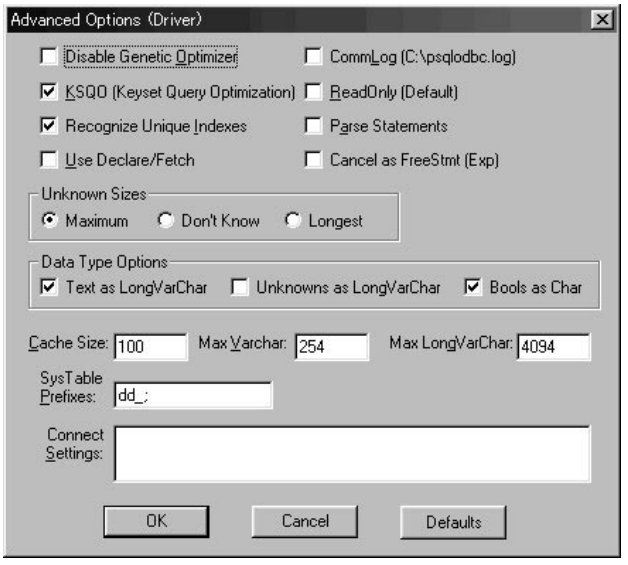
・**Options（Advanced）**

PostgreSQL ODBC Driver全般の動作をカスタマイズするための［Driver］ボタンと、このデータソースの動作をカスタマイズするための［DataSource］ボタンがあります。

Description以外の省略可能な項目を空欄にしておくと、それらをデータソースを使う時点で指定できるようになります（アプリケーションによっては指定できない場合もある）。これを利用して、画面例のようにほとんど空欄のデータソースをひとつ作成しておけば、PostgreSQL用の汎用データソースとして非常に便利です。必要な項目を指定したら、［OK］ボタンを押します。**画面4**のように、今作成したデータソースが一覧内に表示されます。

なお、初めてPostgreSQL用のデータソースを作成する際には、PostgreSQL Driver Setup画面の［Driver］ボタンと［DataSource］ボタンを押して、オプションの設定を行ってください。というのも、デフォルトの状態ではいろいろと問題が発生しやすいのです。ちなみに、ODBCデータソースアドミニストレータ画面からオプションの設定を行うには、PostgreSQLへアクセスする登録済みデータソースを選択して［構成］ボタンをクリックします。

画面4　新規作成したデータソースが表示される



画面5　Advanced Options（Driver）の画面

PostgreSQL Driver Setup画面の［Driver］ボタンを押すと表示されるのが、Advanced Options（Driver）画面です（**画面5**）。

今回は詳しい解説を省略しますが、とりあえず画面例と同じ状態に設定しておくことをお勧めします。

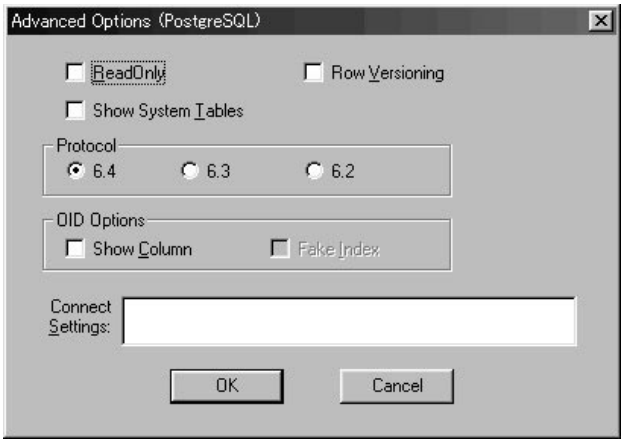
PostgreSQL Driver Setup画面の［DataSource］ボタンを押すと表示されるのが、Advanced Options（PostgreSQL）画面です（**画面6**）。

こちらのオプションも、とりあえず画面例と同じ状態に設定しておくことをお勧めしますが、Protocolの選択だけは、PostgreSQLサーバのバージョンに合わせて選択してください。なお、PostgreSQL6.5.xでは、“6.4”でOKです。

ここでひとつだけ注意があります。ODBC対応アプリケーションとして、Microsoft Access 2000を利用する場合には、Protocolに“6.4”を選択する必要があります。それに伴って、PostgreSQLサーバのバージョンも6.4以上でなければなりません。このように設定しないと、Access 2000からは、PostgreSQLのテーブルにアクセスできない場合があることがわかっています。

●

PostgreSQL ODBC Driverのインストールと設定方法を紹介し終わったところで、誌面が尽きてしまいました。残念ながら、実際にWindows環境からPostgreSQLデータベースをアクセスする具体例までは紹介できませんでしたので、次回に行いたいと思います。



画面6　Advanced Options（Postgre SQL）の画面

Rubyで行こう

先月号で解説したとおり、Rubyのブロック機能はCLUのイテレータが起源です。しかし、拡張によって、イテレータ本来の「繰り返し」としてよりも、「コールバック」としての機能が色濃く出ています。今回は、この「コールバック」について説明します。

第6回　コールバック

文：赤松智也
Text : Tomoya Akamatsu

先月も少し説明したとおり、Rubyの「ブロック」は、CLUという言語の「イテレータ（繰り返し子）」という機能が起源になっています。しかし、いろいろな改良が加えられて、現在ではRuby独自のものになっています。

Rubyでは、ブロックは繰り返しというよりも、「コールバック」として使われます。今月は、このブロックについて詳細に解説します。

イテレータ

イテレータは、以前はRubyでもそう呼ばれていました。しかし、数々の改良のおかげで、もはや「繰り返し」という表現は適切でなくなったので、最近はあまりイテレータとは呼ばなくなりました。でも、バイブルである「Ruby本」ではまだイテレータと呼んでいます。

先月の復習になりますが、もともとのCLUのイテレータは以下のような形式でした。

```
for i:int in int$from_to(1, string$size(s)) do
  ...
end
```

これは「イテレータfrom_toが次々と与える値を変数iに代入して、doからendまでを繰り返し実行する」という意味になります。

CLUのイテレータは「ループの初期化」、「次の値の取得」、「ループ終了条件の検査」を1つの手続きに閉じ込めて、ユーザーに詳細を見せないようにする効果があります。これを「ループの抽象化」と呼びます。

それぞれのデータ構造に適したループというのは、整数を用いたごく単純なものを除けば、

- **内部構造を知る必要がある**
- **インデックスの管理などが繁雑になりやすい**
- **ループ終了条件を間違えやすい**

という問題が生じやすいのです。しかし、イテレータがあれば、そのような問題はループ抽象化でまとめることにより回避できます。

イテレータはCLUの文法ですが、Cのような言語でもイテレータ相当を実現するテクニックがあります。それが関

▼リスト1　Cによるイレテータfrom_to

```
void
from_to(int from, int to, void (*func)()) {
  int i;

  for (i=from; i<to; i++) {
    (*func)(i);
  }
}
```

数ポインタを使った「コールバック」です。たとえば、イテレータfrom_toをCで実装すると**リスト1**のようになるでしょう。

使うときにはループの中身に当たる関数（ここではf）を用意して、

```
from_to(1,10,f)
```

のようにします。

しかし、この方法には問題があります。ループの抽象化はたしかに実現できているのですが、以下の点が解決されていません。

- **ループの中身を別の関数にするのは面倒なだけでなく、処理があちこちに分散して分かりにくくなる**
- **関数from_toの呼び出し側と関数fの間で、情報を簡単に共有する方法がない**

逆に、コールバックにはイテレータにない特徴があります。from_toの例では使っていませんが、処理関数の戻り値を使うことができる点です。実際、Cではコールバックはイテレータのようなループの抽象化よりももっと別の領域、すなわちイベント処理（GUIのボタンが押されたときに開始される処理の登録）などに多用されます。

Rubyでは、上記の2つの問題をCLUのように文法で対応することで解決しています。また、CLUにない「ブロックの値」を得る方法を提供することにより、単なるループの抽象化を越えた領域で使えるものになっています。

Rubyのブロックは以下のように使います。

```
10.times do |i|
  print i, "\n"
end
```

「10.times」というのがメソッドの呼び出しです。この呼び出しにdoからendまでのブロックが付いています。timesメソッドは、0からその数の直前まで繰り返します。ですから、「10.times」は0から9まで繰り返すわけです。「‖」の間に挟まれた変数には、メソッドからブロックに渡された値が代入されます。

ブロックは、ブレース「{}」を使っても表現できます。したがって、上のプログラムは以下のように書き換えられます。

```
10.times {|i|
  print i, "\n"
}
```

「do … end」と「{}」は同じ働きをするのですが、いくつかの点で異なっています。1つは、「do … end」は他の制御構造と同様に「end」で終わるので、なじみがあります。たとえば、**リスト2**よりも**リスト3**のほうが、違和感が少ないとは思いませんか？

もう1つの違いは結合強度です。「結合強度」というと難しいのですが、要するに、前後の要素とどれくらい強く結びついていると解釈されるかということです。たとえば、

```
0.upto 10 do |i|
  print i, "\n"
end
```

は、「0.upto 10」が引数のかっこが省略されているメソッド呼び出しと解釈されて、それにブロックが付いていることになります（uptoはその数から引数の数まで繰り返すメソッド）。ところが、

▼リスト2　「{}」によるブロックの表現

```
class Foo
  def foo
    10.times {|i|
      if i % 2 == 0
        print i, "\n"
      end
    }
  end
end
```

▼リスト3　「do … end」によるブロックの表現

```
class Foo
  def foo
    10.times do |i|
      if i % 2 == 0
        print i, "\n"
      end
    end
  end
end
```

```
0.upto 10 {|i|
  print i, "\n"
}
```

では、ブロックは10に付いてしまいます。10は、整数でメソッド呼び出しではありませんし、メソッド呼び出しでないものにブロックは付けられないので、これは文法エラーになってしまいます。この場合は、引数の周りにかっこを付けて、

```
0.upto(10) {|i|
  print i, "\n"
}
```

とすれば大丈夫です。

Smalltalk

Smalltalkにもブロックがあります。Smalltalkでは、ブロックはいろいろな形で使われています。実際、単純な条件判断さえ（少なくとも構文上は）ブロックを使ったメソッド呼び出しで実現されています。Smalltalkのブロックは、[] で括られた範囲です。Smalltalkの条件式は以下のように表現します。

```
cond ifTrue: [ ... ] ifFalse: [ ... ]
```

ifTrue:、ifFalse:はBooleanクラスのメソッドで、真であればifTrue:に指定されたブロックを、偽であればifFalse:に指定されたブロックを実行します。

寄り道が多いですが、この連載はあくまでもRubyの連載ですから、Smalltalkについてはこれ以上説明しません。興味のある方は、Smalltalkについて解説した書籍を参照してください。

Smalltalkのブロックは、Rubyのブロック同様にいろいろな目的に使われるのですが、その違いは以下のとおりです。

- **Smalltalkのブロックは（おおむね）独立したオブジェクトだが、Rubyのブロックはオブジェクトとしては渡されない（後述のとおり、明示的にオブジェクト化可能）。**
- **Rubyのブロックはメソッド特殊構文で、メソッド1つに対して1つしか指定できない。**

Rubyのブロックが、Smalltalkのように最初からオブジェクトではなく、明示的にオブジェクト化する必要がある理由は、私にははっきりとはわかりません。推測するに、CLUの構文から発生してきた歴史的事情か、デフォルトでオブジェクト化しないことによる効率の追求かの、いずれかの理由ではないかと考えられます。

繰り返し

さて、もともとループの抽象化から生まれたRubyのブロックですから、繰り返しは得意です。ループを実現するRubyのメソッドはたくさんありますが、すでに登場した整数クラスのtimes、upto以外にもたくさんあります。繰り返し系のメソッドを**表1**にまとめます。

collectは、やや特殊で使い勝手の良いメソッドですから、特に説明しておきます。collectは、各要素に対してブロックを評価し、その値を集めます。ブロックの値とは、一番最後に評価した式の値です。ですから、たとえば、

```
[1,2,3,4].collect{|i| i*2}        ⇨ [2,4,6,8]
```

を実行すると、配列の各要素に2を掛けたものを要素とする配列が得られます。

もう1つ、eachメソッドについても説明する必要があります。eachというメソッドは、配列などの各要素を1つずつブロックに渡す基本的な繰り返しメソッドです。そして、このeachメソッドはRubyのfor文に使われています。つまり、

```
for i in obj
  ...
end
```

というfor文は、実は内部的に、

メソッド名	クラス	機能
times	Integer	n回繰り返し
upto	Integer	nからmまで繰り返し
step	Integer	nからmまでsずつ繰り返し
each	Array、Hash、etc.	各要素に対する繰り返し
collect	Array、Hash、etc.	ブロックの評価値を集める

表1　繰り返し系メソッド

```
obj.each do |i|
  ...
end
```

というeachメソッドの呼び出しによって実現されています。ですから、eachメソッドを持っているオブジェクトであれば、どんな種類のオブジェクトでもfor文を適用することができます。

条件

ブロックの値を使って、条件を指定するタイプのメソッドもあります。このタイプのメソッドとしては、ArrayやHashクラスに定義されているselect、detect、rejectの各メソッドがあります。

selectは条件を満たす要素の配列を返します。

```
[1,2,3,4].select{|i| i % 2 == 0}          ⇨ [2,4]
```

detectは、条件を満たす最初の要素を返します。条件を満たす要素がなかった場合には、nilを返します。

```
[1,2,3,4].detect{|i| i % 2 == 0}          ⇨ 2
[1,2,3,4].detect{|i| i % 5 == 0}          ⇨ nil
```

rejectは、条件を満たす要素を削除した配列を返します。

```
[1,2,3,4].reject{|i| i % 2 == 0}          ⇨ [1,3]
```

コールバック

そのほか、ブロックには無限の応用があります。その一部を紹介しましょう。

ArrayやHashクラスに定義されているsortメソッドは、要素をソートした配列を返しますが、ブロックによって比較の条件を指定できます。

```
[1,6,5,4].sort                            ⇨ [1,4,5,6]
[1,6,5,4].sort{|a,b| b<=>a}               ⇨ [6,5,4,1]
```

この例では単に逆順にしただけですが、ここには任意の条件を指定できます。ブロックは<=>演算子と同様、2要素の大小関係に応じて、正の数（大きいとき）、ゼロ（等しいとき）、負の数（小さいとき）のいずれかを返す必要があります。

「‖」の間に変数が複数あるときには、多重代入と同じルールが適用されます。sortメソッドは2つの値をブロックに渡しますから、「｜a,b｜」でそれぞれに代入されるわけです。もし仮にこれを1つの変数で受けると、その変数には2要素の配列が代入されます。

```
[1,6,5,4].sort{|i| i[1]<=>i[0]}           ⇨ [6,5,4,1]
```

Stringクラスのsubメソッドとgsubメソッドは、文字列の置換を行います。subメソッドは最初の1回だけ、gsubメソッドはマッチする場所すべてを置換します。

これらのメソッドが引数を2つ受けた場合には、単に1番目の引数で指定されたパターンにマッチする部分を、2番目の引数で置換するのですが、引数が1つしか与えられず、かつメソッドにブロックが与えられると、そのブロックの値を使って置換します（メソッドはブロックにマッチした部分文字列を渡す）。

たとえば、マッチした部分を大文字にするためには、

```
"abcdefg".gsub(/[acf]/){|x| x.upcase} ⇨ "AbCdeFg"
```

のようにします。この形式は、置換する処理が複雑になるときに効果を発揮します。

Procオブジェクト

Smalltalkとの比較で、Smalltalkのブロックはオブジェクトだが、Rubyのブロックはそうではないと説明しました。しかし、もちろんすべてがオブジェクトのRubyのこと、ブロックもオブジェクトとして扱うことができます。

以下のメソッドを使うと、ブロックをオブジェクトして取り出すことができます。

```
proc
lambda
Proc.new
```

これらのメソッドはどれも同じ働きをします。lambdaというのはLispからきた名前です。好きな人はこれだけで嬉しくなってしまうようです。得られるのは、Procクラス

のオブジェクトで、実行にはcallメソッドを使います。

```
c = proc{print "foobar\n"}
c.call
```

上記のようにすると、callメソッドを呼び出した時点でfoobarを出力します。

procメソッド（またはその別名）は、与えられたブロックをオブジェクト化しますが、もしブロックなしで呼ばれたらどうなると思いますか？

なんとなくエラーになりそうにも思えますが、実はprocを呼び出したメソッドに与えられているブロックをオブジェクト化します。procにブロックが与えられず、procを呼び出したメソッドも、ブロックを与えられていないならエラーになります。

```
def mkproc
  proc                      # ブロックのオブジェクト化
end

mkproc{print "mkproc\n"}  # オブジェクト化
mkproc                    # エラー
```

オブジェクト化したブロックは、クロージャ（閉包）とも呼ばれます。クロージャと呼ばれる理由は、ブロックが環境（ローカル変数の値）も閉じ込めているからです。これはちょっと説明するのが難しいので、実例を見てみましょう（**リスト4**）。

▼リスト4　クロージャの実例

```
def closure
  n = 0
  a = proc{n += 1}
  b = proc{n -= 1}
  c = proc{print n, "\n"}
  return [a,b,c]
end

a,b,c = closure()
a.call
c.call  # => 1
a.call
c.call  # => 2
b.call
c.call  # => 1
```

わかりますか？ a、b、cの3つのProcオブジェクトはclosureメソッドの実行が終わってしまっても、そのローカル変数nを閉じ込めて共有しているのです。ですから、aを実行するたびにnの値は1増え、bを実行するたびにnの値は1減ります。そして、cを実行するたびに現在のnの値を出力するのです。

closureメソッドを実行するたびに違うクロージャが作られますから、もう一度closureメソッドを実行して得られるProcオブジェクトは、別のローカル変数を共有することになります。

```
c.call ⇨ 1
d,e,f = closure()
d.call
d.call
f.call ⇨ 2
c.call ⇨ 1
```

Procオブジェクトは、ブロックとして与えられた手続きを保存しておいて、後で実行する場合などにとても役立ちます。

たとえば、GUIライブラリでボタンが押された場合に、実行する処理がブロックで与えられたときには、与えられたブロックに対応するProcオブジェクトを生成して保存しておきます。そうすれば、ボタンイベントに応じて後でそのブロックを実行できるわけです。

ブロックの使い方

さて、ブロックを受け取るメソッドの使い方を説明してきたわけですが、実際に自分でブロックを受け取るメソッドを書くためにはどうしたらよいでしょう。

ブロックの使い方には、以下の方法があります。

- **yield**
- **proc/lambda/Proc.new**
- **ブロック引数**

1つずつ説明しましょう。

■yield

yield文は、メソッドに与えられたブロックに制御を移す構文です。yieldという名前は、CLUから来ているよう

です。yield文に渡す値が、ブロックの変数部に代入されます。ブロックの実行が終わると、またyieldの続きから実行を継続します。yield文の値は、実行したブロックで最後に評価した式の値です。

```
def yieldtest
  n = yield(22)
  print n, "\n"
end
yieldtest{|x| print x, "\n"; 55}
```

このプログラムを実行すると、ブロックにはyieldに与えられた値である22が渡され、ブロック内のprintによってそれが出力されます。ブロックの値は最後に評価された式、すなわち55で、これはyield文の値となって、yieldtestのローカル変数nに代入されます。そして、nの値がprintによって出力されるわけです。

実例として、Arrayクラスのeachメソッドを Rubyで実装してみましょう。

```
def each
  i = 0
  while i < self.size
    yield self[i]
    i += 1
  end
end
```

yieldに複数の値が与えられると、それを配列としてブロックに渡します。

つまり、

```
yield(1,2,3)
```

は、

```
yield([1,2,3])
```

と同じ意味です。yieldのかっこは、メソッド呼び出しと同様に省略できます。

この、複数の値が与えられると配列として渡す点と、かっこが省略できる点はreturn文も同じです。

■proc/lambda/Proc.new

すでに説明したように、procはブロックなしで呼ばれると、呼び出し元のメソッドに与えられているブロックをオブジェクト化します。これを利用して、ブロックをオブジェクト化し、callメソッドを使って実行させることができます。

procを使ったイディオムとしては、メソッド引数の省略値として使い、引数としてProcオブジェクトを指定しても、メソッドにブロックを与えても、どちらでも動くメソッドを手軽に定義するリスト5のような方法があります。

このイディオムはRuby/Tkなどでも用いられています。

```
btn = TkButton::new()
btn.command{print "hello\n"}
btn.command(proc{print "hello\n"})
```

■ブロック引数

「ブロック引数」とは、メソッドに間接的に渡されるブロックを明示的な引数として受け渡す方法です。ブロック引数を受け取るには、メソッド定義の末尾に「&」を付けた引数を追加します。この引数には、ブロックがProcオブジェクトとして代入されます。

```
def blkarg(arg, &block)
  block.call(arg)
end
blkarg(24){|x| print x, "\n"}  # 24を出力
```

メソッドにブロックが与えられなかった場合には、ブロック引数にはnilが代入されます。

ブロックを引数として受け取ることができるように、引数としてブロックを渡すこともできます。こちらも「&」を使います。

引数としてブロックを渡すには、メソッド呼び出しの引数の末尾に「&」を付けた式を追加します。式の値は、

▼リスト5　ブロックのオブジェクト化

```
def pp(block=proc)
  print "<p>"
  block.call
  print "</p>"
end

pp{print "test"}        # <p>test</p>
pp(proc{print "test"})  # 同じ働き
```

ProcオブジェクトまたはMethodオブジェクトである必要があります。

```
def blkpass(n)
   print yield(n), "\n"
end
p = proc{|x| x + 22}
blkpass(11, &p)             # 33を出力

m = [1,2,3].method(:index)  # Methodオブジェクトを取得
blkpass(3, &m)              # 2を出力
```

ブロック引数は、「暗黙の引数」ともいえるブロックを明示的に取り扱う方法です。ブロック引数を使えば、ブロックの引渡しを直接に行うことができます。

たとえば、再帰的な処理のためにブロックを渡したいとき、ブロック引数を使わなければ、以下のリストのようになります。

```
def traverse(ary)
  ary.each do |i|
    if i.kind_of?(Array)
      traverse(i){|x| yield x}
    else
      yield i
    end
  end
end
```

これをブロック引数を使って実現すると、次のようになります。

```
def traverse(ary, &block)
  ary.each do |i|
    if i.kind_of?(Array)
      traverse(i, &block)
    else
      yield i
    end
  end
end
```

ブロック引数を使ったほうが、直接的な印象がありますね。「{｜x｜ yield x}」というのは、ブロックに渡された値をそのまま上位のブロックに引き渡すイディオムですが、やや分かりにくい気がします。

ブロック引数を使えば、

```
def ev_test(obj, &block)
  obj.instance_eval(&block)
end
```

という方法で、メソッドに渡されたブロックをそのまま別のメソッドに渡すことができます（instance_evalはブロックをそのオブジェクトのコンテキストで評価するメソッド）。

しかし、これを上で紹介したyieldを使ったイディオムで、

```
def ev_test2(obj)
  obj.instance_eval{|x| yield x}
end
```

のように記述してもうまく動きません。instance_evalで評価されるブロックは、あくまでも直接渡したブロックだからです。ブロック引数は、ブロックを他のメソッドに直接渡す唯一の方法でもあります。

このように、利用の方法によっては、分かりやすい記述となるブロック引数なのですが、毎回Procオブジェクトを生成することになるため、yieldを用いた場合と比較してちょっと遅い傾向があるようです。適材適所で用いてください。

イレテータを征する者はRubyを制す

今月は、Rubyのブロック（イテレータ）について詳細に解説しましたが、おわかりいただけたでしょうか。テーマがテーマだけに、やや地味めな内容になってしまったかもしれませんね。

昔「イテレータを制するものはRubyを制す」ということわざ（？）があったそうです。呼び名がブロックに変わっても、これがRubyの大きな特徴であることに変わりはありません。ブロックを使いこなすことができれば、さまざまな場面でとても役に立つことは間違いないでしょう。

これを機会にして、ぜひブロックの使いこなしにチャレンジしてみてください。

Column

今月のRuby 1.5

今月は「嬉しさ」に注目して、Ruby 1.5の新機能などを紹介したいと思います。

■シンボルのオブジェクト化

先月は非互換部分として紹介したシンボルのオブジェクト化ですが、シンボルと整数というまったく違ったものが異なるオブジェクトとして分離されたことは、プログラムの書きやすさ、読みやすさ、エラーの見つけやすさに貢献します。

■ScriptError例外

これも非互換ですが、プログラムそのものの間違いによる例外をScriptError例外のサブクラスとして分離したことによって例外名を指定しないrescueが、StandardError例外のサブクラスをすべて捕捉してしまういわゆる「rescue文によるうっかり捕捉問題」が軽減されています。

■再帰的ワイルドカード展開

「あるディレクトリ以下で、拡張子が.cであるすべてのファイルの一覧がほしい」と思ったことはありませんか？ zshではこのような場合に、「＊＊/＊.c」というワイルドカードを使うことができます。1.5系では、以下のように、Dirクラスのglobメソッドにこの機能が追加されました。

```
list = Dir.glob("**/*.c")
```

■fnmatch（わたなべ版）の採用

これもワイルドカード関係です。fnmatchというのは、ワイルドカードによるマッチを行うライブラリ関数です。これはPOSIXで定義されている関数ですが、実は多くのプラットフォームで微妙に挙動が異なります。そこで、ライブラリのfnmatchを使うのを止めて、わたなべひろふみさんによるfnmatchを使うようになりました。

■rescue修飾子

Ruby 1.4以前では、例外を捕捉するには**リストc-1**のようにする必要があります。

つまり、まずcodeAがあり、「あ、例外を捕捉しなきゃ」と思った場合、beginから始まる結構な分量のコードを書き足す必要があります。

1.5系では、修飾子としてのrescueが追加されました。これを使えば**リストc-1**の例は、

```
codeA rescue codeB
```

になります。codeA、codeBともに、単純な文であった場合には相当すっきり書けますね。ただし、rescue修飾子では例外の種類が指定できません。ですから、いつもStndardError例外のサブクラスすべてを捕捉することになります。

■freeze

freezeとは、オブジェクトを「凍結」して、それ以降オブジェクトの状態を変更しようとしたときに、例外を発生させる機能です。1.4以前ではArray、Hash、Stringだけが対象であったfreezeですが、全オブジェクトが対象になりました。

```
a = [1,2]
a.freeze
a[0] = 1        # 例外
```

一度freezeしたオブジェクトを「解凍」する方法はありません。これはインタプリタが内部的にfreeze機能を使っていて、解凍されると都合が悪いからだそうです。

■環境変数RUBYOPT

環境変数RUBYOPTにオプションを指定すると、インタプリタ実行時にそのオプションも指定したと見なすようになりました。たとえば、インタプリタのデフォルト文字コードをいつもシフトJISにしたい場合は、次のようにします。

```
export RUBYOPT="Ks"
```

類似のワザとして、カレントディレクトリのファイルを優先的にロードする

```
export RUBYOPT="I."
```

というものもありますが、「セキュリティ上の理由からそれは避けよう」というアドバイスがメーリングリストで出ています。

■クラス変数

Rubyでは、クラス変数を実現したいときには定数を利用して、**リストc-2**のようにするのがイディオムでした。1.5系ではクラス変数が導入されましたので、**リストc-3**のように自然に記述できます。クラス変数は、名前が「@@」で始まる変数です。

リストc-1　Ruby 1.4以前のrescue修飾子

```
begin
  codeA
rescue
  codeB
end
```

リストc-2　1.5系以前のクラス変数

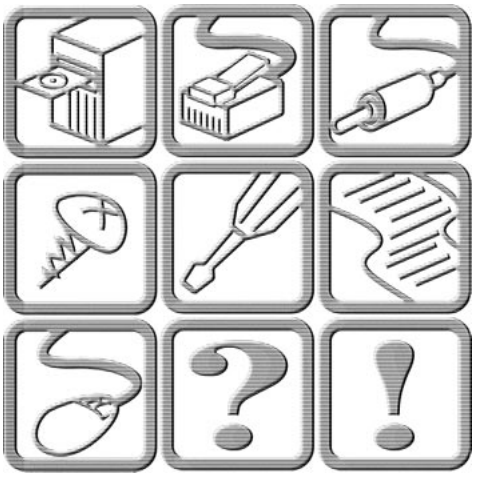
```
class Foo
  FooClassVar = [nil]
  def set_foo(n)
    FooClassVar[0] = n
  end
  def get_foo(n)
    FooClassVar[0]
  end
end
```

リストc-3　1.5系のクラス変数

```
class Foo
  @@FooClassVar = nil
  def set_foo(n)
    @@FooClassVar = n
  end
  def get_foo(n)
    @@FooClassVar
  end
end
```

Try & Try

[trái] [trái]



ACPI その2

文:政久忠由
Text : Tadayoshi Masahisa

昔の記憶というものは、漠然と思い出そうとしても思い出せるものではないし、そもそもそうやって思い出そうとすること自体が少ないものである。けれども、ふとしたきっかけで鮮明に思い出すことがある。きっかけは、視、聴、嗅、味、触という人が持つ五つの感覚からの刺激なわけだが、僕の場合、とりわけ音楽がそのトリガーとして機能しているようである。つい先日もどこかのTVドラマのテーマソングとして使われていたMARTIKAのTOY SOLDIERSを耳にして、当時の記憶が台風一過の晴天のように色彩やディテールを取り戻した鮮明な映像として僕の思考に流れ込んできた。

当時高校生だった僕は、受験勉強なんてくそ食らえ、という考えで、いつ終わるともしれない集中豪雨のように降り注ぐ課題や宿題には、興味のあるごく一部を除き、まったく手をつける気がしなかった（実際ほとんど提出していない）。唯一、頭を働かせていたのは、科学雑誌「SCIENTIFIC AMERICAN」を読むときだけだった。当然、当時の僕にこの雑誌の内容をすらすらと理解するだけの能力があるはずもなく（今なら大丈夫というわけでもないんだけど）、頭がオーバーヒートしているのを実感しながら、何度となく読み返し、なんとか読み進めていた（理解というには程遠いレベル）。

このオーバーヒートした頭を冷やすときによく聞いていたのが先ほどの曲である。天井を見上げ、密閉された空間で別の世界へ消えていくように拡散していくタバコの煙を広角的に追いながら、さまざまなとりとめもない思考をめぐらせていた。

ここであまり思い出話をしても仕方ないのでやめておくが、当時DNAを中心とするバイオテクノロジーが花盛りであった。それは今、ヒトゲノムの解析という大きな山場を迎えている。コンピュータ分野はというと、いまだにシリコン文化が発展し続け、ハードウェア性能も向上した。ソフトウェアも格段に操作しやすくなった。そしてインターネットという副産物は、僕たちの生活スタイルを一変する勢いの潮流となっている。インターネットは、地道な進歩が世界を変えるというひとつのよい例だ。けれども決定的な革新というものはまだ見えてこない。しかし、日の目を見ないだけで数多くの技術革新が潜在していることは確かだと思う。

そんなわけで、僕は今でも革新のきっかけとなり得る、ごくごく小さな新発見（大発見はちょっとおこがましいからね）に思いをはせながら、拡散していくタバコの煙を追っている。そしてたぶん思考できなくなるまで、そうしていくのだろうと思っている。

またもや前置きが長くなってしまったが、今回は前回に引き続きACPIについて見てみることにする。

システムのACPI構成情報

コンピュータシステムを構成するデバイスのオペレーテ

ィングシステムによるコンフィギュレーションとパワーマネージメント（電源管理）機構の標準化を目指しているACPIの概要は前回紹介した。オペレーティングシステムやデバイスドライバのサポートが重要であることは理解できていると思うが、今回は、ACPI BIOS（ファームウェア）の部分を見てみることにしようと思う。

最近のOSにおいて、いわゆるシステムBIOS ROMに格納された実行ルーチンは、機能的に貧弱でマルチタスク処理の足を引っ張ることから、システムの初期化時に実行するだけで、制御権がOSに移行してからは基本的に使用されることはない。またシステムBIOS ROMに収められた、もしくはBIOS経由で取得できるハードウェアの情報についても同様で、OSは起動時に吸い上げ、そのコピー情報を保持することで対処し、稼動中にBIOSに問い合わせることもない。ハードウェアの操作は、すべてデバイスドライバが直接アクセスすることで行われている。

ACPI対応のシステムでは、各デバイスのACPI関連の操作は、ACPI BIOS、ACPIレジスタ、ACPIテーブルのそれぞれのインターフェイスを介して行うようになっている。ACPI BIOS、ACPIレジスタ、ACPIテーブルは、各システム、各ハードウェアデバイスの有するACPIサポートに直結したものである。これらの操作は、各デバイスを担当するデバイスドライバが直接行うのではなく、ACPIを統括するドライバが一手に引き受けて処理をしている。このような構成になっている理由は言うまでもないと思うので省略するが、まぁ当たり前の常套手段である。

ここでこれらのシステムのACPIに関する情報はどこに格納されているのだろうか？　一般的なPCシステムの場合、先に述べたシステムBIOS ROMの一部として格納され0xe0000～0x100000のメモリ空間にマップされている。もちろん通常のシステムBIOS同様、これを直接利用することはせず、OSは起動時にこのACPI情報を読み取り、カーネル（ACPIドライバ）が管理するACPI用のメモリ空間にコピーして、利用、そして動的な変更を含め操作している。

これからシステムのACPI情報を見ていこうと思うが、その前にいくつかの関連用語の説明をしておこう。

●RSDP（Root System Description Pointer）

ACPI情報のスタートポイント（だからルートと呼ばれる）で、次に説明するRSDTの位置を表している。

●RSDT（Root System Description Table）

RSDPによって示されたこの領域は、システムのACPI情報の目次的な役割を持つ。ここの内容は、各エントリとそのポインタというインデックス情報が納められている。

●FACP（Fixed ACPI Description Table）

先のRSDPで示されたエントリのひとつであるFACPは、システムごと（依存した）のACPI構成の情報を示している。FACPには、後述するDSDTやFACS、ACPIハードウェアレジスタブロック（PM1x_BLKなど）と呼ばれる各ハードウェアデバイスの情報が納められている。

●DSDT（Differentiated System Description Table）

DSDTは、FACPに含まれ、各システムにおける実際のデバイスの実装や構成情報が納められている。OSは、システムの起動時に、いわゆるACPIネームスペースにこの情報を取り込みACPI操作のためのベース情報として活用する。ここに納められた情報は、各オブジェクトデータ（各デバイスと考えればよい）の構成と、それに対する各種操作を既定するメソッドと呼ばれるデータで構成され、これらはツリー状に配置されている。またこれらの情報は、OSの稼動時に動的に変更することもできる。

●FACS（Firmware ACPI Control Structure）

FACSは、OSとファームウェアとのやり取りに使用されるメモリ空間として定義されている領域だ。wake vector、shared lockのために利用される。たとえば、システムをスリープ状態に移行したとき、OSはシステムの制御、特にプログラムカウンタと呼ばれるプロセッサに処理させるメモリのアドレスの情報を失うことになる。実際、目覚める作業はOSがコントロールするというよりも、ACPI BIOSが実権を握っているのだが、FACSにその覚醒の際に処理を再開する物理メモリアドレスを納めることで、ACPI BIOS（ファームウェア）にエントリポイントの指示が行えるようになっている。つまりプロセッサの初期化後、ACPI BIOSによって、この物理アドレスがプロセッサに渡され、やっとOSに処理が戻り、再開となる。

よりわかりやすいかどうかはわからないが、とりあえず依存関係を整理すると、システムのACPI情報はRSDPをスタートポイントとし、このRSDPはRSDTの場所を示している。RSDTには、1つ以上、FACPなどのシステム依存ACPI情報のインデックスが収められている。FACPは、

ACPIのジェネリックなレジスタ情報のほか、システム依存のACPI情報の実体である各オブジェクトとメソッドを定義したDSDT、FACSなどで構成されているというわけだ。そして、これらの情報は、すべてOS管理でOSの初期化時にROMから読み出され、その後は状況に応じてOSが自由にデータ内容を変更しながら操作できるのである。前回説明した各デバイスのC0～C3やD0～D4といった状態の遷移は、DSDTで定義されている情報に基づきACPIドライバ経由で操作されるというわけだ。

前回、ACPIドライバを有効にすると/proc/sys/acpiにいくつかの項目が現れることは説明したが、それぞれの内容までは触れなかった。でも、これまでの説明でdsdtとfacpが何者であるかは分かったと思う。ただ、バイナリデータなのでcatしても有意義なデータとしては表示されないので注意してほしい。

ACPI構成情報を見てみる

システムのACPI構成情報を見るために、ここでは前回紹介したACPI4Linux project（http://phobos.fs.tum.de/acpi/）で提供されているpmtoolsを利用する。このpmtoolsには、ACPIテーブル情報のダンプツールとそれを視覚的に表示するためのフィルタ変換ツールで構成されている。

まずは、acpidmpコマンドを実行してみよう。オプションを付けずに実行するとスタートポイントであるRSDP以下のすべての内容が項目ごとに見出しとそのメモリアドレスが付加されて、ニーモニックとともに表示される。通常、1画面内には収まりきらないので、適当なページャを利用しよう。**リスト1**がその内容の一部だ。

RSDPは、アドレス0xf6aa0から始まっていて、その内容は、ACPIの提供OEM名はPTLTD、そしてそのRSDTは、0x0bffd02eから始まっているということが示されている。ここで指し示しているメモリアドレスは最後の4バイトが該当するのだけれど、リトルエンディアンプロセッサ（x86系）の場合、バイトオーダーの関係で2e d0 ff 0bのようにメモリ内容が表示されることになるので注意しておこう。特に知っておく必要はないのだが、実はRSDPの場所は、ACPI BIOS ROMがマップされる0xe0000から0x100000の範囲ということしか分からない。そこで、そのメモリ範囲を頭のRSD PTRというシグネチャで検索することで見つけているのだ。このRSDPを見つければあとはポインタで指し示されているのでそれをたどるだけで済むようになっている。

RSDT以降は、見やすくするためにフィルタを通して見てみる。以下の例では、acpidmpの引数として該当するテーブル名（RSDT）を指定してその部分だけを表示するように指定し、その内容をacpitblで整形している。

▼リスト1　ダンプしたACPI構成情報の一部

```
# ./acpidmp | less
RSDP "PTLTD" @ 0x000f6aa0
  0000: 52 53 44 20 50 54 52 20 31 50 54 4c 54 44 20 00  RSD PTR 1PTLTD .
  0010: 2e d0 ff 0b                                      ....

RSDT @ 0x0bffd02e
  0000: 52 53 44 54 2c 00 00 00 01 cf 50 54 4c 54 44 20  RSDT,.....PTLTD
  0010: 20 20 52 53 44 54 20 20 00 00 04 06 20 4c 54 50    RSDT  .... LTP
  0020: 00 00 00 00 65 fb ff 0b d9 fb ff 0b               ....e.......

FACP @ 0x0bfffb65
  0000: 46 41 43 50 74 00 00 00 01 0a 4d 41 54 42 49 4f  FACPt.....MATBIO
  0010: 43 46 2d 4d 31 45 20 20 00 00 04 06 4d 41 54 20  CF-M1E  ....MAT
  0020: 00 00 00 01 c0 ff ff 0b 5a d0 ff 0b 00 00 09 00  ........Z.......
  0030: b2 00 00 00 f0 f1 f2 00 00 10 00 00 00 00 00 00  ................
  0040: 04 10 00 00 00 00 00 00 22 00 00 00 08 10 00 00  ........".......
  0050: 0c 10 00 00 00 00 00 00 04 02 01 04 04 00 00 00  ................
  0060: 01 00 e9 03 00 00 00 00 01 03 0d 00 32 00 00 00  ............2...
  0070: a1 00 00 00                                      ....

DSDT @ 0x0bffd05a
  0000: 44 53 44 54 0b 2b 00 00 01 67 4d 41 54 42 49 4f  DSDT.+...gMATBIO
  0010: 43 46 2d 4d 31 45 00 00 00 00 04 06 4d 53 46 54  CF-M1E......MSFT
  0020: 0b 00 00 01 10 12 5f 50 52 5f 5b 83 0b 43 50 55  ......_PR_[..CPU
  0030: 30 00 10 10 00 00 06 10 4b 1a 5f 53 42 5f 5b 80  0.......K._SB_[.
  0040: 53 4d 49 30 01 0c 00 fe 00 00 0c 02 00 00 00 5b  SMI0...........[
  0050: 81 0b 53 4d 49 30 00 53 4d 49 43 08 5b 80 53 4d  ..SMI0.SMIC.[.SM
  0060: 49 31 00 0c bc fd ff 0b 0c 00 02 00 00 5b 81 16  I1...........[..
（以下省略）
```

```
# ./acpidmp RSDT | ./acpitbl
Signature:        RSDT
Length:           44
Revision:         0x01
Checksum:         0xcf
OEMID:            PTLTD
OEM Table ID:     RSDT
OEM Revision:     0x06040000
Creator ID:       LTP
Creator Revision: 0x00000000
```

RSDTには、これといって特筆すべきものがないので同様に次のFACPを見てみる。

```
# ./acpidmp FACP | ./acpitbl
Signature:        FACP
Length:           116
Revision:         0x01
Checksum:         0x0a
OEMID:            MATBIO
OEM Table ID:     CF-M1E
OEM Revision:     0x06040000
Creator ID:       MAT
Creator Revision: 0x01000000
FIRMWARE_CTRL:    0x0bffffc0
DSDT:             0x0bffd05a
INT_MODEL:        0x00
SCI_INT:          9
SMI_CMD:          0x000000b2
ACPI_ENABLE:      0xf0
ACPI_DISABLE:     0xf1
S4BIOS_REQ:       0xf2
PM1a_EVT_BLK:     0x00001000
PM1b_EVT_BLK:     0x00000000
PM1a_CNT_BLK:     0x00001004
PM1b_CNT_BLK:     0x00000000
PM2_CNT_BLK:      0x00000022
PM_TMR_BLK:       0x00001008
GPE0_BLK:         0x0000100c
GPE1_BLK:         0x00000000
PM1_EVT_LEN:      4
PM1_CNT_LEN:      2
PM2_CNT_LEN:      1
PM_TM_LEN:        4
GPE0_BLK_LEN:     4
GPE1_BLK_LEN:     0
GPE1_BASE:        0
P_LVL2_LAT:       1
P_LVL3_LAT:       1001
FLUSH_SIZE:       0
FLUSH_STRIDE:     0
DUTY_OFFSET:      1
DUTY_WIDTH:       3
DAY_ALRM:         0x0d
MON_ALRM:         0x00
CENTURY:          0x32
Flags:            0x00000000
```

FACPでは、システム依存であることをうかがわせるヘッダ情報とともに、DSDTとFACS（FIRMWARE_CTRL）のスタートアドレス、そしてACPIレジスタの値が含まれていることが分かる。＊_BLKがACPIレジスタと呼ばれるもので、ハードウェアのチップセットのレジスタに直結したものである。なかには/proc/sys/acpiの項目に対応するものもあるが、/proc/sys/acpiの項目のいくつかは、ACPIドライバの動的なデータとして保持されている。DSDTとFACSについても同様に見ておく。

```
# acpidmp DSDT | acpitbl
Signature:        DSDT
Length:           11019
Revision:         0x01
Checksum:         0x67
OEMID:            MATBIO
OEM Table ID:     CF-M1E
OEM Revision:     0x06040000
Creator ID:       MSFT
Creator Revision: 0x0100000b

# acpidmp FACS | acpitbl
Signature:              FACS
Length:                 64
Hardware Signature:     0x00000ff1
Firmware Waking Vector: 0x00000000
Global Lock:            0x00000000
Flags:                  0x00000000
```

ACPIのオブジェクトとコントロールメソッド

acpitblフィルタを通して、ざっとACPI構成情報を見た気分になったが、実はもっとも重要な部分をまだ見ていない。DSDTの本当の内容、ACPIのオブジェクトデータとそのコントロールメソッドである。acpitblフィルタではこれらの情報を表示することができないので、今度は、acpidisasmフィルタを使用する。

```
# acpidmp DSDT | acpidisasm | less
00000000: Scope _PR_
00000006:   Processor CPU0
0000000d:     0x00
0000000e:     0x00001010
00000012:     0x06
000027ff: Scope _SB_
0000001a:   OpRegion SMI0
00000020:     0x01
00000021:     0x0000fe00
00000026:     0x00000002
0000002b:   Field SMI0
00000032:     0x00
00000033:     NamedField SMIC
```

ACPIのネームスペースでは、オブジェクトデータとそのコントロールメソッドはツリー構造になっていると説明したが、これがその内容である。ルートの直下には、Scopeとして表示されている5種類の大きなカテゴリ（パッケージとも呼ばれる）、_PR、_SB、_SI、_GPE、_TZがあり、それぞれプロセッサオブジェクト、デバイス／バスオブジェクト、システムインジケータオブジェクト、汎用イベントレジスタブロックオブジェクト、温度管理オブジェクトが含まれている。実際にScopeを検索してみると上記のカテゴリのほかに、_SBはISAやPCI0といったバスでカテゴリ分けされていることがわかる。

各デバイスオブジェクトは、Deviceというキーワードで示されていて、それぞれいくつかのデータオブジェクトとコントロールメソッドで構成される。特にコントロールメソッドは、AML（ACPI Control Method Machine Language）コードと呼ばれる中間コードで記述されており、OSが有するインタープリタでデコードして利用することになる。ちなみに_STAは、該当するデバイスの電源状態を取得する手順を意味する。またPSRVは電源状態を取得する手順の汎用的な部分を抜き出して、各デバイスのメソッド_STAなどから再利用できるようにしたコントロールメソッドである。

#コントロールメソッドの例

```
000013a6:         Method _STA
000013ac:           0x00
000013ad:           If
000013af:             LNotEqual
000013b1:               And
000013b2:                 MethodCall
00000000:                   PSRV (00000096)
000013b6:                   0x9b
000013be:                   0x03
000013c0:                 0xff00
000013c3:                 <NULL>
000013c4:               0x0200
000013c7:             Return
000013c8:               0x0f
000013ca:           Else
000013cc:             Return
000013cd:               0x00
```

コントロールメソッドの内容には、条件分岐や他のメソッドの呼び出し（MethodCall）があることもわかると思う。ちなみに、同じスコープ内ではメソッド名だけを指定しているが、スコープ外のものは_SB_.PSRVというような呼び出しが行われるようになっている。

次にスリープ状態の変移で利用されるコントロールメソッド_PTSや_PSxを見てみよう。_PTSコントロールメソッドは、システムをスリープ状態にするための準備手順である。また_PS0から_PS3は、該当するデバイスをD0からD3にそれぞれ移行するための手順だ。_PTSでは、システムのモード遷移に向けて各デバイスの電源管理メソッドを呼び出していっているのがわかる。またテスト環境システムのIDEデバイスでは、D0とD3のメソッドだけが定義されていて、D1やD2は用意されていない。電源オンとオフしかないということだ。

#_PTSコントロールメソッド

```
00001e1a: Name _S0_
00001e21:     0x03
```

```
00001e22:     0x05
00001e28: Name _S3_
00001e2f:     0x03
00001e30:     0x01
00001e44: Name _S5_
00001e4b:     0x03
00001e4c:     0x00
00001e9c: Method _PTS
00001ea3:   0x01
00001ea4:   Store
00001ea5:     Arg0
00001ea6:     \_SB_.CPTS (00001e52)
00001eb0:   If
00001eb3:     LAnd
00001eb4:       LEqual
00001eb5:         \_SB_.CPTS (00001e52)
00001ebf:         0x01
00001ec1:       LEqual
00001ec2:         \_SB_.CSST (00001e5f)
00001ecc:         0x03
00001ece:     MethodCall
00000000:       \_SB_.PSRV (00000096)
00001ed8:       0xa5
00001ee2:     MethodCall
00000000:       \_SB_.PSRV (00000096)
00001eec:       0xa6
00001ef6:   Else
00001ef8:     MethodCall
00000000:       \_SB_.ECWR (0000016a)
00001f02:       0x1d
00001f06:     MethodCall
00000000:       \_SB_.PSRV (00000096)
00001f10:       0xa7
```

（以下省略）

#デバイスIDEの電源管理コントロールメソッド

```
00000f6f:       Method _PS0
00000f75:         0x00
00000f76:         Store
00000f77:           0x00
00000f79:           SVFL (00000f9c)
00000f7d:       Method _PS3
00000f83:         0x00
00000f84:         If
00000f86:           SVFL (00000f9c)
00000f8a:           Store
00000f8b:             B20_ (00000fac)
00000f8f:             R20_ (00000e68)
00000f93:           Store
00000f94:             B04_ (00000fa3)
00000f98:             R04_ (00000e60)
```

これらの内容は、前回紹介した仕様書を参考にしながら見ていくとある程度何をしているかが分かると思う。

ちなみに、ここではacpidmpを利用してACPIの構成情報を取得した。acpidmpは/dev/memをオープンし、まずRSDPを検索し、あとは内容を解釈してポイント先のデータを取得している。FACPとDSDTに関しては、/proc/sys/acpiのfacpとdsdtを参照することで取得できるので、次のようにすることもできる。まぁ好きな方法で見てもらいたい。

```
cat /proc/sys/acpi/facs | acpitbl
cat /proc/sys/acpi/dsdt | acpidisasm
```

今回見てきたのは、あくまでACPI BIOS ROMのファームウェアの内容をOSで取り込んだところまでで、これから先、これらの情報を利用して各デバイスを操作するには、カーネル、ACPIドライバ、各デバイスドライバに適切な処理が含まれていなければならないし、一般にはデバイスが一定期間ある状態が経過するとスタンバイ、スリープモードへ移行させるデーモンのようなサービスも必要なのだが、Linuxへの実装は、まだ開発段階で完全に効率よく機能するまでには至っていない。そのうち整備されると思うが、実際にノートPCなどモバイル環境で使用することを考えると、残念ながら現状は操作環境の整った従来のAPMを利用したほうが便利な状況なのである（現段階ではACPIを使用するとバッテリ状態も自分で作らないと見られないんだなぁ、これが）。

まぁ、今のところハードウェア環境がそれほどACPIでなくちゃダメという状況でもないから、そのうちでよいといえばよいのだけれど。

次回は、カーネル2.4で実装される新しいネットワーク／パケットフィルタリング機能とその構成を設定するiptableツールを使用してみようと思う。

ツールを組み合わせることの妙

Emacsはじめました

第4回　メールを使う

Emacsパワー全開。いよいよテキストエディタの殻をやぶって、さまざまなアプリケーションを渡り歩く放浪の旅に出かけます。今回は現代に生きるLinuxerの死活生命線、電子メールに進出です。

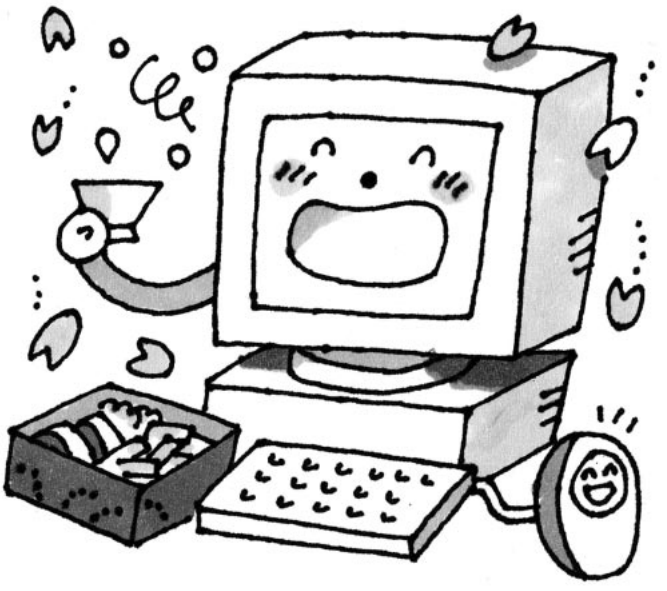


Illustration : Manami Kato

文：佐々木太良
Text : Taroh Sasaki

暮らしのEmacs

みなさん、元気にEmacsしてますか？　これまでの3回でエディタの基本操作、そして日本人なら味噌汁、納豆と並んでたいせつな（え？ 納豆はお嫌い？）日本語入力を見てきました。今回からは、Emacsをアプリケーションとしてばりばり使っていきます。名付けて「Emacsの中で暮す」編。

まず手始めはメーラです。むろんEmacsはテキストエディタで、それ自体にメーラの機能はありません。ところが、Emacs Lisp（elisp）で書いたプログラム（マクロ）を実行させ、外部のプログラムと通信できることを利用して、Emacsで編集した文章をそのままメール関連の外部プログラムに渡しちゃおう、と思いついた人がいたわけです。

今回は、そうしたマクロのなかでも代表的な、Mewの仕組みと操作を解説します。そして次回により高度な使い方を紹介したあと、次世代のメーラとして注目されているWanderlustを見ていきます。

メール配達の仕組み

一般に、ユーザーがメールをメールサーバから手元に取ってきたり、手元からメールサーバに送り出すのに使うソフトウェアを、MUA（Mail User Agent）といいます。MUAにはこのほか、メールの読み書きや整理の機能もあるのが普通ですが、本質ではありません。これに対してメールサーバと呼ばれるソフトウェアをMTA（Mail Transfer Agenet）といいます。MTAは常時ネットワーク上で動作し、MUAから受け取ったメールを中継して相手先のMTAに届けます。

UNIX系の有名なMUAをいくつか**表1**に挙げます。最後の2つは、Emacsから駆動するのではなく、単体で動作するものです。それ以外は、もちろん単体でも動作しますが（達人はよく「生で」といいます）、Emacsから使ってやるととても使い勝手が良くなります。では、どうしてわざわざEmacsと組み合わせるのかいうと、2つの理由が挙げられるでしょう。ひとつはEmacsの強力な編集機能を利用してメールを書けること（読んだり整理することも）、もうひとつはカスタマイズが簡単で超強力だからです。このほか日本人の場合、シェルから駆動して「生」で使うと、ツールによっては日本語が入力できなかったり、日本語のドキュメントを表示させただけで謀反を起こしてしまうものがあります。こんなとき、MUAと日本語をやりとりするのがシェルではなくEmacsなら安心だね、というわけです。

さらに近年のメールでは、MIMEやPGPというオプション形式が普通にやりとりされるようになってきました。MIMEはメールに画像などのデータを「添付」するのに使われる規約です。PGPは暗号化の技術で、内容を暗号化して途中で傍受されても読めないようにしたり、「このメールを書いたのは（なりすました別人じゃなくて）確かにオ

MUA	Emacsからの駆動に使用するマクロ
Mail（ucbmail）	mail
MH	mh-e
	Mew （ver. 1.7まで）
im	Mew （ver. 1.85以降）
	Wanderlust
Netscape	——
kmail	——
⋮	

表1　MUAと、Emacs対応のマクロ

レだよ」という電子署名に使われます。これらに対応していない古典的なツールでは、自分で別に作成した添付データや暗号化部分を付加してやらなければならないので面倒でした。そういった定型操作をEmacsにさせてしまおうというのがMew以降の近代的なメーラです。

■UNIXのメーラの歴史

さてちょっとだけUNIXのメーラの歴史を見てみましょう。mailコマンドは古くから標準でUNIXに備わっていたもので、MUAがMTAを兼ねているという古典的なスタイルです。シンプルながら、メールボックスの整理機能などもあります。しかし、日本語に対応したバージョンでないかぎり、日本語を使うのは困難であるといえましょう。

メールサーバが管理するシステムメールボックス（郵便局の私書箱だと思ってください）に届いたメールは、mailによってinboxと呼ばれるユーザーごとのメールボックス（ご家庭の郵便受け）に取り込まれます。その後読んだり捨てたり、必要ならば内容や差出人ごとに分けたフォルダ（状箱）に整理します。この基本スタイルは、その後登場したより高機能なフリーソフトウェアのMHやimでもほぼ同じです。

MHは少々変わっていて、メール操作のひとつひとつがそれぞれシェルから起動できるコマンドになっています。つまり、mailコマンドの場合は、いったん起動したあと終了するまでmailのプロンプトに対してコマンドを入力して操作するのですが、MHでは「次のメールを読む」「捨てる」「返事を書く」などという操作がバラバラのコマンドになっていて、独立したUNIXコマンドとして起動するのです。

Mewは当初、メールの取り込みやメールボックスの整理にMHのコマンド群を駆動していましたが、その後メールとニュースの統合を目的にMHの代替となるimを作り、これを駆動するように変更されています。imはPerlで書かれていることも特色のひとつで、現在も改良が続けられています。また、POP3で「取り込んだあともシステムメールボックスにメールを残す」という設定をした場合、MHではシステムメールボックスに残したメールを次回に

Column

○POPとIMAP

現在、メールサーバのシステムメールボックスに届いたメールを、ローカルにあるユーザーのメールボックスに取り込むときには、POP3プロトコルが広く使われています。POP3では、メールをローカルにあるユーザーのメールボックスで管理することが前提で、メールサーバからユーザーのメールボックスに取り込んだメールは、基本的にはサーバには残りません。サーバから削除しないように指定して取り込むことも可能ですが、いくつかのマシンからメールを読むような場合には、どこまで取り込んだかの情報の整合性がとれないので、ユーザーのメールボックスの管理がやっかいになります。

これに対して、メールをメールサーバ側のメールボックスで管理することを前提としたIMAP4プロトコルを使う方法があります。郵便物にたとえれば、「内容別」や「差出人別」の状箱は郵便局に置き、必要なものだけ郵便配達人に配達してもらって家の状箱にコピーする感じです。ユーザーにしてみれば、ローカルに多数のメールを置かなくてすみ、何台かのマシンからメールを読み書きするにも好都合です。反面、郵便局はあっという間に建物がパンクしてしまうでしょう。限られたユーザーがゆとりのあるメールサーバを使う場合はともかく、不特定多数のユーザーをかかえる大規模なメールシステムの運用ではまだまだ導入が躊躇されることでしょう。

ディストリビューション（バージョン）	TurboLinux 4.2	TurboLinux 6.0	Vine 1.1CR	Vine 2.0	Red Hat 6.2J	LASER5 6.0
Emacs+mule	19.34+2.3		19.34+2.3			
Emacs				20.5	20.5	20.5
XEmacs (with mule)	20.4	21.1		21.1		
Mew	*4	*1	*4	*4	*2	*2
Wanderlust			*2	1.0.3*3	*2	*2
ほかのメーラ	kmail	kmail	vmail	vmail	kmail	kmail

*1：XEmacs21以降にMewは標準装備
*2：RPMから取ってくりゃいいじゃん
*3：WanderlustはXEmacs21.1にのみ対応
*4：標準装備、設定が必要

表2 ディストリビューションとEmacs/メール環境

再度取り込んでしまうことがありましたが、imではこの問題を回避できます。なお、Mewは現状、POPだけでなくIMAPプロトコルを選ぶこともできますが、完全な対応は今後の課題となっています。

Wanderlustは、見た目がスマートで、現時点でIMAP4を利用するもっとも優れたツールだろうと考えられます。

Mewが使えるどうか確かめる

さてうんちくはこれくらいにしましょう。まずはMewを使えるようにする準備からです。準備は、システムにMewがインストールしてあることと、個人設定の2つですが、後者についてはいろいろとカスタマイズすれば便利になりますので、あとで別に見ていきましょう。

Mewそれ自体はEmacs、XEmacsのバージョン各種に対応した版があります。Red Hat6.0J、LASER5 6.0などは、RPMを利用してインストールしてください。XEmacs21以降は標準のパッケージとして付属しているので、特別なインストールはいらないでしょう（**表2**参照）。

まず、Mewが起動できるか試してみましょう。もしMewが使える状態になっていれば、**M-x mew**を実行したとき、Emacsなら“MEW”をかたどったロゴが回転します（**画面1**）。XEmacsなら可愛い2匹の子猫の写真が表示されます。

Mewはインストールされているけれども個人設定の問題で使えないときには、たとえば次のようなメッセージが

画面1　Mew起動時の画面（Mew＋Emacs19）

表示されることがありますが、慌てることはありません。「メーラを設定する」に読み進んでください。

```
Mew errors: Must set 'mew-mail-domain-list'
```

■Mewのインストール

気をつけなければならないのは、Mewは内部でimを使っているということです。たとえEmacsのマクロとしてMewがインストールされていても、imがインストールされていなければ使えません。またimはPerlで書かれているので、インストールされているPerlがversion 5.004_04以降でないといけません。これは、シェルから“perl --version”とすれば確認できます。

▼リスト1　Mewの設定

```
(setq load-path
      (cons (expand-file-name "/usr/local/share/mule/site-lisp/mew")
        load-path))
(require 'mew-mule-startup)
;
(autoload 'mew "mew" nil t)
(autoload 'mew-send "mew" nil t)
(setq mew-mail-domain-list '("taroh.org"))
(setq mew-icon-directory "a directory where Mew's image files are installed.")
(autoload 'mew-user-agent-compose "mew" nil t)
(if (boundp 'mail-user-agent)
    (setq mail-user-agent 'mew-user-agent))
(if (fboundp 'define-mail-user-agent)
    (define-mail-user-agent
      'mew-user-agent
      'mew-user-agent-compose
      'mew-draft-send-letter
      'mew-draft-kill
      'mew-send-hook))
```

インストールしたディレクトリなど

使用者のメールドメイン名に関する情報

その他、必要に応じて変えてやる情報など

MewとimはRPMを使ってインストールすることができます。また、RPMによらず自力でインストールしたいときは、ftp://ftp.Mew.org/pub/Mew/に行ってみれば（NetscapeなどのWebブラウザで開いてみてください）、Mewおよびimのさまざまなバージョンがダウンロードできるようになっています。

まずはここからmew-faq.tar.gzというファイルを適当なディレクトリにダウンロードして、

```
% tar xvzf mew-faq.tar.gz
% netscape FAQ/index-j.html
```

として見てみましょう。詳しいインストール方法は、このドキュメント（FAQ－よくある質問）に書いてあります。

メーラを設定する

Mewを自分で設定するなら~/.emacs（または$EMACS/site-lisp/site-start.el）に**リスト1**のような設定が必要です。ディストリビューション付属のMewを起動してみたら、Mewもimもあるのにエラーが出て挙動不審、という場合もここをチェックしてみましょう。最低限、Mewをインストールしたディレクトリが設定してあれば動きます。最近のMewは、メール用のディレクトリを作るなど、面倒くさいいろいろな設定を最初の起動時に済ませてくれます。しかし、さすがにメールドメイン名に関する情報などは自動では設定されませんので、それも設定します。

次に、imの設定をしなくてはいけません。imは、個人ごとにホームディレクトリの下の隠しディレクトリ~/.im/以下に設定をするのですが、これを対話形式で行ってくれるimsetupというツールがimには付属しています。

imsetupから示される一連の質問は、先の質問にどのように答えたかによって変わります。ここでは、ごく一般的な「POP3を使ってメールを取り込み、サーバには残さない」という条件を想定してリスト2に設定例を示します。

imsetupからの質問には、デフォルトの設定値が［……］内に示されます。とくに変える必要がなければ、〈Enter〉キーを押します。POPを使用してプロバイダからメールを取り込む場合、**リスト2**と同じ選択をし、POPサーバ名とSMTPサーバ名とアカウントをプロバイダから指定された内容に設定してください。

ネットワーク接続を確認する

Linuxマシンはネットワークにちゃんと接続されていると仮定します……と書きたいところですが、ダイヤルアップユーザーの場合、PPP（または互換ツール）の設定やモデムとの接続、LAN接続の場合のネットワークカードの設定などなど、ネットワークにちゃんと接続することはなかなかの難関です。メールをうまく読み書きできないとき、問題がネットワーク接続にあるのか、Mewの設定なのか

▼リスト2　imsetupの実行例

```
taroh@zephyr.taroh.org[166]imsetup
Where is your home directory? [/home/taroh]
Where is your Mail directory? [/home/taroh/Mail]
Where is your News directory? [/home/taroh/News]
What is your E-mail address (es)? [taroh@taroh.org,tarohfwd@taroh.org]
What kind of mail spool do you use? (local/POP/IMAP) [local]
What kind of POP authentication mechanism? (POP/APOP/RPOP) [POP]
What is your POP account name? [taroh] tarohfwd
What is your POP server name or IP address? [mail.taroh.org]
Do you want to preserve messages?
0 (delete immediately), -1 (preserve forever),
N > 0 (delete messages after N days)  [0]
What is your SMTP server name or IP address? [mail.taroh.org]
Do you want to use value of Content-Length header for delimitation for local
mail? (Answer yes if your OS supports Content-Length header like Solaris 2.x, otherwise answer no.) [no]
Does your system can detect write errors without fsync(2)? (You can answer yes, if your home directory is on local
file system, otherwise answer no.) [no]
/home/taroh/.im is already exist.
Backup /home/taroh/.im/Config to /home/taroh/.im/Config.bak.
Setup /home/taroh/.im/Config.
```

ホームディレクトリの位置
メールボックスの作成場所
ニュースの保存場所
使用するメールアドレス
メール取り込みの方法
POP使用時のPOP認証方式
POPサーバ上でのアカウント名
POPサーバ名
取り込んだメールをサーバから削除するなら0、保存するなら1、N日間保存するなら数字N
SMTPサーバ名
Content-Lengthヘッダを付けるかどうか
NFSサーバ上にhomeがあればyesのこともある

がわからないと往生してしまうでしょう。

そこでまず、ネットワークに接続されている（はずの）状態でシェルから次のコマンドを実行して、パケットがサーバに届くかどうか確かめてみます（“ mail.hoge.ne.jp ”は、プロバイダから知らされているSMTPサーバ名やPOP／IMAPサーバ名に置き換えてください）。

```
% ping mail.hoge.ne.jp
64 bytes from 123.45.67.89: icmp_seq=1 ttl=255
time=0.043 ms
64 bytes from 123.45.67.89: icmp_seq=1 ttl=255
time=0.043 ms
^C
```

のように応答が返ってきたら接続はOKです（動かないとすればMewの設定が悪いのです）。ちなみに、このコマンドを止めるときは^C（Emacs的に書くとC-c）を入力します。

接続を確認したら、メールを取り込んでみましょう。Emacsが起動した直後（Mewを1回も起動していない状態）からMewを起動するには、M-x mewコマンドを使用します。例のアニメーションまたは子猫が表示されます。

このほか、いきなりメールを書くときは**M-x mew-send**による起動方法もあるのですが、筆者の場合「書く前に読む」主義で、またメールが読める状態になっていればワンアクションでメールを書ける状態に移行できるため、M-x mew-sendを使ってMewを起動することはまれです。

Emacsの使用中に再度メールを取り込みたくなったら、M-x mewでMewを起動してもよいのですが、+inboxバッファなどのMewフォルダが表示されていれば、〈i〉コマンドひとつでメール取り込み状態に移行できます。

画面2　サマリーの画面、読み込まれたメールが表示される

ここでミニバッファに“ Enter password: ”と表示されたら、メールサーバのパスワードを入力しましょう。パスワードを聞かれないようであれば、パスワードを聞かれずにメールを取りに行ってしまうか、パスワードを聞かれる状態にならないかのどちらかで、Mewの設定がおかしいのでしょう。前に戻って~/.emacsなどを確認します。

パスワードの入力後、ネットワークは接続されていてパスワードも正しいのに“ connection failed ”“ Password is wrong! ”などと表示されたら、POP3かIMAPか（POPならばPOPかAPOPかRPOPか）の選択などが違っている可能性があります。これまたimsetupに戻って、

*1：基本的な情報は、info（C-h i、キーバインドが変更時は連載の第2回を参照）またはhttp://www.mew.org/info/index-j.htmlを参照。カスタマイズやトラブルシューティングについてはhttp://www.mew.org/FAQ/index-j.html、メーリングリストについてはhttp://www.mew.org/FAQ/FAQ06/index-j.html#6.1を参照。

キー操作	動　作
フォルダ内の移動	
C-p	前へ
C-n	次へ
⋮	（そのほか、カーソル移動のコマンドが使える）
p	前の（マークなし）メッセージパートへ
n	次の（マークなし）メッセージパートへ
〈SPC〉	メッセージを表示
o*	メッセージを他のフォルダに移動
d*	削除
u	*でつけたマークを消去
x	*でマークをつけたコマンドを実行
y	メッセージをファイルとしてセーブ
A	メッセージを引用して返信（→メール送信へ）
f	フォワードして書き始める（→メール送信へ）

表3　メールを読むのに使用するコマンド

キー操作	動　作
w	あらたにメールを書く　（→メール送信へ）
g	ほかのフォルダを表示する
s	フォルダ内の再スキャン
O	フォルダ内のパック
S	フォルダ内のソート
/	フォルダ内の検索
q	Mewからの脱出
メッセージ表示中のコマンド	
〈SPC〉	次のページ（パートメッセージ）へ
〈DEL〉	前のページへ
C-c C-l	化けているメールを再表示してみる
MIMEマルチパート表示中のコマンド	
C-c C-e	対応するアプリケーションの起動
C-c 〈TAB〉	パートをそのまま見る
y	パートをファイルとしてセーブ

画面3　メッセージを表示させる

画面4　削除（d）とフォルダへの移動（o）をマークする

プロバイダからもらった接続条件を見て設定しなおしましょう。このほかのトラブルについては、Mewのホームページに具体的な事例が豊富ですので、そちらを見てください。*1

パスワードの入力後、新着メールはバッファ+inboxに取り込まれていきます（**画面2**）。+inboxは、さきほどのたとえでは家の郵便受けに相当するところで、取り込んだメールのサマリーが一覧表示されます。

じつは、デフォルトどおりにユーザーのメールボックス（＝郵便受け）とフォルダ（＝状箱）がディレクトリ~/Mail/下に作られたとすると、バッファ+inboxは~/Mail/inbox/ディレクトリに対応し、この中のファイルが一覧表示されるのです。このディレクトリには、メッセージひとつひとつがファイルになったMH形式でメールがため込まれています。これを知っていれば、Emacsを起動していないときでもある程度の閲覧や操作（プリントアウトなど）が可能だし、Mewを使わずにgrepコマンドで高速に検索できて便利かもしれません（なんか連載の主旨と反しますが）。

メールを読む

さて、メールボックスの中身を読むには、**表3**のコマンドを使います。

■メッセージを選ぶ

要するに、通常のバッファと同様、カーソル移動や検索コマンド、画面移動やウィンドウ切り替えなどのEmacsコマンドは全部使えるので、これらを活用して目当てのメッセージにカーソルを移動すれば、効率よく選択できます。

特殊かつ便利なのは、〈n〉や〈p〉でマークしたメッセージを飛び越して前後のメッセージを選択できることです。

■メッセージを表示する

メッセージを選択したうえで〈SPC〉キーを押すと、メッセージが表示されます。メッセージ表示ウィンドウがないときは、自動的にウィンドウが分割されます（**画面3**）。ここでさらに〈SPC〉キーを押すと、メッセージテキストの次の画面が表示され、〈DEL〉キー（キーバインドを変えていれば〈BS〉かもしれません）で前の画面に戻ります。メッセージの終端までくると〈SPC〉は次のメッセージを選択して冒頭から表示ということになるので、すべてのメッセージを流し読みするだけなら〈SPC〉キーの連打でこと足ります。メッセージは、〈y〉でセーブできます。

たまに文字が化けたメールに出くわすことがありますが、これはヘッダの「このメールは××という文字コードを使っているよ」という宣言と内容が一致していないためです。**C-c C-l**でヘッダの宣言を無視すれば、自動的に文字を判別して表示してくれます（それでも化けるようなら、ほんとうにおかしな内容なので、あきらめてください）。

メッセージの整理

このやり方はEmacs内のさまざまなツール（dired……ファイルシステム内を編集するコマンドなど）で用いられているので覚えておくとよいでしょう。メッセージを他のフォルダに移動させるときは〈o〉、削除するときは〈d〉でまずメッセージにマークをつけます（**画面4**）。この操作はすぐには実行されず、次に〈x〉をタイプしたときに一括して実行されます（**画面5**）。したがって、間違ったと思ったときは〈u〉で取り消すことができます。

画面5　マークした操作を実行した後（x）

画面6　日時でソートし、パックした後

■メッセージを削除する

〈d〉による削除はC-uで引数を指定できるので、前後に並んでいる数通のメッセージを消去するときは便利です。

■別のフォルダに分類する

+inboxのメールは、フォルダ（状箱）に分類して整理することができます。フォルダに対応するバッファは、バッファ名が“＋”付きで表示されます。

さて移動ですが、〈o〉とタイプすると移動先のフォルダ名を聞いてきます。指定したフォルダがないときは、確認のあと作られます（フォルダに対応するディレクトリがすぐに作られる）。もちろん〈TAB〉キーや〈SPC〉キーによる補完がききますので、入力の手間は最小限ですむようになっています。

さらにデフォルトで「最も可能性の高い移動先フォルダ」が表示されています。新規メッセージの場合はお勧めのフォルダ名まで作ってくれますが、びっくりしてはいけません。移動先のフォルダは、Mewがメッセージ内の各種情報（送信者など）から推定しているのです。

フォルダバッファは、読み出し専用で編集はできないようにしているので、文字は挿入できません。また試してみればわかりますが、カーソルは左右に動きますが、行のどこにあってもカーソルが位置しているメッセージに対して表示や削除などの操作ができます。

これらの操作を組み合わせれば、数十通のメールに目を通して手動で整理分類しても数分とかかりません（次回では自動整理の方法にも触れます）。筆者は、1日平均400通程度のメールを処理しています。よく、マウスを活用するWindowsのメーラに慣れているユーザーから「そんなにたくさんのメールを処理して時間はなくならないの？」と聞かれます。できればヘビーなメールユーザーには、Mewをお勧めしたいところですが、Emacsの基本操作ができていないと苦痛以外の何ものでもありません。もちろん、みなさんは大丈夫ですよね？

■別のフォルダを表示する

表示されているものとは別のフォルダに移るには、〈g〉とタイプします。フォルダ名を聞かれるので、それに答えれば移動します（フォルダ名に補完がきくのは同様ですから、以降ではいちいち触れないことにします）。移動先のフォルダのサマリーに未表示のメッセージがあれば、そこだけ先にサマリーを作って表示されます。さらに移動先フォルダに未処理のものがあると、処理するかどうか聞いてきます。

ちなみにMewは、毎回すべてのフォルダのサマリーを作るのではなく、変更があったところだけを作り直すので、大量のメッセージがあっても表示はたいへん高速です。筆者は+inboxに10万通近くため込んでしまったことがありますが、それでも毎日の差分のサマリーを作成する時間（数秒）だけで表示が開始されます。え？　どうして10万通もためこむかって？　偉そうなこと（処理が楽）を書いたわりには、毎日未処理のメールが数十〜100通たまってしまうのさ。へっへっへ……。

パック、ソート、検索

筆者の+inboxのように（泣）、未処理のメッセージがあるうえに取り込みを繰り返していると、あとからメッセージを処理したときにメッセージ番号が歯抜けになってしまいます。これが気になる人は、〈O〉でパックしてしまい

画面7　返信を書く（A）

ましょう。

また、フォルダにメッセージが格納される順番は、フォルダ内に移動した順番ですので、たとえメールが時刻順で+inboxに到着しても、別のフォルダに移動してしまえば、時刻順で並ばないこともあります。こんなときは〈S〉でソートすればわかりやすくなります（**画面6**）。

■ソート

〈S〉コマンドは、時刻のほかさまざまな要素で並べ替えることができます（「さまざま」がわからなければ、いきなり補完してみましょう。全部表示されます）。たとえば、特定のヘッダや発信人からのメールをまとめて削除してしまいたいとき（未読のメールニュースがたまりすぎたけど古いので読む気がしないときなど）、SubjectやFromフィールドの内容で並べておいて、まとめてガバっと削除する方法があります。

〈/〉による検索は、ある条件（これまた最初から補完してどんな条件で検索できるかみてみましょう）で検索したメッセージのみのサマリーを表示させるものです。これまで触れませんでしたが、不要なメッセージをサマリーに表示させない、という技を使うことができるのです。したがって（これまた）ある条件のメッセージだけ表示して、まとめて〈d〉マークをつけて削除、などというときには便利でしょう。

キー操作	動　作
C-c C-s	シグネチャを付ける
C-c C-c	書き終わり（メールを送信する）
C-c C-a	MIMEメッセージの準備

表4　メールを書くのに使用するコマンド

■検索

検索が済んだあとなど、表示されていないメッセージも含めて全サマリーを表示したいときは、〈s〉コマンドを使います。〈s〉をタイプした直後に“Range”と聞かれますが、ここでよく答えるのがupdate（デフォルト）とallです（ほかに何が指定できるかは「最初から補完」を活用のこと）。updateは、フォルダを移動した直後に実行されるサマリーのスキャンと同様、前回サマリーを作成した状態からの差分をリストアップします。なんらかの理由で表示が狂ってしまったとき、また検索の用が済んだあとは、allでフォルダの中身を一からスキャンさせます。

パックやソートの直後にも、サマリーを一新する必要からallと同様のスキャンが行われます。したがって筆者の+inboxのように（そんな奴いねーよ？）メッセージをしこたまため込んだフォルダで検索、パック、ソートなどの操作をすると、しばらく往生することになるので注意が必要です。

メールを書く

メーラを使ってメールを書く場合、新規にメールを書き始める場合と、もらったメールに返信する方法の2通りがあります。前者はMewの場合、サマリーの一覧表示の状態からいきなり〈w〉で書き始めることができます。新規の場合はどういうSubject（タイトル）を付けてもかまわないのですが、返信の場合は以下のようなことに注意します。

- **かならず相手のメッセージを引用する**
- **リプライで返す**

Mewでは〈A〉コマンドを使うことで、これらを満たして返信を書き始めることができます（**画面7**）。

引用をするのは、自分のメールに返信を返されて「はい、そうですね」などと書いてあっても、すぐにはわからないからです。オリジナルの文を引用して、それに対する答えを付けるのが相手に対する思いやりです。逆に引用部分が多すぎるのも考えものです。Mewの〈A〉コマンドで引用し、編集しながら（Emacsなので楽々ですね）自分のメッセージを付け加えれば、このようなマナーにかなったメールを柔軟に作成できます。

リプライで返すのは、マナーというより実用上の問題を含んでいます。一般的に返信メッセージには“In-Reply-

To ”というヘッダが付き、機械的にどのメッセージへの返信かわかるようになっています。可能ならこのフィールドを見ていただきたいのですが（コラム**「ヘッダって何？」**参照）、メーラによってはこのフィールドをもとに関連のあるメッセージを一覧表示（スレッド表示）して話題の流れを追いやすくしているものがあります。議論の内容をWebページで読めるようにしているメーリングリストも同様です。このような場合、いかに引用していようと、リプライでなく新規メールで出してしまうと、「スレッドが切れた」といって怒られる可能性大です。

さてMewの使い方というよりは、メールのマナーにちょっと踏み込んでしまいました。本文の書き方などは、新規でも返信でも以下のとおりです（**表4**）。

・シグネチャを付ける

シグネチャ（署名）を~/.signatureというファイルに作っておくと、**C-c C-s**でメールの末尾に付加できます。

・書き終わったら

C-c C-cで送ることができます。これとシグネチャの付加を組み合わせて、メッセージを書き終わったら「コントロールキーを押したまま〈c〉〈s〉〈c〉〈c〉」と覚えておきましょう。

■他人のメールをフォワードする

サマリーが表示されている状態で〈f〉をタイプすると、選択しているメールを別の人にフォワード（転送）できます。元のメッセージが新しいMIMEメッセージの別のパートに入れられ、いきなり本文の編集にかかれます。

■一部だけ引用する

これもサマリー表示での操作です。もとのメッセージの本文の引用範囲をコピーしておいてから〈A〉コマンドをタイプすると、その範囲だけ引用して書き始めることができます。長いメールに対してさわりの部分だけ引用すればよいような場合は便利でしょう。

さて次回は

パーソナルUNIXユーザーのなかには、ほかの仕事は全部UNIXなのに、「メールのやり取りは（デュアルブートで使える）Windowsのほうが便利」という人がいます。ですが、いったんMewの魅力にはまってしまうと、1個のアプリケーションとして孤立しているほかのメーラなど、かったるくて使えなくなっちゃいます。みなさんも早速試して、Emacsで暮らす第一歩を踏み出してください。

では来月もまた、Happy Hacking!

Column

ヘッダって何？

メールヘッダは、MTAがメールを配達するのに使用されます。Mewでメッセージを開くと、ヘッダの主要な情報のはじまりSubject行から表示され、それより前のヘッダはバッファの表示範囲より上に隠れます。その部分を見たいときは、1画面上を見るのに使用する〈BS〉キーを押します。

メッセージを作成するとき、はじめに表示されるのはTo、Subject、X-Mailerくらいです。X-Mailerは使用するメーラを表しますが、“X-”で始まるものは配達には関係のないヘッダです。それ以外にCcやBccなどを加えたければ、Toの行の次あたりに空行を挿入し、“Cc: ”などと書きます。よく使われるフィールドには補完が使えるので、綴りがあやふやな場合には試してください。

最低限必要なヘッダ（自動的に付く）	
From ;	差出人のメールアドレス
To：	受取人のメールアドレス
Subject：	タイトル
Date：	発信された日付・時刻
必要に応じて付け加えるヘッダ	
Cc：	カーボンコピー（同じ内容の控えメール）の受取人メールアドレス
Bcc：	Cc：と同様だが、他の受取人にはここに列挙されている人が見えない
Reply-To：	書いた人と差出人が違う場合、あえて「この人に返してね」と宣言するヘッダ。MLに投稿するメッセージに付加すると、議論を止めることになるので注意
Errors-To：	エラーメッセージの送付先
Return-Receipt-To：	付けておくと、相手のメールアドレスがあるホストのMTAが、受信しましたよ、と受領証を返してくれることがある（期待してはいけない）。メーリングリストに投稿するメッセージに付けると自分が死ぬ
知ってると便利なヘッダ	
Content-Type：	メッセージのMIMEタイプと文字コード
Message-Id：	メッセージのID（固有番号）
X-Mailer：	メーラの種類
In-Reply-To：	返信したメッセージのID（新規メッセージにはつかない）
Received：	受け取ったメールがたどってきた経路を逆探知できる

表4　メールを書くのに使用するコマンド

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2回
【editor】（えでぃた）
【ex】（いー-えっくす）
【vi】（ぶい-あい）
【詰めvi】（つめ-ぶい-あい）
【Emacs】（いーまっくす｜いまっくす）
【Mule】（みゅーる）
【XEmacs】（えっくす-いーまっくす）
【cat】（きゃっと）

editor
【えでぃた】


人間のeditor

（1）編集プログラム。文書＝テキストを編集するものを“テキストエディタ”、画像を編集するものを“グラフィックエディタ”と呼ぶ。単にエディタと言った場合は一般にテキストエディタを指す。英語の「編集者」（editor）に由来する語だが、締め切りを過ぎた原稿をかかえていても、ユーザーに催促したり恨んだり夜道で後ろから襲いかかったりしないところが異なる。ふだんの恩を忘れてこういうことを書いていると、本当に刺される可能性があるので注意。
（2）コンピュータとともに発生した新手の宗教。
（3）触っていると仕事しているように見えるもの。

ex
【いー-えっくす】

文書を行単位でしか編集できない、“ラインエディタ”のひとつ。ふつうのスクリーンエディタを「2次元での操作が可能なもの」だとすると、exは1次元での編集しかできないものだといえる。非常に不便なのでできれば一生使わずに過ごしたいが、テロリストに拉致されて核融合レーザー砲をつきつけられ、ラインプリンタしかない環境でコーディングや執筆作業をしなければ妻子の命はないと脅されても使うのはごめんこうむりたい。ただし、

```
% setenv EDITOR /usr/bin/ex
```

などとして、デフォルトのエディタに指定しておくと、日常作業時にマゾ欲求を満たすことができるという点では非常に重宝するであろう。LinuxなどUNIX互換OSのほとんどにexが標準搭載されているのは、開発者にこういったマゾヒストが多いためだと思われる。

vi
【ぶい-あい】

（1）テキストエディタ。2次元の操作が可能なスクリーンエディタに見えるが、その実体はハードリンクされた1次元世界の住人“ex”であることが多い。exが鏡に自分の姿を映して「ミラースパーク！」と唱えると、正義の味方、viとなって悪のバグと闘う。2次元と1次元のあいだに生まれたハーフであるため、操作性がイマイチという悲壮な宿命を背負っている。がんばれ！　たたかえ！　僕らのvi！　ちなみに、裏番組は「シルバー仮面」だった（編集部注：一部、1970年以降に生まれた方にはわかりにくい記述がありましたことをお詫びいたします）。
（2）Linux教に古代から伝わる神器。これを使いこなせるようになることは、Linuxシステム管理者にとって割礼にもひとしい重要な通過儀礼のひとつである。システム設定やユーザー情報を司るファイルは神聖であるため、この神器以外で触れることはまかりならないとされている。かつて不遜にも/etc/passwdをEmacsで編集しようとしたアダムは、罰と

してエデンの園を追放されたという（旧約聖書より）。たぶん、shadow化されていなければOKだったんじゃないか。
――ユーザーを見分ける方法：1. Escキーに画鋲をしかけておくと悲鳴をあげる、2. rogueがうまい（単に仕事をしていないだけの人間と間違える可能性があるので注意）、3. 血が赤い（必要条件だが十分条件ではない）

詰めvi
【つめ-ぶい-あい】

難解だが使いこなせば絶大な力を発揮するviのコマンド。その応用の腕を競うために考え出されたテストを詰めviと呼ぶ。受験者は、出された問題に対して最適な操作法を答えなければならない。熱心なvi士を集めての全国大会も開かれており、本因坊戦、名人戦、王座戦など7大タイトルが争われる。最近、詰めvi大会参加者の高年齢化が懸念されている。今後のviの発展と詰めviのいっそうの普及のため、編者が個人的にファンである梅澤由香里3段のようなプロ女流vi士の登場を期待したい。

- **詰めvi例題**
  viを起動せよ
- **回答例**

```
$ alias microsoft_word=vi
$ microsoft_word
```

- **詰めvi例題**
  テキストファイル中の「emacs」を「vi」に置換せよ
- **回答例**
  samba経由で該当ファイルにMicrosoft Wordを使ってアクセス。メニューの［編集］－［置換］から一括置換を行う

※日本vi院院生試験問題集より。回答例は一部bashの場合のみ

Emacs
【いーまっくす｜いまっくす】

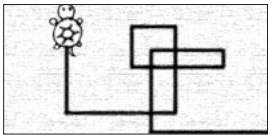
(1)「GNU Emacs」（ぐにゅー・いーまっくす）。Lispインタプリタの一種。エディタ機能がついているため、本来の用途を誤解しているユーザーが多い。なげかわしいことだ。なかんずく、メールの読み書き、Webアクセスやファイル管理に用いるなど言語同断である。なお、Lispとは、編者のまわりに使える人間が誰もいないので詳しくはわからないが、主にカッコの組み合わせだけで命令語を表せる、驚異のAI機能をもったコンピュータ言語らしい。
(2) ときどき、これで/etc/passwdを直接編集したくなる。
(3) サーバのディスクメンテナンスも、できればdiredでやってみたい。
――ユーザーを見分ける方法：1. Ctrlキーに画鋲をしかけておくと悲鳴をあげる、2. 世話好きな女性にあこがれている、3. 血が赤い（十分条件だが必要条件ではない）

Mule
【みゅーる】

世界中の言語に対応したLispインタプリタ。

XEmacs
【えっくす-いーまっくす】

絵が描けるようになったLispインタプリタ。LOGOみたいなものだと思う。


XEmacsの画面（想像）

cat
【きゃっと】

Linux標準装備の軽快なエディタ。その動作の軽さは、他の追随を許さない。起動は、

```
% cat > filename
```

とするだけのシンプル設計。それでいて、

```
% cat -n > filename
```

とすることで利用できる“自動行ナンバリング”など、多彩な機能を備えている。また、保存しておく価値のない駄文、プログラムを自動的に捨てるという高度な動作モードも搭載。コマンドラインで

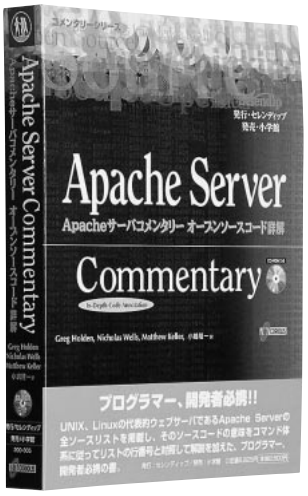
```
% cat > /dev/null
```

とすれば利用できる。現在ではマルチメディアへの対応が検討されており、将来的には音楽、グラフィックなども編集できるようになる予定。これらの拡張がほどこされてもユーザーインターフェイスは基本に忠実で、

```
% cat > /dev/sndpcm
```

のようなかたちで対応できる（2000年4月段階でのsnapshotドライバの場合）。

Books



Apacheサーバコメンタリー　オープンソース詳解

Greg Holden／Nicholas Wells／Mathew Keller 著　小嶋隆一 訳
セレンディップ／小学館
A4変形判／672ページ／CD-ROM付き　　本体価格 8500円

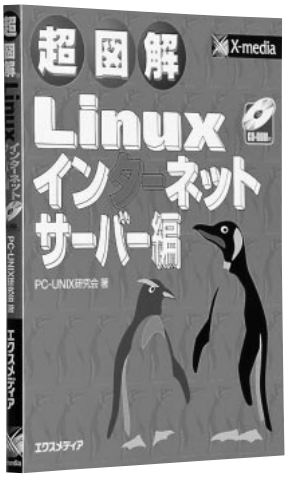
今やApacheは、コマーシャルベースの製品を含めてナンバーワンWebサーバの地位を確立した感がある。Apache Software Foundationを中心とした開発体制は、オープンソースのモデルとしてLinuxよりも優れているという評価もある。なにより、接続しているユーザーを考えれば世界で最も利用されているソフトウェアであるかもしれないのだ。

本書に掲載されているバージョンのApacheのソースコードは、4万1450ステップ、ページ数にして437ページ（2段組）である。すべてではないが、タイトルどおりソースコードに対する詳細なコメントが記述されおり、まさにApacheの全貌を明かしている。モジュールの構造やアルゴリズムなど、Apacheの開発者だけでなく、ソフトウェアの開発に関わる者であれば興味を引かれるところだろう。まったく関係のない話だが、一説には2000万ステップともいわれるWindows 2000であったら……、と考えるとちょと見てみたい気もする。

遺伝的アルゴリズムと遺伝的プログラミング

平野廣美 著
パーソナルメディア
A5判／384ページ／CD-ROM付き　　本体価格 3800円

近頃ヒトゲノムという言葉が巷で騒がしい。本書はタイトルに「遺伝子」という言葉を含んではいるが、直接の関係はない。遺伝的アルゴリズムとは、問題の最適解を求める手法のひとつ。①適当な配列を持つ解の集団を用意する。②あらかじめ決めておいたルールのもとで、その解集団を評価する。③評価した結果、優れていると判断したものを複製する。④複製した新しい解とそれまでの古い解を交差させる。⑤交差させた解の一部の配列を突然変異（ビット配列の反転など）させる。というダーウィン進化論を応用したシミュレーションだ。前半部分でC++の基本的な部分を解説し、後半部分でCD-ROM収録のC++ソースをもとにして「最適な時間割作成」など、遺伝的アルゴリズムを用いた問題解決に迫っている。本書はWindows環境を想定しているが、Cygwin環境で動作可能なプログラムもあるので、腕に覚えのある読者はLinux上でも試してみてほしい。

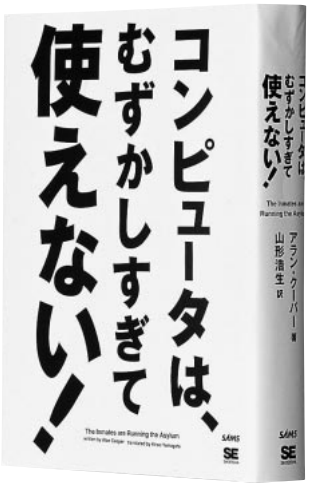


超図解Linux インターネットサーバー編

PC-UNIX研究会 著
エクスメディア
B5変形判／222ページ／CD-ROM付き　　本体価格 1880円

2、3年ほど前からだろうか、オールカラーで図版を多く掲載した初心者向けのユーザーガイドが低価格で販売されるようになった。それらの多くは、200ページほどで、いわゆるユーザーアプリケーションを対象とした入門書的な体裁であった。OS（つまりはWindowsなのだが）を対象としたものでも、基本操作を解説する内容にとどまっていた。

そしてついにLinux本である。Linuxもユーザー層が広がっているのだから、優れた入門書が出ることに異議はない。むしろ歓迎である。ただ、Linuxのインストールにはじまって、ファイル操作などのウィンドウマネージャの使い方が続く、その同じ本の中にApacheやBINDの設定が載っているのはどうだろうか。やはり無理があるのではないかと思う。常時接続でWebサーバ構築を目指す読者を対象としているようだが、あくまで入門書として「知識を身につけるため」に読むべきだろう。



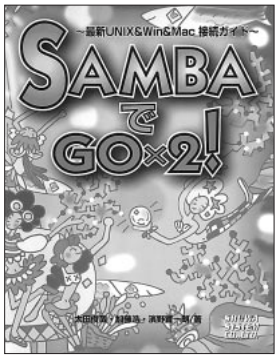
コンピュータは、むずかしすぎて使えない！

アラン・クーパー 著　山形浩生 訳
翔泳社
A5判／480ページ　　本体価格2200円

刺激的なタイトルだが、中身はいたってまじめなソフトウェアデザインの解説書（考察といったほうが的確かもしれない）。ユーザーインターフェイスを軸に、ソフトウェアに関連する人々の意識のズレを解き明かしていく。「人々」という中には、プログラマー、デザイナー、開発責任者、そして忘れられがちな「ユーザー」が含まれる。意識のズレから生まれる悲しいソフトウェアの例は枚挙にいとまがない。肥大化するアプリケーションとOS。充実した機能も使いこなせなければ、ないも同然だろう。コンピュータ（ソフトウェア）とユーザーとの乖離は進んでいくばかりである。

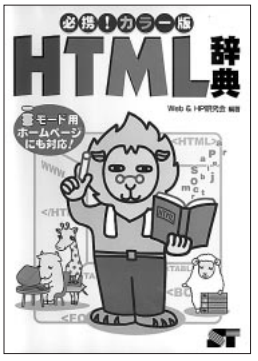
マウス操作を中心としたグラフィカルなインターフェイスは、Mac OSの登場以来、少しも進展していないような気がする。そんな疑問を抱いたことのある人にお勧めの1冊だ。もちろん、ユーザー不在のソフトウェア開発を進めてきた人は必読である。

SAMBAでGO×2！



太田俊哉／加藤浩／浜野賢一郎 著
秀和システム
B5変形判／224ページ
本体価格 2600円

HTML辞典



Web & HP研究会 編著
ソーテック社
A5判／296ページ
本体価格 1380円

[実践] Linux徹底活用ガイド

塚越雄一 著
技術評論社
A5判／294ページ
本体価格 2280円

UNIXの便利ワザ ファイル小技編／ネットワーク裏技編

Kirk Waingrow 著
金田芳明 訳
アスキー
A5判／240&220ページ
本体価格 2000円

読者の声

俺にもいわせろ！

ゴールデンウィークはいかがでしたか？担当も久しぶりにたっぷりと遊び倒しました（現実になりますように）。

5月号特集1へのおハガキ

「今日から使えるLinux」大変参考になりました。なにせLinux歴1日なもんですから……。Linuxを使おう！と思ったのも、この本を見てからでした。実際、使ってみると初心者には難しいものでした。ハードディスクの容量はなくなっちゃうし（泣）。早く使いこなせる日が来てほしいです。

（北海道　堀合純平さん）

え？　Linuxって1種類じゃないのね、っていうレベルの初心者が5月号を初めて買ってしまいました。見よう見まねでTurboLinux Workstation 6.0をインストールしてみましたが、解像度が何か変。さっそく、Win98からTurboLinux系のサイトを巡回してみるが、用語がわからずにチンプンカンプン。先は万里の長城より長そうですが、これから頑張ろうと思いますので、よろしくお願いします。

（大阪府　CTBTさん）

どんなことでも、最初は初心者からのスタートです。Linuxは焦らずのんびり、長く使うとよいOSなのかもしれません。あまり気合いを入れすぎると、だんだん辛くなってしまいますものね。末永くおつきあいくださいますよう、よろしくお願いいたします。

5月号特集2へのおハガキ

「覚えておきたい必須コマンド30」は役に立ちました。Linux導入から半年以上たちますが、いまだにこのようなコマンドに慣れていませんでした。このように、基礎からやってくれる記事はありがたいです。次はファイルシステムでもお願いします。

（東京都　青木洋児さん）

「覚えておきたい必須コマンド30」はとてもよかったです。このような、保存版お役立ち情報も、よろしくお願いします。きれいに切り取れるような感じになっていると良いのではとも思います。

（東京都　加藤教幸さん）

UNIX由来のコマンドは、ひとつひとつは高機能ではないのですが、組み合わせ方によっては極めて複雑なことができます。こんなコマンド、いったい何に使うんだというものも、熟練コマンド使いの手にかかれば、アラ不思議、まるで専用ツールを使っているみたい。そんなコマンドの使い手を目指して担当も勉強しているのですが……トホホ。シェルでポコポコと遊ぶのがやっとです。いまのところGUIとうまく使い分けるのがちょうどいいのかもしれません。

付録CD-ROMへのおハガキ

OpenLinux eServer 2.3をネットで見つけ、ダウンロードしようとしたが、重くて4日がかりでも全部落とせず、あきらめかけたところで5月号を発見！　MMX Pentium 166MHzに96Mバイトのメモリでもそこそこ動く。メインのサーバに使ってます。
P.S.　家のマシンはどれもTurboLinuxと相性が悪く、まともにインストールできたことがありません。

（秋田県　虻川章一さん）

付録CD-ROMからTurboLinuxを入れて、Windowsと併存させようと思って5月号を購入したのに……。CD-ROMドライブがお亡くなりになってしまったーー!!　FTPダウンロードする根性がないので、この計画はちょっと延期です。しくしく。

（東京都　鈴木結花さん）

OpenLinux eServerは、Webブラウザで遠隔管理ができたり、画面表示に特化してデザインされたフォント「さいもん」を搭載していたり、いま注目のサーバ製品ですね。

CD-ROMドライブ、ご愁傷様でした＞鈴木さん。いっそのこと、DVD-ROMドライブを購入してしまうのはどうでしょう。

TurboLinux Workstation 6.0のLILO

TurboLinux Workstation 6.0は、LILOの書き込みエラーが発生し、ハードディスクにインストールできませんでした。

（東京都　藤崎正和さん）

ご迷惑をおかけしました。WindowsなどのFATパーティションが複数あると、インストーラの「起動可能パーティション」画面で、それらのパーティションに「dos」という起動ラベルが重複して設定されてしまい、LILOのインストールエラーが発生します。重複しているラベルのうち、Windowsの起動パーティション以外はラベルを消去してください。

また、TurboLinuxをインストールするパーティションがハードディスクの8Gバイトを超える部分を含んでいる場合は、2つ目の「LILO設定」画面で、［Linearモード］と、［LBA32モード］にチェックを入れてください。BIOSが8Gバイト以上のディスクに対応していれば、これでインストールできるはずです。

自分だけのLinuxマシンを

Linux magazineのいいところは、ちょっとハード寄りというか、自作系の話がたまにあることです。最近、パーツ系とかの雑誌を読まなくなったので。

（千葉県　鈴木大輔さん）

今月の「Linuxマシンを作る！」はいかがでしたでしょうか。自作特集チームと、イーサネット特集チームが同時に秋葉原へ繰り出し、あっちへウロウロ、こっちへフラフラした結果、今月号が完成しました。電気街へ行くと、各編集部員の妙なコダワリが露見して面白いものです。なかなか帰ってこないのはみんな同じですが。

炸裂新連載！Linuxガベコレ

「Linuxガベコレ」に妙にハマってしまいました。

（岡山県　花田善仁さん）

「Linuxガベコレ」、良いですね。こういうのを待ってました。

また、特集2の「必須コマンド30」も連載にしてくれるとうれしいですね。

（大阪府　伊藤真和さん）

ありがとうございます。担当も特にお気に入りのコーナーです。著者のしのはらさんはとてもシャイなかたなので、きっと照れまくると思います。これからも応援をお願いします。

コマンドの連載もいいですね。検討させていただきます。

祝！　ご生誕！

3月2日に子供（第1子）が誕生しました。名前を梨乃（りの）と付けましたが、Linuxとは関係ありません？　のであしからず……。

（長野県　大澤 斉さん）

おめでとうございます。梨乃ちゃん、とてもかわいらしい名前ですね。Linuxのように、みんなから愛されて育つこと間違いなしです。

好きなだけって……

プレゼントが当たったら、好きなだけLinuxをインストールするつもり。

（滋賀県　西脇達也さん）

読者プレゼントは、抽選担当の編集者が毎月こわ〜い顔をして選んでいます。読者の声担当としては、おハガキを紹介した方を当選させたいところなのですが、抽選担当が怖ろしくて裏工作できずにいます。あ、でも読者の声に掲載されたおハガキも抽選の対象になっていますのでご安心を。

今月の遠藤さん

毎度!!　山形の遠藤です。先日、ハードディスクの交換をしました。というのも、コンパイル中にハードディスクがぶっ飛んだからです。もともと容量が少なかったため、いい機会だと思いました。大容量のものに交換したおかげで、とても快適です。しかし、1つだけ快適でないものを挙げるとすると、お財布の中身です。大容量のハードディスクのように、お金も大容量に貯まりませんかねェ（ブツブツ…）。おかげで飲み代が1回分なくなりました（ブツブツ…）。そんでもって、家のカミさんにわけもなく怒られました（ブツブツ…）。そんでもって、今まで蓄えてきたデータもきれいになくなりました。ああ無念!!　(^^;)

（山形県　遠藤浩二さん）

今月のトリは、常連の遠藤さんのおハガキです（遠藤さんのコーナー新設か？）。えーと、この場合、お財布＝大容量ハードディスク、お金＝データという関係が成り立つのではないでしょうか。とすると、データも消えてしまったワケですから……。

それにしても、ディスククラッシュは恐いですね。先日、ハードディスクのコネクタ破壊という事故を目撃しました。これもディスククラッシュ……なんでしょうか。

では、またお会いいたしましょう。おハガキをお待ちしております。