MS Officeと互換性のある無料のOfficeツール登場！

米ThinkFree.comから、Microsoft Officeと互換性を持ったWebベースのオフィススイート「ThinkFree Office」（以下TF Office）がリリースされた。Write（ワープロ）、Calc（表計算）、Show（プレゼンテーション）、Mail（電子メール）の機能を持っていて、それぞれMSの、Word、Excel、PowerPoint、Outlook Expressとルック＆フィールを合わせたものになっている。TF OfficeはJavaでプログラムされていて、現在のところWindows版以外のUNIX版、Linux版、MacOS版は開発中である。

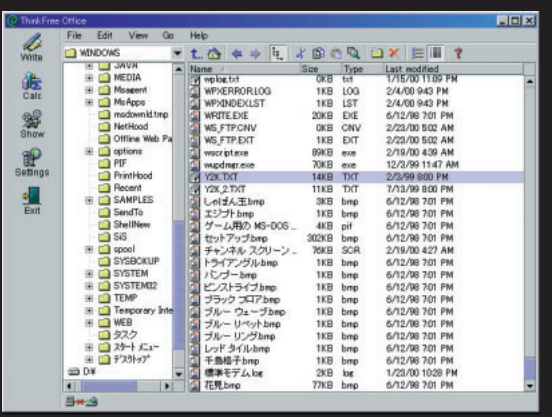
TF Officeを使うには、同社のWebサイトにアクセスする（http://www.thinkfree.ne.jp/）。あらかじめ無料で入手したIDとパスワードを入力してログインすると、プログラム自体をWebからダウンロードして実行するため、インストールという作業は必要ない。

まずエクスプローラのようなファイル管理プログラムがロードされる。文書ファイルをダブルクリックするか、左側のメニューからアプリケーションを選ぶかすると、また必要なプログラムをダウンロードしてくる仕組みだ。

最初にIDとパスワードを登録すると20Mバイト分のレンタルスペースがもらえる。データファイルもローカルのPCではなく、インターネット上に預けておくことができるのである。

ところで、ThinkFree.comは無料で配布するソフトウェアから、どうやって利益を得ようとしているのだろうか。TF Officeを使う人は、必ずThinkFree.comのWebサイトを訪れなければならない。ということはYahoo!のようなポータルサイトとして成立する。とりあえずはユーザーを増やすことが肝心だ。

この仕組みを利用すれば、もう個人用のソフトは不要になるかもしれない。



Hardware　発売日　2000年2月21日

Linux専用3Dグラフィックスカード「E&S Lightning 1200LX」

URL　http://www.tgi.co.jp/

発売　テクノグラフィー株式会社
TEL　03-5468-2414
価格　9万9000円

テクノグラフィーは、米Evans & Sutherland社製のLinux専用Open GLグラフィックスアクセラレータカード、「E&S Lightning 1200LX」（以下1200LX）の発売を開始した。1200LXは、E&S社初のLinux対応製品で、Linux上でWindows NTやほかのUNIXと同等のOpen GL環境が得られるとしている。対応ディストリビューションは、Red Hat Linux 6.1（日本語版）。

アクセラレータチップに、E&S社のREALimage 1200を採用し、330万ポリゴン／秒、330万ベクトル／秒の3D描画性能がある。また、15Mバイトの3DRAM（フレーム&ローカルバッファ用）と16MバイトのCDRAM（テクスチャ用）のメモリを搭載し、70万ピクセル／秒でテクスチャマッピングが行える。

動作確認されているソフトは、Side Effect softwareの「Houdini 4.0」、Engineering animation Incの「World ToolKit 9.0」で、今後各社の3D系アプリケーションへの対応を行っていくという。

Software　発売日　2000年4月7日

マルチOSブートローダ「システムコマンダー2000」

URL　http://www.softboat.co.jp/

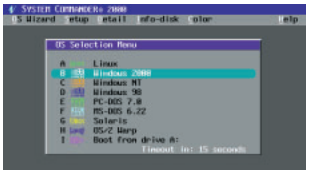
発売　株式会社ソフトボート
TEL　03-3256-4712
価格　1万4800円

ソフトボートは、米V Communications社が開発したマルチOSブートローダ「システムコマンダー2000」（以下SC2000）を4月7日から販売する。価格は、1万4800円だが、初回ロットの2万本はキャンペーン価格で1万2800円。

OSウィザード機能により、PCに新しいOSをインストールするために必要な作業をすべて自動化できる。また以前のバージョンではFATのみに対応していたパーティションサイズの変更が、NTFS、Linux ext2、Linuxスワップでも可能になり、パーティション操作の自由度が増した。

BackStepウィザードにより、SC2000で行ったパーティション操作を元に戻すことができるようになった。

LinuxのLILOのインストールでは、MBRとSuperblockの両方に対応する。



Hardware　発売日　2000年3月1日

Linux搭載のインターネットPC「Web COM」

URL　http://www.logicaleffect.com/

発売　ロジカルイフェクト株式会社
TEL　03-5822-3322
価格　4万9700円〜

ロジカルイフェクトは、Linuxを用いたインターネット専用PC、「Web COM」の販売を3月1日から開始した。価格は4万9700円から。

チップセットには、ビデオ機能を統合したSiS620を用いている。また、ネットワークカードを標準で搭載しており、家庭内の2台め以降のPCとして、またはLAN環境のあるオフィスでの、メールおよびWebブラウズ用途を想定している。

OSには、Vine Linux 1.1を採用している。標準ではCD-ROMドライブを搭載していないため、購入後に操作ミスなどが原因で、起動できなくなるといったようなトラブルには、有償の再インストールで対応する。

CPUにCeleron 466MHz、64Mバイトのメモリ、4.3Gバイトのハードディスクを搭載したモデルが4万9700円。

Software　発売日　2000年3月30日

マルチOSブートローダ「VirtualPartition」

URL　http://www.mvi.co.jp/

発売　株式会社メディアビジョン
TEL
価格　1万800円

メディアビジョンは、マルチOSブートローダ「VirtualPartition」を3月30日から発売する。価格は、1万800円。

独自開発のBootWare機能により、1台のPCに複数の異なるOSをインストールし、共存させることができる。また、1つのパーティションに複数のOSをインストールすることも可能。

IDE、SCSI、USB接続のハードディスクに対応している。30Gバイト以上のハードディスクでも利用できる。

ハードディスクのデータを保持したまま、パーティションの作成、削除、変更、移動の処理を行える。またFAT、NTFS4.0、Linuxスワップ、Linux ext2などのファイルシステムに対応し、FAT16／FAT32間や、FAT／NTFS間などのファイルシステム変換もサポートしている。

Software 発売日 2000年2月20日

テキストデータ抽出プログラム「デ変研DocCat」

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

発売 有限会社データ変換研究所
TEL 075-958-1114
価格 4600円（1ユーザーライセンス）

データ変換研究所は、MS Wordなど、Windows用のアプリケーションの文書ファイルからテキストデータを抽出するプログラム「デ変研DocCat」の販売を2月20日から開始した。

Linux、FreeBSD、NetBSD版があり、価格はいずれも4600円。またサーバライセンスは、16万8000円。Linux版は、TurboLinux 4.2、LASER5 Linux 6.0、Red Hat Linux 5.2、Vine Linux 1.1、Debian GNU/Linux、Caldera OpenLinux 2.3、Slackware 3.5、Plamo Linux 1.4、Kondara MNU/Linuxと多くのディストリビューションに対応している。

テキストの抽出が可能なWindows用のアプリケーションは、Word（95～2000）、Excel（95～2000）、PowerPoint（97～2000）、一太郎（9、10）、OASYS（6、7）、Lotus Word Pro。出力コードは、EUC、SJIS、JIS、UTF-8、UCS-2から選択できる（デフォルトはEUC）。

データ変換研究所では、同社の製品「デ変研TEXT」の購入者には、「デ変研DocCat」のライセンスを無料で提供するとしている。

Hardware 発売日 2000年1月31日

Linuxをバンドルしたサーバ「GRANPOWER5000 オールインワンタイプLinuxサービスバンドルモデル」

URL http://www.fujitsu.co.jp/hypertext/granpower/gp5/linuxtype/linuxtype.html

発売 富士通株式会社
TEL 03-3216-7976
価格 36万円～

富士通は、「GRANPOWER5000オールインワンタイプ Linuxサービスバンドルモデル」（以下GP5000）を1月31日に発売した。出荷は3月上旬を予定している。価格は、Celeron 466MHz、128MバイトのECC機能付きメモリ（最大768Mバイト）、13GバイトのUltraDMA/33対応ハードディスク（最大72.8Gバイト）を搭載したモデルが36万円。同じ構成で、Pentium III 700MHzを採用したモデルは58万円。

GP5000は、同社のエントリー向けPCサーバ「GRANPOWER5000 ES200」に、米Caldera Systems社のLinuxディストリビューション「OpenLinux eServer 2.3 日本語版」をバンドルし、1年間有効のサポートサービス（20件まで）とインストール代行サービスを加えたもの。サポート範囲は、インストール時のトラブルシューティングから、RAID構築に関する質問、実運用時のトラブルに関する質問などまでを含む。

Software 発売日 2000年3月2日

リレーショナルデータベース管理システム「InterBase 5.6」

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

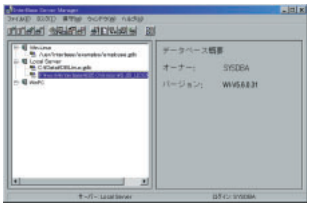
発売 インプライズ株式会社
TEL 05-5350-9380
価格 4万9800円

インプライズは、リレーショナルデータベース管理システム（RDBMS）のLinux対応版「InterBase 5.6 for Linux」を3月2日に発売した。対応するディストリビューションは、「Red Hat Linux 6.1 日本語版」と「LASER5 Linux 6.0」。なお同時に、Windows対応版「InterBase 5.6 for Windows 95 and Windows NT」も発売された。

価格は、どちらも4万9800円で、1サーバライセンスと1クライアントライセンス（4セッションまで接続可能）が付いている。

今回のバージョンは、Linux版とWindows版の間で、データベースの互換性を実現し、ファイルを転送するだけで、両プラットフォームでの相互運用を実現できるのが特徴だ。

またJava 2対応のJDBCドライバ「InterClient 1.51」を装備し、Java 2対応の開発ツール「JBuilder」などとの親和性を向上させている。



Hardware 発売日 2000年1月28日

ネットワークストレージアプライアンス「MaxAttach」

URL http://www.tradepia.or.jp/nesol/

発売 日商エレクトロニクス株式会社
TEL 03-3544-8235
価格 18万9000円～

日商エレクトロニクスは、米Maxtor社製のネットワークストレージアプライアンス「MaxAttach」の出荷を1月28日から開始した。ストレージアプライアンスとは、ファイルサーバ機能に特化したPCのこと。

価格は、記憶容量18Gバイトのモデルが18万9000円、36Gバイトが29万円、72Gバイトが44万5000円。

LANに接続し、管理用PCからセットアップすることで、10分以内にファイルサーバを構築することが可能としている。ユーザーライセンスは無制限で、接続数に制限はない。

同社は全モデル合計で初年度1万台の出荷を見込んでいる。

ABIT、同社製マザーボードのための「Gentus Linux」を発表

2000年2月22日

ABIT Computerは2月21日、同社のマザーボードに特徴的な機能 ――特に、ATA/66コントローラのサポート―― を強化したディストリビューション「Gentus」を発表した。

Red Hat Linuxをベースにパッチをあて、ABITマザーボード向けのツールを追加したもののようだ。

最大の利点は、HighPoint TechnologiesのATA/66コントローラのサポートだ。ABITのマザーボードにはこのコントローラが実装されており、最大8台までのIDEドライブを接続することができる。しかし安定版のLinuxカーネルは、これをサポートするパッチがあたっていないために、ATA/66ドライブにLinuxをインストールするためには、ATA/33モードでドライブをつながなければ、インストールできなかった。Gentusにはすでにパッチがあたっていて、直接ATA/66にインストールできるという。

ABIT Computer
(http://www.abit.com.tw/)

米Eazel、「次世代のユーザーインターフェイス」をLinuxに

2000年2月22日

米Appleと米America Onlineの元在籍者が、使いやすいLinuxを提供するために、新たな会社を設立した。

新会社名は「Eazel」。CEOは、Mike Boich。彼は、米Appleの元在籍者で、初期Macintoshのプロジェクトに関わっていた。同社に彼のほかにも、初代Macintoshの開発メンバーで、システムソフトウェアの多くを開発したAndy Hertzfeldや、米America OnlineのSenior Vice Presidentだった、Mike Homerなどが在籍している。

同社では、Linuxに「次世代のユーザーインターフェイス」をもたらすとしており、現在、GNOMEの環境をベースに、「Nautilus」と呼ばれるファイルマネージャを開発している。現在同社のWebページからは、あまり情報は得られないが、「Nautilus」 の開発更新情報を見ることが可能。

また、同社が開発したソフトウェアのライセンスを、GPLとすることを表明している。

米Eazel（http://www.eazel.com/）

Red HatがItanium用GNUProコンパイラをリリース

2000年2月17日

米Red Hatは2月16日、IA-64用のコンパイラを含む開発ツール「GNUPro tools for IA-64」をリリースした。

GNUProは、gccなどのGNUツールを拡張した製品で、先日Red Hatに買収された米Cygnus Solutionsが開発を行っている。

コンパイラとは、人間が読める形式のソースコードを、CPUが実行できる形式に翻訳するソフトウェア。今年後半に登場する「Itanium」から始まるCPUアーキテクチャ「IA-64」では、コンパイラが作成するコードの品質が性能を非常に大きく左右するという。

IA-64に対するLinuxカーネルの移植も着実に進んでおり、Linux関連企業が参加している Trillian Projectは 2月 2日 、Linux/IA-64のソースコードをリリースしている。

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

Inprise、「InterBase 6」のサポートを行う新会社を設立

2000年2月17日

米Inpriseは、2月14日、オープンソースとなる「InterBase 6」をサポートする新会社「InterBase」の設立を発表した。

新会社の設立は、同社のリレーショナルデータベース管理システム（RDBMS）の新バージョンである「InterBase 6」をオープンソースとする、1月3日の発表を受けたもの。

InterBaseの経営陣として、社長にAnn Harrison氏、セールス・マーケティング担当副社長にPaul Beach氏が就任、「InterBase」の初代アーキテクトのJim Starkey氏がソフトウェア・アーキテクチャの技術アドバイザーとなる。

「InterBase 6」のソースコードは「MOZILLA PUBLIC LICENSE Version 1.1（以下、MPL 1.1）」に準拠した形で2000年半ば頃にオープンソース化される予定。

「MPL 1.1」は、オープンソースのWebブラウザ「Mozilla」で採用されたオープンソースライセンス。

米Inprise（http://www.inprise.com/）

米Scriptics、「Tcl/Tk 8.3」をリリース

2000年2月15日

米Scripticsは、「Tcl/Tk 8.3」をリリースした。

「Tcl（The Command Language）」は、オープンソースのスクリプト言語。開発当初は、各種ソフトウェアに搭載されているマクロ言語を統一することを目的としていたが、現在では、TclとTkの組み合わせによる簡易GUIプログラミング環境として利用されることが多い。たとえば、XFree86の設定ツールである「XF86Setup」は、「Tcl/Tk」を利用している。「Tcl/Tk」は、Ver. 8.0から国際化対応がなされているが、日本語の取り扱いに問題があり、日本語に対応したLinuxディストリビューションの多くは、8.0.xに日本語パッチを適用したものを使用している。

「Tk（Tool Kit）」は、Tclから利用されるGUIを構築するためのライブラリとなるツールキット。このウィンドウ環境に「Perl」や「Ruby」などとのインターフェイスを実装した「Perl/Tk」、「Ruby/Tk」なども開発されている。

米Scripticsは、「Tcl/Tk」開発者であるJohn Ousterhout博士が、「Tcl/Tk」をサポートするために設立されたソフトウェアベンダー。

Tcl/TKは、さまざまなプラットフォームに対応しており、

Windows 95／98／NT 4.0
Solaris 2.5／2.6／7
HP-UX 10.20
Linux（Red Hat Linux 5.0以上／SuSE Linux 6.0以上）
SGI IRIX 6.3以上

などをサポートしている。

「Tcl/Tk 8.2」からの主な変更点は、・X Window System上でのホイールマウスのサポート
・コンパイルされる環境の自動認識の強化
・グラフィック描画のスピードアップ
・検索機能の強化
・バグフィックス

などとなっている。
米Scriptics
(http://www.scriptics.com/)

レーザーファイブとドットネットが、Linux教育分野で提携
2000年2月10日

レーザーファイブとドットネットは、Linux教育分野で提携した。この提携により、ドットネットが東京・お茶の水で4月21日に開校する、ネットワーク技術者養成スクール「.net（ドットネット）」において、Linuxコースを開講する。同スクールでは、レーザーファイブの「LASER5 Linux」を教材として採用する。カリキュラムは両社で検討し、セミナーなどにはレーザーファイブからも社員を講師として派遣する予定だという。

Linuxコースには、3カ月間の入門者向けコース「Linux入門コース」や、6カ月間の管理者向けコース「ネットワーク基礎管理コース」などが用意されている。

このほか、ネットワーク技術者を養成するための総合的なコース「ネットワーク総合本科コース（1年間）」などが用意される。ネットワーク総合本科コースには、インターネットテレビ電話でリアルタイムに授業を行う「通信科」もある。通信科は5月開講。

入学金と授業料を合わせた費用は、Linux入門コースの25万円から。

ニュースリリース
(http://www.laser5.co.jp/press/000210.txt)
.net（ドットネット）
(http://www.kabusikigaisha.net/)

米Microsoft、UNIX互換環境「Microsoft Interix 2.2」をリリース
2000年2月8日

米Microsoftは、「Microsoft Interix 2.2」をリリースした。

「Microsoft Interix 2.2」は、Windows上のUNIX互換環境で、同製品の特徴としては、

・IEEE準拠POSIXサブシステム
・Windowsデスクトップに統合
・300以上のユーティリティ（Shell環境やX Window SystemのRuntimeを含む）
・バッチ処理可能なリモート管理ツール
・ソースコードの書き換えなしでUNIXアプリケーションをコンパイル可能
・UNIXのAPIとWin32APIを同時に使用可能

などが挙げられ、WindowsとUNIXの相互運用性を向上させられる。

価格は99.95ドル。以前のバージョンの登録ユーザーは、3月1日より、以下のダウンロードサイトから無料でアップデート可能。

「Interix」は、Microsoftが1999年9月に買収した米Softway Systemsの製品で「Interix」となる前は「OpenNT」と呼ばれていた。

また、Microsoftは以前、「Windows Services for UNIX」によってPOSIXサブシステムを提供していたが、米Softway Systems買収により「Microsoft Interix」に置き換えられた。類似の製品として、現在まだベータ版ではあるが、米Red Hatに買収された米Cygnusの「Cygwin」がある。

Microsoft（http://www.microsoft.com/）
ダウンロードサイト
(http://www.microsoft.com/windows/sfu/)

カナダCorelと米Inpriseが合併
2000年2月8日

WindowsやLinux用の開発ツールをリリースしている米Inpriseと、Corel LINUX OSやWordPerfect Office 2000などで知られるカナダのCorelが、合併することで最終合意した。社名は「Corel」となる。Linux、Windowsそのほかのプラットフォーム上のアプリケーションやサービスを提供していくという。

CorelのCEOであるMichael Cowpland氏がCEOとなり、InpriseのCEOであるDale Fuller氏が取締役会長となる。取締役会には、Corel側から4人、Inprise側から2人が参加する。

これで、コンシューマーからエンタープライズまで、そしてWindowsからLinuxまで、すべてをカバーする企業が出現したことになる。同社の報道資料を見る限り、今回の合併でもっとも重要な位置を占めているのがLinuxだ。同社では、「Linux Powerhouse」として活躍していくという。

Corelは将来的に、インターネットアクセスのためのインフラや、オフィス製品、エンタープライズのための開発者向けソリューションを提供し、WindowsからLinuxへの移行を促進する先頭に立つかまえだ。

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

Linux/IA-64（Itanium）のパッチが公開

2000年2月7日

LinuxをIntelの64ビットプロセッサアーキテクチャ「IA-64」に移植するTrillianプロジェクトは2月2日、Linux/IA-64のソースコードを初公開した。パッチのサイズは、展開後で約1.6Mバイト。

Trillianプロジェクトには VA Linux Systems、 Red Hat、 SuSE、 Caldera Systems、TurboLinuxなどのLinux企業および CERN、 Hewlett Packard、 IBM、Intel、SGIなどが参加している。

公開されたソースコード
(http://www.kernel.org/pub/linux/kernel/ports/ia64/)

Corel、「Corel WordPerfect Office 2000 for Linux」を正式発表

2000年2月4日

米New Yorkで開催中の「LinuxWorld Conference & Expo」において、Corelは「WordPerfect Office 2000 for Linux」を正式に発表した。

「WordPerfect Office 2000 for Linux」には、Standard版とDeluxe版があり、北米での発売は春から。Standard版に含まれるアプリケーションは、

「WordPerfect 9」（ワードプロセッサ）
「Quattro Pro 9」（表計算）
「Corel Presentations 9」（プレゼンテーション）
「Corel CENTRAL 9」（PIM）

となっている。Deluxe版ではさらに、

「Paradox 9」（リレーショナルデータベース）

テクニカルサポート
「entertainment pack」（「Railroad Tycoon - Limited Edition」や、米Lokiのゲームの体験版など）

などが含まれる。また、これらのアプリケーションは、Windows版「WordPerfect Office 2000」のデータと互換性があるという。価格は、Standard版が109ドル、Deluxe版が159ドル。

また、同社は「OpenSource.Corel.com」というWebサイトの開設も発表した。同Webサイトでは、同社のオープンソースでの開発の詳細情報や、ソースコードのダウンロードができるという。「WordPerfect Office 2000 for Linux」の新しいリリースについての情報も同Webサイトで得られるという。

Corel （http://linux.corel.com/）

Linux のビジネス活用を促進する団体「Linux Solution」が設立

2000年2月1日

アークワークス、システムデザイン研究所、ネットエージェント、ボニー電子技研の4社は1月31日、業務システム構築におけるLinuxの活用を推進する任意団体「Linux Solution」を設立した。Linux Solutionは、同団体のWebサイトを通じて、Linuxを使ったシステムの提案や事例などの情報提供を行っていく。

Linux Solutionが予定している活動内容は次のとおり。

・企業における情報戦略の中での Linux の普及啓蒙活動、システムの提案活動
・Linuxを活用した情報システムに積極的なサポート企業、個人の情報公開
・より現実的なシステム構築に則した技術情報の公開
・Windowsほかの異種混在環境下での具体的活用法の公開
・協賛各社と連携しての、全国各地におけるセミナーの開催

設立理由は、（Linuxの利用が）「まだまだネットワークや RDBMS のインフラのごく一部に留まったままであり」、「Linuxによるビジネス市場の拡大には企業の業務により則した形でのLinuxの活用方法の提案、セキュリティを向上させていくための具体的ノウハウ、サポート企業のリストアップなどの情報整理が必要と考え」たためだという。

Linux Solutionは、自身を企業に対して中立的な団体と位置づけている。そのため、誰でも情報を登録・閲覧でき、当面の間は会員資格の制限や会費の徴収は、特に行わない。

Linux Solution
(http://www.linuxsolution.org/)

Webブラウザ「Mozilla」α版リリース

2000年1月28日

ソースコードの公開以来、長らく待ち望まれていた「Mozilla」のα版がリリースされた。製品版でない現在のバージョンナンバーは、Milestoneと呼ばれ、今回リリースされたのはMilestoneの13番目にあたる「Mozilla Milestone 13（以下Mozilla M13）」。

現在、サポートされているOSは、

Win32
MacOS 8.5
Linux
OpenVMS
FreeBSD

である、Linux版は、glibc 2.1.xが必須。バイナリのファイルサイズは、それぞれ、

i386: 5.6Mバイト
PPC: 7.6Mバイト
SPARC: 8Mバイト

となっている。また、ソースコードは、すべてのプラットフォーム共通で、ファイルサイズは20.9Mバイトある。

今回の「Mozilla M13」Linux版では、フォント印刷などをバグフィックスしているが、まだまだバグは多く存在するという。

The Mozilla Organization
(http://www.mozilla.org/)

Distribution ▶▶▶

Kondara MNU/Linux 1.1、4月1日発売

デジタルファクトリジャパンは、「Kondara MNU/Linux 1.1」（以下Kondara 1.1）を4月1日より発売開始すると発表した。価格は7800円。Kondara MNU/Linux 1.0の登録ユーザー向けのアップグレード版は、4800円。学生ユーザー向けのアカデミックパックは、5200円。

Kondara 1.1は、世界で初めてシングルバイナリで日本語／英語両環境に対応したLinuxディストリビューションだ。また、デスクトップ環境や使用する言語を容易に選択できる「sdr」など、Kondara 1.1独自の機能を多数備えている。

カーネルは2.2.14、Cライブラリはglibc 2.1.3など最新のコンポーネントを採用する予定。今までの電子メール、FAXによるサポートに加え、新たに電話サポートも開始する。また、Kondara 1.0ではサポートの対象外だったAlphaマシンに対しても、90日3件までのインストールサポートを行うとしている。

デジタルファクトリジャパン
（http://www.digitalfactory.co.jp/）
Kondara Project（http://www.kondara.org/）

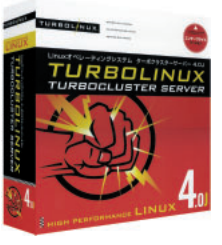
ビジネス用クラスタリングシステムTurboCluster Server 4.0J発売

ターボリナックス ジャパンは、同社のTurboLinux Serverをベースにした、クラスタリングサーバ「TurboCluster Server 4.0J」（以下TCS4.0J）を発売した。価格は、2ノード版が12万8000円、ノード数無制限版が24万8000円。

TCS4.0Jは、Webサーバやメールサーバをはじめとした、ビジネス向けのサービスをクラスタリング処理するための製品だ。

クラスター構成に組み入れられた各サーバ（クラスタノード）を、Cluster Manager（以下CM）と呼ばれる管理サーバが監視し、各ノードの異常を検出すると、そのノードを切り離して他のノードに肩代わりさせる。また、リクエストを負荷の低いノードに振り分ける「ロードバランシング」も可能。

サポートは、インストールおよびTCS4.0Jの設定バックアップまでで、電話、Webサイト、電子メールによる、サポートが90日間3件まで日本語で受けられる。



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

OpenLinux 2.3 eServer日本語版、3月10日発売

ネオナジーは、eBusiness用サーバに特化した「Caldera OpenLinux eServer 2.3 日本語版」（以下eServer 2.3）を3月10日に発売する。標準価格は、2万9800円。

eServer 2.3には、ファイル／プリントサーバ、ネットワークサーバ、VAR用カスタムサーバなど、いくつかの典型的なサーバ形式が、あらかじめ設定されており、インストール、設定、管理が簡単にできるようになっている。

Webブラウザベースの管理ツール「Webmin（日本語対応）」を採用しており、Webブラウザがあればサーバ設置場所以外からでも管理が可能。

商用アプリケーションとして、IBM WebSphere Application Server Standard Edition V2.03 for Linux（90日間トライアル版）、IBM VisualAge for Java for Linux（製品版）、Fontface4550 R Symon ［さいもん］バージョン1.2（製品版）が付属している。

サポートは、30日、5件までのインストールサポートが電話と電子メールで受けられる。また、Caldera社のナレッジベースWebサポートデータベースが、インターネット上で常時利用可能。

ネオナジー（http://www.openlinux.ne.jp/）

LinuxPPC 2000、2月28日出荷

米LinuxPPCは、同社のLinuxディストリビューション「LinuxPPC 2000」を2月28日に出荷すると発表した。価格は、CD-ROMセットが20USドル、ボックスセットが49USドル。

LinuxPPC 2000は、PowerMacintoshなどのPowerPC搭載マシンに対応したディストリビューションで、MacOSを利用せずにブート可能になった。以前のバージョンでは、BootXというMacOS上のプログラムを利用して、起動していた。

LinuxPPC上でMacOSを動かすことのできるGPLのフリーソフトウェアである、Mac-on-Linuxを採用している。エミュレーションではなく、ネイティブに動作するため、実用的な速度で動作するという。

日本国内では、3月下旬にアミュレットから、LinuxPPC 2000日本語版として4800円で発売が予定されている。

米LinuxPPC（http://linuxppc.com/）
アミュレット（http://www.amulet.co.jp/ppclinux/）

LinuxWorld Conference & EXPO in N.Y.

文・写真：宮原 徹
Text : Toru Miyahara

Linux専門のイベント、LinuxWorld Conference & EXPO（以下、LinuxWorld）が2月1日～2月4日の5日間にわたって開催された。

3回目となる今回は、前回、前々回までの開催地であったSan Joseから東海岸に舞台を移し、New Yorkでの開催となった。昨年のLinux界では、Red HatやVA Linux Systemsといった大手ベンダーのIPO（株式公開）が、従来のコミュニティベースからビジネスへの傾斜の現れとしてとらえられているだけに、この経済の中心地、New Yorkでの開催もかなり象徴的なことといえよう。

実際、開催地という形式的なことだけでなく、LinuxWorldのいたるところでそういった流れを感じさせる内容であった。このあたりを踏まえながら、今回のLinuxWorldの様子をレポートする。

基調講演

Linus Torvalds

Linuxのオリジナル開発者であり、現在も開発の中心にいるLinus氏が多くの人の前で話すのは、今のところLinuxWorldしかない。そのため、会場には3000人以上の聴衆が押しかけ会場は満員の大盛況であった。これだけの人を集められるのは、ほかにはBill Gates氏、Steve Jobs氏ぐらいしかいないという、主催者のIDG社社長Chary Gleco氏の弁もあながち大げさとはいえないだろう。昨年の開催では夕刻から始まるパターンだったのだが、今回はビジネス、特にウォール街や最近東海岸で伸びつつある「シリコンアレイ」と呼ばれるIT関係ビジネス関係者などを意識したのか、初日（前日にチュートリアルがあるため、本当は2日目なのだが、展示会なども2日目から始まるので実質的には初日）の9時半からの講演開始となった。

期待を持って臨んだ基調講演だったが、残念ながら今回の講演では特筆すべきことはほとんど話されなかった。ただ、8月に行われた前回の講演では「12月にはリリース」としていたLinux kernel 2.4は、LinuxWorldが終わってからリリース準備に入ると述べ、Linux kernel 2.4のリリースが間近であることを表明した。しかし、詳細については後述するが、この新カーネルのリリースの遅れが今回の展示やそのほかに及ぼした影響は大きかったと思う。

Linus氏は、いつものごとくいくつかの聴衆からの質問に答えたあと、これまた恒例になっている「IDG/Linus Torvalds Community Award」の表彰に入った。今回は、X Window Systemのフリーな移植を行っているXFree86 Projectが受賞し、賞金の$25,000を受け取った。

Larry Augustin

2日目はVA Linux Systemの社長、Larry Augustin氏の基調講演で幕を開けた。同社は株式のIPOを行い、公開初日の株価上げ幅でNASDAQの記録を更新するなど、Linuxビジネスのトップランナーと目されているだけに、その動向が注目されている。Linus氏の基調講演には及ばないものの、1500名以上の聴衆を前にVA Linuxの行っている活動などを紹介した。

最初にIntelを含めた数社で進めているLinuxのIA-64（Intelの64ビットアーキテクチャCPU）への対応についての進捗状況を明らかにした。IA-64のCPUは、今年の中ごろに最初の製品であるItanium（コード名はMerced）が出荷される予定であるだけに、各OSベンダーともその対応競争が繰り広げられているところである。デモンストレーションでは、従来のようなIA-64のソフトウェアエミュレーションではなく、実際のIA-64 CPUを使ったマシン上で3DのXアプリケーションなどの動作を見せていた。

また、同社が開発したオープンソース開発支援サイト「SourceForge」が紹介された。SourceForgeは、従来ばらばらになっていた掲示板やメーリングリストなどのコミュニケーション手段やCVS（Control Version System）のようなソースコード履歴管理システムをすべてWebをインターフェイスとした統合したシステムとしてコミュニティに提供するというものである。

最後に、同社がSlashdotやFreshmeatといったコミュニティの情報交換サイトを運営するAndover.netを買収したことが発表された。IPOで得た膨大な経営資金の行き先は、企業価値を増大させようとするM&Aや、ユーザーやコミュニティへの還元を指向する支援活動などが考えられるが、今回のAndover.netの買収は後者を目指したものと説明した。今後の同社の動きでこの買収の持つ意味がはっきりするだろう。

展示会場

今回の展示会場を見て回って、いくつか新しい傾向があることに気づいた。ここでは、私の分析を交えながらレポートしてみよう。

■ディストリビューションは飽和状態

昨年のLinuxビジネスの主役は、Linuxディストリビューションであったといえる。しかし、今回の展示会場ではこの状況がほぼ飽和状態に達していた感があった。出展していたディストリビューションベンダーは、ざっと挙げただけでも10以上はある。

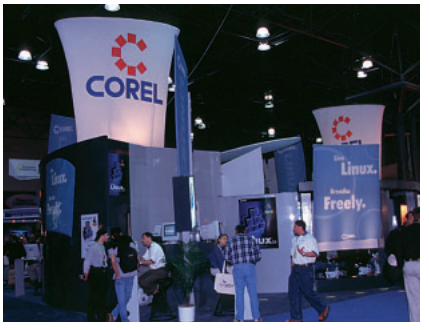
Linux kernel 2.4のリリースが遅れているため、どのディストリビューションもkernel 2.2.11～13あたりを組み込んでおり、それぞれに目立って大きな違いが感じられるわけではない。市場的には「踊り場」という感じで、今年は「差別化」が焦点になるだろう。

方向性としては、ビジネス・ハイエンドユースを目指す組と、コンシューマー・ローエンドを目指す組に分かれつつあるといえる。そんな中にあって、イメージカラーをイエローに統一し、クラスタリングなどのハイエンド向けを指向し始めた展示をしていたTurboLinuxのブースが印象に残った。ローエンドでは、Corel LINUXの完成度が高いといわれているが、展示会場内の即売ブースを見る限りでは、Linux Mandrakeのほうが売り手、買い手ともに関心が高かったようだ。

勢ぞろいしたディストリビューションのブース（上段左から右へ）

Red Hat、Corel、TurboLinux、Conectiva、SuSE、Caldera、Stormix、Mandrakeなど、相変わらずにぎやかな感じだ。

これ以外にも、Slackware、Debian GNU/Linuxなども出展していた。



■落ち着きつつあるハードウェア

数が多すぎて混乱状態にあるディストリビューションベンダーに対して、ハードウェアベンダーのほうは、数は多くてもそれほど戸惑いは感じなかった。これは、ハードウェアベンダー各社が、Linuxをすでに選択肢の1つとして普通にとらえて対応してきているためで、受け手の側もそれぞれのベンダーの従来の戦略の延長上にLinux戦略をとらえられるからであろう。IBMやCOMPAQ、DELLのように、他の商用UNIXやWindows NTなども手掛ける総合ハードウェアベンダーがまさにそれで、Linuxのハードウェア周りは、よい意味で安定期に入ったといえるかもしれない。

一方、VA LinuxやSGIといったベンダーはまさにLinux一色という感じで、VA Linuxなどは入り口正面という一等地に大きくブースを出して存在感をアピールしていた。ただし、同社は企業向け、ハイエンド向けなどのソリューションに乏しく、ブースに足を止めていってくれる人も少なかったようだ。このあたりのスケーラブルなビジネスに対応できていないところが同社の弱点であろうか？

■立ち上がりつつある組み込み系ビジネス

組み込み系ベンダーの出展は、昨年までは「なんでもやります。だから仕事ください」というような受託開発中心のものが多かったのに対して、今年はLINEO社の「Embedix」を始めとして、使用できるハードウェアも明確になってきている。

たとえば、タッチパネルと連動してキオスク端末が作れるといったように、きちんとパッケージ化されたものが多く目立ってきた。これだけで、Linuxの組み込み用途を想定したビジネスが立ち上がったと判断するには早計かも

■LPI
Linuxの技術者検定試験の標準をまとめている団体「Linux Professional Institute」のブース。日本でも準拠した検定試験の実施が計画されている。

■Transmeta
Linus Torvalds氏が在籍していることで有名なTransmetaの新型CPU「Crusoe」の紹介。

■SGI
ローエンドソリューションをWindows NTからLinuxにシフトし勝負するSGI。IRIXでの技術をLinuxに投入しつつあるが、今後はどのあたりを差別化要因にしていくのか注目される。

■Sun Microsystems
Java2のLinux版リリースでコミュニティとの軋轢を生んでしまったためかどうかは不明だが、Java関係はちょっと控え目。会場内の至るところで見かけたのが昨年同社が買収した「StarOffice」（商標の関係で、国内ではSunOfficeとなるらしいが）のトライアル版CD-ROM。何枚作ったんでしょう？

しれないが、Linuxのコア開発者であるLinus氏自身がTransmeta社でmobile Linuxを開発していることなども市場に好材料と判断されて、今後徐々に浸透していくのではないかと思われる。

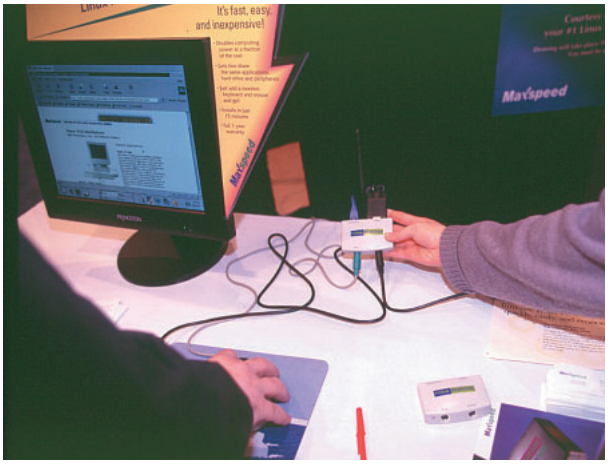
■待たれる最新カーネル

Linux kernel 2.4のリリースの遅れの影響もあってか、今回のLinux Worldには特に目新しいトピックはなかった。そういう意味では、前回、前々回のLinuxWorldと比較すると、やや盛り上がりに欠けるような感じがしたのは否めない。

しかし、出展数は着実に増えつつあり、その内容も回を追うごとにまとまってきている。そういう意味では、Linuxが、昨年のめまぐるしい動きから一息ついて、じっくりと市場形成に注力できる安定期に入ったともいえるのかもしれない。これに最新カーネルのリリースが加われば、一段と弾みがつくはずで、期待は大きい。

ただ、本格的なLinux市場の形成には新しいコンセプトのビジネスモデルが必須となるはずなのだが、Red HatやVA Linuxといった、Linuxビジネスのトップを走っている企業も、未だに新しいビジネスモデルを築けずにいるのも事実だ。

年率25%ともいわれるLinuxの成長が、今後どこに向かっていくのかは、離陸したばかりの今年が1つの分岐点となることは間違いないだろう。今年8月に、再びSan Joseで開催されるLinuxWorldで、その行き先を確認できるはずだ。



■+One Station

会場で見かけたMaxSpeedの面白い新商品「+One Station」。この箱とマシンにセットしたPCIカードをケーブルで接続すると、マシンとの入出力が2系統になり1台のマシンを2人で同時に使えるようになるという優れもの。

■VMware

Linux上で仮想マシンを動かす、お馴染みVMwareもVer2.0がお目見え。旧バージョンよりもパフォーマンスが向上するようです。でも、日本での販売は未定とのこと。早く国内版も出してほしいですね。

■IBM

Linuxをソリューション戦略の中心に据えるIBM。しかし、注目のLotus NotesのLinux版は見当たらず。どうしちゃったんでしょう？

■SourceForge & Andover.net

VA Linux Systemが開発したオープンソースコミュニティ向け環境「SourceForge」は、同社のブースできっちりデモンストレーションを（左）。一方、同社に買収された「Andover.net」のブースでは床に座り込んでメールを見るでもなし、話をするでもなし、というGeekやNerdの集まりが（右下）。好対照の両者がLinuxの現状を表しているような…。

JAPAN Linux Pavillion

過去2回、San Joseで開催されたLinuxWorldを見に行って、日本と米国とのLinux環境の違いや温度差みたいなものを感じてはいたのだが、それはあくまでも傍観者という視点からのもので、今ひとつ実感に欠けるものだった。それなら今回のLinuxWorldでは、情報提供側となる出展社として参加することができないかと考えた。幸い、IDGジャパンの協力と4社2団体の賛同を得て実現することができた。

さて、現代の咸臨丸として上陸を果たした、JAPAN Linux Pavillionの様子についてもレポートしていこう。

■日本ユニシス

http://www.unisys.co.jp/

日本ユニシスは、クロスプラットホームを実現したオブジェクト指向開発ツール「TIPPLER for Linux」と、Webから利用できるナレッジマネジメントツール「WebWorker for Linux」の2製品を展示した。「TIPPLER for Linux」は、業務向けの開発ツールの出展が少なかったため、関心を集めていたようだ。また、一方の「Web Worker for Linux」も、米国ではASP（Application Service Provider）のようにWebベースの業務アプリケーションの普及が見込まれているらしく、注目が集まっていた。

■アクアリウムコンピューター

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

マイクロインターネットサーバ「blue grass」および「white neon」を出展。オフィスが広い米国でも、小さいマシンはカッコよく見えるようで、注目の的となっていた。また、米国といえでも、誰もが高速なネットワークを利用して、ホスティングサービスやらASPを利用しているわけではなく、コンパクトタイプのサーバの需要は多いことなど、貴重な声を聞くことができた。

■ホライズン・デジタル・エンタープライズ

http://www.hde.co.jp/

Webを利用して簡単にLinuxを管理できる「HDE Linux Controller」を出展。今回の展示のために用意した1000枚のトライアル版CD-ROMが飛ぶように次から次へと来場者の手に。UNIX、Linuxの管理ができる技術者が豊富にいるわけではないという状況は、米国でも日本でも同じようである。

■日本リヌックス協会

http://jla.linux.or.jp/

日本での活動紹介、および4月に行われるLinux Conference 2000の案内を行った。米国の場合、国土が広い関係もあって、大都市にはユーザーグループが存在するが、それ以外の地方では、いろいろなテーマに基づいて活動するグループが点在しているという感じらしい。そういう意味では、コミュニティ形成のための地理的要因が日本とはかなり異なるため、来場者の興味を引いたようである。

■Linux Internationalisation Initative（LI18NUX）

http://www.li18nux.org/

Linuxの国際化プロジェクトに関する出展を行った。アジアから遠い東海岸という場所にも関わらず、アジアからの参加者は思った以上に多かった。これまでは日本語化が大きな関心事だったが、これからは国際化にもっと関心を寄せるべきだろう。そして日本国内だけでなく、アジア、そして世界を視野に入れながらのさまざまな活動が必要なのではないだろうか？

Products

ひとつのキーボード、マウス、ディスプレイで8台までのPCをコントロール可能

PShare 8

複数のPCを使っていると、それぞれのキーボードやマウスが場所を取って邪魔なものである。ディスプレイ切り替え器を使ってひとつのモニタで済ませている人もいるだろうが、PShare 8を使えば、キーボード、マウス、ディスプレイを同時に切り替え可能だ。

製品名	PShare 8
価格	11万円
問い合わせ先	ぷらっとホーム株式会社 TEL 03-3251-7611 http://www.plathome.co.jp/

ぷらっとホーム株式会社から、8台までのPCを1セットのコンソールからコントロールできる、PC切り替え器「PShare 8」が発売された。定価は11万円、同社のインターネットショップ「ぷらっとオンライン」では9万9800円で購入できる。

ディスプレイだけでなくキーボード、マウスも1個でOK！

個人でも会社でもLinuxを使うようになって、Linux専用機とWindowsマシンの2台のPCを同時に使い分けている人は結構多いと思う。

ディスプレイ自体にBNCコネクタとD-SUBコネクタの2つの入力があるものを使用したり、ディスプレイ切り替え器を利用している人もいるだろう。その場合、画面は切り替わっているのに、見えていないほうのPCのキーボードやマウスを間違えて操作してしまうこともよくある話だ。

PShare 8を使えば、ディスプレイと同時にキーボード、マウスも切り替わるので間違える心配はない。もうPCごとに別のキーボードやマウスをつないでおく必要もないので、PCの周りがすっきりと片づく。

設置は簡単で、まずキーボード、マウス、ディスプレイを本体背面（**写真2**）の一番右側のコンソールポートにつなぐ。次に専用ケーブル（別売、1セット定価5000円）を使用して各PCと接続すれば終了である。PCのマウスポートからの電源を利用するため、外部電源は不要となっている。

使い方だが、本体前面のスイッチパネル部の右側にある1～8のスイッチを押すことで、それぞれのPCに切り替えることができるほか、キーボード操作（ホットキー）によっても切り替えが可能である。

写真1　ラック取り付け時には分離可能
スイッチパネル部とコネクタ部が分離できるため、19インチラックに取り付ける時の作業が楽。高さはわずか42mm（1Uサイズ）と薄いので場所を取らない。

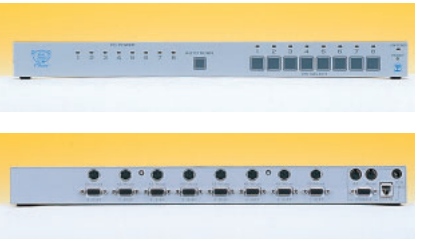


写真2　PShare 8前面（上）、背面（下）
正面左側がPCの電源状態、右側の1〜8のスイッチを押せばそのPCに切り替わる。背面のコネクタから各PCへ接続する。マウスポートとキーボードへは専用ケーブルを使用する。

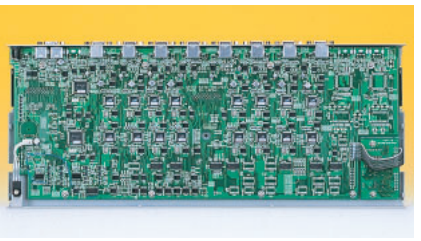


写真3　PShare 8内部基板
各ポートごとに専用の制御用マイコンを搭載していて、PCごとにキーボードやマウスの状態を記憶している。切り替わったときにはCAPSやNUM Lockの状態も正しく復帰する。

ホットキーは3種類用意されていて、CtrlとAltとShiftの3つのキーを同時に押す、Ctrlキーを連続して押す、Scroll Lockキーを連続して押すのどれかを実行すると、ホットキーモードになる。ホットキーモードで数字キーを押すと、番号に応じたPCに画面が切り替わり、Enterキーでそのを選択する。ESCキーを押せばホットキーモードから抜け、切り替え前のPCに戻る。

アプリケーションによっては、ホットキーによって使えないキー入力が起こる可能性がある。その場合には、本体裏面にあるディップスイッチで3種類のホットキーを、それぞれ有効／無効に設定できるようになっている。

また、オートスキャン機能も持っていて、一定時間ごとに起動しているPCを順番に切り替えて表示することが可能だ。全部のPCを一定間隔でモニターするといった用途に使える。

カスケード接続で最大64台までのPCに対応

写真1のように、本体は高さが42mmと非常に薄く、横幅も19インチラックに合わせたサイズとなっている。専用のラックマウントホルダーを使用すれば1Uサイズのスペースに収納できる。そのため、ISPなどで1台のラックに複数のサーバを搭載している場合に最適である。

PShare 8は1台で8台のPCを切り替えることができるが、それぞれのポートからさらに別のPShare 8をカスケードに接続することで、8ポート×8ポート＝64台までのPCを接続することが可能だ。

内部回路には、専用の制御用マイコン（MPU）を使用していて、それぞれのPCごとの状態を記憶しているため、キーコードモードやNum Lock、Caps Lock、Scroll Lock、マウスの状態などは、選択したPCごとに切り替え前の最後の状態に戻るようになっている。

また、ディスプレイは最大1600×1200ドット、帯域幅200MHzまでサポートされているので、高解像度表示にも十分使える実力があるといえよう。

個人向けの2ポート製品がほしい

実際に3台のPCをつないで使用してみたのだが、ディスプレイのすぐ前にあるキーボードですべて操作できるのは快適である。本体前面のスイッチで切り替えるのは最初のうちだけで、そのうちにホットキー機能だけを使用するようになってしまった。ただし、画面を切り替えたはずなのに、スクリーンセーバで画面が消えていて、何も表示されなくてあわてるといったこともあった。まあ、これはPShare 8とは関係ないことだが。

PShare 8は、8ポートというポート数の多さや価格の点から、やはりISPなどでサーバをラックに入れて使っている企業向けという製品である。

4ポートのPShare 4という製品もあるのだが、定価で6万4000円（ぷらっとオンラインで3万9800円）と個人で買うにはちょっと抵抗がある価格かもしれない。また、同時に使うPCは2台で十分というような人のために、ぜひ2ポートで安価なPShareの発売を期待したい。

対応機種		PC/AT、DOS/V互換機
接続台数		最大8（カスケード接続時：最大64）
選択方式		セレクトボタン、キーボード（ホットキーモード）、フットスイッチ
ホットキーモード		マニュアル／オートスキャンモード
オートスキャン周期		3、5、10（初期値）、20、40、60秒
コンソールポート	KB	PS/2、Mini DIN 6Pメス×1
	マウス	PS/2、Mini DIN 6Pメス×1
	モニタ	Mini D-SUB 15Pメス×1
PCポート	KB／マウス	PS/2、Mini DIN 10Pメス×8
	モニタ	Mini D-SUB 15Pメス×8
対応キーボード		PS/2準拠101〜109キーボード
ディスプレイ解像度		最大1600×1200（60Hz）、200MHz
外形寸法（mm）		437（W）×200（D）×42（H）
重量		2.8kg
最大消費電力		125mA（PC1台）

表1　PShare 8の主な仕様

ツールバーが付いて日本語変換機能が使いやすくなった

VJE-Delta Ver.3.0 for Linux/BSD

入力モードは常にツールバーに表示されているし、VJEを常に起動しておいても操作に不都合はない。WindowsのIMEやATOKと比べると、どうも使い勝手が悪かったLinuxの日本語入力ソフトだったが、このVJE-Delta 3.0で操作性が抜群に向上した。

製品名	VJE-Delta Ver.3.0 for Linux/BSD
価格	5800円
問い合わせ先	株式会社バックス TEL 042-724-9200 http://www.vacs.co.jp/

株式会社バックスから、LinuxおよびFreeBSDで動作する日本語入力システム「VJE-Delta Ver.3.0 for Linux/BSD」（以下VJE 3.0）が発売された。1998年8月より発売されていた「VJE-Delta Ver.2.5 for Linux/BSD」をバージョンアップした製品で、従来の9800円から5800円に価格を改定した。

MS-DOSの時代から連文節変換を実現していたVJEの最新版は、豊富な辞書と最新の文法解析技術、AI変換アルゴリズムによって最適な日本語変換を実現した。また、日本語ワードプロセッサVJE-Penをフリーソフトとして添付しているので、簡単に使えるエディタとして利用するのもよいだろう。

最近のソフトウェアのほとんどがA4サイズ程度のパッケージに入っているが、VJE 3.0は、コンパクトなCDケースで提供されている。そのためマニュアルは、CD-ROMに収録されているオンラインマニュアルを利用することになる。なお、簡単なインストール方法はパッケージのふたに付いている紙に書かれている。

画面1　Deltaツールバー
Windows版と同じようにツールバーが表示され便利になった。起動時には上のように灰色だが、日本語入力がオンの状態だと下のようにカラーになる。

インストールの実際

CD-ROMのlinuxディレクトリには、VJE 3.0が下記のような3種類のアーカイブで収録されている。

vje-delta-3.0-0.1.i386.rpm
vje-delta-3.0-0.1.deb
vje-delta-3.0.tgz

1番上はRed Hat系のRPM用、2番目はdebian用、3番目はSlackware用である。同じディレクトリに、インストールに関する諸注意が書かれたreadme.txtファイルがあるので、それを読んで作業を行う。

今回はLASER5 Linux 6.0 Rel.2にVJE 3.0をインストールした。CD-ROMをマウントして、rpmコマンドでインストールすればセットアップは終了である。

```
# rpm -Uvh vje-delta-3.0-0.1.i386.rpm
vje-delta    ####################~
VJE-Delta 3.0 Install Start.
VJE-Delta 3.0 Install Complete.
```

展開されたファイルは、/usr/local/vje30ディレクトリに置かれる。オンラインマニュアルは、/usr/local/vje30/docディレクトリに、delta.htmというHTMLファイルで置かれている。X Window Systemのktermから、“netscape /usr/local/vje30/doc/delta.htm”と入力すれば、Netscape Navigatorでオンラインマニュアルを簡単に見ることができる。VJE 3.0を操作すると

画面2
VJE 3.0に付属している日本語ワープロソフトVJE-Penは、フリーソフトとして添付されている。エディタ代わり使うことも可能だ。

きの画面表示と同じ画像を張り付けてあるため、わかりやすいだろう。

VJE 3.0は、アプリケーションへ日本語入力データを渡す方法として、X Window Systemで標準的に使われるXIM（X Input Method）を用いている。

VJE 3.0を使うには、環境変数をセットしておく必要があるので、ユーザーごとに.bashrcファイルに、以下の2行を追加する。

```
export LANG=ja_JP.eucJP
export XMODIFIERS=@im=vje
```

次に、コンソールから“vjed”と入力してVJE-Delta変換サーバを起動する。そして、“vje &”としてVJE-XIMクライアント（IMサーバ）を起動する。正常ならば、ウィンドウ画面の右下にツールバーが表示される（**画面1**）。

VJE-Delta 3.0の新機能

Linux上のアプリケーションの全部で日本語入力が行えるわけではない。ktermやNetscape Communicatorなどはもちろんだいじょうぶだが、XIMに対応していることが必須である。とりあえずVJE-Penを使ってみて、VJE 3.0の使い勝手を見てみよう。

日本語入力モード（以下日本語モード）にするには、漢字キーを使用する。Ctrlキーとスペースキーを同時に押してもよい。ツールバーがカラーに変われば日本語モードに切り替わっている。

てきとうに文章を入力していくのだが、英数字モードと日本語モードの切り替えは、スペースキーのすぐ右にある変換キーを押すだけで行える。

また日本語モードのままでも、EnterキーやTABキー、Back Spaceキーなどがアプリケーションに伝わるのでストレスなく入力できる。今までのLinux用日本語入力ソフトは、この点が不便であった。日本語モードでは、これらのキーは文字を確定させるためにしか使えなかったので、ふだんは日本語モードをオフして使うのが普通だった。すなわち、切り替えのためにシフトキー＋スペースキーを押す回数が非常に多かったのである。

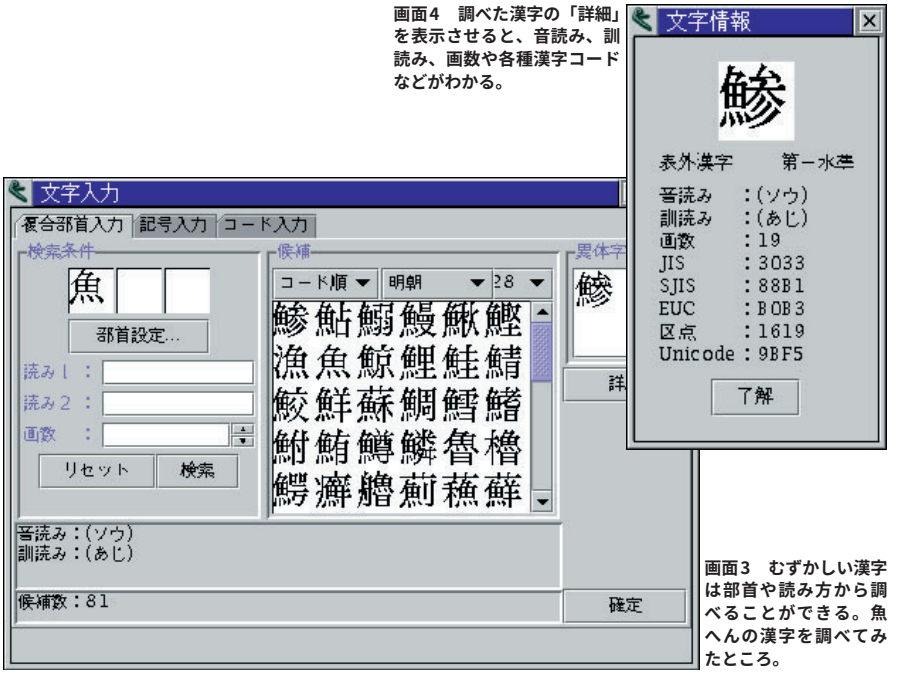
■辞書も充実

さて、肝心の日本語変換の精度であるが、離れた文節間の係り受け関係を解析して、最適な日本語を選び出す構文解析機能によって誤変換を少なくしている。文節ごとにこま切れに確定しても精度が落ちないように、確定後の文字列との係り受けを判断するといった工夫が施されている。

辞書は、Windows 95/NT版のVJE-Delta Ver.2.5と完全互換であり、23万語が登録されている基本辞書のほかに、郵便番号辞書（11万8000）、人名辞書（25万）、全国地名辞書（7万）、単漢字辞書（2万）、学習辞書、顔辞書が用意されていて、最大10ファイルの辞書を並列に引くことが可能となっている。

VJE 3.0から文字入力アクセサリが追加された。読み方のわからない漢字があった場合に、部首や画数を組み合わせて文字を見つけることができるユーティリティである（**画面3、4**）。そのほかに文字入力アクセサリは、記号の入力や漢字コードからの入力もサポートする。

文字入力アクセサリは便利な機能であるが、Java Runtime Environment 1.1.7以上が必要となっている。Java環境を含んでいないディストリビューションも多いので、動作させるためには、SunのWebサイト（http://java.sun.com/）からLinux用のJavaをダウンロードしてくる必要がある。この点は不便なところである。



画面4　調べた漢字の「詳細」を表示させると、音読み、訓読み、画数や各種漢字コードなどがわかる。

画面3　むずかしい漢字は部首や読み方から調べることができる。魚へんの漢字を調べてみたところ。

Distribution

新着ディストリビューション

Red Hat Linux 6.1 改訂 日本語版

RPM系ディストリビューションの本家、Red Hat Linuxの日本語版がマイナーバージョンアップ。安定度を増すとともに、日本語への対応を一層強化した。今回のバージョンアップによる強化内容とインストール方法を解説する。

TurboLinux 4.5

適切な初期設定値で、インストール直後から実用的な日本語環境を提供するTurboLinuxがセキュリティホール対策やバグフィックスをしてリフレッシュした。最新版のXFree86 3.3.6も同梱して、より多くのグラフィックスカードにも対応。FTPや雑誌付録などで配布するこの最新バージョンを紹介する。

マイスター Linux Mandrake 6.1

五橋研究所は、最新のオープンソースソフトウェアを低価格で提供するOSパッケージ「マイスター」シリーズを新たに展開。シリーズの先頭を切って発売されたのがこのマイスター Linux Mandrake 6.1だ。フランス生まれのおしゃれなディストリビューションをお目にかける。

Red Hat Linux 6.1 改訂 日本語版

レッドハットは、2月10日からRed Hat Linux 6.1 改訂 日本語版（以下Red Hat 6.1改）を1万2800円で販売開始した。本製品は、昨年11月より販売されているRed Hat Linux 6.1 日本語版をマイナーバージョンアップしたもので、バグフィックスのほか、日本語化されたSambaなどを追加収録している。既存のユーザーは、レッドハットのFTPサイトよりRed Hat 6.1改を入手することができる。

より安定した環境を提供

今回のバージョンアップでは、カーネル（2.2.12）や標準Cライブラリ（glibc 2.1.2）、XFree86（3.3.5）など、基本的なシステム部分はバージョンを上げず、リリース番号だけが上がっていることから、安定度の向上に主眼をおいたバージョンアップだといえよう。

また、ATOK12 SEやDynaLabの日本語TrueTypeフォントといった商用ソフトウェアがRPMパッケージとして提供されるようになった。従来のバージョンではtarアーカイブで提供されていたのだが、RPM化によりインストール作業が格段に楽になった。

このほか、セキュリティホールをふさぐため、ネットワーク関係のデーモン類が数多くバージョンアップされている。このため、Red Hat Linux 6.1 日本語版のユーザーは、バージョンアップすることをお勧めする。

日本語対応を強化

新規に追加されたパッケージは、日本Sambaユーザ会が作成した日本語ファイルネーム対応のSamba 2.0.5aJP2、XEmacs、 Netscape Communicator 4.7など、日本語が利用できるソフトウェアが中心だ。このほか、Linux Japanese Locale Working Groupが公開した指針に則ってja_JP.eucJPロケールをサポートしたり、Linuxconfを日本語化するなど、日本語への対応が強化されている。また、X Window SystemでTrueTypeフォントを利用するための、X-TTが追加されているとのことだが、編集部で試用した限りでは、日本語TrueTypeフォントが正しく表示されなかった。パッケージ依存関係の警告を無視して、Linux Mandrake日本語化キットに含まれるXFree86 3.3.6をインストールすると正しく表示できたので、メーカーには修正情報の公開をお願いしたい。

付録CD-ROMにFTPを収録

本誌の付録CD-ROMに、Red Hat 6.1改のFTP版を収録した。このFTP版には商用プログラム、商用フォント、サポートは含まれないので注意してほしい。

Linux上級者を自認するなら、一度はインストールしておきたいディストリビューションだ。

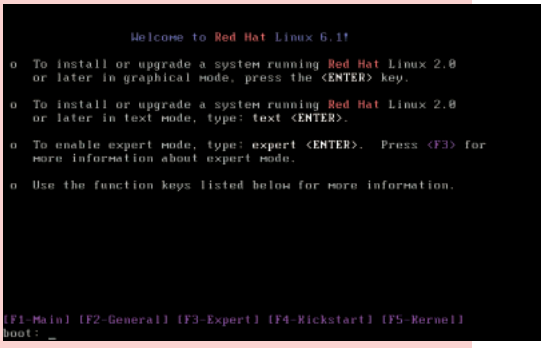
画面1　従来のバージョンではインストールが面倒だったATOK12 SEなどがRPMパッケージ化され、気軽にインストールできるようになった。

製品名　Red Hat Linux 6.1 改訂 日本語版
価格　1万2800円
問い合わせ先　レッドハット株式会社
03-3257-0411
http://www.redhat.com/jp/

Red Hat Linux 6.1 改訂 日本語版のインストール

CD-ROMブート可能なマシンでは、Red Hat Linux 6.1 改訂日本語版（以下Red Hat 6.1J改）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、Windows上で起動用フロッピーを作ります。手順は以下の通り。

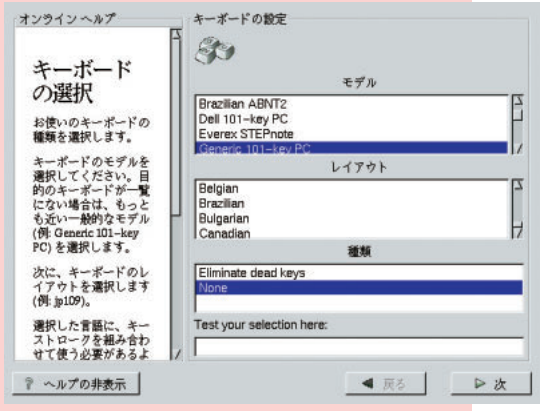
(1)Red Hat 6.1JのCD-ROMをドライブにセット。(2)DOS窓を開き、D: [Enter]と入力。CD-ROMドライブが「D:」以外の場合は、それを入力。(3)cd \dosutils [Enter]と入力。(4) rawrite -f \images\boot.img -d a [Enter]と入力。(5)「Please insert a formatted diskette into drive A: and press --ENTER--:」と表示されたら、フォーマット済みのフロッピーディスクをAドライブに入れ、Enterキーを押し、しばし待つ。



インストーラの起動

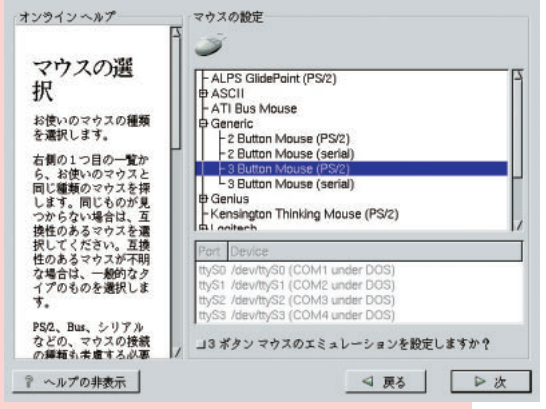
CD-ROM、またはフロッピーから起動すると、黒地に「Welcome to Red Hat Linux 6.1!」と表示されますので、[Enter]を押します。

旧バージョンと同じテキストベースでインストールしたい場合、またはグラフィカルインストーラが動作しなかった場合は、text [enter]と入力します。



キーボードの選択

キーボードの種類を選択します。106または109タイプのキーボードならば、モデルは「Japanese 106-key」を、レイアウトに「Japanese」を選びます。



マウスの選択

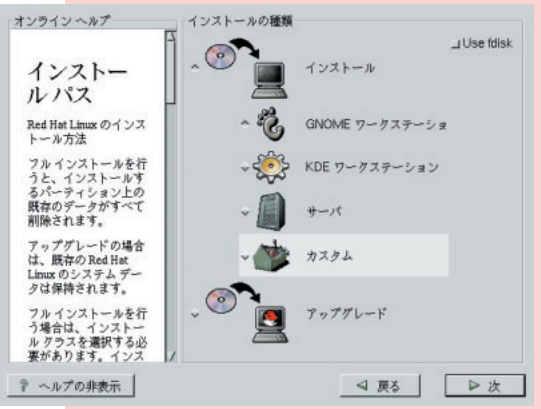
現在使われているマウスは、ほとんどがPS/2マウス、またはシリアルマウスです。これらを使っているなら、「Generic」という項目の下から選びます。2ボタンマウスを選ぶと、自動的に「3ボタンマウスのエミュレーションを設定しますか？」という項目が有効になります。これは左右のボタンを同時に押すことで、中ボタンの代用をさせる機能です。

インストールパス

インストール方法を選択します。「GNOMEワークステーション」「KDEワークステーション」「サーバ」は、そのマシンをRed Hat 6.1J改専用にする際に便利な選択肢です。**これらを選択すると、ハードディスク内の今あるパーティションを削除します。すでに別のOSがインストールされていて、デュアルブートで使用するつもりなら、絶対に選択してはいけません。**Windowsとのデュアルブート環境や、インストールするソフトを自分で決めたいときは、「カスタム」を選びます。また、「Use fdisk」でfdiskによるパーティション設定もできます。

以下、「カスタム」を選択したと想定して、説明を続けます。

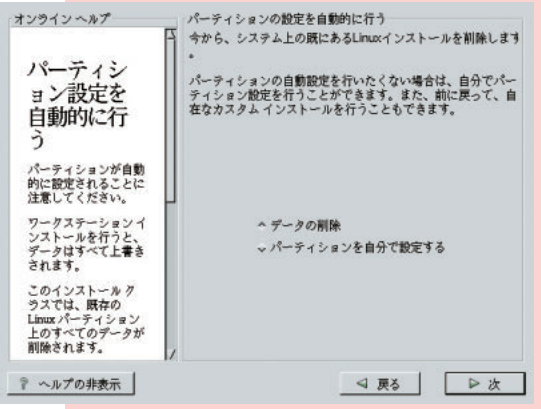
「アップグレード」は、Red Hat Linux 6.0などに上書きアップグレードする際に選択します。



パーティション設定を自動的に行う（前画面で「カスタム」以外を選択時）

この画面はインストールパスの設定画面で、「カスタム」以外を選択したときに現れます。「データの削除」を選ぶと、今あるパーティションやデータをすべて削除して、Red Hat 6.1J改用にパーティションを切り直しますので、Red Hat 6.1J改専用マシンにするとき以外は選択してはいけません。

「パーティションを自分で設定する」を選ぶと、インストールパスの画面で「カスタム」を選択した場合と同じ画面に移動します。

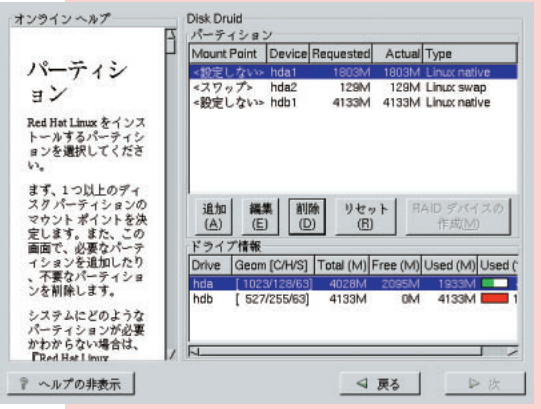


パーティション

手動でパーティションを設定します。Red Hat 6.1J改をインストールするには、少なくともLinuxシステム用とスワップ用の2つのパーティションが必要です。Linuxシステム用のパーティションを作るには「追加」ボタンを押し、サブウィンドウで「マウントポイント」を「/」に、「パーティションタイプ」を「Linux native」に、「サイズ」をMバイト単位で指定します。全部のパッケージをインストールするには、2Gバイト（2048Mバイト）程度あれば大丈夫です。

スワップパーティションは、「パーティションタイプ」を「Linux swap」にします。サイズは64〜128Mバイトくらいが適当です。

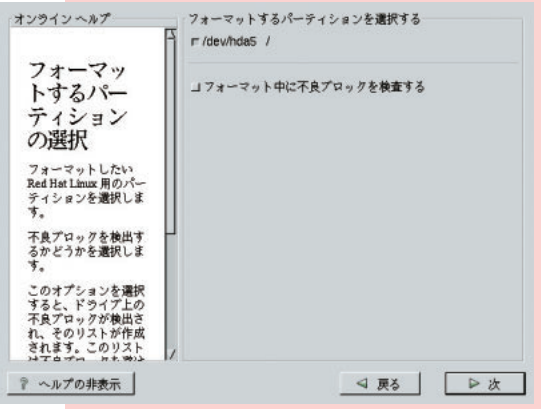
デュアルブート環境を作る際には、パーティション構成をメモしておくことをお勧めします。

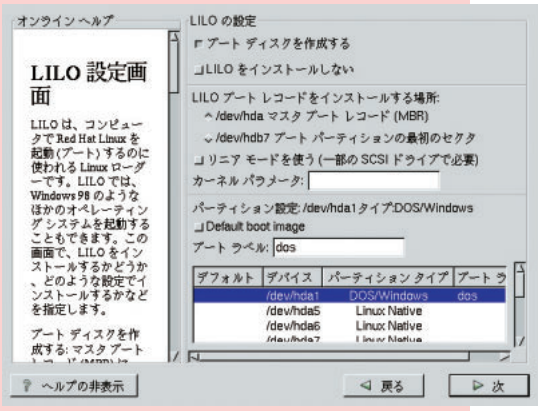


フォーマットするパーティションの選択

ここではLinuxをインストールするパーティションが表示されています。画面では/dev/hda5になっています。チェックが入ったパーティションはフォーマットされ、その中のデータは削除されます。そのパーティションを本当にフォーマットしてもよいのか、もう一度確認しましょう。

「フォーマット中に不良ブロックを検査する」にチェックをしておくと、フォーマット中にハードディスクの不良ブロックを確認します。フォーマットに時間がかかりますが、チェックしておくのが安全です。



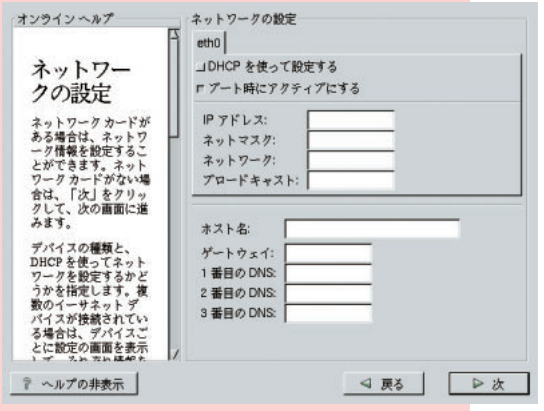


LILO設定画面

Linuxを起動するローダの設定です。「ブートディスクを作成する」をチェックすると、この後で起動用フロッピーを作成します。備えあれば憂いなしなので、作成しておきましょう。「LILOをインストールしない」は、常に起動用フロッピーを用いる場合にチェックします。

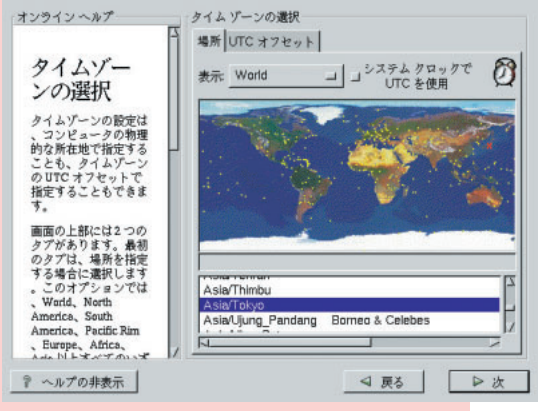
「LILOブートレコードをインストールする場所」は、システムコマンダーなどのブートセレクタを利用しているならば、「ブートパーティションの最初のセクタ」を選びます。Red Hat 6.1J改専用マシンにする時や、Windows 9xとのデュアルブートをさせる時は、「マスターブートレコード」を選びます。

画面下のリストからLILOに登録するOSを選択し、ラベルを付けると、そのOSを選んで起動できるようになります。



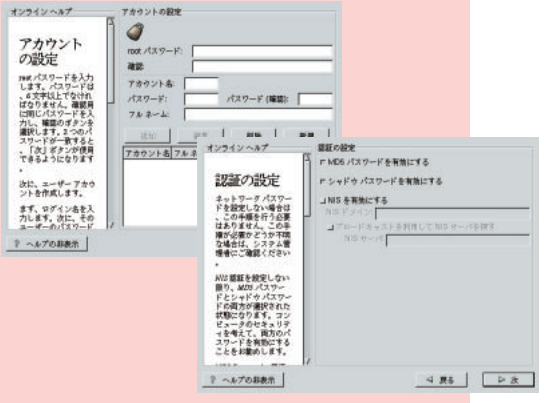
ネットワークの設定

インストーラがネットワークカードを自動認識していると、この画面が表示されます。「DHCPを使って設定する」は、LAN内のDHCPサーバからIPアドレスを取得する場合に選びます。IPアドレスを固定的に設定する場合は、IPアドレス以下の各項目を指定します。



タイムゾーンの選択

マシンの設置場所と協定世界標準時（UTC）の時差を設定します。日本国内で使うならば、世界地図上で東京を指定します。



アカウントの設定、認証の設定

ユーザー登録とパスワードの設定を行います。「rootパスワード」とすぐ下の「確認」欄には、rootユーザーのパスワード（同じもの）を入力します。

必要に応じて一般ユーザーの登録も行います。数字のみのユーザー名は、登録できません。（例、326→不可、suzuki3→可）

認証の設定画面では、ネットワーク上のNISサーバで認証を行う場合に「NISを有効にする」をチェックします。詳細についてはネットワーク管理者へ問い合わせてみてください。家庭内で小規模なLANを試してみる際には、設定を変更する必要はありません。

12

パッケージグループの選択

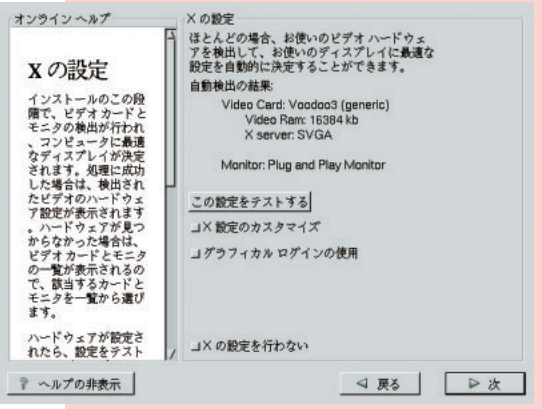
インストールしたいパッケージを選択します。どれを選んでよいのか分からなかったら、「Everything（全部入り）」にしてみましょう。

13

Xの設定

インストーラがビデオカードの自動検出を行った結果が、表示されています。ここで正しく認識されていれば、そのまま進みます。うまく自動認識されていない場合は、下の「Xの設定を行わない」をチェックして先に進み、再起動後にXF86Setupなどを使って設定します。

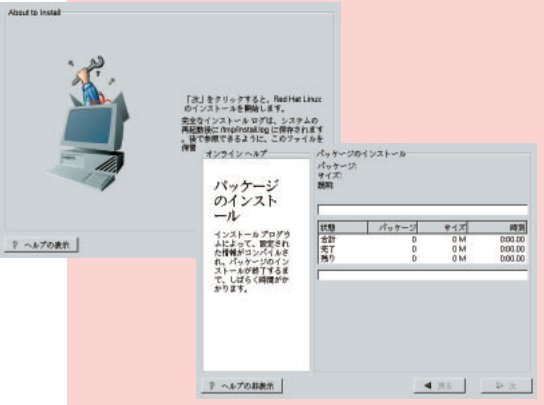
「X設定のカスタマイズ」をチェックすると、次の画面で解像度や色数を指定できます。「グラフィカルログインの使用」をチェックするとマシン起動時にグラフィカルなログイン画面が表示されます。



14

インストール開始

以上でインストールに関する設定は、終わりました。「次」を押すとパッケージのインストールが始まります。少々時間がかかりますが、じっと待ちましょう。進行状況が棒グラフで表示されます。



15

インストール終了

これでインストール完了です。CD-ROMやフロッピーディスクをドライブから抜いて、右下の「終了」ボタンを押すと、マシンが再起動し、Red Hat Linux 6.1 改訂 日本語版が起動します。

TurboLinux 4.5

TurboLinux 4.5は、ターボリナックス ジャパンが公開・配布しているディストリビューションだ。現在のところ、単独のパッケージで販売はされておらず、FTPや雑誌・書籍を通じて配布されている。

また、PFUアクティブラボと東電コンピュータサービスが共同開発した、Linux教育ソフト「Let's Try Linux編」（Windows用）とTurboLinux 4.5を組み合わせて「TurboLinuxラーニングキット」として販売されている。「TurboLinuxラーニングキット」の価格は、4980円。

TurboLinux 4.5は、TurboLinux 4.xシリーズの最新版で、TurboLinux 4.2発売以降にセキュリティホールの解消、バグフィックスなどでアップデートされたパッケージを組み込んでいる。また、それ以外でも更新のあったプログラムについては、新しいパッケージを採用している。

カーネルは、2.2.13、Cライブラリはglibc 2.0.7を採用している。ほかのディストリビューションでは、glibc 2.1.xが用いられることが多いが、TurboLinux 4.xシリーズの一貫性を重視して、あえて2.0.7が用いられている。

またX Window System関係では、デスクトップ環境にGNOME 1.0.54（Octoberリリース）（**画面1**）とKDE 1.1.2（**画面2**）を採用している。ほかにもAfterStep、Window Maker、ICE Window Managerなど多数のウインドウマネージャが収録されているので、好みやマシンスペックに合わせて使い分けられる。

XFree86については、RPMSディレクトリに収録されているのはバージョン3.3.5であるが、別のディレクトリに最新の3.3.6も収録されており、インストール終了後にアップデートすることができる。3.3.6では、ATI Rage128、ATI Rage Mobility、SiS 540／630、SiS 300、Silicon Motion Lynx、S3 Savage2000、NVIDIA GeForce256、Intel i810など最新のアクセラレータに対応しているため、これらを搭載したPCでTurboLinux 4.5を使用するためには、必須のアップデートだ。（Intel i810については、別途カーネルモジュールも追加する必要がある）。

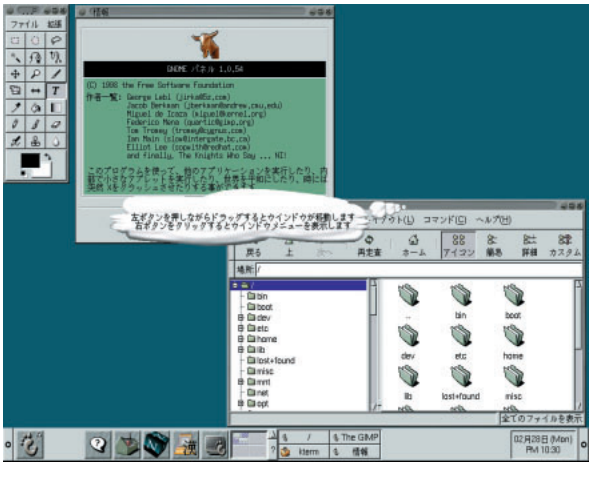
円熟のディストリビューション

TurboLinux 4.5は4.0以来、大規模な改変を加えずに、改良が加えられ続けてきたディストリビューションの最新版だ。動作は非常に安定しているうえに、細部の設定までよく考えられた、円熟のディストリビューションといえよう。

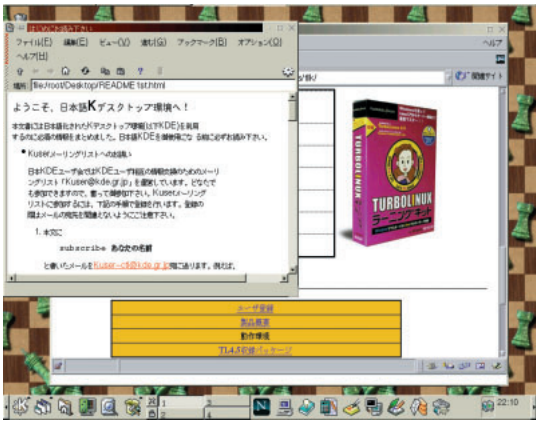
たとえば、日本語表示1つとってみても、デフォルト設定のままで適切なサイズとバランスで表示されるようになっている。スクリーンショットで確認してみて欲しい。

安定した環境を望むユーザーだけでなく、あれこれ手を加えずに、整った環境を手に入れたいユーザーや、初心者に適したディストリビューションといえるだろう。

本誌付録CD-ROM Disk 2に、TurboLinux 4.5を収録しています。インストールに関しては、233ページからの記事を参照してください。非商用のソフトウェアだけが収録されていますので、SystemCommander Liteは含まれていません。また、ターボリナックス ジャパンは、この製品に対するサポートは行っておりません。



画面1 GNOMEデスクトップ
Enlightenmentと組み合わされたGNOME 1.0.54。



画面2 KDEデスクトップ
安定版最新の1.1.2が採用されている。

マイスター Linux Mandrake 6.1

五橋研究所から、マイスター Linux Mandrake 6.1が1月28日より販売されている。Red Hat Linuxをベースとしたフランス生まれのディストリビューションLinux Mandrakeに日本語化キットと商用の日本語フォントを同梱したパッケージとなっており、サポートを行わないことで3840円という低価格を実現している。同社では、オープンソースソフトウェアをいち早く、安価に提供することを目的としたOSパッケージ「マイスター」シリーズを展開しており、本製品がシリーズの第1弾になる。

Linux Mandrakeを日本語化

Linux Mandrakeは、仏MandrakeSoft社の販売するディストリビューションで、欧米では“Better Red Hat”として評判が高い。本製品は、最新版のLinux Mandrake 6.1に日本の有志が作成した日本語化キットをセットにすることで、日本語に対応している。

システムの基本構成は、カーネルのバージョンは2.2.13、標準ライブラリはglibc 2.1.1、XFree86 3.3.5となっている。インストール後に、アップデートキットを使うことで、XFree86 3.3.6をインストールすることも可能だ。また、安定版カーネルの2.2.14、開発版の2.3.36のソースファイルも付属するので、これらをコンパイルして使うこともできる。

標準のデスクトップ環境にはKDE 1.1.2を採用しているが、GNOME 1.0.9を選ぶことも可能だ。日本語化キットを用いることでメニューなどが日本語化される。メニューの日本語は正しく表示されないが、このほかにWindow Makerやicewmなどのウインドウマネージャも利用可能だ。

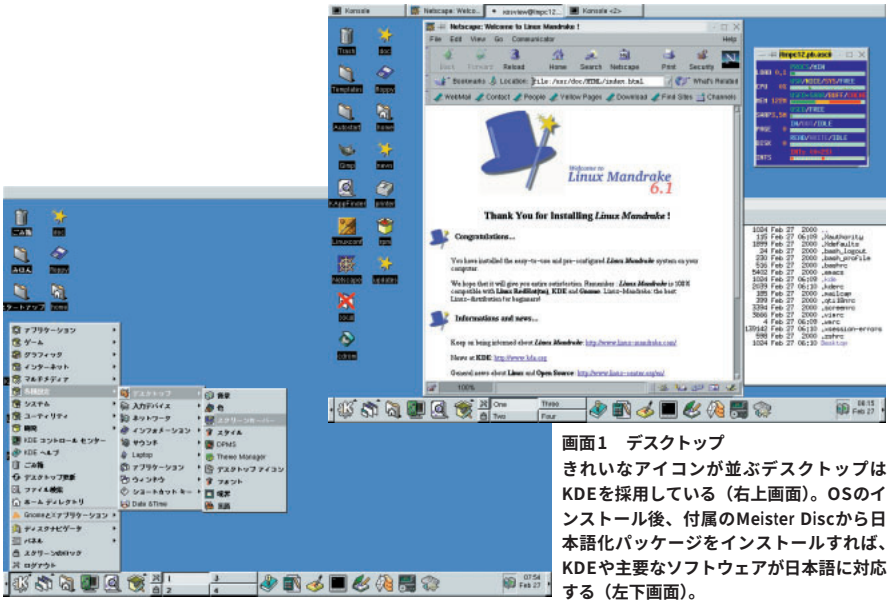
日本語化キットには、DynaLabの商用日本語フォントや、Canna、kon、ktermなども含まれているので、入力や表示など、基本的な日本語利用に対応する。

日／英両対応パッケージ？！

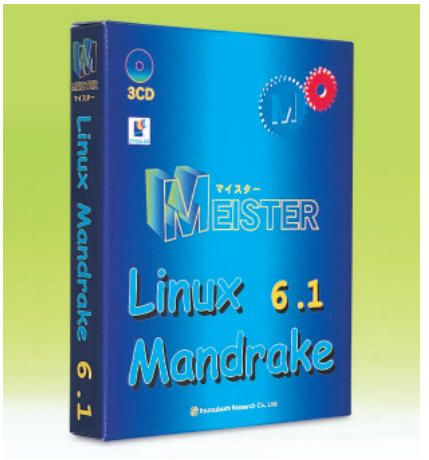
インストーラは、Red Hat Linux 5.2などと同様にテキスト画面で構成され、メッセージは英語で表示される。日本語で書かれたインストールマニュアルが付属しているので、英語が苦手な人でも困ることはないだろう。

日本語を利用するためには、インストール後に日本語化パッケージを適用しなければならず少々面倒だが、サーバ用途など英語版のままでも使いたい場合は逆に便利かもしれない。1台のPCごとにライセンスを購入する必要のないLinuxでは、複数のマシンにサーバやデスクトップクライアントなど、別々のソフトウェア構成でインストールすることも多いだろう。

メジャー系だけど、ひと味違ったディストリビューションを使ってみたい人にお勧めしたい。



画面1　デスクトップ
きれいなアイコンが並ぶデスクトップはKDEを採用している（右上画面）。OSのインストール後、付属のMeister Discから日本語化パッケージをインストールすれば、KDEや主要なソフトウェアが日本語に対応する（左下画面）。

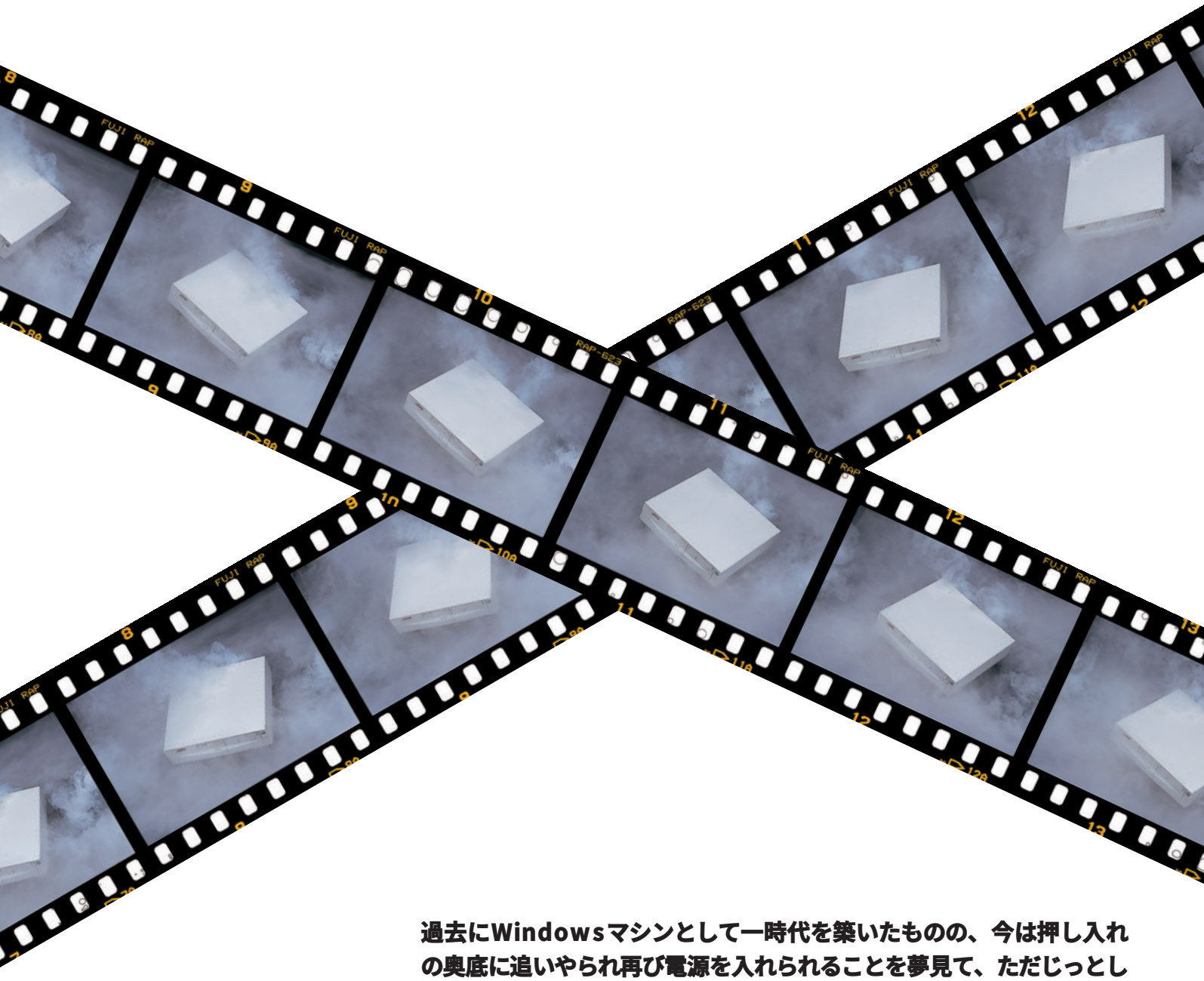


製品名　マイスター Linux Mandrake 6.1
価格　3480円
問い合わせ先　株式会社五橋研究所
03-5818-6608
http://www.cdrom.co.jp/

往年の名機 復活大作戦

往年の名機復活委員会

竹内充彦
山岸典将
Tux Heaven
藍　植緒



過去にWindowsマシンとして一時代を築いたものの、今は押し入れの奥底に追いやられ再び電源を入れられることを夢見て、ただじっとしているだけのマシン達。Windowsマシンとしては物足りなさを露呈し、話題のWindows 2000にいたってはインストールすら拒否されるかもしれない。ああ、やはり捨てられてゆくのか……。　でもあきらめてはいけない！　なにもOSはWindowsだけじゃあない。そう、Linuxがあるじゃないか。これならまだまだいける。いや、りっぱな現役マシンに格上げだ。まさに生き残りをかけた旧型マシン復活大作戦。

古いマシンは本当に使えないのか？

文：編集部　Text : Linux magazine

かつて、マイコンと呼ばれた機械があった。簡単な命令セットとわずかなメモリ空間を持つ8ビットのCPUは、現在のPentium IIIやAthlonといったCPUに比べれば数万分の1くらいの能力しか持っていなかった。しかし、それはそんなに昔の話ではない。最初のマイコンキットが発売されたのはたかだか25年ほど前なのだ。わずか4半世紀の間に、電卓とそう変わらないような機械は、現在のPCとなり、その進化は今なおとどまるところを知らない。最近では、搭載メモリ量は3年で2倍となり、CPUは2年で3倍の能力を持つといわれる。

もちろんこの間には、旧式化とともに使われなくなり、消えていったマシンが数多くある。こうした機械たちも、ある日突然使い物にならなくなって引退したわけではない（故障は別として）。新しくて、より高速なPCにその地位を明け渡し、だんだんと使われなくなってついには消えていったのだ。家庭、職場や学校、どこにでもPCは導入されている現在、あなたのまわりにもそのようなPCがきっとあるだろう。

ちょっと待った！　そのマシンは本当に使い物にならないのだろうか。ハードウェアを強化したり、使い道を工夫すれば、まだまだ現役で働けるのではないだろうか。Linux magazineでは、総勢4名からなる「往年の名機復活委員会」を急きょ組織し、各委員がPCを1台ずつ担当して活用への道を探ることにした。この特集は、崖っぷちに立たされたマシンを復活させるべく立ち上がった4人の男が苦闘する様をつづったドキュメントである。

ちょっと歴史をひもといてみる

16ビットCPUを搭載したマシンが発売されたころからだろうか、マイコンはパーソナルコンピュータと呼ばれるようになっていた。OSの主流はMS-DOSとなり、日本を除く世界中でIBM PC/AT互換機が主流となった。日本では、日本語を表示するために専用のハードウェアを搭載したNEC PC-9800シリーズが広く普及し、一般の企業でも業務に使われはじめた。

Intelから32ビットCPUの80386が発表され、CPUは32ビットで動作するようになったが、依然として16ビットOSのMS-DOSが使われ続けていた。これは、適当な32ビットOSがなかったためである。PCメーカーも、どうせ16ビットOSのMS-DOSを使うなら、高価な80386ではなく、16ビットCPUの80286で十分だと思ったのか、これらのCPUを搭載したマシンが並行して販売されていた。

1990年にMicrosoft Windowsのバージョンが3.0となって386 Enhancedモードが使えるようになると、やっと1Mバイト（！）以上のメモリを自由に使えるようになった。ご存じの通り、GUI環境はCPUに対し、大きな負担となる。そこで、80386と浮動小数点演算コプロセッサ80387の機能とキャッシュメモリを1つのパッケージにまとめ、高速化を図ったi486が登場すると、IBM PC/AT互換機メーカーはこぞってi486搭載モデルを発売した。

このころ、日本語の表示をすべてソフトウェアで行うように拡張されたAT互換機用のDOSが日本アイ・ビー・エムから発売された。これがDOS/Vだ。DOS/Vの登場によって、安価なIBM PC/AT互換機でも日本語を扱えるようになった。一部のパソコンファンの間では、486マシンを個人輸入してベンチマークテストの結果を競い合うのが流行った。これは、日本でもIBM PC/AT互換機が浸透し始めたことを意味している。

実は、Linuxが誕生したのもこのころなのだ。1991年に誕生したLinuxは、最初は80386のアーキテクチャに依存したOSだったが、生まれたときから完全な32ビットOSだった（Windows 3.xは16ビットOSだ）。そして、開発は着実に続けられ、1994年にカーネルのバージョン1.0が発表された。しかし、まだOSとしての知名度は低かった。その一方で、メジャーなOSとなったのがWindowsであることはみなさんもご存じの通りだ。

Windows 95の発売にいたっては、社会現象といわれ、PCを持っていない人までが購入したという噂まで飛びかった。そしてブームがやってきた。Pentium 100MHz～200MHzを搭載し

項目	スペック
CPU	Pentium 133MHz以上
メモリ	32Mバイト以上（64Mバイト以上推奨）
ハードディスク	空き容量850Mバイト以上（全体で2Gバイト以上推奨）

表1　Windows 2000 Professionalの要求スペック

たPCが大量に売れ、世界中でリブート、再インストールが何度も繰り返された。Windows 95が、標準でTCP/IPネットワークをサポートしていたこともあったのだろう。インターネット利用が爆発的に広まったのもこの頃からだ。その後、元気を失っていたMacintoshがiMacで人気を回復するなど、熱狂的なWindowsブームは去るが、後継バージョンのWindows 98はPCの標準OSともいえる位置を占めている。

Windows 2000おめでとう

2000年問題騒ぎも治まり（問題はまだ終わってないんですけどね）、2月18日にWindows 2000が発売された。Windows 2000は、Windows NTの流れを汲むOSであり、Windows 95／98とは違った位置づけの製品だが、デスクトップ用途においてもその人気は高いようだ。

さて、Windows 2000は、要求するマシンスペックものなかなか厳しい。デスクトップ向け製品のWindows 2000 Professionalが必要とするマシンスペックを**表1**に示す。

486マシンで使うのは自殺行為だ。また、Windows 95とともに出荷されたPentiumマシンでも快適に利用するのは難しいだろう。Windows 98も、Pentium 133MHz、メモリ32Mバイトのマシンで利用するとかなりイライラする。そんなときはLinuxをインストールしてみるとよいかもしれない。X Window Systemを使わなければ486マシンでも軽快に動作するし、PentiumマシンのCPUやハードディスクを強化してWindowsとともに使ってもよいだろう（もちろんLinuxのみで使っても構わない）。これでこの特集の意図がおわかりいただけたであろう。

ターゲットとするPC

今回の特集では、ハードウェアの拡張が難しいメーカー製マシンをターゲットとした。Windows 95世代のPentiumマシンとして、FMV DESKPOWER 5100D5、Compaq PRESARIO 7242を取り上げ、ハードウェアの拡張で快適なデスクトップLinuxマシンに仕立てる。また、486マシンであるIBM Aptiva 530は、拡張のためのハードウェアが入手困難となっているため、Linuxでの有効な使い方を探る。さらに、PC-98アーキテクチャのマシンにもLinuxをインストールし、その可能性を追求した。

みなさんの手元に眠れるマシンがあるなら、**図1**のフローチャートで特集の読みどころを確認して、今後の参考にしてはいかがだろうか。

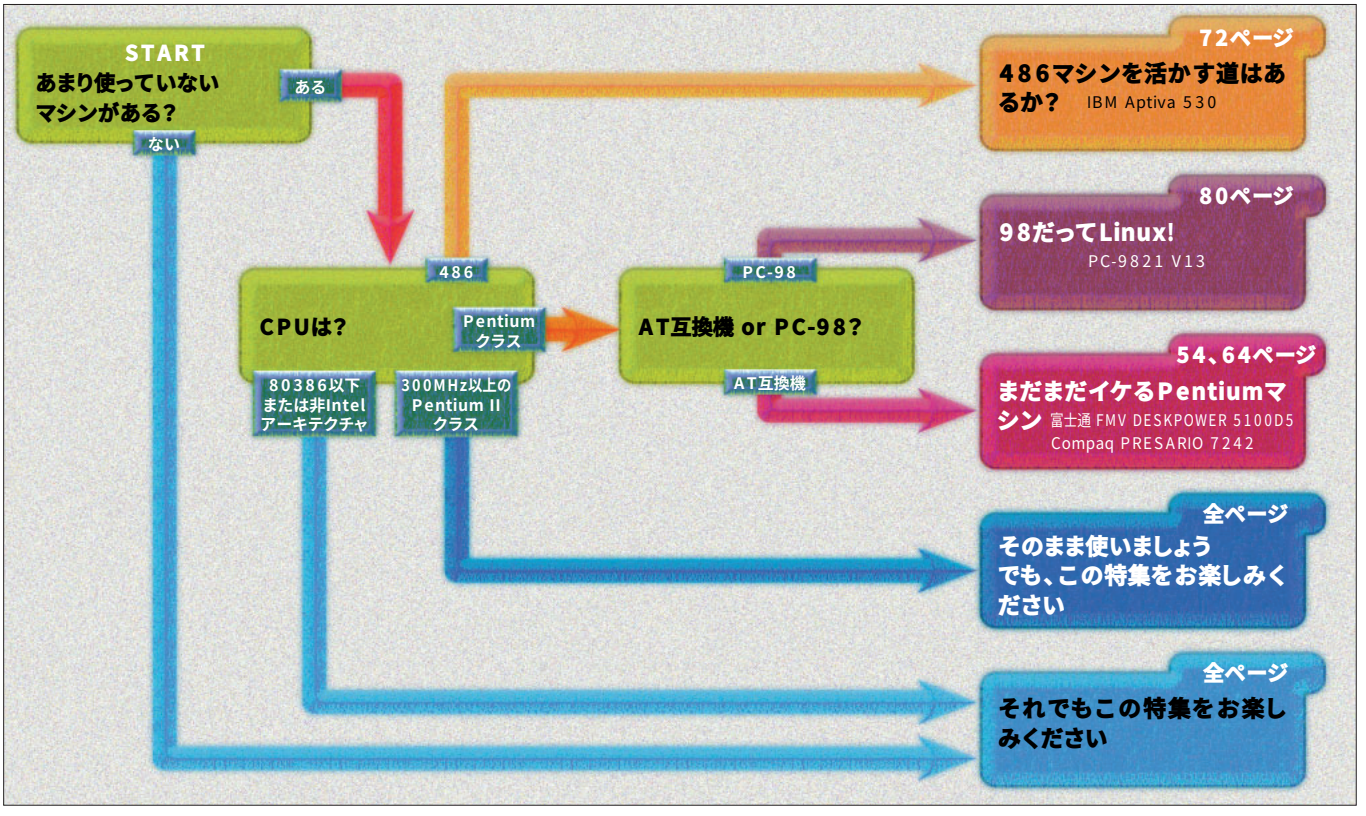


図1　この特集の読み方チャート

FMV DESKPOWER 5100D5

文：竹内 充彦 Text : Michihiko Takeuchi

X ワークステーションの構築

ここでは、現役を退いたPCを引っ張り出してきて、少ない投資によるハードウェアの増強と、LASER5 Linux 6.0 Rel.2のインストールにより、X Window Systemを使うワークステーションとして再利用してみたい。と、立派なことを言ってみるが、要するに、Linux自体をいじり倒したり、GNOME対応アプリケーションを使ってみたり、ゲームをしたり（これが一番パワーを要求するけど）と、一通り遊べるマシンになったらいいなぁ、というのが本音である。

編集部の協力を得て、倉庫に眠る数々の旧型PCの中から選ばれたのは、一世を風靡したベストセラー機である富士通FMV DESKPOWER 5100D5である（**写真1**）。ポピュラーな機種なら問題ないでしょ！？ とお気楽に選んでしまった…。

さて、LASER5 Linux 6.0 Rel.2が要求するハードウェア仕様は**表1**のようになっている。まずは、これに照らして、対象機を調べてみることにしよう。

写真1 FMV DESKPOWER 5100D5
Pentium 100MHz搭載のベストセラー機。あなたの身近にも眠っているのでは？

対象機のスペック

同機の基本的な仕様については、**表2**を参照してほしい。同機は、すでに現役を退き、会社の倉庫に保管されていたのだが、私の手元に来たときには、Windows 95がインストールされたままだった。

会社で利用されていたこともあり、後から追加されたと思われる3Com社のネットワークカード、EtherLink III（3C509TP）が装着されいた。電源を入れてみると、無事起動する。しかし、相当に使い込まれていたのか、CPU冷却用ファンなどは、綿埃が溜まっていたせいで滑らかに回らなくなっていた。掃除機でケース内の埃を取り、点検しながら、各部を見ていくことにしよう。わくわく。

■CPU

CPUソケットはSocket7で、搭載されているCPUはPentium 100MHzだ（**写真2**）。FSB 66MHzで、内部クロック倍率を1.5倍にして動作している。このCPUは、いわゆるP54Cと呼ばれるタイプで、CPUコアとI／Oに供給する電圧とも3.5Vというものだ。Pentiumは、後にP55Cと呼ばれるタイプが登場するが、そちらは、コア電圧とI／O電圧が異なり、コア電圧2.8V、I／O電圧3.3Vというものである。CPU交換の際にはこの点にも注意しておきたい。このマシンのマザーボード上にはP55C用に供給電圧を切り替えるジャンパと、内部クロック倍率を切り替えるジャンパも備わっている（**写真3**）。ジャンパ設定により、200MHz（66MHz×3）のP55Cまでは搭載可能であることがわかる。

項目	仕様
コンピュータ本体	i486以上および互換性のあるCPUを搭載したPC/AT互換機
メモリ	16Mバイト以上のRAM（X Window Systemを使用する場合32Mバイト以上）
ハードディスク	フルインストール時　約1.3Gバイト必要
ビデオカード	XFree86 3.3.5がサポートするビデオカード、ディスプレイ
周辺機器	CD-ROMドライブ（ATAPI／SCSI）、3.5インチFDD（1.44Mバイト）ドライブ、キーボード、マウス

表1 LASER5 Linux 6.0 Rel.2が要求するハードウェア仕様

項目	仕様
CPU	Pentium 100MHz（Socket7）
ハードディスク	850Mバイト（E-IDE）
CD-ROMドライブ	東芝製4倍速 AXM-5302TA（E-IDE）
メモリ	8Mバイト（4スロット中2スロット使用）
ビデオ	ATI TurboPro PCI Mach64CT（オンボードPCI接続）
拡張スロット	PCI／ISA×2、ISA×1
モデム	富士通FAX Voiceモデム（オンボード）
サウンド	Sound Blaster16 PnP（ISA×1使用）
電源容量	145W

表2 FMV DESKPOWER 5100D5の仕様

■メモリ

メモリスロットは、72ピンSIMMスロットが4本搭載されているが、4Mバイト SIMMが2枚（計8Mバイト）装着されている（**写真4**）。LASER5 Linux 6.0 Rel.2の要求するハードウェア仕様では、メモリは16Mバイト以上とされている。ましてX Window Systemを使うワークステーションを目指す場合は、32Mバイト以上必要ということなので、ここは増設を余儀なくされる。幸い、空きスロットが2本あるので増設は可能だろう。

■大容量ストレージ

大容量ストレージデバイスのインターフェイスには、E-IDEを2系統搭載しており、そこに850MバイトのハードディスクとCD-ROMドライブが各1台接続されている。筐体には、あと1台ハードディスクを搭載できるスペースがある。LASER5 Linux 6.0 Rel.2は、フルインストール時に1.3Gバイト必要とのことなので、850Mバイトでは、少し心もとない。容量の大きいハードディスクを増設すればよいのだが、いっそのこと、インターフェイスとハードディスクを共に増設するという手もある。これだと、Ultra ATA/33や66といった、より高速なデータ転送を実現するIDEインターフェイスカードを選ぶことができるかもしれない。

■グラフィックス

グラフィックスチップもマザーボード上に実装されており、ATI社のMach64チップに2Mバイトのグラフィックスメモリを搭載している（**写真5**）。Mach64はXFree86でも動作が確認されているチップであり、アクションゲームをプレイするわけでなければ、十分なスペックとも思える。しかし、グラフィックスメモリ2Mバイトだと、表示能力は1024×768×16ビットカラーにしかならない。ここでは、あえてグラフィックスカードの増設にもチャレンジしてみたい。

■サウンド

同機はサウンドカードにSound Blaster 16 PnPを搭載している（**写真6**）。このカードは定番といってもよく、Linuxとは最も相性の良いカードだろう。もちろん、同時発音数や、音源、3Dサウンドなど、最新のサウンドカードにはかなわない。とはいえ、サウンドは鳴るので、チューンアップの優先度は低い。

■ネットワーク

冒頭でも触れたが、同機には後付けと思われる、3Com社のネットワークカード EtherLink III（3C509TP）がISAスロットに装着されていた（**写真7**）。同カードは、10BASE5と10BASE-Tのコネクタを装備しており、どうやら10BASE-Tで接続されていたようである。このカードも定番である。

現在では、より転送速度の速い100BASE-TX対応のネットワークカードが安価で市販されている。しかし、筆者が持つLAN環境は、未だ10BASE-Tのままなので、これはそのまま使わせてもらうことにしよう。しめしめ。

■拡張スロット

チューンアップと言えば、拡張スロ

写真4　SIMMスロットは4本
うち2本に4MバイトSIMMが装着されているので2本空きがある。

写真5　マザーボード上のMach64チップ
XFree86での動作も確認されており安心。グラフィックスメモリ2Mバイトは、ちょっと頼りない。

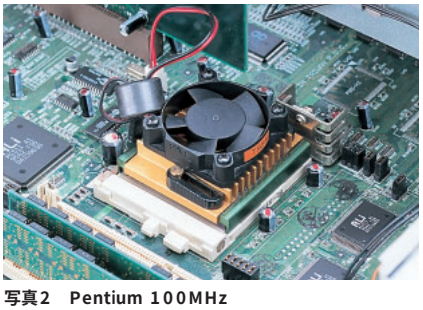


写真2　Pentium 100MHz
FSB 66MHzの1.5倍速。CPUコア、I／Oとも3.5Vで駆動する。

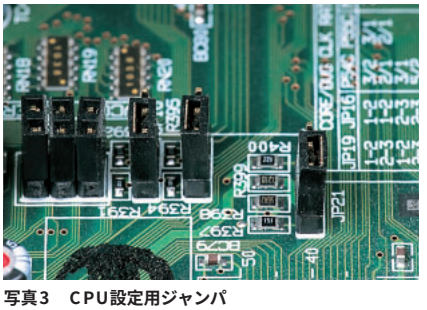


写真3　CPU設定用ジャンパ
P55C用に供給電圧を切り替えるジャンパと、内部クロック倍率を切り替えるジャンパ。

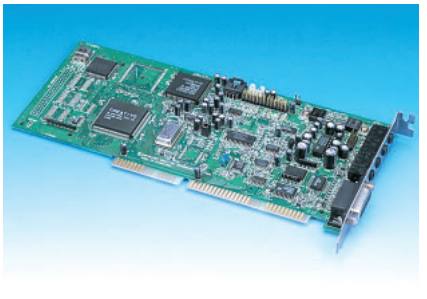


写真6　定番サウンドカードSound Blaster 16 PnP
Linuxでのサポートは万全といってもよく、安心材料である

ットについても確認しておきたい。同機の拡張スロットは、ライザカードで提供されており、ISA×1、PCI／ISA共用×2となっている。しかし、サウンドカードとネットワークカードでISAスロットを2つ消費している。つまり、あと1枚しかカードを挿せないのだ。悩ましいところだ（**写真8**）。

■その他

同機には、14.4kbpsのモデムも内蔵されている。これは、専用のコネクタを介して接続されているため拡張スロットは消費していない。しかし、今や14.4kbpsでは電話代が厳しい。幸い、筆者の環境では、ISDNダイヤルアップルータが稼動しているので、LAN経由でインターネットに接続することにして、モデムには手を入れないことにする。

マザーボード上には、CPU外部キャッシュ（L2キャッシュ）が256Kバイト搭載されている。非同期SRAMで、マザーボードに直付けされており、これを拡張することはできない（こともないだろうが、あんな細いピンをハンダ付けする技量は持ち合わせない）。

少ない投資で大きな成果

さて、ざっと対象機種を眺めてみたわけだが、ここで、チューンアップの目星をつけておこう。以下のようにしたい。

●メモリ増設
●CPU換装
●ハードディスク増設
●グラフィックスカード増設

なぜこの4点かと言えば、パフォーマンスの向上を体感できそうだからである。PCIスロットが1つしか余っていないということもあり、グラフィックスカードの増設に重きを置いたが、HDD増設については速度の追求のためUltra ATA/66カードも試してみたい。

では、これらの各項目について順に見ていこう。

メモリ増設
費用：1万6900円

先ほど、増設が必須と闇雲に言い切ってしまったが、本当だろうか？ 実は、LASER5 Linux6.0 Rel.2の動作条件には満たないことを知りつつ、増設する前に、メモリ8Mバイトのままでインストールを試みた。インストーラによる作業自体は、途中まで順調に進む。インストールタイプは、ワークステーション、サーバとも、ディスク容量不足でダメだったので、カスタムを選んで、パッケージを個別に指定する。しかし、X Window Systemの設定を終え、実際にXを起動しようとすると、スラッシングが発生して、全然進まなくなる。つまりXの正しいインストールができないのだ。

用途によっては8Mバイトでも使えないことはないかもしれないが、本項の目的は達成できない。やはり増設するしかない。

■メモリ選び

まず知っておきたいのは装着の単位である。72ピンSIMMの場合、外部からのアクセスバス幅が32ビットとなるため、Pentiumのような64ビット幅でメモリにアクセスするCPUでは、同容量のSIMMを2枚単位で装着しなければならない。つまり、32Mバイト増設するなら16Mバイト SIMM×2枚、64Mバイト増設するなら32Mバイト SIMM×2枚を用意しなければならないのだ。

装着可能なメモリはFPM（Fast Page Mode）タイプか、EDO（Extended Data Out）タイプと呼ばれるDRAMだ。EDOタイプのほうがFPMよりも高速にアクセスできる。しかし、このマシンは生産時期によってはEDOタイプが利用できないものもあるようなので、FPMのほうが無難である。なお、現在主流のSDRAM（Synchronous DRAM）は使えないので注意が必要だ（もっとも、SDRAMはほとんどが168ピンDIMMで、72ピンSIMMなど売っていないので、間違える心配はないかもしれないが）。

SIMMには、パリティあり／なしの種別がある。メモリ内容の整合性を保つため、パリティ情報を保持するものと、そうでないものだ。同機は、BIOS設定を変えることで、どちらにも対応可能である。しかし、すでに装着されているメモリがパリティなしなので、

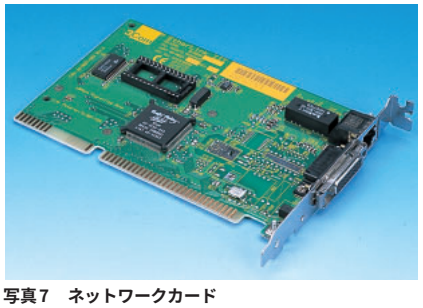


写真7　ネットワークカード
定番ネットワークカード3Com EtherLink III。ISDNダイヤアップルータが10BASE-T対応なのでこれで十分だ。

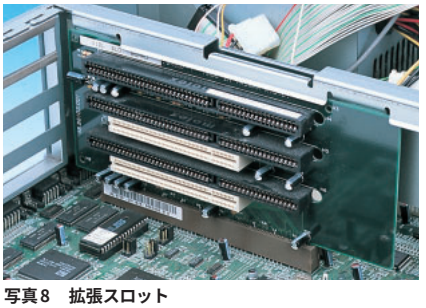


写真8　拡張スロット
ライザーカード上の拡張スロットはISA×1、PCI／ISA共用×2だ。しかし、空きは、PCI／ISA×1しかないのが悩ましい。

増設するならばパリティなしでかまわない。

さらに、SIMM購入時に、メモリの読み出し速度が選択できる。現在売られているSIMMは多くは60ns（ナノ秒）のものだが、70nsのもの売られている。もちろん60nsのほうが高速な読み出しが可能であるが、このマシンの場合、さほど気にすることではなく、70nsでもかまわない。

ここでは、32Mバイト FPM パリティなし 60ns×2枚を選んでみたい（**写真9**）。近所のショップでの店頭価格は1万6900円だったが、ちょうど編集部に同タイプのメモリがあったので、それを流用させてもらうことにした（ラッキー！）。

8Mバイトに64Mバイト増設したので、合計72Mバイトになった。これなら余裕だ（**写真10**）。

■独断的お勧め度！

メモリの増設はお勧めも何もない。必須のアイテムである。しかし、値段については、納得がいかない。メモリの価格は下落傾向にあるというが、それは最新のSDRAMの話である。72ピンのFPMについては、極端な価格の上下動はないだろう。中古パーツで探すとか、知人から譲り受けるとか、もしも安く調達できる方法があるなら、お勧め度は星4つなのだが…。

メモリ増設お勧め度！
★★★☆

CPUチューン
費用：2万6800円

本誌3月号の最新CPUの記事を読むと、やれ、FSB 133MHzだ、CPU内部800MHzだのと、豪快な数値が目に飛び込む。すでに発表会での展示レベルでは1.5GHzで駆動するCPUも登場しているという。いっぽう同機は、FSB 66MHzの Pentium 100MHzである。圧倒的に非力だ。もっとも引退して倉庫に眠っていたものを、最新のものや、まして開発段階のCPUと比べようということに土台無理があるか…。しかし、いまや、激安PCでも400～500MHzのCPUを搭載しているのが常識だ。せめて、そこまでなら、何とかならないものだろうか。

■CPU選び

同機が搭載しているCPUソケットはSocket7である。Socket7で利用できるCPUの選択肢は豊富で、Intel社のPentium以外にも、AMD社のK6シリーズ、Cyrix社（現VIA傘下）／IBM社の6x86（M I）やM II、IDT社（現VIA傘下）のWinChip C6など、さまざまなものがある。しかし同機のマザーボードは、内部クロック倍率の設定信号や、供給電圧の問題から、実際に利用できるCPUの種類は限定されてしまうのだ。マザーボード上での設定では、P55Cの200MHzというのが最速だが、今どきP55Cの200MHzのCPUなんて、中古パーツ屋さんを回らないと手に入らない。選択肢として、WinChip C6というのもあるが、いずれにせよ、スピードは最近の激安PCの足元にも及ばないだろう。

そこで登場するのが、CPUアップグレードキットだ。正しい電圧を供給するレギュレータと内部クロック倍率信号を供給する回路を装着した（ゲタを履かせた）CPUである。もちろん、発熱量に応じたCPU冷却ファンも装着されている。これを使うと、マザーボードの設定可能項目に関係なく高速なCPUを利用することができるのだ。また、メルコやアイ・オー・データ機器といった、実績あるメーカーの製品を購入すれば、動作確認機種なども明記されているため、安心して購入できる。

ここでは、このマシンも動作確認機種に含まれている、メルコの「HK6-ms V2」を選んでみた（**写真11**）。

「HK6-ms V2」は、CPUにAMD社のK6-IIIを搭載しており、FSBの6倍のクロックで動作するようになっている。つまり、FSB 66MHzのこのマシンに装着すれば、400MHzで動作することになるのだ。これなら、Celeron 400MHzを搭載した激安PCにもクロック周波数で見劣りはしない（FSBも同じ66MHzだし）。しかも、内部キャッシュの容量についてはCeleronを上回っており、L1キャッシュ64Kバイト（Celeronは32Kバイト）、L2キャッシュ256Kバイト（Celeronは128Kバイト）となっている。さらに、マザーボード上のキャッシュ（ここでは256Kバイト）をL3キャッシュとして利用可能になる

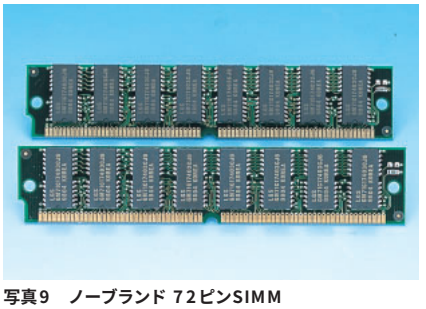


写真9　ノーブランド 72ピンSIMM
32Mバイト、パリティなし、60ns。店頭価格は2枚で1万6900円だが、今回は編集部のものを流用。

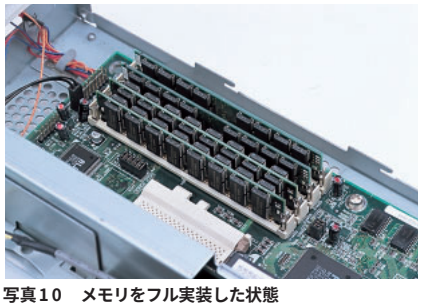


写真10　メモリをフル実装した状態
装着は斜めに挿してから、垂直に立てるようにする。

のだ（Celeron用のマザーボードにはキャッシュは実装されていない）。少なくとも同クロックのCeleronマシンを凌駕するのではなかろうか。いやがうえにも期待が高まるというものだ。

■換装作業

まず、マザーボードからPentium 100MHzを外そう。ソケットの横に付いているレバーを引き上げ、CPUを抜く。次にHK6-ms V2をソケットに挿す。この時、ピンの配列に注意しよう。本体を正面から見たとき、ちょうどレギュレータ基盤部が左側にくるように挿さるはずだ。正しく挿せたら、レバーを戻し固定する。次に電源コネクタを接続する。これを接続しないと、冷却ファンだけでなく、CPU自体に電気が供給されないので、PCは起動しない。

さて、換装したら、電源を入れてみる。CPUをHK6-ms V2に換装しても、システムは何ごともなかったかのようにブートする。ところで、LASER5 Linuxをインストールしただけでは、CPUのコンフィグレーションは386のままになっている。たとえ、そのPCがK6マシンだろうが、はたまたPentium IIIマシンだろうが変わりはない。最低限のスペックに合わせて、i386であるという前提のもとにインストールされるのだ。せっかくCPUも速くなったのだから、ここでCPUに最適化したカーネルを構築しておこう（本当は、Pentium 100MHzの状態でも、やっておけばよかったのだが…）。

/usr/src/linuxに移動し、次のコマンドで「Linux Kernel Configuration」を起動する。

```
# make distclean; make xconfig
```

「Linux Kernel Configuration」が起動したら、［Processor type and features］をクリックして、一番上の［Processor family］の欄の［386］と書かれたボタンをクリックし、メニューから［Pentium/K6/TSC］を選ぶ（**画面1**）。メインメニューに戻り、結果を保存して「Linux Kernel Configuration」を終了する。

コンパイル作業に取り掛かろう。

```
# make dep; make clean
# make bzImage
# make modules
# make modules_install
```

カーネルが/usr/src/linux/arch/i386/boot/bzImageという名前で作成されているので、それを、/bootに新しい名前でコピーしよう。まだディレクトリ/usr/src/linuxにいるのであれば、以下のようにする。

```
# cp arch/i386/boot/bzImage /boot/
vmlinuz-2.2.5-k6
```

そうしたら、/etc/lilo.confを**リスト1**のように書き換えて、/sbin/liloを実行し、システムを再起動する。

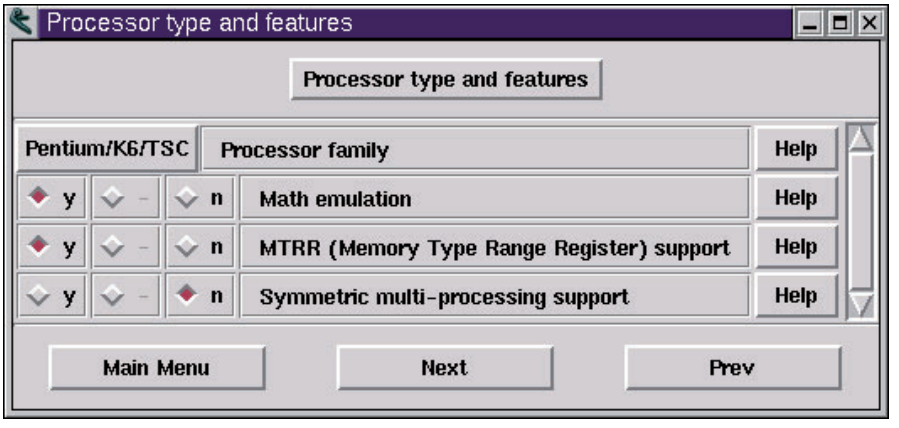
■独断的お勧め度！

さすがにCPU速度の向上は、カーネルの再構築時のコンパイル時間などでも十分に体感できる。本当に激安PCに追いついたかどうかは微妙なところだが。メモリの価格に比べれば、CPUアップグレードキットの価格は納得がいくものだ。また設定の手軽さという点についても評価できる。したがって、満点の星4つ！

メルコでは、同じくK6-IIIを使ったアップグレードキットで、500MHzで動作する製品も発売予定だそうなので、そちらにも期待したい。

CPU換装お勧め度！
★★★★

写真11　メルコ HK6-ms V2
K6-III搭載のCPUアップグレードキット。FSB66MHz時は400MHzで動作する。購入価格2万6600円



画面1　［Processor type and features］で［Pentium/K6/TSC］を選択する。

ハードディスクチューン
費用：1万5000円

このマシンのハードディスク容量は850Mバイトである。現在ショップで売られているハードディスクの容量は、そのほとんどが数G〜数十Gバイトである。もはや、桁が違う。実際、850Mバイトのハードディスクだと、LASER 5 Linux 6.0 Rel.2のお勧めインストールパッケージである「サーバ」「ワークステーション」ともに、容量不足でインストールすることができない。ここでは、ハードディスクの増設に挑戦したい。

さて、冒頭で調べた通り、このマシンには、E-IDEインターフェイスが2系統備わっている。それぞれにディスクを2台ずつ接続できるわけだ。そして、現状では、ハードディスク×1台とCD-ROMドライブ×1台の計2台が接続されている。接続台数でいえば、あと2台は接続可能だ。さらに、筐体には、3.5インチのドライブベイが1つ空いているため、IDE接続タイプのハードディスクを1台買ってくれば、すぐにでも増設できるのだ。

▼リスト1　lilo.conf

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux-k6
image=/boot/vmlinuz-2.2.5-22lv3
        label=linux
        root=/dev/hda1
        read-only
image=/boot/vmlinuz-2.2.5-k6
        label=linux-k6
        root=/dev/hda1
        read-only
```

■ハードディスクとIDEカード選び

ここでは、15Gバイトの容量を持つハードディスク、IBM DPTA351500を増設することにする（**写真12**）。実は、このハードディスク、Ultra ATA/66対応でもある。せっかくだから、Ultra ATA/66インターフェイスカードを試してみよう。Ultra ATA/66インターフェイスカードといえば、本誌1月号でもPromise Technology社のカードを扱っている。そこで、ここでは別の製品、I will社のSIDE-Pro66を選んでみた（**写真13**）。これで、ハードディスクのデータ転送速度も格段に向上するはず！？

■まずはハードディスクの増設

まずは、マシンのE-IDEインターフェイスにハードディスクをつないで、LASER5 Linuxをインストールしよう。ハードディスクをベイに固定して、IDEケーブルを接続する。ここでは、既存の850Mバイトのハードディスクから外したケーブルを、そのままDPTA 351500に接続してインストールを行った。

旧型のマザーボードでは、BIOSが原因で、大容量ハードディスクを認識しないという問題が生じることがある。若干懸念していたのだが、このマシンでは問題はなかった。Disk Druidで、いとも簡単に認識し、フォーマットもできた。ただし、Linuxのブートパーティションに、15Gバイト（実際には、swapパーティション取得後の残りだが）を丸ごと確保しようとすると怒られるので、約5Gバイトを/に、残りを/usrに割り振った。

さすがにこれだけ容量があると、「サーバー」も「ワークステーション」も問題なくインストールできる。ちなみに、「カスタム」を選び「すべてを選択」にした。ハードディスクのアクセス速度自体が速いためか、多くのパッケージを選んでいるにもかかわらず、体感的にはインストールも速い。

■Ultra ATA/66カードの増設

ハードディスクへのインストールが終わったら、いよいよUltra ATA/66による高速データ転送である。SIDE-Pro66が採用しているHighPoint Technology製のHPT366というチップは、最近では、440BXチップセットを使ったマザーボードに、Ultra ATA/66をオンボード実装するために搭載されているのを見かける（**写真14**）。さっそくHighPoint Technology社のWebサイトへアクセスしてみると、Linux用のドライバがパッチで提供されている。本誌1月号で、Promise

写真12　IBM DPTA351500
容量15Gバイト。Ultra ATA/66対応。購入価格1万5000円

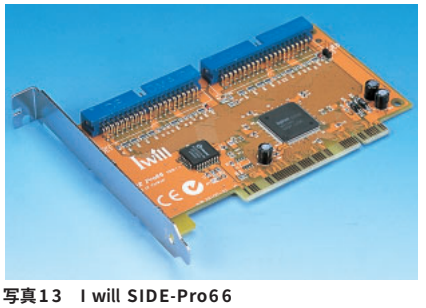


写真13　I will SIDE-Pro66
HPT366チップ搭載のUltra ATA/66カード。購入価格3980円

Technology社のUltra66をインストールしたのと同じUnified IDEドライバである。

しかし、LASER5 Linuxで採用しているkernel 2.2.5に対応したパッチが見当たらない。かといって、バージョンを下げるのも悔しい。ここでは、大胆にも開発版であるkernel 2.3.41にバージョンアップしてしまうことにした。何ゆえ2.3.41という半端なバージョンなのかと問われれば、それは、UIDEドライバのパッチのバージョンに合わせただけである。

さっそく、The Linux Kernel Archivesからkernel 2.3.41をダウンロードする。ソースはtar.gz形式でも17Mバイトあるので、64kbpsでダウンロードしても数十分かかる。ちょっと覚悟が必要だ。そして、HighPoint Technology社のサイトからUnified IDEパッチ（ide.2.3.41-2.200000124.patch.gz）もダウンロードする。

ダウンロードしたらそれぞれ、以下のように展開して、パッチを当てる（/tmpにダウンロードした場合）。

```
# cd /tmp
# tar xvzf linux-2.3.41.tar.gz
# gunzip ide.2.3.41-2.200000124.
patch.gz
# patch -p0 < ./ide.2.3.41-2.
200000124.patch
```

写真14　HighPoint Technology社のHPT366チップ
ABITのマザーボードにもオンボード搭載されているメジャーなチップだ。

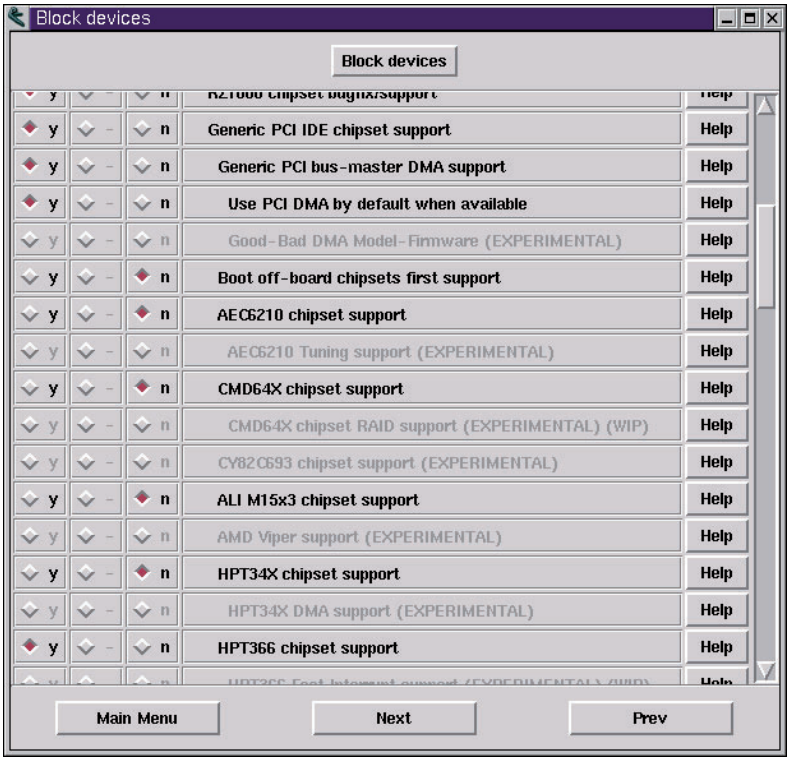
パッチを当て終わったら、make xconfigで設定を変更しよう。[Block devices]をクリックして、[Generic PCI IDE chipset support]で[y]を選ぶ。すると、リストの下のほうにある[HPT366 chipset support]が選択可能になるので、[y]を選ぼう（**画面2**）。それ以外にも、現在搭載しているハードウェアに関する設定を行っておく必要がある。CPU換装の項で行ったK6の設定もそうだが、[General setup]でAPMの設定を、[Network devices]－[Ethernet（10 or 100Mbit）]で3C509の設定を、[Sound]でSound Blasterの設定をしておかないと、機能しなくなるので注意しよう。

設定を変更したら、makeを使ってカーネルをコンパイルする。コンパイル手順はCPU換装の時と同じだ。そうして、Ultra ATA/66であることをlinuxに認識させるためlilo.confのブートイメージのセクションにappend行を追加し（**リスト2**）、/sbin/liloで書き込む。

■大ショック！ BIOSチェックでハング

さて、まずは、カーネルを入れ替えて、カードを認識させて正しくディスクが読めるかどうか試してみよう。SIDE-Pro66に付属のUltra ATA/66用のケーブルでカードとハードディスクを接続する。Ultra ATA/66用のケーブルは、コネクタこそ、通常のIDEケーブルと変わらないが、ケーブルの芯数が80ピンとなっている（**写真15**）。

電源を入れる。ここで異変が…。なんと、BIOSチェックで、SIDE-Pro66をうまく認識してくれない。具体的には、1回目は認識するが、システムを終了し、2回目以降のブートでは、ハングアップしてしまうのだ。試しにカードを外し、ブートし、終了する。もう1度カードを挿す。やはり、1回目は認識するが、終了後、つまり2回目以



画面2　[Generic PCI IDE chipset support]で[y]を選ぶと、[HPT366 chipset support]が選択可能になる。

降はカードを正しく認識しない。その繰り返しである。

インターネットでFMV DESKPOWER 5100D5の情報を集めると、初期出荷状態のBIOSに問題があり、一部のカードが使えないことがあるという。試しに、BIOSのアップデート（富士通は非公認）をしてみたが、やはりダメであった。う〜む、ここまで来ながら悔しい限りだ。

■独断的お勧め度！

Ultra ATA/66による速度的なチューニングは断念したが、大容量のハードディスク増設は、効果絶大である。

ハードディスク増設お勧め度！
★★☆☆

グラフィックスチューン
費用：4000円

Linuxでも、Windowsに負けない3Dアクションゲームが続々と登場している。このマシンに搭載されているATI社のMach64CTチップは残念ながらハードウェア3Dアクセラレーション機能が備わっていないため、こうしたゲームを楽しむためには、どうしても不利である。

また、2MバイトのVRAMを搭載しているが、これだと最大表示能力は1024×768×16ビットカラーになってしまい、デジカメで撮るような大きな画像を扱ったり、複数のウィンドウを同時に開く場合には頼りない。

そこで、PCIバスながら、3Dアクセラレーション機能を備え、グラフィックスメモリを多く備えたグラフィックスカードを増設してみたい。

■グラフィックスカード選び

グラフィックスカードは選びがいがある。いくらAGPスロット対応のカードが主流だとはいえ、まだまだ最新のグラフィックチップを搭載したPCIスロット対応のカードも販売されている。Microsoft Windowsのマルチディスプレイ対応もこれを助けているかもしれない。しかし、あまりに最新のグラフィックスチップを選ぶと、XFree86のドライバが対応していないかもしれないので危険である。またこのマシンは、グラフィックスカード増設時に注意が必要で、S3社のViRGE系のカード以外の増設は難しいという報告もある。

そこで、選んだのは、枯れてはいるが安心感のあるS3社のViRGE/GX2を搭載したカードだ。激安のバルク品である。グラフィックスメモリにSGRAMを4Mバイト搭載している。Quakeのような激しいゲームは難しいが、そこそこイケるのではないかと期待している。

■グラフィックスカードの増設

まず、念のために、システム起動時にXを起動しないように設定変更しておこう。システムメニューから［システム］−［linuxconf］を選択し、［デフォルトのブートモード］を選択する。［デフォルトのブートモードの設定］で［テキストモード＆ネットワーク］を選択する（**画面3**）。［了解］ボタンをクリックすると変更を実行するか問い合わせてくるので、［実際に変更分を実行する］ボタンをクリックしてから［閉じる］ボタンをクリックする。そうしたら、システムをシャットダウンしよう。

このマシンでは、PCIスロットにグラフィックスカードを挿すと、マザーボード上のMach64が自動的に無効になり、新しいグラフィックスカードが有効になる。カードを挿し、ディスプレイのコネクタをつなぎ替えて電源を入れる。ブート中に新しいカードを検出し、Xconfiguratorがドライバのありかをたずねてくるので、LASER 5 Linux 6.0 Rel.2のCDをCD-ROMドライブに入れて、ディレクトリ/mnt/cdrom/LASER5/RPMSの下にある、XFree86-S3-3.3.5-1.5LL6.i386.rpmを指定する。ディスプレイの種類を一覧から選び、使用する解像度を選べば、設定は終了だ。ブート後、startxでXを起動する。

▼リスト2：Ultra ATA/66用にappend行を追加したlilo.conf（抜粋）

```
image=/boot/vmlinuz-2.3.41-ata66
        label=linux-ata66
        root=/dev/hda1
        read-only
        append "ide2=ata,ide3=ata"
```

この行を追加する

Webサイト	URL
HighPoint Technology社	http://www.highpoint-tech.com/
The Linux Kernel Archives	http://www.kernel.org/

表3　ドライバおよびカーネル入手先URL

写真15　Ultra ATA/66ケーブル
隣にある通常のE-IDEケーブルと比べると、ケーブルの芯が細く、数が多いのがわかるだろうか。

ブート時に設定できなくても、ブート後に、コマンドラインからXconfiguratorを起動すれば設定できる。このとき事前にkonを起動しておくと日本語版が起動するのでわかりやすい。グラフィックスチップは自動的に認識してくれる（**画面4**）。

■表示が乱れGXは断念

しかし、ここで問題発生。Xは起動するのだが、画像が乱れる。いろいろ試してみたが、どうにもならない。このカードに特有の問題らしく、X終了後のテキスト画面の乱れなどは、インターネット上にも報告されている。そこで、急遽、手持ちのDIAMOND Stealth 3D 2000（86C325）を持ち出すことにした（4年前の製品）。増設手順は前記と同様。あっさり認識される。これにより、1280×1024×16ビットカラーの表示能力が得られた。

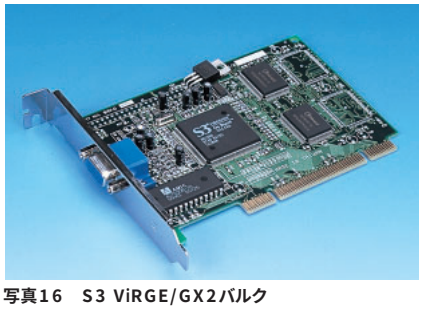


写真16　S3 ViRGE/GX2バルク
S3社のViRGE/GX2、SGRAM 4Mバイトを搭載したグラフィックスカード。購入価格4000円。

■独断的お勧め度！

ちょっと無理やりではあるが、xengineを使って計測すると、1024×768×16ビットカラー表示時に、Mach 64で平均250、ViRGE/GX2で平均720という値になる。約3倍ほど高速になっていることがわかる。が、しかし、使い物にはならない…。DIAMOND Stealth 3D 2000では1280×1024×16ビットカラー表示時に平均825という数値であった（ちなみにViRGE/GX2では同解像度で平均1520！やっぱり惜しいなぁ…）。

まずこのマシンでは主にViRGE系しか動作報告がないこと、最新のViRGEチップが動かないことなどを考えると、Mach64でも十分という気がしてくる。急遽テストしたStealth 3D 2000は動作することから、中古パーツとして同カードが入手可能であれば、大画面が使えるという点でお勧めである。設定は簡単なので文句はないのだが。というわけで、Stealth 3D 2000という前提でのお勧め度は星1つだ（ViRGE/GX2では星はゼロ）。

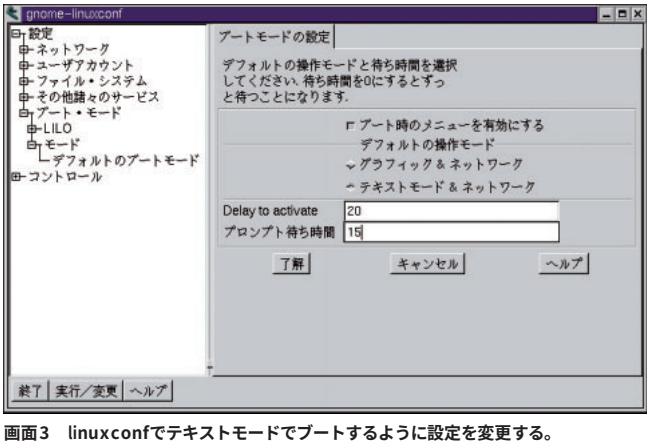
グラフィックスカード増設お勧め度！
★☆☆☆

目的は達成されたか？

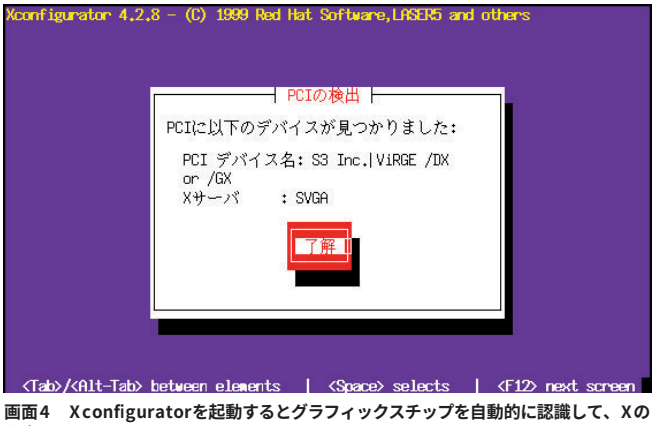
さて、結局どうなったかというと、**表4**のような仕様になった。実際に組み込んだパーツの費用は、CPUとハードディスク代で4万1600円になる。メモリとグラフィックスカードは0円だ。この仕様だと、3Dアクションゲームは難しいが、「Civilization：Call To Power」などのシミュレーションゲームはちゃんと遊べるようになった。GIMPで大きな画像も扱え、速度的にもさほどストレスは感じない。ちょっとしたワークステーションとして使おうという気になるレベルには達したものと思う。元がFMV DESKPOWER 5100D5だと思えば十分なスペックになったが、イマイチ諸手を挙げて喜べないのも事実だが…。

項目	仕様	値段
CPU	HK6-ms V2（K6-III 400MHz）	2万6600円
メモリ	72Mバイト	1万6900円（実際は0円）
HDD	E-IDE 15Gバイト	1万5000円
グラフィックス	DIAMOND Stealth 3D 2000（VRAM 4Mバイト）	0円（新品は入手不可能）

表4　最終仕様



画面3　linuxconfでテキストモードでブートするように設定を変更する。



画面4　Xconfiguratorを起動するとグラフィックスチップを自動的に認識して、Xの設定をしてくれる。

Column

ハードウェアの相性

今回のように旧型のPCを扱うと、思わぬ面で苦労を強いられることがある。FMV DESKPOWER 5100D5で言えば、1995年のモデルということもあり、拡張スロットとBIOSの部分がそれに相当した。本文で紹介したように、同機のスロット構成は、ISAスロット×1、PCI／ISA共用スロット×2となっており、3つのスロットすべてがISAデバイスで埋まることも想定されている。

このことからもうかがえるように、同機はまだPCIデバイスよりもISAデバイスのほうを使う機会が多い時期の製品だったのである。現在でこそ、PCIとISAの間を取り持つブリッジは安定しているが、PCIデバイスが登場したての時期には、いろいろ問題も多かったようだ。

事実、同機では、発売当初、Adaptec社のバスマスタSCSIカードを認識しないという問題があった。しかし当時は、Adaptec社側でこれを対応させてしまったという。現在ほどには多種多様なデバイスが登場しているわけでもなかったので、こうしたことが対応可能だったのかもしれないし、あるいは、FMVの出荷台数を考えたうえでの対応だったかもしれない。

また、ビデオカードの増設についても、S3社のViRGE系チップを搭載したカード以外での動作報告がほとんどなく、ユーザーにとっては、システム拡張も慎重に行わなければならなかっただろう。

こうした互換性の問題を「相性」と呼ぶことが多い。PC/AT互換機を自作する場合に、よく「このマザーボードとこの拡張カードは相性が悪い」とか、「このビデオカードとこのSCSIカードは相性がいい」というように使われる。PCのパーツ同士に相性もないもんだと思うかもしれない。まぁ、実際には、どのデバイスも定められたPCIの規格に則って設計されているわけで、微妙なタイミングのずれや、規格にない部分の解釈等で互換性を逸しているのだろうが、具体的にどこが互換性を欠いているのかユーザーからは計り知れない。そのため試行錯誤して見つけ出した互換性を「相性の良し悪し」と呼ぶことで納得しているのだ。ショップなどでパーツ購入の際に「相性」という言葉を聞いたら、そういう意味である。

■BIOSアップデート

今回、PCIデバイスの認識の問題で、最終手段として選択したことにマザーボードのBIOSアップデートがある。BIOS（Basic Input Output System）は、電源投入時に実行されるプログラムで、PCの電源投入時にちゃんとFDDやHDDに記録されているOSのブートプログラムを読みにいくのもこのBIOSのおかげである。

このBIOSには、ブート機能以外にも、マザーボード上の各種ハードウェアデバイスの設定や、自己診断などのプログラムが組み込まれている。そのためハードウェアを拡張する際に、このBIOSが正しく認識してくれないと、動作しないことがあるのだ。つまり、このBIOSが開発された時期に存在していないハードウェアについては、正しく認識してくれない可能性が高い。しかしその反面、BIOSのバージョンをアップデートさえすれば、動く可能性もあるわけだ。

BIOSはFlash ROMという、ソフトウェアで書き換え可能なROMに搭載されている。マザーボードメーカーでは、最新BIOSのアップデートプログラムを自社サイトで配布している。しかし、FMVのような大手メーカー製PCの場合、BIOSアップデートは、自社のサービスセンターへの持ち込みでしか対応していないことも多い。これは、BIOSのアップデートが危険な行為でもあるためだ。なぜなら、書き換えに失敗すると、そのPCは二度と起動しなくなってしまうからである。この理由から、今回もBIOSアップデートは最後の手段として、とっておいたのだ（しかし、結果は、アップデートに成功しても、UltraATA/66、ViRGE/GXとも動作しなかったわけだが…）。

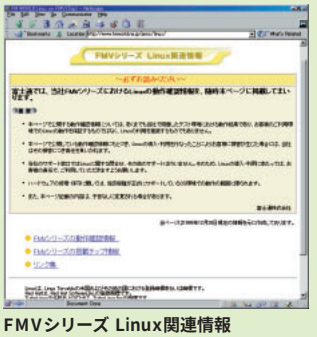
今回はメーカーのサービスセンターへ持ち込まず、自分で書き換えを行ったが、これはメーカーのサポート範囲外であり、もし失敗してから慌ててメーカーへ持ち込んでも取り合ってくれない。しかも、目的とするデバイスが動作しなかったこともあり、ここでその具体的方法論を書くことは避けるが、旧型PCをチューンアップする際には、BIOSの新旧という点についても、追求してみる余地があるだろう。

■FMVユーザーお役立ちサイト

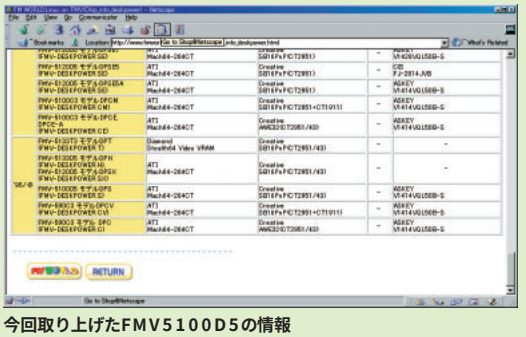
最後になったが、今回のようにFMVをLinuxで再利用する場合、オンボード搭載されたチップの型番や、セット売りされていたディスプレイのスキャン周波数帯域などが知りたくなるだろう。そうしたときに役立つサイトを以下に示しておこう。

同サイトでは1995年以降のFMVシリーズのチップやディスプレイの周波数帯域が一覧できる。また比較的新しい型番であれば、Linuxのディストリビューションパッケージごとの動作結果も掲載されている。FMVユーザーは、Linux導入前に一度覗いてみることをお勧めする。

FMVシリーズ Linux関連情報
http://www.fmworld.ne.jp/annc/linux/index.html



FMVシリーズ Linux関連情報



今回取り上げたFMV5100D5の情報

Compaq PRESARIO 7242を限界までチューンする

文：藍 植緒　Text : Ueo Ai

今回取り上げるPRESARIO 7242は、1996年3月に発売された家庭向けの薄型デスクトップマシンだ。スピーカー付きのディスプレイとセットで販売され、Windows 95がプレインストールされた本体には、Pentium 133MHz、1Gバイトのハードディスク、4倍速のCD-ROMドライブなどが装備されていた（**表1**）。

発売された当時は、Windows 95ブームで、世間ではマルチメディア時代と騒がれたものだったが、4年も経った今となっては力不足の感が否めない。アスキー社内において、このマシンがどのような使われ方をしてきたのかはわからないが、退役マシン保管倉庫にひっそりと眠っていたのもを借り受けてきた。セットになっていたはずのディスプレイは見あたらなかったが、すっきりとしたデザインのスリムな筐体は古さを感じさせない（**写真1**）。

みなさんのまわりにも、Windows 95ブームの頃のマシンが使われることなく放置されていたりするのではないだろうか。今回は、このマシンをパワーアップし、バリバリのLinuxマシンとして復活させてみよう。

写真1　Compaq PRESARIO 7242
薄型の筐体なので、華奢に見えるが、ケースには厚い鉄板を使用し、最近の安物マシンとは一線を画す。機能的にまとめられたデザインにも好感が持てる。

Linuxをインストールする

ノーマルといっても、このマシンにはメモリが増設されており、合計64Mバイトになっていた。標準の16MバイトのままではX Window SystemでGNOMEのようなデスクトップ環境を利用するにはつらいだろう。メモリの増設は、72ピンのSIMMで行う（**写真2**）。マザーボードのシルク印刷を見ると、“EDO MEMORY ONLY”と書かれている。FPM（Fast Page Mode）RAMに比べ、データの連続読み出し性能を向上させたEDO（Extended Data Out）RAMが必須のようだ。また、アクセス速度は、60nsか70nsのものが利用可能で、マザーボード上のジャンパピンで設定する。搭載されていたSIMMは60nsのものだったので、ジャンパ設定もそれに合わせた。

さらに、本機のISAバススロットには、3Com社製のイーサネットカードEtherLink III（3C509B-TP）が接続されていた（**写真3**）。3C509B-TPは、10BASE-Tと、10BASE-5に対応したカードで、ベストセラーモデル3C509にバッファを増設した名機だ。Linuxはもちろん、数多くのOSに対応しており、安定度も高いのでそのまま使うことにした。

インストールするLinuxのディストリビューションには、本誌の付録CD-ROMにも収録したTurboLinux 4.5を選んだ。早速インストールを開始しようとTurboLinuxのCD-ROMをドライブに入れ、本体の電源を入れる。おや？　Windows 95が起動した。BIOSの設定でCD-ROMから起動しないようにしてあるのかと思い、BIOS設定プログラムを起動する。Compaqのマシンには、ハードディスクからBIOS設定プログラムを起動するものがある。このマシンもそのひとつで、起動時にF10キーを押すことで、専用のパーティションから設定プログラムを呼び出す。ところが、CD-ROMから起動する設定項目がない。そうだ、この頃のマシンはCD-ROMブートできないのが普通だったのだ。

しかたがないので、Windowsからブートフロッピーを作成し、これで起動してインストールをした。インストーラはイーサネットカードもあっさりと認識した。インストールの途中で既存のパーティションをすべて消し、

項目	仕様
CPU	Pentium（P54C）133MHz（Socket7）
ハードディスク	Seagate社製 ST5180A 1Gバイト（E-IDE）
CD-ROMドライブ	SONY製 4倍速 CDU77E（ATAPI）
メモリ	64Mバイト（標準16Mバイト）
ビデオ	S3 Trio64V+ 2Mバイト（オンボードPCI接続）
拡張スロット	ISA×2、PCI／ISA×1、PCI×1
電源容量	150W

表1　Compaq PRESARIO 7242の仕様
参考URL：http://www.compaq.co.jp/products/old/old_presario7242.html

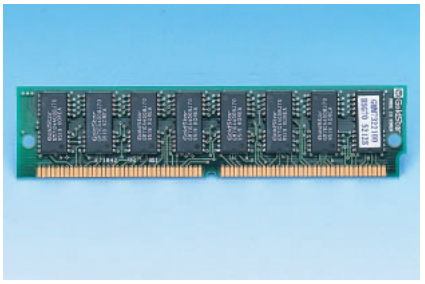


写真2　72ピン 16MバイトEDO SIMM
最近ではすっかり品薄になり、SDRAMの載ったDIMMより高価になってしまった。このSIMMはパリティなしのタイプ。

Linuxのために新たにパーティションを作成する。しかし、1Gバイトのハードディスクでは容量が足りず、すべてのソフトウェアパッケージを選択することができない。とりあえず必要になりそうなパッケージだけを選んでインストールは終了した。

CPUパワーアップ作戦

マシンをリブートしてLinuxを起動する。インストール時にグラフィカルログインを選んだのでXも起動する。デスクトップ環境はGNOMEだ。うーん、メニュー表示の反応が鈍い。使えないというほど遅いわけではないのだが、もったりとした感じとでもいおうか、表示がワンテンポ遅れるのでちょっと使いづらい。よし、CPUを高速なものに乗せ換えよう！

■CPUを交換するぞ

と、決めたはいいが、まずはこのマシンで使えるCPUにはどのようなものがあるのかを調べなければならない。標準でSocket7にPentium（P54C）133MHzが搭載されているのは前述の通りだ（**写真4**）。ところが、Socket7を採用しているとはいえ、このマザーボードは3.5Vのシングルボルテージしか供給できないため、いわゆるMMX Pentium（P55C）やAMDのK6シリーズ、Cyrixの6x86やM IIなどをそのまま載せることはできない。単純にCPUを交換するとすれば、Pentium（P54C）かIDT Winchip C6、Winchip2から選択することになる。残念ながら、これらのCPUはすでに品薄で入手が難しいうえに、300MHz以上のものがないため劇的な高速化は望めない。

そこで、今回はアイ・オー・データ機器のCPUアップグレードキットPK-K6H400/DVを利用することにした（購入価格1万8600円）。このキットは、AMD K6-2 400MHz、電圧変換ソケット（通称ゲタ）、CPUファンをセットにしたもので、デュアルボルテージに対応していないマザーボードでも利用できるのが特徴だ（**写真5**）。実際にCPUが動作する周波数はFSBの6倍となる。このマシンではジャンパピンの設定でFSBを40／50／60／66MHzから選べるが、もとのPentium 133MHzが66MHz×2で動作していたのでそのままFSB 66MHzのまま使う。従って、K6-2は400MHzで動作するのだ。

■CPUアップグレードの注意点

ここで、CPUをアップグレードする際に注意すべき点に触れておこう。自作マシンのように、さまざまなCPUに対応したマザーボードを使用している場合は、マザーボードのマニュアルを参照して利用可能なCPUを入手すればよい。しかし、今回のようなメーカー製PCでは、CPUの交換はPCメーカーの保証外となる。

また、CPUアップグレードキットを利用する場合も注意が必要だ。これらのキットがすべてのPCで動作するとは限らない。実際に、PK-K6H400/DVをDELLのOptiplex GXMT166というPentium 166MHzのマシンに差したところ、全く起動しなかった。どうやら、BIOSでCPUの種類をチェックしているらしい。このような場合は、MR BIOSなど、サードパーティ製のBIOSを購入するという解決方法もある。旧式マシンのパワーアップにそこまで手間と費用をかけるのはどうかとは思うが……。

さらに、CPUアップグレードキットには電圧変換ゲタやCPUファンがあるため、もとのCPUを装着している状態よりも大きなスペースが必要になるこ

写真3　3Com 3C509B-TP
抜群の安定性と信頼性を誇るベストセラーのイーサネットカード。しかし、それなりの価格だったので個人で買うとうらやましがられた（筆者は買えなかった）。

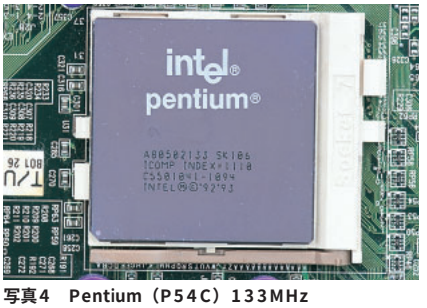
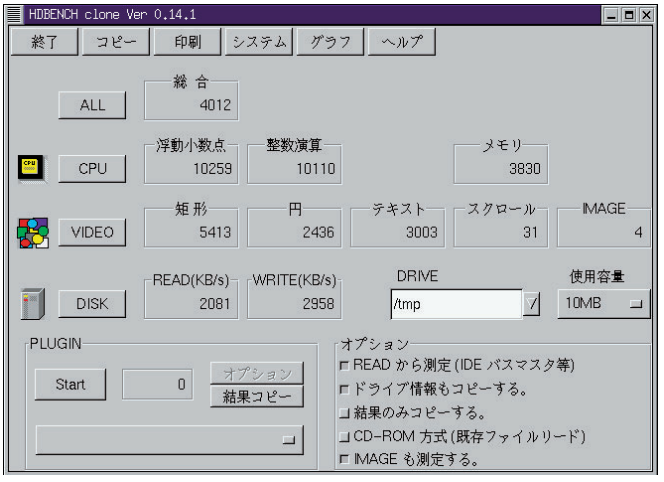


写真4　Pentium（P54C）133MHz
セラミックのPGAパッケージが懐かしい。実際に装着されているときは大きなヒートシンクがついているので、この面は見えていない。

写真5　アイ・オー・データ機器のPK-K6H400/DV
AMD K6-2 400MHzを採用したCPUグレードアップキット。電源から12Vの供給を受け、レギュレータで必要な電圧に降圧してCPUを動作させる。



画面1　HDBENCH cloneの画面
Windowsでは有名なHDBENCHにそっくりな外観をしている。GPLに従って配布されているフリーソフトウェア。

とが多い。このため、CPUまわりのスペース（特に高さ）も事前の確認が必要だ。このほかにも、消費電力の大きな最近のCPUを使うためには、電源容量に余裕がなければならないので注意しよう。

CPUアップグレードキットのメーカーは、Webページで対応機種を公開しているので、自分のPCが対応しているかどうか確認しておくと安心だ。

実は、PK-K6H400/DVはPRESARIO 7242に対応していないのだが、動けばラッキーという突撃精神で試したところ問題なく動作した（もっとも、対応機種であってもLinuxは正式にサポートしていない）。

小うるさいことはこれくらいにして、CPUアップグレードの効果のほどを見てみよう。CPUの動作周波数が一挙に3倍になるのだからさぞや速くなっているに違いない。

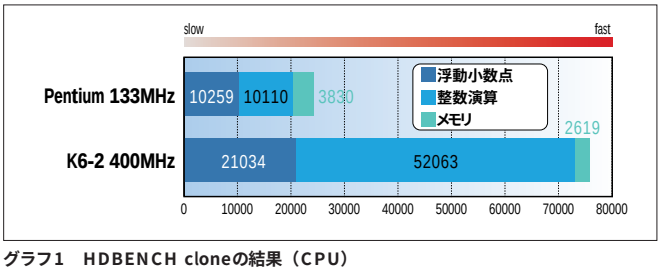
■CPUアップグレードの効果

二之宮 祐樹氏が作成し、公開しているHDBENCH cloneというベンチマークプログラムを使い、CPUアップグレードによるパフォーマンスの違いを見てみよう。このHDBENCH cloneは、Windowsでよく使われるベンチマークプログラムHDBENCHにそっくりな画面を持っているが、見た目がそっくりなだけで中身は全く別のものだという（**画面1**）。Pentium 133MHzとK6-2 400MHzで実行した結果は**グラフ1〜3**のようになった。Xの解像度は1024×768ドット、色数は16ビット色にして測定している。

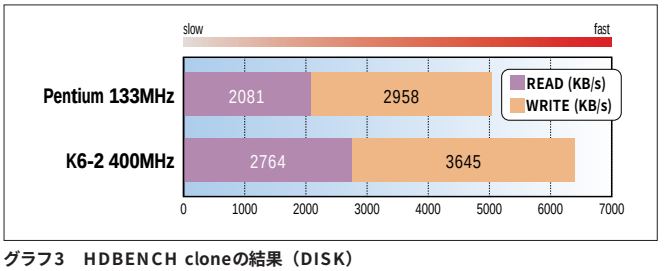
CPUのパートは大きく向上した。浮動小数点演算が2倍、整数演算はなんと5倍だ。ただ、メモリは逆に成績が悪化している。VIDEO、DISKも高速化できている。

このマシンにはオンボードPCI接続のグラフィックスアクセラレータ、S3社 Trio64V+が搭載されているため、X Window SystemはXFree86のS3用サーバを使っている。このサーバは、S3社製チップのアクセラレーション機能を使うため、CPUが直接すべてを描画しているわけではない。このため、CPUが劇的に速くなってもXの画面描画はそれほど速くならないのだろう。

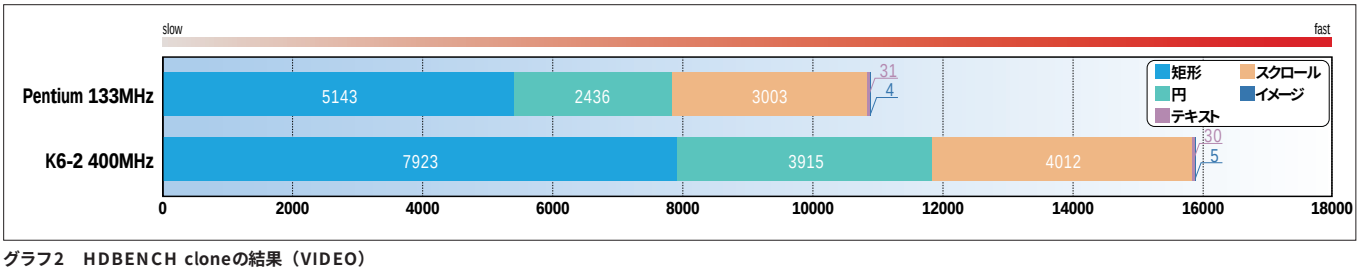
次に、DISKの結果だが、これはハードディスクとのデータの転送をCPUが行うPIOモード（Program I/O



グラフ1　HDBENCH cloneの結果（CPU）



グラフ3　HDBENCH cloneの結果（DISK）



グラフ2　HDBENCH cloneの結果（VIDEO）

mode）を使っていることから、CPUの能力アップとともにDISKの成績も向上したものと思われる。

しかし、CPUパートにあるメモリの成績が落ちたのはどうしてだろうか。これを調べるため、別のベンチマークプログラムLMBENCHを使用してみた。これはBit Moverが公開しているベンチマークプログラムで、メモリやネットワークなどの帯域幅や遅延速度などを計測することができる。今回は、このLMBENCHでメモリの遅延速度（レイテンシ）を測った。まず、CPU内蔵の1次キャッシュメモリの遅延速度を見てみよう。Pentium（P54C）は16Kバイト、K6-2は64Kバイトの1次キャッシュを持つ。測定の結果は、Pentium 133MHzが 15ns、 K6-2 400MHzが5nsだった。動作周波数の高いK6-2のほうが高速だ。そしてメインメモリへのアクセス遅延は、Pentiumが250nsなのに対し、K6-2は473nsもかかっている。これはあくまで筆者の推測なのだが、PK-K6H400/DVは、遅いメモリを搭載した古いマシンでもメモリアクセスが正しくできるようにウェイトを挿入しているのではないだろうか。まあ、安全第一だし、システム全体でのパフォーマンスが大幅にアップしているのだから細かいことは気にしないことにしよう。

あれ？ 2次キャッシュは？ と思った方もいるだろう。なんとこのモデルでは2次キャッシュはオプションとなっており、標準では装備されていないのだ。これを考えると、このマシンには、CPUに1次キャッシュだけを内蔵するK6-2ではなく、256Kバイトの2次キャッシュもCPU内に持つK6-IIIを使ったCPUアップグレードキットを使ったほうが効果があったのかもしれない。

グラフィックスアクセラレータを増設！

CPUを強化することで、GNOMEのデスクトップ環境もずいぶん快適になった。しかし、オンボードPCI接続のグラフィックスアクセラレータには2Mバイトのビデオメモリしか用意されていない（**写真6**）。このメモリ容量では1024×768ドット16ビット色（厳密は1152×864ドット）までの環境しか得られない。そこで、アクセラレータカードを増設してみよう。 このマシンにはAGPバススロットなどという新しいものはもちろん付いていないので、今回はたまたま持っていたMatrox Millennium PCI 8Mバイトを接続した。このカードもPRESARIO 7242に負けないくらい古いものだが、一世を風靡したベストセラーなのでお持ちの方も多いのではないだろうか。8Mバイトのビデオメモリがあれば1600×1200ドット32ビット色まで対応できる。

オンボードのアクセラレータを自動的に切ってくれるのかわからなかったので、BIOSの設定プログラムを呼び出そうとしてハタと困ってしまった。TurboLinuxをインストールするときに、BIOS設定プログラムの入っているパーティションもきれいさっぱり消してしまっていたのだ。しまった、うかつだったと自分のアホさ加減に歯がみしていてもしょうがない。こんな時はメーカーのWebサイト（http://www.compaq.co.jp/）を見るに限る。あった、設定プログラムがあった。よかったーと胸をなでおろし、専用の起動フロッピーを作成した。PCIスロットにMillenniumを挿し、このフロッピーで起動するとBIOS設定プログラムが立ち上がった。どうやらオンボードのグラフィックスチップは自動的に動作を止めているようだ。

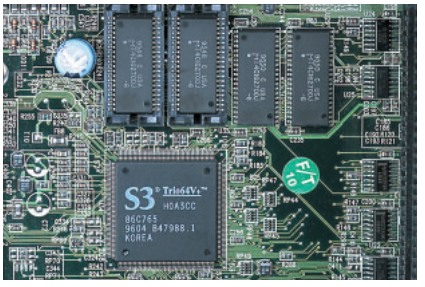


写真6 PRESARIO 7242に搭載されているグラフィックスアクセラレータ
S3 Trio64V+（下）とビデオメモリ（上4つ）がマザーボード上に実装されている。

気を取り直してLinuxをブートする。白黒のコンソール画面でログインプロンプトが表示され、画面が不気味に点滅している。ログインもできない。むむ。そうだグラフィカルログインに設定していたので、S3チップ用のXサーバを起動しようとしては失敗しているのだ。再び己の間抜けさに脱力しながら再起動する。そしてLILOのboot:プロンプトから、

```
boot: linux 3
```

とランレベル3を指定してLinuxをブートする。TurboLinux 4.xでは、X Window Systemはランレベル5で動作し、ネットワーク機能付きのマルチユーザーテキストモードはランレベル3だ。従って、これでテキストログインが可能になるのだ。今度は無事にログインできたので、XF86Setupコマンドを使ってXFree86の設定をする。MillenniumはXFree86のSVGAサーバを使うので、TurboLinuxのCD-ROMをマウントしておく必要がある。

Xを1280×1024ドット32ビット色で立ち上げると画面が広い。

■パフォーマンスはどうか

さて、パフォーマンスはどう変わっただろうか。XFree86は、Millennium

シリーズでのアクセラレーションがかなり効くと聞いている。しかし、S3サーバも長い時間をかけて改良されてきているのでできは悪くないはずだ。比較のため、先ほどと同じ1024×768ドット16ビット色のモードにしてHDBENCH cloneを実行した。**グラフ4**がその結果だ。イメージ以外は速くなった。特にスクロールは見ていて驚くほどに速くなった。これでGNOMEやKDEのような重いデスクトップ環境でもサクサクと小気味よく動作するようになった。

もっと大きなハードディスクを！

CPUとグラフィックスアクセラレータを強化するだけで、PRESARIO 7242は十分高速なXワークステーションになった。しかし、大事なことを忘れてはいけない。ハードディスクに空き容量がほとんどないことだ。TurboLinux 4.5をフルインストールできないし、それよりもデータを保存できないのだ。これでは困る。そこで、もっと大きなハードディスクに換装してみよう。

というわけで、IBM製の15GバイトIDEハードディスクDPTA-351500を買ってきた。これは、ディスク回転数5400rpmの普及モデルで、購入価格も1万5000円と安価だ。そうはいっても、PRESARIO 7242に標準で付いているSeagateのハードディスクの実に15倍もの容量があるのだ。当然プラッタ（ハードディスクの中の円盤）あたりの記録密度も上がっているため、ディスクへの読み書きの速度も向上している。

■ハードディスクが認識されない

購入して戻るやいなや、嬉々としてディスクをつなぎ換え、BIOSのセットアッププログラムを起動した。PRESARIO 7242では、ハードディスクの容量をC／H／S（シリンダ／ヘッド／セクタ）というパラメータで設定する。購入してきたDPTA-351500のラベルを見ると、C／H／Sは29104／16／63だと表示してあった。ところが、シリンダが16644までしか設定できないのだ。このBIOSは、C／H／Sの数字を掛けた数、すなわちディスクの全セクタ数が24ビット以内に収まらないと認識しないようだ。このマシンが誕生した頃にはこんな大容量のハードディスクを接続することは想定されていなかったので仕方がない。BIOSのアップデートで何とかならないかとCompaqのWebサイトを探してみたが、残念なことに新しいBIOSは見つからなかった。

■外付けのIDEカードを使う

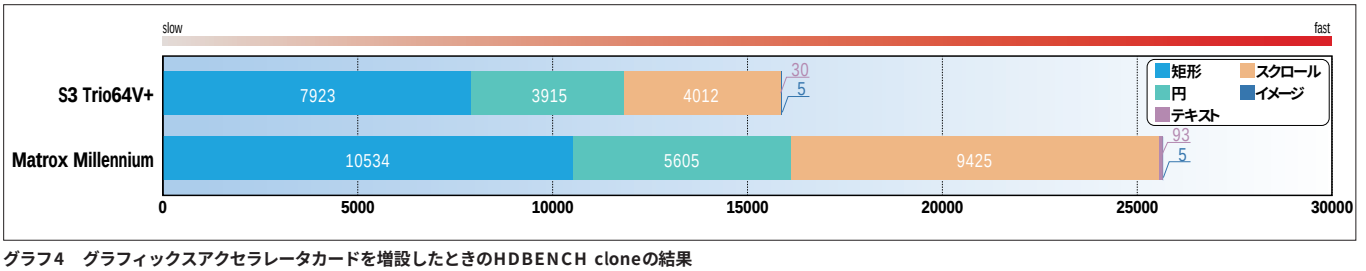
いまさら小容量のハードディスクを探して買い直すのはくやしいので、PCIバススロットに接続するIDEカードを購入することにした。最近は、大容量のハードディスクに対応し、高速な転送モード、UltraDMA/66に対応したIDEカードが何種類か販売されている。これらのカード上には、独自のディスクBIOSが搭載されており、PRESARIO 7242のような旧式マシンでも最新のハードディスクが使えるようになるのだ。今回は、Promise製のUltra66というカードを購入した（**写真7**）。化粧箱なしのバルクのもので、購入価格はおよそ4000円だった。

2つあるPCIバススロットの1つにはグラフィックスアクセラレータカードのMillenniumが挿してあるので、Ultra66を装着するとPCIバススロットはすべて埋まる。ふう、あぶないあぶないと思いきや、正しく認識されない。BIOS設定プログラムを駆使していろいろな設定を試したのだが、残念ながらこれら2枚のカードを共存させることはできなかった。断腸の思いでどちらかをあきらめなくてはならない。つらい選択だったが、ハードディスクを選んだのだった……というか、ここでやめたらこの特集企画が吹き飛んでしまう ^^;;。

■TurboLinuxの再インストール

Ultra66を装着してUltraDMA/66に

写真7　Promise Ultra66
UltraDMA/66に対応した安価なIDEカードの1つ。ハイエンド向けになりつつあるSCSIに比べコストパフォーマンスは高い。



グラフ4　グラフィックスアクセラレータカードを増設したときのHDBENCH cloneの結果

対応したケーブルでDPTA-351500に接続する。電源を入れるとUltra66のBIOSがDPTA-351500を認識していることがわかった。さて、もう一度TurboLinuxをインストールしよう。ブートフロッピーからインストーラを起動してインストールを進めると、CD-ROMドライブが見つからないといって止まってしまった。CD-ROMドライブは、オンボードIDEインターフェイスのセカンダリマスタとして接続していたのだが、プライマリインターフェイスに接続してあったハードディスクがなくなった（ハードディスクはUltra66につないだため）のでうまく認識されていないようだ。CD-ROMドライブをプライマリマスタとして接続すると、インストーラが認識した。

ところが、今度はハードディスクが見つからないという。どうもUltra66カードをインストーラ（のLinuxカーネル）が認識できないようだ。そこで、Linux Documentation Project（日本のミラーサイトhttp://mirror.nucba.ac.jp/mirror/LDP/）にある“The Linux Ultra-DMA Mini-Howto”を調べてみると、次のようにすればインストールできることがわかった。

1. **Ultra66の使うI/Oポートアドレスを調べる。**
2. **調べたI/Oポートアドレスをインストーラに知らせる。**

下準備としてUltra66の使うI/Oポートアドレスを調べる必要がある。インストーラを起動し、CD-ROMドライブを認識したらAlt＋F2キーを押す。するとbashのコンソール画面になるので次のように入力する。以前のハードディスクからLinuxを起動して、bashから入力してもよい。

```
# cat /proc/pci
```

すると、画面にPCIバスに接続されたデバイスの情報が表示されるので、その中から**画面2**のようなUltra66のリソース情報を探す。①～④のI/Oアドレスから、インストーラに渡すオプションが決定する。オプションの決め方は次のようになる。

```
ide2=①,②+2 ide3=③,④+2
```

画面2の例では、

```
ide2=0x1050,0x1062  ide3=0x1058,0x1066（実際は1行）
```

がインストーラに渡すオプションだ。

```
  Bus  0, device   5, function  0:
    Unknown mass storage controller: Promise Technology IDE UltraDMA/66 (rev 1).
      Medium devsel.  IRQ 11.  Master Capable.  Latency=64.
      I/O at 0x1050 [0x1051].      ←①      この①～④のアドレスを調べる
      I/O at 0x1060 [0x1061].      ←②
      I/O at 0x1058 [0x1059].      ←③
      I/O at 0x1064 [0x1065].      ←④
      I/O at 0x1000 [0x1001].
      Non-prefetchable 32 bit memory at 0x44000000 [0x44000000].
```

画面2　cat /proc/pciを実行してUltra66のリソースを調べる

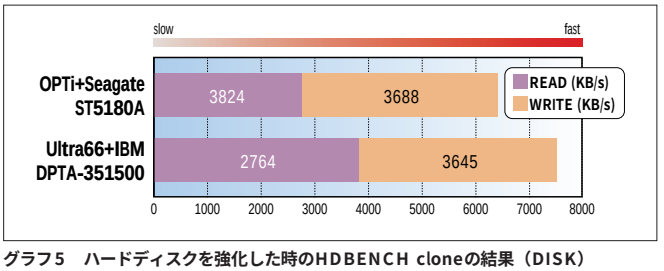
実際には、インストーラが起動して、boot:が表示されたら、

```
install ide2=0x1050,0x1062 ide3=0x1058,0x1066（実際は1行）
```

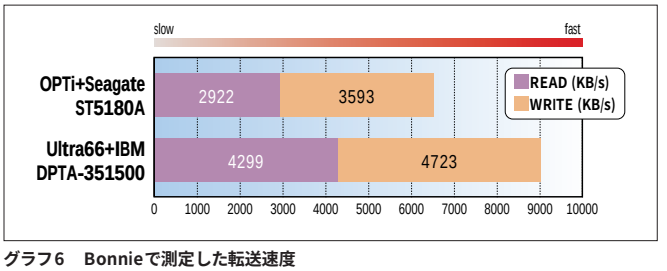
と入力する。インストールが進み、「LILO設定」の画面になったら、インストーラに渡すオプション欄に上記のide2＝…というオプションを入力し、「linearモードを使用する」にチェックを入れる。あとは通常通りインストールを進める。これでDPTA-351500にLinuxをインストールすることができた。

■不本意な結果？！

HDBENCH cloneでディスクのパフォーマンスを見てみよう。オンボードIDEコントローラOPTi 82C621にSeagateの1GバイトハードディスクST5180Aを接続していたときと、Ultra66にIBM DPTA-351500を接続したときの結果を示したのが**グラフ5**だ。確かに速くなってはいるが、そんなに差はない。特に書き込みはほとんど同じくらいではないか！



グラフ5　ハードディスクを強化した時のHDBENCH cloneの結果（DISK）



グラフ6　Bonnieで測定した転送速度

プログラム	URL
HDBENCH clone	http://www.enjoy.ne.jp/~gm/program/hdbench/index-ja.html
LMBENCH	http://www.bitmover.com/lmbench/
Bonnie	http://www.textuality.com/bonnie/index.html

表2　使用したベンチマークプログラムのWebページ

プログラム	URL
Unified IDEパッチ	ftp://ftp.ring.gr.jp/pub/linux/kernel.org/kernel/people.hedrick/

表3　Unified IDEパッチのURL

そこで、ディスク専門のベンチマークプログラムBonnieを試してみた。Bonnieは、キャラクタ単位での読み書きと、ブロック単位の読み書き速度を測ることができる。**グラフ6**に100Mバイトの連続読み込みと書き込みの測定結果を示す。HDBENCH cloneよりは差が開いたが、実力の差はこんなものではないはず……。

調べてみたところ、バージョン2.2系のカーネルはUltra66に使われているIDEコントローラチップ（PDC20262）に対応していないため、このままではPIOモードでしか利用できないことがわかった。要するにUltra66は真の力をまだ見せていなかったのだ（自分が悪いのだが）。

■Unified IDEパッチ

Ultra66の実力を引き出すためには、カーネルを最新のIDEコントローラチップに対応させるUnified IDEパッチをカーネルのソースにあててコンパイルし直す必要がある。パッチはRing Server ProjectのFTPサーバから手に入れた（**表3**）。

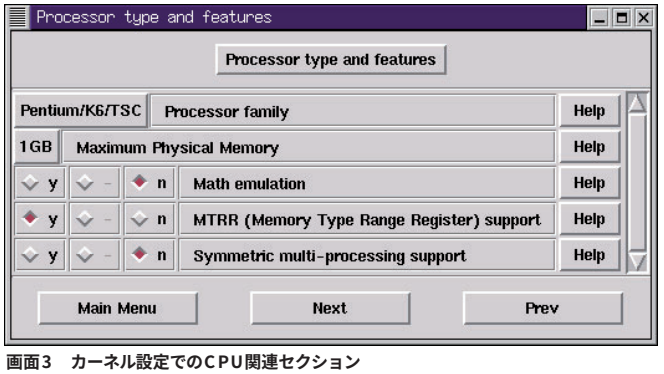
どうせカーネルを再構築するのならと、カーネル設定を行う際に「Processor type and features」でProcessor familyを“Pentium／K6／TSC”にし、“MTRR（Memory Type Range Resister）support”も組み込むことにした（**画面3**）。さて、このパッチをあてると、「Block devices」の画面で“PROMISE PDC20246／PDC20262 support”を選べるようになる（**画面4**）。Ultra66に搭載されているチップはPDC20262なので、この項目をyにしてカーネルに組み込む。さらに、起動時からDMA転送が有効になるように“Use DMA by default when available”もyにした。このあたりの設定は、2000年2月号の「ハードディスク大増設指南」で詳しく解説されているので、バックナンバーをお持ちの方はご覧いただきたい。

カーネルをコンパイルし、LILOの設定をして再起動すれば転送モードはUltraDMA/33になる。また、LILOのboot:プロンプトで“ide2＝ata66 ide3＝ata66”と指定するか、/etc/lilo.confでこのオプションを指定すれば転送モードはUltraDMA/66になる。

気になるパフォーマンスは、Bonnieで測定した結果を**グラフ7**に示す。まさに圧倒的な速度差だ。UltraDMA/66モードでは、標準での環境に比べ、読み出しで3.8倍、書き込みで3.1倍も高速になった。もう元の環境には戻れない。UltraDMA/33とUltraDMA/66で速度差がほとんどないのは、ディスクの性能が限界に達しているためだと思われる。UltraDMA/33モードでの転送速度は33Mバイト／秒、UltraDMA/66では66Mバイト／秒なので、規格上はまだまだ余裕があるのだ。7200rpmや10000rpmのハイエンド向けハードディスクをつなげば有意な差が見られるのかもしれないが、ここでの目的は古くなってしまったPRESARIO 7242の復活なので、15Gバイトのディスクが利用できただけでも大成功だといえよう。といいつつ、メインマシンにも導入したいと思っているのは筆者だけではないはずだ。

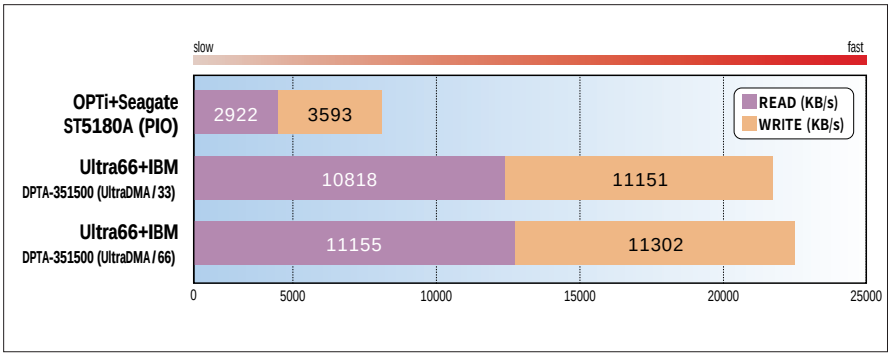
MTRRを設定する

先ほど、カーネルの再構築でMTRR（Memory Type Range Resister）を有効にした。K6-2の後期型やK6-III、あるいはPentium II／III、Celeronは、



画面3　カーネル設定でのCPU関連セクション
K6-2後期型やK6-III、Celeron以上のCPUではMTRRを有効にしておくといいことがあるかもしれない。

画面4　カーネル設定でのBlock device関連セクション
Ultra66で使われているPromiseのE-IDEチップを有効にする。



グラフ7　UltraDMAモードでの転送速度

Write Combiningという機能を持っており、MTRRという特殊なレジスタで指定した範囲のメモリに対してこの機能を有効にすることができる。さて、そのWrite Combiningとはどのような機能かというと、1バイトとか2バイトといったメモリへの書き込みを留保しておいて、8バイト（64ビット）溜まったところで一気に書き込むというものだ。K6シリーズは、64ビット単位でメモリへの書き込みを行うので、連続したアドレスに対し、1バイト（8ビット）ずつ書き込むより64ビット溜めたほうが効率がよくなり、速度の向上が見込まれる。

ただし、書き込みタイミングにシビアなアドレスに対してこの機能を有効にしてしまうと、システムが正しく動作できなくなってしまう。そこで、MTRRでメモリの範囲を指定し、非キャッシュ／Write through／Write back／Write Combiningといった動作を指定するのだ。

一般に、Write Combiningを有効にするのに向いているのはグラフィックスカードのフレームバッファ領域だといわれている。実際に試してみよう。フレームバッファのアドレスを調べるために、次のようにログを取りながらXを起動する。

```
# startx &> Xlog.txt
```

出力されたXlog.txtファイルに次のような記述があるはずだ。

```
(--) S3: PCI: Trio32/64 rev 40,
Linear FB @ 0x40000000（実際には1行）
          :
(**) S3: videoram:  2048k
```

これでフレームバッファのアドレスが0x40000000から2048Kバイト（16進数表記で0x200000バイト）だとわかる。この範囲をWrite Combiningに設定するには、次のようにする。

```
# echo "base=0x40000000 size=0x
200000 type=write-combining" >|
/proc/mtrr（実際には1行）
```

こうして設定した効果を見るために、首藤一彦氏の作ったxengineを実行した（**画面5**）。xengineは、レシプロエンジンの仕組みを動画像で表示するプログラムで、そのときの回転数を数値で表示する。これにより、Xの描画速度を測るというものだ。強化したPRESARIO 7242で実行すると、409rpmほどの回転数をマークした。ここでMTRRを設定すると457rpmほどに回転数がアップした。およそ1割の性能アップだ。

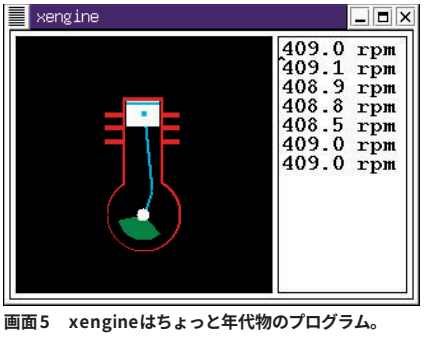
Write Combiningは、あくまでCPUがメモリに書き込む際に威力を発揮するものなので、グラフィックスアクセラレータが描画を行う部分では効果を持たない。HDBENCH cloneで試すとVIDEOの数値は変わらなかったので、ほとんどの描画処理はアクセラレータが行っているのだろう。また、現在のXFree86 3.x.xではWrite Combiningの設定は行われないが、XFree86 4.x.xからはWrite Combining機能を積極的に利用するという。

MTRRの設定については、カーネルに付属のmtrr.txtで詳しく説明されているので、興味のある方は読んでみてほしい。

PRESARIOは生き返ったか？

CPU、グラフィックスアクセラレータ、ハードディスクを強化した結果、十分体感できるほどの高速化ができた。もちろん、最新スペックのマシンに比べればメモリやチップセットの遅さが目立ってしまうが、よほど負荷の大きな作業をしない限りPRESARIO 7242はデスクトップワークステーションとして十分に使えるようになった。

あまり費用をかけすぎると、もっと速いマシンが1台買える出費になってしまうので、必要な部分を強化するのがよいだろう。この記事がいくらかでも参考になれば幸いである。



画面5　xengineはちょっと年代物のプログラム。最近のマシンでは速すぎて回転が見えないかもしれない。

IBM Aptiva 530

文：Tux Heaven　Text : Tux Heaven

そもそも486とは？

最近PCを使い始めた方の中には、「Pentium」をブランド名ではなく部品の名前だと思っている人がいるそうだ。これは極端な例だし、CeleronやAMD Athlonの立場はどうなるのか、という気もするが、実話だ。実際に1993年発表のPentiumから、最新のPentium IIIまで、インテルの主力CPUは常に「Pentium」と名乗っているので、このような微笑ましい勘違いが起きるのもやむをえないだろう。

PCに搭載されているCPUは、「x86系プロセッサ」のように表記されることが多い。これは、Pentiumの祖先にあたるCPUが、8086（16ビット）、80386（32ビット）のように「86」で終わる型番を持っているので、それらをまとめて「x86」とするのが慣例なのだ。

80486（**写真2**）は80386の高速版で、i486あるいは単に486とも呼ばれる。日本ではAT互換機が普及し始めたころ、主力だったCPUだ。当時から使っていたユーザーなら、80486搭載機種を1台くらいは所有したことがあるだろう。

そして、次はきっと「80586」だろう、と誰もが思っていたところに発表されたのがPentiumだった。さらにその次のCPUは、開発コードが「P6」だったことから、「Hexium」（ヘクシアム？）とでも命名されるのかと思っていたら、製品名はPentiumProだった。以後のPentium II／III、Celeronは、いずれもP6のコアを元にさまざまな改良が加えられたものである。つまり80486は、CPUのコアでいうと現在の2世代前の製品ということになる。

追加購入部品はなし!

Linuxは、低スペックのマシンでも使えると言われている。それを確認するためにも、今回の特集に486マシンを加えることになった。用いたのは、アスキー社内の倉庫から発掘してきた、IBMのAptiva 530である。オリジナルのスペックと現状を表1に示した。ハードディスクにはWindows 95がインストールされていた。メモリやハードディスクが増設されているとはいえ、かなり苦しいスペックだ。

古いマシンを活用する場合、ここでいろいろな部品を買い足すのが一般的なスタイルだ。この特集でもPentiumを搭載している機種については、Socket7に対応したCPUアクセラレータを使ってパワーアップを試みている。またPCIの拡張スロットを用いて、グラフィックスカードの交換や、より高速なIDEのインターフェイスを追加している。

だが486マシンの活用法を検討するにあたっての原則は、「金をつぎ込まない」である。というか実際には、つぎ込みようがないといったほうが正しいかもしれない。

たとえば486が装着されているSocket3に対応するCPUは、現状ではほぼ入手不可能である。たとえあった

項目	仕様
CPU	486DX 100MHz
メモリ	8Mバイト（現在は56Mバイト）
ハードディスク	720Mバイト（現在は1Gバイト）Western Digital AC21000H
CD-ROM	4倍速（ATAPI）
ビデオ	Cirrus Logic CL-GD5430（VRAM 1Mバイト）
ネットワーク	なし（現在はIntel EtherExpress PRO/10）

表1　IBM Aptiva 530のスペック（発売時）

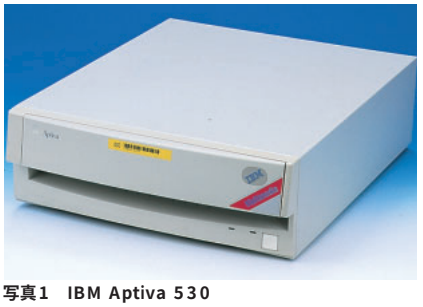


写真1　IBM Aptiva 530
はたして486マシンを活かす道はあるか？

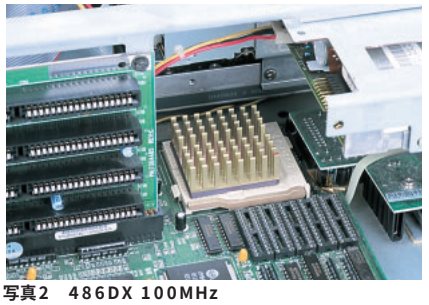


写真2　486DX 100MHz
FSBは33MHzで、コアは3倍の100MHz。

写真3　ISAのみの拡張スロット
最近のマザーボードには、PCIのみの製品もあるので、ISAスロットを見たことがない人もいるのでは？

としても理不尽とも思える高値が付けられていることが多い。

またAptiva 530の場合、拡張スロットはISAのみであり（**写真3**）、ネットワークカード以外のISA拡張カードがほとんど販売されていない現状では、拡張カードを追加してのパワーアップもこれまた望み薄だ。

結局、486マシンに追加する価値がある部品は、メモリと、あとで述べるネットワークカードだけと断言しておこう。486マシンは、8～16Mバイト程度で発売されていた機種がほとんどだ。しかしWindows 3.1や95で使用する場合には、メモリを追加して16～32Mバイト程度で使用されていたことが多いだろう。これならLinuxで使うにも十分だ。

もし8Mバイトしか載っていなければ、メモリの追加を考えたほうがいい。486マシンでは、多くの機種が72ピンのSIMMを用いている。72ピンのSIMMで提供されているメモリには、FPM（Fast Page Mode）とEDO（Extended Data Out）がある。自分のマシンに合ったものを選ぼう。FPM、EDOともに最近は少々手に入りづらいが、皆無というほどではない。値段は16Mバイトが5000円、32Mバイトで8000円程度だ。64MバイトのSIMMも売られているようだが、古いマシンでは使えないことが多いので、避けたほうが無難だ。なお、現在のメモリの主流であるSDRAMを用いたDIMMとは互換性がないので、注意が必要だ。

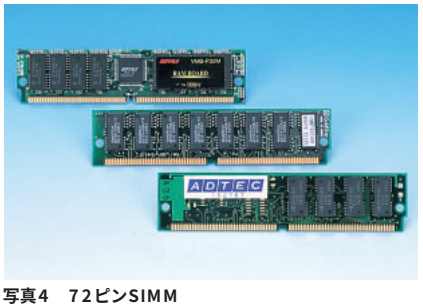


写真4　72ピンSIMM

今回使うAptiva 530は、メモリについては、そこそこ恵まれていた。3つあるSIMMスロットには、オリジナルの8MバイトSIMMに16M、32MバイトSIMMが1本ずつ追加されており、合計56Mバイト搭載されていた（**写真4**）。

ネットワークカードは必須だ

マシンの性能にかかわらず、今やコンピュータはネットワークにつないで使うのが基本だ。足りないリソースは、ネットワークを介して、ほかのマシンから分け与えて（奪って？）もらえばいいのだ。

もし複数のマシンを所有しているならば、ネットワークカードは必ず手に入れよう。PCIが使えるマシンなら、Linuxに対応した製品がいくらでもあるので、好みで選べばよい。しかしAptiva 530のようにISAスロットしかないマシンでは、使えるカードが限られてしまう。

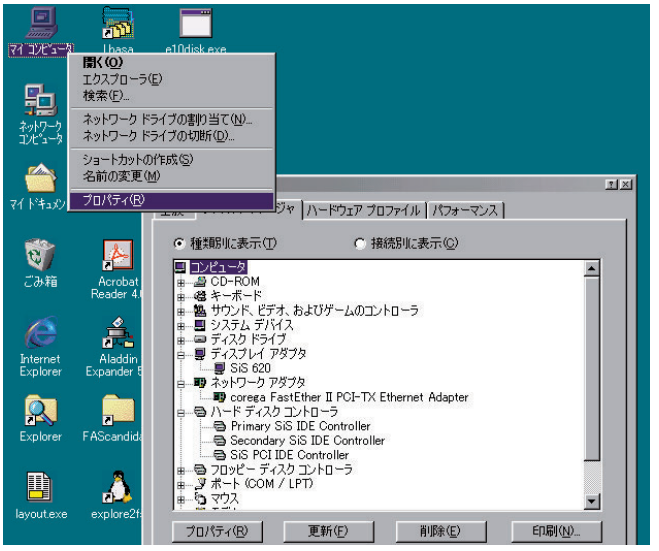
今でも多く販売されているのは、いわゆる「NE2000コンパチブル」カードだ。Linuxで使う場合、IRQやI/Oアドレスをユーザーが設定する必要があったり、性能の面ではあまり評判のよくない製品ではあるが、値段も安価（たいてい2000円以下）だし、ネットワークなしで使うよりははるかにましだ。

そのほかには、3Com社の3C509Bシリーズがある。NE2000コンパチブルほど多く見かけないし、運良く見つけられても6000～1万円（コネクタ形状により異なる）と少々高値だが、Linuxでのサポートは非常に優れており、定番中の定番だ。もし会社の倉庫で本体を探している際に、余っている3C509Bを見つけたら、有無を言わさず即ゲットしておくべきだ。

今回使ったAptiva 530には、あとから追加されたと思われる、インテルEtherExpress PRO/10（**写真5**）という10BASE-T対応のISAネットワークカードが挿してあったので、これをそのまま用いることにする。

情報集めが重要

古いマシンに限ったことではないが、Linuxを使う際には、組み込まれている部品に関する情報を集めておくと、インストール時に手間をかけずにすむ。



画面1　Windows 9xで情報収集。もちろんAptiva 530で撮った画面ではない。

古いマシンでマニュアルが揃っていない場合には、いろいろ手をつくして情報を集めよう。

筐体を開いて、直接目で見て確かめるのもいいが、もしWindows 95がインストールしてあったら、こいつを活用しよう。Linuxで上書きインストールする前に、「マイコンピュータ」のプロパティを開いて、デバイスマネージャでIRQやI/Oアドレスを調べ、メモしておけばいいのだ（**画面1**）。筆者は、この作業を怠り、いきなりLinuxをインストールしたため、ネットワークカードの設定でずいぶんと手間取ってしまった。

クライアントとして使ってみると？

まず、特に486マシンということを意識せずに、Linuxを普通にインストールしてみることにした。用いたのは、Vine Linux 1.1CRだ。パッケージは、「すべて」を選択した（**画面2**）。

インストールは、だいたい1時間程度で問題なく終了した。筆者が常用している350MHzのK6-2、15Gバイトのハードディスクを装備した「今どき」のPCでは、30分弱で終わる作業だ。2倍の差は、ハードディスクが遅いためと思われる。

再起動後、コンソールからEmacsなど大きめのアプリケーションを使ってみたところ、起動こそ時間がかかるものの、特に遅いと感じることもなかった。

だがそこそこ快適に使えたのは、ここまで。「startx」としてX Window Systemを起動してみたが、かなりストレスがたまりそうな動作速度だった。

Vine Linuxがデフォルトのウインドウマネージャとして採用しているWindow Maker（**画面3**）は、マウスの右クリックでメニューを出すのが操作の基本だ。Aptiva 530ではその右クリックのレスポンスが不確かで、すぐにメニューが出てくるときもあるのだが、数秒待たされることもあった。この状態で常用するのは、つらいだろう。

ウインドウマネージャをfvwm（**画面4**）に切り替えると、だいぶ軽快に動作し、テキストベースの作業なら使えるレベルになった。だが、画像関係のプログラムは全般的に動作が遅く、xvで画像を表示するだけでも実用にはならないと感じた。メモリ不足でスワップが起きているわけではないので、純粋にCPUパワーが足りないようだ。やはり486マシンにXワークステーションを演じさせるのは、荷が重いようだ。

「Pentium互換」ディストリビューションは動くか？

最近のディストリビューションは、「Pentium互換」を動作条件に掲げているものが多い。これらのディストリビューションを使いたくても、486マシンは門前払いされてしまうのか？ 念のため、確認しておこう。

使用したのは、TurboLinux 4.5だ。今月号の付録CD-ROMにも収録されている。

結論からいえば、TurboLinux 4.5は、問題なくAptiva 530にインストールできた。X Window Systemをハイカラー（16ビット）以上に設定できなかったが、これはグラフィックスチップの問題だ（VRAMは1Mバイト搭載

ディストリビューション	CPU	メモリ
TurboLinux 4.5	Pentium互換	32Mバイト以上
Official Red Hat Linux 6.1 改訂 日本語版	Pentium互換	16Mバイト以上
LASER5 Linux 6.0 Rel.2	i486互換	16Mバイト以上
Vine Linux 1.1 CR	i486DX66以上	32Mバイト以上

表2　主なRPM系ディストリビューションの動作条件

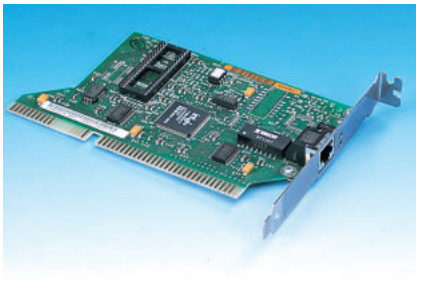
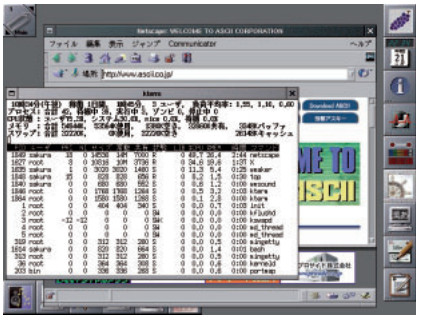


写真5　インテルEtherExpress PRO/10。10BASE-T対応のネットワークカード。

画面2　Vine Linuxインストーラ画面。とりあえず「すべて」でインストールしてみたのだが…。



画面3　Window Maker。Vine Linuxの標準ウィンドウマネージャ。486マシンには荷が重い…。

画面4　fvwm。見た目がシンプル。ktermをいくつか開いての作業なら十分に使用可能。

されているので、640×480モードならハイカラー表示できるはず)。

ウィンドウマネージャは、Turbo Linux標準のAfter Stepを用いてみたが、Vine Linuxでfvwmを動かしたのと同程度の使用感だった(**画面5**)。つまり、ktermでいくつかコンソールを開いて作業する分には、何とか使えるというレベルだ。「Pentium互換」を要求するディストリビューションだからといって、特に動作が重いということもないようだ。

同じクロックで比較して、Pentiumが486より速いのは、命令を実行するパイプラインを2本持っており、簡単な命令については、2つ同時に実行できるからだ。Pentium用に最適化をかけたプログラムは、同時に実行できる命令の対が並ぶように工夫されているわけで、Pentium専用の命令を使っているわけではない。実際、CPUの命令セットという点では、Pentiumと486にほとんど差はない。そのため、Pentium用に最適化をかけたプログラムが、486で動かないということは、まずないのだ。

テキストベースのクライアントPC

クライアントとして486マシンを使うなら、テキストベースの作業を中心に行えばよいだろう。実際、メールのやりとり、ネットニュースの購読、文書作成など、コンピュータを使う作業は、テキストが扱えれば済むものが多いのだ。

fvwmなど負荷の小さいウインドウマネージャを用いるのもよいが、ここでは以下の理由から、コンソールで利用することをお勧めする。

●仮想コンソールを用いれば、複数のコンソールを切り替えて使用できる。
●メモリ必要量が少なくてすむ。
●カーネル2.2のフレームバッファコンソールに対応していれば、高解像度で利用できる。

なお、フレームバッファコンソールでは、konが動作しないが、桝一氏の作製したJFBTERM/ME(**画面6**)を使

画面5　TurboLinux。After Stepのデスクトップ。壁紙だけを取り替えたのではなく、ちゃんとTurboLinuxをインストールしたものだ。

Column

Vine Linux 1.1をカーネル2.2化する

今回用いたVine Linux 1.1は、カーネル2.0.36を採用している。2.2に置き換えるためには、カーネル以外にもいくつかのコンポーネントを更新する必要がある。Ring Server ProjectなどのFTPサイトには、「kernel-2.2.x-kit」のような名前でまとめられている(画面c-1)。

まずこれらのファイルを取得してきて、

```
rpm -Uvh dhcpcd-1.3.16-0.i386.rpm
```

のようにして、アップデートしよう。

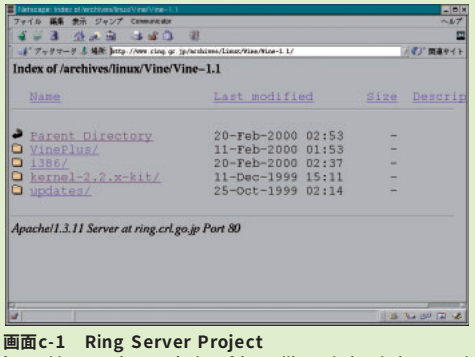
その後、カーネルソースを展開し、フレームバッファコンソールを有効にする。続いて、

```
# cd /usr/src/linux
# make xconfig（Xの場合）
```

または、

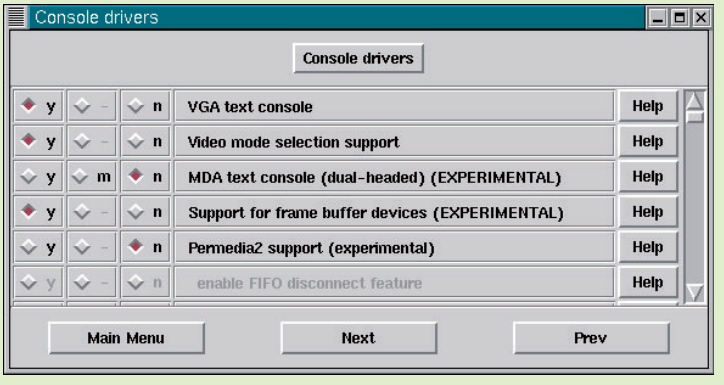
```
# make menuconfig（コンソールの場合）
```

として、カーネルの設定を行おう。「Console drivers」を開き、「Support for frame buffer devices (EXPERIMENTAL)」の項目を「Y」にする(画面c-2)。結果を保存して、カーネル設定を終了する。その後のコンパイル作業については、FMV DESKPOWERの項(88ページ)に詳細に書かれているので、そちらを参考にして欲しい。



画面c-1　Ring Server Project
http://www.ring.gr.jp/archives/linux/Vine/Vine-1.1/

画面c-2
カーネルの設定



うことで、日本語表示が可能になる。桝一氏のWebサイト（http://www3.justnet.ne.jp/~nmasu/linux/jfbterm/indexn.html）から、tarボールを入手可能だ。

ネットワーク端末として使う

もし486マシンのほかにも、もうちょっと高性能なPCを所有していて、そちらにもLinuxが入っているならば、486マシンをネットワーク端末として用いるといいだろう。以下の文章では、486マシンをクライアント、高性能マシンのほうをサーバと表記する。

ネットワーク端末といっても特別なものではなく、サーバにtelnetで接続して、作業を行うということだ。ディスプレイが用意できるなら、サーバを自室に置き、居間にあるクライアントからサーバを操作するといった使い方も可能だ。

●サーバの設定

上記のような使い方をする際には、サーバがtelnet接続を許可するように設定されているか、確認しよう。サーバ上では、クライアントからのtelnet接続要求を処理するプログラムが動作している必要がある。このようなプログラムをデーモン（daemon）と呼ぶ。LinuxやそのほかのUNIXでは、telnet以外にも、FTP（ファイル転送）、HTTP（Webサーバ）、cron（定期的なプログラム実行）など多くのデーモンがある。起動中のデーモンは、psコマンドで確認できる（**リスト1**）。

これら多数のデーモンプログラムを常に動作させておくのは、メモリの無駄使いになってしまう。そこで、使用頻度がそれほど高くないと思われるデーモンについては、インターネットスーパーサーバデーモンと呼ばれるinetdを動作させておき、クライアントから要求があったときに、inetdがそれぞれのデーモンを起動するように設定するのが一般的だ。

inetdから呼び出すデーモンのリストは、/etc/inetd.conf（**リスト2**）というファイルに記述する。まずsuコマンドでroot権限を得て、このファイルをテキストエディタで開き、「telnet」の記述がある行を見つけよう。行頭の「#」は、コメントとみなされるので、付いていたら消して、inetdがtelnetデーモンを呼び出せるようにする。変更後、ファイルを保存したら、コマンドラインから以下のように入力して、inetdの設定ファイルを再読み込みさせる。

```
# killall -HUP inetd
```

これで、サーバ側の準備は完了だ。

●クライアント

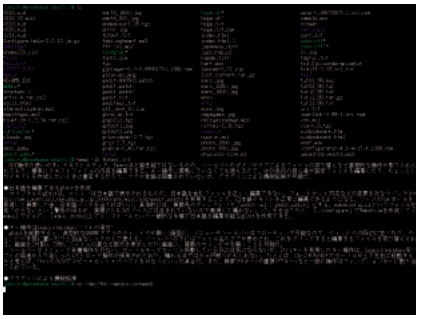
クライアントでは特に何もする必要がない。サーバ側の準備が終わっていれば、

▼リスト1　psコマンドで起動中のデーモンを確認。「ax」オプションを付ける必要がある

```
$ ps ax
  PID TTY      STAT   TIME COMMAND
                        ≀
  187 ?        S      0:00 syslogd -m 0
  194 ?        S      0:01 klogd
  203 ?        S      0:00 /usr/sbin/atd
  212 ?        S      0:00 crond
  222 ?        S      0:00 inetd
  241 ?        S      0:01 /usr/bin/jserver
  262 ?        S      0:05 sshd
  267 ?        S      0:00 /usr/sbin/cannaserver -syslog
  277 tty1     S      0:00 login -- root
                        ≀
  293 tty1     S      0:00 -bash
  302 ?        S      0:05 in.telnetd
  303 ttyp0    S      0:00 login -- sakura
  304 ttyp0    S      0:01 -bash
  785 ttyp0    R      0:00 ps ax
$
```

▼リスト2　/etc/inetd.conf（抜粋）

```
# These are standard services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#
```



画面6　JFBTERM/ME。1024×768の解像度で使用中。

```
client$ telnet （ホスト名）
```

と入力して、ログインプロンプトが出たら、ユーザー名とパスワードを正しく入力する。

```
Trying 192.168.1.1...
Connected to
server.foo.ascii.co.jp.
Escape character is '^]'.

Vine Linux 1.1CRW (Rheingau)
Kernel 2.0.36 on an i486
login:        ←ユーザー名
Password:     ←パスワード
Last login: Thu Feb 14 13:40:02
from client
server$
```

プロンプトがサーバマシンのものになったら、成功だ。なお、ほとんどのディストリビューションで、rootユーザーのリモートログインはできないように設定されている。基本的に、一般ユーザーで作業をし、rootユーザーの権限が必要な時だけsuコマンドを使うようにしよう。

フロッピーLinuxで静音マシンに

慣れてしまうとそれほど気にならないが、コンピュータはわりと騒がしい機械だ。騒音の主因はファンとハードディスクだ。最近の高性能マシンには、電源、CPU、ハードディスク、グラフィックスカード…、といたるところにファンがついているうえに、ハードディスクも性能が上がるにつれてにぎやかになってきている。

Aptiva 530の場合、CPUファンがないのでファンは電源内部のものだけだが、古いハードディスクのシーク音がなかなかにぎやかで、やはり静かな場所では気になってしまう。

居間にクライアントPCがあれば便利だが、うるさいコンピュータを置くと家族の目が冷たい、ということもあるだろう。

そんなときは、いっそのことハードディスクなしで使ってはどうだろう？Windowsのように巨大なうえに中身が分からないブラックボックス的なOSは、ハードディスクなしで使うことなど不可能だ。しかしLinuxはソースが公開されているため、必要のない機能をばっさり削って、小型化することが可能だ。実際、基本的な機能をフロッピーディスク1枚に収めるというプロジェクトが、数多くある。フロッピーディスクから起動可能なカーネルを構築し、できるだけ多くの機能を追加していくというパズルにも似た作業は、ハッカー諸氏の心を揺さぶるようだ。

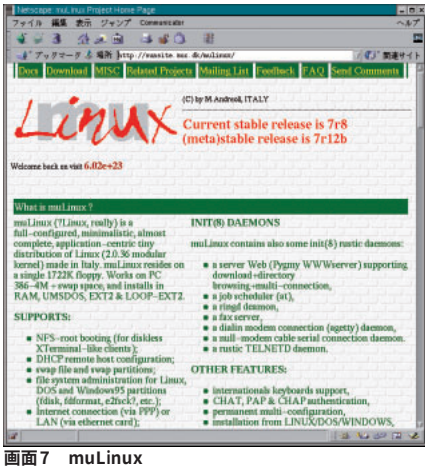
muLinux

1フロッピーLinuxの中で、今回のクライアント用途に適していると思われるのが、このmuLinuxだ。muLinuxは、82トラック、21セクタという特殊なフォーマットで記録された1.722Mバイトのフロッピーディスクに、非常に多くの機能が詰め込まれている（**表3**）。

muLinuxは、tarボールで提供されている。Webサイト（http://sunsite.auc.dk/mulinux/）から、DOS上で使うインストーラともども入手しよう。特殊なフォーマットもインストーラが行ってくれる。完成したフロッピーディスクで起動し、設定項目に答えていくと、そこはもうLinuxの世界だ。

muLinuxの特徴は、機能を追加する拡張フロッピーディスク（ADD ON）が提供されていることだ（**表4**）。SCSIサポートやFTPサーバ、Sambaが使えるようになる「EXT」、X Window Systemが使えてしまう「X11」、リモートディスプレイシステムの「VNC」などがある。フロッピーディスクだけでここまでやるのか…、といった感じがする。

Aptiva 530のハードディスクを外し、muLinuxを起動してみたが、当然ハードディスク付きのときとは比較に



画面7　muLinux

クライアント	FTP、telnet、NFS、IRC、 DHCP、Web、NNTP、SMTP、POP3
ネットワークカード	3c509、3c59x、ne、ne2k-pci、wd、smc-ultra
そのほか	vi、IPマスカレード、PPP、nmap、 cron、FAX送受信、MP3再生

表3　muLinuxの機能

名前	機能
EXT	SCSIサポート、FTPサーバ、Samba、CD-R、PCMCIAサポート
X11	Xサーバ（VGA）
VNC	リモートディスプレイシステム
GCC	gcc
TCL	Tcl/Tkプログラム

表4　muLinuxのADD ON

ならないほど静かだ。これなら、居間に置いても邪魔者扱いされることもないだろう。ネットワーク端末として用いるなら、muLinuxでほとんど不便はない。106キーボードにきちんと対応できていないのだけが、残念なところだ。

そのほかの1フロッピーLinux

muLinux以外にも、さまざまな特徴を持った1フロッピーLinuxのプロジェクトが存在する。

●Linux Router Project

名前が示すように、ルータ用途に特化した小型のディストリビューションである。Ethernet以外に、ISDNやモデムのダイヤルアップ接続にも対応し、幅広い分野で応用できる。基本的なパケットのフィルタリングやIPマスカレードも可能で、普及が期待されるケーブルTVやxDSLによる高速な常時接続環境にも最適だ。これらの機能は、モジュール化されており、必要に応じて組み込み／取り外しが行える。

またSNMPに対応しており、負荷の監視が行えるほか、次世代のIPv6にもアップグレード可能であり、長期間にわたっての運用が期待できる。比較的小規模なLAN環境を、低コストで構築できることから、システムインテグレータの採用実績もあるようだ。

●Tom's root/boot

緊急時のレスキューディスクを主用途とした1フロッピーディストリビューションである。muLinuxと同じ、1.722Mバイトのフロッピーディスクを用いている。10種類以上のファイルシステムに対応し、SCSIアダプタもAdaptec社製品を中心にサポートされている。マシンが数台あるような環境でも、このフロッピーディスクを1枚持っておけば、対処できるだろう。

深刻なハードディスクのトラブル以外にも、ブートローダの設定ミスでLinuxが起動しなくなるといった事態は、起こりがちだ。保険のつもりでTom's root/bootを用意しておくといいだろう。

分散コンピューティングへの参加

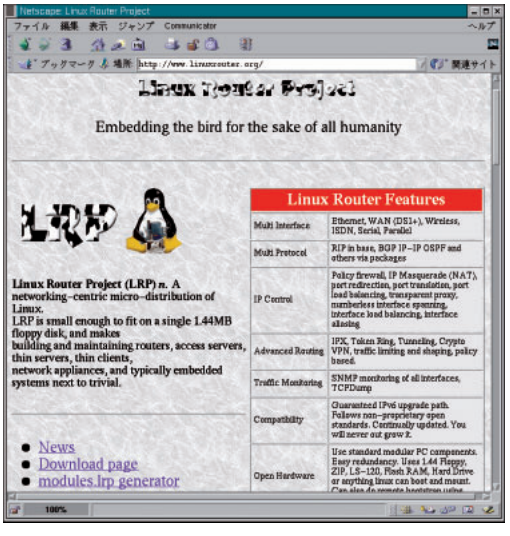
ここまで紹介した、軽量クライアント的な用途では、使うときになって電源スイッチを入れるのでは、使い勝手がよくない。必要なときにすぐに使えるようにしておきたいものだ。また、Linux Router Projectのフロッピーディスクなどを利用して、ルータとして運用する際にも、電源は常に入れたままである。

この状態ではほとんどの時間、CPUは使われていないことになる。xosviewのようなリソースメーターでこのことを知ってしまうと、なんとなくもったいない感じがするのは、筆者だけではないはずだ（**画面10**）。何とか有効に活用する方法はないだろうか。

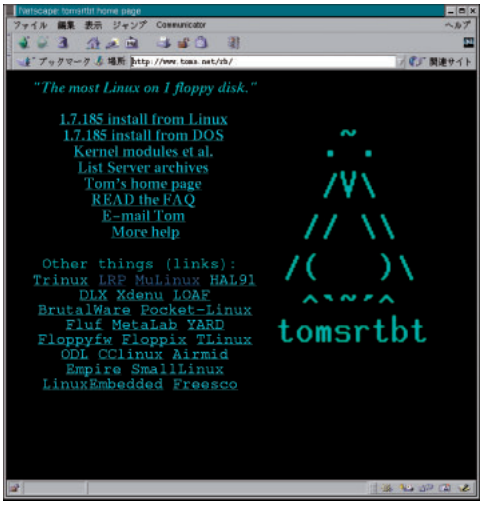
もしCPUの空き時間がもったいないと感じたら、「分散コンピューティング」のプロジェクトに参加するといいだろう。分散コンピューティングとは、ひたすら計算量が多くて、個人や一企業では事実上不可能な作業を、世界中のコンピューターの空き時間を使って片付けようというものだ。インターネットの普及により、このような手法が使えるようになったといえよう。

●distributed.net

インターネットを使った分散コンピューティングの草分け的存在が、distributed.net（**画面11**）だ。暗号の



画面8　Linux Router Project



画面9　Tom's root/boot

秘密鍵発見コンテストでの勝利を、何回かものにしている団体だ。

インターネットが普及し、その利用法が多様化していくにつれて、データを他人から隠蔽することが必要になってきた（クレジットカードの暗証番号を他人に知らせたい人は、いないはず）。

そのためにデータの暗号化が必要になってくる。元のデータ（平文）と「鍵」と呼ばれるデータとを組み合わせて、複雑な処理を行うのが暗号化だ。鍵なしで暗号化されたデータから平文を再現するのはほぼ不可能だ。distributed.netは、これに対してすべての鍵を試してみるという「力技」で挑戦している。当然、鍵は長いものほど発見が困難になる。

distributed.netは、今までに鍵の長さが56ビットのDES、およびRC5という暗号の鍵を比較的短時間で発見することに成功してきた。2000年2月現在、64ビット長のRC5鍵を探すコンテストが継続中だ。

●SETI@home

SETIとは「Searching for Extra Terrestrial Intelligence」の略で、SETI@homeは、電波望遠鏡で得られたデータを、世界中で解析して地球外生物からの電波を発見するというプロジェクト（**画面12**）だ。「宇宙人を探す」という夢のある目的のせいか、参加者が殺到し、サーバがダウンしたことが報道されたのを覚えている方もいるだろう。

プロジェクトは1999年5月に始まり、2年間続けられる予定だ。電波望遠鏡が受信したデータにデジタル信号処理を行い、周波数帯ごとの強度を測定している。この非常にCPUパワーを要求する作業を、世界中で分担して行っている。2000年2月現在の、参加者は約175万人だ。（distributed.netは、約24万人）

●継続は力なり

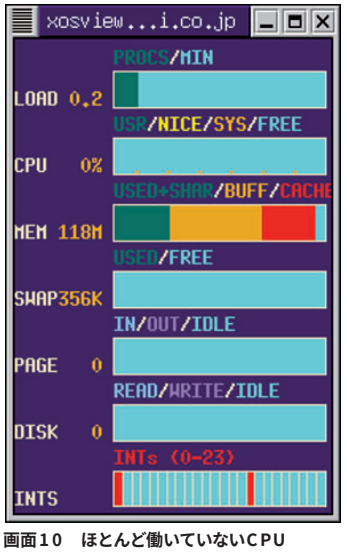
これらのプロジェクトでは、いくらでもCPUパワーを必要としている。常時稼動しているマシンがあったら、参加してみよう。

「486マシンではパワーが足りない」と思うかもしれない。確かに、最新のPentium IIIあたりと比較すると、486マシンは、一桁小さい結果しか出ない。でも1日に2時間しか使わないPentium IIIマシンと、24時間稼動している486マシンだったら、結果は同じだ。毎日ずっと動かしていくことが大事なのだ（**画面13**）。

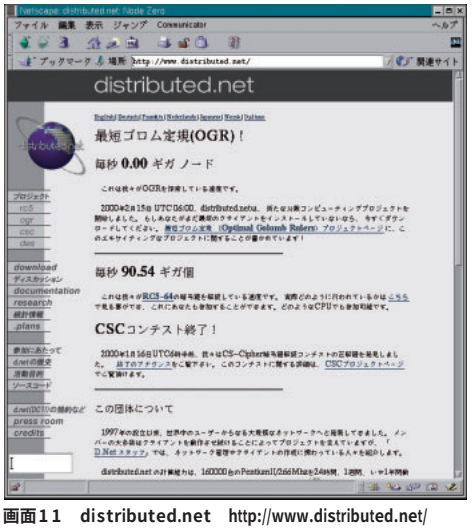
それに、distributed.netの秘密鍵にしても、SETI@homeの宇宙人探しにしても、「当たる」確率は非常に小さいものなのだ。だから遅いマシンを使っていたとしても、「当たりにくさ」は変わらないといってもいいくらいだろう。必要なのは、「運の良さ」なのかもしれない。

もし地球外文明からの信号を見つけるのが、あなたの「クラシックな486マシン」だったとしたら、それはちょっとすてきなことかもしれない。

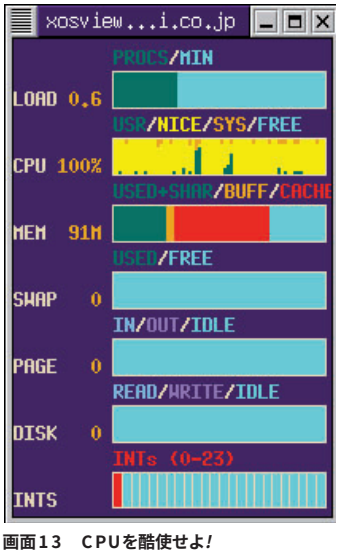
画面12　SETI@home
http://www.vacia.is.tohoku.ac.jp/~s-yamane/articles/setiathome/home_japanese.html



画面10　ほとんど働いていないCPU



画面11　distributed.net　http://www.distributed.net/



画面13　CPUを酷使せよ！

Linuxで蘇れ、PC-98～Plamo Linux 98

文：山岸典将　Text : Norimasa Yamagishi

今でこそ、日本のパーソナルコンピュータ市場はIBM-PC互換機で溢れているが、（ハードウェアではなく、本来の意味でのOSの）DOS/V登場以前に、日本でパーソナルコンピュータといえば、日本電気のPC-9800シリーズ（以下、PC-98）だった。現在では、現役を退きつつあるPC-98だが、Linuxを使うのであればまだ十分なパワーがありそうだ。ここはひとつ、Plamo Linuxで、PC-98を甦らせてみよう。

Plamo Linuxとは

Plamo Linuxは、こじまみつひろ氏がSlackwareをベースに日本語化を行ったディストリビューションだ。このPlamo Linuxに、京大マイコンクラブのメンバーがPC-98用に移植したLinuxカーネル（Linux/98）を組み込んだのがPC-98用のPlamo Linux（以下Plamo 98）である。Debianで「Linux/98」を動作させているという話もあるが、現時点ではPC-98対応を宣言しているディストリビューションは、Plamo Linuxしか存在しない。

Plamo LinuxやSlackwareは、最近Linuxに関わり始めたユーザーにとっては、なじみが薄いかもしれない。Slackwareは、現在メジャーなRed Hat LinuxやTurboLinuxよりも歴史の古いディストリビューションであり、日本でも「LinuxといえばSlackware」という時代があった（このあたりはPC-98に似ている？）。しかし、グラフィカルなインストーラやパッケージ管理システムを用意した新しいディストリビューションに比べると、Slackware系のディストリビューションは初心者にはとっつきにくかったのか、現在は普及しているとは言いがたい。

しかし、Plamo Linuxは、日本人が日本語を使用することを前提に作ったディストリビューションだけあって、インストーラの日本語化はもちろん、Muleや日本語に対応したTeXなど、日本語を使うためのソフトウェアがちゃんとインストールされるようになっている。また、日本語のドキュメントとそれらの検索システムnamazuもインストールされる。このように、日本人にとっては使いやすいものになっている。また、パッケージ管理システムがないということは、Linuxの素の部分がよく見えるということでもあり、Linuxを学ぶには向いているともいえよう。そして、機能をしぼったサーバとして利用するならば、ほかのディストリビューションのような豊富すぎるソフトウェア群は必要ない場合もあるだろう。そういう意味では、Plamo/98は十分に試してみる価値のあるディストリビューションだ。なにより、もうお役御免となったかと思われたPC-98が活用できるのだ。

現在最新のPlamo Linux安定版はバージョン1.4、Linux/98の最新版はRev.0.20となっている。今回はPlamo 98の一次配布元であるFTPサイト、ftp://ftp.linet.gr.jp/pub/Plamo/で配布されているPlamo 1.4、Linux/98 Rev.0.15をもとに解説を進める（**表1**）。

カーネル	PC互換機用2.1.57ベース
パッケージ	Slackware 3.4ベース
日本語環境	PJE 0.1.1ベース
X	XFree86 3.3.3.1

表1　Plamo 98 1.4のスペック

写真1　今回Plamo 98をインストールしたPC-9821 V13

PC-98とPC互換機の違い

まず、必要とされるPC-98のスペックだが、80386以上のCPUと3.6Mバイト以上のメモリを搭載した機種となっている。Windows 95が動作するハードウェアなら間違いなく動作するし、Windows 3.1が動作している機種でもほぼ動作するだろう。もちろん、これは最低条件で、快適に使用するためには、それなりの性能が必要となる。

Plamo/98では、基本的にはPC互換機のバイナリプログラムをそのまま使うことができる。しかし、ハードウェアに強く依存したX Window System

作者のこじま氏が公開している「Plamo LinuxのWebページ」 http://www.linet.gr.jp/~kojima/Plamo/
京大マイコンクラブが公開している「Linux/98」 http://www.kuis.kyoto-u.ac.jp/~kmc/proj/linux98/

表2　Plamo/98の情報源、それぞれに、メーリングリストも用意されている。

項目	仕様
CPU	Pentium（P54C） 133MHz
メモリ	40Mバイト
ハードディスク	1.2Gバイト
ビデオ	Cirrus Logic CL-GD5440
ネットワークボード	LGY-98

表3　PC-9821 V13のスペック

（XFree86）や、PC-98独自のハードウェアであるCバス（本体の裏からカバーを外さずに差し込む形で使用する拡張スロット）用ハードウェアなどを利用する際には、PC-98専用のソフトウェアを使う必要がある。ただし、PCIバス経由で接続するハードウェアに関しては、PC互換機と同様にカーネルがサポートしていれば使えるようだ。

今回はインストールターゲットとして、PC-9821 V13（VALUESTAR）を用意した（**写真1、表3**）。

インストール

インストールの前に、まずはPlamo 98を手に入れよう。入手からインストールまで、順を追って解説する。

●Plamo 98の入手

Plamo Linux 1.4xのPC-98版は、前述のとおり、FTPサイトftp://ftp.linet.gr.jp/pub/Plamo/Plamo-1.4.x/で入手できる。インストール時点で必要なのは、この中の「98（約56Mバイト）」「plamo98（約183Mバイト）」の2つのディレクトリ以下の中身だ。そのほか、「Update」ディレクトリにはリリース後に見つかったバグへのパッチ、「contrib」ディレクトリにはPlamo Linuxで動作するさまざまなソフトウェアが置かれている（**画面1**）。

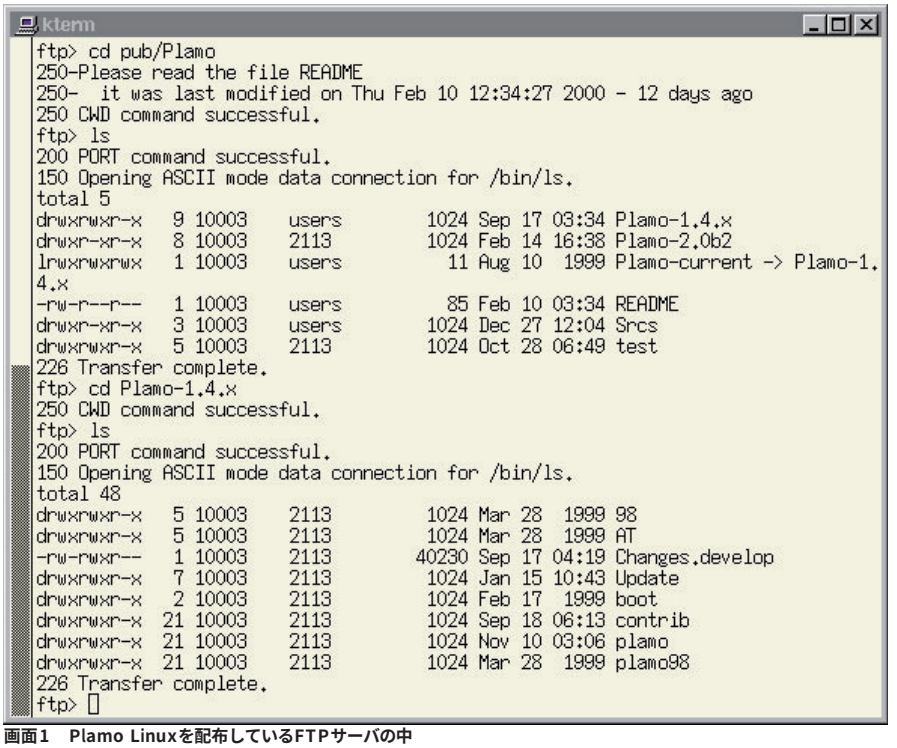
さて問題は、ここから取ってきたPlamo 98をどのようにしてインストールするかだ。Plamo 98は、CD-ROM、ハードディスクのDOSパーティション、NFSといったインストールメディアをサポートしている。今回は、FTPで取ってきたデータをCD-Rに焼いてインストールすることにした。「98」、「plamo98」ディレクトリをそれぞれCD-Rのルート直下に配置してCD-Rを作成する。

FTPで大きなファイルを取ってこられない場合などは、Linux4u.net（http://www.linux4u.net/）のようなCD-Rによる配布サービスを利用するとよいだろう。

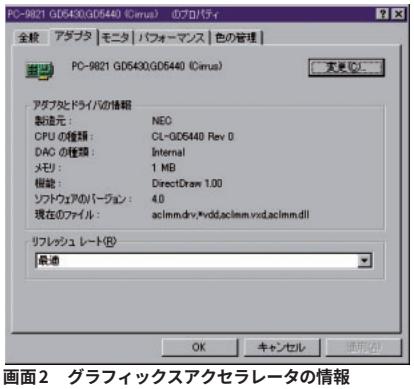
●事前の計画および準備

Plamo 98を利用する場合、ハードディスクにPlamo 98だけをインストールしてしまってもよいのだが、最初はWindows 95／Windows 98と共存させることをお勧めしたい。その理由は、Windowsを使うとPlamo 98のインストールに必要なデバイス情報を取得しやすいからだ。もっとも、PC-98にはPC互換機におけるFIPSのような、パーティションの再編成を簡単に行えるツールがないので、Linuxをインストールする空きパーティションが確保できない場合には、Windowsの再インストールが必要になるのだが、トラブル解決のためには、Windowsが入っているほうがよいだろう。また、パーティションを決定する際には、Plamoがどれだけのディスクを必要とするかも考える必要がある。Plamoの勧めるパッケージセットを入れるには約300Mバイト、さらにswap領域に100Mバイト程度用意しておけばとりあえずは十分だろう。

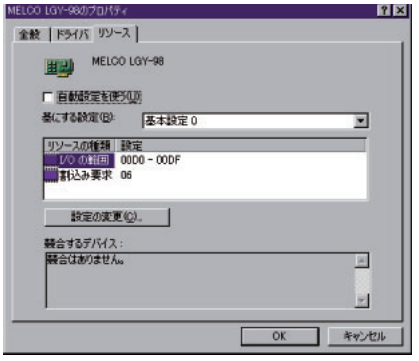
デバイス情報の中でもとくに重要なのが、グラフィックスアクセラレータと、ネットワークボードの情報だ。今



画面1　Plamo Linuxを配布しているFTPサーバの中



画面2　グラフィックスアクセラレータの情報



画面3　ネットワークボードの情報

回は、Windowsでこれらのデバイス情報を取得してみよう。

まず、グラフィックスアクセラレータだ。Windowsのコントロールパネルを開き、［画面］—［ディスプレイの詳細］—［詳細プロパティ］で［アダプタ］タブを選ぶと、グラフィックスチップの種類と搭載しているビデオメモリの量が確認できる（**画面2**）。Xを使わないのであれば、グラフィックスアクセラレータの情報は必要ないが、Xを使うならば、必ず調べておこう。

次はネットワークボードの情報だ。コントロールパネルの［システム］—［デバイスマネージャ］—［ネットワークアダプタ］を開いて、ネットワークボードの種類、およびそのIRQ（割込み要求）、I/Oポートアドレス（I/Oの範囲）を確認しておく（**画面3**）。

さらに、PC-98の電源ケーブルを抜き、ケースを開けて、それぞれのデバイスに使われているチップを確認しておけば確実だ（**写真2、写真3**）。

●インストール用ディスクの作成

いよいよインストールだ。最初に、インストールに使うブートフロッピーを作成する。ブートフロッピーは、1.25Mバイトと1.44Mバイトの2種類のフォーマットが使えるが、ネットワークを利用したり、PCMCIA（PCカード）を利用する場合は、1.25Mバイトフォーマットだとディスクが2枚必要になる。

ブートフロッピー作成プログラムは、MS-DOS（WindowsのMS-DOSプロンプトでもよい）で動作するもので、フォーマッタも付いているが、事前にWindowsでフォーマットしておいたほうが無難だろう。

ブートフロッピー作成プログラムは、先ほど作成したCD-Rの「98」ディレクトリの下の、「144」ディレクトリと「125」ディレクトリにあるバッチファイル「makeplfd.bat」だ。1.44Mバイトフォーマットなら「144」ディレクトリのものを、1.25Mバイトフォーマットなら、「125」ディレクトリのものを使う。次のように、MS-DOSプロンプトでCD-ROMドライブに移り、作りたいフォーマットのディレクトリでバッチファイルを実行する（**画面4**）。ここでは、CD-ROMがQドライブで、1.44Mバイトフォーマットのブートフロッピーを作成する例を示す。

```
A:¥WINDOWS>q:
Q:¥>cd 98¥144
Q:¥98¥144>makeplfd
```

今回は、1.44Mバイトフォーマットのブートフロッピーに「デスクトップ機用ミニLinuxシステム」を書き込むことにした。このシステムは、SCSIとネットワークをサポートし、これらのデバイスを介したインストール方法に対応している。なお、makeplfdは何度か失敗することがあった。理由は不明だが、一度失敗すると、その後も失敗することが多かったので、リセットをかけてからもう一回やり直したほうがよさそうだ。

●Linux用パーティションの作成

次に、ハードディスクにLinux用のパーティションを作成する。作成したブートフロッピーで起動すると、ブートローダ「GRUB/98」の画面が表示される。いくつかのネットワークボードを選べるようになっているが、今回はCD-Rからインストールするので、どれを選んでもかまわない。「インストールディスクへようこそ」のメッセージが表示されたあと、ログインプロンプトが表示されるのでrootでログインし、fdisk98コマンドを実行する。

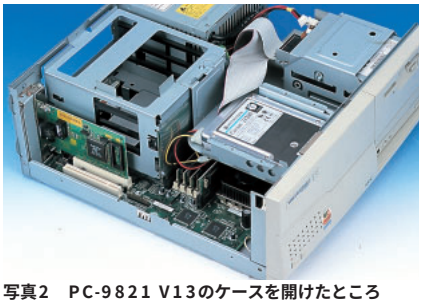


写真2　PC-9821 V13のケースを開けたところ
感電しないように電源コードを抜いておこう

写真3　グラフィックスアクセラレータはCirrus LogicのCL-GD5440だ

```
plamo login:root （rootでログイン）
# fdisk98
```

必要ないパーティションがあれば、dコマンドで削除し、nコマンドでLinux nativeのパーティションとswapパーティションをそれぞれ1つ作成する。もし間違ってしまったら、qコマンドで中止できる。wコマンドを使わない限り、パーティションの設定はハードディスクに書き込まれないので安心してよい。wコマンドで変更を書き込んだあと、#プロンプトに戻ったら、「CTRL」＋「GRPH」＋「DEL」キーを押してリセットしよう。

●実際のインストール開始

マシンが再起動したら、rootでログインしてsetupコマンドを実行し、実際のインストール作業に入る。インストール画面ではカーソルキーが使える。

まず、最初の画面でKEYMAPをPC-98用に設定するとよいだろう。これを設定しておかないと、キートップと実際に入力される文字が異なったものになってしまうからだ。

次に「Plamo Linuxのインストールを開始」を選び、「スワップパーティションの設定」を行う。swapパーティションの初期化では「メモリの少ないマシンにインストールする場合」の警告が表示されるが、6Mバイト以上のメモリがあれば大丈夫だろう。そして、「swap領域を使用可能にする」を選ぶ。スワップパーティションの初期化が終了すると、そのままインストールを進めるか、メニューに戻るかを聞かれるが、ここではインストールを進めよう。

そして、「インストール先ディスクの設定」に移る。インストールするパーティションを選択し、「スローフォーマット（バッドブロックの調査付き）」を選ぶ。i-node密度の選択ではデフォルトの4096を選べばよい。この後、Windowsが残っている場合は、Linuxから見えるようにしたいかを聞かれる。LinuxからもWindowsパーティションを利用する場合は、リストに表示されている「/dev/hda1」のようにWindowsパーティションを指定する。

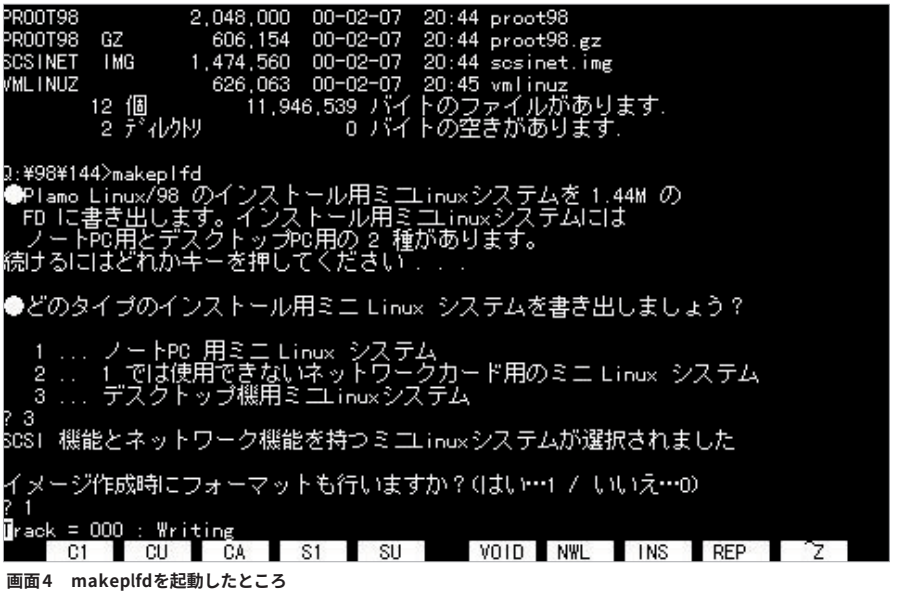
次は、「インストール元の設定」になる。今回はCD-ROMからインストールすることにしたが、CD-ROMドライブも問題なく自動認識された。

パッケージの選択では「お勧めパッケージセット」を選択すると、さらにノート用かデスクトップ用か、日本語入力にWnn、Cannaのどちらを使うかを聞かれる。以上を決定すると、ファイルのインストールが始まる。

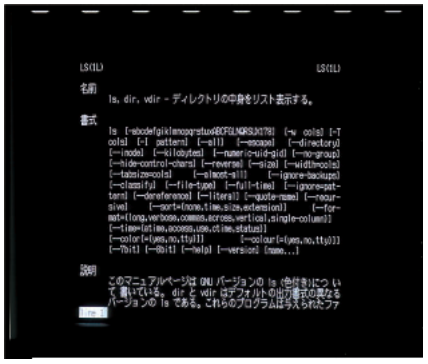
ファイルをインストールする際には、各パッケージの説明が表示される。インストールが進むと、Xサーバの選択を求められることになる。ここでは、XF98_NEC480という、PC-9821（一部のB MATEを除く）が標準装備しているPEGC用のXサーバ（640×480ドット、256色）を選択する。PC-9821以外にインストールするなら、XF98_EGCというEGC用のXサーバ（640×400ドット、16色）を選択すればよい。これらはXの設定プログラムXF98Setupを使う時に必要なので絶対に選んでおこう。さらに、自分のマシンで使われているグラフィックスアクセラレータ用のXサーバを選択する。もしわからなければ、それらしきものをすべて入れておけばよいだろう。最終的には、XF98Setupで詳細な機種名、ボード名で選択することができる。今回はCirrus Logic用のXF98_NKVNECを選択した。

このあと「カーネルのインストール」をどうするかを質問されるが、これは自動的に組み込まれているので、「skip」を選ぶ。続く「システム設定」では、ブートディスクの作成を行う。

そして、モデム、マウスの設定を行ったあと、PC互換機のLILOにあたる、grub98のインストールを行う。ただし、LILOとは違い、grub98がないとLinux/98は起動することができないので、必ずインストールする必要がある。「Linuxをインストールしたパーティシ



画面4　makeplfdを起動したところ



画面5　コンソールでman lsを表示した
konなどを使わなくてもそのままで日本語を表示できる

ョン」にインストールするのが無難だろう。MBRにインストールすると、固定ディスク起動メニューを壊してしまうからだ。次は「ネットワークの設定」を行う。ネットワークを使いたければ、ここで設定しよう。最後に「時間帯の設定」となり「Japan」を設定（それ以外は選べない）してインストールは終了する。

インストールが終了したら、マシンを再起動しよう。Windowsと共存させた場合は固定ディスク起動メニュープログラムが立ち上がるので、Plamo 98をインストールした領域を選択すると、さらにブートローダのGRUB/98が起動する。メニューの中に項目は1つしかないので、そのままリターンキーを押せばPlamo 98がブートする。

インストールが完了したら、まず行いたいのがrootのパスワードの設定だ。最近のディストリビューションでは、インストール中にパスワードを設定することが多いが、Plamo 98の場合、パスワードはインストール後にrootでログインしpasswdコマンドで設定することになる。

なお、PC-98は日本語を表示するためのハードウェアを持っているので、PC互換機のように、konを使わなくても、コンソールに日本語を表示できる（**画面5**）。コンソールモードの切替えは「vf・1（日本語EUC）」、「vf・2（raw）」、「vf・3（シフトJIS）」キーとなっている。JISコードはどのモードでも表示できる。rawモードは英語モードのようなものといえる。

基本的なシステムの設定

ここまでで、システムファイルのインストールが終わり、Linuxが起動するようになったわけだが、せっかくLinuxを使うのだから、X Window Systemとネットワークは利用できるようにしたい。

●X Window Systemの設定

PC互換機では「XF86Setup」というコマンドでX Window Systemを設定するのだが、PC-98では「XF98Setup」というコマンドを使う。ただし、このXF98Setupはマシンとの相性がかなりあるようだ。今回はPC-9821 V13以外に、PC-9821 Xa16でもテストをしたのだが、Xa16が内蔵しているTrident社製のアクセラレータチップを使った高解像度表示は実用にならなかった（**画面6**）。ただし、XF98Setupではうまく設定できなくても、Xの設定ファイルXF86Configを手作業で書き換えるとうまくいく場合もあるようだ。

そのため、まず、XF98Setupではなにも設定せずに、設定を完了することを勧めたい。そして、その結果作成されるXF86Configをバックアップしておき、再びXF98Setupで設定を行う。さらに、生成されたXF86Configを手でいろいろと書き換えてみるという方法をとるとよいだろう。**画面7**は、こうして設定したPC-9821 V13でのX Window Systemの画面だ。

また、設定に失敗しているとなにも表示されなくなってしまうことがあるが、そういう時はあわてずに「CTRL」＋「GRPH」＋「BS」キーを押してXサーバを終了しよう。リセットボタンは最後の手段だ。

●ネットワークの設定

X Window Systemが設定できたら、ネットワークの設定をしよう。PCIバスに接続するネットワークボードは特に設定をしなくても認識するようだが、Cバスに接続するネットワークボードを利用するにはカーネルの再構築が必要となる。その手順を紹介しよう。まず、カーネルのソースを展開する。

```
# cd /usr/src
# tar zxvf lx98-b15.tgz
```

次に、カーネルの構成を決めるわけだが、慣れていないのならば、用意されているテンプレートを元に、必要な

画面6　PC-9821 Xa16のXの高解像度表示
マウスカーソルの残像が残るなど、表示が乱れている。

画面7　PC-9821 V13のXの高解像度表示

ところを変更することをお勧めしたい。ここでは、config.scsi55というファイルをテンプレートにする。このファイルは、日本電気製の55ボード互換のSCSIボードを使うための設定ファイルだ。実際にはSCSIを使っていなくても心配はいらない。そのまま利用しても特に問題は発生しないし、気になるなら構成メニューの中でSCSIボードを無効にすればよい。

```
# cd linux
# cp Configs/config.scsi55.config
# make menuconfig
```

カーネルの構成メニューから「Network device support」、「NEC PC-9800 C-Bus Ethernet Board」を「Y」キーで選ぶ。今回の例では、メルコのLGY-98というネットワークボードを使うので［Melco LGY-98］も選んでおく。設定を保存し終了したら、makeする。

```
# make dep ; make clean ; make
zImage
```

この結果、/usr/src/linux/arch/i386/bootにzImageというファイル名のカーネルができるので、これをルートディレクトリにvmlinuz.newというファイル名でコピーしよう。

```
# cp arch/i386/boot/zImage /vmlinuz.new
```

新しいカーネルが準備できたら、マシンを再起動して新しいカーネルをテストすることになる。ブートローダGRUB/98上で、パラメータを編集して起動しよう。まず、カーネルのファイルを指定し、ネットワークボードを有効にするために以下のパラメータを加える。

```
ether=6,0d00,2,eth0
```

それぞれ、6はIRQ、0d00はI/Oポートアドレス、2はボードの種類、eth0はデバイス名を示している。ボードの種類については、**表4**を参照し、決めてほしい。したがって、全体としては以下のようなパラメータで起動することになる。

```
kernel = /vmlinuz.new root=/dev/hda2
ro  ide=serialize  wd33c93=level2:1
ether=6,0d00,2,eth0
```

新しいカーネルで、問題なく動作するようだったら、ブートローダそのもののメニューを書き換えてしまおう。メニューは/boot/grub/menu.lstというテキストファイルになっている。もし、ネットワークボードを認識しない場合は、Windows上でIRQ、I/Oポートアドレスを変更してみるとうまくいく場合がある。

なお、Plamo 98はセキュリティを考えて、初期状態ではほかのマシンからのネットワークによるアクセスをすべて拒否する設定になっている。ほかのマシンからアクセスしたい場合には、/etc/hosts.allowを修正すればよい。たとえば、

```
ALL : LOCAL
ALL : 192.168.5.
```

と書けば、192.168.5から始まるIPアドレスのアクセスを許可する設定になる。

これで、一般的なディストリビューションのインストーラで用意される環境が、ひととおりできあがったことになる。

アプリケーションのインストール

今度は、Plamo 98で動作するアプリケーションについて見ていこう。

Plamoの場合、他の多くのディストリビューションでみられるように、インストール時点でさまざまなアプリケーションがインストールされるということはない。必要なアプリケーションは、自分でインストールすることになる。前述の「contrib」ディレクトリの中にPlamo用に調整されたパッケージが用意されているので、その中からいくつかのアプリケーションをインストールしてみよう。今回はGimp、Netscape Communicator、Sambaの3つのアプリケーションをインストールしてみた。Slackware系のソフトウェアのインストールは基本的にinstallpkgコマンドで行う。

●Gimp

まずは、画像処理ソフトのGimpをインストールする。rootになりinstallpkgコマンドで、gtkとGimpをインストールするだけなので非常に簡単だ。

```
# installpkg gtk.tgz
# installpkg gimp.tgz
```

メーカー	種類	
アイ・オー・データ	LA-98	1
メルコ	LGY-98	2
マクニカ	NE2098	2
メルコ	EGY-98	3
ICM	27xET	4
CONTEC	CNET-98/EL	5

表4　Linux/98で使えるCバス用ネットワークボードの種類と指定する番号

以上でインストールは終了だ。Gimpは数多くの機能を持つ強力なフォトレタッチソフトウェアなので、ぜひ、試してみてほしい（**画面8**）。

contribの中のmemo.txtというドキュメントの中には、gimpの起動時に設定ファイル用のディレクトリが作れないため、エラーが発生する可能性があることと、その対策が書いてあったが、今回はそのようなエラーは起きなかった。もし「undefined font type」というエラーが発生したらドキュメントを読んでみよう。

●Netscape

Netscape Communicatorは、contribディレクトリに用意されているが、installpkgコマンドではなく、自分でtarボールを展開し、インストールする必要がある。

```
# tar xvzf communicator-v407-export.x86-unknown-linux2.0_libc5.tar.gz
# cd communicator-v407.x86-unknown-linux2.0
# ./ns-install
```

次に江後田氏が作成したNetscape communicatorの日本語化キットをインストールする。

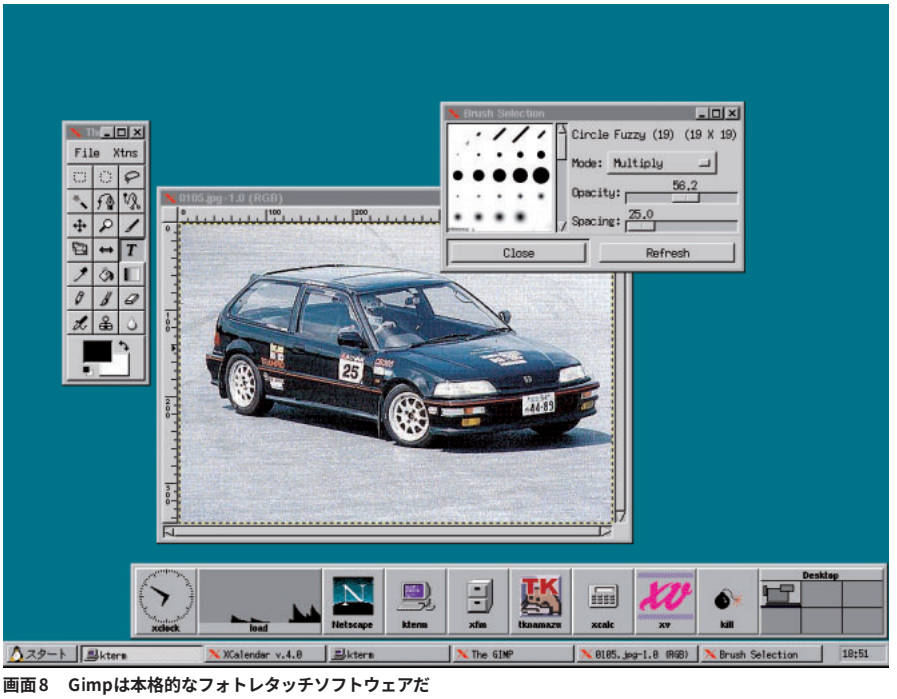
```
# tar xvzf /cdrom/contrib/Netscape/communicator-ja-v407-export.x86-unknown-linux2.0.tar.gz
# cd netscape-ja
# ./install-ja.sh
```

このままだと、libg++.so.27がないといわれてしまうので、ライブラリをリンクする。

```
# ln -s /usr/ix86-linux/lib/libg++.so.27 /usr/lib/libg++.so.27
```

さらに、/usr/local/netscape/jcommunicatorを/usr/local/bin/netscapeにリンクしておくとよいだろう。

```
# ln -s /usr/local/netscape/jcommunicator /usr/local/bin/netscape
```



画面8　Gimpは本格的なフォトレタッチソフトウェアだ

これで、日本語化されたNetscape Communicatorが使えるようになる（**画面9**）。

実際に、長期にわたって使用したわけではないが、安定性、速度に関して、同スペックのPC互換機との差異は体感できなかった。

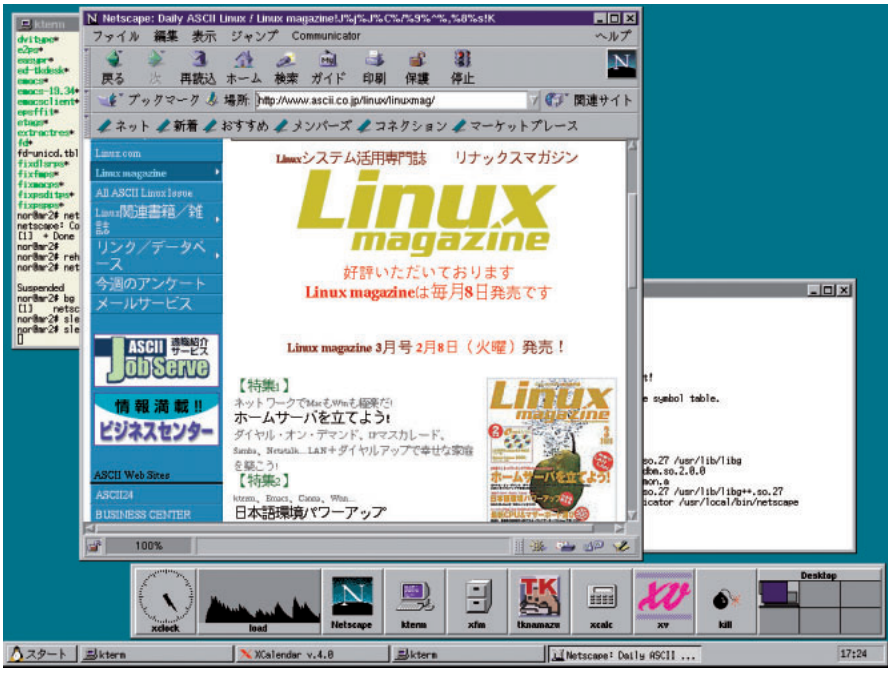
●Samba

Sambaはcontribディレクトリにも用意されているが、今回は、日本Sambaユーザ会のWebページ（http://www.samba.gr.jp/）で公開されている最新のSamba日本語版2.0.5aJP2をソースからコンパイルしてみた。

```
$ cd samba-2.0.5a/source
$ ./configure --prefix=/usr
--libdir=/etc
--with-lockdir=/var/lock/samba
--with-privatedir=/etc
--with-swatdir=/usr/share/swat
--with-smbwrapper --with-automount
--with-quotas
$ make
$ su
# make install
```

Sambaユーザ会のWebページに書かれているコマンドとの違いは、PAMパスワードを有効にするための--with-pamオプションを付けないことだ。**画面10**は、SambaサーバであるPlamo 98マシンに、Windows 98のエクスプローラでアクセスしたところだ。通常のWindowsマシンと同様に見えているのがわかるだろう。

ここでは、3つのアプリケーションをインストールしてみたが、インストールする時に注意したいのが、Plamoではそれぞれのファイルの依存関係など



画面9　Netscape Communicatorも動作する

は管理しないということだ。すなわち、なにも考えずに、インストールやアンインストールを繰り返していると、必要なファイルが別のバージョンに上書きされてしまったり、消されてしまったりすることがありうる。installpkgコマンドでインストールされるパッケージは、普通のtarボールなので、事前にファイルの中身を見て確認しておくとよいだろう。

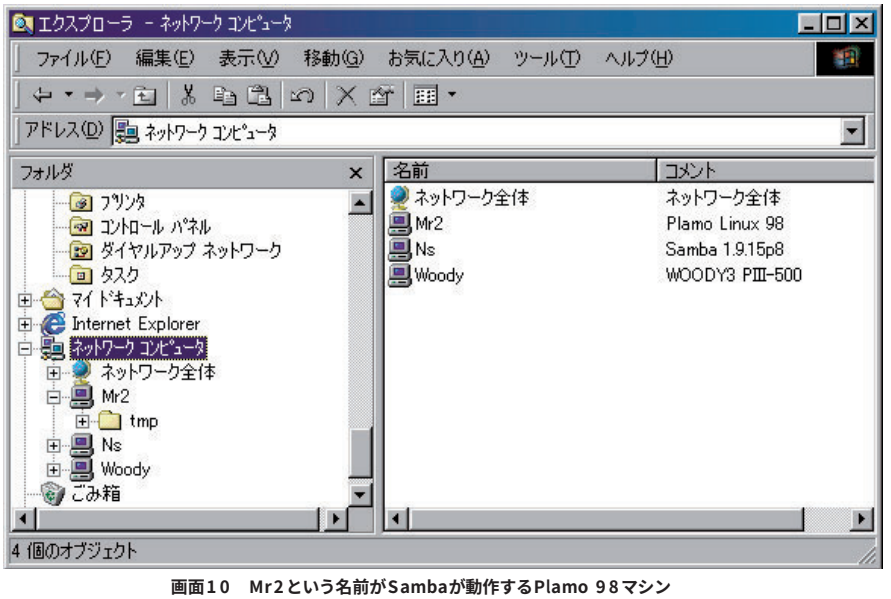
98は健在だ

今回、ネットワークボードを認識させるのに少々手間取った。Windowsでは、IRQが3で動作するのに、LinuxではIRQが6でないと動作しないという、結果だけを言うと簡単なことだった。一見ソフトウェア上では認識もされていて、動作しているように見えるのに、さっぱり通信ができないのだ。もっとも、Linux/98は開発中ということもあるし、なにより、日本のみで開発されているものだから、やむを得ないことなのだろう。そういったデバイス固有の問題を除けば、Plamo 98はPC互換機上のLinuxとなんの違いもなく動いている。パーソナルユースで使う分には、十分実用になるといえよう。

また、今回は取り上げなかったが、音源ボードやCバス用のモデムボードなどにも対応しており、サポートされるデバイスも徐々に増えつつある。もし余っているPC-98があるのだったらLinuxの学習用、また小規模ネットワークのサーバとしてぜひ使ってみてほしい。

実は、今回利用した、Linux/98のカーネルは若干古い。2月20日時点での最新バージョンはRev.0.20となっており、このバージョンでは今回あった不具合も解消されているかもしれない。2.2系、2.3系のカーネルの移植も進んでいるようだ。

また、Plamo 2.0βが昨年の12月1日に公開された。現在はPC互換機用のみの公開だが、カーネル2.2.10、XFree86 3.3.4、glibc-2.1.1に日本語Localeパッチを当てたものに、日本語化したGNOME、KDEが加わったものとなっている。また、PC-98用に関しても、Linux/98 kernel 2.2.12、glibc2.1.2で動作しているようなので、じきに公開されると思われる。



画面10　Mr2という名前がSambaが動作するPlamo 98マシン

Column

Plamo 2.0がやってくる PC-98用Plamo 2.0 β公開

原稿を書き終って、PlamoのWebサイトを見ていると、PC-98用のPlamoのβ版が公開されているのに気が付いたので、早速使ってみることにした。テストマシンは本文と同じくPC-9821 V13だ。

FTPサイトのディレクトリはPlamo-2.0b2となっているので、12月1日のβ1からバージョンが上がって、β2になったということだろう。

ドキュメントを読まずにいきなりインストールして、玉砕するというのもそれはそれで面白いのだが（編註：ダメです。）、βと銘打っているだけに未完成な部分も残っているはずだ。それを知っていてハマるのはばからしいので、まず、README.EUC、Changes.log、TODO.txtといったドキュメントに目を通すことにする。

PC-98用のPlamo 2.0βは、カーネル2.2.12pre002ベースで、glibc2はglibc-2.1.2に日本語localeパッチを当てたものを使用。XFree86のバージョンは3.3.6になっている。

●インストール

まず、Plamo 1.4と同様の手順で、ブートフロッピーを作る。ブートフロッピーで立ち上げると、メニューに表示されるネットワークボードも増えている。ただ、こじまみつひろ氏もすべてのネットワークボードを調べることはできないのだろう。新たにサポートされているボードについては、親切に（未テスト）の注意書きが付いていた。

インストーラは基本的には、Plamo 1.4のものと同じ。パッケージのインストール中に「本来あるべきパッケージが存在しない」というメッセージが出たほかは、1.4と同じ手順ですんなりと終了した。このメッセージは、「tkdesk.tgz」という名前のはずのファイルが「tkdesk」という名前でFTPサーバに置いてあったことが原因となって表示されている。そこで、「tkdesk」を「tkdesk.tgz」にリネームしてやり直すと、問題なくインストールが完了した。

なお、今回も「お勧めのパッケージセット」をインストールしたが、あとで調べると、ディスクは550Mバイト程度を使っていた。

●いよいよ立ち上げ

さて、インストールが完了したので、立ち上げてみる。いままでのバージョンでは、最初にインストールされるカーネルがネットワークボードをサポートしていないため、起動するたびに「ネットワークが使えない」というエラーが出ていたが、今回はそのメッセージが出ない。このバージョンではネットワークボードのサポートも標準的に組み込まれたようで、Cバス接続のネットワークボードを使う場合にも、カーネルのリコンバイルは必要ない。またIRQ、I/Oポートアドレスも自動認識できるようになっているようだ。これはLinux/98のバージョンが上がったためだろう。

このバージョンからは、日本語localeに対応して、メッセージも日本語化されているということなので、rootでログインし、「ls -al」と入力してみた。ところが、いきなり文字化けしたメッセージが表示されてしまった。どうも元の文字コードがEUCのものをシフトJISで表示しているようだ（**画面1**）。文字コードがEUC（ja_JP.eucJP）に設定されていたのを思い出し、「vf・1」キーを押してコンソールの文字コードをEUCに変えて、再度試すと、ちゃんと日本語で表示された（**画面2**）。

また、tcshのカタログ機能を使ってメッセージを入れ換えてある。たとえば、実際にはないnocommandというコマンドを実行しようとすると**画面3**のようなエラーメッセージが表示される。これは、気持悪い(笑)。

次に、Xの設定をしてみよう。XF98Setupを起動する。が、これが立ち上がらない。正確にいうと、起動後に画面がブラックアウトし「CTRL」＋「GRPH」＋「BS」キーも効かなくなってしまった。実は付属のドキュメントを読むと、こじま氏のところにはXの動作するPC-98が存在しないので、Xまわりはテストしていないと書いてある。そこで、自分でXF98Configを書いて（正確には/usr/X11R6/lib/X11/XF86Config.98を/etcにコピーし、それを書きなおした）、startxで立ち上げてみると、起動したのは懐かしいtwmだった。

以上、簡単にしかテストできなかったが、まだ、未完成ではあるものの、Plamo 2.0の進化を感じることができた。インストールも楽になったうえに、カーネルバージョンが2.2系になり、glibc2に対応したのが嬉しい。正式版のリリースを期待したい。

P.S. この原稿は、こじま氏にもお送りしたので、読者の皆さんが実際にPlamo 2.0を試す時には、上に書いた問題点はなくなっている……と思います:-P

```
bash# ls -al
total 6
drwxr-x--x   2 root     root         1024  2ｷ 23ﾋﾞ  06:57 .
･rwxr-xr-x  18 root      root         1024  2ｷ 23ﾋﾞ 2000･ｯ ..
-rw-r--r--   1 root     root           39  2ｷ 23ﾋﾞ  06:57 .･ash_history
-rw-r--r--   1 root     root          112  8ｷ  5ﾋﾞ 1999･ｯ .exrc
-rw-r--r--   1 root     root           48  9ｷ 11ﾋﾞ 1996･ｯ .less
-rw-r--r--   1 root     root          114  5ｷ  8ﾋﾞ 1993･ｯ .lesskey
bash#
```

画面1　漢字コードの違いで化けたlsの日本語メッセージ

```
bash# ls -al
total 6
drwxr-x--x   2 root     root         1024  2月 23日  06:57 .
drwxr-xr-x  18 root     root         1024  2月 23日 2000年 ..
-rw-r--r--   1 root     root           39  2月 23日  06:57 .bash_history
-rw-r--r--   1 root     root          112  8月  5日 1999年 .exrc
-rw-r--r--   1 root     root           48  9月 11日 1996年 .less
-rw-r--r--   1 root     root          114  5月  8日 1993年 .lesskey
bash#
```

画面2　正しく日本語表示されたlsコマンド

```
nor@mr2  nocommand
nocommand: コマンドさーん、どこですかー.
nor@mr2#
```

画面3　どこですかーっていわれても……（^^;

往年の名機？　まだまだ現役！

文：編集部　Text : Linux magazine

2000年前半のコンピュータ業界最大の話題といえば、やはりWindows 2000（**画面1**）の登場ということになるだろう。思ったより性能がいいとか、バグが6万個も残っているとか、いろいろな話が伝わってくる。

Linux magazineだからといって、「Windows ＝ 悪」と思っているわけではない。それどころか、個人的にはWindows 2000を「Windows NTの完成品」と考えているくらいだ。

ただ、その機能を享受するためには、高性能のCPUと多くのメモリが必要だ。マイクロソフトは、Windows 2000がPentium 133MHzと64Mバイトのメモリで動くと主張しているが、ハードウェアメーカーからは、500MHzのCPUと128Mバイトのメモリが本当の最低ラインだという意見が聞こえてくる。「動く」と「使える」は違うのだ。いずれにしても、今回取り上げた「往年の名機」たちでWindows 2000を動かすのは、ちょっと非現実的だ。

地球に優しくない？ コンピュータ社会

確かに、メーカー主導でOSのバージョンアップとともに、ハードウェアも全部取り替えて、一からやり直すというのも、決して悪くはない考えだ。

実際、新しいOSには便利な機能が付加されているし、必要な機能を実現できるのが新しいOSだけだというなら、導入するしかないだろう。

そうやって需要を喚起して、経済を活性化していくことで世の中が回っているというのは、ある意味真実だ。

それに以前と比べれば、コンピュータの価格も安くなっている。古いマシンのパワーアップにお金をかけるくらいなら、新型を買ってしまうほうが安くて高性能だったりするから、たまらない。

しかし、この手法を続けると、地球全体で数年に一度、大量にコンピュータを廃棄し続けることになる。これは「リサイクル社会」を目指した、世の中の流れに反しているのではないだろうか。

こんな大上段に構えなくても、コンピュータのように、比較的「高価」というイメージがある製品は、個人ではそう簡単に買い替えられないのが現実だ。その点、部品の交換なら買ったことがばれにくい（？）ので、安心だ。

もう1つの利点

往年のマシン復活大作戦の利点は、このような経済的な面だけではない。いろいろ試していくことで、今までブラックボックスとして使っていた、PCという機械、LinuxというOSに対する理解が深まるという利点があるのだ。

PCの筐体を開けて部品を取り替えるという作業は、難しいと感じるかもしれないが、注意深く作業すればそんなに困難なことではない。静電気に注意するなど、基本的なことを守れば、部品を壊したりすることもめったに起きないものだ。

また1フロッピーLinuxを運用したり、ルータとして利用してみるといった作業を通じて、難解だったLinuxのさまざまな設定項目の意味や必要性が理解できてくるだろう。

ハードにしてもソフトにしても、自分でいろいろと触れてみることが、理解の近道だ。PCの買い替えで、メインの座を降りたマシンは、実験マシンとして活用していくうちに、次の用途が見つかることだろう。

Windows 2000発売は、Linuxにとってもチャンスだ

何と言われようと、Windows 2000は、これから徐々に普及していくだろう。当然、速いCPUと大容量のメモリを積んだマシンといっしょにだ。ということは、現在Windows 98やWindows NTが動いているマシンが大量に宙に浮くことになる。それらの機種は、今回とり上げたような「往年の名機」だけでなく、もうちょっと新しい機種まで含まれるだろう。つまり、特に使いみちのない「Linux ready」の機種が大量に出現するのだ。

「マシンが余ったから、いっちょLinuxでも試してみるか」と思ったときに、今回の特集が役に立つことを願っている。

画面1　Windows 2000のWebサイト

サーバだけじゃもったいない！

マルチメディア Linuxで遊ぼう

**MP3、MIDI、GIFアニメーション、ビデオ中継…。
Linuxでもここまでできる、を一気に紹介。**

村田 勉　（Tsutomu Murata）
四本淑三　（Toshimi Yotsumoto）
藤沢敏喜　（Toshiki Fujisawa）
豊福 剛　（Tsuyoshi Toyofuku）
インタウェア　住吉龍一（Ryuichi Sumiyoshi）
たなかとしひさ　（Toshihisa Tanaka）

Q Linuxをインストールしました。次に何をすればいいですか？（自営業、42歳）

A なるほど、それはたいへんな問題です。よく似た質問に、「C言語を勉強していますが、何を作ればいいですか？」というのがありますね。答えは「とりあえず、何かに使ってみましょう」です。でも次に、何に使えばいいのでしょうか、という質問がきそうですので、もう少し詳しくお答えしましょう。

Linuxというと、サーバというイメージがあります。たしかに正解です。しかし、サーバが必要になる人は、世の中そんなに多くありません。サーバを立てたくてマシンをもう1台用意した、という話も聞いたことがありません。ま、必要になるまで、サーバなんかとかかわるのはやめておいたほうがいいでしょう。

次に、テキスト処理というのがあります。これもたしかに正解です。でも、これはちょっと敷居が高いです。厳密なマッチングが必要ない人には、正規表現はワケわかんないでしょうし。これも、もう少したってから始めましょう。

では、何をやるか？　そうです。マルチメディアです。Linuxで、音楽を聴き、デジカメのデータをいじくり、TVも見る。まったくもって、これが正しいLinuxの利用法です。なんたって、人間「楽しい」が一番です。「AWKの正規表現って、もともと決定性有限オートマトン型だったんだよね」も楽しいかもしれませんが、病院にも行ったほうがいいかもしれません。

というわけで、これからLinuxでマルチメディアを楽しむ方法を紹介します。さぁ、肩の力を抜いて、Linuxで遊びましょう。

Section 1

サウンド

Sound Graphic Video

まずはサウンド。サウンドカードの設定からMP3の利用法までを紹介します。

Linuxでサウンドカードを使う

文：村田 勉　*Text : Tsutomu Murata*

メーカー製のDOS/V機やMacintoshは、初めからCDを入れれば音楽が聴けるし、ゲームをすれば効果音で気分が盛り上がる。そんな中みなさんお使いのLinuxマシンからは「音」は出ているだろうか？ 起動時に“ピー”とか“プッ”といったビープ音が鳴るだけの人も少なくないのではないだろうか。

Windowsユーザーには当たり前、Macユーザーはなぜ？ という感じになってしまうかもしれないが、Linuxの場合、Windows同様、サウンドカードを動かすにはドライバが必要になってくる。当たり前のことだが、そのドライバにサウンドカードが対応していなければ、いくら高価なサウンドカードを挿したところで動かないのである。そこで今回は、LASER5 Linux 6.0 Rel.2（以下LASER5 Linux）を使用し、次の6枚のサウンドカードを使って、それぞれに対応するドライバを試してみた。

画面1 sndconfig
SoundBlaster PCI64カードを自動検出した。

- **SoundBlaster 16**
- **SoundBlaster PCI64**
- **Xwave-320**
- **Xwave-6000**
- **Monster Sound**
- **SoundBlaster Live！ Platinum**

SoundBlaster 16 SoundBlaster PCI64

Creative Technology（以下、Creative社）のSoundBlaster 16（以下、SB16）シリーズは、ひと昔前のPCでは標準的に採用されていたサウンドカードである。今回試す中では唯一のISAバス仕様だが、歴史があるぶんLinuxでのドライバは充実している。

SoundBlaster PCI64（以下、SB64）は、サウンドチップにCreative社が1997年に買収したENSONIQ社のES1370を使用している。

SB16もSB64もLASER5 Linux標準のドライバ（OSS/Free）で動作する。コンソールで使用しているならそのまま、X Window Systemを立ち上げているなら、ktermなどを起動して次のコマンドを入力する。

```
# /usr/sbin/sndconfig
```

最初の画面で「了解」を選択すると自動的にサウンドカードを探してくれる。うまい具合にサウンドカードを見つけられると**画面1**のように表示される。このあとサンプルの音声（Linus氏の話し声？）と、MIDIの音のテストが行われる。それぞれ「それが聞こえたか？」という質問には、聞こえたならば「はい」を選択する。次に「conf.modulesに書き込んでよいか？」という質問にも、「はい」を選択してsndconfigを終了させる。

ドライバが組み込まれたあとのテスト方法を紹介しておこう。一番簡単なのは先ほどのサンプル音を利用する。このサンプル音は/usr/share/sndconfigディレクトリにあるので、次のようにして試してみよう。

```
# cat /usr/share/sndconfig/sample.au
> /dev/audio
```

サンプル音が流れるが、ちょっとノイズが混じったりするのが気になる。playコマンドを使用してみよう。

```
# play /usr/share/sndconfig/sample.au
```

これで、ちゃんと聞こえると思う。次にMIDIフォーマットのファイルだが、これはplaymidiコマンドで再生することができる。サンプルMIDIファイルの再生は以下のように行う。

```
# playmidi /usr/share/sndconfig/
sample.midi
```

SB16、SB64ではMIDIファイルの再生はどうもFM音源による音らしく、かなりチープな音になることは否めない。

ALSAドライバを試す

今度はSB64を使って違うドライバを試してみよう。ALSA（Advanced Linux Sound Architecture）はOSS/Freeと互換性を保ちながら、高機能なLinux用サウンドカードドライバALSAを開発している。

http://www.alsa-project.org/から「alsa-driver-0.x.x」「alsa-lib-0.x.x」「alsa-utils-0.x.x」「alsaconf-0.x.x.」（**表1**）をそれぞれダウンロードする。ちなみにこの記事を書いている時の最新バージョンは0.5.1aだった。

今回は/tmpディレクトリにダウンロードして、作業することとして話を進めよう。

```
# cd /tmp
# tar xvfz alsa-driver-0.5.1a.tar.gz
# tar xvfz alsa-lib-0.5.1a.tar.gz
# tar xvfz alsa-utils-0.5.1.tar.gz
# tar xvfz alsaconf-0.4.2.tar.gz
```

そして、それぞれコンパイルしてインストールを行う。まずは、

```
# cd /tmp/alsa-driver-0.5.1a
# ./configure
# make install
```

として、ドライバのインストールから始める。

同様にalsa-lib、alsa-utilsもインストールしていく。このときに「alsa-driver」→「alsa-lib」→「alsa-utils」の順番にコンパイル、インストールしないと参照ファイルがないといったエラーになるので注意してほしい。

次に、alsa-driverディレクトリにあるsnddevicesというコマンドでデバイスファイルを作成する。

```
# cd /tmp/alsa-driver-0.5.1a
# ./snddevices
```

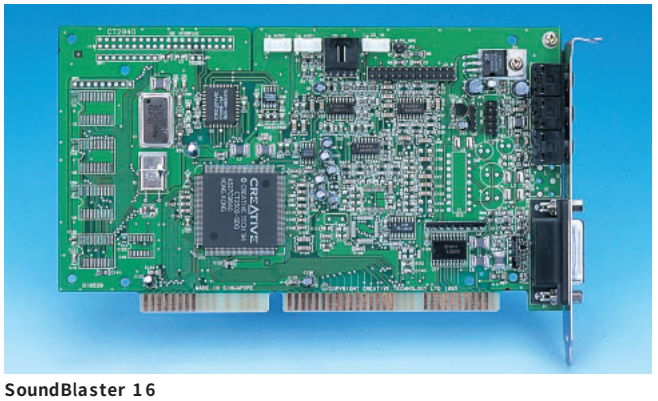
デバイスファイルができたところで次にサウンドカードの認識をさせよう。

```
# cd /tmp/alsaconf-0.4.2
# ./alsaconf
```

このalsaconfコマンドでは、起動と同時に自動認識を試みてサウンドカードを見つけた場合はそのカードを表示し設定画面に移動する。運悪く見つけられない場合はサウンドカードの選択画面に変わる（**画面2**）。サウンドカー

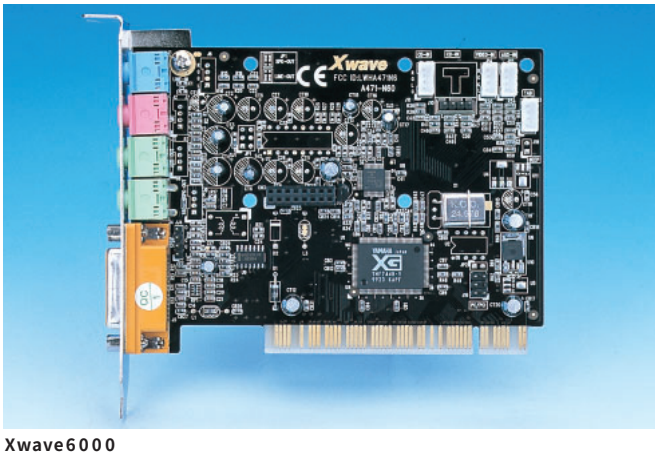
ファイル名	FTPサイト
alsa-driver-0.5.1a.tar.gz	ftp://ftp.alsa-project.org/pub/driver
alsa-lib-0.5.1a.tar.gz	ftp://ftp.alsa-project.org/pub/lib
alsa-utils-0.5.1.tar.gz	ftp://ftp.alsa-project.org/pub/utils
alsaconf-0.4.2.tar.gz	ftp://ftp.alsa-project.org/pub/driver/alsaconf

表1　ALSA関連パッケージの入手先URL

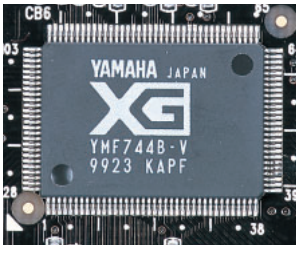


SoundBlaster 16
ブラケットにはライン入力、マイク入力、ラインおよびスピーカー出力、MIDIおよびジョイスティックポートがある。昔のカードなので妙にでかい（笑）。

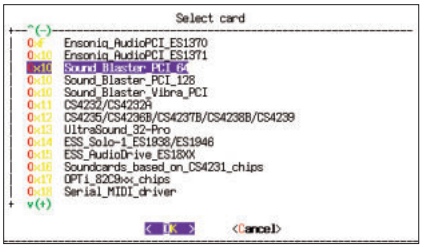
音源チップ（名称不明）
8ビットおよび16ビットのステレオ録音と再生。サンプルレートはユーザー選択可能。PC上でDATデッキに匹敵するオーディオ録音機能（本当か!?）。



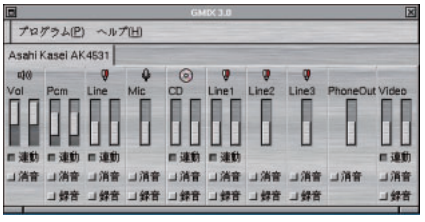
Xwave6000
カード上にTAD-IN、2種類のCD-IN、VIDEO-IN、AUX-INが、ブラケットにはライン入力、MIC入力スピーカー出力がフロント、リアの2つ。GAME/MIDIポートがある。



YMF-744
あちこち調べたのだが詳細不明。位置づけとしてはYMF-724の後継機種となっている。



画面2　サウンドカード選択画面
alsaconfでSoundBlaster PCI64を選択する。



画面3　GMIX3.0、これでボリューム調整をする。

ドを選択したら自動認識したときと同様、サウンドカードの設定画面に移行する。

まずサウンドカードの認識番号を聞かれるがデフォルトのままで問題ない。次にサウンドカードのdac（D/Aコンバータ）やrecord時のフレームの大きさを設定する。どのくらいの数値に設定すればよいか見当もつかないと思うが設定できる数値の範囲は決まっているので、リミット一杯にするか、それが恐い人は余裕を持って半分くらいにしておけば問題は起きないだろう。最後に「conf.modulesを書き換えてよいか？」の質問に「OK」を選択すると、conf.modulesファイルに設定情報が書き込まれる。

次に、モジュールを呼び出すために、

```
# modprobe snd-card-0
```

と入力する。これでモジュールが読み込まれる。起動のたびにこのコマンドを入力するのは面倒なので、/etc/rc.dディレクトリのrc.localに「modprobe snd-card-0」と書いておくとよいだろう。

さて、音を鳴らしてみよう。alsa-utilsに含まれるaplayコマンドを使う。

```
# aplay /usr/share/sndconfig/sample.au
```

どうだろう？　何も鳴らないはずである。デフォルトで音量が0になっているので、今のままでは音は出ない。ボリュームを上げるためにミキサを起動しよう。GNOMEのメニューから［マルチメディア］―［オーディオミキサー］を選択する。もしくはktermなどから「gmix &」と行ってもよい。

ミキサを呼び出したらVolとPcmのボリュームを上げる（**画面3**）。これで音が出るはずだ。ボリュームを上げる方法としてはほかに「alsa-utils」に含まれるamixerを使うというのもある。ただし、使用方法が複雑なのでただボリュームを上げるだけなら、より簡単なgmixを使用することをお勧めする。

Xwave-320 Xwave-6000

LabwayのXwave-320とXwave-6000は、YAMAHAの音源チップYMF-7xxを使用している。このチップを使ったサウンドカードは種類が多い。Xwave-320はYAMAHAのYMF-724を使っている。Xwave-6000はYMF-744を使用している。この2つはどちらとも4Front TechnologyのOSS/Linuxドライバで動かすことができる。

まずはドライバを以下のURLからダウンロードしよう。

http://www.opensound.com/download.cgi

このページに必要事項を記入して「What soundcard（s） are you using？」の欄に「YMF-744」と入力し、「Select the version of OSS？」の欄は「Linux 2.2.x（UP）」を選択して「Submit」ボタンを押す。するとダウンロードのページに移るので、そのページからダウンロードして展開する。

```
# tar xvfz osslinux393c-glibc-2214-UP.tar.gz
```

展開すると4つのファイルしかないことにびっくりさせられる。商用アプリケーションなのでソースファイルがないから少なくて当然ということなのだろう。oss-installを実行してインストールする。

```
# ./oss-install
```

ここでLASER5 Linuxの場合は警告が出た。terminfoのリンクを張ってくれということらしい。そこでメッセージにしたがってリンクを張っておく。

```
# ln -s /usr/share/terminfo /usr/lib/terminfo
```

再びインストールをしてみよう。今度は何もいわれない。

リブートして、何か音を出してみよう。今回は音楽CDでもかけてみようか。CD-ROMドライブになにか音楽CDを入れて、CDプレーヤを立ち上げよう。そして再生……、あれ？　音が出ない。そうそう、その前に、

```
# /usr/local/bin/soundon
```

とドライバを起動する。これでもう一度再生してみると、ほら鳴りましたね？　このドライバは試用版なので起動後20分で音が止まってしまう。ドライバを止める（unloadする）には、

```
# /usr/local/bin/soundoff
```

を実行する。継続して使うには20USドルを支払ってライセンスを購入する必要がある。今のところYMF-7xxをサポートしているドライバがほかにないため、このサウンドカードを使いたい場合は購入するほかない。

Monster Sound MX300

Monster Sound（以下MSound）は、派手なパッケージが多いDiAMOND Multimedia Systems社の製品。Aureal Vortex 2テクノロジー採用によるA3Dアクセラレーションが特徴である。この技術のおかげでヘッドフォンや2組のスピーカを使って、臨場感あふれる音を作り出すことができる。Aureal A3Dは、Half-Life、Unrealなど最近流行したゲームなどにも採用された、オーディオレンダリングテクノロジーだ。

このカード（音源チップはVortex2）は、これまでOSS/Linuxがβ扱いで対応しているだけだったが、今年になって、サウンドチップの開発元であるAurealからLinux用のドライバがリリースされた。

Aureal Linux Web（http://linux.aureal.com/）からドライバau88xx-1.0.5.tar.gzをダウンロードする。インストールは以下の手順で行う。

```
# tar xvfz au88xx-1.0.5.tar.gz
# cd au88xx-1.0.5
# make install
```

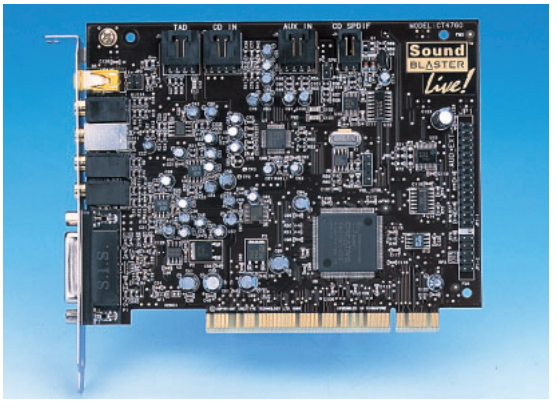
なお、残念ながらこのLinux用ドライバはA3Dに対応していない。

SoundBlaster Live ! Platinum

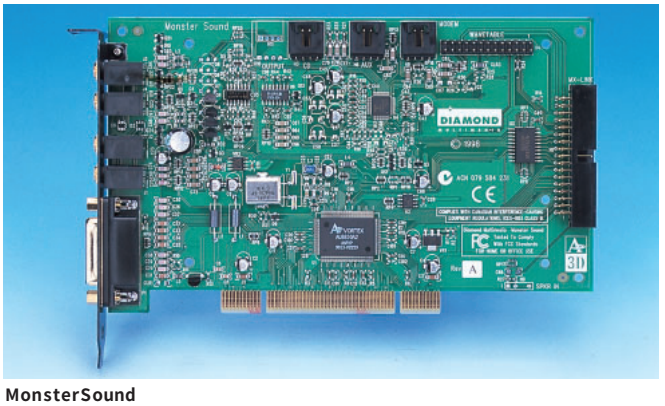
次はCreative社のSoundBlaster Live !（以下SBLive）だが、今はこのカードが定番中の定番だろう。このSBLiveに使われているEMU10K1チップ、ALSAのドライバもサポートしているのだが、Creative社が自社開発でLinux用のドライバを配布しているので、今回はそちらを使うことにしよう。

同社のWebサイト（http://opensource.creative.com/snapshot.html）からemu10k1-20000xxxx.tar.gzをダウンロードする。xxxxには日付が入る。

```
# tar xvfz emu10k1-20000215.tar.gz
```

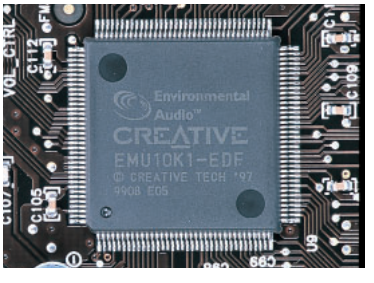


SoundBlaster Live !
カード上にはCD入力、Aux入力、TAD入力、CDデジタル入力！、拡張カード用コネクタ。ブラケットにはマイク入力、ライン入力、スピーカー出力×2、MIDI/GAMEポート、そしてこちらもデジタル出力！



MonsterSound
カード上にウエーブテーブル用の拡張コネクタ、CD、AUX、モデム、MXリンク用のコネクタがある。ブラケットにはスピーカー出力が2組、MIC入力、ライン入力がある。なによりブラケットが黒いのがいかす。

Aureal Vortex2
オーディオ圧縮形式はQuad AC97。48kHzまでの録音/再生。S/N比は90dB以上！（本当か！）



EMU10K1
8ビットから16ビットの選択可能なビットレート。
8kHzから48kHzの選択可能なサンプリングレート。アナログ出力、デジタル出力。デジタル出力のサンプリングレートは48kHz固定。

Live! Drive II
SB Live! Platinumには、ドライブベイ前面から入出力できるLive Driveが付属する。同軸＆光のデジタル入出力やライン／マイク／AUX入力、ボリュームコントロール付きヘッドフォン出力、MIDI入出力が用意されているので接続が簡単だ。

emu10k1ディレクトリができて、その中に展開される。さらにそのディレクトリの中に2.2というディレクトリができるが、READMEによるとカーネルのバージョンが2.2.xの場合はこれも一緒にコンパイルしろということなので、そのとおりにしておこう。

```
# cd 2.2
# mv emu_wrapper.h ../
# make ; make install
```

これで「emu10k1.o」というモジュールが作られ「/lib/modules/2.2.5-22lv3/misc」ディレクトリにインストールされる。しかし、ここまでしてくれるのに「conf.modules」の書き換えはしてくれないので、エディタで書き換えておこう。付け加えるのはこの1行。

```
alias sound emu10k1
```

ここでリブートするか、modprobeコマンドでemu10k1モジュールを呼び出せば動作する。サウンドテストには、何度か紹介したplayコマンドでLinus氏の声でも聴いてみよう。

ドライバのバージョンからβは取れたが、まだまだ開発途上であって、SBLiveの機能を全部使うことができないのはいささか残念だ。しかしほかのメーカーがLinux用のドライバ開発に積極的ではないのに比べると、Creative社の取り組み方はとても好感が持てる。

MIDI音源を使ってみよう

読者の中でMIDI用の外部音源（以下音源モジュール）を持っている人はどれくらいいるのだろうか？ 今回使ったのはRolandのGS音源モジュールSC-88である。この音源モジュールはコンピュータのシリアルポートにつないでMIDIデータを再生することができる。

まず音源モジュールとコンピュータをMIDI専用ケーブルでつなぐ。さて、プレーヤを探しに世界中を駆け回ってみますか？ と思いきや、ここ日本によいソフトがありました。しかもフリーソフトで。まずは下のホームページに飛んでみる。

http://matu.cc.kyushu-u.ac.jp/~nagano/linux/unixsoft0.html

このページの「midiplayちょっとだけ改良版」というのをダウンロードしてどこか適当なところに置いて展開する。midiplay-0.5+09というディレクトリができるので、そこに移動し、

```
# ./configure
# make ; make install
```

としただけですんなりとインストールできた。さっそく動作テストをしてみよう。

```
# midiplay /usr/share/sndconfig/sample.midi
```

音は出ただろうか？ さてMIDIデ

SC-88
RolandのMIDI音源モジュール。32パート64ボイスのGS音源で、DTM音源として標準的に使われているリファレンスモデル。

Column

OpenLinuxとOSS/Linux

OSS/LinuxやOSS/Free、ALSAと、サウンドカードのドライバを作っている代表的なプロジェクトはこの3つ（表）なのだが、OSS/Free、ALSAがフリーソフトウェアなのに対して、OSS/Linuxは商用アプリケーションとなっている。逆に商用アプリケーションなので、いろんなサウンドカードに対応できているともいえる。このOSS/Linux、ホームページから試用版がダウンロードできるのだが、使える時間がOSS/Linux起動時から20分と限られているのに加えてSMPカーネルには対応していない。

OpenLinux 2.3日本語版はインストールが非常に簡単にできるようになっていて、今までのLinuxに比べるとはるかに楽になった。しかし、ここに落とし穴があった。これは後になって判明したのだが、OpenLinuxはデフォルトでSMPカーネルをインストールしているのである。これによってOpenLinuxにOSS/Linuxをインストールして使おうとしたときに「これは試用版なのでSMPカーネルには対応していません」という意味の言葉に出くわすことになる。

今回はなるべく手間を掛けずに話を進めたいのでOpenLinuxを使うことは避けた。もちろん、カーネルを入れ替えれば動くと思われるのはいうまでもない。さらにいえばOSS/Linuxの正式版はSMPに対応しているので、マルチプロセッサ環境の人で音を鳴らしてみたい人は正式版を購入すれば問題なく動くはずである。

プロジェクト	WebサイトURL
OSS/Linux	http://www.opensound.com/
OSS/Free	http://www.linux.org.uk/OSS/
ALSA	http://www.alsa-project.org/

各プロジェクトのWebサイト

ータだが、インターネットで探せば趣味で作ったものを公開している人もいるし、ダウンロード販売しているWebサイトもたくさんある。私は秋葉原で、エリックサティのピアノ＆オーケストラというのを購入した。YMOのあの曲や坂本龍一のあの曲なども売っていたのだが……。

ソフトウェアMIDIを使う

さて、最近のあり余るCPUパワーを使わない手はないだろう。Pentium IIやIIIの高速CPUを載せている人も多いと思う。だとしたらソフトウェアシンセサイザというか仮想WAVEテーブルとでもいうか、そういった手段でMIDIデータを再生することもできる。

まず、ソフトウェアを用意しよう。TiMidity++（**画面4**）を使用する。

http://www.goice.co.jp/member/mo/timidity/dist/download.html

ここから「Timidity++-2.8.1」をダウンロードしてくる。これをどこか任意の場所で解凍して、READMEとINSTALLを読んでコンパイルしてインストールする。

```
# ./configure
# make ; make install
```

これでTimidity++自体はできあがり。しかしまだ用意するものがある。音源データとなるGUS互換のパッチファイル（以下、GUS/pat）と、このパッチファイルがどこにあるかなどを記述したtimidity.cfgだ。このGUS/patによってMIDIデータがWAVEデータに変換され、MIDIデータの再生が可能になるのだ。

timidity.cfgはデフォルトでは、/usr/local/share/timidityに置くことになっている。timidity.cfgを開いてみてみると、そのほかのコンフィグファイルは/usr/local/lib/timidityに置いておくことになっているので、あらかじめディレクトリを作っておこう。

まず、http://www.goice.co.jp/member/mo/timidity/dist/cfg/からtimidity.cfgを/usr/local/share/timidityにダウンロードしよう。

この時このWebページにあるすべての「*.cfg」というファイルを、/usr/local/lib/timidityにダウンロードしておく。

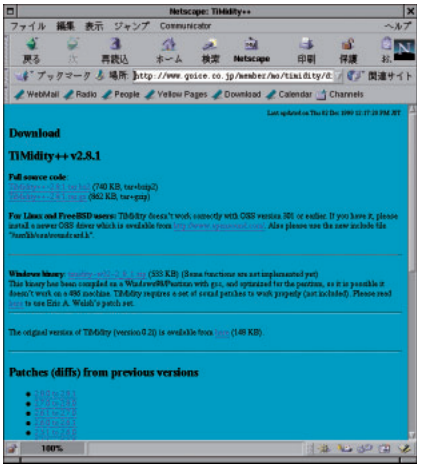
■まだまだこれから……

次に音源データGUS/patを拾いにいこう。先に言っておくがこのGUS/patがめちゃめちゃ大きい。小さいもので4Mバイト。大きいものは数十Mバイトのもある。何が違うのかというと、大きければ大きいほど音数が増えたり、音色が本物に近くなったりする。今回は間を取ってというか、聴くに耐えられる程度のものを用意したかったので、ここから落としてきた。

http://www.goice.co.jp/member/mo/release/tim-patches.html

13分割された「guspat-*」というファイルがあると思うが、全部を落とすと33Mバイトになる。できれば全部落としたいところだがそんなに電話代は払えないとか、会社なのでログが残るからあんまり目立ったことはできない（笑）という人は必須というファイルだけは取得するようにしてほしい。

そうしないと再生されない音が出てきてとても悲しい曲を聴くことになる。ちなみに私はオプションと書かれたものまで落としてみた。timidity.cfg以外



画面4　TiMidity++ダウンロードページ

のコンフィグファイルを置いたディレクトリ/usr/local/lib/timidityにダウンロードした場合、GUS/patは1つ上のディレクトリ/usr/local/libで展開する。するとtimidityディレクトリの中にGUS/patファイルを解凍してくれる。

```
# cd /usr/local/lib/
# tar xvfz guspat-*
```

さて、再生してみよう。再生のしかたは特に変わったところはない。

```
# timidity （MIDIファイル名）
```

これで音が流れるはずである。このときエラーが出るようであればMIDIデータに対応するGUS/patファイルが存在していないことになる。完全に再生されるまで頑張って集めるか、各コンフィグファイルを編集して違う音色でごまかすしかない。

サウンドカードのドライバによっては、TiMidity++コンパイル時の./configureを実行する際、オプションを付けてMakefileを作らないとちゃんと音が鳴らない場合もあるので、READMEなどはしっかりと目を通していただきたい。

MP3でミュージックライブラリ

文：四本淑三　Text : Toshimi Yotsumoto

このところ、MP3という言葉はすっかり市民権を得た感じだが、Linux上では市民権を得るところまでいっているかは疑わしいところだ。MP3が使いたくてWindowsにしました、というユーザーが増えないよう、ここではLinuxにおけるMP3ファイルの作成および利用法について解説していこう。

なぜMP3？

オーディオCD（CD-DA）の音楽データをMP3（MPEG Audio Layer-III）形式のデータに変換して聴くメリットは何だろうか？　MP3そのものが持つメリットは、いうまでもなくデータサイズを1/10～1/12にするという圧縮率の高さ（1分当たり約1Mバイト）と、44.1KHz/16ビット/ステレオというCD並の音質にある。だが、パソコンで聴く場合のメリットは、ランダムアクセス性と検索性の高さが挙げられるだろう。具体的には、次のようなものだ。

■MP3は早い、安い

ハードディスクに記録されたMP3ファイルは、巻き戻しや頭出しの操作が不要で、呼び出せばすぐに再生できる。CDの差し替え操作も不要になる。仮に、CD1枚の収録時間を1時間とすれば、30Gバイトのハードディスクに、おおむね500枚分のCDを収録できる計算になる。壁一面にあるCDコレクションを1台のハードディスクに移し替えることもできるわけだ。

それに、ハードディスクの価格が1Gバイト＝1000円を切っている今なら、1時間当たりの単価は60円以下。これは、カセットテープやMDよりも桁がひとつ少ない価格といえるわけで、コストパフォーマンスも高い。

■ID3タグによる整理／選曲

蓄積された音楽データをファイル名だけで識別するのは難しい。本格的に管理しようとするならデータベースを利用した管理が必要になってしまうだろう。でも、フツー音楽のためにそこまでやる気はないはずだ。そういう場合に便利なのがMP3のID3タグだ。

ID3タグを使えば、

- **タイトル**
- **アーティスト**
- **アルバム**
- **発表年**
- **ジャンル**
- **コメント**

といった6種類のデータをMP3ファイル自身に書き込むことができる。1項目につき最長30文字（ただし、発表年は4文字）までという制限はあるのだが、このデータを利用すれば、MP3プレーヤから望みの曲を楽に探し出せる。むろん、このデータ入力も全部手動でやらなければならないのなら意味はないが、CDDB（http://www.cddb.com/）などのCDに関する情報を集めたデータベースサーバを利用すれば、自動で書き込みをすることができ、省力化できるのだ。

こうしたMP3の特徴を鑑みつつ「個人所有のCDをMP3に変換して、ミュージックライブラリ化してしまおう、しかもラクをして」というのが今回の目的である。なお、今回紹介したツールはすべて、付録CD-ROMのLinuxmag/multimedia-tools/Soundディクトリ以下に収録してあるので、すぐに試せるはずだ。

インストールと設定

CDの音楽データをMP3ファイルへ変換するには、CDから音楽データを抽出する「リッピング」、抽出されたファイルを圧縮する「エンコード」という、2つのプロセスを経る。ID3タグを利用するなら、さらに「CDDBからのデータ取得」と、エンコード済みファイルへの「ID3タグ書きこみ」というプロセスも加わる（**図1**）。

これらは「リッパー」「エンコーダ」「ID3タグエディタ」と呼ばれるそれぞれ別のアプリケーションで処理するわけだが、バラバラに起動して手動で処理するのはたいへんだ。そうした面倒くさがり向けのツールとして、Linuxの世界ではGripが人気を得ている。

■Grip［rpm/deb/tar.gz］

http://www.nostatic.org/grip/

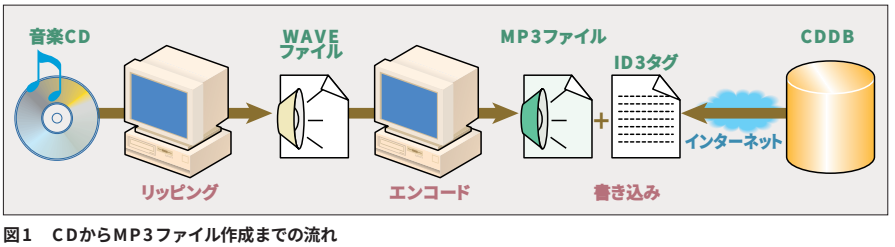


図1　CDからMP3ファイル作成までの流れ

grip-2.91-1.i386.rpm
grip-2.91-1.deb
grip+cdpar-2.91.tgz

Gripの人気の秘密は、その「使いやすさ」にある。本来Gripは、単独ではMP3ファイルへの変換まではできないのだが、外部のリッパー／エンコーダ／ID3タグエディタのフロントエンドとしても機能するので、外部ツールさえ用意してGripに登録しておけば、一発でMP3ファイルを作成することができるのだ。MP3ファイルの作成は、CPUに負荷がかかるタスクなので、使っていない夜間にMP3ファイルを作る場合も、この一発作成のありがたみは大きい。

また、最新バージョンのGripには、最初からリッパーとしてcdparanoia（http://xiph.org/paranoia/）が組み込まれているので、あとはエンコーダとID3タグエディタをインストールするだけだ。Gripには、それぞれのアプリケーション向けの設定があらかじめ用意されており、設定も簡単である。

付録CD-ROMに収録したRPMバイナリ（grip-2.91-1.i386.rpm）は、Red Hat 6.0以降をベースにしたRPM系のディストリビューションなら、

```
# rpm -Uvh grip-2.91-1.i386.rpm
```

とするだけで、ライブラリ関係の問題もなくそのままインストールできるはずである（以下に紹介するツールは、すべてこのようにしてRPMバイナリを指定すればインストールできるはずである）。このRPMバイナリは、gtk-1.2.0以降、glibc2.1以降を必要とするため、これより古いディストリビューションを使用している場合は、これらのライブラリをアップグレードしておくか、次に紹介するtarボールからのインストールを行ってほしい。

tarボールのソースをインストールする場合の手順は次のようになる。

```
$ tar xvfz grip+cdpar-2.91.tgz
$ cd grip-2.91/cdparanoia
$ ./configure
$ make lib
$ cd ..
$ make
$ su
# make install
```

ここでktermなどから、

```
$ grip
```

として、Gripが起動できればインストール成功である。なお、曲名の表示に日本語を使いたい場合は、ソースファイルのgrip.cの5030行目にある、

```
gtk_init(&argc,&argv);
```

という行の前に、

```
gtk_set_locale();
```

という1行を追加して、GTK+の国際化機能を有効にしてからビルドすればよい。

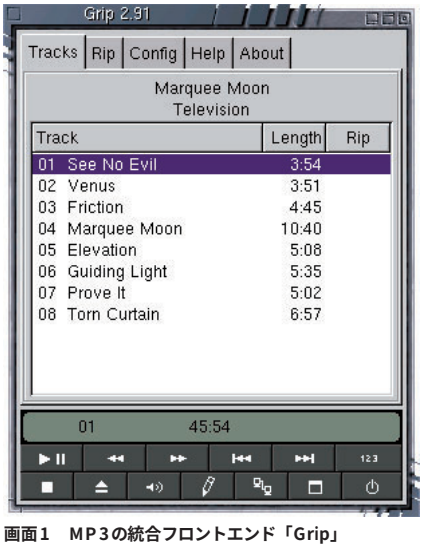
次に動作の確認をしてみよう。まずインターネットへ接続し、CD-ROMドライブにオーディオCDを挿入してみる。何も問題がなければ、GripはCDDBへアクセスし、挿入したCDの楽曲の情報がCDDBにあれば、その情報をダウンロードし、曲名などを表示する（**画面1**）。インターネットに接続していないか、あるいはCDDBに楽曲のデータがなければ「Query failed」のダイアログとともに、「Unknown Disk」と表示される。が、これはちゃんと動作している証拠なので、問題はない。

CDを挿入しても何の表示もされなかった場合は、CD-ROMのデバイスファイルをうまく読めていない可能性がある。Gripでは、CD-ROMドライブとして/dev/cdromを読みにいくが、これがシンボリックリンクとなっていて、/dev/cdromが実際に使用しているCD-ROMドライブとリンクされていない場合、Gripは動作しない。ATAPIのCD-ROMを複数接続している場合や、SCSIドライブの場合は、該当するデバイスファイルから/dev/cdromにリンクを張り直す必要があるかもしれない。

たとえば、1台目のSCSI CD-ROMドライブなら/dev/scd0だから、これを/dev/cdromへリンクするなら次のようになるだろう。

```
# rm /dev/cdrom
# ln /dev/scd0 /dev/cdrom
```

あるいは次のように、Gripの起動時に-dオプションをつけ、任意のドライ



画面1　MP3の統合フロントエンド「Grip」
CDDBから取得してきた曲名が表示されている。残念ながら、CDDBに登録されているのは、「ほとんど」洋楽CDばかりで、邦楽CDは「あまり」登録されていない。

ブを指定する方法もある。

```
$ grip -d /dev/scd0
```

また、rootユーザーでは再生できるのに一般ユーザーではうまく動作しない、というトラブルに遭遇したら、CD-ROMドライブへのアクセス権限が一般ユーザーに与えられていないことが原因かもしれない。このときは、

```
$ su
# chmod 666 /dev/scd0
```

として、適切なアクセス権限を与えてやればよい。実は、これはGripの作者の「root権限で実行すべき」という信念によるものだが、このままでは不便だし危険なので、許してもらうことにしよう。

ここまでうまく動いたら、次はMP3エンコーダの準備だ。いろいろあるが今回は「午後のこ〜だ」を紹介しておこう。

■午後のこ〜だ［rpm/tar.gz］
http://homepage1.nifty.com/herumi/soft.html
gogo-2.25-1.i386.rpm

日本のハッカーの手によるMP3エンコーダで、すば抜けた高速動作が特徴。「午後のこ〜だ」は、フリーのエンコーダとして開発が進められているLAME（http://www.sulaco.org/mp3/）をベースにしたものだが、SMPのサポートや3D Now!、MMXなどx86系CPUの拡張命令に最適化されたバイナリが用意されている点が異なる。機能的には、最新のLAMEには及ばないが、非力なマシンでも必要十分な速度でエンコードできるため利用価値は高い。ちなみに、手元にあるCeleron 433MHzのSMPマシンでは、実演奏時間の1/8程度でエンコードを終えてしまう。さらに、PentiumやPentiumPro、K6/K7などに最適化されたバイナリも用意されているので、自分のマシンに合ったものを選んでインストールしよう。

Gripで「午後のこ〜だ」を利用するには、Gripの［Config］－［MP3］－［Encorder］タブを開いて、ポップアップメニューから［gogo］を選べばよい。

次に、ID3タグを書き込むツールを用意すれば、MP3ファイルを作成するための準備は完了だ。

■mp3info［rpm/tar.gz］
ftp://bimbo.hive.no/pub/mp3info/
mp3info-0.2.16-1.i386.rpm
mp3info-0.2.16.tar.gz

mp3infoは、CDDBからダウンロードしてきた楽曲データをGripから受け取り、エンコードずみのMP3ファイルにID3タグを書き込むツールである。地味なツールだが、快適なMP3環境を作るためには欠かせないものだ。

```
$ tar xvfz mp3info-0.2.16.tar.gz
$ cd mp3info-0.2.16
$ ./configure --prefix=/usr
$ make
```

Column

フリーエンコーダ事情

LAMEは、早くからVBR（Variable Bit Rate）に対応し、自前の心理聴覚モデルを持つなど、非常に意欲的なプロジェクトである。またオープン開発ということでバージョンアップも早く、さまざまな派生バージョンも生んでいる。今回紹介した「午後のこ〜だ」もその流れの中にある。

しかし、これらのフリーなMP3エンコーダは、著作権や特許権でやや問題を抱えているのが実情である。というのも、MPEG1が標準化されたあとに、その標準化に参加したドイツのFhG-IIS社がエンコーダ関係の特許を主張し始めたからである。そのため、さまざまなフリーエンコーダがライセンス料の要求を受け、公開が中止となった。MPEGで標準化されたのは再生時のデータ形式だけであり、エンコードのアルゴリズムについては標準化されていない。そのアルゴリズムの権利は、FhG-IIS（Fraunhofer-Gesellschaft IIS）が一部所有しているというわけだ。

こうした問題を回避するために、LAMEは音声ストリームを非可逆圧縮するための規格のサンプルソース「dist10」（ftp://ftp.tnt.uni-hannover.de/pub/MPEG/audio/mpeg2/software/technical_report/）に当てるパッチというスタンスをとっており、オープンに開発が進められている。

このように、LAMEや「午後のこ〜だ」では著作権の及ぶ範囲を明確に分離しているため、ソースコードはLGPLにしたがって世界中へ配付可能となっている。一方、MP3エンコーダで使用されているアルゴリズムは、日本国内では現在のところ特許が成立していないという理由から、本来ライセンス料が必要になるバイナリファイルも、日本からは配付可能となっている。

そもそも、広く一般に使われるべきMPEG規格で、このような権利関係が発生すること自体が問題である。これは、LAMEの性能の高さでも実証されているオープン開発の利点を阻害してしまうものにほかならないからだ。先ごろ標準化が確定したMPEG4は、MP3よりも圧縮効率の高いAAC（Advanced Audio Cording）が含まれており、この音声形式にもすでにFAAC（http://www.slimline.net/aac/）というエンコーダのオープン開発が始まっている。今後に注目だ。

```
# make install
```

configure時にオプションとして指定した“ --prefix=/usr ”は、インストール先を/usr/binディレクトリ以下にするためのものである（Gripのデフォルト設定が、/usr/binディクトリになっているため）。もしも、この指定をしないでtarボールからインストールした場合は、実行ファイルのインストール先が違うので、Gripのパス設定が必要になる。

“ --prefix=/usr ”をつけなかった場合、実行ファイルのインストール先は/usr/local/binディレクトリになってしまうが、Gripを起動し［Config］－［ID3］タブを開いて、［ID3 executable］のパスを“ /usr/local/bin/mp3info ”と書き換えればよい（**画面2**）。

MP3ファイルの作成

MP3ファイルを作るには、まずGripの［Tracks］画面からエンコードしたい曲目を右クリックでチェックする。CDDBにデータがなく「Unknown Disk」だった場合や、タイトルデータを書き直したいときは、ペンのアイコンをクリックすると入力フィールドが表示され、データの編集が可能になる（**画面3**）。編集が終わったらフォルダのアイコンをクリックすると~/.cddb以下にデータは保存される。

エンコードを始めるには、［Rip］タブを開いて［Rip+Encord］ボタンを押すだけ。ほとんどの場合、これでチェックした曲が次々に変換されていき、リッピングとエンコーディングの進行状況がバーグラフで表示されるはずである（**画面4**）。

ここでバーグラフが動かず、読み取りが止まってしまう場合は、CD-ROMドライブそのものに問題があるかもしれない。古い世代のCD-ROMドライブではデジタルオーディオデータの抽出に対応していない機種があり、この場合リッピングはできない。これに遭遇したら、残念だがあきらめるしかない。

なお、「読み出し40倍速」などと表示されたCD-ROMドライブであっても、オーディオデータがその速度で読み出されるわけではない。最新のドライブではデジタルオーディオの抽出速度を上げた製品もあるが、通常は2～4倍程度である。

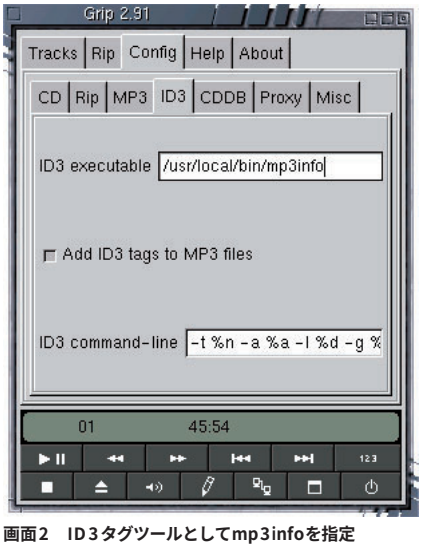
問題なく変換が終わった場合には、ホームディレクトリ以下に、

~/mp3/アーティスト名/アルバムタイトル/

のような形でMP3ファイルができているはずだ。それと同時に、

~/mp3/アーティスト-アルバムタイトル.m3u

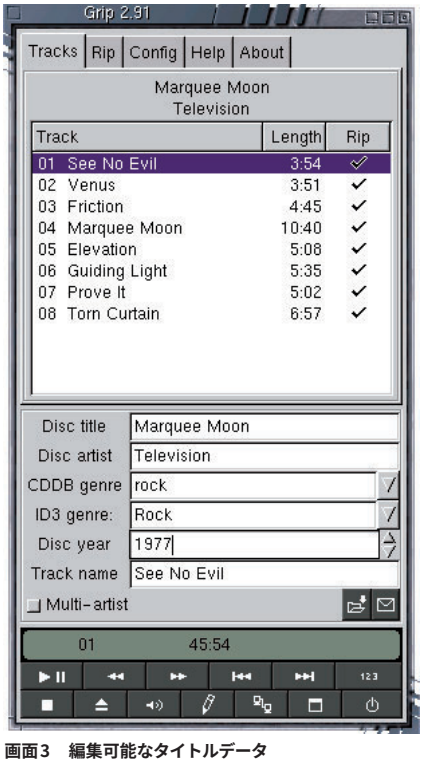
のようなファイルも生成されている。これは、エンコードしたmp3ファイルの絶対パスが書かれたファイルで「プレイリストファイル」と呼ばれる。再生時にMP3プレーヤから利用するものだ。



画面2　ID3タグツールとしてmp3infoを指定

■ビットレートの調整

MP3の音質と圧縮率を決定するパラメータが「ビットレート」で、MP3の場合は128kbpsが標準的な値になっている。この値は1秒当たりのデータ量で、小さいほど圧縮率は高い。だが圧縮率を上げ過ぎると音質は悪化してしまうし、エンコードするソースによっては、128kbpsでも音質的に満足でき



画面3　編集可能なタイトルデータ

画面4　MP3データへのエンコード中のGrip

ないことがある。特に、フリーのエンコーダでは、128kbpsという値は、音質が保持できるギリギリのビットレートのようで、高音域に多く情報を持った音は変調を起こしがちになる。

変調の問題はビットレートを上げてやれば解決するが、肝心の圧縮率が犠牲になってしまう。そこで、音の情報に応じて連続的にビットレートを調整し、その他の部分は抑えることで、全体としての圧縮率を確保しようというのがVBR（Variable Bit Rate；可変ビットレート）である。

エンコードの速度は半分以下に落ちてしまうが、「午後のこ〜だ」でもこのVBRが使える。VBRでエンコードするにはGripの［Config］－［MP3］－［Encorder］タブを開いて、次のように［MP3 Command-line］に-vオプションとパラメータを書き加える。

たとえば、「128kbpsの音質は不満だが、圧縮率は1/10程度で収めたい」というのなら、まず次のような設定を試してみるといいだろう。

```
-v 5 -b %b %f %o
```

-vオプションに続くパラメータは、0〜9の値を取り、値が大きいほど高圧縮率（低音質）で、小さいほど低圧縮率（高音質）になる。音の違いは、実際にエンコードして確認してほしい。

代表的なMP3プレーヤ

さっそく、できあがったMP3ファイルを試聴してみよう。Linuxで使える代表的なMP3プレーヤはmpg123とxmmsで、どちらもマシンへの負荷が少なく、Windowsでよく経験する音の途切れなども少ない。

自分の好みに合ったプレーヤでパソコン上でのオーディオライフをどうぞ。

■mpg123［tar.gz］
http://www.mpg123.de/
mpg123-0.59r.tar.gz

コマンドラインで使えるMP3プレーヤのスタンダード的な存在（**画面5**）。ディストリビューションによっては最初から入っているものもある。

動作が軽く、Xを起動せずに使えるので、非力なマシンでも安定して再生できる。

また、コンパイルオプションでALSAへの対応やesdへの出力も可能になる。素直にOSS向けにインストールするなら次のようになる。

```
$ tar xvfz mpg123-0.59r.tar.gz
$ cd mpg123-0.59r
$ make linux
$ su
# make install
```

使い方は簡単で、次のようにコマンドに続いてMP3ファイルの名前を入力するだけで良い。

```
$ mpg123 Nazz_-_Open_My_Eyes.mp3
```

コマンドラインでは曲の呼び出しが面倒のように思えるが、Gripでエンコードした場合はアーティストやアルバムタイトルでディレクトリが作成されているし、キーボード操作に慣れた人には便利だろう。また、-@オプションを使えば、Gripが自動生成したプレイリストファイル（〜.m3u）からの再生も可能だ。

```
$ mpg123 -@ nazz-nazz.m3u
```

この場合、アルバムに入っていた曲順で再生される。

■GQmpeg［rpm/tar.gz］
http://gqview.netpedia.net/
gqmpeg-0.6.3-1.i386.rpm
gqmpeg-0.6.3.src.tar.gz

画面5　コンソールベースのMPプレーヤー「mpg123」

画面6　mpg123のフロントエンド「GQmpeg」

画面7　ID3タグを利用したGQmpegの「プレイリストエディタ」

mpg123には非常に多くのフロントエンドが存在するが、ID3タグをフルに使ったプレイリストエディタを持つのが、このGQmpegである（**画面6**、**7**）。ID3タグの項目をキーにした並べ替えや、演奏時間の合計などの表示が可能（ただし、VBRでエンコードされたファイルは正しい演奏時間は表示されない）。

これらの表示を有効にするには、プレーヤ画面上を右クリックして表示されるポップアップメニューから［Preference...］を選択して起動する［Preference］ダイアログボックスで設定する必要がある。具体的には、［Preference］ダイアログボックスの［General］タブで、［Read file headers］と［Use ID3 tags when available］のチェックをオンにしておけばよい。

tarボールからインストールする場合は、次のようになる。

```
$ tar xvfz gqmpeg-0.6.3.src.tar.gz
$ cd gqmpeg-0.6.3
$ make
$ su
# make install
```

インストールにはgtk1.2.0以降のほか、Imlib1.9以降のバージョンが必要になる。もしも、自分のマシンにImlibがインストールされていない場合は、ftp://ftp.enlightenment.org/pub/enlightenment/enlightenment/libs/からダウンロードできる。当然のことながら、GQmpegの動作にはmpg123が必要になる。また、mpg123とGQmpegの実行ファイルは、PATHの通った同じディレクトリになければならないことにも注意。

■xmms［rpm/tar.gz/deb］

http://www.xmms.org/

xmms-1.0.1-1.i386.rpm（glibc2.1用）

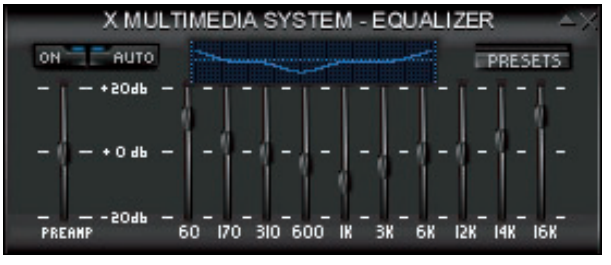
xmms-1.0.1.tar.gz

xmms_1.0.1-1.deb

その昔、X11Ampという名前で知られていたGUIのプレーヤ（**画面8**、**9**）。プレイリストエディタはGQmpegほど機能的ではないが、その代わり10バンドのグラフィックイコライザを持つ（**画面10**）。ALSAやesdにはプラグインで対応。エコーやサラウンドなどのエフェクトや、外部のスペクトルアナライザなどもプラグインでサポート（**画面11**）。ネットワーク経由のストリーム再生にも対応していてネットラジオも聴ける。

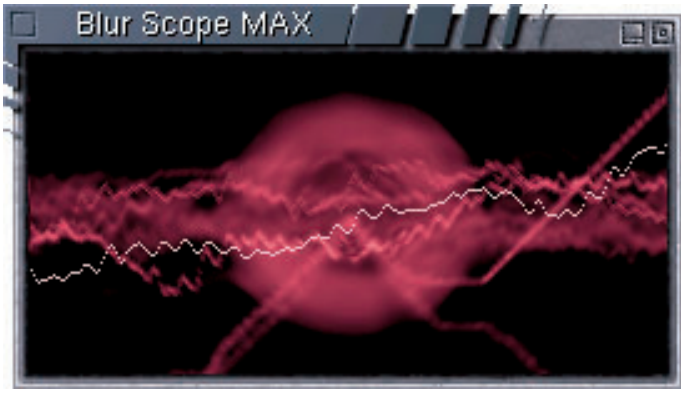
```
$ tar xvfz xmms-1.0.1.tar.gz
$ cd xmms-1.0.1
$ ./configure
$ make
$ su
# make install
```

画面8　X11Ampから名称変更した「xmms」
「スキン」と呼ばれるテーマ機能（画像を変更して、見栄えを変える）による高いカスタマイズ能力が、xmmsの人気の秘密。http://www.xmms.org/skins.html/やhttp://www.winamp.com/から登録されているスキンをダウンロードできる。



画面10　xmmsの「グラフィックイコライザ」

画面9　GQmpegの「プレイリストエディタ」



画面11　xmmsの「プラグインビジュアライザ」
テクノ野郎向けのプラグインビジュアライザ“Blur Scope MAX”。音楽に合わせて複雑な表示をする。表示傾向についても設定できる。

Section 2

グラフィック

Sound Graphic Video

ここでは、画像取り込みからGimpによる加工処理のテクニックまでを紹介します。

画像取り込みのテクニック

文：藤沢敏喜　Text：Toshiki Fujisawa

パソコン上で、写真の画像を入出力するためには、スキャナやプリンタといったデバイスが必要になる。しかし、このようなデバイスは、ハードディスクやMOなどと違って、製品ごとに仕様がバラバラであるため、そのコントロール方法は統一されておらず、メーカー独自であることが多い。

デジタルカメラやスキャナの場合、WindowsやMacではTWAINと呼ばれる規格が定められており、その規格やサンプルソースコードがWebサイト（http://www.twain.org/）上で公開されている。しかしこの規格は、アプリケーションとの画像データのやり取りの方法だけを定めたものであり、ハードウェアアクセスやGUIといった部分は各メーカーから提供されているドライバによって実現されている。

このドライバは、一般的にオブジェクトコードが提供されているだけなので、ユーザーからは完全なブラックボックスとなっている。したがって、メーカーがドライバを提供するWindowsとMac以外のOSから利用することは非常に難しい状況である。

しかし、PC-UNIXユーザーの多大な努力により、最近ではデジタルカメラの画像を取り込んだり、フラットベッドスキャナやフィルムスキャナで画像をスキャンできるような環境がかなり整ってきている。

なお、ひと口に写真画像といっても、次のようにさまざまな入力方法がある。

- **デジタルカメラ**
- **写真店経由でのPHOTO-CD**
- **フラットベッドスキャナ**
- **フィルムスキャナ**

ここでは、これらのデバイスからLinux上で画像入力する方法について紹介しよう。

デジタルカメラからシリアル経由で入力

最近では、USBインターフェイスを搭載したデジタルカメラが発売されるようになってきているし、LinuxでもUSBがサポートされつつある。

しかし現時点では、USBを利用したデジタルカメラからLinuxへの画像転送は、ごく一部のカメラで実現されているだけであり、まだまだシリアルインターフェイス（RS-232C）経由が主力である。

Linux上で、デジタルカメラからシリアル経由で画像を転送するアプリケーションとしては、gPhoto（http://www.gphoto.org/）が有名である。このgPhotoは、X Window System上で動作するアプリケーションで（GTK+を使って書かれている）、GPLに則って配布されている。

gPhotoがサポートしているデジタルカメラは105機種以上になるが、そのほとんどは、Windowsアプリケーションとの通信時にシリアルラインを観測して、どのようなプロトコルで動作しているかを解析して作られたものである。この観測ツールはsnooperと呼ばれ、萩野純一郎氏（http://www.itojun.org）によって開発されたものである。このツールのおかげで、gPhotoで多くのデジタルカメラがサポートされるようになった。ただし、gPhoto自体の開発者向けのメーリングリストには日本人があまり参加していないようである。デジタルカメラのほとんどの機種が日本製であることを考えると、これはちょっと残念なことである。

1. gPhotoのインストール

Linux上でgPhotoを利用するなら、http://www.gphoto.org/gphoto/download.htmlに、RPMやDEB形式のバイナリパッケージが用意されているので、ここからダウンロードするか、付録CD-ROMのLinuxmag/multimedia-tools/Graphic/gphotoディレクトリにあるバイナリデータを利用してほしい。

たとえば、公認Red Hat Linux 6.1日本語版（以下、Red Hat 6.1）で動作させるには、

gphoto-0.4.2-1.i386.rpm

をダウンロードし、ルート権限で、

```
# rpm -U gphoto-0.4.2-1.i386.rpm
```

としてインストールを行えばよい。

ただし、このRPMでは、glibc2.1が要求されるため、TurboLinux 4.2のようにglibc 2.0ベースのディストリビューションではうまく動作しない。その場合は、同サイトにある、tarボール（gphoto-0.4.2.tar.gz）をダウンロードするか、付録CD-ROMを利用して、

```
$ tar xvzf gphoto-0.4.2.tar.gz
$ cd gphoto-0.4.2
$ ./configure
$ make
$ su
# make install
```

とすればよい。

2. 起動とカメラの設定

gPhotoを起動するには、コマンドラインから、

```
$ gphoto
```

とする。起動すると、シリアルポートを設定するダイアログボックスと、開発者の名前を表示するウインドウが表示される。それらを閉じると、**画面1**のようなメインウインドウが表示される。

まず、デジタルカメラの機種と接続されているシリアルポートの設定が必要になる。これは、[Configure]メニューから[Select Port-Camera Model]を選択して表示されるダイアログボックスで設定できる（**画面2**）。

3.インデックス画面表示と転送画像の選択

[Camera]メニューから[Get Index]－[Thumbnails]を選択すると、画像のインデックス画面が表示される。このインデックス画像の中から転送したい画像をマウスでクリックすると、その画像の枠が赤くなり、選択される（**画面3**）。

そして、[Camera]メニューから[Get Selected]－[Images]－[Open in window]を選択すると、選んだ画像が次々に取り込まれていく。取り込まれた画像は、上部に現れるタブをクリックすれば、取り込んだ画像を表示できる。また、[File]メニューから[Save Opened Image(s)]を選んで、画像を保存することもできる。

CFカードやスマートメディアを読む

gPhotoを使えば、シリアルポート経由で画像を取り込むことができる。しかし、数百万画素が当たり前になりつつある昨今のデジタルカメラでは、RS-232Cの通信速度はあまりにも遅く、かなりの忍耐が要求されるはずだ。

また、デジタルカメラを本格的に使いだすようになると、コンパクトフラッシュ（CF）カードやスマートメディアを何枚も持ち歩くことになる。家へ帰ってから何枚ものカードをカメラに入れ替えながら、速度の遅いシリアルポート経由で転送するというのは耐えがたい作業になるはずだ。

そこで、スマートメディアを利用して、シリアルポート経由よりも高速に画像転送をする方法も紹介しておこう。

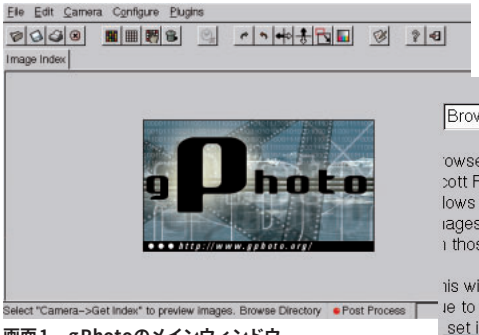
■ノートPCで読み込む

最近のノートPCにはPCカード用のスロットがあるので、CFカードやスマートメディアなどを、PCカードへの変換アダプタを使うことにより、直接マウントしてファイルにアクセスすることができる。

たとえば、Red Hat 6.1をノートPCにインストールした環境なら、スマートメディアをPCカードアダプタに入れ、

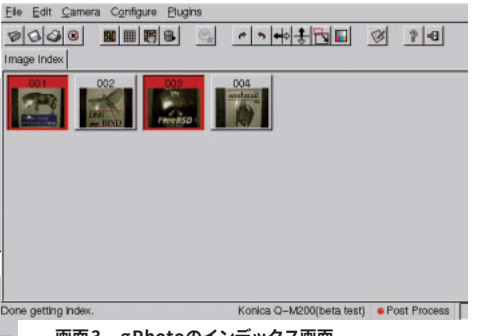
```
# mount -t msdos /dev/hde1 /mnt
```

と入力することにより/mntディレクトリへマウントすることができる。なお、筆者が所有しているノートPCとスマー



画面1　gPhotoのメインウィンドウ

画面2　シリアルポート設定のダイアログボックス



画面3　gPhotoのインデックス画面

トメディアの組み合わせでは、最初のアクセスの際に、

```
DriveReady Seek Complete Error
DriveStatus Error
```

というメッセージがコンソールに出力されるが、実際の読み取りには支障がないようである。

ちなみに、ソニーのVAIOなどでは、メモリスティックスロットが標準搭載されているが、手元にあるFreeBSDを載せたPCG-Z505FXでは、

```
# mount -t msdos /dev/wd2s1 /mnt
```

としてメモリスティックもマウントできる。このマシンは、Linuxのインストールが面倒なので（CD-ROMドライブが手元にない）試していないが、Linuxでも/dev/wd2s1の代わりに/dev/hdcとすることで同様にマウントできると思う。

■カードリーダで読み込む

ノートPCを所有していない場合や、デスクトップマシンで画像を管理したいなら、カードリーダを利用すればよいだろう。PCカードリーダには、主に3つのタイプがある。それぞれの特徴は次のとおりだ。

・USB接続の特定媒体専用リーダ

これは、最近Windowsなどで主流になりつつあるタイプで、CFカードやスマートメディアをそのまま差し込める製品など、さまざまなものが発売されている。これらの製品は手軽で便利なのだが、PC-UNIX環境ではストレージクラスのUSBサポートはまだ未熟で、実際使うにはまだ早いようだ。

・ISAバスに接続する汎用のPCカードリーダ

このタイプはノートPCのPCカードスロットと同じ環境をデスクトップマシンで再現できるようにするもので、CFカードやスマートメディアだけでなく、PCカードのネットワークカードやモデム、PHS、SCSIカードなどもデスクトップマシンで使うことができるので、汎用性は高い。ただし、ISAバスが存在しないマザーボードでは使えないので、新しいマザーボードを購入すると使えなくなる可能性もある。また、PCカードを使うためにはpcmcia-csの設定が必要になるディストリビューションもあり、多少手間がかかる。

・SCSI接続の記録系カード専用のリーダ

このタイプは、SCSIホストアダプタが別途必要になる。また、CFカードやスマートメディアといった記録媒体だけしか扱えず、PCカードタイプのネットワークカードなどを使うことはできない。

しかし、最近のSCSIホストアダプタは安価になっているし、SCSI接続のMOやスキャナなどの接続にもSCSIホストアダプタは必須なので、SCSIホストアダプタが必要ということはさほどデメリットとはいえないだろう。

また、デスクトップマシンで、PCカードのモデムカードやネットワークカードを使う必然性もあまりないし、将来ISAバスがないマザーボードに交換した場合でも問題なく使える。そして、SCSIなら標準カーネルで認識されるので、特別な設定は必要なく、簡単である。

このような理由により、筆者の自宅のマシンには1万円弱で売られているSCSIタイプのカードリーダを搭載している。

このSCSIタイプのカードリーダも、ノートPCと同じように、

```
# mount -t msdos /dev/sda /mnt
```

としてマウントすることができるので、あとは普通のストレージと同じように読み書きするだけである。

写真店経由でのPhoto CD入力

フィルムメーカーは世界で4社（アグファ、コダック、コニカ、フジ）しか

写真　コンパクトフラッシュとPCカードソケット
PCカードソケットに装着することにより、Linux上からもコンパクトフラッシュ上の画像データにアクセス可能となる。写真は、アイ・オー・データ機器のPCCF-64M（3万1000円）。

写真　スマートメディアとPCカードスロット
スマートメディアもPCカードスロットに装着することにより、Linuxからアクセスが可能となる。写真のPCカードスロットは、アイ・オー・データ機器のPCFDC IV-ADP（8000円）。

写真　SCSI接続の外付けカードリーダ
コンパクトフラッシュ、スマートメディア、マイクロドライブ、フラッシュATAカード、ATAハードディスクといった記録系のカードが読み込めるようになる。写真は、メルコのMCR-S（1万9800円）。

ないが、各社とも銀塩フィルムからデジタル画像へ変換するサービスを提供している。その中でも歴史的に有名なのがPhoto CDで、最初に提唱したコダックだけでなく、コニカやフジでもこの形式に画像を保存するサービスを行っている。

Photo CDは、通常のCD-ROMと同じく、

```
# mount -r -t iso9660 /dev/cdrom /mnt
```

としてマウントでき、Photo CDの中のimageディレクトリには、image00xx.pcdというようなPCD形式の画像が保存されている。PCD形式では、複数の解像度の画像が1つのファイルの中に保存されているが、これをUNIX環境での標準形式であるpnm（ppm）に変換するツールがhpcdtoppmコマンドで、ftp://ftp.gwdg.de/pub/linux/hpcdtoppmでも入手できる。かなり古いツールではあるが、Red Hat 6.1でもコンパイルできた。

このコマンドには**表1**のようなオプションがあり、PCD形式の中から目的の解像度の画像を取り出すことができる。

したがって、次のようにパイプで流し込むことにより、さまざまなフォーマットで保存できる。

```
# hpcdtoppm -4 image00xx.pcd | cjpeg > out.jpg
# hpcdtoppm -2 image00xx.pcd | ppmtobmp -windows -24 >out.bmp
```

また、画像操作ソフトウェアであるImageMagickや、シェアウェアのxvでも直接Photo CDファイルを扱うことができる。

サービスサイズの写真プリントからの入力

デジタルカメラがはやっているとはいえ、他人からもらった写真、そして数十年前の思い出の写真など、デジタル化されていない銀塩写真プリントもまだまだ多い。そして、これらもデジタル化したいと思うことも少なくないに違いない。

また、現在のデジタルカメラは、解像度に関してはそれなりに進歩したといえるが、階調再現の点では銀塩フィルムに大きく劣っている。たとえば、顔に強い陰ができるような晴天時の記念写真や、奥行きのあるストロボ撮影では、デジタルカメラで撮影するよりもレンズつきフィルムで撮影した銀塩写真プリントをフラットベッドスキャナでスキャンしたほうがよい画像が得られることが多い。

そこで、フラットベッドスキャナからのデータ取り込みについても触れておこう。Linuxからフラットベッドスキャナを利用するには、SANEというアプリケーションを使うのが定番であり、数多くのフラットベッドスキャナがサポートされている。

ここでは、筆者が所有しているEPSONのフラットベッドスキャナ「GT-7600S」を例にとって、SANEからの利用について解説しよう。

オプション	機能
-0	オーバービューファイルからのサムネイル画像の抽出
-1	イメージファイルからの128x192サイズのファイルの抽出
-2	イメージファイルからの256x384サイズのファイルの抽出
-3	イメージファイルからの512x768サイズのファイルの抽出
-4	イメージファイルからの1024x1536サイズのファイルの抽出
-5	イメージファイルからの2048x3072サイズのファイルの抽出

表1　hpcdtoppmのオプション

ちなみに、このスキャナにはUSBモデルとSCSIモデルがあり、両モデルとも定価が4万4800円である。しかし、去年の秋に購入したときの実販売価格は、USBモデルが約2万5000円、SCSIモデルが約3万4000円であった。USBモデルの価格は魅力的ではあるが、Linuxユーザにとっては、SANEから利用できて、USBよりも高速なSCSIモデルがお勧めである。

■動作させたマシンの環境

今回、フラットベッドスキャナを接続したマシンとその環境は**表2**のとおりである。今となっては古くて貧乏くさいハード構成だが、自宅のマシンの中では、これが一番速いマシンである。画像を扱うにはもう少し速く、メモリの多いマシンがほしいところではあるが、この程度のマシンでもまあなんとか使える。

ただし、このマシンにはFreeBSDとWindows、そしてLinuxディストリビューションを多数インストールしてあるため、画像データを扱うにはディスクが足りない。そのため、6台のHDDと7連奏CD-ROMを接続し、前述したSCSIタイプのPCカードリーダを内蔵したFreeBSDマシンを画像保存用ファイルサーバと使っている。

さて、フラットベッドスキャナを接続する場合、SCSIホストアダプタに注

写真　EPSONのフラットベッドスキャナ「GT-7600S」4万4800円 。透過原稿ユニット：1万円（オプション）。

意する必要がある。Linuxでサポートされ ているSCSIホストアダプタでも、ハードディスクやCD-ROMドライブは動作しても、スキャナなどの特殊な機器を接続すると動作しないものは多い。数年前、スキャナデバイス用にFreeBSDとLinuxで数十枚のSCSIホストアダプタを評価したことがあるが、まったく動作しないものや、画像転送をしている途中で転送が止まってしまったり、最悪カーネルごと落ちてしまうものまであった。

また、フラットベッドスキャナ以外のSCSI機器を同じバスに接続すると動作しなくなることもある。

このため、SCSIホストアダプタについては業界のスタンダード的なメーカーの、広く使われているものを選択し、さらにSCSIホストアダプタにはフラットベッドスキャナ以外のSCSI機器を接続しないのが賢明である。

特に、メールサーバのような重要な働きをしているマシンに、フラットベッドスキャナを接続するのはやめておいたほうが無難である。もしも、そのようなマシンでどうしても使う必要があるようなら、SCSIホストアダプタを2枚挿すなどの対策をする必要があるだろう。

ちなみに、筆者はSCSIのハードディスクを好んで使っているのだが、フラットベッドスキャナを接続しているマシンでは、ハードディスクとCD-ROMドライブはIDEタイプにして、SCSIホストアダプタにはスキャナ以外の機器を接続しないようにしている。

なお、このようなスキャナデバイスに関する注意点はLinuxに限ったことではなく、Windowsでも同様のことがいえる。

■SANEのインストール

SANEは、公式Webページ（http://www.mostang.com/sane/）から、ソースコードをダウンロードできる。また、Red Hat 6.1のフルインストール環境なら、/usr/doc/JFディレクトリ以下にSANE-outline-JP.txt.gzとSANE-tutorial-JP.txt.gzというドキュメントが2つ用意されている。

SANEにはRPM形式のパッケージも用意されているが、ここではいつでも最新版を使えるようにするためコンパイルを行ってみることにする。コンパイルといっても、GTK+などの環境が整っていれば非常に簡単である。Red Hat 6.1をフルインストールした状態では、ソースを展開し、ルート権限で、

```
# ./configure
# make
# make install
```

を行うだけである。

なお、sane-1.0.0は、Red Hat 6.1でコンパイルするとエラーになるので、sane-1.0.1をダウンロードする必要があるので注意してほしい。

インストールが終わったら、スキャナがどのデバイスに接続されているかをdmesgコマンドや、SANEで提供されるtools/find-scannerコマンドを使って調べる。

ちなみに、SCSIバスに1台のスキャナを接続している場合なら、/dev/sgaとなっているはずなので、ルート権限で、

```
# chmod 666 /dev/sga
# ln -s /dev/sga /dev/scanner
```

を実行しておく。また、/dev/scannerがない場合や、/dev/sgaにアクセス権限がない場合、後述のxscanimageコマンドの実行時に「xscanimage: no devices available.」というエラーメッセージが表示されるので、リンクやモードが正しいかをよく確認しておこう。

■スキャン

画像を取り込むアプリケーションとして、SANEに標準で用意されているxscanimageコマンドを利用した。xscanimageを起動すると、初期設定の画面が表示される（**画面4**）。ここで、スキャンモードを［Binary］から［Color］に変更してから、［Scan resolution［dpi］］のスクロールバーでスキャンする解像度を設定する（**画面5**）。

次に、［Preview Window］ボタンをクリックすると、プレビューウインドウが表示され、左下にある［Acquire Preview］ボタンをクリックするとプレビューが始まるので、スキャンしたい範囲をマウスで選択する（**画面6**）。

選択が終わったら、先ほどの設定ウインドウにある［Scan］ボタンをクリックすれば、本スキャンが始まる。本スキャン画像は、デフォルト設定では“out.pnm”というファイル名で保存される。このPNM形式は、UNIX環境でよく使われるフォーマットで、Gimpなどでも表示が可能である。

パーツ	製品名
マザーボード	ASUS T2P4/AT（K6-2用に改造）
プロセッサ	K6-2/300MHz（ファンレス巨大放熱板付）
電源	静音ATタイプ（サイレンサー）
メインメモリ	128Mバイト
ハードディスク	IDE 10Gバイト
CD-ROMドライブ	IDE
SCSIホストアダプタ	Tekram DC390U
OS	公認Red Hat Linux 6.1日本語版

表2　筆者が試したマシン環境

フィルムスキャナを使った画像入力

フラットベッドスキャナを使って、プリントされた写真からスキャンができれば、フィルムスキャナは必要ないように思えるかもしれない。しかし、得られる画質には非常に大きな差があり、そのメリットは少なくない。

■フィルムスキャナを使うワケ

読者の多くの方も経験があると思うが、同じネガフィルムからのプリントでも現像所によって濃度や色が大きく違う。これはなぜかというと、ネガフィルムに再現されている階調情報が膨大なのに対し、プリントで再現できるのはその一部にすぎないからである。

たとえば、日向と日陰が混在しているシーンがある場合、日向の部分をきれいにプリントしようとすると、日陰の部分は真っ黒になり、日陰の部分を再現しようとすると、日向の部分は真っ白に飛んでしまう。この場合、どちらを優先して再現するかは現像所のプリント担当者の主観で決定される。また、色についても、ネガからは正しい色を判断することができず、プリントする人間の主観に左右される。

その点、フィルムスキャナを使えば、ネガフィルムが持つ豊富な情報から自分の好みの画像を引き出すことができる。また、ネガフィルムからの写真というものは、あくまでもコピーにしかすぎず、オリジナルのネガフィルムを直接スキャンするほうが画質がよいのは当たり前だ。

ちなみに、GT-7600Sには、オプションとして透過原稿ユニットが用意されていて、筆者も4×5インチのポジフィルムスキャンに用いているが、フィルムとガラス面との接触によるニュートンリングが頻繁に発生するため、フィルムのセットが難しく、操作性の面でもフィルムスキャナが有利である。

■Linuxで使えるフィルムスキャナ

さて、Linuxから利用できるフィルムスキャナには、ニコンやキヤノンの数機種が前述のSANEを使って利用できるようである。このほかに、Linux（FreeBSDも）で使えるフィルムスキャナとして、コニカのQscanシリーズがある。このQscanには、xqscanというアプリケーションが1997年から提供されている。なお、Qscanの最初の初期型は1996年に発売されたが、現在でも全国の写真店からの注文が可能である。

■SANEとxqscanのユーザインターフェイス

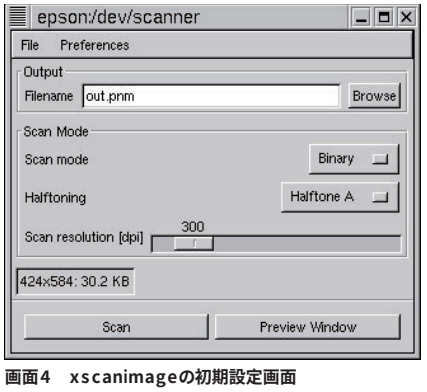
SANEは、元々フラットベッドスキャナのようにダイナミックレンジの狭い反射原稿をスキャンすることをメインに考えた設計になっている。しかし、フィルムスキャナは、ネガフィルムの濃度が激しく違うなどダイナミックレンジが広いため、フィルム濃度によってプレスキャンゲインを変えたり、人間の主観によって本スキャン時のRGBのアンプゲインを設定しないと満足する画質を得ることができない。

また、SANEのユーザーインターフェイスは、1枚ずつのスキャンが前提になっており、Qscanのように6コマを一度に連続してスキャンできるようなスキャナには向いていない。

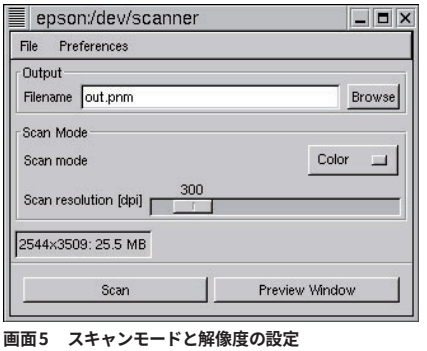
その点、これから紹介するxqscanは、このような点にも配慮したユーザーインターフェイスになっている。

■インストール

xqscanは、付録CD-ROMにも収録されているし、QscanのWebサイト（http://www.konica.co.jp/qscan/）からダウンロードすることも可能である。xqscanでは、さまざまなさまざまなディストリビューションが存在することを考慮して、



画面4　xscanimageの初期設定画面



画面5　スキャンモードと解像度の設定

画面6　xscanimageのプレビューウィンドウ

写真　コニカのフィルムスキャナ「Qscan」
ネガフィルム、ポジフィルム、APSフィルムの連続スキャンが可能。LinuxやFreeBSDのドライバも用意されているのがうれしい。7万8000円。

```
# cd /
# tar zxf /xxx/yyyy/ja-xqscan-linux-X_XXA.tgz
# xqscan-install
```

というような単純な方法でインストールすることが可能になっている。

また、xqscanはSANEとは違って、/dev/sga、/dev/sgb、/dev/sgc…というように自動的にマシンに接続されたQscanを探し出してくれるため、SANEの設定では必要になる/dev/scannerへのリンクを張ったりする必要はない。ただし、/dev/sg[a-z]以外のデバイスを使っている特殊なディストリビューションでは、後述の例のように環境変数XQSCAN_DEV_NAMEにデバイス名を設定する必要がある。

なおxqscanでは、Qscanが見つからなかったときは自動的にデモモードになるので、Qscan自体を持っていなくても、操作や画面表示などを確認することが可能である。

■スキャン

マニュアルモードでxqscanを起動すると、設定画面が表示される（**画面7**）。ここで、［スキャン開始］ボタンをクリックするとプレスキャンが始まり、プレスキャン画面が表示される（**画面8**）。ここで各画像の回転指定や、画像をクリックして色調の選択が行える（**画面9**）。なお、この色指定によってアンプのゲインを変えているため、ここで色指定をしたほうが、本スキャンしたあとに色を変化させるよりも階調豊富な画像が得られる。

6コマ分の色指定が終了すると、6コマ分を連続して取り込んだあと、コマごとに番号をつけたファイル名として保存される。

■外部コマンドの呼び出し

UNIX環境ならではの機能として、xqscanには画像を保存する際に外部コマンドを呼び出す機能が用意されている。

これは、環境変数XQSCAN_IMAGE_PIPEに外部コマンドを指定しておくことで、ファイルを保存する際にこの外部コマンドを呼び出すというものだ。

これにより、pnm（ppm-raw）形式で入力されたデータに外部コマンドを経由させて出力することによって、保存形式などを変更することができるのだ。

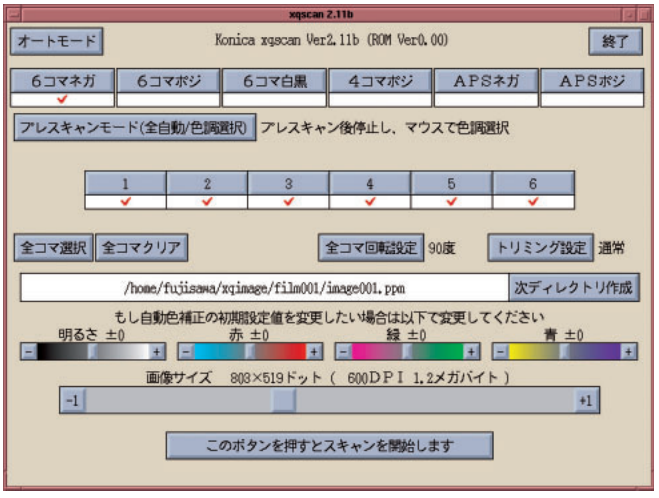
たとえば、

```
$ export XQSCAN_IMAGE_PIPE="pnmtotiff"
$ export XQSCAN_IMAGE_FORMAT="tif"
```

という指定によりTIFF形式での保存が可能になる。

■2台のQscanでの並列動作

メーカーでは正式に保証していないものの、SCSIホストアダプタを2枚以上挿し、次のように環境変数を設定すれば、2台以上のQscanを並列に動作させることができる。



画面7　xqscanの設定画面

画面8　プレスキャン画面

```
$ export XQSCAN_DEV_NAME=/dev/sga
$ export XQSCAN_TMP_DIR=/home/xxxx/xqtmp0
$ export XQSCAN_IMAGE_DIR=/home/xxxx/xqimage0
$ export XQSCAN_GEOMETRY=+0+0
$ xqscan &
$ export XQSCAN_DEV_NAME=/dev/sgb
$ export XQSCAN_TMP_DIR=/home/xxxx/xqtmp1
$ export XQSCAN_IMAGE_DIR=/home/xxxx/xqimage1
$ export XQSCAN_GEOMETRY=-0+0
$ xqscan &
```

フィルムのスキャンには時間がかかるので、大量にスキャンさせる場合などは、このように交互に動作させて、フィルム交換を行えば便利である。これもWindows版には用意されていない機能で、Linux/FreeBSD版ならではの使い方といえよう。

写真画質画像の印刷

さて、さまざまなデバイスからの画像入力について述べてきたが、ここで写真画質の印刷について考えてみよう。

現状では残念ながら、Linuxからはダイレクトに写真画質の印刷を行うことができない。しかし、Linuxから簡単に印刷する方法がないわけではない。たとえば、先月号の211ページからの連載「プログラミング工房」で紹介したように、WindowsマシンをLinuxのプリントサーバとしてしまう方法がある。詳しくは、先月号の記事を参照してほしいが、この方法を応用すれば、Gimpなどから、あたかも直接プリンタが接続されているかのように写真画質の印刷を行うことができる。

まだ、筆者は試していないのだが、この記事を読んだ読者からの情報によると、Win32版のGhostscriptと、Redmonなどのプリンタポートをリダイレクトするプログラム、そしてSambaなどを組み合わせることでも同様のことができるようである。

さらに、こちらもまだ、実際に試してみたわけではないので詳細は不明だが、EPSON製のプリンタ（PM-700C／3300Cなど）限定ながら、アドバックシステム（http://www.advac.co.jp/）が発売を開始したプリンタユーティリティ「GPR」も期待が持てそうだ（**画面10**）。

大切な画像はやはり銀塩カメラで

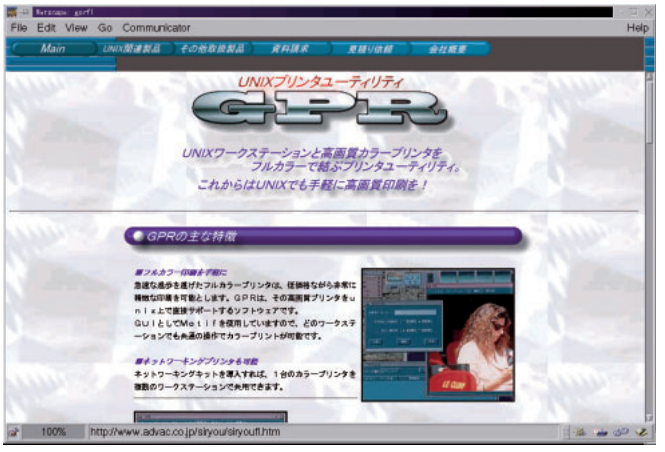
デジタルカメラは、Webに一時的に掲載することだけを目的とする宴会写真や、出張報告といった、特定の目的の利用には非常に便利である。

しかしながら、レンズのイメージサークルの小ささや、階調再現など当分解決の見込みがない問題も多い。また、デジタルデータは保存性に問題があると筆者は考えている。ディスクは頻繁にクラッシュするし、記録媒体の移り変わりが激しく、すぐに媒体変換を行わなければならない。そしてデータ移行の際にミスをすることも多く、10年以上にわたって画像データを保存しつづけることは非常に難しい。

したがって、七五三や結婚式、海外旅行などの人生において重要なシーンは銀塩フィルムで撮影し、100年プリントなどの紙媒体で保存することを基本とするのが、画質や保存の面で安心である。

銀塩カメラで撮影したシーンを、Web用や年賀状のためにデジタル化したい場合でも、今回述べたように、Linuxからでも簡単に取り込みを行うことができる。

画面9　xqscanの回転指定や色調選択



画面10　プリンタユーティリティ「GPR」

GimpによるGIFアニメーション作成

文：豊福剛　*Text : Tsuyoshi Toyofuku*

Gimp（GNU Image Manipulation Program）は、当時UCBの学生だったPeter MattisとSpencer Kimballの2人によって作成された画像処理ソフトウェアである（http://www.gimp.org）。GPLに則って配布されているフリーソフトウェアでありながら、商用フォトレタッチソフトウェアの定番「Adobe Photoshop」に迫るほど機能は充実しており、Photoshopキラーとさえいわれている。最新の安定バージョンは1.0.4で、開発バージョンは1.1.17である。ほとんどの日本語ディストリビューションには標準で搭載されているので、**画面1**のような起動スプラッシュを見た方も多いだろう。

■Gimpは使えないのか？

機能が豊富で、しかもフリー。いいことづくめにも関わらず、Gimpを使うためにLinuxを始めたというデザイナーがいるかといえば、残念ながら未だにそういう声は聞いたことがない。機能的にまだ完全にはPhotoshopに追いついていないという面もたしかにある。しかしむしろ、画像処理ソフトウェアを常用するような人は、すでにPhotoshopで慣れてしまっているだろうし、グラフィックスに関する知識のない人にとっては、そもそも画像処理ソフトウェアを使う必然性そのものがない、というのが本当の理由のような気がする。

では、そういう人たちにもGimpの良さを知ってもらおうということで、今回はGIFアニメーションの制作をテーマに、簡単な画像処理からレイヤの使い方から、Gimpのマクロ言語「Script-Fu」の活用までを紹介する。GIFアニメーションそのものには興味はなくても、流れのなかで紹介する個々の処理は参考になると思うし、Photoshopユーザーにも Gimpの強力なマクロ機能は魅力だろう。

そういう意味では、Gimpの「機能」よりも「使い方」の紹介に重点を置くつもりだ。一応、**画面2**にツールボックスの説明を示しておくが、個々の機能を詳しく知りたいという方は、拙訳の『Gimpパーフェクトガイド』（Michael J. Hammel著、豊福 剛訳、エムディーエヌコーポレーション刊、2800円）などを参照してほしい。

■では作ってみよう

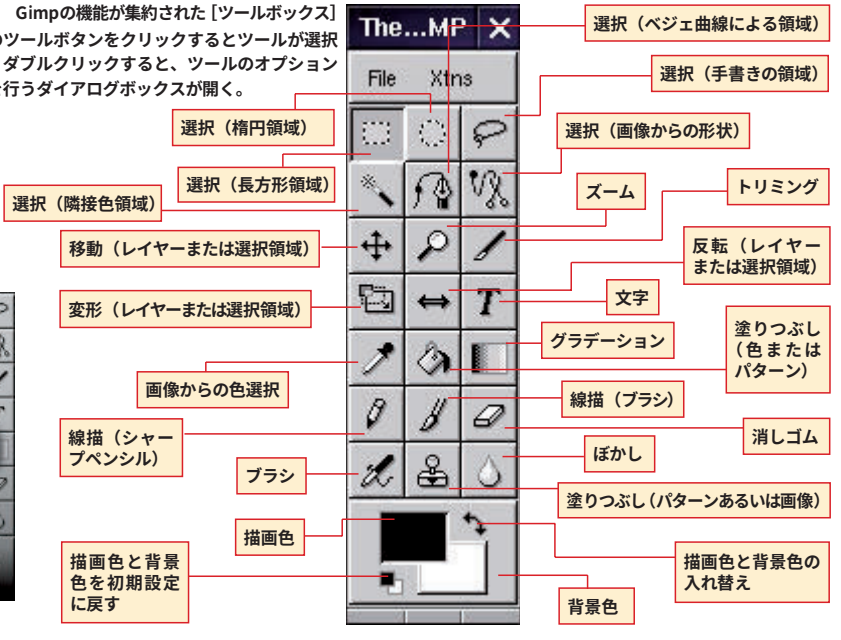
さて、さっそくGimpによるGIFアニメーションの作成手順を説明していこう。手順は、大きく3つの段階に分かれる。まず、ソース画像の準備、画像ソースのレイヤ化、そして、GIF形式での保存だ。それぞれの段階を色分けしてあるので、わかりやすいはずだ。

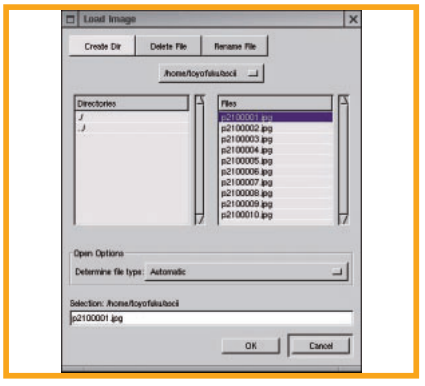
今回作成するのは、デジカメで撮影した素材をGIFアニメーション化するというもので、アニメなどの世界で人形などのポーズを1コマ1コマ撮影する、「クレイアニメ」という手法に近い。ただし、キャラクター人形の撮影は権利関係が難しいという理由から、今回はチェス盤が素材となっているのはご容赦いただきたい。

なお、筆者が使用したGimpのバージョンは1.0.2で、日本語化はされていないものだ。メニューなどの訳語はディストリビューションごとに異なるので、今回はTurboLinux PRO 日本語版 4.2に収録されているGimpの訳語をカッコ書きで示してある。

画面1 Gimp 1.0.2の起動画面

画面2　Gimpの機能が集約された［ツールボックス］任意のツールボタンをクリックするとツールが選択され、ダブルクリックすると、ツールのオプション設定を行うダイアログボックスが開く。





1. ソース画像の取り込み

ソース画像を取り込むには、ツールボックスの［File］（ファイル）メニューから［Open］（開く）コマンドを選択する。

［Load Image］ダイアログボックスが表示されたら、ソース画像のあるディレクトリまで移動する。［Files］（ファイル）リストボックスの一覧にあるソース画像を選択して、［OK］（了解）ボタンをクリックする。

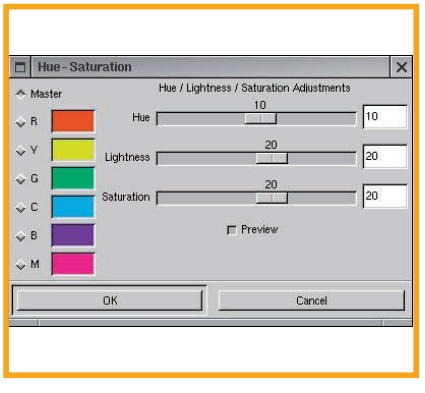
2. 画像ウィンドウの作成

ソース画像（この例ではp2100001.jpg）の読み込み処理を示すプログレスバーが表示されたあと、ソース画像に対応する画像ウィンドウが作成される。

3. ポップアップメニューの利用

マウスポインタを画像ウィンドウの中に置いて、右クリックすると、ポップアップメニューが表示される。このメニューを利用して、まず画像データの色相／彩度を調整してみよう。

ポップアップメニューから［Image］（画像）—［Colors］（色彩）—［Hue-Saturation］（色相-彩度）を選択する。



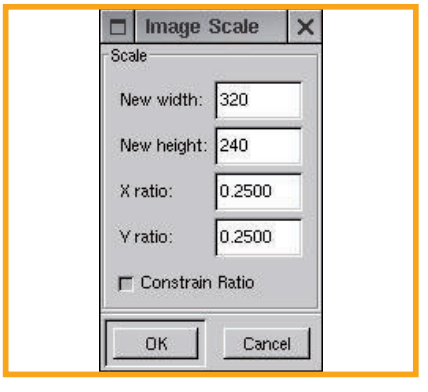
4. 色相／彩度の調整

［Hue-Saturation］ダイアログボックスが表示されたら、Hue（色相）、Lightness（明るさ）、Saturation（彩度）の値を調整する。ここで［Preview］（プレビュー）チェックボックスがオンになっていると、変更した設定が画像ウィンドウの表示にフィードバックされる。また、スライダバーの横にあるテキストボックスに数値を直接入力することもできる。

5. 画像サイズの変更

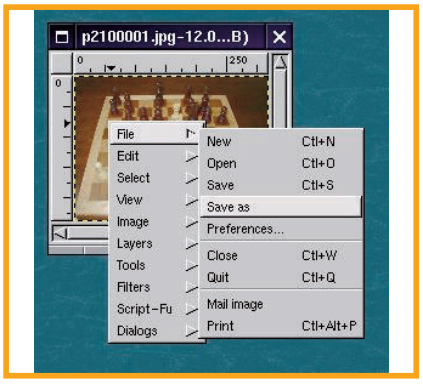
ソースとなっている画像データは、サイズが1280×960ピクセルもあり、GIFアニメーションにするには大きすぎるので、画像を縮小することにした。

画像を縮小するには、ポップアップメニューから［Image］（画像）—［Scale］（縮小拡大）を選択する。



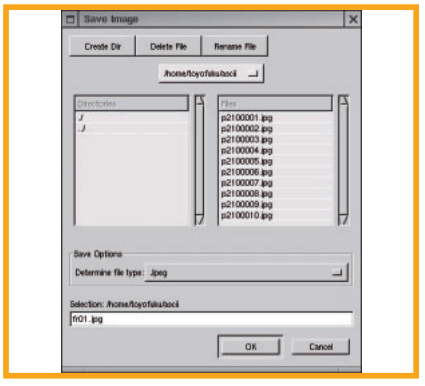
6. 画像サイズの縮小

［Image Scale］ダイアログボックスが表示されたら、縦横の長さを入力する。ここでは、幅「320」、高さ「240」にして、元の画像の1/16サイズに縮小する。



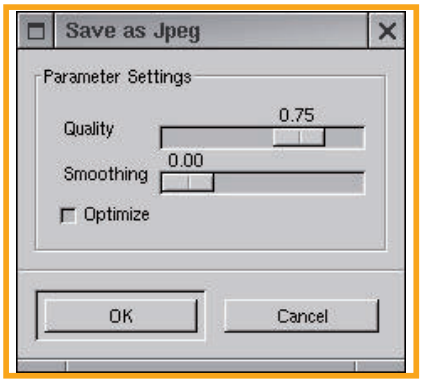
7. 画像の新規保存

ここまでのソース画像に対する変換操作が終わったら、ポップアップメニューから［File］（ファイル）―［Save as］（新規保存）を選択し、別ファイル名で保存しておく。



8. ファイル名の指定

［Save Image］ダイアログが表示されたら、新しいファイル名（ここではfr01.jpg）と入力して［OK］（了解）ボタンをクリックする。

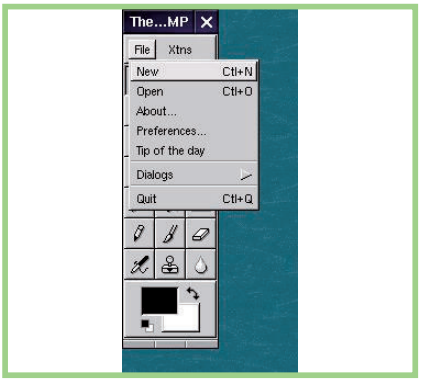


9. JPEG形式の保存

画像フォーマットとしてJPEG形式を選択すると、［Save as Jpeg］ダイアログボックスが表示される。

ここで、［Quality］の値を小さくすると、画像の圧縮率が高くなるが、その分画質は落ちる。［Smoothing］の値を大きくすると、画像がぼけた感じになる。

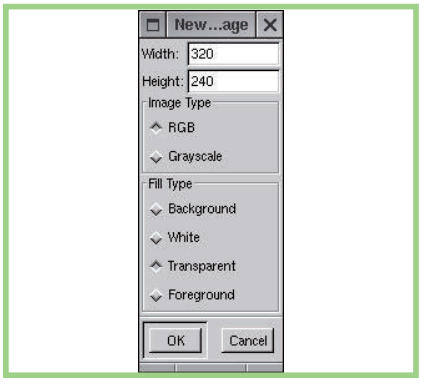
以上の操作を、残りのファイルについても同様に適用していけばよい。



10. レイヤ化の開始

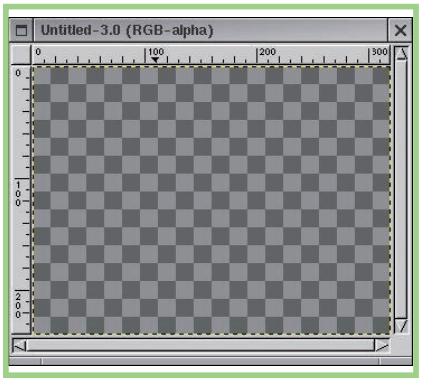
GimpをつかってGIFアニメーションを作成するには、GIFアニメーションの個々のフレームに対応する画像ソースを、レイヤにしておかなければならない。

レイヤを作成するには、まず空の画像ウィンドウを作成する。ツールボックスの［File］（ファイル）メニューから［New］（新規）コマンドを選択する。



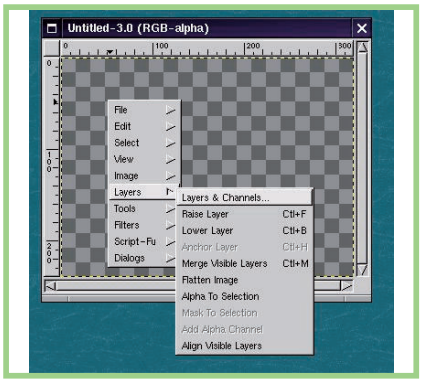
11. 新規画像のサイズ指定

［New Image］ダイアログボックスが表示されたら、作成する画像サイズを、縮小後の画像ソースの大きさに合わせて、幅「320」、高さ「240」に設定する。また、［Fill Type］（画像の種類）では［Transparent］（透明）を選択して、［OK］（了解）ボタンをクリックする。



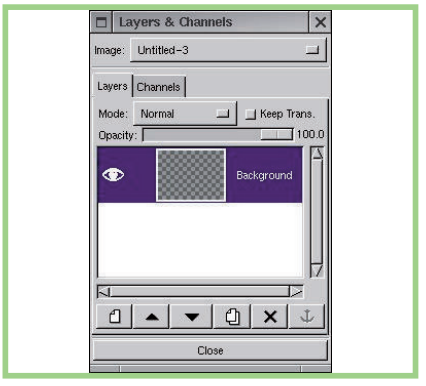
12. 新規画像ウィンドウの生成

新規画像ウィンドウが作成される。チェック模様の表示は、透明（Transparent）であることを示している。



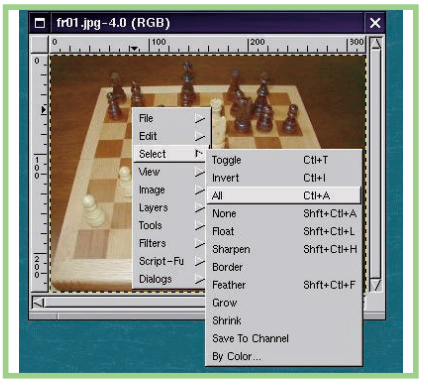
13. レイヤ作成の準備

新規画像ウィンドウ上で右クリックしてポップアップメニューを表示し、［Layers］（レイヤー）—［Layers & Channels］（レイヤー&チャネル）を選択する。



14. レイヤ状態の表示

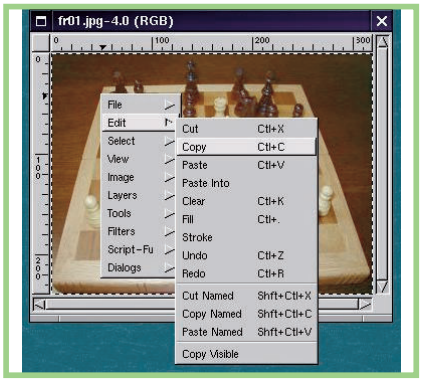
［Layers & Channels］ダイアログボックスが表示され、現在のレイヤは、Background（背景）1枚しかない状態であることがわかる。



15. 画像データの領域選択

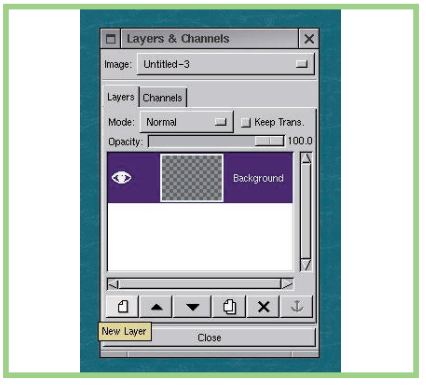
［File］—［Open］メニューを利用して、先ほど保存しておいた変換後の画像ファイルを開く。開いた画像ウィンドウのポップアップメニューから［Select］（選択）—［All］（全選択）を選択する。

これで、画像全体が選択範囲として指定された状態になる。



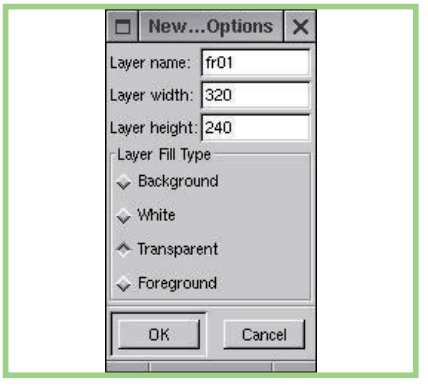
16. 選択した領域のコピー

ポップアップメニューから［Edit］（編集）—［Copy］（複写）を選択し、選択した領域をメモリに読み込ませて、あとで貼り付けられるようにしておく。



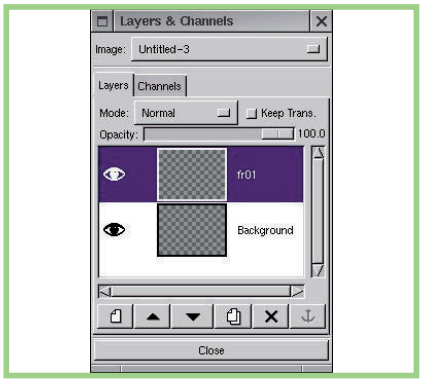
17. 新規レイヤの作成

［Layers & Channels］ダイアログボックスの下部にあるコマンドボタンの左端にある［New Layer］（新規レイヤー）ボタンをクリックする。



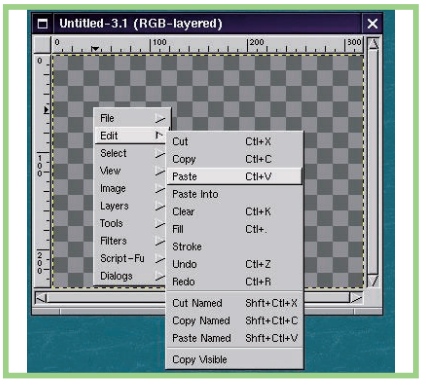
18. レイヤの設定

［New Layer Options］ダイアログボックスが表示されたら、［Layer name］（レイヤー名）に「fr01」と入力する。さらに、サイズや［Fill Type］（画像の種類）では［Transparent］（透明）に設定されていることを確認してから［OK］（了解）ボタンをクリックする。



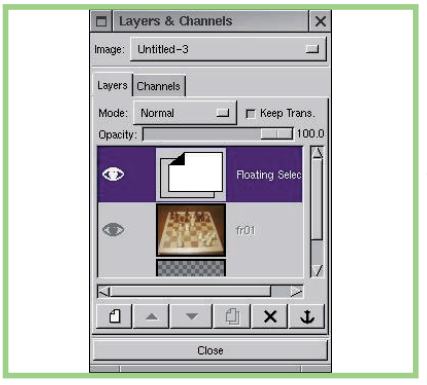
19. レイヤ状態の確認

再び、［Layers & Channels］ダイアログボックスを参照すると、先ほど作成した、「fr01」という名前の新規レイヤが追加されているのが確認できる。



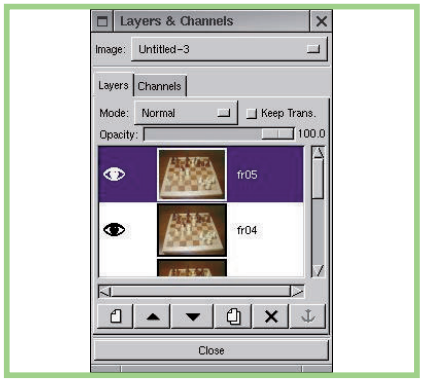
20. 画像の貼り付け

新規画像ウィンドウに戻って、ポップアップメニューから［Edit］（編集）―［Paste］（貼り込み）を実行すると、新規画像ウィンドウに先ほどコピーした画像がペーストされる。



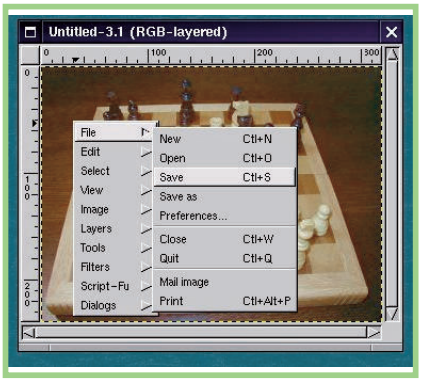
21. レイヤ状態の確認

［Layers & Channels］ダイアログボックスを参照すると、fr01レイヤの表示が変わり、さらに、Floating Selection（フロートレイヤ）が追加されていることが確認できる。このフロートレイヤは、選択範囲の指定に対応しており、フロートレイヤ上で、右クリックで表示されるポップアップメニューから［Anchor Layer］を選択すれば下のレイヤに結合される。



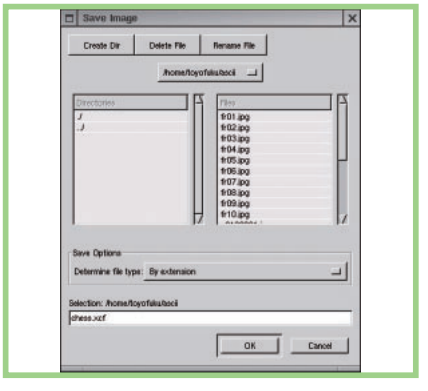
22. レイヤ作成の繰り返し

残りの画像ソースについても、新規レイヤの作成からコピー&ペーストまでの、一連の操作を繰り返す。すると、［Layers & Channels］ダイアログのレイヤに、画像が順に積まれていくはずだ。



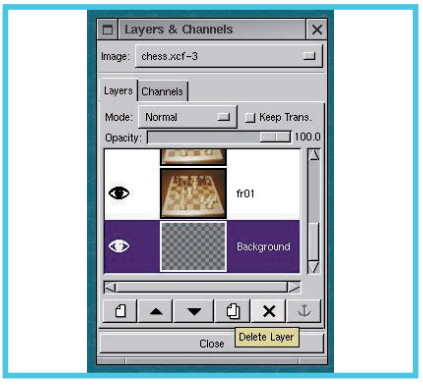
23. 画像ファイルの保存

すべての画像ソースがレイヤにペーストできたら、新規画像ウィンドウのポップアップメニューから［File］（ファイル）―［Save］（保存）を選択して、Gimpの標準フォーマット（XCFフォーマット）で保存しておく。



24. 拡張子指定による保存

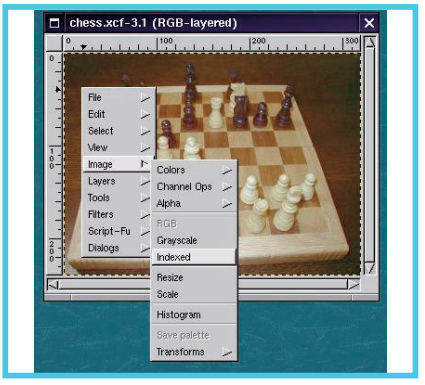
XCFフォーマットで保存するには、［Save Options］のプルダウンメニューから選択するか、［By extension］を選んでから、ファイル名に.xcfという拡張子をつけて保存すれば、自動的にXCF形式で保存してくれる。



25. Backgroundレイヤの削除

画像ソースを動画フレームとしてレイヤにしてしまえば、あとはGIF形式で保存するだけである。

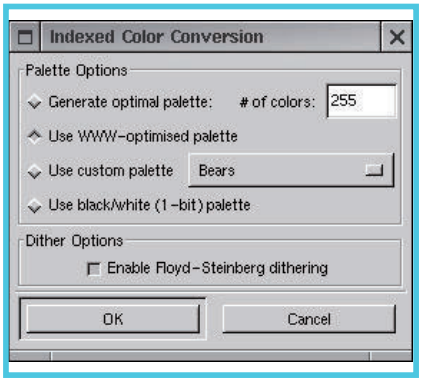
［Layers & Channels］ダイアログボックスのレイヤ表示の最下位にあるBackground（背景）レイヤを選択してから、コマンドボタン行の右から2番目にある［Delete Layer］（レイヤー削除）ボタンをクリックして削除する。



26. インデックス形式への変換

GIF形式で保存するときは、画像を最大256色のパレットを参照するインデックス形式にしておかなければならない。

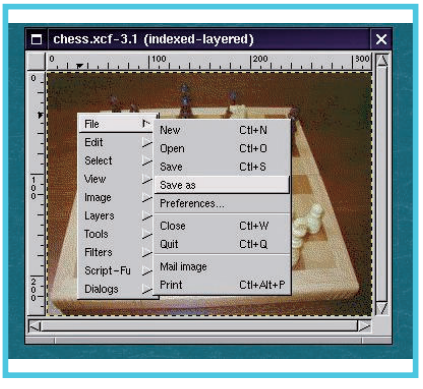
新規画像ウィンドウのポップアップメニューから［Image］（画像）―［Indexed］（インデックス画像）を選択して、インデックス形式へ変換する。



27. パレット種類の選択

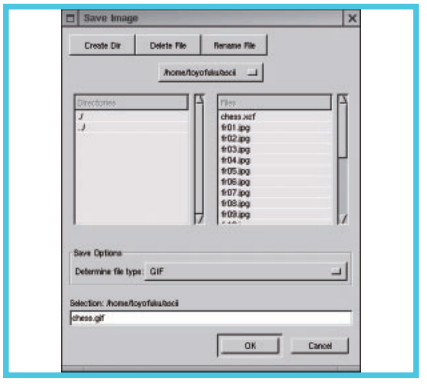
［Indexed Color Conversion］ダイアログボックスが表示されたら、使用するパレットの種類を選択する。

ここでは［WWW-optimised palette］（WWW用パレットを使用）を選択して、Webページの使用に最適な色数（ウェブセーフカラー）に対応させておく。



28. 画像データの新規保存

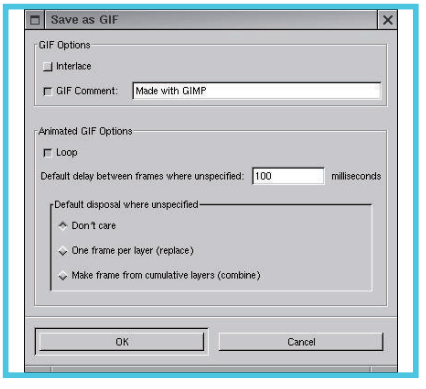
新規画像ウィンドウのポップアップメニューから［File］（ファイル）―［Save as］（新規保存）を選択する。



29. GIFフォーマットでの保存

［Save Image］ダイアログボックスが表示されたら、［Save Options］プルダウンメニューから「GIF」に選択して、ファイル名を入力する。

もちろん、前述したように、プルダウンメニューで［By extension］にしてから、.gifという拡張子をつけてファイル名を指定してもよい。



30. オプションの設定

［Save as GIF］ダイアログボックスでは、インターレース設定のほか、GIFアニメーションのオプション設定ができる。［Loop］チェックボックスをオンにしておくと、全体がループする設定になる。また、ミリ秒単位でフレーム表示の間隔を設定できる。

これでGIFアニメーションの完成である。Netscape Navigatorなどで出来上がりをチェックしてみよう。

Script-Fuによる自動変換

Gimpには、Script-Fuと呼ばれる強力なマクロ機能がある。このScript-Fuは、SchemeというLispベースのインタプリタ言語を利用しており、見た目はとっつきにくいかもしれないが、一度マスターしてしまえばGimpを自由自在に扱うことができる。Gimpを使う最大の利点は、さまざまな画像処理をScript-Fuでスクリプトとして記述できる点にあるといえる。

たとえば、作成手順で行った色相／彩度の変更と縮小変換の処理などは、Script-Fuを利用すれば、値の入力を省略できるだけでなく、バッチ処理としてシェルスクリプトから実行することもできる（ただし、Xは起動していなければならない）。

Script-Fuを使いこなすためには、利用可能な関数を一覧できるDB Browserの使い方をまず覚えてほしい。DB Browserは、ツールボックスの［Xtns］（拡張）メニューから［DB Browser］（手続きデータベース一覧）コマンドを選択すれば起動する。ここで例として、色相／彩度の変更に対応する正確な関数名を探してみよう。DB Browserの［Search］フィールドに”hue”と入力して、［Search by names］ボタンをクリックすると、関数名がgimp-hue-saturationであり、6つの引数をとることがわかる（**画面3**）。

では、さっそくScript-Fuを体験してみよう。なお、ここで紹介するScript-Fuスクリプトは、付属CD-ROMのLinuxmag/multimedia-tools/Graphic/script-fu_sampleディレクトリに収録してあるので利用してほしい。

リスト1は、入力と出力の画像ファイル名をインタラクティブに受けつけるスクリプトである（説明の便宜上、行番号を入れてある）。

この**リスト1**の12行目と13行目の先頭のコメント“;”を削除したものを、“sample01.scm”というファイル名で、ホームディレクトリの.gimp/scripts

▼リスト1　画像変換スクリプト「sample01.scm」

```
1   (define (sample01 inFile outFile)
2   ; JPEGファイルをロード
3     (set! img (car (gimp-file-load 0 inFile inFile)))
4     (set! drawable (car (gimp-image-active-drawable img)))
5   ; アンドゥ機能をオフにする
6     (gimp-image-disable-undo img)
7   ; 色相・彩度を変換
8     (gimp-hue-saturation img drawable 0 10.0 20.0 20.0)
9   ; 画像を縮小する
10    (gimp-image-scale img 320 240)
11  ; ウィンドウで確認するときは、以下の２行のコメントを外す
12  ;  (gimp-display-new img)
13  ;  (gimp-displays-flush)
14  ; JPEGファイルを保存
15    (file-jpeg-save 1 img drawable outFile outFile 0.75 0.0 0)
16  ; アンドゥ機能をオンにする
17    (gimp-image-enable-undo img)
18  )
19
20  ; Script-Fuの登録書式
21  (script-fu-register "sample01"
22                      "<Toolbox>/Xtns/Script-Fu/myProj/sample01"
23                      "sample01"
24                      "Toyofuku Tsuyoshi"
25                      "Toyofuku Tsuyoshi"
26                      "2000"
27                      ""
28                      SF-VALUE "Infile"  "\"ascii/p2100001.jpg\""
29                      SF-VALUE "Outfile" "\"ascii/frxx.jpg\""
30  )
```

“;”で始まる行はコメントとみなされ、実行には影響を与えない

gimp-file-load関数とgimp-image-active-drawable関数を実行し、これらの関数が返すリストの先頭の要素を、それぞれ変数imgと変数drawableにバインドする。[File]－[Open]という操作に対応

メモリキャッシュの使用量を節約するためアンドゥ機能をオフ

色相／彩度の変換と画像の縮小に対応

sample01関数の定義領域

変換後の画像をJPEGファイルとして保存

sample01の引数、inFileとoutFileのデフォルト値を設定

[Script-Fu]メニューに[sample01]を登録

ディレクトリに保存する。

保存が終わったら、ツールボックスから［Xtns］（拡張）—［Script-Fu］（スクリプトFu）—［Refresh］（再初期化）を選択して、［Script-Fu］メニューを最新状態に更新する（**画面4**）。

そして、［Xtns］（拡張）—［Script-Fu］（スクリプトFu）—［myProj］—［sample01］を選択して（**画面5**）、作成したスクリプトを実行すると、［sample01］ダイアログボックスが表示されるはずだ（**画面6**）。このダイアログの［Infile］フィールドと［Outfile］フィールドのファイル名をそれぞれ適当な名前に変更して、［OK］（了解）ボタンをクリックする。［Current Comannd］フィールドに、現在処理されているコマンドが表示される。

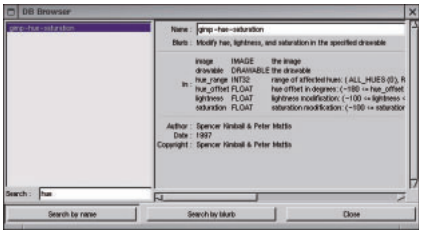
問題なく正常に処理されたら、最後に変換後の画像ウィンドウが表示される。エラーが発生した場合は、エラーメッセージをもとに、関数名や引数の指定に誤りがないか確認し、誤りがあったら修正する。正常に動作したら、sample01.scmの12行目と13行目のコメントを元に戻しておく。

また、**リスト2**はsample01を連続実行するバッチ処理のスクリプト（batch01.scm）である。

このbatch01.scmをsample01.scmと同じディレクトリ（.gimp/scriptsディレクトリ）に置き、再び［Script-Fu］メニューを最新状態に更新し、［myProj］メニューに［batch01］を追加する。ここでGimpを終了させ（［File］—［Quit］）、シェルプロンプトで以下のコマンドラインを入力する。

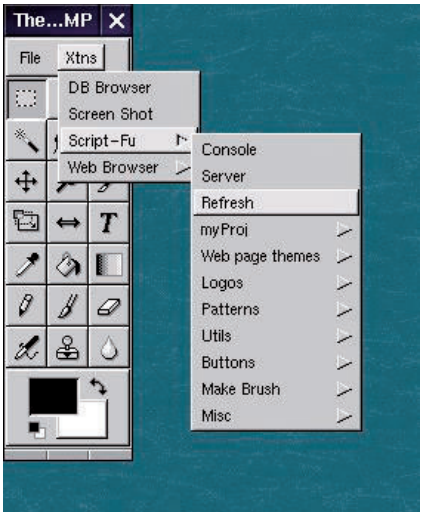
```
$ gimp -n -b '(batch01 1)'  '(gimp-quit 0)'
```

このように、gimpのコマンドラインオプションでバッチ実行を指定すると、Script-Fuに登録されたバッチスクリプトが、コマンドラインから実行できるのである。

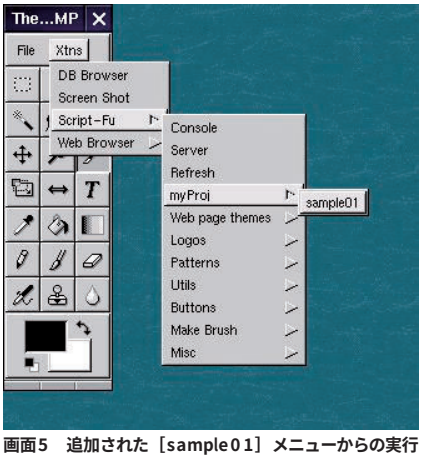


画面3　DB Browser

［DB Browser］の左部にあるリストボックスに、Gimpの関数名が一覧表示されている。リストボックスで選択された関数に対応して、右側のペインにその関数に対する簡単な説明や、引数の型と意味が表示される。



画面4　［Script-Fu］メニューの初期化



画面5　追加された［sample01］メニューからの実行

画面6　［sample01］ダイアログボックス

▼リスト2　sample01.scmを連続実行するスクリプト「batch01.scm」

```
1       (define (batch01)
2         (sample01 "ascii/p2100001.jpg" "ascii/fr01.jpg")
3         (sample01 "ascii/p2100002.jpg" "ascii/fr02.jpg")
4         (sample01 "ascii/p2100003.jpg" "ascii/fr03.jpg")
5         (sample01 "ascii/p2100004.jpg" "ascii/fr04.jpg")
6         (sample01 "ascii/p2100005.jpg" "ascii/fr05.jpg")
7         (sample01 "ascii/p2100006.jpg" "ascii/fr06.jpg")
8         (sample01 "ascii/p2100007.jpg" "ascii/fr07.jpg")
9         (sample01 "ascii/p2100008.jpg" "ascii/fr08.jpg")
10        (sample01 "ascii/p2100009.jpg" "ascii/fr09.jpg")
11        (sample01 "ascii/p2100010.jpg" "ascii/fr10.jpg")
12      )
13      (script-fu-register "batch01"
14                          "<Toolbox>/Xtns/Script-Fu/myProj/batch01"
15                          "batch01"
16                          "TOYOFUKU Tsuyoshi"
17                          "TOYOFUKU Tsuyoshi"
18                          "2000"
19                          ""
20      )
```

Sample01.scmが受け取る引数を指定

Section 3

ビデオ

Sound Graphic Video

最後に今後の注目株、ビデオの取り込みやインターネット中継の方法を紹介します。

PicPomでTVを見て、撮る

文：住吉龍一　*Text : Ryuichi Sumiyoshi*

意外というか何というか、Linuxで動作するTVチューナー&ビデオキャプチャカードは結構ある（1月号の94ページでも2機種紹介している）。基本的に、Conexant（旧Brooktree）製のコントローラチップである、Bt848やBt878を採用している製品なら動かすのはそれほど難しくないようだ。

今回は、2000年1月から出荷開始された、インタウェア製のTVチューナー&ビデオキャプチャカード「PicPom Linux」（**写真1、2**）を使って、TVを見たり、ビデオキャプチャをする方法について解説しよう。

PicPom LINUXって？

「PicPom LINUX」（以下、PicPom）は、コントローラチップとしてBt878（**写真3**）を採用したLinux専用のビデオキャプチャカードで、FM／TVチューナーも内蔵している。対応しているディストリビューションも11種類と豊富で、それぞれに対応したインストーラがバンドルされている。

また、製品のCD-ROMには、デバイスドライバ、各種アプリケーションの全ソース、主要APIの解説、Linux標準の動画形式「Video for Linux」（V4L；Video4Linux）の解説の和訳も付属しており、ユーザーによるアプリケーションの研究や開発が行いやすいようになっている。

さらに、インタウェアでは動画データをMPEGデータにエンコードするツール「MPEG Tool」評価版の公開も予定している。このツールは、PicPomで録画した動画データをMPEGデータとしてエンコードするもので、PicPomのWebサイト（http://www.interware.co.jp/products/PicPomLinux.html）からダウンロードできるようになる予定だ（開始日は未定）。このツールを利用すれば、FMラジオを聞く、テレビを見る、動画を録画する、PicPomを複数枚挿ししてテレビ、ビデオを複数同時に見る、というPicPomの基本機能に加えて、PicPomで録画した動画データをMPEGデータに変換して自分のWebページに置くといった、より広い用途にPicPomを利用することができるようになるだろう。

Video for Linuxって？

Video for Linux（V4L）は、Linuxでビデオ（ラジオ、文字放送も含む）を扱うためのAPIである（http://roadrunner.swansea.uk.linux.org/v4l.shtml）。Linux kernel 2.0のころはビデオ周りに関するAPIが統一されておらず、各

写真1　FM/TV対応ビデオキャプチャカード「PicPom LINUX」
開発：インタウェア（http://www.interware.co.jp/）
価格：オープンプライス

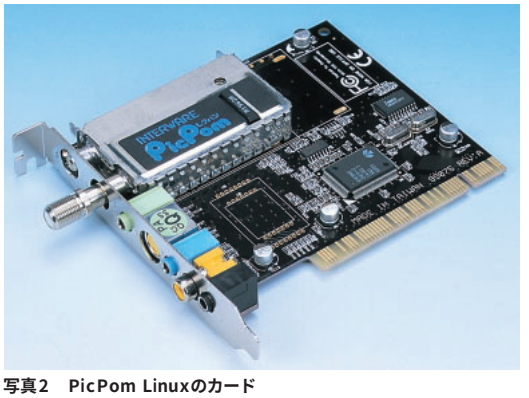


写真2　PicPom Linuxのカード
TV／FM出力端子のほかS-VHS入力端子、サウンドの入出力がある。音声出力は、サウンドカードへ接続する。

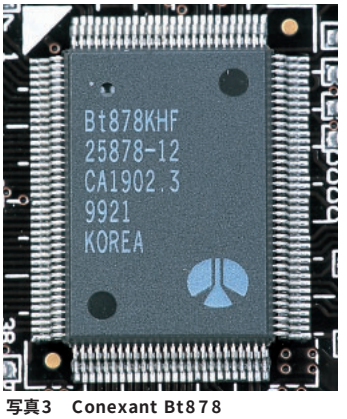


写真3　Conexant Bt878
Bt84xシリーズの改良版となるコントローラチップ。

チップごとにバラバラにデバイスドライバが開発されていた。そこで、Linux 2.1.1xxのころにAlan Cox氏が中心となって、ビデオ周りのAPIを統一するべく開発されたのがこのV4Lである。V4Lがサポートしているコントローラチップには、

- **Bt848／848a／849／878／879系（PCIバス用）**
- **Zoran**

などがある。PicPomが利用しているアプリケーション、cRadio、XawTVは、このV4Lに準拠して作成されたものである。現在は、このV4Lに拡張性と、より多くのデバイスサポートを持たせたVideo for Linux2（V4L2；Video4 Linux2）が開発されており、ドライバなどもリリースされている（http://millennium.diads.com/bdirks/v4l2.htm）。

TVを見る

Linuxも使いたいが、やっぱり俺はドラマの続きが気になる、という気が気でない状態でLinuxを使っているユーザーはいないだろうか？　せっかくTVと似たような装置と向き合っているのだから、TV放送もLinux上から楽しんでしまおう。

■TVチューナー

PicPomは、TVチューナーを内蔵しており、VHF1～12、UHF13～62チャンネルに対応している。

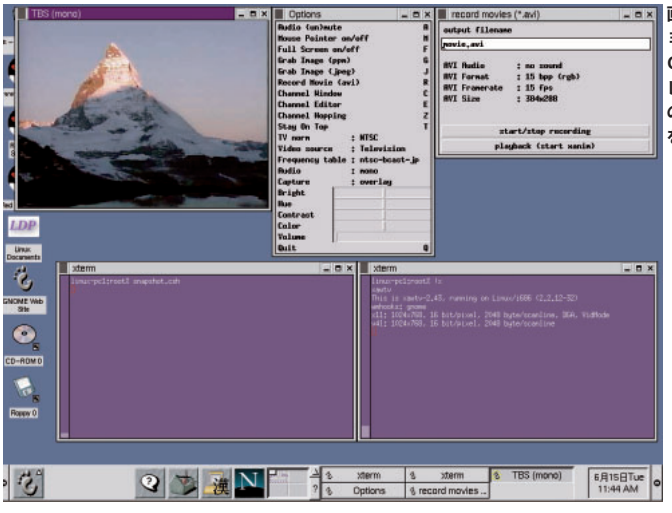
グラフィックスカードの機能に対応して、テレビやビデオテープレコーダなどの入力時にも、滑らかなオーバーレイ表示が可能である（ビデオ表示サイズは、最大640×480ピクセル）。

オーバーレイ表示は、キャプチャカード自身がバスマスタになり、直接グラフィックスカードのフレームバッファに画像データを送り込む機能で、データがPCIバスの上を一度しか流れないうえ、CPUの負荷を軽減することができるので、最新のマシンでなくても十分TV放送を楽しめるはずである。

この機能を利用するには、デバイスドライバにフレームバッファのベースアドレス、サイズ、デプスなどを適切に伝える必要がある。このあたりの設定は、xawtv付属のツール、v4l-confから行える。

■xawtv

PicPomでは、TVを見るアプリケーションとしてxawtvが用意されている（**画面1**）。このxawtvは、Gerd Knorr氏によって作成され、GPLに則って公開されているXアプリケーションである。PicPomにバンドルされているxawtvは、メニューやボタンなどが日

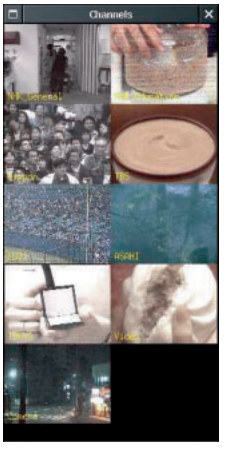


画面1　TV表示アプリケーション「xawtv」
CPUに負荷をかけないオーバレイ表示により、比較的古めのマシンでも快適にTV放送を楽しめる。

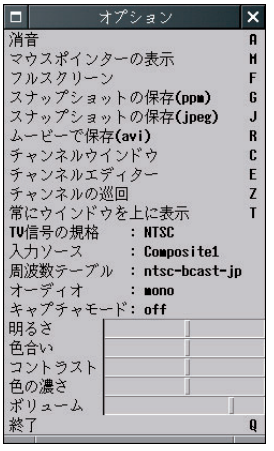
▼リスト1　PicPomの.xawtv設定

```
norm = NTSC
capture = on
source = Television

freqtab = ntsc-bcast-jp
pixsize = 128 x 96
jpeg-quality = 75
fullscreen = 640x480
lauch =
sound,x,/usr/X11R6/bin/xmixer
mixer = vol

[NHK_General]
channel = 1
key = 1

[NHK_Education]
channel = 3
key = 3

[Nippon]
channel = 4
key = 4

[TBS]
channel = 6
key = 6

[FUJI]
channel = 8
key = 8

[ASAHI]
channel = 10
key = 0

[TOKYO]
channel = 12
key = 2

[Camera]
source = Composite1
key = k

[S-VIDEO]
source = S-Video
key = s
```

画面2　チャンネルエディター



画面3　チャンネルウィンドウ



画面4　オプションウィンドウ

画面5　動画のキャプチャを行う「AVI保存ウィンドウ」

本語化されているうえ、標準で東京地区のチャンネルが設定されている。

xawtvは、各ユーザーのホームディレクトリにある$HOME/.xawtvを変更することで、さまざまなカスタマイズを行うことができる。なお、このファイルは起動時に参照されるので、必ず作成しておかなければならない。

・チャンネルエディター

xawtvの設定ファイル、$HOME/.xawtvを編集するGUIツールとしてチャンネルエディターが用意されている（**画面2**）。自動でTV局を探す機能が用意されているが、電波の状況によっては自動では検出できない場合もあるので、手動で周波数を変えてTV局を探すこともできるようになっている。

ただし、チャンネルエディターはあくまでも簡易設定ツールで、さらに細かいカスタマイズをするためには、$HOME/.xawtvをエディタで直接編集しなければならない。

・メインウィンドウ

$HOME/.xawtv設定に問題がないようなら、xawtvを起動すれば、**画面1**にあるようなTV画像を表示するメインウィンドウが起動するはずだ。

xawtvでは、マウスやキーボード上のキーに、チャンネル変更からチューニングの微調整、明るさの調節など、さまざまな機能が割り当てられており、よく利用する機能は、キーマップを覚えてしまえばかなり快適に利用することができる。

・チャンネルウィンドウ

チャンネルウィンドウは登録されているすべてのチャンネルを一覧するためのウインドウである（**画面3**）。サムネイルのような画面をクリックすると、そのチャンネルに切り替わる。これらの画面は常に更新されているわけではなく、Ctrl＋Zを押すことによって、全局のイメージが更新されるようになっている。負荷をかけないという意味でも、これは納得できる仕様といえるだろう。なお、画面サイズは、設定ファイル中のpixsizeエントリで指定することができる。

・オプションウィンドウ

オプションウィンドウは、xawtvの各種設定を行うためのウインドウである（**画面4**）。メニューの右端の文字はキーボードショートカットの文字を意味している。なお、明るさ、色合い、コントラスト、色の濃さについては、チャンネルごとに設定値が記憶される。

・xawtv-remote

リモートホスト上で動作しているxawtvをコマンドライン上から制御するためのコマンドで、一種のリモコンと呼べるものである。チャンネルはもとより、明るさやコントラストなどもすべてコマンドラインのオプションで変更可能であり、リモートメンテナンスが可能である。

TV／ビデオ画像の保存

PicPomでは、キャプチャカードの名が示すとおり、内蔵TVチューナから見ているTV画像や、RCAピンジャック（コンポジット）やS端子に接続した外部のAV機器から、静止画、動画として取り込むことができる。

■静止画の保存

PicPomでは、見ている画像を静止画として取り込むことができる。保存データは、JPEG形式とPNG形式の2種類があり、メニューやショートカットキーからどちらの画像フォーマットでも選択することができる。

もちろん、保存した静止画は、Gimpなどの画像処理ソフトウェアから参照することができるので、さまざまな加工を施すことも可能である。

■動画の保存

xawtv上でRキーを押すか、オプシ

ョンウィンドウからAVI保存ウィンドウを起動すれば、動画も保存することができる（**画面5**）。保存可能なデータ形式は、AVI形式である。キャプチャできる画面のサイズはTV画面のウインドウサイズに連動しており、アプリケーションを起動したディレクトリに保存される。なお、取り込み中は、画面の更新が停止する。

保存したAVIファイルを再生するには、xanimなどの再生用のアプリケーションを利用すればよい。

なお、320×240ピクセルで録画すると、1秒当たり7Mバイト弱のデータサイズとなるため、たとえば2Gバイトのハードディスクなら約5分程度の動画が保存できる。

なお、このxawtvのキャプチャ機能は、内部でstreamerというコマンドを呼び出すことで実現されている。このstreamerは、コマンドライン上から単独のコマンドとして利用して、静止画、動画のいずれもキャプチャさせることも可能で、cronなどと組み合わせて、バッチ実行をさせる際には有効に活用することができるだろう。

PicPomの複数使用の例

PicPomは、接続するマシンのPCIスロットやデバイスのIRQの競合さえなければ、複数枚での使用が可能である。インストール時に、/devディレクトリ下に“video?”（?には任意の数字が入る）というキャラクタがデバイスが作成されているので、xawtvの実行時に-cオプションをつけて、次のように実行すればよい。

```
$ xawtv -c /dev/video0 &
$ xawtv -c /dev/video1 &
$ xawtv -c /dev/video2 &
```

また、PicPomではFM放送とTV放送の同時受信ができないので、複数枚利用することで、これらの同時使用も可能となる。

Webを使った画像配信

PicPomは、自分で楽しむだけでなく、Webを利用した画像配信も行うことができる。これを利用すると、カメラを設置しておいた場所の映像を定期的に参照するといった使い方もできる。これは、たとえば防犯カメラをイメージしてもらうとわかりやすいかもしれない。そして、その映像を受信するクライアントがTVではなく、Webブラウザになるというわけだ。

PicPomには、クライアントサイドのECMAScript（旧JavaScript）を利用して、リアルタイムキャプチャを行うCGIスクリプトも用意されているので、工夫次第でいろいろな用途に利用できるはずだ。

Column

FMラジオも聞けるPicPom

何か作業をしながら、音楽を聴くという人は多いはずだ。TV放送では、人によっては気が散ってしまう人がいるかもしれないが、FM放送から流れてくる音楽ならさほど邪魔にもならないかもしれない。そんなあなたには、PicPomでラジオ放送を聴くことをお勧めしよう。

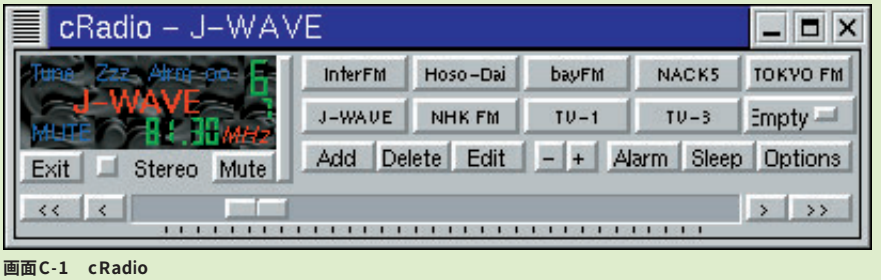
■cRadio

PicPomでは、ラジオ放送（FM放送のみ）のプレーヤとして「cRadio」が用意されている（**画面c-1**）。このcRadioは、GPLに則って公開されているXアプリケーションで、GUIツールキットとしてQtを利用している（http://www.leg.uct.ac.za/cRadio/）。アプリケーションパネルの各チャンネルボタンは、/etc/cRadio.confで設定された周波数帯域で受信可能なFM局を表示している。また、パネル下にあるスライダーは50kHzずつの変更（設定）ができる。

cRadioが対応している周波数は70～108MHzで、設定ファイルである/etc/cRadio.confを修正することで、好みの放送局をプリセットしたり、さまざまなカスタマイズを行うことができる。PicPomに収録されているcRadio.confには、初めから東京地区のFM局が設定されている。なお、このcRadio.confは、（デフォルトでは）書き込み権限のあるスーパーユーザーだけが変更することができる。

また、テキストファイルであるcRadio.confの、チューニングの周波数を追加、削除、編集するGUIインターフェイスや、各種の設定をカスタマイズするためのGUIインターフェイスも用意されている。

ただし、cRadioは/dev/audio（サウンド）デバイスを占有するので、xawtvなどをバックグラウンド（もしくは、ほかのユーザー）が使用しているときは利用できない。



画面C-1　cRadio

RealSystemによるインターネット中継入門

文：たなかとしひさ　Text : Toshihisa Tanaka

ここではこのセクションのまとめとして、Linuxを使ってインターネット中継をする方法について解説する。具体的には、インターネット中継を実現するシステムとして、RealSystemを取り上げ、簡単にRealSystemソフトウェアの紹介しよう。

インターネット中継

インターネット中継とは、文字どおり「インターネットの回線を通じて、映像／音声をリアルタイムに配信するシステム」のことである。インターネットの爆発的な普及と、高速な回線の利用により、まさにオンデマンドで、今起こっている出来事を中継することが可能となったのだ。

RealSystem

RealSystemとは、RealNetworks（http://www.real.com/）から供給されているソフトウェア一式のことである。RealSystemのソフトウェアは、大別すると次の3種類に分けられる。

■Realエンコーダ

Realエンコーダは、映像／音声データを入力し（キャプチャといいます）、RealVideo形式にデータ変換を行い（エンコード）、Realサーバに送信する。

■Realサーバ

Realサーバは、Realエンコーダから送られてきたデータを蓄積し、後述するRealプレーヤの要求に応じ、Real VideoデータをRealプレーヤに送信する。

■Realプレーヤ

Realプレーヤは、Realサーバから送られてくるデータを再生する。

このように、RealSystemでは各機能がソフトウェア的に分担されている。したがって、Realエンコーダが、直接Realプレーヤと通信することはないし、Realサーバが、直接キャプチャデータをエンコードすることもない。つまり、上にあげた3つのソフトウェアが、それぞれ連係し合うことによって、初めてRealSystemによる中継が実現できるというわけである。

それぞれのソフトウェアの働きと関係のイメージを**図1**に示す。

RealSystemが動作するディストリビューション

RealSystemのソフトウェア（Realエンコーダ／Realサーバ／Realプレーヤ）は、（基本的に）すべてLinux上でも動作する。したがって、「すべてLinuxを使って」RealSystemを構築することは可能なのだが、ディストリビューションの差異により、そのままでは動作しない場合もある。

主なディストリビューションにおけるRealSystemへの対応は、筆者らが調べた限りでは、**表1**のとおりである。なお、この表中にあるRealServer G2のバージョンというのは、動作を確認できたRealServer G2のバージョンのことで、最新のRealServer G2のバージョンは7.0である。

Realエンコーダ

Realエンコーダは、ビデオからの映像／音声データを取り込み、変換し、Realサーバに配信する。

Linux上で動くRealエンコーダは、次のとおり、

- **RealProducer G2 for Linux（無料）**
- **RealProducer "Plus" G2 for Linux（有料：2万9800円）**

2種類ある。

基本的に、RealSystemのソフトウェアは、名称に“ Plus ”がつくのが有料版で、ついてないものが無料版となっている。

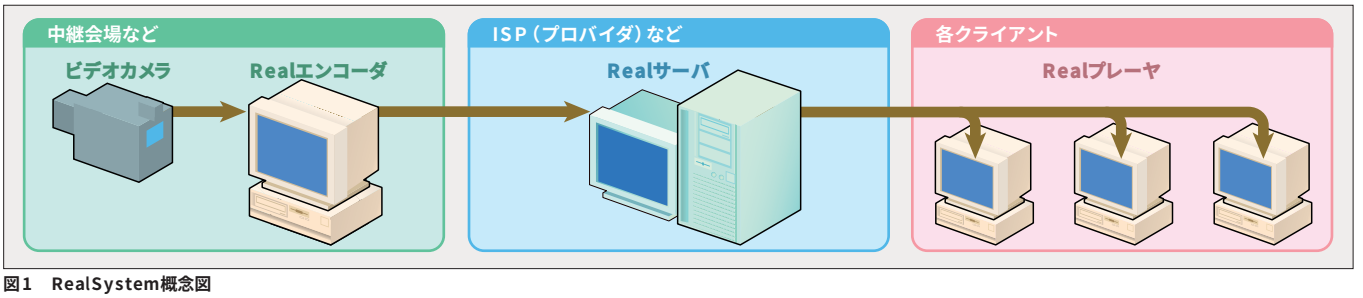


図1　RealSystem概念図

■2つのRealエンコーダ

RealProducer "Plus" G2と、RealProducer G2の主な違いは、RealPlayer 5.0形式のエンコードができる／できないというのが挙げられる。クライアントOSがWindowsの場合、現在はRealPlayer G2が標準となっているため、無料版であるRealProducer G2を利用しても特に問題はない。

しかし、Linux版のRealPlayer G2はまだβ版しかないため、RealPlayer 5.0がまだまだ主流である。そのため、Linuxをクライアント OSとして配信する場合は、RealProducer "Plus" G2を購入し、RealPlayer 5.0互換でエンコードさせるほうが問題は少ないだろう。また、RealProducer "Plus" G2は、コマンドラインベースで動作させることも可能であり、このあたりもLinuxとの親和性は高いといえる。

■ビデオキャプチャカード

さらに、RealProducer G2を動作させる場合、Video4Linuxに対応したビデオキャプチャカードが必要となる。これは、Bt848ベースのチップを搭載したビデオキャプチャカードなら、ほとんど問題なく動作するはずなので、Bt848ベースのチップを搭載したビデオキャプチャカードを購入しておくことをお勧めする。

なお、筆者らは、RealNetworks推奨ということもあり、Osprey 100という、Bt848ベースのビデオキャプチャカードを利用している。Bt848ベースのLinux用ドライバに関する情報は、http://www.thp.uni-koeln.de/~rjkm/linux/bttv.htmlから入手することができる。

Bt848ベースのビデオキャプチャカードは、Linux kernel 2.0.xカーネルでも動作するので、必ずしもカーネルを2.2.xにアップグレードしなければならないわけではない。

ただし、kernel 2.0.xにはVideo4Linuxは標準で含まれていないので、別途Video4Linuxドライバを入手する必要はある。kernel 2.2.xなら、Video4Linuxが標準でバンドルされているので、Video4Linuxをダウンロードする必要はない。もちろん、Linuxカーネルを新たにダウンロードしたときは、Video4Linuxを有効にしてカーネル再構築を行う必要がある。

■ノートPCでのエンコード

ノートPCでのエンコードについては、九州大学の岡村氏が、IBM Smart Capture CardのVideo4Linux対応のドライバを作成している。この場合、Linux kernel 2.2.xが必須となるが、ノートPCを持っていて、IBM Smart Capture Cardを持っている人なら、ノートPCでもRealVideoエンコードが可能である。

岡村氏作のIBM Smart Capture Card Video4Linux対応ドライバは、ftp://sourceforge.org/pcmcia/contrib/より入手可能である。

■インストールの実際

Plamo Linux 1.4でRealProducer G2を動かす場合は、CD-ROMやFTPサイトにある、contrib/Glibc2/glibc2.tgzをインストールする必要がある。glibc2.tgzのインストールは、rootでログインし、

```
# /sbin/installpkg glibc2.tgz
```

とすることでインストールできる。

RealProducer G2は、http://www.jp.real.com/products/tools/producer/より入手可能で、ここのページから［RealProducer "Plus" G2］か［RealProducer G2］を選択し、メールアドレスと質問事項を選択すればダウンロードページに移るので、そこからダウンロードすることができる。

ただし、RealProducer G2はglibc2.0ベースで開発されているため、glibc2.1ベースで開発されているRed Hat Linux 6.1やSlackware 7でReal Producer G2を利用するには、パッチを当てなくてはならない。これらのユーザーは、RealProducerのダウンロードと同時に、このパッチファイルもダウンロードしておこう。

RealProducer G2のインストールは、次の手順で行う。

```
# zcat rprodplusg2_linux.tgz | tar
-xvf -
# ./install
```

デフォルトでは、/usr/local/rprodディレクトリ以下に、RealProducerがインストールされる。また、インストールの途中にある、ライセンスキー番号を入力するフェーズでは、Free版をダウンロードしたなら“free”と入力し、有料版のRealProducerをダウンロ

	エンコーダ RealProducer G2 (有料／無料)	サーバ BasicServer G2 (有料／無料)	プレーヤ RealPlayer 5.x	プレーヤ RealPlayer G2
Slackware 7.0	△	○ (Ver 7)	△	○
RedHat 5.2	○	○ (Ver 6)	○	△
RedHat 6.1	△	○ (Ver 7)	△	○
Vine Linux 1.0	○	○ (Ver 6)	○	△
Plamo Linux 1.4.x	△	○ (Ver 6)	○	×
Plamo Linux 2.0β	△	○ (Ver 7)	×	○
Debian GNU/Linux	△	○ (Ver 6)	○	○

表1　各ディストリビューションのRealSystemへの対応　○：そのままで使用可能　△：条件つきで動作　×：動作せず

ードしたなら、別途メールで送られるライセンスキー番号を入力する。

Red Hat Linux 6.1やSlackware 7では、インストール完了のあと、パッチを当てる必要がある。/usr/local/rprodディレクトリ以下にRealProducer G2をインストールした場合なら、パッチは、

```
# cd /usr/local/rprod
# zcat rprodg2_patch.tgz | tar -xvf -
# cd bin
# ./patchrprod
Please enter the directory where the
realproducer is installed.
[/usr/local/rprod] <- リターンキーを押す
```

という手順でパッチを当てることができる。

インストールが完了すれば、

```
# xrealproducer
```

と入力すれば、RealProducerが起動する（**画面1**）。

■Realエンコーダの転送方式

RealSystem G2では、“SureStream”という転送方式が導入されている。前バージョンとなるRealSystem 5.0では、1エンコードデータ（ストリーム）の転送bpsが固定されていたが、SureStreamでは、1エンコードデータ中に複数の転送bpsを含めることが可能である。すなわち、1つのRealProducer G2で、20kbpsやそれ以外のbpsのエンコードデータをReal サーバに転送することができるのである。

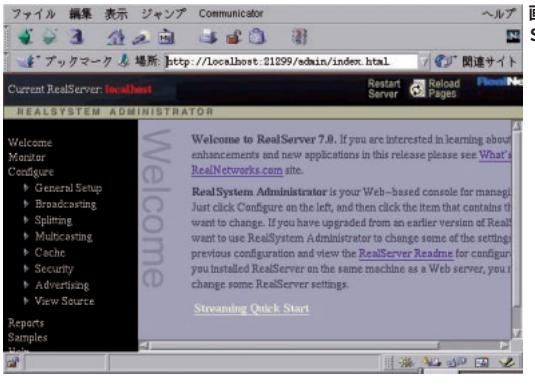
ここで、RealProducer G2を使って、エンコードデータをRealServer G2に送信する場合に注意すべきことがある。

RealSystem 5.0のときのデータ転送では、基本的にRealEncoder 5.0 → RealServer 5.0 への一方通行だったので、あいだにIPマスカレードが入っても特に問題なく通信できた。しかし、RealSystem G2では、クライアント（RealPlayer G2）とRealServer G2の接続回線速度から、エンコードbpsを変化させることができるようになったため、RealServer G2からRealProducer G2に対しても通信を行っている（セッションを張りに行く）。

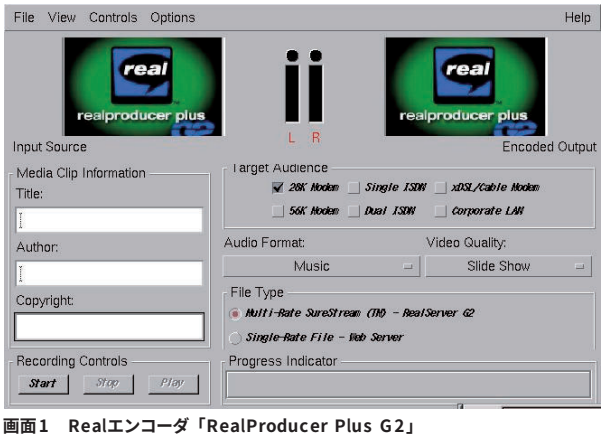
したがって、RealProducer G2を使ってRealServer G2と通信する場合、途中でIPマスカレードさせたりすることができない（1対1対応となるNATなら通信が可能である）。つまり、RealProducer G2は、RealServer G2からダイレクトに通信できるところに設置する必要がある。

RealProducer ⟷ RealServerの場合は、標準インストールを行った場合、TCP/IPの4040番ポートを使用するので、このポートさえ、相互に通信できる状態であれば、エンコードデータを送信することが可能である。

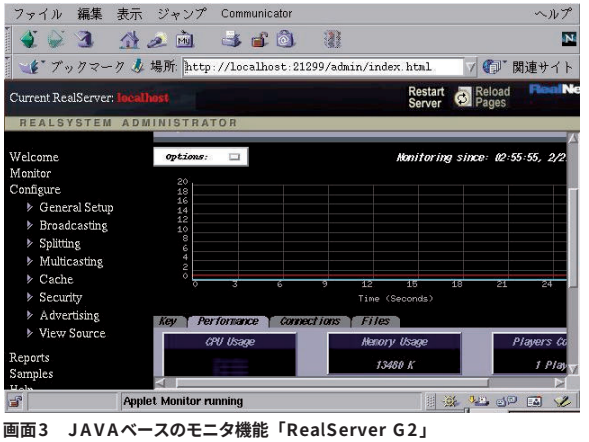
なお、RealProducer G2は、RealServer 5.0とは通信することができないので、RealProducer G2を使用する場合、転送先のサーバは、必ずRealServer G2でなければならない。



画面2　Webブラウザから管理可能な「RealServer G2」



画面1　Realエンコーダ「RealProducer Plus G2」



画面3　JAVAベースのモニタ機能「RealServer G2」

Realサーバ

Realサーバは、Realエンコーダからエンコードデータを受け取り、実際にクライアントであるRealプレーヤ側に配信する。Linuxで動作するRealサーバは、いくつかの種類が販売／配布されているが、パーソナルで使用するなら、

- **BasicServer G2 （無料／25クライアントまで）**
- **BasicServer "Plus" G2 （有料：47万円／60クライアントまで）**

の2種類が適当だろう。ただし、無料版はあくまでも個人使用、あるいは評価向けに配布されており、企業や資金のある団体は有料版を購入する必要があるので、注意が必要である。

■BasicServer G2のインストール

BasicServer G2は、http://www.jp.real.com/solutions/basic/plus.htmlより入手可能である。ここでも、名前やe-mailといった質問事項に答えることで、ダウンロードページに移動し、ダウンロードすることができるようなる。

BasicServer G2のインストール方法は、以下のとおりである。

```
# chmod +x g2_7_0gold-linux-c6.bin
# ./g2_7_0gold-linux-c6.bin
```

このインストールの途中で、インストール先のディレクトリを質問してくるが、RealProducer G2のときと異なり、ディレクトリの標準設定がない状態で聞いてくるので、適宜入力が必要である。また、これもインストールの途中で、Web経由でのメンテナンスのためのポート（Adminポート）も質問してくる。これは、ランダム値が設定されるため、BasicServer G2 をインストールするたびに、値は異なるので注意してほしい。

筆者が試した環境では、次のようになった。

- **PNAポート：　7070番**
- **RTSPポート：　554番**
- **HTTPポート：　8080番**
- **Adminポート：　21299番**

ただし、これらの値はBasicServer G2をインストールしたあとも変更可能である。インストールが完了したら、別途メールで送られてくるライセンスキーファイルを、BasicServer G2をインストールしたディレクトリのLicenseディレクトリ以下にコピーする必要がある。

ライセンスファイルをコピーしたら、BasicServer G2をインストールしたディレクトリに移り、

```
# ./Bin/rmserver rmserver.cfg &
```

と入力すれば、BasicServer G2が起動する。

BasicServer G2の起動が失敗する原因として、IPアドレスのバインドに失敗するケースがよく挙げられる。もしも、BasicServerG2の起動に失敗するようなら、hostnameコマンドで得られるホスト名のIPアドレスが利用可能であるかどうか確認し、再度起動を行ってみてほしい。

■BasicServer G2の負荷

BasicServer G2自身は、あまりCPUリソースを必要としないため、マシンにあまり負荷はかけない。

たとえば、20kbpsの品質で25クライアント同時接続程度なら、CPUはPentium 166MHzクラスのものでも十分運営できるようになっている。CPUよりもむしろ、Realサーバに求められるのは、

1. **速い回線（1Mbps程度の回線容量）**
2. **RealVideoコンテンツを格納できるだけのディスク容量**
3. **128Mバイト程度のメモリ（-mオプションで使用メモリ量を設定可能）**

である。

なお、どれくらいの回線容量が必要になるか、という目安については、おおよそ次の式で求めることができる（ただし、この式は筆者の経験則に基づくもので、正式なものではない）。

必要回線容量＝（エンコードボーレート×1.5）×同時接続クライアント数

たとえば、エンコードする（配信する）ボーレートを20kbpsとし、同時に25クライアント接続しているとすると、

（20k×1.5）×25＝750kbps

分の回線容量が必要（使用されている）になるということである。

Realサーバを構築する場合は、まず予想される接続クライアントを考慮し、それに基づいて上の式にあてはめて、必要回線容量を見積もるといいだろう。

■BasicServer G2の保守と運用

BasicServer G2は、Webブラウザから保守を行うことが可能である（**画面2**）。また、BasicServer G2は、JAVAベースのモニタリング機能も持っているので、接続状態の監視もWebブラウ

ザから行うことが可能である（**画面3**）。

ただし、WebブラウザからBasic Server G2を保守するには注意することがある。それは、BasicServer G2は、Webブラウザから接続があると、そのWebブラウザのバージョンチェックを行い、

- **Netscape Communicator 4.06より前**
- **Internet Explorer 4.0より前**

というバージョンのWebブラウザから接続された場合、Basic Server G2は接続を拒否するということである。したがって、WebブラウザからBasic Server G2を制御する場合、上記に挙げた以降のバージョンのWebブラウザで接続しなければならない。

Realプレーヤ

Realプレーヤは、Realサーバと接続し、実際に映像データを参照することができるアプリケーションである。Realプレーヤも、大別すると

- **RealPlayer 5.0**
- **RealPlayer G2**

2種類がある。これは、どちらも無料で配布されている。RealPlayer 5.0（**画面4**）は、ほぼすべてのディストリビューションで動作するが、RealPlayer G2の場合は、最新（1999年9月以降）のバージョンでは、glibc2.1.xベースのディストリビューションでないと、動作しないという制限がある。

また、前述のとおり、RealPlayer G2（**画面5**）のLinux版は現在β版しかない。ただ、β版とはいえ、Linux版のRealPlayer G2はかなり安定して動作しているので、RealPlayer G2が動作する環境を持っているなら、RealPlayer G2を導入しても大きな問題はないと思う。

■RealPlayer G2のインストール

Red Hat Linux 6.0なら、RealPlayer G2のインストールが可能なので、ここでは、RealPlayer G2のインストール方法について説明しよう。

1. http://www.real.com/products/player/linux.htmlに接続し、必要項目を入力する。なお、［Select platform］は、［RedHat 5.2/6.0 RPM］を選択する。必要項目の入力と選択が終了すると、［Download FREE RealPlayer］ボタンをクリックし、ダウンロードサイトに移る。

2. ダウンロードサイトに移動すると、ネットワーク的に一番近いと思われるサイトを選択し、ダウンロードを開始する。

3. ダウンロードが完了すると、

```
# rpm -Uvh G2player-6.0-0.99092901.
i386_rpm
```

として、rpm コマンドを使ってインストールを行う。なお、インストール時は、必ずroot権限で行わなければならない。

インストールが完了すると、あとはコマンドラインから、

```
# realplay
```

と入力するだけで、RealPlayerが起動する。

■RealPlayer 5.0のインストール

旧バージョンとなるRealPlayer（RealPlayer 5.0）は、http://proforma.real.com/real/player/blackjack.htmlよりダウンロード可能である。RealPlayer 5.0は、Linux kernel 2.0.x系のディストリビューションなら問題なく動作するが、kernel 2.2.x系のディストリビューションで動作させると、先頭部分だけ再生して、あとはエラーに

画面4　Realプレーヤ「RealPlayer 5.0」

画面5　最新のRealプレーヤ「RealPlayer G2」

なってしまうという問題がある。

その場合、http://onramp.i2k.com/~jeffd/rpopen/にある“rpopen.tar.gz”をダウンロードしてインストールすれば、この問題を回避することができる。このファイルは、/usr/local/bin/rvplayer5.0にRealPlayer 5.0がインストールされているとすると、

```
# zcat rpopen.tar.gz | tar -xvf -
# cd rpopen
# make
# cp open.so rplayer /usr/local/bin
/rvplayer5.0
```

という手順を実行すれば、以後/usr/local/bin/rvplayer5.0/rplayerで、RealPlayer 5.0を利用することができる。

Netscapeから Realプレーヤを起動させる 方法について

Windows上で動くWebブラウザなら、RealVideoのリンクを選択するだけでRealプレーヤが起動し、再生を始めさせることができる。Linux上でも、これと同様のことは可能である。

たとえば、Netscapeで、.rm（.ram）のリンクを選択したとき、Realプレーヤを起動させるなら、~/.mailcapに、

```
audio/x-pn-realaudio; realplay %s
```

の行を追加し、~/.mime.typesに、

```
type=audio/x-pn-realaudio
exts=rm,ra,ram
type=audio/x-pn-realaudio-plugin
exts=rpm
```

を追加すればよい。これで、.rm（.ram）ファイルを選択すると、Realプレーヤ（realplay）が自動的に起動し、ストリーム再生を開始するようになる。

また、RealVideoコンテンツを提供する側は、RealVideoコンテンツファイル（.rmファイル）を直接リンクするのではなく、テキストファイルで、

```
pnm://real.netfort.gr.jp/g2video.rm
```

と書かれたファイルを、“g2video.ram”などの名前で保存し、これをリンクするようにしておけば、

1. **Webブラウザ（Netscape）が、g2video.ramファイルを選択**
2. **g2video.ramファイルが選択されたので、NetscapeはRealプレーヤ（realplay）を起動**
3. **起動されたRealプレーヤ（realplay）は、g2video.ramファイルを開き、中に書いているpnm://real.netfort.gr.jp/g2video.rmを再生**

という手順にしたがって、マウスで選択するだけで、Netscapeからストリーム再生ができるようになる。

これからの展望

2月19日に、NLUG（Nagoya Linux Users Group）主催で、Linux勉強会（http://www.nagoyalinux.org/events/20000219/）が開催され、筆者もこの勉強会に参加させていただいた。

この勉強会でも、RealSystemを利用したインターネット中継が実施されたのだが、特筆すべきなのは、ビデオキャプチャと音声入力にUSBデバイスを利用して、Real中継が実施されたということである（**画面6**）。

NLUG主催のLinux勉強会で実施されたReal中継のエンコーダマシンには、

- **Debian GNU/Linux（potato）**
  ※Linux kernel 2.2.14に usb-2.3.44-for-2.2.14.diff.gzのパッチを当てたもの
- **USBビデオカメラ（Creative Labs）**
- **USB音源（Roland UA-30）**

が使われた。今回使われたUSBビデオカメラにはズーム機能がなかったため、始めは会場の後ろで中継したり、エンコーダマシンを丸ごと会場の前に持っていて中継するなど、いろいろあったが、中継そのものは滞りなく行われた。

今回、利用したエンコーダマシンはデスクトップマシンだったが、最近のノートPCでは、PCカードスロットが1つしかない代わりに、USBポートが標準装備されているものも多く、USBデバイスを利用してReal中継が実現できたという経験は、あまり前例がないうえに、今後の大きな糧となったと思う。また、筆者自身にも大きな励みとなった。今後は、USBデバイスが主流となっていくと思われるので、インターネット中継においても、LinuxのUSB対応は注目される点である。

最後に、NLUG主催のLinux勉強会で、いろいろな質問に答えいただいたNLUGの皆さんにこの場を借りて深く感謝したいと思う。

画面6　NLUG勉強会の中継に使用したUSBデバイス
中継に使用したCreatve LabsのUSBカメラ「WebCam 3」（手前）とRolandのUSB音源「UA-30」（奥）。

64ビットRISCプロセッサは高速数値計算に最適！

Linux for Alpha

MacintoshのPowerPCや、SunのSPARC／Ultra、Cobalt Qubeで使用されているMIPSなど、Linuxはx86以外にも多くのCPUに移植されている。Alphaプロセッサは、CPUのベンチマークテストによく使われているSPECint95、SPECfp95で常に世界最高速の位置に座り続けてきた高速CPUである。今回は64ビットCPUの世界をAlpha Linuxを通して見てみよう。

文：編集部
Text：Linux magazine

Alphaは64ビットRISCプロセッサ

IntelのCPUは、80386からPentium IIIまで高速化を図りながら進歩してきたが、基本的な命令セットを含むアーキテクチャは32ビットのままで変わっていない。しかし、今年中に同社からは初の64ビットCPU、Itaniumが発売される見込みであり、やっと次の世代へ踏み込むことになる。

ところで、主にサーバ機に使われているRISCプロセッサの多くは、ずいぶん前から64ビットCPUだ。その中のひとつ、Alphaプロセッサは1992年に発売された時から64ビットで設計され、当時からずば抜けた高性能を発揮してきた。

RISCプロセッサは、命令を単純にすることで回路を単純化し、チップ面積が小さくなることもあって高速なクロックで動作する。そのため、同時期に発売されたx86プロセッサに比べると数倍の性能を得ていた。

特に浮動小数点演算を多く使う科学技術計算の分野では、その性能の高さからRISCプロセッサの利用者が多いようである。

■Alphaプロセッサの歴史

Alphaプロセッサは、1991年に最初に発表された21064から、21164、21264と3世代のCPUが開発されている。そして製造プロセスに次世代のものを利用した改良型（Aが付く）もある（**表1**）。現在出荷されている製品には、21164A、21264、21264Aの3タイプが使用されている。そして次の21364（EV7）を含め、3世代先までの開発計画が発表されている。

21164Aは、4命令同時実行機能を持ったスーパースカラ構造で、命令用8Kバイト、データ用（ライトスルー）8Kバイトの1次キャッシュと、96Kバイト（3ウェイ・セットアソシエイティブ）のライトバック2次キャッシュを内蔵している。

21264、21264Aは、命令／データともに64Kバイトの1次キャッシュを内蔵し、4本の整数演算パイプラインと2本の浮動小数点演算パイプラインを持ち、4命令の同時実行が可能となっている。この21264からアウトオブオーダー機能を備えたので、プログラムの並び順に関係なく、実行可能な命令を投機的に実行していくことで性能の低下を防いでいる。

CPU	クロック	製造プロセス
21064（EV4）	150〜200MHz	0.75μm
21064A（EV45）	233〜300MHz	0.5μm
21164（EV5）	233〜433MHz	0.5μm
21164A（EV56）	400〜700MHz	0.35μm
21264（EV6）	466〜500MHz	0.35μm
21264A（EV67）	600〜750MHz	0.25μm

表1　Alphaプロセッサのスペック

Alphaプロセッサは、Digital Equipment Corporation（DEC）社が開発していたが、1998年にCOMPAQに買収されたため、現在ではCOMPAQがAlphaプロセッサの開発元というわけである。DECの半導体製造部門をIntelに売却したこともあって、現在ではAlphaプロセッサの製造はIntelとSamsungに委託している。一時期、三菱電機もAlphaを製造していたが撤退してしまった。

Alphaプロセッサを秋葉原などで売っているのを見ることはほとんどないが、Samsungの子会社であるAlpha Processor社（**画面1**）では、CPUとマザーボードなどを販売、サポートしている。これを利用して、Alphaマシンを組み立てて販売しているメーカーもある。しかし、多くのユーザーは、COMPAQから発売されているAlphaServer、AlphaStationを使用していると思われる。

■Compaq AlphaServer、AlphaStation

コンパックコンピュータから発売されているLinux対応のAlphaプロセッサ搭載マシンは、サーバ向けのAlphaServerとワークステーション向けのAlphaStationがある（**表2**）。

なお、同社では3月31日までという期限付きで「AlphaStationアドバンテージキャンペーン」を実施中で、最大51%引きという特別価格を設定している。AlphaStation XP900 Cコンパイラ付きパッケージは、Alpha 21264プロセッサ466MHz、256Mバイトメモリ、9.1GバイトUltra2 Wide SCSI（LVD）ハードディスク、日本語Tru64 UNIX（2ユーザーライセンス）、Cライセンス付きで88万円となっている。

そのほか、Alpha使ったハイエンドサーバを最大128ノードまで接続し、最大4096CPUという大規模な構成でスーパーコンピューティングを実現する「Compaq AlphaServer SCシリーズ」の発表も行われている。

Alpha Linuxのディストリビューション

Alphaプロセッサは、Tru64 UNIX（DIGITAL UNIX）、OpenVMS、Windows NT/Alpha、Linuxといったオペレーティングシステムでサポートされているが、Windows NT/Alphaだけは32ビットOSである。残念ながら、NTの後継であるWindows 2000ではAlphaはサポートされないようだ。

Alpha Linuxは64ビットOSとして作られている。カーネル2.2からカーネルのソースコードにAlpha対応部分が組み込まれている。そして、カーネルと各種のソフトウェアをAlphaに対応

画面1　Alpha Processor, Inc.
http://www.alpha-processor.com/
韓国のSamsungは、Compaqからライセンスを受けてAlphaプロセッサを製造している。1998年7月に設立された同社の子会社であるAlpha Processor, Inc.は、Alphaプロセッサとマザーボードの販売を行っている。

画面2　コンパックのLinux情報
http://linux.compaq.co.jp/
コンパックのLinuxに関する情報を集めた専用Webページ。Linux Alpha版のCompaq Fortran、Compaq Cコンパイラの情報も掲載されている。

製品名	CPU	メモリ	ハードディスク	価格
AlphaStation XP900	21264 466MHz	64Mバイト	9.1GバイトUltra2 Wide SCSI（LVD）	115万円～
AlphaStation XP1000	21264A 667MHz	128Mバイト	9.1GバイトUltra Wide SCSI	181万円～
AlphaServer DS10	21264 466MHz	128Mバイト	なし	99万円～
AlphaServer DS20E モデル6/500	21264 500MHz	256Mバイト	なし	225万3000円～

表2　Compaq Linux対応Alphaシステム

させたディストリビューションが、x86用Linuxと同様に発売されている。Alpha Linuxのユーザーは、絶対数が少ないために情報も少ない。そういう人のためにAlpha Linuxに関する情報を集めているWebサイトがある。**画面3**のThe AlphaLinux（http://www.alphalinux.org/）を見にいくとよいだろう。

■Red Hat Linux 6.1 for Alpha

米Red Hatより2000年1月12日より、Red Hat Linux 6.1 Deluxe for Compaq Alpha Systems英語版（以下Red Hat for Alpha）がリリースされた。Red Hatは以前からAlpha版Linuxディストリビューションパッケージを販売していて、今回のリリースはIntel版Red Hat Linux 6.1と同様のバージョンアップを行ったものだ。

Red Hat for Alphaの主な特徴は、以下のようになっている。

- **グラフィカルなインストーラ**
- **インターネットサーバ向け高機能クラスタリング**
- **LDAP（Lightweight Directory Access Protocol）と高機能なシステム管理の統合**
- **priority.redhat.comから迅速かつ簡単に最新情報を入手可能**
- **グラフィカル・ユーザーインターフェイスとして人気のあるKDE、GNOMEを標準デスクトップとして選択可能**
- **新PPPダイアラーでインターネットに簡単接続**

なお、Red Hat for Alphaの日本語版も開発しており、近日中にリリースされる予定となっている。

また、2月2日にRed HatはAPI（Alpha Processor社）と技術提携し、クラスタリング技術開発センターを設立した。Alphaを32プロセッサ組み込んだPowerRAC Chassis 320を2台で、64プロセッサのシステムを構成し、高いパフォーマンスを得るクラスタ技術の開発を行う。

■LASER5 Linux 6.0 for Alpha

レーザーファイブから2月14日に、LASER5 Linux 6.0 Server Edition for Alpha（以下LASER5 for Alpha）が発売された。当初昨年暮の発売予定であったが、出荷が遅れていたものである。

LASER5 for Alphaは、日本語redhat Linux 6.0 Server Edition（Intel版）をAlpha版に移植したもので、サポート権付きが4万9800円、サポート権なしが1万円となっている。サポートの内容は、インストールからサーバ設定まで180日間に3回までの質問を電話、FAX、Web（電子メール）で対応するというもの。

なお、サポート権付き、およびサポート契約はコンパックコンピュータが認定するAlphaプロセッサ搭載「Linux-readyモデル」のみが対象となっている。

日本語のインストーラを備え、データベースにPostgreSQL、日本語全文検索システムにMitakeSearch、ワークステーション用のデスクトップにはGNOME、セキュリティとしてセキュリティマネージャメニューやSSHなどが付属している。商用の日本語

画面3
The AlphaLinux
http://www.alphalinux.org/
Alpha Linuxに関する情報を探すなら、まずここを見よう。Alphaに関するニュースやドキュメント、プレスリリース、関連サイトへのリンクが掲載されている。

画面4
Alpha Users' Group
http://alpha-usr.gr.jp/
Alphaを使用しているユーザー同士の情報交換・交流の場を提供している。Alphaに関するメーリングリストが用意されている。

TrueTypeフォントは含まれていない。

■Kondara MNU/Linux

デジタルファクトリジャパンから発売されているKondara MNU/Linuxは、ひとつのパッケージにPC／AT互換機用とAlpha PC用の両方が含まれている。発売中のKondara MNU/Linux 1.0は6800円、4月1日発売予定のKondara MNU/Linux 1.1は7800円である。

1.0ではAlpha版に限ってサポートなしだったが、1.1からは電子メール、FAXによるサポートに加え24時間365日の電話サポートも行う。

Alpha PC対象機種は、COMPAQ Linux-readyモデル、Visual Technology VT-Alphaシリーズとなっている。

■TurboLinuxも？

ターボリナックスジャパンからも、はっきりとした時期は未定だが今年中にTurboLinux for Alphaのリリースが予定されている。

DS10にAlpha Linuxをインストールする

さて、今回はCompaq AlphaServer DS10（以下DS10）に、LASER5 Linux Server Edition for Alphaをインストールしてみた。DS10は、Alpha 21264 466MHzを採用したエントリーレベルのサーバである。ベースモデルは99万円からとなっているが、Linux版ならば80万円台になるとのことだ。ハードディスクはオプションなので、別途購入する必要がある。今回使ったDS10は、メモリを512Mバイトに拡張し、9.1GバイトのUltra2 Wide SCSIインターフェイスのハードディスクが装着されていた。

ここでは細かいことは省いて、x86版とLinux Alphaとの違いを説明しながら雰囲気をお届けしよう。実際にインストールする際には、ディストリビューションに付属するドキュメントをよく読んでいただきたい。

DS10にLinuxをインストールする場合、CD-ROMブートはできないため、あらかじめほかのLinuxマシンかWindowsマシンで、ブートフロッピーとRAMDISKフロッピーを作成しておく。

AlphaマシンのBIOSは見慣れたx86用のものとは違っている。DS10の電源を入れると各種の診断が実行されたあと、SRM（System Resource Manager）という高機能なモニタプロ

CPU	Alpha21264 466MHz（2MバイトL2キャッシュ）
メモリ	512MバイトECC SDRAM（100MHz DIMM、最大2Gバイト）
バス	128ビットメモリデータバス、クロスバースイッチ（最大1.3Gバイト／秒）
ハードディスク	9.1Gバイト Ultra2 Wide SCSI（LVD）×2台
CD-ROM	32倍速 ATAPI CD-ROM
SCSIコントローラ	Qlogic 1040B
グラフィックスカード	ELSA Gloria Synergy（PERMEDIA2）
ネットワーク	デュアル10/100Base-T
拡張スロット	PCIスロット×4（64ビットPCI×3、32ビットPCI×1）
パフォーマンス	SPECint95=24.6、SPECfp95=47.9
外形寸法（mm）	444（W）×460（D）×130（H）
重量	14.5kg

表3　今回使用したCompaq AlphaServer DS10システムの主な仕様

写真1　Compaq AlphaServer DS10
高さ13cmで3Uサイズ。ハードディスクドライブは3台まで内蔵可能。

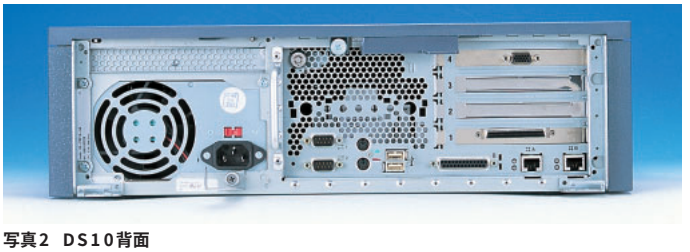


写真2　DS10背面
Ethernetポートはデュアル10/100Base-Tを搭載。

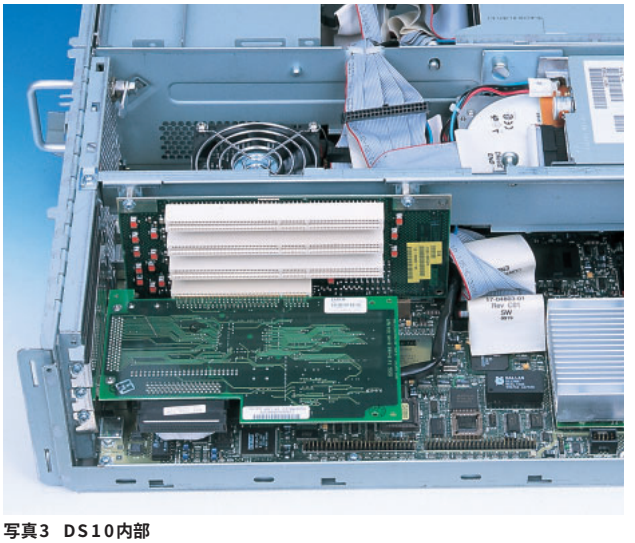


写真3　DS10内部
上部の少しだけ見えるファンの下にAlpha21264（466MHz）が搭載されている。
PCIスロットは4ポートで、QLogic 1040B（Ultra2 Wide SCSI）が装着されている。

グラムのプロンプト“P00>>>”が表示される。

そこで、“show device”と入力すると接続されているデバイスの情報が表示される。このDS10では、ハードディスクはSCSI、CD-ROMはATAPI（IDE）で接続されていて、SRMでのデバイス名は、ハードディスクはDKA0、CD-ROMはDQA0、フロッピーはDVA0となっている。ハードディスクやCD-ROMなどのシステム構成によってデバイス名が違ってくるので注意していただきたい。

次にフロッピーからブートするようにSRMパラメータをセットする。

```
P00>>set bootdef_dev dva0
P00>>set boot_file vmlinux.gz
P00>>set boot_osflags "root=/dev/fd0
  load_ramdisk=1 bootdevice=/dev/fd0"
```

先ほど作っておいたブートフロッピーを入れて、“boot”と入力するとLinuxのインストーラが起動する。途中、RAMDISKフロッピーに交換するようにメッセージが出力されるのでフロッピーを交換し、その後はx86版と同様に対話式にインストールを行っていく。

ハードディスクのパーティショニングを行うところで、fdiskを起動するが、ここで注意がある。BSD形式のディスクラベルを指定して、第3シリンダからパーティションを開始する必要があるという点だ。最初から第2シリンダまではブートローダのために空けておかないといけない。そのブートローダはLILOではなく、SRMブートローダというものを使用している。

Alpha Linuxの使用感はx86版とまったく変わらない

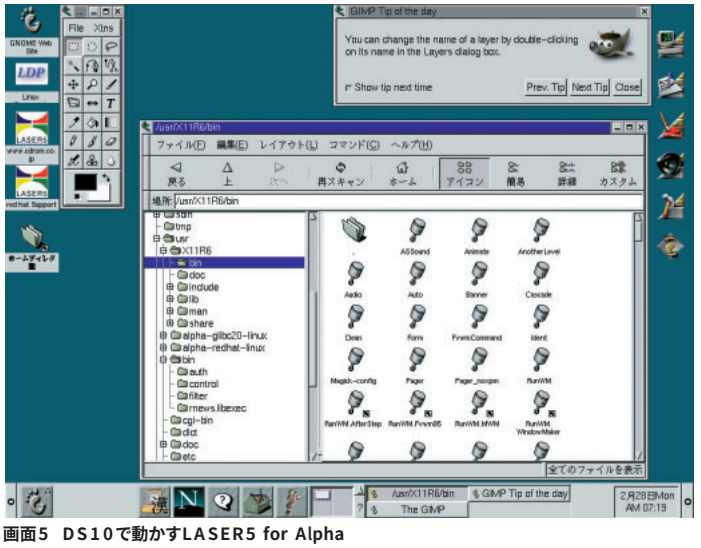
LASER5 for Alphaには、XFree86 3.3.3.1が用意されていて、GNOME、KDEといったデスクトップ環境もある。デモCDに入っているGNOME用の日本語パッチを当ててしまえば、Alphaマシンであることを忘れてしまうだろう（**画面5**）。

また、Netscape CommunicatorはLASER5 for Alphaに標準では入っていないが、COMPAQのWebサイト（http://www.compaq.com/partners/netscape/downloads/）から、Linux用のNetscape Communicator 4.7 for Alpha Linuxがリリースされている。さっそくDS10にもインストールしてみた。英語版だが日本語のWebサイトを見ることもできるし、日本語フォントの設定も可能だ（**画面6**）。

Alphaプロセッサ専用コンパイラ

Alpha Linuxでは、標準のCコンパイラとしてx86版Linuxと同様にGNUのgcc／egcs（以下gcc）が採用されている。Alphaプロセッサ用のgccは、たとえばlong型のデータサイズは8バイト（gcc-i386では4バイト）、ポインタ型のデータサイズは8バイト（gcc-i386では4バイト）といった具合に、64ビットCPU用に拡張されている。

コンパックコンピュータから2月9日に、Linux Alpha版Compaq Fortran V1.0（以下Compaq Fortran）と、Linux Alpha版Compaq C V6.2（以下Compaq C）コンパイラが発売された。これらは、Tru64 UNIXやOpenVMSといったOSで提供されていたコンパイ



画面5　DS10で動かすLASER5 for Alpha
LASER5 Linux 6.0 Server Edition for Alphaでは、GNOMEデスクトップ環境を構築することができる。日本語化パッチを当てたGNOMEを使うとワークステーション用途でも実用になる。

画面6　Netscape Navigator for Alpha Linux
Linux Alphaに対応するNetscape Navigatorもベータ版がリリースされた。画面はAlphaServer DS10でLASER5 Linux上で動作させたもの。

ラをLinux用に移植したものである。Alpha用に最適化されたオブジェクトを生成するとともに、専用の数値計算ライブラリを使用することによって、gccに比べて数倍から十数倍のパフォーマンスが得られるという。

Compaq Fortranは、1システム用が6万8000円、5システム用が15万3000円、10システム用が25万8000円となっている。Compaq Cは1システム用が8000円である。

そのほかに、Compaq Fortranには専用デバッガ（Ladebug）も提供されている。

■Compaq Cコンパイラを試す

gccに比べてCompaq Cコンパイラがどれくらい優秀なオブジェクトを出力するのか、先月号の「Linuxぽりぽりチューニング」でも使われていたnbenchプログラムで試してみた。

nbench（http://members.xoom.com/Gavrilov_Con/nbench/）は、10種類のアルゴリズムをテストして、最後にメモリ、整数演算、浮動小数点演算の結果をAMD K6 233MHzを1としたときの相対値で表示してくれる。Compaq Cコンパイラでは、オプションに“ -arch ev6 -O4 ”を指定した。その結果は**リスト1**のように出力される。

Alphaだけでなく、参考までにPentium III 500EMHzとCeleron 400MHzの結果も計測してみた（**グラフ1**）。

gccと比較してみると、浮動小数点演算の結果が2.75倍も速くなっている。同じマシンでコンパイラの性能だけでこんなに違うとは驚きだ。コンパックのリリースよるとアプリケーションによっては最大4倍も高速になるとのことである。

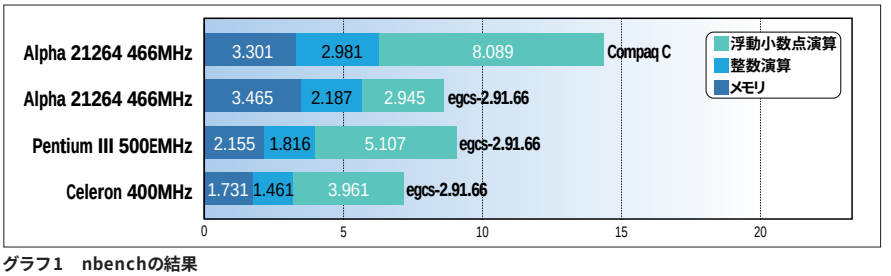
また、CPUによる違いを見てみると、同じegcsでコンパイルした場合には、ほぼクロック周波数に応じた性能に見える。このベンチマークでは傾向しかわからないが、CPUの性能を測るためのベンチマークプログラムを作っている SPEC（ Standard Performance Evaluation Corporation、http://www.spec.org/）によると、Pentium IIIの性能（＋コンパイラの性能？）は相当上がっていて、Alphaといえども侮れない状況になってきているようだ。

製造量の違いとはいえ、Pentium IIIとAlphaでは価格に差があるため、Alphaを手軽に使えるわけではないが、Pentium III Xeonといったサーバ用途のプロセッサを検討するときには、じっくりとパフォーマンスを比較してみるとよいだろう。

▼リスト1 nbenchをCompaq Cコンパイラで実行した結果（Alpha21264 466MHz）

```
BYTEmark* Native Mode Benchmark ver. 2 (10/95)
Index-split by Andrew D. Balsa (11/97)
Linux/Unix* port by Uwe F. Mayer (12/96,11/97)

TEST                : Iterations/sec.  : Old Index   : New Index
                    :                  : Pentium 90* : AMD K6/233*
--------------------:------------------:-------------:------------
NUMERIC SORT        :          360.03  :       9.23  :       3.03
STRING SORT         :          42.811  :      19.13  :       2.96
BITFIELD            :     7.5631e+07   :      12.97  :       2.71
FP EMULATION        :          14.672  :       7.04  :       1.62
FOURIER             :           12821  :      14.58  :       8.19
ASSIGNMENT          :          4.5439  :      17.29  :       4.48
IDEA                :          975.99  :      14.93  :       4.43
HUFFMAN             :          408.25  :      11.32  :       3.62
NEURAL NET          :          9.6311  :      15.47  :       6.51
LU DECOMPOSITION    :          265.49  :      13.75  :       9.93
==========================ORIGINAL BYTEMARK RESULTS==========================
INTEGER INDEX       : 12.479
FLOATING-POINT INDEX: 14.585
Baseline (MSDOS*)   : Pentium* 90, 256 KB L2-cache, Watcom* compiler 10.0
============================LINUX DATA BELOW============================

number of CPUs      : 1
C compiler          : ccc (unknown version)
libc                : libc-2.1.1.so
MEMORY INDEX        : 3.301
INTEGER INDEX       : 2.981
FLOATING-POINT INDEX: 8.089
Baseline (LINUX)    : AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38
* Trademarks are property of their respective holder.
```

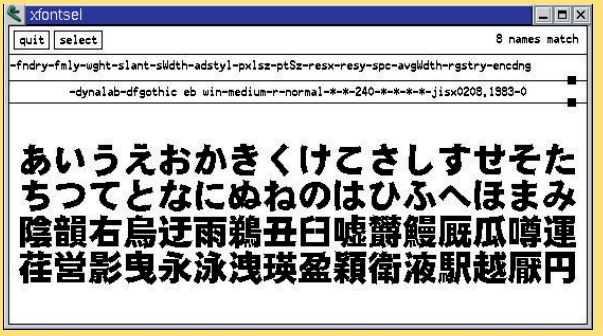
この部分をグラフにした



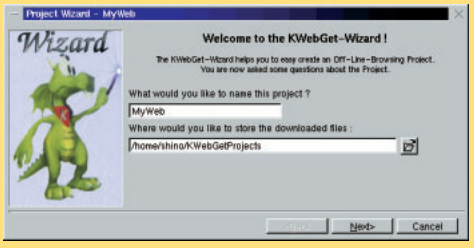
グラフ1 nbenchの結果

Free Application Showcase

文：出井 一
Text : Hajime Dei



ttftool P.137



KWebGet P.148

kishido P.151

※紹介したソフトは、すべて付録CD-ROMに収録されています。

TrueType／Type1フォントの設定をGUIで行う

ttftool

バージョン：4.3.1／2.9　　種別：GPL

http://www2.famille.ne.jp/~mituiwa/#font

ttftoolは、TrueTypeフォントやType1フォントのインストールや設定をコントロールパネルで行うソフトだ。X-TTやVFlibなどでこうしたフォントを扱う際、従来は設定ファイルをエディタを使って書き換える必要があった。ttftoolでは、インストールするフォントを選択するだけで、あらかじめ用意されたテンプレートに従って設定ファイルを作成してくれる。動作にはGTK＋が必要だ。

●ビルドとインストール

ttftoolは、tarボールとRPMパッケージの両方で配布されている。RPMはVine／Turbo／LASER5といったディストリビューション別に用意されているが、バージョンが2.9と古く、Type1フォントに対応していない。以下の説明では、最新版（バージョン4.3）のtarボールを利用する。

tarボールを展開したら、まずはディストリビューションごとにサブディレクトリに分けられたテンプレートを/etcにコピーする。たとえば、LASER5の場合は、

```
# cp config.laser5/* /etc
```

とすればいい。

ビルドは、ttffiles、getttinfo、getatmfinoのそれぞれのディレクトリで、「make」→「make install」とする。

ただし、そのままだとインストール先が/usr/binになる。/usr/local/binにインストールしたいなら、「make TTF_TOOL_INST_DIR=/usr/local/bin install」としよう。

●コントロールパネルで登録する

ttftoolでは、TrueTypeフォントとType1フォントをインストールするディレクトリの設定を、/etc/startup.ttfdefから読み込む。初期値はどちらも「/usr/X11R6/lib/X11/fonts/TrueType」だ。なお、LASER5では、これらの「TrueType」を「truetype」に修正する必要がある。

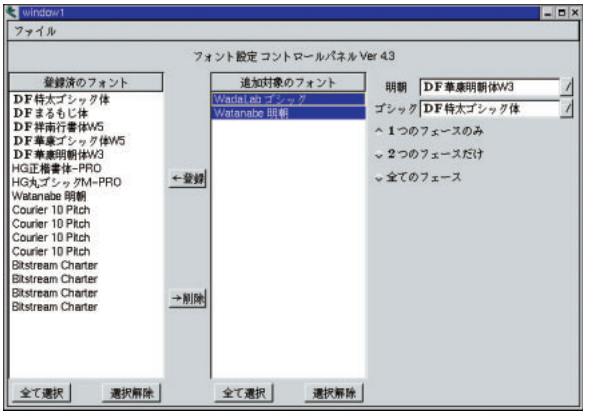
「ttffiles」としてコントロールパネルを起動すると、上記のインストール先ディレクトリに置かれているフォントの名前が、[登録済みのフォント]に一覧表示される（**画面1**）。

新たにフォントをインストールするには、[ファイル]－[インストール]で、目的のフォントが含まれるディレクトリを開く。続いて、[追加対象のフォント]からインストールしたいフォントを選択しよう（複数選択可）。[登録]ボタンを押すと、選択したフォントがインストール先ディレクトリにコピーされる。

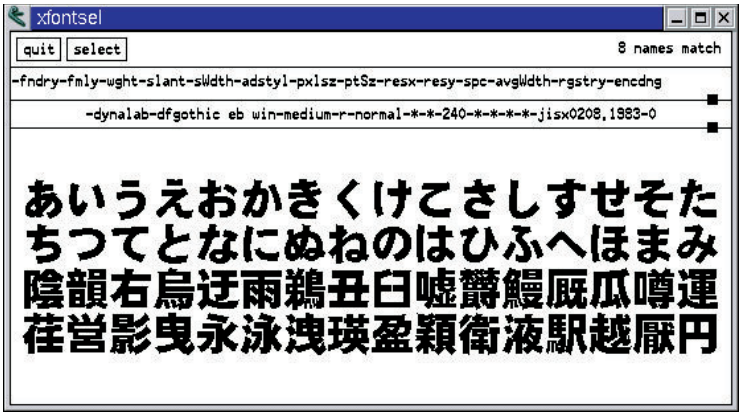
fonts.dirやfonts.aliasなどの設定ファイルは、[ファイル]－[設定ファイルの書き出し]で、テンプレートの設定に従って作成される。既存の設定ファイルの内容は参照されず、そのまま上書きされてしまうことに注意されたい。自分でfonts.aliasなどを書き換えて使っていた場合は、あらかじめ別の名前で保存しておこう。

X-TTの場合、新しいfonts.dirとfonts.aliasの設定は、Xを再起動するか、「xset fp rehash」とすることで有効になる。

追加したフォントをxfontselで確認しよう（**画面2**）。



画面1
コントロールパネルでは、簡単な操作でフォントを登録できる。



画面2
インストールしたフォントの外見をxfontselで確認する。

フローチャートなどのダイアグラムを描画

Dia

バージョン：0.83　　種別：GPL

http://www.lysator.liu.se/~alla/dia/dia.html（オリジナル）
http://www2.famille.ne.jp/~mituiwa/#dia（日本語版）

Diaは、フローチャートや回路図、組織図、ネットワーク構成図など、科学・ビジネス分野で用いられる「ダイアグラム」を効率よく描けるソフト。オリジナルのDiaには、日本語のカタログファイルが付属するものの、ダイアグラム中では日本語が使えない。このため、日本語フォントを利用可能にするパッチが開発され、日本語版として公開されている。動作にはGTK＋1.2以降が必要だ。

●ビルドとインストール

Diaは、ファイル一式をtar＋gzipしたtarボールで配布されている。日本語化パッチ（dia-0.83-jp3.diff）も忘れずに取得しよう。

日本語化パッチを当てるには、適当なディレクトリ（/usr/srcなど）でtarボールを展開した後、展開先のディレクトリ（dia-0.83）に移動し、

```
$ patch -p1 <（パッチを置いたディレクトリ）/dia-0.83-jp3.diff
```

とすればいい。

なお、Dia付属の日本語カタログ（ja.po）には、ポップアップメニューの項目が正常に表示されないという不具合があるため、この問題を解消するパッチ（dia-0.83-ja.po.patch）を用意した。日本語化パッチと同様、

```
$ patch -p1 <（パッチを置いたディレクトリ）/dia-0.83-ja.po.patch
```

としてja.poを修正しよう。

修正が済んだら、「./configure --with-included-text」→「make」→「make install」という手順でビルドとインストールを行う。configureスクリプトのオプションは、日本語を扱うために必要だ。

●メインウィンドウの表示

ktermなどのコマンドラインで「dia」として起動すると、描画用のボタンが並んだメインウィンドウが開く（**画面1**）。なお、動作中にコンソールに表示される警告メッセージが気になるようなら、「dia &> /dev/null」としてメッセージを/dev/nullにリダイレクトするといい。

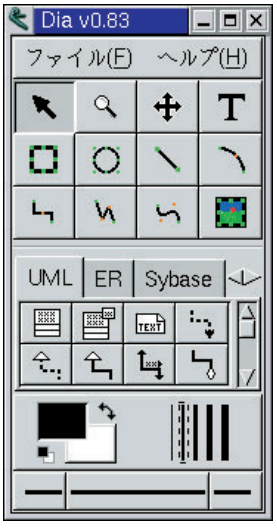
ウィンドウ上部には、ダイアグラムを描く際に使われる部品（オブジェクト）のうち、基本オブジェクト9種類（テキスト・ボックス・楕円・線・弧・ジグザグ線・連続線・ベジエ曲線・イメージ）を生成するボタンと、オブジェクトの変更や表示の拡大・スクロールといった操作を行うボタンが並んでいる。

さらに、ウィンドウ中央部には、フローチャート、ネットワーク、回路図など分野別の専用オブジェクトが、タブ付きページで分類されている。なお、起動時のウィンドウサイズでは、タブが一度に2、3個しか表示されないため、切り替えが面倒だ。ウィンドウサイズを変更して、すべてのタブが一度に表示される大きさにするとよいだろう（**画面2**）。

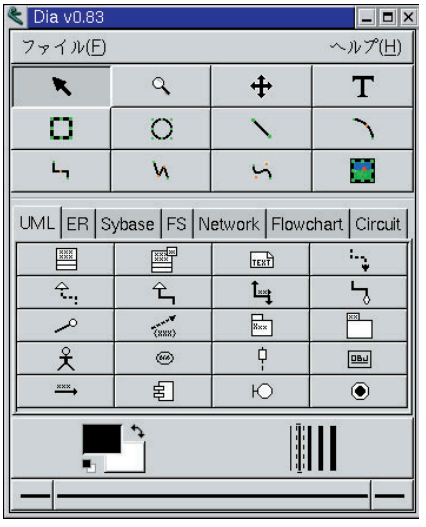
ウィンドウ下部には、オブジェクトの前景色・背景色、線の太さ、線の属性（両端に付ける矢印の形状や線の種類など）を変更するためのボタンが用意されている。

●基本オブジェクトを生成する

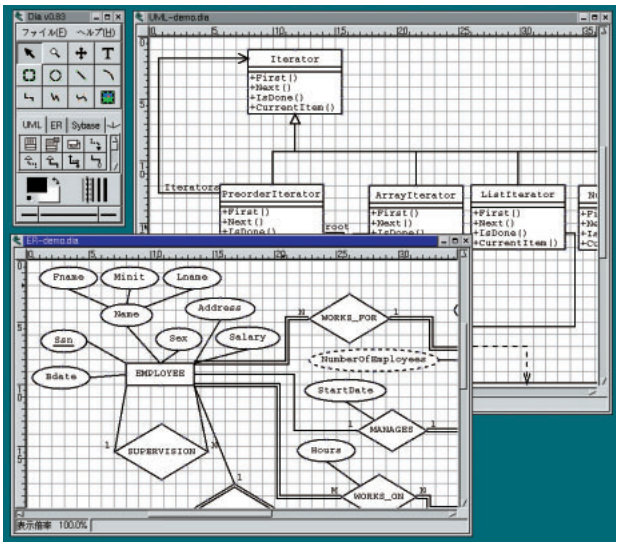
メニューの[ファイル]－[新規]／[開く]を選択すると、ダイアグラムを描くためのビューウィンドウが開く。Diaでは、複数のビューを同時に扱える（**画面3**）。それぞれのビューには、位置決めの参考となる「グリッド」が表示されている。グリッドの幅を変えたり、常にグリッドに沿ってオブジェクトを配置ことも可能だ。



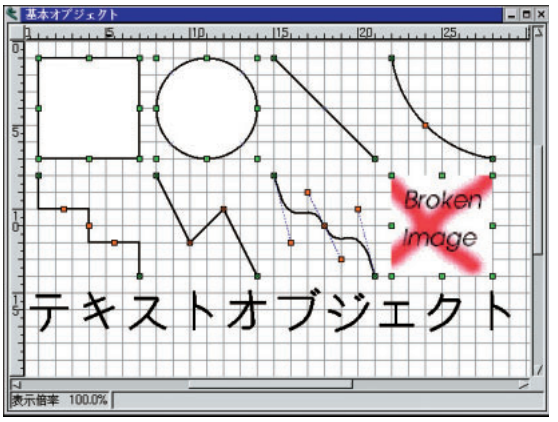
画面1
メインウィンドウには、描画用のボタンが並んでいる。



画面2
ウィンドウを広げて、すべてのタブを表示したところ。



画面3
ダイアグラムを描くビューウィンドウは同時に複数開ける。



画面4
7種類の基本オブジェクト。ハンドルで大きさなどを変更可能。

まずは、基本オブジェクトをビューに描いてみよう（**画面4**）。たとえば、[ボックスの生成]ボタンを押し、ビュー上でクリックすると、小さな矩形が表示される。クリックの代わりにドラッグすると、そのままボックスのサイズを変更できる。

オブジェクト上に表示されている緑の点は、オブジェクトの大きさや長さを変更するための「ハンドル」だ。[オブジェクトの変更]ボタン（矢印）が押された状態でこれらをドラッグすることで、オブジェクトの大きさを自由に変更できる。また、ハンドル以外の部分をドラッグすると、オブジェクトが移動する。

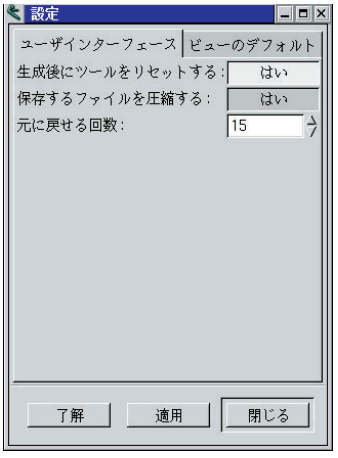
オブジェクト生成後、すぐにハンドルを操作したいなら、[ファイル]－[設定]で設定ダイアログを開き、[生成後にツールをリセットする]を[はい]に変更しよう（**画面5**）。以後は、生成系のボタンから[オブジェクトの変更]ボタンへの切り替えが自動的に行われるようになる。

●オブジェクトの属性を変更する

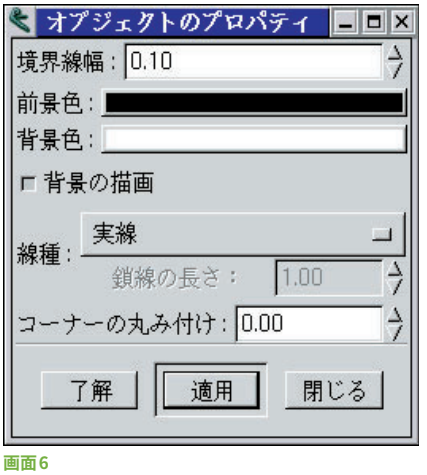
生成したオブジェクトの属性は、左ボタンのダブルクリックで開く「プロパティダイアログ」や、中ボタンのクリックでポップアップする「オブジェクトメニュー」で変更する。

プロパティダイアログの内容はオブジェクトの種類によって異なり、オブジェクトごとに独立した設定が可能。たとえば、ボックスのプロパティでは、境界線の幅、前景色、背景色、背景の描画（あるいは透過）、境界線の種類、コーナーの丸み付けなどを設定できる（**画面6**）。

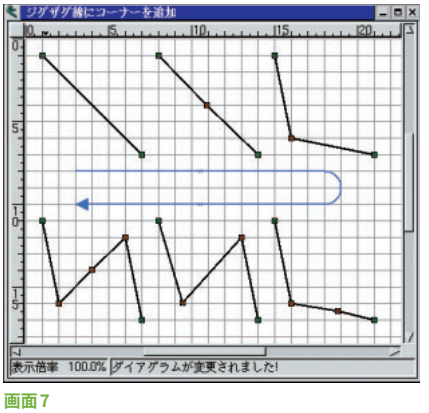
一方、オブジェクトメニューは、ジグザグ線や連続線、ベジエ曲線のように、ハンドルの追加が可能なオブジェクトにのみ用意されている。連続線の場合、生成時の外見は線のオブジェク



画面5
設定ダイアログで[生成後にツールをリセットする]を[はい]に。



画面6
プロパティダイアログの内容はオブジェクトの種類ごとに異なる。



画面7
中ボタンクリックのメニューで連続線にコーナーを追加する。

トと区別できない。オブジェクトメニューでコーナーを追加することではじめて、途中でいくつも折れ曲がった形状に変えられるのだ（**画面7**）。ジグザグ線やベジエ曲線も同様にハンドルを追加できる。

●オブジェクトのグループ化

操作の対象となるオブジェクトを切り替えるには、[オブジェクトの変更]ボタン（矢印）が押された状態で、目的のオブジェクトをクリックすればいい。通常は、一度にひとつのオブジェクトだけが選択されるが、Shiftキーを押しながらオブジェクトをクリックするか、ドラッグによる範囲指定を使うと、複数のオブジェクトをまとめて選択できる。

複数オブジェクトを選択した状態では、右クリックメニューの[オブジェクト]－[水平位置]／[垂直位置]以下の項目により位置合わせが可能だ（**画面8**）。等間隔に並べたり、隣接させることもできる。

また、[オブジェクト]－[グループ化]では、選択されたオブジェクト全体をひとつのオブジェクトとして扱うグループ化を行える。複数のオブジェクトの位置関係を変えたくない場合に利用するといい。元の状態に戻すには、[オブジェクト]－[グループ解除]を選択する。

●面と線を結びつける接続点

複数のオブジェクトをまとめて扱う方法には、グループ化のほかに「接続点」もある。これは、面タイプのオブジェクト（ボックスなど）に複数用意されているもので、線タイプのオブジェクト（連続線やベジエ曲線も含む）と結合するために利用できる。

接続点は初期設定では画面に表示されない。右クリックメニューの[表示]－[接続点の表示]をオンにすると、面タイプのオブジェクトの接続点が×印で表示されるようになる。

これらの接続点に、線タイプのオブジェクトのハンドルを近づけると、自動的に結合してハンドルの色が赤に変化する。この状態でボックスや線を移動しても結合は解けず、自動的に線の長さが変化する（**画面9**）。結合を解除するには、結合部分のハンドルをドラッグすればいい。

●レイヤによる前後関係の変更

オブジェクトの前後関係は、右クリックメニューの[オブジェクト]－[背面に送る]／[前面に送る]で変更する。しかしオブジェクトの数が増えてくると、この方法で前後関係を調整するのがしだいに面倒になる。

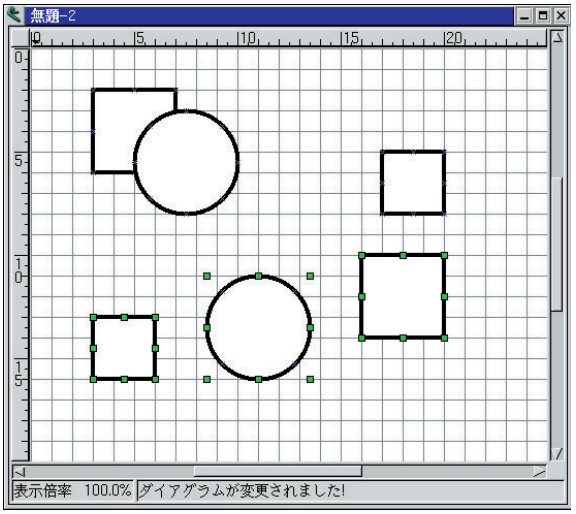
そこで、ドロー系ソフトでおなじみの「レイヤ」が用意されている。レイヤの前後関係を変更することで、各レイヤに属するオブジェクトの前後関係を一気に変更できるわけだ。

右クリックメニューの[ダイアログ]－[レイヤ]でレイヤダイアログを開こう（**画面10**）。レイヤの生成や削除、前後関係の変更はこのダイアログで行う。レイヤ名をダブルクリックすることで名称の変更も可能だ。

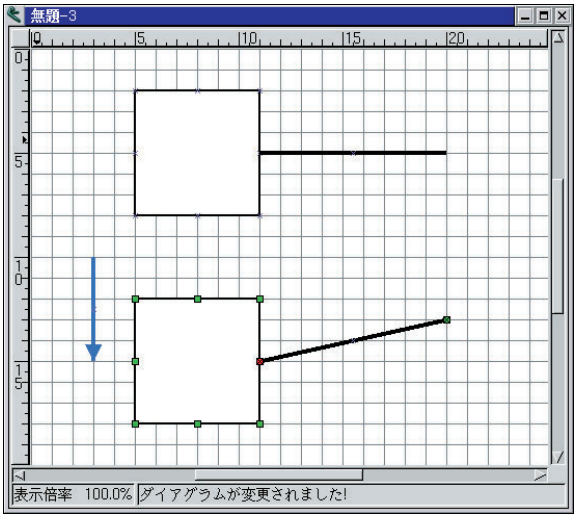
複数のレイヤが生成された状態でも、ビュー上には全レイヤの内容が表示される。ただし、オブジェクトを生成、選択できるのは現在選択されているレイヤだけだ。レイヤ間のオブジェクトの移動には、右クリックメニューの[編集]－[切り取り]／[貼り付け]を利用する。オブジェクトを選択してカットし、レイヤを切り替えてからペーストすればいい。

●分野別のオブジェクトを使う

Diaには、基本オブジェクトのほかにも、フローチャートや回路図、ネッ

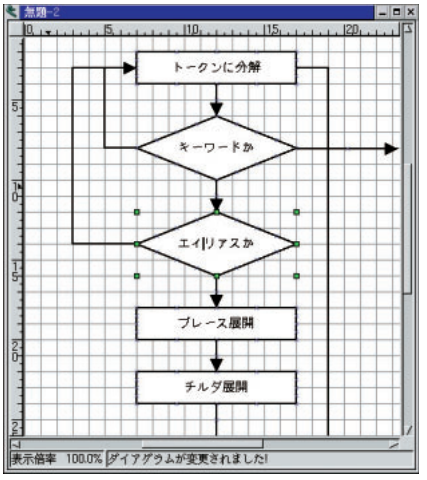


画面8
複数のオブジェクトを選択して位置合わせを行う。

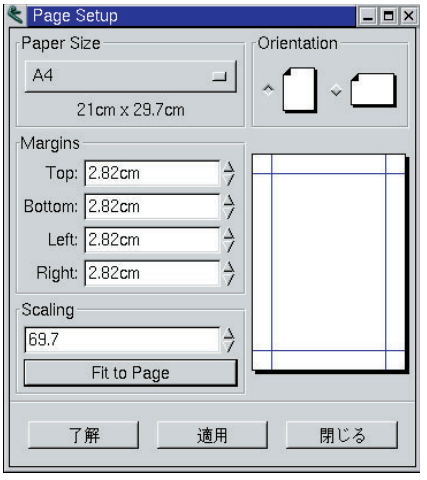


画面9
接続点に結合したオブジェクトは、移動しても結合したままだ。

画面10
前後関係の整理に便利なレイヤーを操作するダイアログ。



画面11
フローチャートのオブジェクトはサイズが自動的に変化する。



画面12
ページ設定で印刷する用紙のサイズや向きを設定する。

トワーク構成図、データベース、UML（統一モデリング言語）など、分野別のオブジェクトが用意されている。なお、UMLは、1997年に標準化されたオブジェクト指向モデルの表記法だ（http://www.rational.com/uml/index.jtmplなどを参照）。

これらのオブジェクトを利用すると、基本オブジェクトを組み合わせるよりもはるかに簡単に専門分野のダイアグラムを作成できる。たとえば、フローチャートの場合、内部に記述するテキストの長さに合わせて、オブジェクトのサイズが自動的に変化するといった具合だ（**画面11**）。

オブジェクトの生成自体は、基本オブジェクトの場合と違いはない。ただし、サイズ変更のできないオブジェクトや、操作の取り消し・やり直しに対応していないオブジェクトがあるなど、基本オブジェクトに比べて制限が多いので注意されたい。

ところで、こうした分野別オブジェクトや基本オブジェクトの多くはC言語により記述されているが、XMLを使ってオブジェクトの形状や接続点の情報などを記述し、Diaに組み込むことも可能だ。実際、回路図用のオブジェクトはこの方法により組み込まれている。

詳細は、ソース中のdoc/custom-shapesを参照されたい。

●ダイアグラムの保存と印刷

作成したダイアグラムは、[ファイル]－[保存]／[名前を付けて保存]でDia専用形式で保存できる。また、[ファイル]－[エクスポート]でEPS／SVG／CGM形式に変換することも可能だ。

ダイアグラムの印刷は、Dia自身から直接行える。まず、[ファイル]－[ページ設定]で、印刷する用紙のサイズや向きを設定し、ダイアグラムをそれに合わせてスケーリングしておく（**画面12**）。

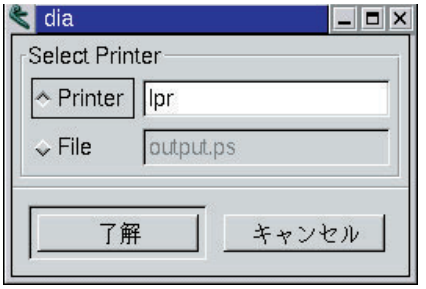
続いて、[ファイル]－[ダイアグラムを印刷]で、lprへの出力か、PS形式でファイルに保存するかを選択する（**画面13**）。

もちろん、いったんEPS形式でエクスポートしたファイルを、GVなどを使って印刷してもよいが、その場合は用紙サイズの設定は反映されないので注意されたい。

●日本語化について

操作中に、Shift－スペースキーによる日本語入力の切り替えが効かなくなることがあった。新しいビューでは切り替えられるので、全オブジェクトを新しいビューにコピー＆ペーストすれば回避できる。このほか、日本語のテキスト中に文字を挿入すると文字化けしてしまうといった不具合がいくつか見られた。

また、現時点では、日本語のフォントは「明朝」（-*-mincho-medium-r-normal-*-%d-*-*-*-*-*-*-*-）のみに限定されている。将来的には日本語化パッチにより、多書体の日本語TrueTypeフォントがサポートされる予定だ。



画面13
Diaからlprへ出力して直接印刷することも可能だ。

Vzライクな特徴を持つ高速テキストエディタ

ne

バージョン：3.00pre17（安定版）/00年02月06日版（実験版）種別：GPL

http://www3.justnet.ne.jp/~ele/ne/index.html

neは、ktermなどの端末画面で軽快に動作する日本語対応のテキストエディタだ。MS-DOS用の高速テキストエディタとして一世を風靡したVzエディタによく似たインターフェイスが特徴。モードレスなWordstar系エディタをDOSやWindowsで愛用している人には、viやEmacsよりもはるかにとっつきやすいだろう。なお、キー割り当ては設定ファイルを変更することで柔軟にカスタマイズできる。

●ビルドとインストール

neは、ファイル一式をtar＋gzipしたtarボールで配布されている。現在の最新版は、安定版の3.00pre17と実験版の00年02月06日版だ。どちらも、「./configure」→「make」→「make install」という手順でビルドとインストールを行える。

なお、ktem上で使用する場合、そのままではF1～F4キーが使えない。これはneのせいではなく、実際のF1～F4キーの設定と、端末情報データベースterminfoのktermエントリの設定が一致していないためだ。

そこで、次のようにしてterminfoのktermエントリを修正する。対象ファイルは、LASER5では/usr/lib以下、VineやTurboでは/usr/share以下のterminfo/k/ktermだ。

terminfoのエントリはバイナリなので、「su -」としてスーパーユーザーになった状態で、infocmpによりテキストのソースに変換する。

```
# infocmp kterm > kterm.src
```

続いて、ソース中のF1～F4キーの設定「kf1=\EOP」「kf2=\EOQ」「kf3=\EOR」「kf4=\EOS」を、「kf1=\E[11~」「kf2=\E[12~」「kf3=\E[13~」「kf4=\E[14~」に修正する。

最後に、修正したソースをticを使ってバイナリに変換する。元のエントリも別名で保存しておこう。

```
# mv kterm kterm.orig
# tic kterm.src
```

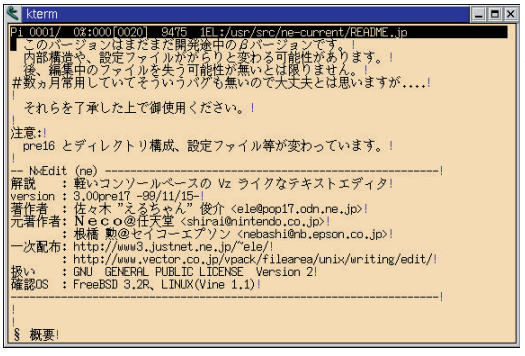
●画面と基本操作

ktermなどのコマンドラインで「ne ファイル名」として起動すると、端末画面にそのファイルの内容が表示される（**画面1**）。JISやシフトJISで書かれたファイルは自動的に日本語EUCに変換され、保存時に元の文字コードに戻される（変更可）。

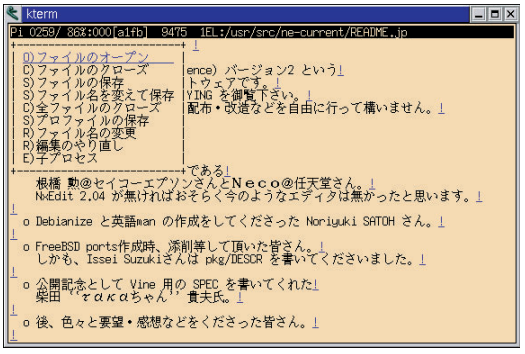
画面最上行には、ページングモード（後述）、挿入・上書きモード、カーソル位置の行番号、桁数、文字コード、編集中のファイル名といった情報が表示される。

初期設定では、Vzライクなキー割り当てになっている。詳細は/usr/local/etc/ne/key.vzを参照されたい。もちろん日本語の入力や編集も問題なく行える。なお、各行末の青い「!」は改行コードを表わす（表示しない設定も可能）。

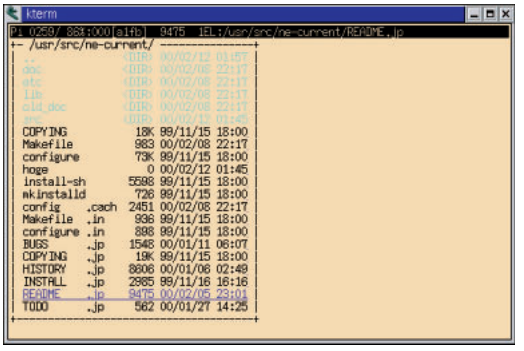
最初からファンクションキーやカー



画面1
kterm上でneを利用する。画面表示も含め、動作は高速だ。



画面2
ファンクションキーで開くメニューが用意されている。



画面3
ファイルオープン時などに利用できるファイラ。

ソルキーが使えるので、初心者にも使いやすいだろう。たとえば、ファイル関係の操作はF1キーのメニューから選択できるし（**画面2**）、ファイラーの一覧画面でオープンするファイルを選択できる（**画面3**）。

一方、ホームポジションから手を移動したくない中上級者は、Ctrl－QやCtrl－K、Esc（Ctrl－[）キーで始まる2ストロークキーを利用しよう。たとえば、ファイルのオープンやセーブは、Esc（Ctrl－[）キーに続けてOキーやSキーを押すことで実行できる。

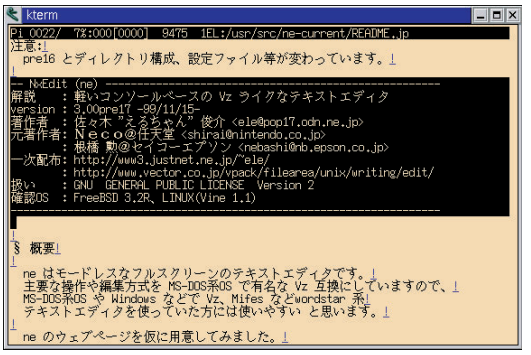
●検索とページングモード

Vzエディタと同様に、neでは検索文字列の指定と検索の実行が独立している。検索文字列の指定はF6（Ctrl－QF）キーで行う。また、F5（Ctrl－L）キーでは、カーソル位置の単語が検索文字列となる（複数回押すことで後ろの単語を追加可能）。

こうして検索文字列が設定されると、画面最上行左端のページングモードの表示が「S」（文字列検索モード）に変わり、PageUp／Down（Ctrl－R／C）キーの動作が、検索文字列の後方／前方検索に変化する。

文字列検索モードを抜けるには、Ctrl－@キーを押せばいい。ページングモードの表示が「P」（通常ページングモード）になり、PageUp／Downキーは通常のページ切り替えに戻る。

画面4
ブロックモードでは、選択状態のテキストが反転する。



●スタッカブルなカット＆ペースト

neでカット（コピー）＆ペーストを行うには、まずF10（Ctrl－B）キーを押してブロックモードに入る。この状態でカーソル移動やページ切り替えを行うと、選択状態のテキストが反転表示になる（**画面4**）。

続いて、カットならF8（Ctrl－Y）キー、コピーならShift－F8（Ctrl－KK）キーを押す。なお、カーソル行だけをカットする場合は、ブロックモードに入らないでそのままF8（Ctrl－Y）キーを押すだけでいい。F9（Ctrl－J）キーを押すと、カット（コピー）した内容がカーソル行の上にペーストされる。

特徴的なのは、カット（コピー）＆ペースト用のブロックバッファがスタッカブルであることだ。つまり、複数回カットしてから、同じ回数ペーストすると、最後にカットした内容から順番にペーストされる。1回カット（コピー）した内容を複数回ペーストする場合は、Shift－F9キー（Ctrl－K C）キーでペーストする必要がある。

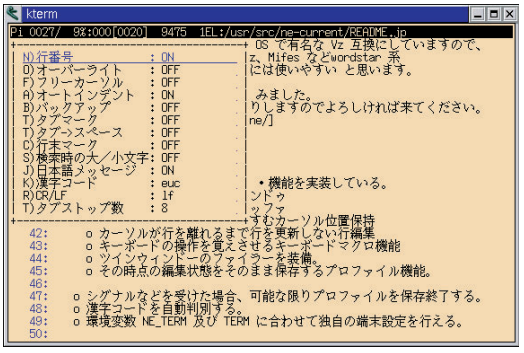
このほか、文字単位の削除についてもスタッカブルなバッファが用意されており、最後に削除した文字から順にCtrl－Uキーでペーストできる。

●柔軟なカスタマイズ機能

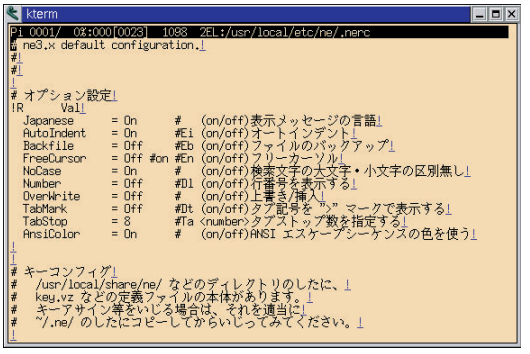
行番号やフリーカーソル、オートインデント、タブ・改行表示、タブ表示幅、検索時の大・小文字の区別といったオプション設定は、F4キーで動作中に変更できる（**画面5**）。

これらのオプション設定の初期値は、/usr/local/etc/ne/.nercという設定ファイルに記述されている。このファイルを各ユーザーの~/.ne/.nercにコピーして内容を変更することで、ユーザーごとに好みの初期設定が可能だ（**画面6**）。

同様に、キー割り当ての変更もこのファイルで行えるが、通常は別ファイルに記述しておき、制御命令「!I」を使って読み込ませるといい。/usr/local/etc/ne/.nercでも、「!I」を使ってkey.vzなどからキー割り当ての設定を読み込んでいる。



画面5
行番号表示などのオプションは動作中に設定を変更可能だ。



画面6
オプションの初期値やキー割り当ての設定は.nercで行う。

英単語などの語彙学習用の簡易データベース

Joe's QVocabulary

バージョン：0.20.4　　種別：フリー

http://www.qvocab.seul.org/

Joe's QVocabulary（以下QVocab）は、英単語・熟語とその意味といった対になる言葉を学習するための簡易データベース。単語の入力や修正、検索、テストなどの機能を持つ。それぞれの単語はテストの結果によって5段階に分類されるため、不得手な単語だけを集中して学習することも可能だ。日本語対応のKDE／Qt上で使えば、Qvocab自体には修正を加えることなく日本語を扱える。

●ビルドとインストール

QVocabは、tarボールとRPM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DebianとSuSe用なので、その他のディストリビューションでは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しよう。

「rpm --rebuild qvocab-0.20.4-1.src.rpm」とすると、/usr/src/redhat/RPMS/i386にqvocab-0.20.4-1.i386.rpmとqvocab-convert-0.20.4-1.i386.rpmが作成されるので、これらをインストールすればいい。

●データベースの新規作成

「qvocab」として起動するとウィンドウが開く。まずは、単語を登録するためのデータベースを新規作成しよう。[File]－[Create new database]を選択し、ユーザーのホームディレクトリが表示されているフィールドに適当なファイル名（拡張子不要）を追加入力して、右側の[Create database]ボタンを押す（**画面1**）。

すると、その下のフィールドにデータベースファイル名（拡張子.qvo）が表示される。

たとえば、hogeというユーザーが「/home/hoge/qvtest」と入力してボタンを押すと、/home/hogeqvtest.qvoというデータベースが作成され、下のフィールドに表示される。

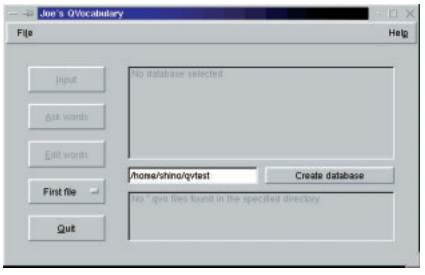
なお、ホームディレクトリのデータベースファイルはQVocab起動時に検索されるので、次からは自動的に下のフィールドに表示される。こうしたデータベースは複数作成でき、QVocab実行中に切り替えることも可能だ。

●単語の入力

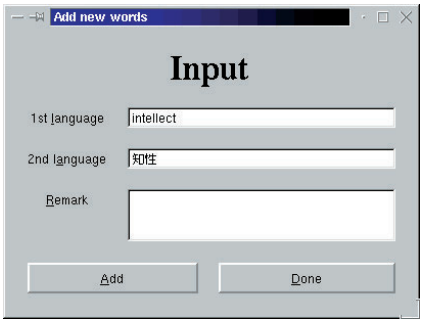
データベースを読み込むには、リストボックスのファイル名をダブルクリックすればいい。[Input][Ask words][Edit words]の3ボタンが使用可能になり、現在登録されている単語の合計数が表示される。

Qvocabのデータベースは、First、Second、…、Fifthの5ファイルに分かれており、登録した単語はテストの結果により自動的にこれらのファイル間を移動する。現在検索されているファイルは、下から2番目のボタンに表示されている（初期設定は「First」）。このボタンを利用すれば他のファイルに切り替えられる。

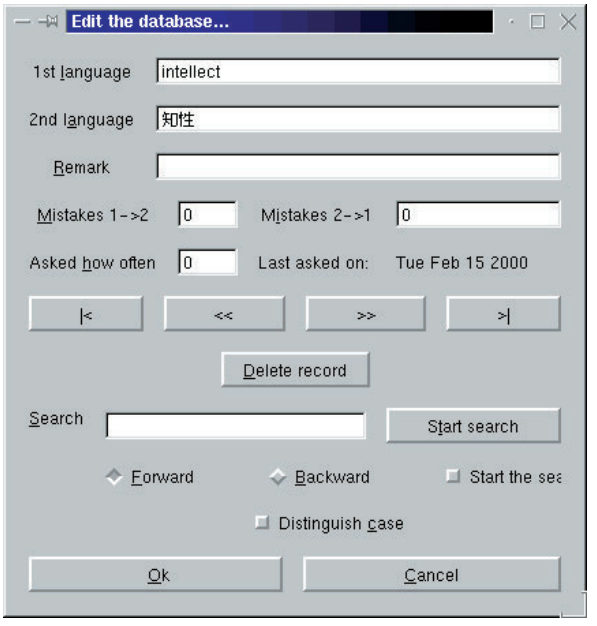
データベースに単語を登録するには、[Input]ボタンを押して入力用ウィンドウに切り替える（**画面2**）。[1st/2nd



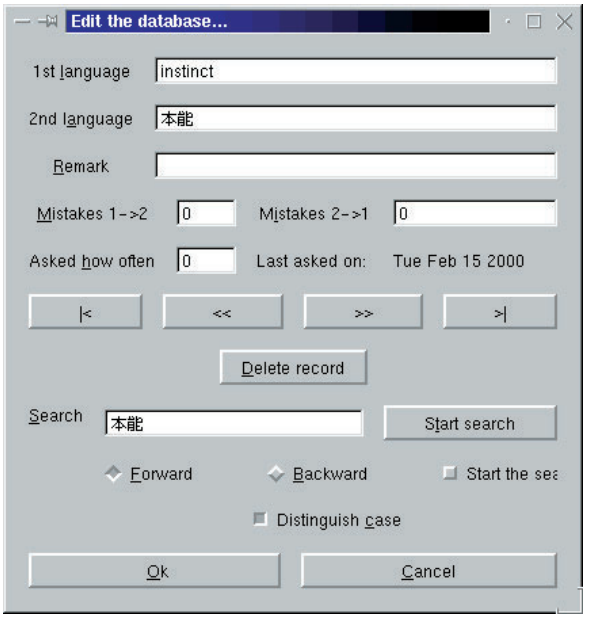
画面1
データベースを作成する。複数のデータベースを切り替え可能だ。



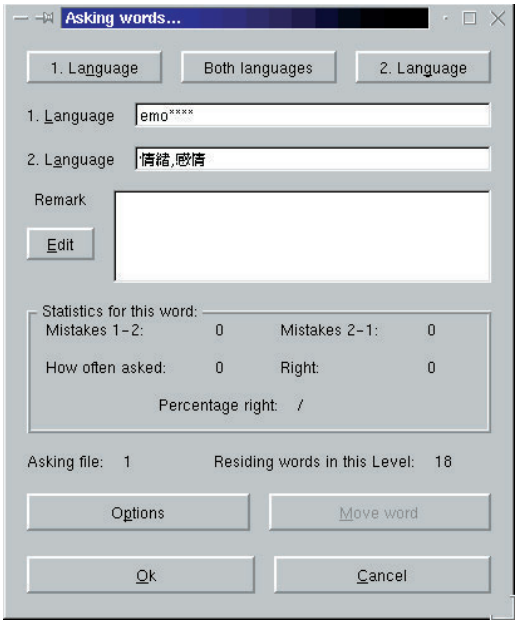
画面2
覚えたい対になる言葉を入力。日本語も問題なく使える。



画面3
登録したレコードの内容を編集する。正解数などの修正も可能。



画面4
検索文字列が日本語の場合は大・小文字を区別する設定で行う。



画面5
テストを実行中。「*」で隠された言葉をキー入力する。

language]の両フィールドには、対になる言葉（たとえば「英単語・熟語」と「訳語」など）を入力する。どちらのフィールドにも、複数の単語をカンマで区切って並べられる。日本語化されたKDE／Qtであれば、日本語の入力も問題ない。

[Add]ボタンを押すと、現在の設定内容をレコードとして登録し、次の入力に移る。入力作業を終了するには[Done]ボタンを押せばいい。

●単語の編集と検索

登録したレコードを編集するには、[Edit words]ボタンを押して編集用ウィンドウに切り替える（**画面3**）。なお、編集はファイル単位で行われるので、下から2番目のボタンを利用して、あらかじめ編集したいファイルに切り替えておこう。

編集用ウィンドウには、入力時の設定内容や、過去の質問数・不正解数などの情報が表示され、それぞれの内容を修正できる。このほか、[<<][>>]ボタンで前後の単語に切り替えたり、[Delete record]で現在のレコードを削除することも可能だ。

また、ファイル内のレコード数が多くなると、[<<][>>]ボタンだけで目的のレコードを見つけるのは大変だ。そこで、このウィンドウにはレコードの検索機能も用意されている。

[Search]に検索文字列を入力して[Start search]ボタンを押すと、[1st/2nd language]の内容の一部に検索文字列が含まれるレコードが表示される（**画面4**）。なお、日本語は[Distinguish case]（大・小文字を区別する）のチェックが外れていると正常に検索されないので気をつけよう。

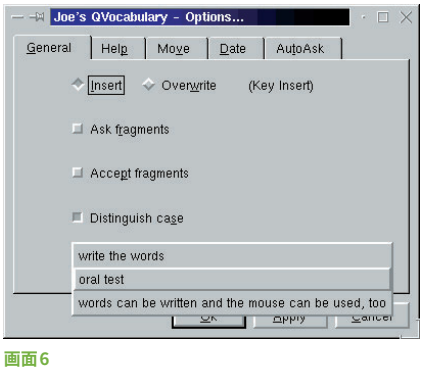
●単語のテスト

ある程度単語を登録したら、いよいよテストを実行しよう。[Ask words]ボタンを押すと、テスト用ウィンドウに切り替わる（**画面5**）。まずは、上部の3つのボタンで、質問と答えに使われる内容を指定する。

たとえば、[1. Language]ボタンを押すと、[2nd language]の内容が質問され、[1st language]の内容を答えるテストが行われる。[2. Language]ではその逆、[Both languages]では両方が混在したテストになる。

いずれにせよ、その下の2つのフィールドの一方に質問が表示されるので、もう一方にキーボードを使って答えを入力すればいい。文字数は「*」で表示される（変更可能）。

このほか、[Options]ボタンで開く設定ダイアログ（**画面6**）では、キー入力する代わりに口頭で述べる（左右ボタンで正解・不正解を指定）、カンマ区切りの内容を分割して質問・答えに利用する、最新質問されていない単語だけに限定する、完全に覚えた単語を別ファイルに移動・コピーするなど柔軟な設定が可能だ。



画面6
設定ダイアログにより細かなカスタマイズが可能だ。

不要なファイルを検索して削除やアーカイブを行う

Kleandisk

バージョン：1.0.0　　種別：GPL

http://www.casema.net/~buursink/kleandisk/

Kleandiskは、異常終了したソフトが作成するcoreなど、一般的には不要とされるファイルをディスク上から検索し、クリーンナップ処理を行うソフト。検索条件の異なるファイルグループを複数登録して、さまざまな種類のファイルを分類できる。クリーンナップ処理は、単に削除するだけでなく、アーカイブや移動して保存し、後で復活させることも可能だ。動作にはKDE／Qtが必要となる。

●ビルドとインストール

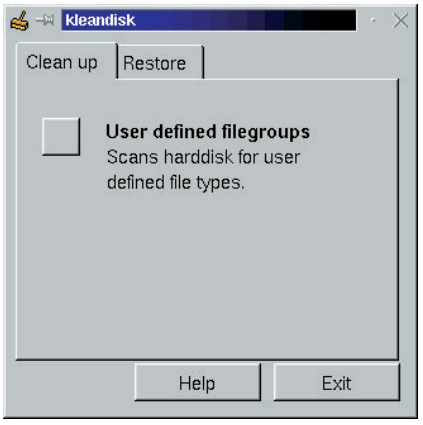
Kleandiskは、ファイル一式をtar＋gzipしたtarボールで配布されている。ビルドとインストールの手順は、「./configure」→「make」→「make install」という一般的なものだ。

なお、Red Hat 5.2系のVine 1.1などegcsのバージョンが古い（1.0.3aなど）ディストリビューションではビルドに失敗する。これは、kleandisk/abprocess.hの「stderr」（38行目）と「stdout」（45行目）が原因だ。これらを別の名前に変更し、ソースファイルで参照している部分も同様に修正すればいい。

これらの処理をまとめて行うパッチ（kleandisk-1.0.0-oldegcs.patch）を用意した。Vine 1.1などでは、tarボールの展開先で、

```
$ patch -p1 < kleandisk-1.0.0-oldegcs.patch
```

としてパッチを当てた後、上記の手順でビルドすればいい。



画面1
起動時に開くウィンドウ。クリーンナップかリストアを選択する。

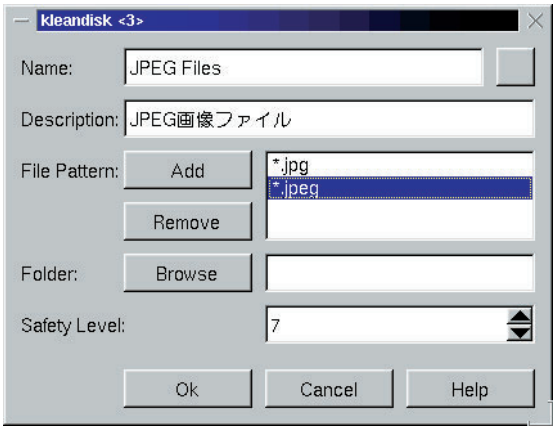
●ファイルを検索する

「kleandisk」として起動すると、クリーンナップとリストアのどちらを行うか選択するウィンドウが開く（**画面1**）。リストアはクリーンナップ後のファイルが対象となるので、まずは[Clean Up]ページの四角いボタンを押そう。

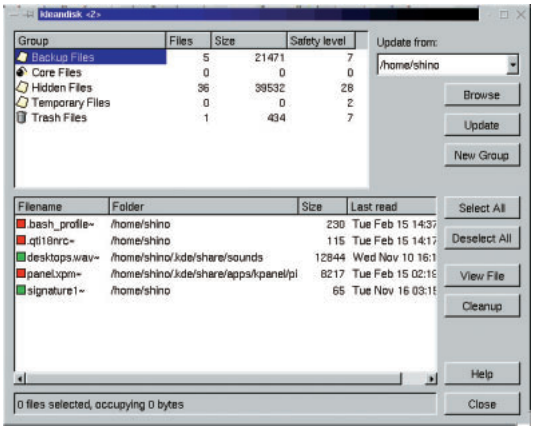
はじめて起動したときには、デフォルトのファイルグループを設定するか聞かれるので、[Yes]ボタンを押す。すると、クリーンナップ用のウィンドウに、「Backup Files」や「Core Files」などのデフォルトファイルグループが表示される。

それぞれのファイルグループは、検索時に使われるファイル名のパターンを持っている（複数可）。たとえば、「Backup Files」なら「*~」、「Core Files」なら「core」といった具合だ。[New Group]ボタンを使ってユーザーがファイルグループを追加したり（**画面2**）、右クリックメニューの[Edit]により既存のファイルグループの設定を変更することも可能だ。

ファイルグループに属するファイルをディスク上から検索するには、まず左上の[Update From]に検索開始ディレクトリを入力する（初期値はユーザーのホームディレクトリ）。続いて、[Update]ボタンを押すと、検索開始ディレクトリ以下のファイルに対して検索が行われ、それぞれのファイルグループに属するファイル数や合計サイズ



画面2
ファイルグループの登録ではファイル名のパターンなどを設定。



画面3
指定したファイルグループに属するファイルの一覧が表示される。

が表示される。

●不要なファイルを処理する

検索後に、適当なファイルグループをクリックすると、そのグループに属するファイルやディレクトリの詳細な情報が下のフィールドに表示される（**画面3**）。ファイル名をクリックしてから[View File]ボタンを押すと、KEditなどを使って閲覧可能だ。

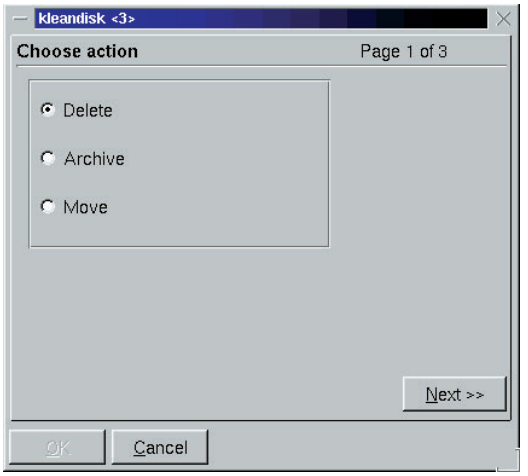
クリーンナップを行うには、まず各ファイルの左端に表示されている四角をクリックして、選択した状態にする（[Select All]ボタンによる一括選択も可能）。

続いて、その下の[Cleanup]ボタンを押すと、処理の内容を設定するウィンドウが開く（**画面4**）。設定は3ページ構成のウィザード形式で、最初のページでは、実行する処理の種類を「削除」「アーカイブ（圧縮も含む）」「移動」の中から選択する。

次のページでは、各処理の詳細を設定する。削除の場合は、バックアップ作成の有無とバックアップ先のディレクトリ、アーカイブならアーカイブを作成するディレクトリ、移動なら移動先ディレクトリだ。

最後のページでは、対象となるファイル数や合計サイズといった処理の内容が表示され、[OK]ボタンで処理が実行される。処理結果のログを見ることも可能だ。

画面4
クリーンナップ処理の設定はウィザード形式で行われる。



●クリーンナップファイルの復活

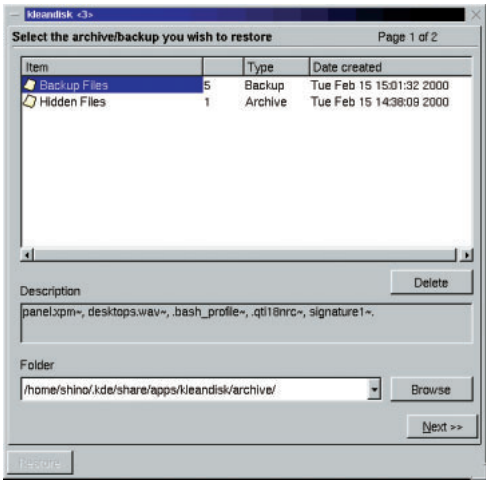
Kleandiskでクリーンナップ処理したファイルは、バックアップなしで削除した場合を除いて、いつでもリストア可能だ。最初のウィンドウ（**画面1**）を[Restore]ページに切り替えて四角いボタンを押し、リストア用ウィンドウを開こう（**画面5**）。

このウィンドウには、過去に削除・アーカイブ・バックアップしたファイルが「Backup Files」などのグループ名で一覧表示される。複数のファイルを一括処理した場合、それらはひとつにまとめられている。

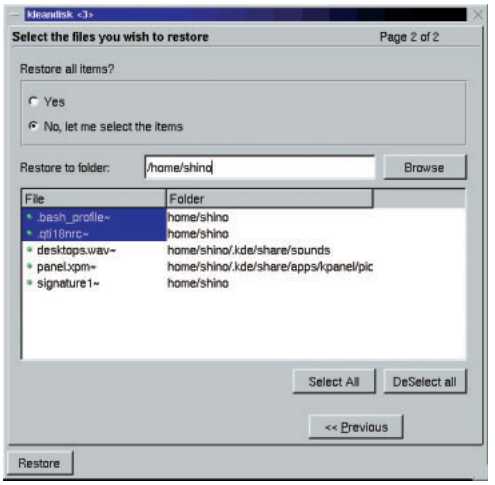
これらのグループ名をクリックすると、そこに含まれるファイルの一覧が下のフィールドに表示される。保存しておく必要がなければ、[Delete]ボタンを押して削除してしまえばいい。逆に、リストアしたいファイルを見つけたら、[Next>>]ボタンで次の画面に切り替えよう（**画面6**）。

この画面では、含まれている全ファイルをリストアの対象とするならば[Yes]、各ファイルを個別にリストアする場合は[No, let me select the items]をチェックして、リストアするファイルを選択する。

最後に、リストア先ディレクトリを[Restore to folder]に入力後、左下の[Restore]ボタンを押すと、実際にファイルが作成される。



画面5
クリーンナップしたファイルはアーカイブ状態で保存されている。



画面6
リストアしたいファイルを個別に選択することも可能だ。

Webページを取得するwgetのフロントエンド

KWebGet

バージョン：0.3　　種別：GPL

http://www.kpage.de/en/kwebget/
http://www.gnu.org/software/wget/wget.html（wget）

KWebGetは、WebサイトやFTPサイトのファイルを一括取得する「wget」のフロントエンドだ。取得を開始するURLをはじめ、リンクをたどる数や、対象とするドメイン、ファイルタイプなどの設定を、複雑なコマンドラインオプションの代わりに、わかりやすく分類されたパネルで行える。また、必要最小限の設定のみ対話的に行うウィザードモードも用意されている。動作にはKDE／Qtが必要だ。

●ビルドとインストール

KWebGetは、ファイル一式をtar＋gzipしたtarボールで配布されている。「./configure」→「make」→「make install」という一般的な手順でビルドとインストールが可能だ。

なお、KWebGetでは、ファイルの取得はwgetを実行することで実現しているため、wgetなしでは正常に動作しない。wgetがインストールされていない場合は、本誌付属CD-ROMやGNUのFTPサイト（国内ではftp://core.ring.gr.jp/pub/GNU/wget/など）からtarボールを入手し、KWebGetと同様の手順でビルドとインストールを行おう。

このほかの作業として、ホームディレクトリに、取得したファイルが格納されるKWebGetProjectsディレクトリ（変更可能）を作成しておこう。

●設定が簡単なウィザードモード

ktermなどのコマンドラインで「kwebget」として起動すると、最初にKWebGetの2つの実行モードを選択するウィンドウが開く（**画面1**）。まずは初心者向きの「KWebGet-Wizard」（ウィザードモード）から説明しよう。

ウィザードモードでは、

- **保存先ディレクトリ**
- **wgetの動作（ミラーリングなど）**
- **取得開始URL**
- **リンクを追う数と最大取得量**
- **再試行の回数**

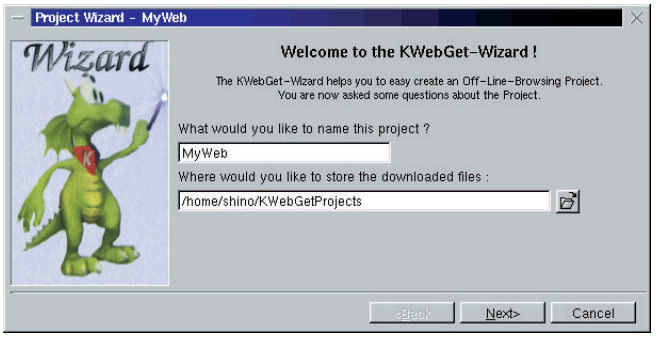
という基本項目について、順を追って対話的に設定する（**画面2**）。設定中は[<Back][Next>]ボタンで前後の画面へ切り替えられ、以前の設定を変更することも可能だ。

最後の画面で[Start-Now]を選択して[Finish]ボタンを押すと、wgetが実行され、ファイルを取得している様子がウィンドウに表示される（**画面3**）。取得が終了したら、保存先ディレクトリ（初期値は~/KWebGetProjects）を見てみよう。URLを反映したディレクトリツリーにHTMLファイルなどが作成されているはずだ。

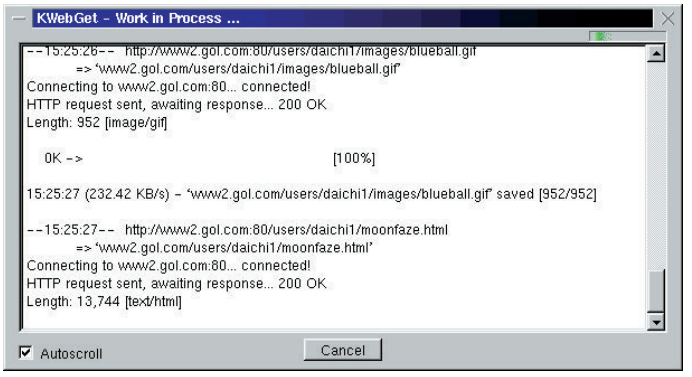
取得するファイルタイプを限定するなど、さらに細かな設定を行いたい場合は、[Advanced-Setup]を選択して[Finish]ボタンを押し、現在の設定を引き継いで後述のアドバンストモードに移行する。

また、とりあえず設定だけ行い、ファイルは後で取得したい場合は、[Save&Exit]を選択して[Finish]ボタンを押し、設定を「プロジェクト」としてファイルに保存しよう。保存したプロジェクトは、アドバンストモードで読み込める。

画面1
起動後に2つの動作モードのいずれか一方を選択する。



画面2
最小限の設定を順を追って行うウィザードモード。



画面3
wgetの実行中の様子は独立したウィンドウに表示される。

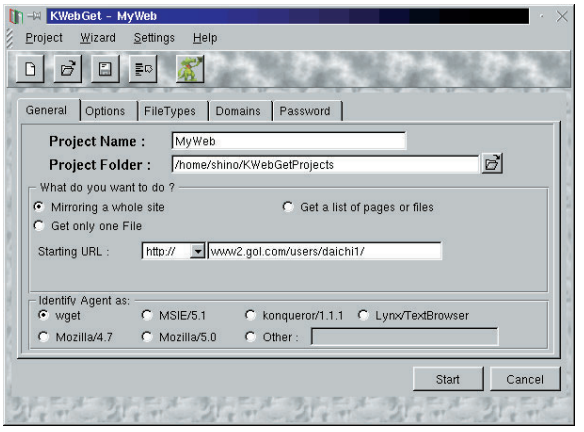
●詳細設定はアドバンストモードで

起動時のウィンドウ（**画面1**）でアドバンストモードを選択するか、ウィザードモードで[Advanced-Setup]を選択して終了すると、wgetの詳細な設定を行うためのアドバンストモードのウィンドウが開く。

設定項目が多岐にわたるため、ウィンドウはジャンル別にページ分けされている。これらの設定は「プロジェクト」としてファイルに保存し、再利用可能だ。[Start]ボタンを押すとwgetが起動する。

以下では、ページごとの主要な設定項目について説明しよう。

画面4
ジャンル別にページ分けされたウィンドウで設定を行う。



・[General]ページ（画面4）

取得を開始するURL（Starting URL）や、保存先ディレクトリ（Project Folder）の設定のほか、wgetの基本動作の選択などを行う。リンクを再帰的にたどってWebページを取得したり、FTPサイトをミラーリングする場合は、[Mirroring a whole site]をチェックしておけばいい。

・[Options]ページ

リンクをたどって複数のWebページを取得するには、[Get files recursive]をチェックし、[Number of Levels]にリンクを追う数（0で無制限）を設定する。既存のファイルを上書きするかどうかなども設定可能だ。

・[Filetypes]ページ（画面5）

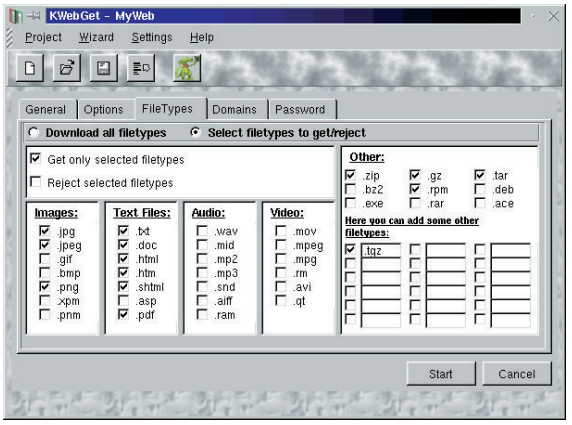
GIF画像は取得しないなど、ファイルタイプ（拡張子）による選別を行うには、[Select filetypes to get/reject]を選択してタイプの設定を行う。取得するタイプを指定する場合は[Get only selected filetypes]、無視するタイプを指定する場合は[Reject selected filetypes]を選択し、目的のタイプにチェックを付ければいい。代表的なタイプは最初から用意されているが、ユーザーが新たなタイプを追加することも可能だ。

・[Domains]ページ（画面6）

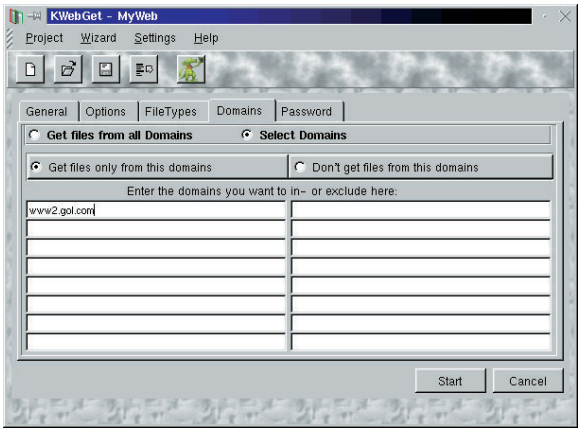
ファイルを取得するドメインを限定するには、[Select Domains]を選択してドメイン設定を行う。同一ドメイン内のリンクのみ追いたい場合は、[Get files only from this domains]を選択し、取得開始URLと同じドメインのみ登録すればいい。逆に、広告サイトなど特定のサイトへのリンクを無視するには、[Don't get files from this domains]を選択して、無視するサイトのドメインを登録する。どちらも複数ドメインの登録が可能だ。

・[Password]ページ

アクセスにユーザー名とパスワードが必要なWebサイトや、匿名（anonymous）ログインを許さないFTPサイトにアクセスする場合は、このページでユーザー名とパスワードを設定し、対応するチェックボックスをチェックする。同様にプロキシに対するユーザー名とパスワードも設定可能だ。



画面5
取得する（あるいは無視する）ファイルタイプを細かく設定できる。



画面6
ファイルを取得する（あるいは無視する）ドメインも設定可能だ。

3次元空間でインベーダーの侵略を防げ

XInvaders 3D

バージョン：1.31　　種別：GPL

http://www.fiu.edu/~dllopi01/xinv3d.htm

XInvaders 3Dは、昔懐かしいインベーダーゲームの3D拡張バージョンとも言うべきシンプルなシューティングゲーム。インベーダーやUFOなどの登場キャラは同じだが、3次元空間でのプレイのためゲーム感覚は大幅に異なる。初めのうちは、上下左右に移動しながら、インベーダーにミサイルを当てたり、インベーダーからのミサイルを避けたりするだけでも難しいかもしれない。

●ビルドと画面構成

XInvaders 3DはtarボールとRPMパッケージの両方が配布されている。バイナリパッケージはRed Hat 6.x用なので、その他のディストリビューションでは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しよう。

「rpm --rebuild xinv3d-1.31-1.src.rpm」とすると、/usr/src/redhat/RPMS/i386にxinv3d-1.31-1.i386.rpmが作成されるので、これをインストールすればいい。

また、tarボールからのビルドも簡単で、単に「make」とするだけだ。インストーラは用意されていないので、xinv3dを/usr/local/gamesなどに手動でコピーすればいいだろう。

起動すると、昔懐かしいベクターグラフィックス風の文字でタイトルが表示される（**画面1**）。スペースキーを押すとプレイ開始だ。画面手前に赤く表示されているのが自機で、その向こうには、左右に移動しながら手前に接近してくるインベーダーや、画面を横切るUFOがいる（**画面2**）。

●3D空間でインベーダーを倒そう

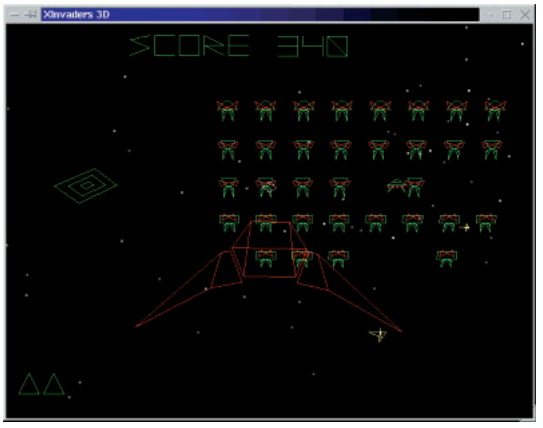
自機はカーソルキーで上下左右に移動する。ミサイルを撃つにはスペースキーを押せばいい（連射不可）。うまく当たらないときは、画面に表示されている◇形の照準を参考にしよう。なお、ミサイルはある程度自機の動きに追随するようだ。

インベーダーは上の列ほど得点が高い（下から順に10/50/100/150/200点）。下の列から順番に倒す必要はなく、上部のインベーダから狙えばいい。また、UFOは500点と高得点なので逃さないようにしたい。

インベーダーのミサイルに接触すると自機が破壊され（**画面3**）、3機失うとゲームオーバー。もちろん、インベーダーが画面手前に到達してもゲームオーバーだ。視点の関係か敵のミサイルが見難く、最初のうちはすぐにゲームオーバーになるかもしれない。

このほか、Pキーを押すとプレイを一時中断でき、Qキーでタイトル画面に戻る。ゲームの終了はEscキーだ。また、Fキーでフレームレートを表示させることもできる。

画面1
ベクターグラフィックス風のタイトル画面。



画面2
画面奥から迫り来るインベーダーを破壊せよ。

画面3
インベーダーのミサイルに当たると、無惨にもこのように…。

36種類の石を配置するパズルゲーム

Kishido

バージョン：1.0.1　　種別：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（オリジナルの石道はPC-8801シリーズ用で1990年4月にアスキーから発売）。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目的だ。上下左右を石で囲まれたマスほど石を置くための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、ファイル一式をtar＋gzipしたtarボールのみ配布されている。ビルドとインストールは「./configure」→「make」→「make install」という一般的な手順をとる。インストールにより、実行ファイルやデータファイル、HTML形式のマニュアルなどが/usr/kde以下の適切なディレクトリにコピーされる。

●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、ボードの四隅にひとつずつ、中央に2つの石が置かれている。

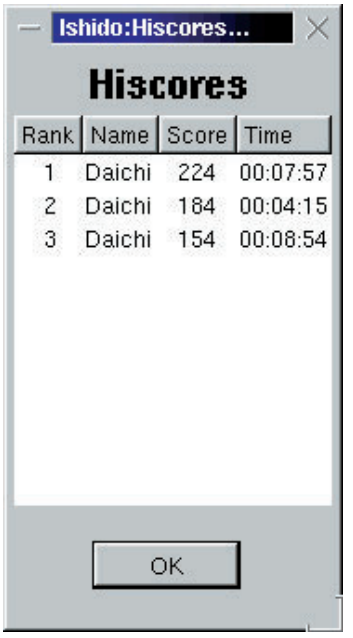
画面右上に次の石が表示されているので、その置き場となるボード上の空きマスをクリックしよう。ただし、石を置けるのは、少なくとも1つの石がすでにその周囲（上下左右の4マス）に置かれているマスだ。また、新たに置く石のマークと色は、以下の条件を満たしている必要がある。

・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

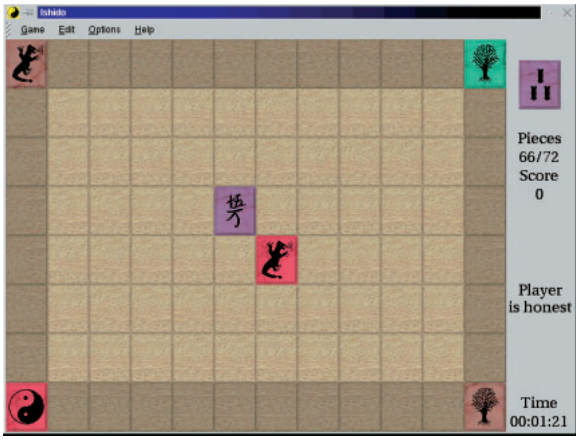
周囲の石が多いほど条件が厳しくなるが、その分得点も高い。周囲の石が1つだと1点、2つなら5点、3つなら20点、4つだと100点も入る。

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（使いすぎるとハイスコアが記録されない）。

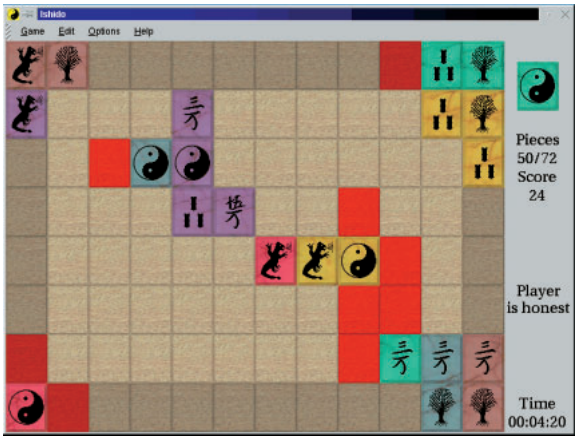
すべての石をボードに置くか、石を置ける場所がなくなるとゲームオーバーだ。ハイスコアには、得点と所要時間に加えてプレイヤー名も登録できる（**画面3**）。



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（使いすぎに注意）。

隠喩としてのコンピュータ

ポストモダンにおける閉塞

文：豊福 剛
Text : Tsuyoshi Toyofuku

シェリー・タークルは『接続された心』(“ Life on the screen ” Sherry Turkle著、日暮雅通訳、早川書房刊）の中で、IBM PCユーザーとMacintoshユーザーへのインタビューを基に、両者のコンピュータ観の違いを提示している。そして、PCとMacintoshにおける「透明性」の意味の違いを、それぞれモダンとポストモダンに対応させて解釈している。

――モダニストには、コンピュータは、透明で、明瞭な構造をもった宇宙に見える。自明な規則に従って機能する宇宙は、プログラミングによって主体的にコントロールすることができる。逆に、日常の環境は、不透明であり、未知の規則に従って機能する捉えがたい制度に満ちていると感じられる。
――ポストモダニストには、コンピュータは、明瞭な普遍的規則に従っては機能していない、日常の環境によく似た、不透明な宇宙に見える。スクリーンの背後にある機械は、見えないところへ後退してしまい、そこに直接手出しすることはできない。コンピュータの機能を把握するための努力や問いかけそのものは、どうでもよくなり、スクリーンを素朴に信頼し、不透明なテクノロジーに身を委ねる。試行錯誤によって、徐々にやり方を見つけていくしかない状況に投げ込まれている。

GUIはポストモダンである

ただし、タークルのインタビューは80年代のユーザーを対象にしたものであり、ここで紹介されているPCユーザーがWindows以前のユーザーである点は補正しておく必要がある。90年代以後の文脈においては、WindowsによってGUIが全面化したPCはMacintoshと等価であり、ポストモダン的ととらえることができるだろう。そういう意味では、タークルが指摘した80年代のPCにおける「透明性」は、そのユーザーインターフェイスから見ても、Linuxに代表されるフリーソフトウェア運動が継承していると解釈できる。

とはいえ、LinuxはGUIを否定しているわけではなく、むしろMacintoshのような透明なインターフェイスの実現も目指している。その意味では、Linuxはポストモダン的要素も取り入れようとしているモダニズムであり、両義的といえるだろう。将来、LinuxにおいてMacintoshと同等、あるいはそれ以上のGUIが実現されたとき、Linuxもやはりポストモダンにのみ込まれてしまうのだろうか？

タークルの論点に即するなら、GUIを実現したコンピュータは、必然的にポストモダンといえる。GUIが実現する「シミュレーショニズム」こそポストモダンの本質だからだ。

GUIの背後を問わないポストモダン的な態度を、タークルは「インターフェイス・バリュー」と形容した。また、ポストモダンにおいては、近代以前の具象的で感覚的な「野生の思考」への回帰が見られると指摘したうえで、スクリーンに発現する具象的なインターフェイスの外見と感覚的な戯れに浸ることを、「野生の思考」への回帰になぞらえている。

ポストモダンが、スクリーンの背後を問わない態度だとすると、あえてGUIの背後を問う態度にこそモダニズムが可能であるはずだ。

コンピュータにおける「現実界」とは

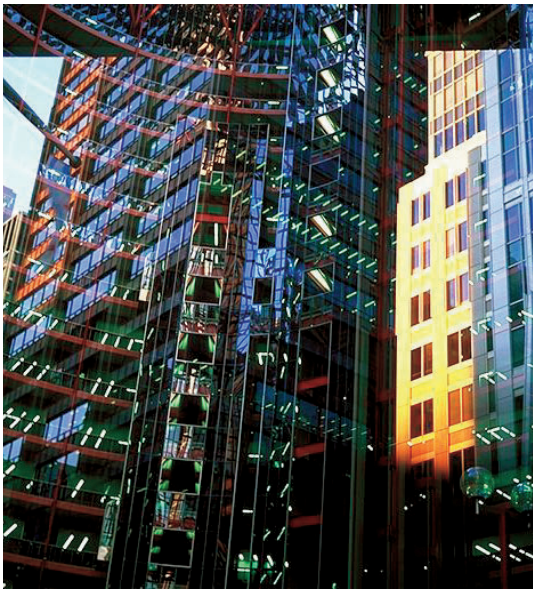
スロヴェニアの哲学者で、ラカン派マルクス主義者でもあるスラヴォイ・ジジェクは、『幻想の感染』(“ The Plague of Fantasies ” Slavoj Zizek著、松浦俊輔訳、青土社刊）で、タークルのインターフェイス論を批判的に検討している。

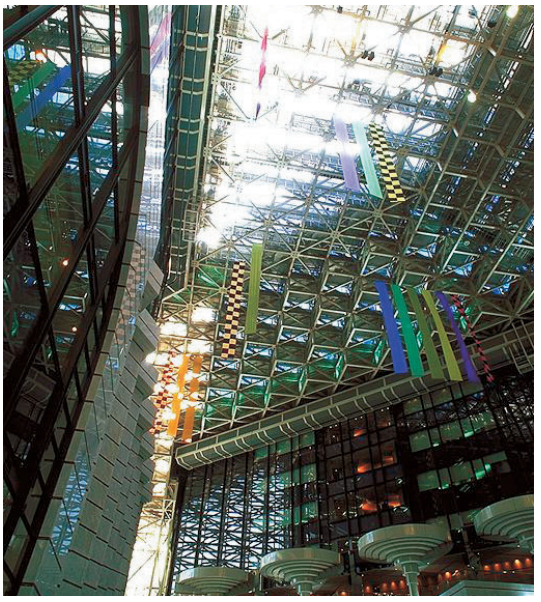
――スクリーン上での外見の戯れにユーザーが浸れるのは、GUIの背後に「現実界」があるからである。スクリーン上に発現する外見は、「現実界」の結果なのである。インターフェイス・バリューにおいては、「現実界」の存在が括り出されているのだ。

ジジェクがここで使っている「現実界」とは、フロイトの流れをくむ精神分析の思想家ジャック・ラカンの用語である。ラカンは、精神分析の場が、象徴界、想像界、現実界という3つの基本領域に区別されると考えた。誤解を恐れずに単純化すると、象徴界は言語の領域であり、想像界は映像の領域である。ただし、現実界は現実の領域ではない。本能が壊れた動物である人間には、現実を直接把握することはできないからだ。現実は、まず映像（想像界）として把握され、その映像が言語（象徴界）によって構成されることによってはじめて、現実として把握される。しかし、象徴界は完全なものではありえないので、象徴界に還元できない残余があることになる。この象徴化しきれない残余の領域を、ラカンは「現実界」と考えた。

それでは、ジジェクがここで使っている「現実界」の意味は、どのように解釈されるだろうか？

GUIでは、ユーザーはスクリーン上の外見を映像として把握する（想像界）が、それらの映像は象徴としてユーザーは理解する（象徴界）。たとえば、ユーザーはアイコンという図像をクリックするとアプリケーションが起動することを理解している。しかし、スクリーン上の外見は、その背後にある





プログラムを投射したものであり、アイコンの意味も、プログラムによって規定されている。そして、この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以上、それは言語であり「象徴界」に属するように思える。

このように考えると、コンピュータにおいては、スクリーン上の図像を構成する象徴体系と、スクリーンの背後にあるプログラムそのものを記述する象徴体系であるプログラミング言語という、2種類の「象徴界」があることになる。

このように、ラカン理論をコンピュータに拡大適用しようとすると、どうにも無理が生じてしまう。それは、映像と言語の関係が、人間とコンピュータでは、ちょうど逆になっているからではないだろうか？　人間は、現実を映像として、映像を言語として認識する。これに対して、コンピュータでは言語が映像を投射する。

ジジェクは、なぜプログラムを「象徴界」ではなく「現実界」に位置づけたのだろう？　それはインターフェイス的主体にとっては、スクリーン上のアイコンなどの象徴（シンボル）が「象徴界」のすべてであるからだ。「象徴界」と「現実界」の区分は、ラカン的な意味とは違って、絶対的なものではなく、コンピュータに対する態度によって異なることになる。モダニストにとっては、スクリーンの背後は「象徴界」であるが、ポストモダンにおけるインターフェイス的主体にとっては、それは「現実界」なのだ。

インターフェイス的主体にとっては、スクリーンの背後は二重に括り出されていることになる。プログラムが「象徴界」から「現実界」に括り出され、さらに、「現実界」そのものが括り出されるからだ。

ここで重要なのは、GUIの背後にあるプログラムのソースコードにアクセスできるかどうかである。ソースコードにアクセスできれば、ブラックボックスとしての「現実界」ではなく、プログラミング言語によって構成された「象徴界」として把握することが可能である。プログラムが「現実界」に括り出されたのは、GUIによって必然的にそうなったのではなく、現在のGUIの多くが、プログラムのソースコードにアクセスできないからではないだろうか？

電脳空間を支えるイデオロギー

スクリーンの背後にある「現実界」を、ソースコードの独占的アクセスと関連づけることで、ジジェクのポストモダニズム批判の政治的意味が明確になるだろう。ジジェクは、GUIだけでなく、電脳空間全体が「現実界」の括り出しによって支えられていると論じているのだ。

たとえば、ビル・ゲイツがあるインタビューで、電脳空間を「摩擦のない資本主義」への展望を拓くものとして祝福していたことを引いて、次のように書いている。

――この表現には、電脳空間資本主義のイデオロギーを支える社会的幻想が完璧に込められている。物質的惰性の最後の痕跡が消滅するどこまでも透明で、エーテル※1のような交換の媒体という幻想である。見逃してならない枢要な点は、「摩擦のない資本主義」という幻想においてなくなる「摩擦」が、いかなる交換の過程をも支えている物質的障害の実在性のことだけを言っているのではなく、何よりも、社会的交換の空間に病的なねじれのしるしをつける、トラウマの元になる社会的拮抗、権力関係などの「現実界」のことを言っているのだという点である。

資本主義イデオロギーだけではなく、民主主義イデオロギーもまた、電脳空間の幻想を支えている。ジジェクは、電脳空間民主主義のイデオロギーを「中心のない網状構造によって個人個人が群れ、参加的な草の根政治システム、不可侵の官僚的国家部局の神秘が退けられる透明な世界を築くことができる」とする社会的幻想としたうえで、ここでも「現実界」の括り出しを指摘する。

――距離が停止し（テレビ会議で地球上のどことも即座にやりとりができるようになる）、文章から音楽からビデオまで、あらゆる情報がこちら側のインターフェイスで即座に利用できるようになる。しかしこの私を遠くの外国人から隔てる距離の停止の裏面は、他者との「現実の」身体的接触が徐々に消えていくせいで、隣人がもはや隣人ではなくなる。隣人はどんどん画面上の画像に置き換わり、誰にでも使えるということは、耐え難い閉所恐怖症を引き起こす。選択肢の過剰が、選べないという体験になる。誰でも直接に参加できる社会は、それだけにそこに参加できなくなっている人を否応なく排除することになる。

「果てしない可能性の未来を拓く電脳空間」という幻想は、「前代未聞の根源的な閉塞」を隠している。幻想の下に隠された「閉塞」は、存在しないのではなく、存在しているにも関わらず、あたかも存在しないかのようにみなされることを意味している。この「閉塞」は「現実界」の括り出しによる結果である。スクリーンの背後にアクセスできないことと、現実の他者との身体的接触が消えていくこととは、ポストモダンの症候として等価なのである。

※1　宇宙空間を満たしていると古代人が想像した霊気。Ethernetのイーサの語源が、このエーテルの意味であるのは興味深い

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。

アジアのおばさんたちのデータベースを作る話

文：安田幸弘
Text : Yukihiro Yasuda

世の中では、女はパソコンに弱いモノだということになっている。男と女のライフスタイルの違いや環境的な要因といった影響もあるのかもしれないが、しかしぼくはそんなもの、思い込みに過ぎないと思うのだ。とはいえ、思い込みだろうが何だろうが、当の女たちが集団ですっかりその気になっているもんだから、「女はパソコンに弱い」という怪しげな命題が、何だか既成事実化しているような気がする。

でもねえ、男だってわかんないヤツは本当に絶望的にわからない。女たちがちゃっかり「女はパソコンに弱い」と先に宣言してしまったもんだから、パソコンに弱いのは女、ということになっちゃったんじゃないだろうか。

ホントは女が良かった？

しかし、最近のようにインターネットがマトモに「使える」ようになってくると、「女はパソコンに弱い」とばかりも言っていられない。ちょっと前ならパソコンと聞くと逃げ出した女たちも、近頃はヘタをすれば男よりずっと積極的にパソコンを活用し始めている。ぼくが首を突っ込んでいるNGOのネットワークに、アジア女性資料センターという女性グループが参加しているのだが、このグループは最近、インターネットを使ってさまざまな情報交換を積極的に試みている。そんな女たちのネットワークのひとつにAWORCというグループがある。

AWORCは、フィリピンのISISという女性問題に取り組んでいるNGOを中心として、日本や韓国、マレーシア、モンゴルなど、主にアジア各国の女性情報をインターネットで共有、交換しようという国際ネットワークだ。いろいろな活動をしているのだが、そのひとつに、各国の女性に関する資料を詰め込んだオンラインデータベースを作ろうという話が進められている。

ところが、「女はパソコンに弱い」からかどうかわからないが、このデータベースのプロジェクトがなかなか進まない。そんなわけで、この2月にソウルで開かれたAWORCのワークショップで、ぼくが「オンラインデータベースをどう作るか」というレクチャーをすることになった。

もっとも、彼女たちとしては本当は女性の技術者にレクチャーをしてもらいたかったらしい。しかし、世の中はそんなに甘くない。性別を問わず、いまどき、まともなインターネットの技術者で、手間暇かけてNGOの支援なんかを手伝ってくれるような人材なんて、そう簡単には見つからない。結局、このデータベース・プロジェクトについて、当初から裏であれこれ動いていた関係で、「この際、男でもしょーがないか」という感じで引っぱり出されたというわけだ。もちろん、このデータベースはエンジンからインターフェイスまで、すべてオープンソースで固める予定だ。うまくいけば、今年の夏までには、それなりのものができあがるだろう。

インターネットは米帝国主義の陰謀？

現在、インターネット上にデータベースがどれぐらいあるのか、見当もつかない。たいていの情報はインターネットで見つけることができると言っても過言ではない。ただし、そこで提供されているほとんどの情報は先進国のものばかりであり、先進国の人々にとって意味のあるものばかりだ、という点は見落とされがちだ。彼女たちが自前のデータベースを作ろうとするのは、そんなネットの空白地帯を埋めようとする試みだ。

このデータベースは、英語の情報ばかりではなく、現地のローカルな言語で書き留められたさまざまな資料のデータベース化を目指している。いまでは日本をはじめとする先進国なら、自国語でパソコンを操り、ネットのコンテンツを作るのはあたりまえだが、タガログ語のコンテンツやらモンゴル語のコンテンツなんてほとんどないらしい。モンゴルの参加者が「モンゴル語のページもあるにはあるけど、せいぜい数千ページぐらいじゃないかなあ?」なんて言っていた。実際、インターネットは世界的な通信網のようでありながら、実は網の目が詰まっているのはG7の各国や一部の工業国ぐらいのもの。アジア、アフリカ、南米などの途上国では、首都クラスの大都市を除けばインターネット以前でさえある。

昨年、別のNGOの会議で、ある人が「インターネットは米国帝国主義者の陰謀だ。われわれ第三世界の民衆は、無条件でインターネットを受け入れるべきではない」というディスカッションがあった。そこまで言うかという気もするのだが、特別なドメイン名に何百万ドルもの値段がついたというようなニュースや、インターネットのガバナンスに関する議論、「ワンクリック」の特許、Eコマースだのバンキングだので盛り上がっているインターネットを考えると、現在のインターネットを米国の陰謀だと言いたくなるのももっともである。このままでは、インターネットを通して提供される膨大な情報は、ガチガチに知的所有権で固められ、ユーザーの動作は逐一監視されてマウスをクリックするたびに米国企業にいくばくかのライセンス料を払うことになりかねない。

さらに困ったことに、オープンソースの世界でも先進国、特に欧米の視点から自由ではない。日本人のLinuxユーザーなら誰でも、glibcのlocaleや、日本語化したKDEの不安定さなど、うっとうしい思いをさせられた経験があるはずだ。今回、計画しているデータベースも、US ASCIIの範囲でなんとかなるタガログ語やマレー語はともかく、日本語と韓国語の共存だの、キリル文字をベースとした特殊なキャラクタセットを使うモンゴル語のサポートといったマルチリンガルの部分がややこしいところ。オープンソースでは、商用ソフトのように言語一覧から使用言語をクリックすれば何語だって使えるというようなわけにはいかないからである。

それでもオープンソースにこだわるのは、高価なRDBMSや開発ツールが買えないという現実的な理由もあるが、コンピュータの基本の習得という意味合いもある。商用ソフトのシステムが使えるようになっても、それは特定のシステム、特定のバージョンに関する知識でしかない。いくらそんなものを覚えても、ツブシがきかないわけだ。もちろん、ユーザーレベルでWindowsのような商用ソフトを否定するものではないが、開発側が特定の商用ソフトに依存するのは好ましくない。'95年の北京女性会議では、女性のエンパワーメントが叫ばれ、コンピュータやインターネットはエンパワーメントのツールとして注目を集めた。しかし、できあいの商用ソフトで商用プロバイダに何かのページを作ったとしても、それがエンパワーメントだろうか。

彼女たちがパソコンに感じる本当の違和感は、先進国の男の文化なんじゃないかと思う。ぼくはぜひ彼女たちに、企業の論理から自由なオープンソースの意義を理解してもらいたいと思う。

そしていまのオープンソースの世界にも、彼女たちの視点を持ち込む努力が求められているのだろう。それは、バブルに浮かれている今のインターネットを正気に引き戻し、持続可能な世界に向けて踏み出す小さな一歩なのである。

●

ところで、ソウルのワークショップが終わり、みんなでビビンバを韓国風にバリバリとぶっかきまぜているとき、「ねえ、ユキヒロ?」と語りかけてきたのは、フィリピンのある女性資料センターの所長だった。

彼女は言った。「今日はバレンタインデーじゃない?」そういえばそうだ。その日は2月14日。昔はちょっとドキドキする日だった。

「バレンタインデーにこんなにたくさんの女性に囲まれて、ハーレムの王様になったみたいな気がするでしょう!?」

思わず、ぼくは居並ぶ恰幅のいいおばさんたちを見回してこう答えた。「その……、ハーレムの王様というよりは、アマゾン軍団に引っ捕らえられた男という気が……」

悪いけど、ぼくは彼女たちが踊るベリーダンスなんて、絶対に見たくない。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

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

年明けからCES、LinuxWorldと続いて海外取材が入ってしまったので、ちょっと厳しい日々が続いている。飛行機の中でバリバリ仕事をと思ったが、ノートパソコンは2時間ぐらいしか動かないし、周りをおばちゃんの団体旅行に囲まれてそんな雰囲気でもないのである。

それに、飛行機の中では、携帯電話でインターネットってわけにもいかない。どこか、席にEthernetを配線して、インターネット使い放題の航空会社は出てこないかしら。

マイクロソフトは忙しいようだ

本誌が出る頃にはすでにWindows 2000も出荷されていると思う。すでに、サービスパックの準備が進んでいるのだとか。内部流出文書によれば、6万以上の修正項目があるらしい。NTはSP3からという言い伝えもあるように、やっぱりしばらく待ったほうがいいのかも。

今回は、シングルバイナリで、サービスパックも全世界対応になるらしいが、それでも修正項目が減るわけでもないみたい。もっとも、SPを当てていきながら、OSの成長する姿を楽しむというのなら、すぐにでも導入してかまわないと思う。

さて、昨年、Windows 98の後継OSをNTカーネルにするのをあきらめたマイクロソフトだが、その後継OSの名前が「Windows ME」となったのだとか。Meとは、ピンクレディーを思い出させる名前だが、Millenium Editonの略である。Second（Edition）の次がMilleniumとは、飛びすぎという気もしないが、「WindowsはVer.3から」というジンクスを崩したいのだろう。

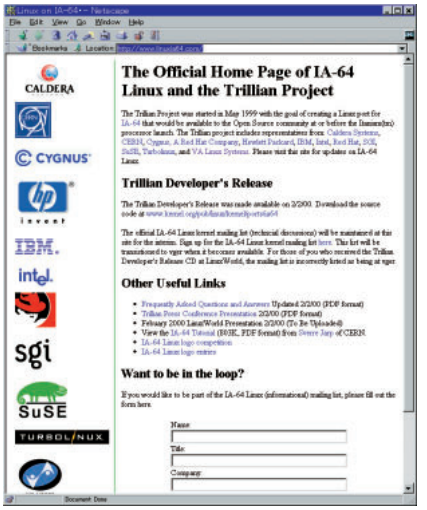
しかし、裁判に絡んで分割というウワサが流れたり、EUではWindows 2000が独禁法の調査対象になるなど、マイクロソフトを取り巻く状況はかなり厳しい。もしかしてGatesがCEOをやめたのは過激な発言をさせないためだったのかも。

インテルは、ちょっと厳しい

年末ぎりぎりに800MHzのPentium IIIを発表して、1900年代最後を最高速CPUで越えたインテルだが、早々にAMDの反撃を食らっている。なんかこの調子だと、夏までには1GHzにいきそうな気がするのだが。

しかし、この800MHzのPentium IIIだが、かなり品不足だという。ちょっと無理しているんじゃない？ という印象が拭えない。全体的に高クロック製品は品不足で、GatewayやDELLといったメーカーは不満が爆発という感じの発言が続く。

Gatewayに至っては、一度AMDは使わないと言ったにも関わらず、Athlon搭載マシンを発表するなど、急速なインテル離れを起こしている。かつてGatewayといえば、インテル製の



64ビットCPU用Linux開発プロジェクトTrillian Project（http://www.linuxia64.com/）

マザーボードだけを使う優良顧客だったのだが。

もう一方のDELLも不満はあるようだし、実際そういった発言もあるのだが、いまだにインテルの忠実な顧客である。よっぽどインテルにいい思いをさせてもらっているのか、それともAMDが嫌いなのか？ あるいは、Gatewayとインテルの関係が悪化するぶん、得するんでしょうか？

IA-64と呼ばれる次世代CPU、Itaniumの最低クロックは800MHzとなるらしいが、LinuxWorldでは、このIA-64用のLinuxが公開された。この64ビット CPUでは、インテルの一人勝ちも予想されているが、そのインテルがもっとも力を入れているのがLinuxである。マイクロソフトは当然やってくれるだろうし、ほかもそのうちついてくると踏んで、Red Hatに投資したり、このプロジェクト（Trillian Project）にも参加していたりする。そのせいかどうかはしらないが、同じくIA-64対応を表明していたSUNの機嫌が悪いのだとか。

Itaniumには、いまのところ有力な対抗馬が見あたらないのもあって、ハイエンド分野はインテルの独壇場となるか、それとも新しい勢力が台頭するのかが、ここしばらくの話題でもある。

そういえば、ウワサのTransmetaがようやく製品を発表。モバイル向けに消費電力を下げ、ソフトウェアでIA-32コード（つまりx86コード）を変換して実行するという。

インテルも直前にSpeedStepと呼ばれる、低消費電力化技術を搭載したCPUを発表しているが、このニュースの前にはちょっと色あせてしまった感じ。

このほかにも、各社は今、低消費電力に力を入れている。ひとつには、一般ユーザー向けには、これ以上どんどんCPU性能を上げても、なかなかうまくCPUパワーを使ってくれるアプリケーションがなく、売れるのはそれほど性能の高くない低価格ものばっかりという状況がある。そこで、技術をクロックを上げることより低消費電力化に向けようという動きがあるのだ。

まあ、デスクトップパソコンよりノートパソコンのほうがまだ高く売れるというのもあるが、家庭用の機器に使われるのに、ファンを組み込むのはメンテナンスなどの関係でちょっと大変という理由や、携帯電話が爆発的に普及して、これからはモバイルだという話なんかが背景にあるからだろう。

その中で、Transmetaは最初から低消費電力に対応したCPUであるCrusoeを発表した。もっとも、世間的にはx86コードの実行に目がいっているようだが、製品の価値としては性能と消費電力のバランスをとるLongRun技術のほうがあるような気がする。特に日本国内ではモバイルものが大人気だし、自宅で使うためにノートパソコンを買う人も大勢いる。しかし、現在では、重さを我慢してでっかいバッテリをつけるなどしないと、外出中に満足できるだけの時間動作するノートパソコンはなく、その意味では、こうしたCPUが登場することで、ほんとうに持ち歩いて使えるようになるのかも。

長時間動く高性能CPUがあったとして、そのうえでそれがx86コードを実行できるなら、既存のソフトウェア資産が利用できるというメリットが生きるわけで、単なるx86互換CPUなら、昨年、NSやIDTが互換CPUビジネスから撤退したように、楽な商売ではないのである。

これに対するインテルのSpeedStepだが、消費電力削減としてはまだまだ

インテルが発表したモバイル用消費電力低減テクノロジーSpeedStep
(http://www.intel.com/ebusiness/mobile/speedstep.htm)

という感じ。個人的には、300MHz以下にクロックを下げてもいいのではと思うのだが。

GEがまた抜かれる

GEといえば、エジソンにまでつながる米国の伝統的な企業だが、昨年マイクロソフトに米国1位の座を明け渡したと思ったら、今度は、ルータのCiscoにまで抜かれてしまった。もっとも、株価で計算した資産総額という面であって、1つの指標にすぎないのだが、伝統的な企業よりもインターネット、パソコンといった企業が米国の主力産業になりつつあるわけだ。

株といえば、インターネット関連株の次は、Linux関連株という話もあるが、みごとIPO（株式公開）を果たしたVA Linuxは、LinuxWorldの会場で、SlashDotなどを持つAndover.net（そういえば、ここもIPOの直後である）の買収発表を行って、ちょっとみんなびっくり。良いか悪いかは別として、なんかLinuxバブルという気がしないでもない。

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台

～Mozilla M13登場～

ソースコードを公開するという快挙により、Netscape社が一躍話題をさらったのは読者の記憶にあることだろう。そのNetscape Communicatorのフリー版といえるWebブラウザがmozillaだ。このほど新バージョンが登場し、日刊アスキー Linux上をにぎわした。　（日刊アスキー Linux編集部・吉川）

人気沸騰！

日刊アスキー Linuxでは、ディストリビューション関連の記事や、新しく登場したフリーのソフトウェアの記事の人気が高いが、最近特に人気が集中したのがMozilla Milestone13（以後M13）のニュースである。特に「『Mozilla 日本語パック』がM13に対応」（http://www.ascii.co.jp/linux/news/today/article/article370976-000.html）は、群を抜いて人気があったといっていい。Mozillaにかける期待が並々ならぬものなのだろう。それが日本語で使えるとなれば、人気が集中するのもうなずける。

ところで、M13はα版として位置付けられたものだ。実際にインストールしてみると、そこそこの安定性を得られている。試用環境はLASER5 Linux 6.0 Rel.2（Linux上では今のところglibc 2.1.xが必要となる）だ。まだ使い始めて間がないので、試したケースはたかが知れているが、いきなりフリーズしてしまうようなことは少ない。これまでに体験した不具合を挙げると、

1.残念でならないのだが、Linux版のM13では、日刊アスキー Linuxをうまく表示できない（しかもWin32版のM13は、メニューのJavaScript以外はきちんと表示する！）

2.「インターネットの検索」で日本語を入力語、検索ボタンを押すと落ちる

3.長い間（数時間）なにもいじらなかったら落ちていた、という事態が数回あった

というところだ。まだまだ実用レベルには至っていないが、お試しで遊ぶには満足できるといえるだろう。

そして、日本語パックも利用してみた。もともとMozillaそのものは、Webブラウザにせよメーラにせよ、日本語は通る。しかし、メニューは英語のままだ。そこで日本語パックの登場となる。Mozillaは、メニューやダイアログボックスのメッセージリソースを日本語に置き換えれば、きちんと日本語のメニューを表示してくれる仕様になっている。Mozilla日本語パックは、国内の有志の手によるもので、Mozillaに日本語メニューを表示させるパックである。

M13用の日本語パックが登場した当初は、所定のディレクトリに日本語パックをコピーする必要があったが、現在はMozillaと統合され、mozilla.org（http://www.mozilla.org/）から日本語版がダウンロードできるようになっている。これをダウンロードすれば、tarを展開→M13を起動といった2ステップだけで日本語版M13を使えるようになる。参考までに日本語版のM13を使うまでの手順を書いておくと、

1.mozilla.orgからtarファイルをダウンロードする（Linux用の日本語版は、約5.6Mバイト）

2.“ tar xvfz ファイル名 ”で適当なディレクトリに展開

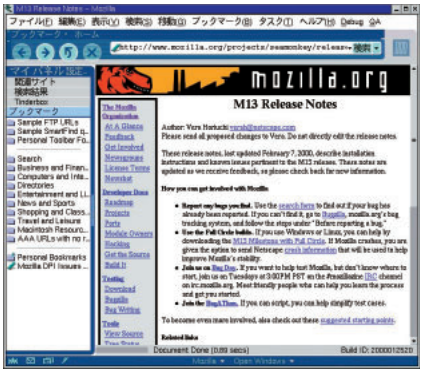
3./packageというディレクトリができるので、その中のmozillaを、./mozillaとして起動

これだけである。なお、Linux以外にも、Win32版やMacOS版も日本語版が用意されている。

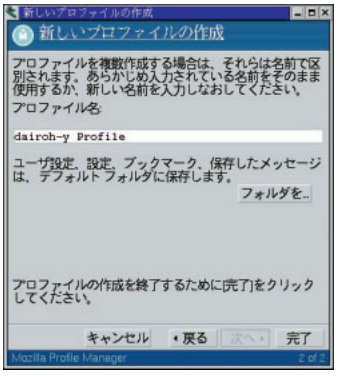
こうして、Mozilla日本語版が使えるのも、開発者の谷口氏、古川氏、山本氏といった国内の有志の努力の結果であり、深く感謝したい。現在、次のM14のために日本語化パックの活動をされている山本和彦氏の「Mozillaで嵐」（http://www3.tky.3web.ne.jp/~progre/mozilla/）のページに、Mozillaに関するメーリングリストや、国内のMozilla関連の情報/リンクがあるので、ぜひご覧いただきたい。

これがMozilla M13だ!

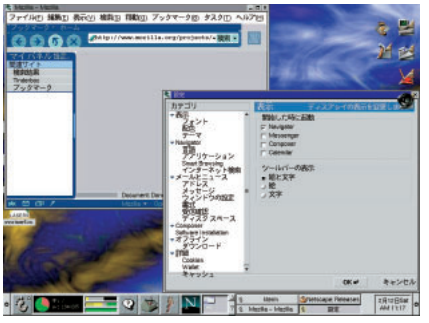
新しい機能が盛りだくさんのMozilla。インストール直後から便利な機能までを大紹介!



Mazillaのα版「Mozilla Milestone13」。LASER5 Linux 6.0 Rel.2にインストールした。一部のWebページが表示できないなどの不具合もあるが、とりあえず試用には耐えられる。特に、新しいユーザーインターフェイスを試したい、といった向きは楽しめるだろう。

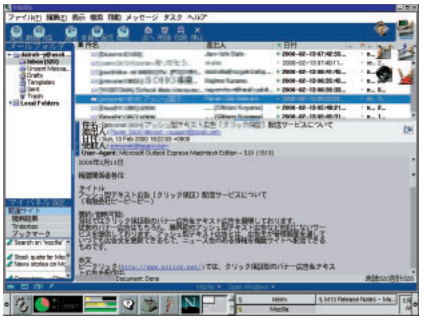


インストール後、Mozilla M13（Japanese）を最初に起動すると、プロファイル作成に入る。この過程を経るだけで、日本語版Mozilla M13を使うことが可能になる。

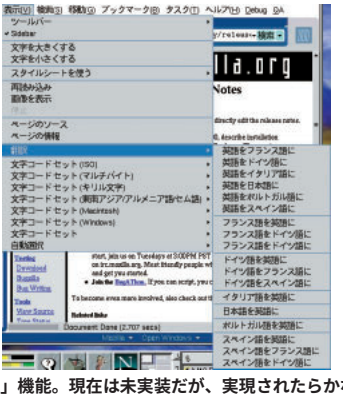


[編集]－[設定]で各種設定を行うところはNetscape Communicatorと同じ。

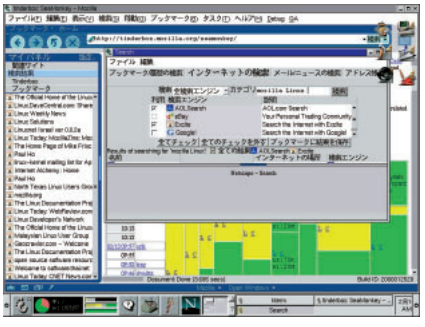
残念ながらLinux版のMozilla M13だと、日刊アスキーLinuxをうまく表示することができない（Win32版では問題ない）。



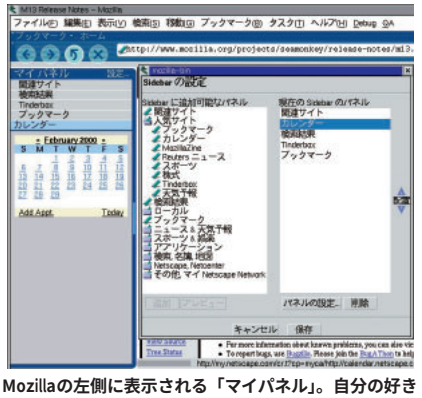
Mozilla M13のメーラ。英語版でもメールの日本語表示は可能だが、日本語化パックのため、メニューも日本語になっている。



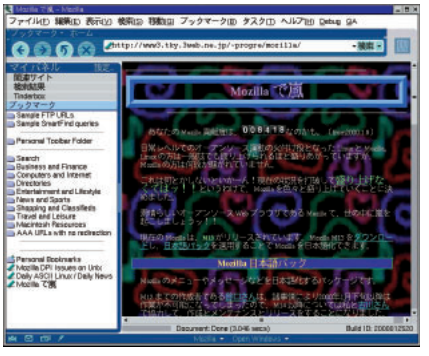
「翻訳」機能。現在は未実装だが、実現されたらかなりお世話になりそうな機能だ。



検索機能も多様に。「インターネットの検索」ひとつとっても、検索エンジンを指定したりと細かい操作が可能。検索結果は「マイパネル」に反映される。



Mozillaの左側に表示される「マイパネル」。自分の好きなパネルを登録できる。ここでは、「カレンダー」を登録してみた。ただし、このカレンダー機能は、Netscape Netcenterのアカウントがないと機能しない。



Mozilla日本語パック山本和彦氏の「Mozillaで嵐」のページ(http://www3.tky.3web.ne.jp/~progre/mozilla/)。国内の各種Mozilla系サイトへのリンクもある。

正しきタコへの道

初級Linuxer養成講座

第7回　ひとり管理者の心得（2）～バックアップはどうする？

Linuxの「システム管理者」として重要な「業務」のひとつとして、「ディスクのバックアップ」が必ず挙げられるのだが、個人で使うマシンでは、はたしてどこまでこの作業が必要になるのだろうか？ バックアップを行うにしても、どのようなコマンドを使って、どのようなメディアにバックアップすればよいのだろうか？ これらの疑問に対する明確な解答を見つけるのは容易ではない。

文：竹田善太郎
Text : Zentaro Takeda

マイクロソフトの「Windows 2000」もいよいよ発売となって、PC用OSの世界はますますにぎやかになってきた。マイクロソフト社の**世界征服の企て**の真偽はともかく、筆者の個人的意見として、Windows 2000は良いOSだと思う。もちろん、さまざまな不都合もないわけではないのだが、さすがに莫大な資金と膨大なマンパワー、そして数年間にもわたる開発とテスト期間を経てきた製品だけあって、とにかく安定して動くOSという印象がある。

しかし、Windows 98やWindows 95だって、登場した当時は「前のバージョンより安定していて使いやすい」という印象があったはずなのだが、どうもWindows系OSは**使い続けるほど不安定になる**という傾向があるようだ。その主な原因は、アプリケーションやツール類、あるいはOSのアップデートモジュールなどのインストールを繰り返すごとに、システムファイルの整合性がだんだんと失われていって、最後にはまともに動かないシステムになってしまうためだ。

OSやアプリケーションのベンダーなどは、**最新のアップデートを適用して使ってくれ**とユーザーに要請していることがほとんどだが、一方で、アップデートファイルに添付されている注意書きなどでは「アップデートファイルを適用した結果生ずる損害の責任はとらない」とか、「現在の環境で問題がなければ、アップデートは適用するな」とかいったような、**相矛盾する記述**があったりする。

経験的に言って、**必要ないパッチは当てない**ほうが安定した環境を維持できるような気がするのだが、やっかいなのは、将来、自分にとって必要のあるアップデートが公開されたときに、過去のアップデートを適用していないといけない場合もある点だ。そのような場合は、古いアップデートを探し出して先にインストールする必要があるのだが、アップデートに複数のバリエーションがある場合などは、アップデートの「経路」が複雑な迷路のようになることもあったりして、面倒くさくなって放り出してしまいたくなる。いわば、**アップデートの無間地獄**（パッチ地獄ともいう）である。

これは、別にWindowsに限った話ではなくて、Linuxの世界でもまったく同様、あるいはもっとひどいのかもしれない。ご存じのように、1つのLinuxディストリビューションは数千もの大小プログラムの集合体で、それぞれのプログラムが日々更新されている。もしも、これらのすべてのプログラムを最新の状態に保ち続けようとしたら、おそらく毎日がパッチ情報の収集と適用作業に費やされてしまうことになるだろう。しかも、Linux上のプログラムは、別のプログラムに依存して機能していることが多いので、あるプログラムを最新のものに更新したら、まったく別のプログラムが動かなくなってしまう、という状況が頻繁に起こる。よほどの物好きか暇人でないかぎり、もはや、個人のLinuxユーザーが、ディストリビューション中のすべてのプログラムの面倒を見るというのは、現実的でない時代になっているのだ（だからこそ、**安くない対価**を払ってディストリビューションを「購入」する意味があるのだ）。

バックアップの必要性

個々のプログラムのパッチ当てが現実的でないとすると、個人ユーザーはディストリビューションの「最新バー

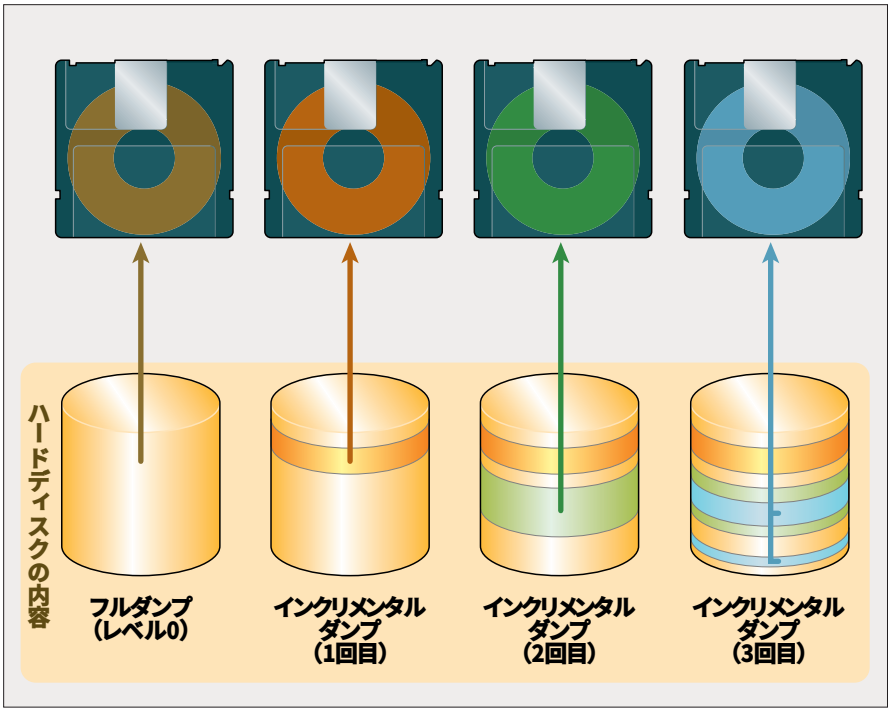


図1 インクリメンタルダンプの原理
色が違う部分は、それぞれ前回のバックアップ後に変更されたデータ（追加・削除・更新されたファイル）を示す。

ジョン」がリリースされるごとに、Linuxのインストールをやり直したり、アップデートすることになる。プログラム単位のパッチ当てとは異なり、ディストリビューション全体を更新するとなれば、ハードディスクの内容の大部分（あるいは全部）を入れ替えることになる。すると、いままでディスク中にあったデータが失われたり、書き換えられたりする可能性があるわけで、何らかの形で今までのデータを**保存**しておく必要がある。また、アップデートするのではなく、まっさらの状態からOSを新規インストールしたほうが都合のよい場合もあるだろう。そのような場合も、古いOS上で使っていたさまざまなデータをどこかに退避しておいて、あとから元に戻すという作業が必要になる。

OSのアップデート以上に危険なのが、ディスクの故障やプログラムの暴走（あるいは破壊的コンピュータウィルス）などが原因で、ディスク上のデータが突然に破壊されてしまう可能性だ。このような場合に備えて、ディスクの内容を定期的に保存しておいて、事故があった際にすぐに復旧できるようにしておく。

このような作業のことを**バックアップ**と呼んでいるのはすでにご存じだろうし、ややこしい講釈を聞くまでもなく、バックアップが重要であるということはだれもが**常識**として知っていることだろう。しかし、PCを個人的に使っているユーザーで、バックアップ作業を忠実に実行している人はどのくらいいるだろうか？ 少なくとも、筆者の周囲にいる人々（多くはPC関連雑誌の編集者とか筆者さんたちだが）に聞く限り、教科書通りのバックアップをしている人は多くない。思いついたときに、必要なデータだけを限定的に保存するという方法を採っている人が多いようだ。

UNIXやコンピュータシステム管理の古典的な本を見ると、必ずバックアップについて解説している章があって、**インクリメンタルダンプ**とか**フルダンプ**とかいうような言葉を目にする（**図1**）。そのような本では、たとえば週に1回はフルダンプを、毎日1回インクリメンタルダンプを行うことを勧めているのだが、これを個人ユーザーが実行するのは現実的でない（もちろん、組織内で共用しているマシンでは重要なことだが）。

個人レベルのバックアップで主な障害となるのは、バックアップ作業に必要となる時間や手間、そしてバックアップ先のメディアをどうするかという問題である。そこで、できるだけ手間をかけずに、効率よくバックアップする方法を考えてみよう。

本命はdumpコマンドだが……

古くから、UNIXの世界でバックアップに使われてきたコマンドが**dumpコマンド**である。もちろん、Linuxでもdumpコマンドが使えるようになっているのだが、dumpコマンドにはいろいろと制約があり、普通の個人ユーザーに向いているとはいえないのが現状だ。

その理由は、dumpコマンドがディスク（ファイルシステム）のバイナリイメージを**磁気テープ装置**に記録することを前提として作られている点だ。いまどき「磁気テープ装置」なんて、めったにお目にかかれないだろうし、もし購入しようとすれば十万円単位の出費を覚悟する必要があるだろう（中古やジャンク品を探せばもっと安いかもしれないが）。しかも、データを記録するためのテープカートリッジが入手は困難なうえ、非常に高価（1本数千円以上もする）なので、もはや個人で手を出せるものとは考えないほうが

よい。しかも、どんなに高価なテープドライブであっても、データの転送速度はかなり遅いうえ、書き込んだデータのエラーチェックをするために、**ちょっと書き込んでは巻き戻し**という操作を延々と繰り返し、Gバイト単位のデータをバックアップしようとすれば、機種によっては数時間を要する大変な作業になってしまう。

dumpコマンドを使って、テープ以外のメディア、たとえばMOディスクやリムーバブルハードディスクなどにバックアップできないわけではないが、複数のディスクにまたがったバックアップがきちんとできる保証がないなど、あまりお勧めできない。また、dumpコマンドはコマンドラインのオプションも複雑で難しく、たとえばバックアップする対象のディレクトリとバックアップ先デバイスの順番を間違えると、**ディスクの内容を一瞬にして壊す**危険性もあるので、**しろうと**を自認するユーザーは**手を出さないほうが賢明**だ。

余談だが、経験豊富で聡明なシステム管理者であっても、dumpコマンドを使ったバックアップ作業は神経を使うという。とくに、前述した「インクリメンタルダンプ」を正しく行うには、いくつかのテープの「セット」を用意して、それらをとっかえひっかえしながら日々のバックアップを行わなければならない。ちょっとうっかりして、上書きしてはいけないはずのテープに書き込んでしまっただけで、それまでのバックアップの蓄積が**パー**になってしまい、フルダンプを行うところからやり直す羽目になったりする。そんなわけで、毎日きちんとバックアップをしているはずのホストなのに、ある日管理者からのアナウンスがあって、**すべてのファイルが1週間前の状態に戻ってしまう**といった事件に何度も遭遇したものだ。かように、バックアップというのは難しい作業なのだ。結局、**個人のファイルは個人で保存**という癖を身につけることになるのである。

保存する対象を選ぶ

教科書どおりのバックアップが現実的でないとすると、どのような手段でどのようなデータをバックアップしておけばよいのだろうか？ 結論から言えば、**自分で作ったり変更したりしたデータだけ**保存するようにすればよい。

大昔のUNIXワークステーションは、OSのライセンス料がものすごく高価なうえ、OSのインストール用メディアが別売りだったり、ユーザーによるOSの再インストールを一切認めていなかったりした（壊れたときには、メーカーのサービスマンが出張してきて、再インストール作業をしてくれるのだ）。このため、ワークステーションを導入したら、まず最初に行うのが「システム全体のフルダンプ」だったのだ。万一ハードディスクの内容が壊れても、このフルダンプしたテープをディスクに書き戻せば、「買ってきたときの状態」に戻すことができる。このへんは、現在のPCに付属する「リカバリーCD」に似ていなくもない。

Linuxの場合も、インストール後にディスクの内容をまるごとバックアップしておけば、トラブルがあったときに再インストールの手間が省けるのでは、と考えるかもしれないが、現在のLinuxディストリビューションは、インストーラの改善が進んでいて、インストールにかかる手間は以前ほどではない。インストール時に入力する情報もそれほど多くないので、**どこかにメモ**しておけば、2回目からのインストール作業はすんなりと終えられるはずである。たとえ、まるごとバックアップしておいたデータがあったとしても、ディスクのパーティション分割、フォーマット、リムーバブルメディアのマウントなどの作業を、すべて**手動で**行われければならないのだから、手間や時間はLinuxを再インストールする場合と変わらないどころか、より高度な知識と神経を使う作業が要求されてしまう。

Linuxでファイルをバックアップするときは、**ディレクトリ単位**でバックアップするのが便利だ。昔は、**ファイルシステム（パーティション）単位**にバックアップしていたのだが、現在のLinuxユーザーでディスクを細かくファイルシステムに分割している人はそれほど多くないだろう。せいぜい、ルートパーティションと/homeパーティションの2つくらいだと思う。あるいは、インストーラのデフォルト設定にしたがって、すべてのディレクトリを1つのパーティションにしていることが多いだろう。したがって、パーティション単位のバックアップでは、ファイルの分量が多くなりすぎて、MOディスクなどにバックアップするのは面倒な作業になってしまう。ディレクトリ単位にバックアップすれば、1回にバックアップする分量はせいぜい数百Mバイトに収まるし、大きなディレクトリなら、さらにそのサブディレクトリを単位にすれば、バックアップの分量をある程度調整できる。

さて、どのディレクトリをバックアップすべきかというと、まず次の2つが挙げられるだろう。

- **/homeディレクトリ**

・/etcディレクトリ

/homeディレクトリは、各ユーザーのホームディレクトリが含まれる。/etcディレクトリには、ネットワークやハードウェアなどの設定情報、ユーザーのパスワード情報などが保存されている。/homeディレクトリについては、OSの再インストール後にそのまま書き戻せばよいだろう。/etcディレクトリは、そのまま新しい環境に上書きしてしまうと都合が悪いのだが、別のディレクトリ（たとえば/old_etcなど）に書き戻すようにして、様子を見ながら新しい環境に少しずつファイルを移転するようにすればよい。

これら以外のディレクトリについては、各個人の使い方の**くせ**によることもあるので、一概にどれが必要と決めるわけにはいかない。たとえば、筆者などはインターネットなどから取得してきたさまざまなファイルを、とりあえず**/tmpディレクトリにため込むくせ**があるので、ファイルのバックアップをするときは、かならず/tmpディレクトリもバックアップすることにしている。

ところで、/usrディレクトリ以下には、さまざまなアプリケーションの実行ファイルや設定ファイルなどが保存されていて、Linuxのインストール後に自分でインストールしたアプリケーションのファイルなどもここに保存されている。だから、**/usrディレクトリもバックアップすべきでは**と考える人もいるだろう。しかし、大昔のUNIXやLinuxとは違って、現在のLinuxでは/usrディレクトリそのものを保存しておいても、Linux自体をアップグレードした後で、アプリケーションがそのまま動作するとは限らない。ちょうど、Windowsなどでアプリケーションのディレクトリを別のマシンにコピーしても、動かないことが多いのと同じである。だから、アプリケーションのバックアップについても、インストール用のRPMファイルとかTARファイルなどだけ保存しておいて、Linuxの再インストール後に改めてアプリケーションもインストールしなおしたほうがよい。余計な手間がかかるように思えるが、結果的にはこちらのほうが効率が良いのだ。

バックアップ先はどうする？

テープドライブが使えないとなると、データを保存する手段（いわゆる「メディア」）はどうすればよいだろうか。**フロッピーディスクドライブ**なら、ほとんどのマシンに装備されているのだが、容量があまりにも小さいのでこれが**問題外**なのはいうまでもない。となると、MOドライブやCD-R、Zipなどの各種リムーバブルディスクということになる。どれを使うかについては、読者の好みに任せるとするが、設定の容易さ、互換性の高さ、記録速度の速さなどから、筆者はMOを使っていた。

MOなどのリムーバブルメディアをLinuxマシンのバックアップ用に使うには、SCSIインターフェイスの準備、メディアのフォーマット、メディアのマウントなどの作業が必要になる。

ところで、たった今、筆者は「MOを使っていた」と過去形で書いたのには理由がある。現在はLinuxマシンのバックアップの目的にはMOディスクは使っていないのだ。ではどうしているかといえば、Sambaなどで接続した**Windowsマシンのハードディスク**にバックアップしているのである。それも、Linux側から直接Windowsマシンにデータを送るのではなく、Linux側のハードディスクの適当な場所（たとえば/tmpディレクトリ）にバックアップファイルをいったん作ってから、Windowsマシン側からそのディレクトリを共有して、Windows側にコピーしている。

ハードディスク上のデータを保存するのに別のハードディスクに転送するのでは意味がないのでは、と不審に思う読者もいるかもしれないが、別々のマシン上に存在する2台のドライブが同時に故障する確率はかなり低いし、それぞれがまったく別のOSで動いているとすれば、たとえばコンピュータウィルスなどによる被害があった場合も、同時に両方がダメになる可能性を低くできる。だから、Linuxマシンのファイルは Windowsマシンに、逆にWindowsマシンのファイルはLinuxマシンにバックアップするという方法を使っているのだ（**図2**）。ハードディスクの記憶容量あたりの単価はものすごく安くなったし、複数のマシンを使っているユーザーには、このような**クロスバックアップ**（筆者の造語だが）の方法をお勧めしたい。

tarコマンドで「固める」

さて、前置きが長くなったが、バックアップするディレクトリと保存先が決まったら、いよいよ実際の作業を開始する。作業の手順自体は簡単である。まず、tarコマンドを使って、保存したいディレクトリの内容を1つの圧縮されたファイルにまとめる。この作業のことを、**固める**といったりもする。ここでは、作成したバックアップファイルを/tmp/home.tgzというファイルに送っているが、Linuxマシンに接続したMOディスクなどに保存する場合は、

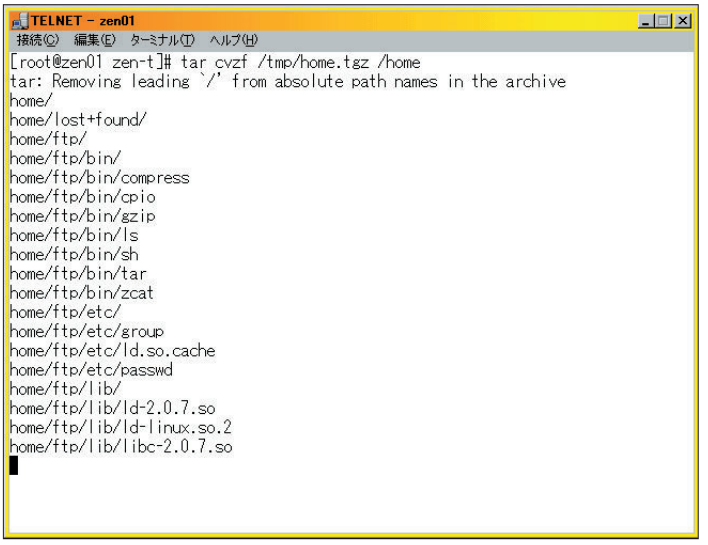
そのディスクをマウントしたディレクトリ上のファイル（/mnt/mo/home.tgzなど）にすればよい。

```
# tar cvzf /tmp/home.tgz /home
```

すると、**画面1**のように、

```
tar: Removing leading `/' from absolute path names in the archive
```

というようなメッセージが表示されてから、ファイルの保存状況を示すメッセージが表示される。最初の警告メッセージは、保存するファイルのパス名を「絶対パス」から「相対パス」に変更したという意味のものだが、ここではそれほど気にしなくてもよい。ちなみに、tarコマンドのオプション「cvzf」の意味だが、それぞれ「c」は「ファイルをtarファイルに追加せよ」、「v」は「追加中のファイル名を表示せよ」、「z」は「ファイルを圧縮せよ」、「f」は「このオプションの次に記述したファイルに保存せよ」というような意味である。



画面1
tarで/home以下を1つのファイルに圧縮するようす。できたファイルをほかのコンピュータにコピーしておく。

保存してあったファイルをディスクに書き戻す（この作業を**リストア**と呼ぶ）ときは、同様にリストアするディレクトリの1つ上のディレクトリに移動してから、同じくtarコマンドを使う。たとえば、/homeのバックアップファイルをリストアしたい場合は、次のようになるだろう。

```
# cd /
# tar xvzf /tmp/home.tgz
```

いきなり同じディレクトリにリストアするのが不安な場合や、アーカイブファイル中の一部のディレクトリだけリストアしたい場合は、適当なディレクトリに一時的に展開してから、改めて元のディレクトリにコピーしたほうがよいだろう。たとえば、/home/zen-tというディレクトリだけを/home/zen-tにリストアしたい場合は、次のようになる（リストア先のドライブに/home/zen-tというディレクトリが存在しない場合）。

```
# cd /var/tmp
# tar xvzf /tmp/home.tgz
# cp -rf /var/tmp/home/zen-t /home
```

tarコマンドを使ってファイルをバックアップ／リストアする場合は、前回に少し触れた「ユーザーID（UID）」や「グループID（GID）」にも注意する必要があるのだが、紙面が尽きたので、次回はtarコマンドのさらに詳しい使い方とファイルのUID、GIDについて解説することにしよう。

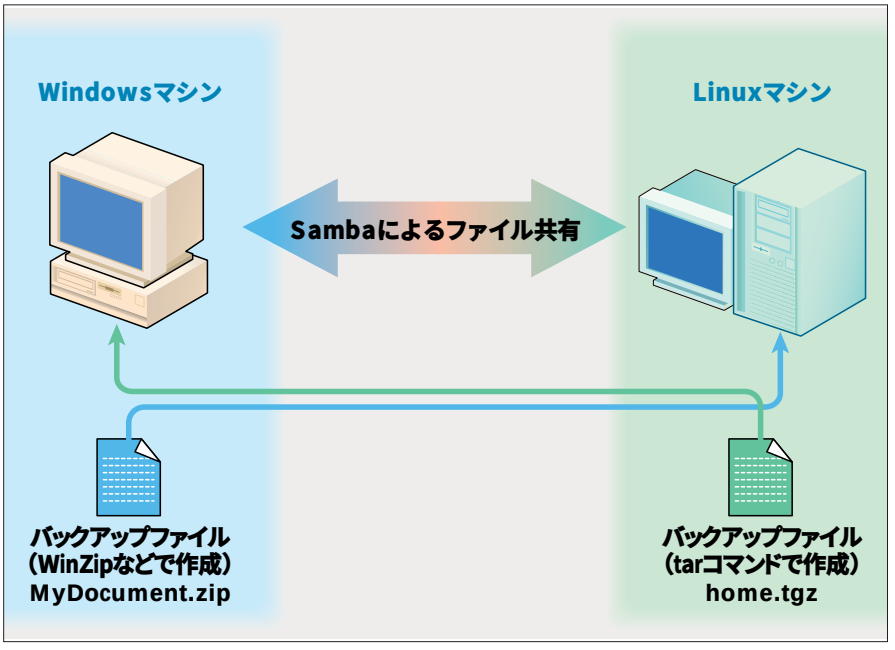


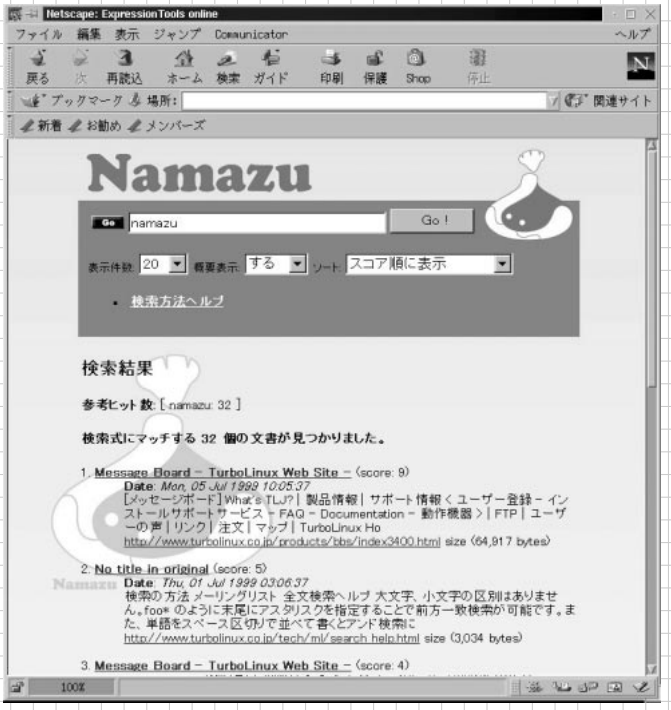
図2 「クロスバックアップ」の様子
ハードディスクのバックアップファイルを、それぞれ相手のディスクに保存する。両方のマシンが同時に故障しない限り、安全は確保できるし、リムーバブルメディアが不要なので、バックアップの手間もかなり低減できる。

Namazuで一発検索！

日本語全文検索ソフトNamazuを使ってWebサイトに検索機能をつけよう！

文：かざぐるま + 香山明久
Text : Kazaguruma + Akihisa Kayama

日々情報が増加するサイトを管理するWebマスターにとって、利用者に使いやすい検索ナビゲーションを提供するというのは結構頭の痛い問題となっているのではないだろうか。
そんな課題を解決してくれる日本語全文検索ソフト。それが「Namazu」だ。LinuxをはじめとするUNIX系OS環境下での利用に加え、一般のWebマスターが利用しているであろうWindows環境からもでも手軽に利用できる、このスーパー便利な検索ソフトを活用すれば、明日からあなたのサイトの魅力も倍増すること間違いなしだ！



日本語全文検索システムとは

事務のOA化が急速に進化し、インターネットによる情報発信が本格化するにつれ、我々の身の回りに溢れる電子化ドキュメントは、いつの間にやら膨大な数へと膨れ上がってしまったようだ。星の数ほどあるホームページから特定の情報を得たい時、ほとんどの人は「Yahoo!」や「goo」といった検索サイトのお世話になっていることだと思う。

Yahoo!のように人手によって情報を収集し、登録しているサイトもあるが、ビット化されるドキュメント量が飛躍的に増加してしまった現代では、人力による情報整理だけではその増加スピードに追いつくことなど到底不可能だ。そのため、検索サイトではgooのようにロボットプログラムで日本中（あるいは世界中）のWebサイトのデータを収集するようにしているところがほとんどである。

また、Web用に用意されたドキュメント以外でも、ワープロで作成した文書やメールなど、たくさんのデータから高速に検索できるシステムが必要になってきた。

こうした状況の中で、「Namazu」をはじめとする日本語全文検索システムの技術は、パーソナルユースはもちろんのこと、インターネットやイントラネットなど、より広範な環境において膨大な電子情報を有効利用するために欠かすことのできないものとなってきている。

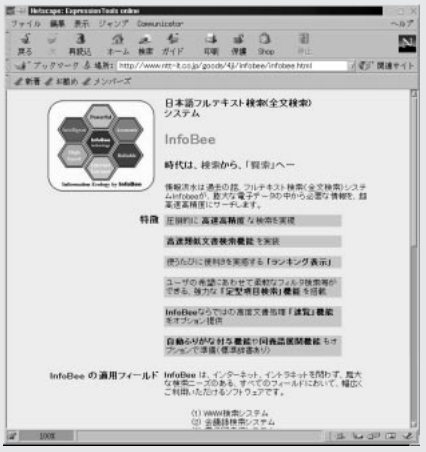
検索サイトで使われている

一般に、ドキュメントの全文検索はさまざまなプロセスを経て実現されている。この一連の作業を行うプログラム群を総称して「全文検索システム」と呼ぶ。

インターネットやイントラネットなどのネットワーク上に存在するドキュメントを対象とする全文検索システムの場合、その中身は検索対象データの収集を担当するロボット部、情報を分析し検索結果を返す検索エンジン部、ユーザーインターフェイスを提供する検索クライアント部の3つに大別される。中でも、検索エンジン部が担う役割は、ドキュメントの内容を正確に解析し、正しい検索結果を返すために大きな位置を占めている。

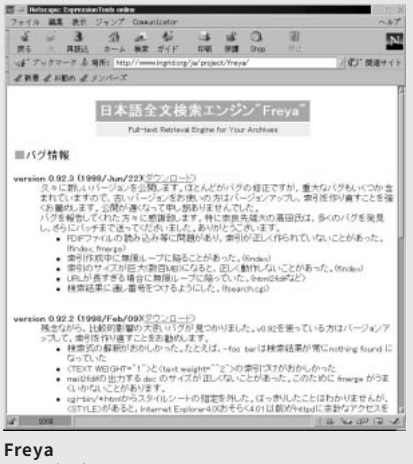
全文検索ソフトウェアには、下の一覧のように、商用からフリーまでさまざまなタイプのものが存在しており、現在国内の主要な検索サイトで採用されている（あるいはされていた）ものだけでもいろいろな種類がある。各ソフトそれぞれに検索時の特徴や癖があ

主な日本語全文検索ソフトウェア



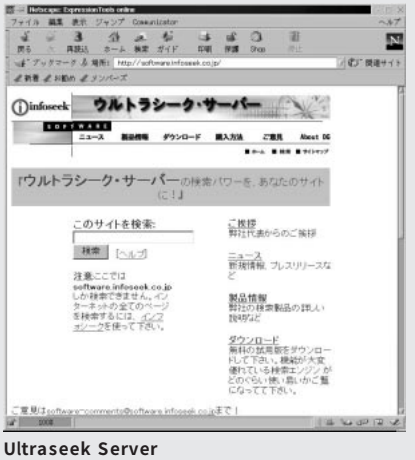
InfoBee
NTTインテリジェントテクノロジ株式会社
http://www.ntt-it.co.jp/goods/4ji/infobee/infobee.html

あの「goo」の検索エンジンにも採用されている実績十分のソフトウェア。高速類似文書検索機能を実装しており、高速高精度な検索が可能。機能は多彩で、PDFファイルの検索にも対応している。Windows NTおよびSolaris上で稼働。



Freya
原田昌紀氏
http://www.ingrid.org/ja/project/freya/

昨年9月まで「ODIN」で稼働していたソフト。フリーソフトウェア。ODINのエンジン部は、Java言語で記述された「Jerky」に継承されたため、Freyaの最新バージョンは'98年にリリースされた0.92.3のままとなっている。



Ultraseek Server
株式会社デジタルガレージ
http://www.software.infoseek.co.jp/

「infoseek Japan」の心臓部を担う検索ソフト。自然言語検索、あいまい検索などさまざまな検索要求に対応可能。PDF、Word、Excel等の主要フォーマットにも対応。試用版はWebサイトから無料でダウンロードできる。

るのだが、これらすべてに共通していることは、『日本語が通る』こと、そして『検索のためのインデックスを作成する』ことである。

こうした特徴は「Namazu」にも共通しているので、まずは、一般的な日本語検索システムの仕組みについて考察してみる。

日本語の解析

一般に、インターネットで検索を行う場合、複数の検索サイトに同じキーワードを入力しても、同じ結果は返ってこない。これは、ロボット部によって収集された対象データ量の差にも左右されるが、それぞれのシステムの検索エンジン部が採用する手法の違いに依存する部分が大きい。特に、日本語を検索対象とする場合、その処理は複雑なものとなる。

全文検索システムは、米国において早い時期から開発され発展してきた。これはもちろん、ワープロなどの普及によりドキュメントの電子化が早い時期から実現していたということが大きな要因ではあるのだが、英語という言語の構造が全文検索システムを構築するに適しているという事実に依るところも大きい。

英語では、文章中にあるそれぞれの単語がスペースで区切られている。それに対して日本語は、連続する単語によって文章が記述される。単語をスペースで認識することができる英語の場合、キーワードの抽出を機械的に処理することが比較的容易なのだが、日本語において単語を切り分けるためには、まず文章の構造を文法的に解析し、それぞれを単語に分解する「形態素解析」を行わなければならない。そのため、日本語の全文検索は、英語よりも複雑な手順が必要となってしまう。

たとえば『融資』というキーワードで全文検索をかけたいとしよう。この場合、単語を適切に切り分けていないと、『融資』という単語以外に『金融資産』という別の熟語も検索結果に含まれてしまう。この『金融資産』という熟語が『金融』と『資産』に分解された形で認識されれば、より精度の高い検索が可能となるわけだが、それをコンピュータが理解するためには、文章を文法的に解析することが必要となるわけだ。

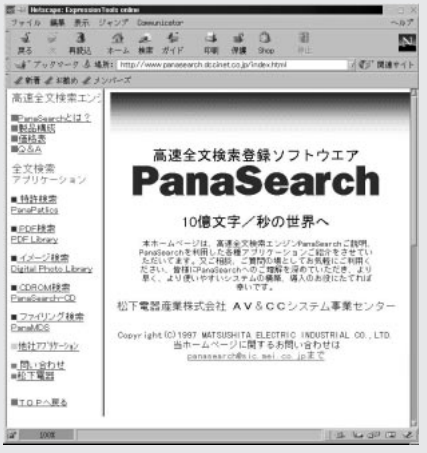
このようにドキュメントを単語に切り分ける作業は、一般に「わかち書き」などと呼ばれ、現在、日本語全文検索システムを実現するための主流となっている。Namazuにもこうしたプロセスが取り入れられている。

こうした手法以外にも、全文検索に用いられる言語処理にはさまざまなタイプのものが存在する。「文字成分表」や「N文字インデックス法」などがその代表だ。

「文字成分表」とは文法的な解析を行わない方式で、文書中に現れるすべての文字をバラバラに記憶し、検索キ

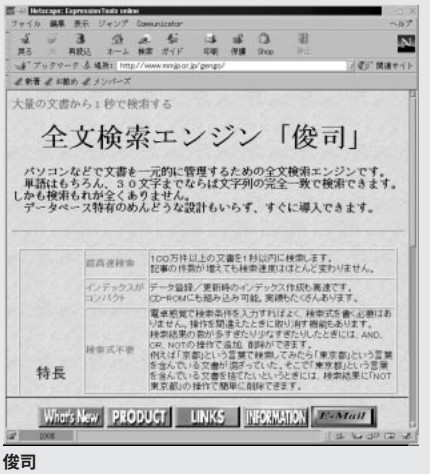
SUFARY
奈良先端科学技術大学院大学
情報科学研究科自然言語処理学講座
http://cactus.aist-nara.ac.jp/lab/nlt/ss/

suffix arrayというデータ構造を用いて、高速な文字列検索を行なうためのライブラリを中心としたパッケージ。「Dragon Next」のキーワード検索部分にも使用されている。商用利用可のフリーソフトウェアとして公開されている。



PanaSearch
松下電器産業株式会社
http://www.panasearch.dccinet.co.jp/

10億文字（新聞記事10年分）を平均1秒で検索するという高速性がウリ。フリーワード検索やクライアントからのリアルタイム情報登録も可能で、PDFファイルにも対応している。「Yahoo! Japan」などへの採用実績あり。



俊司
言語工学研究所
http://www.mmjp.or.jp/gengo/

データ登録や更新時のインデックス作成が高速でコンパクト。さらに、100万件以上の文書を1秒以内に検索する高速性を備える。現在リクルート社が運営している検索サービス「ISIZE」の前身「あちゃらNAVI」などでも活躍していた。

ーワードに含まれるすべての文字と一致する文書を検索結果として表示するもの。「N文字インデックス法」も文法解析を用いない手法で、文中から1文字ずつ、ずらしたN文字の並びを順に取り出し、その並びとキーワードの一致を検索するものだ。

いずれの手法も、形態素解析を必要としないため、システムが複雑化しない、巨大な辞書を持つ必要がない、というメリットがある反面、検索精度が低下してしまう、切り出される単語情報が肥大する、などマイナスの一面もある（**図1**）。このようにキーワードの切り分けに用いられる手法やその精度の違いにより、全文検索ソフトが返す検索結果に差が出る。

検索結果にどれだけ広い範囲の情報を拾い上げるかの指標を「再現率」と呼び、取り出した情報がどれだけ検索結果にマッチしているかの指標を「適合率」と呼ぶが、一般に再現率を上げようとすると、利用者が必要としない情報、いわゆる「ノイズ」を多く拾い上げてしまう。また、適合率を上げようとすると、有益な情報の「漏れ」が生じてしまう。

システムの違いによる日本語検索結果の差は、ソフト内部のアルゴリズムを決定する際に、再現率と適合率のどちらに重きを置いているか、あるいはシステム構成を複雑にするかどうか、などの要因で大きく変化するのだ。

言語処理に用いられる手法

■わかち書き

文章を文法的に解析し、単語を切り出して検索を行う手法。ドキュメントの形態素解析を行うため、一般に大きな辞書ファイルを要求する。正確な検索結果を返すことができるが、単語を切り出す作業に時間がかかる。

```
日本語全文検索システムの仕組み
        ↓
日本＋語＋全文＋検索＋システム＋の＋仕組み
```

■文字成分表

文章中に表記されるすべての文字種を切り分けてしまう手法。処理は高速であるが、「日本」というキーワードで検索をかけた場合、「本日」という単語もヒットしてしまうなど、一般に適合率が低くなってしまう傾向がある。

```
日本語全文検索システムの仕組み
        ↓
日＋本＋語＋全＋文＋検＋索＋シ＋ス＋テ＋ム＋の＋仕＋組＋み
```

■N文字インデックス

文章中からN文字数（Nは変数）を切り出し、それを1文字ずつずらしながら単語を認識する手法。文字成分表よりも精度が高まり、漏れのない検索が可能となる。Nの値を大きくすると処理が重くなり、逆に小さいと適合率が下がってしまう。

```
日本語全文検索システムの仕組み
        ↓
日本
  本語
    語全
      全文
        文検……　（N＝2の場合）
```

図1　言語処理に用いられる手法

インデックスの作成

全文検索を実行する場合、対象データすべてを読み込んで、検索キーワードにヒットする文字列を検査する方法がある。これはfindコマンドやgrepコマンドを実行するようなもので、対象範囲がさほど広くない場合は手軽でよいのだが、多くデータを対象とする場合は検索に長い時間を要し、CPUに与える作業負荷も大きなものとなってしまう。また、ローカル環境ではある程度の効果も期待できるが、ネットワーク環境のように対象データが点在する場合には実用的ではない。

そこで、現在実用化されている多くの全文検索システムでは、対象ファイルから切り出したキーワードを保存するための「インデックス」を作成し、検索クライアントからの要求に応じ、このインデックスから検索結果を返すという手法が採用されている。

このインデックスには各ドキュメントから切り出されたキーワードが登録されるほか、文書要約を表示するタイプの全文検索システムでは、そうした文章もあわせて登録される。

インデックスを用いるタイプの全文検索システムの場合、検索処理を高速に行うことができるが、逆にこのインデックスの作成に手間がかかってしまう。大量の日本語ドキュメントを形態素解析する場合などは、それなりの時間も必要となってくる。また、新規に作成されたデータや更新された情報などは既存のインデックスからは知ることができないため、インデックスをまめに更新することも大切だ。

全文検索システム「Namazu」

CGIを使ったWeb日本語全文検索システム構築を手軽にできるソフトとして、また、電子ネットワーク協議会が主催する「オンラインソフトウェア大賞 '98－99」に入賞した秀作フリーソフトとして、今やメジャーなフリーウェアのひとつとなった「Namazu」であるが、この名前を初めて耳にする方のために、まずはその概要を簡単に紹介する。

Namazuの概要

Namazuは、もともとは高林哲氏が開発したソフトで、GPL2に従ったフリーソフトウェアとして公開されている。現在は高林氏を中心とする「Namazu Project」によって共同開発が行われており、UNIX系OSのみならず、WindowsやOS/2といった幅広い環境に移植されている。開発は非常に活動的で、バージョンアップが頻繁に行われているという点も心強い。

Namazuの開発環境はPlamo Linux＋XFree86＋Mule＋gcc＋Perl＋Apacheとなっているため、たいていのLinux環境で動作が可能だと思われる。動作が確認されているカーネルは、Linux 2.0.18・2.0.30・2.0.32・2.0.33だ。

ホームページ上に公開されている作者の解説によると、Namazuは「手軽に使えることを第一に目指した日本語全文検索システム」であり、「CGIとして動作させることにより中小規模のWeb全文検索システムを構築することができるソフト」となっている。現在、日本語全文検索を実現するためのフリーソフトとしては圧倒的な人気を誇っており、かなりの数のサイトで実際に運用されているため、その実績も十分だ。

前述の解説の中では「中小規模のWeb」を対象にするとあるものの、NamazuのFAQページに「報告のあったもののなかでは878,914ファイル、合計2,167,480,108bytesの文書を対象にしたものが最大のインデックス作成実績」とあり、かなり大規模なサイトでも運用することが可能と思われる。

Namazuは、厳密に言うと日本語全文検索システムを構成するプログラム群のひとつであり、Namazuを使ったシステムを構築するためには、日本語の形態素解析を行うために「KAKASI（kanji kana simple inverter）：馬場肇氏」や「ChaSen（茶筅）：奈良先端科学技術大学院大学」、およびnkfなどの外部ソフトが必要となる。

また、Web上から情報を収集するためのロボットはNamazuには含まれないため、検索サイト的な用途を目的とする場合には、Webサイトから直接ファイルを収集する「GNU Wget」のようなソフトを別途用意する必要がある。

Namazuの特徴

Namazuは、インデックス作成の部分（インデクサ部）をPerlで実行し、検索部分をC言語によるプログラムで行う仕様になっている。検索速度は、公式サイトに掲載の数値によると、Pentium166MHz、メモリ64MバイトのLinux機で数十Mバイトのファイルを元に作成されたインデックスを検索した場合、0.1秒程度とかなり高速である（厳密にはOSのディスクキャッシュに大きく影響されるが…）。

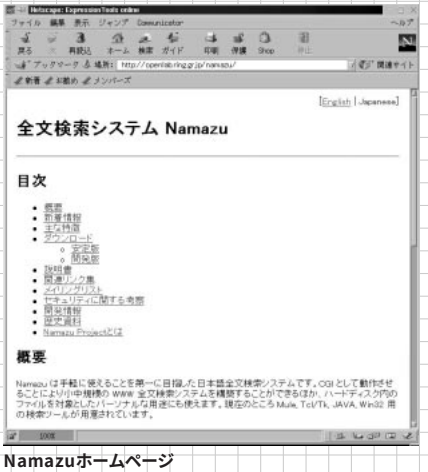
検索時のオプションも、同種の商用ソフトに劣らぬほど多彩だ。主な仕様としては、

- **検索式はand／or／not検索に対応**
- **前方／中間／後方一致検索が可能**
- **記号を含む単語の検索が可能**
- **日付による検索結果のソートに対応**
- **結果表示を、スコアを元に出力可能**
- **検索結果の文書要約を表示可能**
- **検索キーワードのログ取得が可能**

などを挙げることができる。

CGIとして動作させることでWeb全文検索エンジンとして機能するので、インターネットやイントラネット上の検索にはブラウザを検索クライアントとして利用できる。そのほか、X Window SystemやWindows系OS上で動作する検索クライアントも用意されているため、ハードディスク内のデータを対象としたパーソナルな環境でも活用できる。

インストール自体はさほど複雑ではないため、アイデア次第でさまざまな用途で活躍してくれることだろう。



Namazuホームページ
http://www.namazu.org/

Namazuのインストールから活用まで

では、実際にNamazuを導入するための手順を見てみることにしよう。今回のテストにはLASER5 Linux 6.0を使用した。

Namazuのインストール

本記事では、Namazu 1.3.0.11を使用している。記載の情報によると、これより古いバージョンにはCGIに関するセキュリティホールが存在するため、すでに導入済みという方もこの機会にバージョンアップしておいたほうが良いだろう。ちなみに、Namazuは頻繁にバージョンアップが繰り返されているため、最新の情報を得たい場合はNamazuの公式サイトを確認することをお勧めする。

なお、2000年2月20日にNamazu 2.0のリリースが行われ、同時にNamazuの公式サイトが、従来のRing Serverプロジェクトの Namazuオープンラボ（http://openlab.ring.gr.jp/namazu）から、http://www.namazu.org/に変更になった。

まず、Namazuのバージョン1.3.0.11を公式サイトからダウンロードする。

Namazuをインストールするためには、本体以外に以下のソフトを集める必要がある。

- **Perl 5.003 以降**
- **gcc**
- **nkf v1.62**
- **KAKASIまたはChaSen**

このうち、Perl、gcc、nkfは通常であればLASER5のインストール時に導入されているはずだ。もし入っていないようであれば、あらかじめインストールしておく。

KAKASIまたはChaSenは、Namazuで日本語を扱うために必要となるプログラムである。どちらを利用してもかまわないが、今回はKAKASIを選択することにした。KAKASIはFTPサイト（ftp://kakasi.namazu.org/pub/kakasi/）からダウンロードする。

■KAKASIのインストール

Namazuのインストールに先だって、まずはKAKASI（もしくはChaSen）のインストールを行う。

Namazuなどの全文検索システム上でKAKASIを利用する場合、以前のバージョンのKAKASIではでは「わかち書き」を実現するためのパッチを当ててやる必要があったのだが、v2.3.0以降はこのパッチは本体に統合された。最新のバージョンは2.3.1なので、それ以前のバージョンを所有している人は、こちらを利用するようにしたほうがインストールの手間がかからない。

ソースコードからコンパイルする場合の手順は以下の通りだ。

```
$ tar xvfz kakasi-2.3.1.tar.gz
$ cd kakasi-2.3.1
$ ./configure
$ make
$ su
# make install
```

インストールが完了すると、/usr/local/share/kakasiに辞書ファイルが、/usr/local/libにライブラリが、/usr/local/binに実行バイナリが導入される。

■Namazuのインストール

KAKASIのインストールが完了すれば、次はNamazuのインストールだ。まずは、

```
$ tar xvfz namazu-1.3.0.11.tar.gz
$ cd namazu-1.3.0.11
$ cd src
$ ./configure
```

を実行し、ここで作成されたMakefileを必要に応じて編集する。Perl、KAKASI（or ChaSen）、nkfなどのパスは自動検出される。特に細かな設定を変更する必要がないのであれば、

```
OPT_ADMIN_EMAIL =
```

の部分に自分のメールアドレスを登録する程度でよいだろう。なお、CGIでNamazuを動作させるために、CGIDIRの行も修正しておこう。Red Hat系の場合には、

```
CGIDIR          = /home/httpd/cgi-bin
```

と指定する。Apacheをソースからインストールしている場合には、CGIのディレクトリが違うので/usr/local/apache/cgi-binを指定する。

```
$ make
$ su
# make install
# make install_cgi
```

でインストールは完了する。/usr/local/namazuディレクトリに各ファイルが無事インストールされているはず

だ。

■tknamazuのインストール

次にX Window System用の検索クライアントであるtknamazuをインストールする。

Namazuのインストールが無事完了すると、/usr/local/namazu/contribディレクトリ内に、tknamazu-1.11.tar.gzが現れているはずである。これを展開し、インストールを実行する。

```
# cd /usr/local/namazu/contrib
# tar xvfz tknamazu-1.11.tar.gz
# cd tknamazu-1.11
# make install
```

tknamazuではテキスト版Webブラウザのlynxが必要となる。lynxはLASER5 Linux 6.0をインストールする時に、/usr/binディレクトリに導入されているはずだが、tknamazuでは、/usr/local/bin/lynxを呼び出すようになっているので、

```
# ln -s /usr/bin/lynx /usr/local/bin
```

とシンボリックリンクを張っておく。インストールが完了すれば、X Window System上で/usr/local/namazu/contrib/tknamazu-1.11フォルダにあるtknamazuをクリックするだけでプログラムが起動するはずだ。

インデックスの作成

すべてのインストールが完了したら、今度は検索の対象となるファイルのインデックスを作成するためのプログラムであるmknmzを実行する。この時指定する引数は、インデックスを作成する元になるファイルを収めたディレクトリである。Namazuはデフォルトの状態では、テキストおよびHTMLデータを検索対象ファイルとするため、とりあえず今回はサンプルとして、LASER5 Linuxに収録されているGNOMEユーザーズガイド（HTMLデータ）を検索対象に指定してみる。

GNOMEユーザーズガイドが収められているディレクトリは、通常は/usr/share/gnome/helpになっているはずだ。

あとでWebブラウザから見るために、まずこのファイルをApacheのドキュメントディレクトリにコピーして、それからインデックスを作成する。

```
# cd /usr/share/gnome
# tar cf - help | (cd /home/httpd/html; tar xf -)
# mknmz /home/httpd/html/help
```

ファイルの数が多いと結構時間がかかるが気長にインデックスの完成を待とう。

検索の実行

インデックスが完成したら、tknamazuを起動してみよう（**画面1**）。テキストボックス内に検索したいキーワードを入力し、Searchボタンを押すと検索結果の一覧が表示される。tknamazuでは検索結果として表示されるファイルへのリンクが張られるため、ふだんインターネットの検索サイトで行うのと同様の操作で対象ファイルを閲覧することができる。

コマンドで検索を実行する場合は、以下の通りだ。

```
# namazu "キーワード"
```

X Window System用検索クライアントtknamazu

tknamazuの起動画面
見た目は小さなブラウザといったところだ。

検索結果を表示したところ
Web上で検索サイトを利用するのとほぼ同じ感覚で検索できる。

画面1 tknamazu

こちらの場合は、当然ファイルへのリンクはない。ちなみに、Namazuには検索結果をHTMLファイルに書き出してブラウザで読み込むコマンドライン用の検索クライアントbnamazuも用意されているので、こちらを使うと手軽だろう。

Web全文検索としての利用

NamazuをWeb用の検索エンジンとして自分のホームページで公開したいというニーズも多いことと思う。そうした利用の場合には、作成するインデックスに少々手を加えてやる必要がある。パーソナルユースの範囲でNamazuを活用するのであるのなら、さきほど紹介した方法で検索結果を得ることができるのだが、作成したインデックスをWebサーバにアップロードして一般のインターネットユーザーに公開するにはこのままでは具合が悪いからだ。

なぜなら、ここでインデックス化されているリンク情報は、すべてローカルにあるハードディスク上のファイルへリンクされているためである。

たとえば、前ページで作成したGNOMEユーザーズガイドのリンク情報は、

```
/home/httpd/html/help/xxx.html
```

のようになっている。このままでは、当然Web上のURLとしては認識されない。Webへの公開を前提としたインデックスを作成する場合は、

```
http://www.xxxxx.xx.xx/help/xxx.html
```

のような正規のURLにあらかじめリンクを置き換えてやる必要がある。

リンク情報を置き換える方法にはいくつかあるが、URLが固定の場合は「namazu.confファイル」を編集してしまうのが手軽だろう。

Namazuの展開フォルダ内（デフォルトでは/usr/local/namazu/lib）にnamazu.conf-distというファイルがあるので、これをnamazu.confにコピーして内容を編集する。namazu.confファイルをエディタで開くと、

```
#REPLACE   /home/foo/public_html/
http://www.hoge.jp/%7Efoo/
```

という1行があるので、これを、

```
REPLACE  /home/httpd/html/
                http://www.xxxxx.xx.xx/
```

という具合に、それぞれの環境に合わせて書き換える。この際、各項目がTABで区切られていないと正しく認識されないので注意が必要だ。

すでにローカル環境を対象にインデックスを作成している場合は、念のため、いったん/usr/local/namazu/indexディレクトリを空にしてから、mknmzで変更した設定で新たにインデックスを作成し直す。tknamazuに適当なキーワードを打ち込んで、表示されるURLが置き換わっていれば設定は無事完了だ。

もちろんこの場合、Linuxサーバ上でApache（httpdサービス）が稼働している必要があるし、Webブラウザで検索対象のHTMLファイルを見られるようになっていないと正しく表示されない。

CGIとしての利用

NamazuをCGIとして使用するには、Apacheの/cgi-binディレクトリあるnamazu.cgiを動作させる。mknmzでインデックスが作っておいてから、ブラウザからhttp:（サーバ名）/cgi-bin/namazu.cgiとアクセスすれば検索ページが表示される。

CGIとして利用する場合、一般の検索サイトと同様にブラウザが検索クライアントとなる。このデータは、当然のことながらHTMLデータとして吐き出されるのだが、Namazuの場合はこれらを生成するための雛形（テンプレート）が別データになっているため、好みの形にカスタマイズすることも比較的簡単だ。

検索対象を日本語に設定している場合、ブラウザ表示のカスタマイズを行うためには/usr/local/namazu/libディレクトリに収められているNMZ.head.ja、NMZ.foot.ja、NMZ.body.jaをエディタなどで編集する（元データはコピーして残しておくほうがよい）。特殊なタグは特にないので、HTMLの知識がある方なら編集も容易だろう。ただし<!- - - ->で囲まれているタグは、インデックス更新時に情報が書き換えられる部分になるため、そのまま触らずに残しておくこと。

特にブラウザ表示を変更する必要がない場合であっても、NMZ.head.jaに記述されている、

```
<TITLE>Namazu the full text
retrieval search system</TITLE>
```

部分くらいは自分のサイトに合わせて書き換えておくようにしよう。

■プロバイダ上で使えるPerl版Namazu

Namazuをプロバイダにある自分のページでCGIとして動かすためには、Perl版検索クライントを使用すること

になる。Perl版検索クライントの名前はpnamazuで、そのファイルはNamazuの展開フォルダ内（デフォルトでは/usr/local/namazu）の/contribに用意されている。

pnamazu-98.12.16.tgzを展開すると、一連のプログラム群が現れる。pnamazu.cgiとインデックスファイル（NMZ.*）をサーバにアップロードする。CGIの設定などは契約プロバイダにもよるので細かな解説はしないが、複雑な設定をするのでなければさほど難しくない。ただし、インデックスを含めると、サーバにアップロードするデータは結構なサイズになるので、レンタルサーバなどで容量制限が厳しい場合などは注意が必要だ。当然のことながら、プロバイダがCGIの利用を許可していない環境では使用することはできない。

そのほかの便利な活用法

■メーリングリストでNamazu

発言が活発なメーリングリストでは、多い日には何十通、時には何百通ものメールが投稿されることも少なくない。複数のメーリングリストに参加している場合などは、すべての投稿を保存しているだけでメーラが重くなって大変だし、途中からMLに参加している人には過去の話題がつかめなかったりする。Namazuはこういった場面でも非常に役立つ。

MLのアーカイブをインデックス化する際には、まずMHonArc（http://www.mhonarc.org/）のようなコンバータプログラムでメールデータをHTMLファイルに変換しておくのがよい。MHonArcは議論のスレッド単位で相互リンクを張ってくれるため、Namazuのワード検索と組み合わせることでさらに利便性が高まる。

Namazuの展開フォルダ内（デフォルトでは/usr/local/namazu）の/contribにはMHonArc 2.2.0向けに用意された用の日本語化パッチ（MHonArc-2.2.0-Japanize-patch-1.6.gz）が同梱されているので、メーリングリスト管理者はぜひ一度お試しいただきたい。

■いろんな場面でNamazu

このほかにも、Namazuには便利なツールがいろいろと用意されており、さまざまなシーンで活用することができる。これらの開発に携わられた多くの開発者の方々には本当に感謝したい。

Linuxの標準的なメーラであるMuleから検索を実行するインターフェイス（namazu.el-1999.05.10.gz）や、Javaの実行環境で動作する検索クライアント（NamazuJAVA.030.tar.gz）などがパッケージに同梱されており、Namazuをインストールすると、デフォルトで/usr/local/namazu/contribディレクトリに展開される。いずれも便利なツールなので、機会があればぜひ活用していただきたい。

Windows環境での利用

一般的にWebコンテンツは、WindowsやMac環境で制作が行われ、それをLinuxなどのUNIXサーバへアップロードするケースが多いことと思う。このような場合、コンテンツを制作する環境でファイルのインデックス化が行えれば何かと便利である。

残念ながら、Macに対応したNamazuは今のところ存在しないが、広瀬健一氏が開発したWin32用のバイナリパッケージを用いれば、Windows 9x/NT環境下でLinux上と同等のシステムを構築できる。

これは、現在リリースされているフリーウェア系の全文検索システムにはあまり見受けられない便利なものなので、最後にそのインストールと導入について簡単に紹介しておこう。

Namazu for Win32のインストール

■関連ファイルの導入

執筆時点でのNamazu for Win32のバージョンは1.3.0.11。ご本家と同様、セキュリティに関する不具合が修正されたものとなっている。Windows環境にNamazuを導入するためには、プログラム本体以外に以下のファイルが必要となる。

- **KAKASI for Win32**
- **ActivePerl**
- **NKF for Win32**

各プログラムの最新情報は、「全文検索システム Namazu for Win32」（http://www.tama.or.jp/%7Ekenzo-/Namazu/）に記載されているので、導入に際してはこちらを参考にすると良いだろう。

Namazu for Win32を導入するに先立ち、まずはWindows上でNamazuを動かすための環境を整える。なお、以下に掲載する実行ファイル名はすべて執筆時点の最新版であるが、名称はバージョンアップなどにより変更される場合があるのでご了承いただきたい。

まずは、入手したActivePerlをイン

ストールする。APi522e.exeを実行すればインストールは自動的に行われる。各種設定は、すべてデフォルトのままでよいだろう。NKF for Win32はアーカイブファイルのnkf3217.lzhに含まれるnkf32.exeだけを利用する。これをPATHの通っている適当なフォルダにコピーしてやる。Windows 9xの場合は、\windows\systemあたりにコピーしておけば問題ないだろう。

■KAKASI for Win32のインストール

ActivePerlおよびNKFの導入が完了すれば、次はKAKASIのインストールである。kks225w4.exeを実行すると、自動的にファイルが展開される。途中インストール先を尋ねてくるが、これはデフォルトのまま（C:\usr\local）変更しないようにする。

インストールが無事終了していれば、autoexec.batに**リスト1**の内容が追加されているはずだ。これらの環境変数が有効にするため、いったんWindowsを再起動する。

■Namazu for Win32のインストール

再起動後、Namazu for Win32をインストールする。導入は、KAKASIと同様にnmz130110.exeを実行するだけと簡単だ。あとの作業はインストーラの指示に従う。

インストール場所もデフォルトのままにする。インストールが完了すると**リスト2**の内容がautoexec.batに追加されているはずだ。Windowsを再起動すれば、Namazu for Win32が利用できるよう環境が整う。

導入を確認するため、とりあえず動かしてみるには、MS-DOSプロンプトを開いて以下のコマンドを打ち込み、Namazu関連ファイルを作成する。

```
C:¥>mknmz -O C:¥usr¥local¥namazu¥index
C:¥usr¥local¥namazu¥doc¥
```

ファイルの作成を確認するため、

```
C:¥>dir C:¥usr¥local¥namazu¥index
```

を実行して、**画面2**のような表示が現れれば関連ファイルは作成されている。

■namazu.confの作成

すべてのインストールが終了し、Namazu for Win32が利用できる環境が整ったら、次にnamazu.confファイルを編集する。このファイルは、本家Namazuのそれと同じ役割を持つもので、ローカルのハードディスクを検索対象にするような場合には特に設定の必要はないが、Webサーバ上で動かす場合には、すでに述べたようにURL設定の変更を行うために編集しなければならない。

▼リスト1 KAKASIインストール後、autoexec.batに追加される内容

```
REM ###### kakasi for Win32 Environment variable setting
PATH C:¥usr¥local¥kakasi¥bin;%PATH%
SET KANWADICTPATH=C:¥usr¥local¥kakasi¥lib¥kanwadict
SET ITAIJIDICTPATH=C:¥usr¥local¥kakasi¥lib¥itaijidict
```

リスト2 Namazuインストール後、autoexec.batに追加される内容

```
REM ###### Namazu for Win32 Environment variable setting
PATH C:¥usr¥local¥namazu¥bin;%PATH%
SET NAMAZUCONFPATH=C:¥usr¥local¥namazu¥lib¥namazu.conf
```

画面2 試しに作ったインデックスファイルを確認

namazu.confはインストールで自動作成されない。そのため、ファイルはc:\usr\local\namazu\libディレクトリにあるテンプレートファイルnamazu.conf-distを編集して用意する。namazu.conf-distをコピーし、それをメモ帳などで編集してnamazu.confという名前で保存すればよい。オリジナルのファイルは必ず残しておくようにしよう。

設定が必要と思われるの主に以下に挙げる項目である。なお、項目間は本家のnamazu.conf同様、TABで区切らないと正しく認識されないので注意しよう。

```
INDEX C:¥usr¥local¥namazu¥index
  インデックス格納場所を指定
#REPLACE /home/foo/public_h……
  URLを置き換えるための設定
#BASE file://localhost/home……
  検索結果に表示するURLの設定
```

以上の設定が完了すれば、Windows上でも本家Namazuと同様にWebサイトへリンクを展開することができる。

ローカルで利用する際に便利な「Search-S for Namazu」というのWindows用検索ツールもある。Namazuのパッケージには同梱されていないが、Namazu公式サイトに詳しい情報があるので、興味のある人はそちらも参照していただきたい。

大盤振る舞いの基本操作完結編

Emacsはじめました

第2回　エディタとしての基本操作

前回の説明で、どうやら最低限のテキスト編集ができるようになった。ちっちゃなファイルならOKだ。だけど、これだけじゃまだ「メモ帳」も同然、噂に聞く極楽の気分からはほど遠い。そこで、今回は基本操作のあれこれを欲張って大驀進し、最後に便利なコマンドリファレンスまでおまけした。読者の進境はいかに。

文：佐々木太良
Text : Taroh Sasaki

さて今回は、日本語の入力環境を紹介するはず、でした。Emacsを使う主な理由の67%は、日本語の読み書きが楽だからといいます（目の前の2人と私に聞きました）。これからアプリケーションのような使い方をしていけば、それはなおさら重要でしょう。ところが、Linuxのいろいろなディストリビューションを取り巻く状況を眺めると、話はそれほど簡単ではなさそうです。そこで日本語環境は次回に重点的に解説することにして、今回はエディタとしての基本編集操作を完璧にしましょう。

ファイルとバッファとウィンドウとフレーム

これまでなんとなく「ファイル」「バッファ（buffer）」「ウィンドウ（window）」という用語を使ってきましたが、ここでちゃんと調べておきましょう。

まずファイルはLinuxのユーザーが普通に「ファイル」と呼んでいるもののことで、これは問題ないでしょう。ファイルをEmacsで開くと、バッファと呼ばれるスペースに読み込まれます。同じファイルを何回開いたとしても、複数のバッファに読み込まれるようなことはありません。前回登場したC-x C-fコマンドを何度か実行しても、2回目からはすでに読み込まれたバッファが表示されるだけです。

さてウィンドウですが、これはEmacsのバッファを表示させるためにEmacsの画面の中で開く窓のような領域です。ウィンドウを同時にいくつか開くと、Emacsの画面の中で区切って表示されます。バッファにウィンドウ1つが対応しているとはかぎりません。ウィンドウを2つ用意して、バッファの先頭と最後をそれぞれに表示させることもできます。長時間Emacsを使っているうちに20ものファイル（バッファ）を開いていることがありますが、表示しているウィンドウ（編集するために表に出ている部分）は1つか2つだけ、ということも珍しくありません。

Emacsの「ウィンドウ」は、Xの1個のウィンドウの中で区切られた複数の領域それぞれを指しますが、これとは別にEmacsが使用するXのウィンドウを複数開くこともできます。これをEmacsでは「フレーム（frame）」と呼びます。

図1は、以上を説明するものです。X上にフレームが2枚あり、手前のフレームにはウィンドウが2つ開いています。バッファkakko.txtは2つのウィンドウに表示されていますが、別々の場所を表示しているかもしれません。バッファkumasan.txtは別のフレームに表示されています。バッファbuhi.txtは対応するウィンドウはありませんが現在編集中です。

■お星様になるバッファ

Emacsではファイル＝バッファかというとそうでもありません。バッファ名が「＊scratch＊」のようにアスタリスクに囲まれていたら、そのバッファはお星様になる、いや捨てられるものと思ってください。つまり、セーブせず

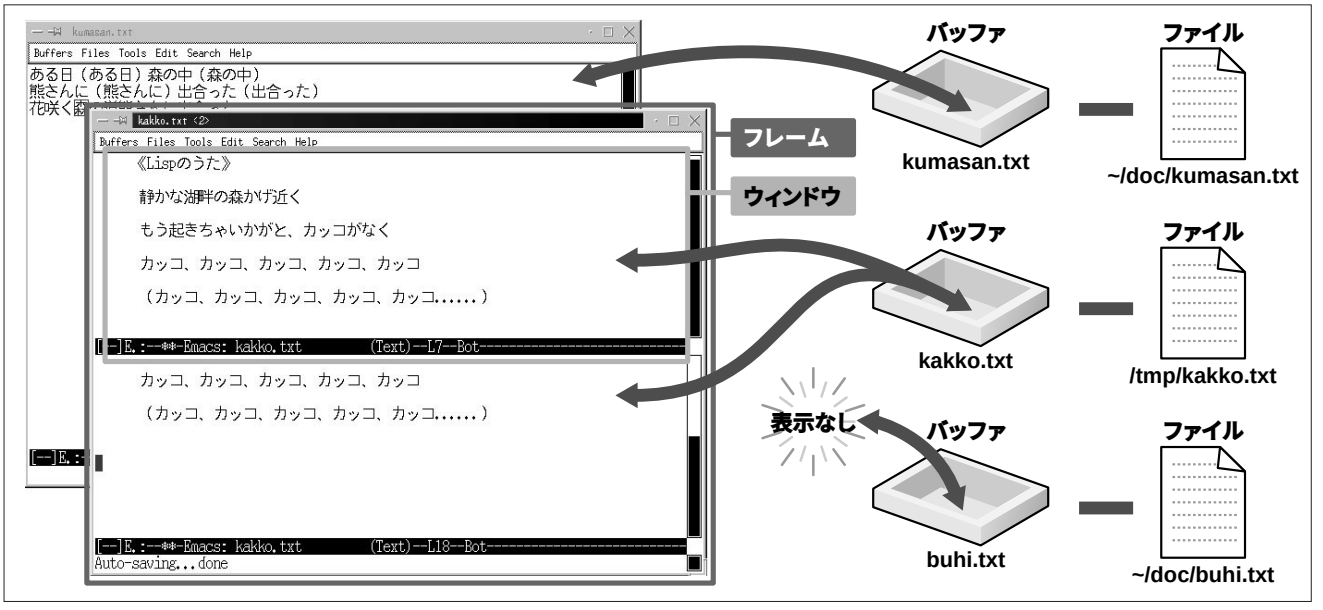


図1 ウィンドウ、フレーム、バッファ、ファイル

に終了しようとしても注意してもらえず（ほかの場合なら注意される）、非常事態が起きても救い出す手段（recover-file、後述）がない、ということです。

とにかく＊scratch＊バッファに意味のあるものを1文字たりとも入力しちゃダメ……というより、Emacsを起動したらまずC-x C-f（find-file）で特定のファイルの編集を始める習慣をつけましょう。

ではなぜ＊scratch＊バッファがあるのかというと、これはちょっとしたelispコマンドの実行のためといわれています。筆者は入力メソッド（かな漢字変換）なしでは字が書けないので、ここを辞書代わりに使っていたりします。

ウィンドウの操作

現在カーソルがあるウィンドウを2分割するには**C-x 2**をタイプします。これは、1つのバッファの同じ場所が上下2つに分割されたウィンドウに表示されるだけです（カーソルは上側のウィンドウにあります）。開いたウィンドウのそれぞれで、スクロールするなり別のファイルを読み込むなり、別のバッファを呼び出すなり、好きな操作ができます。

各ウィンドウにモード行が必要なため、開けるウィンドウの数には限りがあります。あまり開きすぎると画面がシマウマのようになってしまうので要注意です。

■ウィンドウ間を渡り歩く

さて、カーソルを別のウィンドウに移すには……マウスでつついてももちろんかまわないんですが、カッコよくキメるなら**C-x o**（other）でしょう。多数のウィンドウに分割していても、らくらくと移動できるはずです。

■ウィンドウの消去

ウィンドウを閉じて消してしまっても、バッファは消えてなくなるわけではありません。表に出ないだけです。だから、邪魔になったらさっさと消してしまっても心配はありません。カーソルが位置しているウィンドウを閉じるには、**C-x 0**とします。周囲の適当なウィンドウが「う〜狭かった」と言いながら（うそ）べろりん、と広がってきますが、結局Xのウィンドウサイズ（Emacsのフレームサイズ）は変わりません。

■ウィンドウの整頓

こうしてウィンドウの分割と消去を繰り返していると、ウィンドウが気にくわないサイズ（行数）になることがあります。こんなときには**C-x +**とすれば、全ウィンドウが等間隔の幅になります。

ウィンドウの幅を好きな大きさにするには、X上ならウィンドウ間のモード行をつまんで引っ張ってもよいのですが、**C-x ^**とすれば1行ずつ広がり、またしてもカッコよくキメることができます。いらち（意味がわからなかったら最寄りの関西人まで）なあなたは、C-u 6 C-x ^としてみましょう。6行広がります。

画面がシマウマ状態になってしまうと、カーソルがあるウィンドウを残して全部閉じたくなります。そんなときは**C-x 1**です。

■フレームの操作

X上で使っていると、こんなにせせこましくウィンドウ分割するのはイヤだよ、という人もいるかもしれません。ならばC-x 2でウィンドウを分ける代わりに、**C-x 5 2**としてみてください。どうです? 閉じるときはC-x 0の代わりに**C-x 5 0**です。もっとも、1つのフレームにいくつもウィンドウが開いていることにこそ、達人は魅力を感じるのですが……。

バッファとウィンドウの対応付け

新たなファイルを読み込んでそのバッファを表示する場合は、前回説明したようにC-x C-fを使いますが、すでに存在するバッファ（すでに読み込まれているファイルに対応するバッファ）を表示するときはC-x bです。

注意してほしいのは、C-x C-fではディスク上のファイル名を一意に指定しなければいけないため、パス名を~taroh/doc/hoge.txtのように書いてやらなければなりませんが、後者はバッファ名（通常のファイル名と同じになる）でよいため、hoge.txtなどのようになります。どちらでも〈TAB〉キーなどによる補完（第1回参照）が効くため、どんな長い名前でも苦になりません。安心して長いファイル名を付けてください。

C-x C-fの場合でも、指定したファイルと対応しているバッファがすでにある場合は、C-x bと同じ効果になります。またC-x bで指定したバッファがまだ存在しないときは、最初にセーブするときにファイル名を聞いてきます。

では同じバッファ名が2つあったらどうなるでしょうか。やってみればわかりますね。~taroh/tmp/hoge.txt、~taroh/doc/hoge.txt、/var/tmp/hoge.txtなどと異なるパスで同じファイル名を開いていくと、最初のバッファ名はhoge.txtですが、次からはhoge.txt<2>、hoge.txt<3>のように後ろに“<…>”が付きます。

開いているバッファの名前を忘れたり一覧が見たくなったら、**C-x C-b**としてみましょう。自動的にウィンドウが2分割され、他方には＊Buffer List＊というバッファが表示されるはずです。ここではバッファ名のほか、ダーティかどうか、サイズ、モード、結合しているファイル名が表示されます。

このバッファリストに移動してカーソルを上下すれば、かなり長くても全体を見ることができるでしょう。このバッファの面白いところは、ファイル名の上にカーソルを合わせて〈RET〉を押すとそのまま分割したウィンドウ上にバッファが呼び出され、〈1〉をタイプするとC-x 1からの連想からかバッファリストが閉じて選んだファイルの編集にかかれることです。そのほかバッファリストそのものの使い方は、例によって〈?〉をタイプすれば見ることができますから、いろいろ試してみてください。ちなみに、バッファリストもお星様（＊……＊）付きのバッファ名ですから、セーブされることはありません。

ポイントとカーソルとマーク

Emacsでカーソルのことをポイント（point）と呼ぶことは前回も述べました。じつは、カーソルをポイントとそっくり呼び換えているわけではなく、厳密には違いがあります。たとえば1つのバッファを複数のウィンドウ（フレーム）に表示させている場合、ポイントはウィンドウとフレームごとに1つずつあって、バッファ中に複数存在することになります。しかし、カーソルとして表示されている

Column

チュートリアル補足

前回、基本操作はまずチュートリアルを実行してなじみましょう、と書きましたが、Laser5（Emacs20.3.1）やRedHat（Emacs20.4.1）などでは、紹介したC-h-Tコマンドが使えなかったと思います。X上で使用しているときには、メニューから［Help］－［Emacs］－［Tutorial］を選択してください。英語のチュートリアルが表示されるようだったら、［Mule］－［Set Language Environment］－［Japanese］を選択して日本語環境を設定したうえで、もう一度試してみましょう。また、素のEmacs設定状態であれば、M-x set-language-environmentを実行し、“Set language environment（default,English）：”に“Japanese”と入力したうえで、C-h tあるいはM-x help-with-tutorialコマンドを実行しても同じです。

ただし注意すべきことがあります。ディストリビューションによっては使いやすいように独自の初期設定ファイルを用意している場合があります。チュートリアルの説明は何も設定していない「素のEmacs」に関する操作方法なので、説明と操作が一致しない点があります。本記事でもおいおい説明していきますので、とりあえずチュートリアルどおり操作して期待どおりの結果にならなかったら、覚えておいてあとで解明しましょう。疑問と怨恨こそ上達の原動力です。

図2 ポイントとカーソル

のは常に1カ所のみです。当たり前ですね。複数のポイントがあるバッファのウィンドウを閉じると、そのポイントは消えてしまいますが、最後の1つはウィンドウを閉じても（ファイルを閉じてバッファを消してしまわないかぎり）保存され、再度ウィンドウに呼び出したときにはそこから編集を続けることができるのです。

マーク（mark）は、Emacsの「範囲（region）指定コマンド」を実行するうえで、**とっっっても**大切な概念です。範囲指定というのは、マークしたところからポイント位置までの間を範囲として（**図2**）「何か」をすることなのですが、その「何か」がEmacsでは数百種類あります。ここで説明するカット＆ペーストはまだ序の口で、「範囲をスペルチェックする」「範囲を英文和訳する」なんてこともできたりします。

マークは編集中のテキストに付けられた「透明な印」と考えればよいでしょう。マークは**C-SPC**（または**C-@**）で付けられます。無事にマークできればミニバッファには“Mark set”と表示されるものの、なにしろ透明なのでテキスト上で確認することはできません。マークをどこに付けたかわからなくなったら、どうしましょうか。物忘れがひどくなっているせいだけでなく、Emacsでは「自動的にマークが付いてしまう」状況もあって、うっかりしていると予想外の範囲を切り取ってしまい、慌てることがあります（アンドゥ、アンドゥ）。

5秒以上前に付けたマークは賞味期限切れと考えて、範囲指定は直前にマークするという潔い習慣を身につけるのも一法ですが、**C-x C-x**でマークとポイントを交換して確認することも可能です。どういうことかというと、C-x C-xでマーク位置にカーソルが移り、もう一度C-x C-xで確認前の場所に戻ってくるのです。

コピーとカット＆ペースト

切った貼ったはエディタだけでなく、OSのさまざまな局面でも登場します。コピー（copy）とカット（Emacsではdelete）は、範囲指定コマンドです。まず、コピー（カット）範囲の始めをマークしましょう。現在のポイントがコピー範囲の終わりに近ければ、そっちを先にマークしてもよいでしょう。

この状態でマークとは反対側の、範囲の終わり（または始まり）にポイントを移動して、**C-w**とするとカット、**M-w**とするとコピーです。

カットまたはコピーされたブツは、キルリング（kill ring）という、バッファ一覧でも表示されない特殊なバッファに放り込まれます。こうしておいてから、好きなところにポイントを移動して**C-y**とすると、今度はキルリングの中のブツがポイント位置に貼り込まれます。通常はこれをペーストといいますが、Emacsではヤンク（yank）と呼びます。viではヤンクをコピーの意味で使っているのとは対照的です。カットしてペーストすれば移動、コピーしてペーストすれば複写ができます。

なおマウスを使っている場合は、コピーは左ボタンドラッグ、ペーストは中央ボタン（2ボタンマウスの場合は左右同時）クリックで代用できるのですが……何度も言うように、コントロールキーとお友だちになりさえすれば、それに検索なんかを併用してらくらくとポイント移動ができるようになれば、マウスで編集するのなんてかったるくてカッコ悪くてヤんなっちゃいます。

ほかにもカットバッファにブツを放り込む方法は何通りかありますが、よく使われるのは**C-k**です。ポイント位置から行末までをカットしてキルリングに放り込みます。マークは使いません。

検索

Emacsで最もよく使われる検索方法は、インクリメンタル検索です。まず簡単なテキストを用意しましょう（**図3**）。**C-s**をタイプすると、ミニバッファに検索文字列を促がすメッセージが表示されます。

検索する文字列をタイプしていくと、カーソル直後から後ろに向かって**条件に当てはまる文字列**が探し出され、次々とポイントが移っていきます。

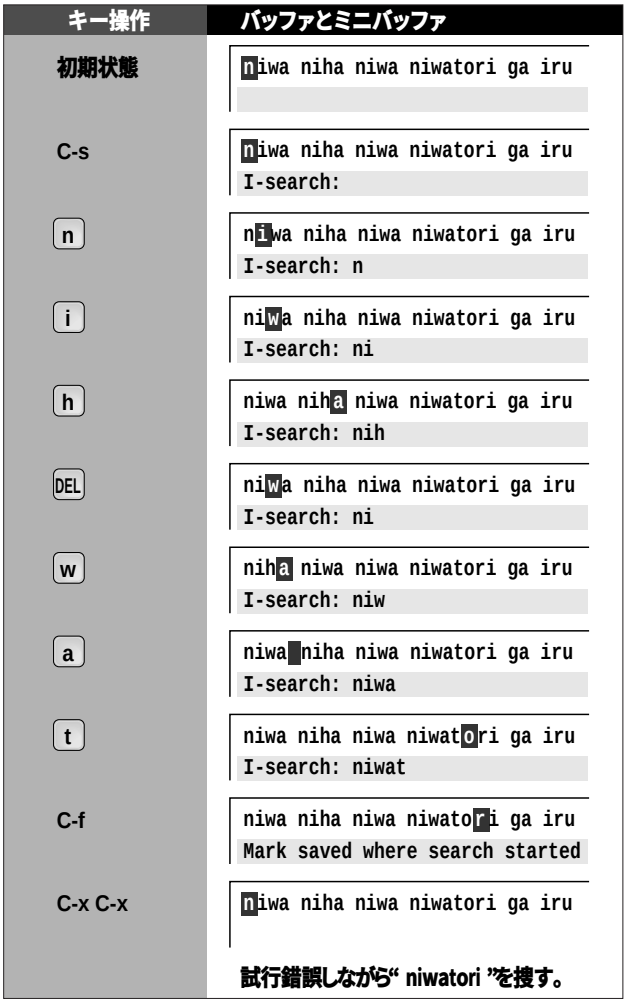


図 3 インクリメンタル検索

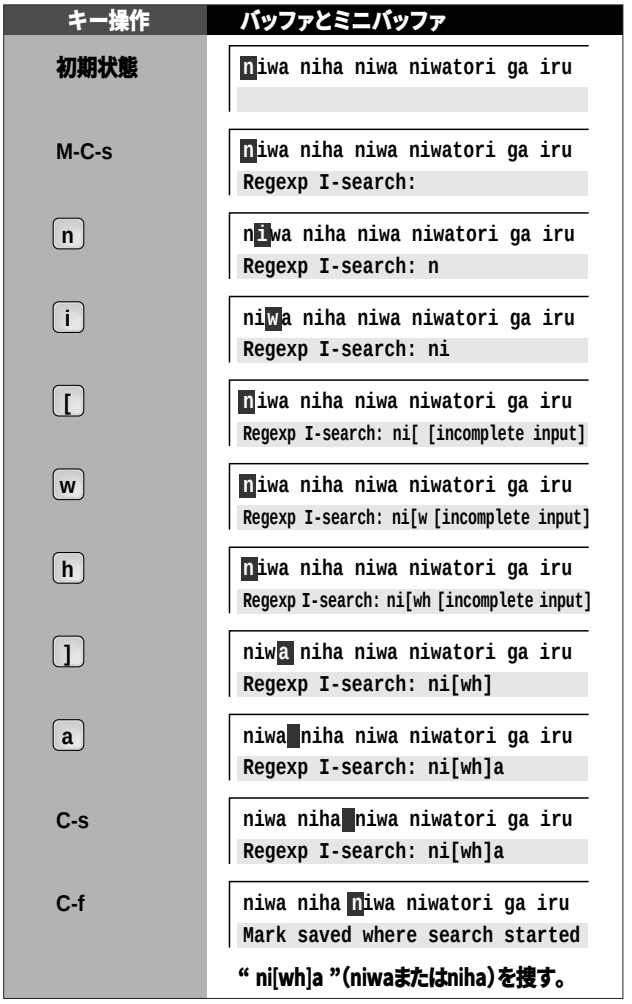


図 4 インクリメンタル正規表現検索

たとえば〈n〉、〈i〉、〈h〉と順にタイプすると、ポイント位置とミニバッファは**図3**のように変化していきます。さてここで“ nih ”ではなく“ niw ”という文字列を探し直すことにしたとしましょう。〈DEL〉をタイプするとミニバッファ中の検索文字列は短かくなり、ポイントはさきほど“ ni ”をタイプしたときと同じ場所に戻ります。

〈w〉、〈a〉、〈t〉とタイプすると、ポイント位置は“ niw ”、“ niwa ”、“ niwat ”にマッチする文字列へと次々と移動します。

目当ての文字列が見つかったところで移動キーをタイプすれば（ここではC-f）、検索モードから脱出します。

このときミニバッファに表示されているように、自動的にマークが検索開始の場所にセットされます（自動的にマークされる一例です）。検索開始前の場所に戻したければ、C-x C-xで戻せます。見つかったものではなく、ずっと後ろにある検索文字列がお目当ての場合は、C-sを何度かタイプしてみましょう。

いかがでしょう。検索する文字列があいまい――っていうか綴りに自信がなければ、消して、タイプして、また消して……ができるため、慣れるとこれなしでは編集ができなくなります。

ここで使った検索は大文字と小文字を区別しません。また、前回と同じ検索を繰り返したければ、C-s C-sで同じ検索文字列（ここでは“ niwat ”）からスタートできますが、さらに文字をタイプすれば検索文字列に追加できるし、〈DEL〉をタイプすれば検索文字列が全部消去されて新たな検索文字列を指定できます。

逆方向の検索は**C-r**（C-r C-r）です。順方向、逆方向のいずれでも、バッファの終わり（先頭）まで到達するとベルが「ぴっ」と鳴って、それ以上マッチする文字列がないことを知らせてくれますが、しつこくC-s（C-r）を叩くと、またバッファの先頭（終わり）から検索を繰り返してくれます。

■固定文字列の検索

日本語の文字列の検索など、インクリメンタルでは具合が悪い場合、あるいはインクリメンタルにする必要がない場合もあります。このようなときは、他のテキストエディタと同様に固定文字列の検索もできます。

図3でC-s（またはC-r）をタイプした状態から、C-kをタイプしましょう。ミニバッファに“＊Enter string...”と表示され、〈RET〉をタイプするまでの文字列が検索対象になります。

■インクリメンタル正規表現検索

正規表現を使った検索は、複数の条件にマッチする文字列検索として大変有効な方法です。Linux上では、viなど他のエディタだけではなく、sedやawkといったツールを使いこなすうえでも非常に重要です。とはいえ正規表現の解説をしていると誌面が尽きてしまいますので、正規表現マニアになりたい方は、Webページで検索してみたり他の本を探してみてください（実はinfoにもRegexpという項目はあるんですね……英語の得意な方はどうぞ）。

インクリメンタル正規表現検索は、M-C-s（逆方向はM-C-r）です。つまり、普通のインクリメンタル検索のキー操作の前に〈ESC〉を押すか、〈Alt〉を一緒に押します。

インクリメンタルに字をタイプしているあいだは、次々と条件にマッチする場所にポイントが飛んでいきます（**図4**）。正規表現が完結していないあいだ、たとえば〈[〉をタイプしてから〈]〉をタイプするまでのあいだは、ミニバッファに警告が出ています。

■置換

じつは、自分ではEmacsの置換というものをあまり使ったことがありません。大量に語句を置換・削除する場合、たいていキーボードマクロ（後述）で済ませてしまうからです。俺って変な奴？　と思って、周りのEmacserたちにも聞いてみましたが、かなりの上級者なのにそんな機能があることすら知らない人もいるありさまです。

置換は範囲指定コマンドではなく、後ろの方向にしか置換できません。1行目に移動してそこからやれ、ということなんでしょうか。なんかやる気のない仕様ですねえ……。

置換操作の開始は**M-%**（つまり〈ESC〉%または〈Shift〉＋〈Alt〉＋5というややこしい操作になります）です。まず置換前の文字列を入力してから置換後の文字列を入力します。該当する文字列が見つかると、置換（y）、スキップ（n）、残り全部を無条件に置換（!）、置換モードから抜ける（q）、などの選択ができます。わからなくなったら〈?〉をタイプしてみてください。

いろいろ便利な機能

■PCキーボードの機能

さんざん小指がつりそうな「C-なんとか」を勧めたあとでナンですが、Emacsが初心者向きと思えるのは、キーボードの刻印にあるような機能の大半がコマンドに割り当てられていることです。カーソルキー〈←〉〈→〉〈↑〉〈↓〉はもちろん、〈PgUp〉〈PgDn〉〈Home〉〈End〉も効きます。変わったところでは、〈Ins〉キーで挿入・上書きモードが切り替えられます（なんか一△郎みたい……）。

ただし、これらのキーはマウスと同様、Emacsに特殊なイベントを渡して処理してもらっているので、いつも使えるとはかぎりません。たとえばTelnetを介して使っているときは、Telnetがイベントを取るのでEmacsにイベントが渡らないことがあります。

■再表示

なんらかの理由で画面表示が乱れたときは、**C-*l***で画面を描き直すことができるといわれていますが、X上のEmacsではほとんど表示が乱れることはありません。この機能はむしろ、今ポイントがある行をウィンドウの中央に持ってくるのに役立ちます。

■ファイルの回復

前回、マメにセーブをするようにお勧めしましたが、Emacsは最近のテキストエディタが採用している自動セーブ機能を昔から備えていました。

Column

余計なお世話だEmacs！

Emacsでは「M-x なんとかかんとか」をタイプして使う機能が山ほどありますが、最近のEmacsはその短縮キー操作法を教えてくれます。たとえばmuleだったら、M-x help-with-tutorial-for-muleとタイプしてリターンキーを押すと、3秒ほど『M-x help-with-tutorial-for-mule』はC-h Tでいいのに、ご苦労さんなこった」（一部脚色）と表示してくれて、ムッとする……いや便利な機能です。だけど、後述のように〈BS〉キーで1文字消去できるようにしていると、ほんとうはC-h TではなくC-H Tなので、表示は間違っていることになります。

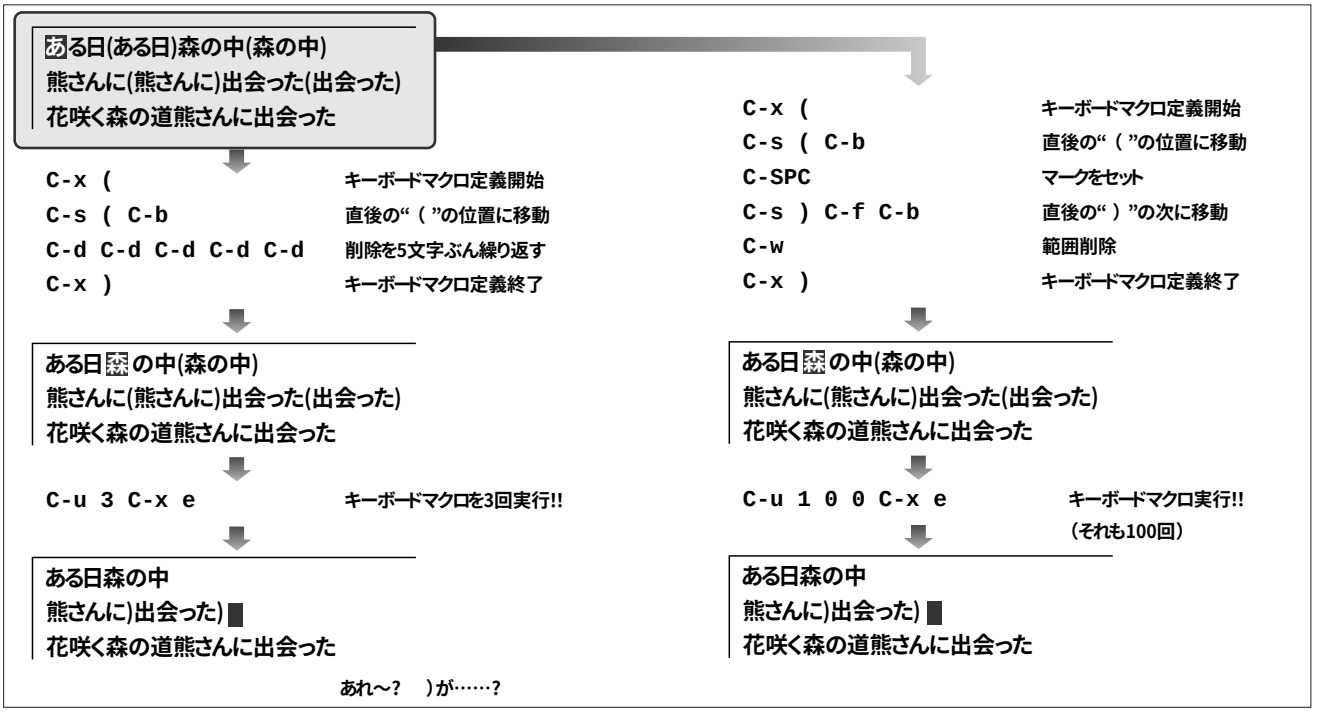


図5 キーボードマクロ

編集中のファイルがあるディレクトリをlsコマンドで覗いてみてください。たとえばhoge.txtというファイルを編集しているとき、#hoge.txt#というファイルができていることがあります。このファイルは、タクシーのメーターのように、一定時間が経過するか、ある程度の文字をタイプすると自動的に作られます。

ほかに、2回以上セーブしたときにはhoge.txt~というファイルができていると思います。これはバックアップファイルといって、セーブしたときに前回セーブしたときのファイルをとっておいてくれるのです。Windowsのアプリケーションでもhoge.bakを作ってくれるものがありますが、それと同じです。ですから、自分でファイルを作るときは末尾に“~”のついたファイル名は避けましょう。

さてファイルhoge.txtがダーティであるあいだ(自動セーブファイルがあるあいだ)にEmacsが無念の死を遂げたとしましょう。次にEmacsでhoge.txtを編集するとき、ミニバッファには“(New file)”の代わりに、“hoge.txt has auto save data; consider M-x recover-file”というアドバイスが表示されます。ここでM-x recover-fileとしてファイル名に“hoge.txt”を入力すると、hoge.txtに対応する#hoge.txt#の大きさや時刻をls形式で表示して確認を求めてくれます。ここで“yes”を入力すれば、ファイルを最後に自動セーブした段階まで戻してくれます。

なおこの機能のために、ミニバッファにときどき“Auto-saving...done”と表示されることがあります。誰もいない部屋で律儀に動き続ける電気冷蔵庫と同じで、ユーザーはとくに(Emacsに謝意を表明するなどの)対応をとる必要はありません。

しまったと思ったら無限アンドゥ、すでにEmacsを終了したあとなら(Emacsはコンピュータの再起動などの必要がないかぎり立ち上げっぱなしにしておくことがお勧めです)recover-file、それもダメならバックアップファイルを覗いてみて、有用そうなら急いで“mv hoge.txt~ hoge2.txt”のようにして回収しましょう。こうしたコツを覚えておけば、Emacsと共に生きていくかぎり、まず悲しい思いをすることはありません。

■繰り返し実行(引数指定)

C-uをタイプしたあと数字を入力して、その後ろのコマンドの実行回数を指定することができます。これは厳密には、コマンドに引数を与えるものです。引数は以下のようなパターンに分類できます。

●**回数指定**――挿入、移動、削除など。たとえば**C-u 72-**でマイナス記号を72個入力したり、C-u 8 〈BS〉で〈BS〉を8回押すのと同じ働きをさせることができます。〈BS〉くらいなら8回押してもよいようなものですが、後述するキーボードマクロと組み合わせると強力です。

●指定しても意味のないもの――たとえばセーブは何回実行しても同じなので、回数を指定しても指定しないときと同じ働きになります。

●別の働きをするもの――任意の引数を付けると別の働きをするものがあります。たとえば検索コマンドのC-sは、引数を指定すると正規表現による検索（M-C-sと同じ）になります。この場合、引数にとくに意味はありません。

●働き方を指定するもの――少数ですが、引数がないと意味のないコマンドもあります。たとえば**C-x f**は自動改行モードで折り返す行の幅をセットするコマンドですが（find-fileのC-x C-fとは違います）、最近のEmacsでは引数が必要です。C-u 72 C-x fとすると72桁で折り返すようになります。自動改行モードにはM-x auto-fill-modeで移れますので試してみてください。モードについてはあとの回で詳しく触れます。

引数のデフォルトはどういうわけか4です。C-uの直後にコマンドを実行すると、C-u 4のあとに実行したのと同じことになります。

■キーボードマクロ

魚が有ると書いてマグロと読みますが（意味なし）、Emacsの場合はあたかも録画して再生するかのごとく、一連のキー操作を自動実行します。

録画の開始はC-x（、録画の終了はC-x ）です。わかりやすいですね。このあいだに行った操作は、C-x eで再生することができます。キーボードマクロは、できるだけ汎用性を持たせて作るのがコツです。たとえば図5のようなテキストからカッコ内を削除したいときは、左のように削除文字数を指定したのでは（カッコ内の文字数が一定ならいいのですが）汎用性のないマクロになってしまいます。右のように検索を活用するのが適当でしょう。

キーボードマクロはC-uと組み合わせると強力です。繰り返しの回数が正確にはわからないときでも、多めに指定してマクロの実行中にベルが鳴るような操作（エラー）があれば実行は中断されます。テキスト全範囲の“(……)”を削除したあと、存在しない“(”を探し求めて永遠に実行するようなことはありません。

マクロは人間のキー操作のぶんが超人的に速くなるだけで、1回1回のコマンドの動作時間は同じです。たとえば、メールの一覧からあるパターンを探し出して別のメールボックスに移動させようとして、10万通もメールが溜まっているメールボックスに対して実行すると（バカですねぇ）、マクロを実行したまま風呂に行って帰ってきてもまだ終わっていない、などということになります。このときでも、C-gをタイプすれば、いつでもマクロの実行を中断できます。

またキーボードマクロの定義中にベルが鳴るような操作が入っていると、そこで定義は中断されます。

次回は……

Emacsの基本操作はこれでおしまい。高度な使い方は回を追って紹介していくつもりです。次回こそは約束のとおり、Emacs/muleを取り巻く日本語入力環境を見ていきましょう。アプリケーションソフトの操作習熟は、なんといっても体育会系。とにかく、たくさん使ってください。そこんとこ、よろしく。

では、happy hacking!!

■

本連載で取り上げてほしいEmacs／muleの話題、ご意見・ご希望などは、アンケート葉書、電子メール（taroh@taroh.org）でお気軽にお寄せください。

Column

ちょっと待ってよEmacs

もうお気づきかもしれません。いくつかの文字をタイプしないと完結しないコマンド、たとえばC-x C-fとかC-h iの途中でタイプする手を休めていると、ミニバッファに途中経過が表示されます。でも、続けてタイプしているとき表示されないのは、なぜでしょうか。

Emacsは、表示してもすぐに失われてしまうような表示を極力避けているのです。複数文字のコマンドの途中で一定時間（普通は約1秒）キー入力がないときだけミニバッファに表示するようになっています。

これを利用して、ミスタイプしたかな？と思ったらタイピングの手を休めて（本当にミスタイプしていたら慌てず騒がずC-g）、途中経過を観察してみるのも一法です。

〈ESC〉xとM-xが等価であることは前回にも触れましたが、この途中経過を観察していると、〈ESC〉をタイプした段階では“ESC”と表示されるのに、xをタイプすると“M-x”に変わってしまいます（謎ですね）。では“ESC”より“M-”のほうが優先的に表示されるのかというとそうでもなくて、C-h bで表示されるキーバインディング一覧を見ると“M-%”（つまり〈Alt〉+〈Shift〉+〈%〉でもよい）は“ESC %”になっていたりと一貫性がありません。

Column

助けてEmacs!! ～ヘルプ機能

Emacsは、help機能が充実していることで知られています。もともとのEmacsでは、C-h（〈BS〉キーを押したときに出る文字と同じ）でヘルプ機能に入れますが、ディストリビューションによっては、〈BS〉で1文字消去ができるように〈BS〉を〈DEL〉と同様の機能にカスタマイズしてある場合があります。前回に、term/bobcat.elマクロを読み込んで〈BS〉を〈DEL〉の機能に入れ替える方法を紹介しましたが、こうした設定をはじめから用意しているわけです。このような場合は、C-H（〈Shift〉と〈Ctrl〉を押しながら〈H〉を押す）でヘルプ機能に入れると思います。試してみてください。1文字削除されずに、いちばん下のミニバッファに、

C-h (Type ? for further options)-

と出たら正解です（ここで中止するならC-gです）。

さてこの状態はhelpのとっかかりで、help-for-helpと呼ばれます。万一〈BS〉でhelpに入れなくてC-Hも効かないようなら、M-x help-for-helpとしても同様です（〈ESC〉xhelp-f〈TAB〉とタイプすれば、ちょっとらく）。

help-for-helpには何種類ものhelpメニューがあります。ミニバッファ行に表示されているように〈?〉をタイプすると、ウィンドウが分割され、以下のような表示になります。要するに、C-h（素のEmacsの場合）のあとにタイプできる文字はa、b、c、f、C-fなどで……という説明なのですが、あることあること（ここで〈SPC〉を押すと、続きも見られます）。さて、ここでは日常よく使うヘルプメニューを説明しておきましょう。

C-h t チュートリアルの開始。日本語、ハングル語、タイ語のチュートリアルは、C-H Tを使うか、あらかじめM-xset-language-environmentを設定したうえで、C-h tを実行する。

C-h b キー操作と機能（elisp関数名）の対応を示す。elisp関数については今後詳しく述べますが、当面は「M-x なんとか」の「なんとか」の部分と考えてください。

C-h k ある操作（キータイプ）で実行される関数名と、簡単な説明を表示する。C-h bで表示される表の逆引きです。

C-h a あるキーワードを含むelisp関数をすべて表示する。

C-h f あるelisp関数の説明を表示する。

C-h i info機能に入る。

▼help機能のオプションの表示

```
You have typed C-h, the help character.  Type a Help option:
(Use SPC or DEL to scroll through this text.  Type q to exit the Help command.)

a  command-apropos.  Give a substring, and see a list of commands
         (functions interactively callable) that contain
         that substring.  See also the  apropos  command.
b  describe-bindings.  Display table of all key bindings.
c  describe-key-briefly.  Type a command key sequence;
         it prints the function name that sequence runs.
f  describe-function.  Type a function name and get documentation of it.
C-f Info-goto-emacs-command-node.  Type a function name;
```

キー操作	elisp関数名	動作
■ファイル操作		
C-x C-f ***filename***	find-file	ファイルをオープン、ウィンドウに呼び出す
C-x C-s	save-buffer	バッファの内容がダーティならセーブ
C-x C-w ***filename***	write-file	ファイルに名前をつけてセーブ
C-x s	save-some-buffers	ダーティなバッファをセーブ
C-x C-c ［y/n［確認］］※1	save-buffers-kill-emacs	終了。ダーティなバッファがあれば、セーブするかどうか確認される
C-x C-k確認	kill-buffer	バッファをセーブせずに消去
C-x C-i ***filename***	insert-file	ポイント位置にファイルを読み込む
■その他の超基本操作		
C-g	keyboardquit	強制中断
C-_	undo	アンドゥ
C-x u	〃	〃
C-h t※2	help-with-tutorial	チュートリアルの開始
C-h T ***lang***	help-with-tutorial-for-mule	***lang***語チュートリアルの開始
C-h i	info	infoを読む
C-h b	describe-bindings	キー操作一覧
C-h k ***key-op***	describe-key	キー操作***key-op***の説明
C-u ***num command***	universal-argument	***num***を後ろの***command***の引数にする
C-l	recenter	画面再表示、カーソルを中央に位置づける

表1 基本操作コマンド一覧表

注1　**y/n**：yまたはnのタイプが求められる。　　**確認**：yes／noの入力が要求される。

注2　C-hは、設定によってはC-HやM-x help-for-helpとしなければいけないことがある。

キー操作	elisp関数名	動作
■カーソル移動		
C-b	backward-char	1文字左へ
C-f	forward-char	1文字右へ
C-p	previous-line	前の行へ
C-n	next-line	次の行へ
C-a	beginning-of-line	行頭へ
C-e	end-of-line	行末へ
C-v	View-scroll-lines-forward	1画面次へ
M-v	View-scroll-lines-backward	1画面前へ
C-l	recenter	現在行を画面中央に表示、画面再表示
M->	end-of-buffer	ファイル末尾へ
M-<	beginning-of-buffer	ファイルの先頭へ
M-x what-line	what-line	現在行の行番号を表示
M-f	forward-word	次のワードへ
M-b	backward-word	前のワードへ
C-x C-x	exchange-point-and-mark	ポイントとマークを交換
■編集		
C-d	delete-char	ポイントの直後の1文字削除
〈DEL〉※3	delete-char	ポイントの直前の1文字削除
C-〈SPC〉	set-mark-command	マークをセット
C-@	〃	〃
C-q ***character***	quoted-insert	***character***を挿入
C-o	open-line	ポイント位置で改行
〈TAB〉／C-i	insert-for-tab-command	タブ
■コピー・カット&ペースト		
C-w	kill-region	範囲の削除、キルリングへ
M-w	kill-ring-save	範囲のコピー、キルリングへ
C-y	yank	キルリングの内容をペースト
C-k	kill-line	行末まで削除、キルリングへ
■検索・置換系		
C-s	isearch-forward	後方検索
C-r	isearch-backward	前方検索
C-s C-k	—	固定文字列の後方検索
C-r C-k	—	固定文字列の前方検索
M-C-s	isearch-regexp-forward	正規表現による後方検索
M-C-r	isearch-regexp-backward	正規表現による前方検索
M-%	query-replace	後方置換
■ウィンドウ、フレーム操作		
C-x 2	split-window-vertically	ウィンドウを分割
C-x o	other-window	直下のウィンドウにカーソルを移動
C-x 0	delete-window	ウィンドウを削除
C-x b ***buffer***	switch-to-buffer	バッファ***buffer***をウィンドウに呼び出す
C-x C-b	list-buffers	バッファの一覧表示と操作
C-x ^	enlarge-window	ウィンドウを1行広げる
C-x +	balance-windows	ウィンドウを等間隔にする
C-x 5 2	make-frame-command	新しいフレームを作る
C-x 5 o	other-frame	次のフレームに移動
C-x 5 0	delete-frame	フレームを削除
■その他の操作		
C-(	start-kbd-macro	マクロ定義の開始
C-)	end-kbd-macro	マクロ定義の終了
C-x e	call-last-kbd-macro	マクロ定義の実行
C-x f	set-fill-column	自動折り返し位置の設定
M-x auto-fill-mode	auto-fill-mode	自動折り返しモードのON／OFF

注3　〈DEL〉は、設定によっては〈BS〉／C-hで代用できる。〈TAB〉の動作はモードによって異なる。

Linux日記

第7回　ファイルシステム（後編）

前回は、i-nodeやリンクを中心にLinuxのファイルシステムを説明しました。今回は、ディレクトリとハードリンクの関係、ファイルシステムのマウントの仕組み、fsckの仕事を説明しましょう。

文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

この連載のタイトル、「Linux日記」ではなくて、「UNIX雑談」でいいんでないのと編集氏には言ってみたのだが、日記でいいんじゃないですかとのことだった。なので、まだしばらくこのタイトルが続くことになるだろう。

一応日記なので、少し近況を書こう。以前に書いたダイヤルアップ接続のトラブルに懲りて、12月に専用線接続を申し込んだ。世間で通信料がどんどん安くなり、従量制でないシステムが増えていこうというときに、何をいまさら専用線と思う人もいるだろうが、筆者の場合はそうもいかないのである。仕事柄、DNSとかメールとかのサーバを動かさなければならないからだ。現在の定額制料金は、クライアントとして端末を接続する場合を想定しており、自前でサーバを用意する環境には不向きなのである。とはいっても、まっとうな専用線接続環境はお金がかかりすぎるので、申し込んだのはOCNエコノミーである（同等のサービスがいくつかあるが、その中からOCNエコノミーを選んだことに特に理由はない）。OCNエコノミーは回線速度保証がないが、うちの用途はメールが中心で、トラフィックの多いWebサーバを動かそうなどという予定はないので、特に問題はないだろう（と思っている）。

ここまで書くと、今の話題の次はインターネット接続のための話になるのが自然だが、例によってそうは問屋が卸さない。DNS、メールサーバは、FreeBSDを使う予定だからだ。筆者は、System V系よりも、BSD系のほうが付き合いが長く、好みもBSDである。ゆえに、業務用のインターネット接続環境はまずBSDで構築する。環境が安定したら、Linuxも組み込もうかと思っている。そうしたら、この記事に書く話題もあるだろう（別に、BSDでもLinuxでも、やることはほとんど変わらないのだが）。

前回はファイルシステムの話を始めた。そして、ハードリンクの話の途中で終わってしまった。普通は、ファイルの構造の話をしたら、そのあとはパーミッションとかオーナーとかの話になるのだが、本連載ではそんな話は飛ばして、ハードリンクの話を始めた。これは、ディレクトリの実装を理解するうえで、ハードリンクが欠かせない要素だからだ。

■ディレクトリとハードリンク

ハードリンクは、ディレクトリの実装に欠かせない機能である。それゆえ、viとexがハードリンクでなくなっても、ハードリンクを知っているユーザーが絶滅しても、ハードリンク機能は使われ続けるだろう（ファイルシステムが完全に改変されたらなくなるかもしれないが）。

ディレクトリには、「.」と「..」という特殊なディレクトリファイルがある。これがそれぞれ、カレントディレクトリ、親ディレクトリを表すものであるということは前回で示した。実はこれらは、ディレクトリファイルへのハードリンクなのである。

前回のlsの例をもう1回示す（**リスト**

1)。今回はlsに、見慣れないiというオプションがついている。iは、i-node番号を表示するオプションである。各行の先頭の数値がi-node番号だ。そして、パーミッションの後の数字は、リンクカウントである。ディレクトリdir1は、i-node番号135169、リンクカウントは3である。このディレクトリのi-nodeは、最初のls（dir1がカレントディレクトリ）では「.」、2番目（dir1の子ディレクトリの内容）では「..」、3番目（dir1の親ディレクトリの内容）ではdir1として参照されている。これら3つのディレクトリは3カ所から異なる名前でハードリンクされているが、i-node番号を見てもわかるように、実体は同じものである。そして、本来の名前が使われているのは、親ディレクトリの中だけで、あとは「.」、「..」という定型的な名前が使われている。

▼リスト1

```
$ cd dir1
$
$ ls -ali          カレントディレクトリ（dir1）の内容。
total 8
 135169 drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 .
 133121 drwxr-xr-x   4 masa     users        1024 Dec  9 04:40 ..
 139265 drwxr-xr-x   2 masa     users        1024 Dec  9 04:41 dir3
 135170 -rwxr-xr-x   1 masa     users        4171 Dec  9 04:41 file2
$
$ ls -ali dir3     子ディレクトリ（dir3）の内容。
total 2
 139265 drwxr-xr-x   2 masa     users        1024 Dec  9 04:41 .
 135169 drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 ..
 139266 -rw-r--r--   1 masa     users           0 Dec  9 04:41 file3
$
$ ls -ali ..       親ディレクトリ（..）の内容。
total 5
 133121 drwxr-xr-x   4 masa     users        1024 Dec  9 04:40 .
  36865 drwxr-xr-x   4 masa     users        1024 Dec  9 04:39 ..
 135169 drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 dir1
 137217 drwxr-xr-x   2 masa     users        1024 Dec  9 04:40 dir2
 133122 -rw-r--r--   1 masa     users          90 Dec  9 04:40 file1
```

ディレクトリの移動はcdコマンドを使う。子ディレクトリに移動する時はその名前を指定し、親に戻る時は「..」を使う。ほとんどのユーザーは、親に戻るためにおまじないのように「cd ..」とやっているだろうが、これで本当の意味がわかったのではなかろうか。「..」は、おまじないではなく、本当に親ディレクトリそのものなのである。

また、「.」も重要である。カレントディレクトリに存在するファイルにアクセスするといった場合は、ファイル名だけを指定すればすむが、カレントディレクトリそのものにアクセスする必要がある場合は、「.」を使うことになる。lsなどでは明示的に「.」を指定

Column

1つのプログラムに複数の名前を付ける

リンクの用途のひとつに、同じプログラムに複数の名前を割り当てるというものがある。通常、プログラムの動作モードはオプションで指定するのだが、この指定をプログラム名でやってしまおうというのだ。これをやっている有名なものとして、viがある。viがexというエディタのビジュアルモードであるということは知っているだろう。exをviという名前で起動すると、exがビジュアルモードで起動するのである。

リストを見ての通り、viもexもvimというファイルへのシンボリックリンクである（Red Hat Linux 6.0 英語版の場合）。つまりviもexも実体は同じということだ。似たような例として、mailqが実はsendmailであるとか、[がtestであるといったものがある。また、wとuptimeももともとは同一プログラムだったのだが、Linuxの実装では、別プログラムになっていた。

同一パーティション中であれば、かつては、このようなリンクにはハードリンクを使っていたが、現在はほとんどシンボリックリンクを使っているようだ。

▼リスト

```
$ cd /bin
$ ls -l vi ex vim
lrwxrwxrwx   1 root     root            3 Jan  4 02:56 ex -> vim
lrwxrwxrwx   1 root     root            3 Jan  4 02:56 vi -> vim
-rwxr-xr-x   1 root     root      1243404 Sep 16  1998 vim
```

する必要はほとんどないが、ファイルのコピーや移動などのコマンドでは、カレントディレクトリを明示的に指定するために、引数として「.」を指定しなければならないことが多い。

■ファイルシステムのマウント

通常のUNIXシステムは、いくつかのファイルシステムを結合して、全体のツリー構造を組み上げている。今まで話してきたことは、個々のファイルシステム上におけるディレクトリのリンク関係である。では、あるツリー位置に別のファイルシステムをマウントすると、これらの関係はどうなるのだろうか？　サブツリーを構成する1つのファイルシステムのルートディレクトリは、ツリー中のあるディレクトリ（マウントポイント）にマウントされる。Red Hat Linux 5.2のシステムでちょっと実験してみよう。/mntの下にtestというディレクトリを作り、そこに別のファイルシステム（ちょっとumountして拝借した/bootである）をマウントしてみる（**リスト2**）。

▼リスト2

```
# cd /mnt
# mkdir test
#
# ls -lai          testのi-node番号は14058である。
total 5
  26105 drwxr-xr-x   5 root     root         1024 Dec  9 08:54 .
      2 drwxr-xr-x  16 root     root         1024 Oct  1 12:48 ..
  28113 drwxrwxr-x   2 root     root         1024 Oct 10  1998 cdrom
  30121 drwxrwxr-x   2 root     root         1024 Feb  7  1996 floppy
  14058 drwxr-xr-x   2 root     root         1024 Dec  9 08:54 test
# mount /dev/sda1 /mnt/test        testにファイルシステムをマウントする。
#
# ls -lai          testのi-node番号が2に変わった。
total 5
  26105 drwxr-xr-x   5 root     root         1024 Dec  9 08:54 .
      2 drwxr-xr-x  16 root     root         1024 Oct  1 12:48 ..
  28113 drwxrwxr-x   2 root     root         1024 Oct 10  1998 cdrom
  30121 drwxrwxr-x   2 root     root         1024 Feb  7  1996 floppy
      2 drwxr-xr-x   3 root     root         1024 Oct  1 13:01 test
# cd test          testに移動
#
# ls -lai          「..」のi-node番号に注目
total 810
      2 drwxr-xr-x   3 root     root         1024 Oct  1 13:01 ./
  26105 drwxr-xr-x   5 root     root         1024 Dec  9 08:54 ../
          :
          :
     11 drwxr-xr-x   2 root     root        12288 Oct  1 12:47 lost+found/
     19 -rw-------   1 root     root        10752 Oct  1 13:01 map
```

最初に作ったtestのi-node番号は14058である。これは/（ルート）ファイルシステム上に作られたtestディレクトリファイルを指している。ここに別のファイルシステムをマウントすると、i-node番号が2に変わった。i-node番号2は、各ファイルシステムのルートディレクトリの「.」と「..」から指されている。つまりこの2は、/mnt/testにマウントしたファイルシステムのルートディレクトリの2なのである。また、よく見るとtestのタイムスタンプ、リンクカウントも変わっている。マウントしている時はサブツリーのルートディレクトリの情報を示し、マウントしていない時は/mntディレクトリに作られたtestディレクトリの情報を示して

Column

マウントポイントの下にファイルを隠す

ファイルシステムをマウントすることで、マウントポイント以下は不可視になるので、このような位置にファイルを置くことは、普通は意味がない。しかし、この機能をうまく使うこともできる。

/varを別ファイルシステムにすることが多いが、そのマウントポイントの下に、マウントする/var以下と同じディレクトリツリーを作っておくなんてことができる。これにより、何らかの理由で/varがマウントできなくても、取りあえず各種デーモンを動かすことができるのだ。システムクラッシュ時には、/varが被害を受ける確率が高い。/varをマウントできなくても、その下にディレクトリがあれば、作業用のディレクトリが見つからないというエラーを防ぐことができる。もっとも、後でそのデータを本来の/varに戻さなければならないのであれば、かえって地獄を見ることになるが。

もう少し実用的な用途として、カーネルのリコンフィグ環境を2種類用意するというものがある。標準の/usr/srcの下には、バイナリパッケージだけを置いておき、そこに/usr/srcファイルシステムをマウントすると、ソースディストリビューション環境に置き換わるといった使い方である。

いるのである。さらに、/mnt/test中の「..」ディレクトリも、/mntディレクトリの「.」の情報を示していることがわかる。

すなわち、マウント処理によって、マウントされたファイルシステム（/mnt/test）のルートの「.」ディレクトリが、マウントポイントのディレクトリを置き換え、そしてマウントされたファイルシステムのルートの「..」は、マウントポイントのディレクトリの親ディレクトリを参照するようになるのである。これで、親のいなかったルートディレクトリに親ができ、しかもその名前はその親ディレクトリ中で定義されることになる（**図1**）。また、マウントポイント以下にもともと存在したサブツリーは、マウント処理によって、不可視になる（もとのディレクトリにcdしたり、その内容を参照することはできない）。

このような処理によって、マウントポイント近辺のディレクトリのi-node番号は、必ずしも自身のファイルシステムのi-node番号とは限らないということになる。

■fsckとlost+found

ファイルシステムの話のついでに、lost+foundというディレクトリについて説明しておこう。Linuxのファイルツリーは、いくつかのファイルシステムから構成されるが、各ファイルシステム（ext2ファイルシステム）のルートディレクトリに、必ずlost+foundというディレクトリがある。普通は、中を見ても空っぽである。これは何に使われるディレクトリなのか？

lost+found、つまり、失われて見つかったというような名前であるが、これはfsckが使うディレクトリである。まずはfsckについて簡単に紹介しておこう。

UNIXはシステムの起動時に、fsckというプログラムを使ってファイルシ

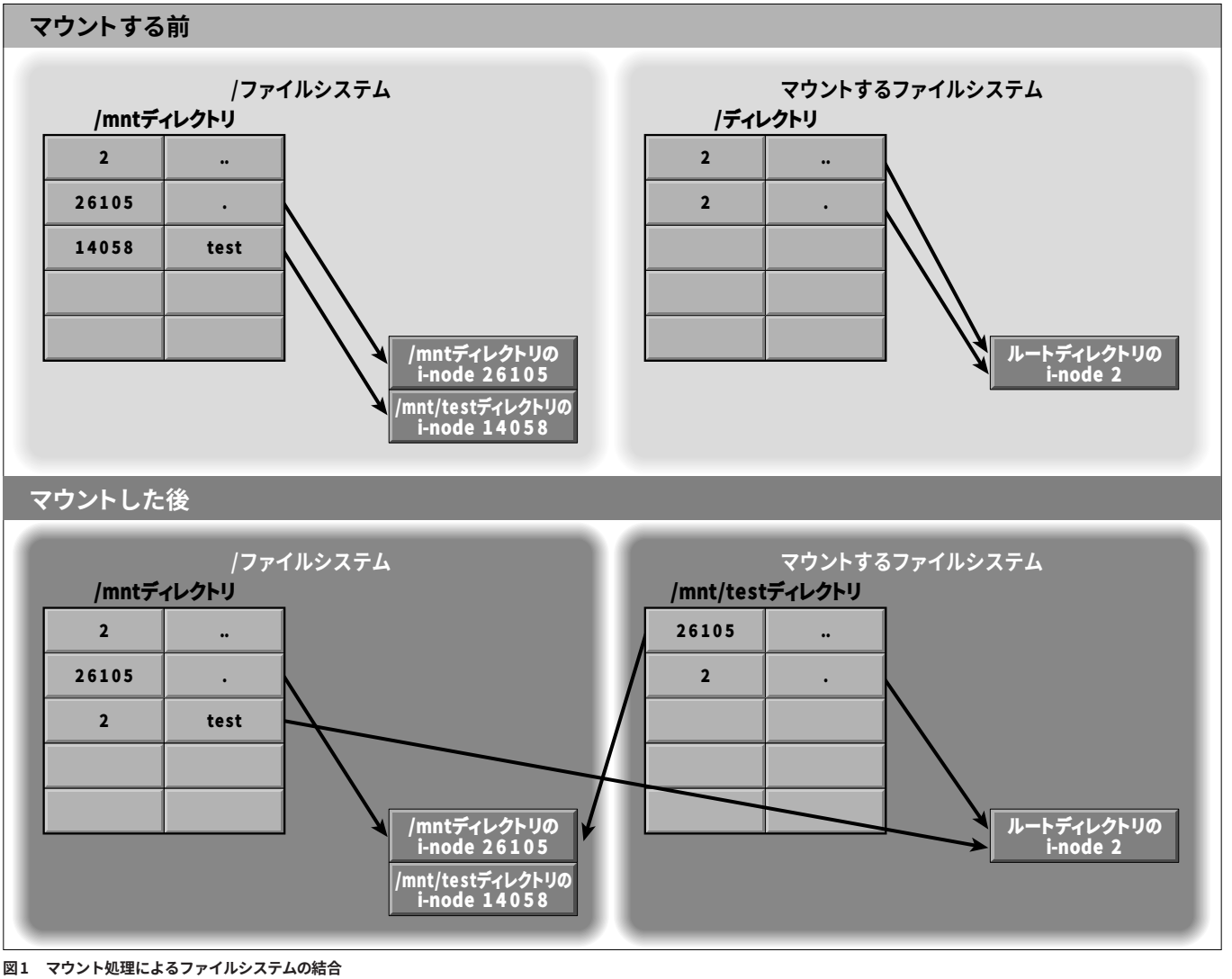


図1　マウント処理によるファイルシステムの結合

ステムのチェックを行う。システムがshutdownコマンドで正常に終了されていれば、ファイルシステムは整合性を維持した状態であり、このチェックでひっかかることはない。しかし、バグによるクラッシュ、停電などによる異常終了の場合は、ファイルシステムの整合性に問題が生じることがある。fsckはこのようなファイルシステムの異常を検出し、可能な限り修復するツールである。ファイルシステムに異常があるままシステムを起動し、運用すると、障害が拡大し、さらにひどいことになる可能性があるため、システムの起動の前に検査、修復するのである。

まず、システムが異常終了すると、なぜファイルシステムに障害が発生する（ことがある）のかを考えてみよう。まず思い付くこととして、ディスクへの書き込み途中でシステムが止まってしまうということがある。データの書き込み途中であれば、そのデータファイルの内容は不完全なものになってしまう。これがもしディレクトリの書き込み途中であれば、そのディレクトリの内容はめちゃくちゃになってしまうだろう。

このような障害は、たとえディスクの書き込み中でなくても発生する。UNIXのディスクI/Oは非同期書き込みを行っているからだ。UNIXのディスク書き込み処理は、2段階に分けて行われる。ファイル出力のシステムコールでは、システムは、ディスクブロックの内容のコピーを収めたメモリ上のバッファに対して書き込みオペレーションを行う。そして、バッファメモリへの書き込みを行った時点で、システムコールはリターンし、プログラムは処理を続行する。これによりプログラムは、実際のディスクアクセスを待つことなく、処理を続けることができる。バッファに書き込まれたデータは、以後適当なタイミングでディスクに書き出される。このような2段階の処理は、システムの効率を大きく改善するというメリットはあるが、突然のクラッシュに対しては欠点となる。バッファに書き込んでから、実際にディスクに書き込むまでにタイムラグが存在するからだ。この間、システムが想定しているファイルシステムの内容と、実際のディスク上のファイルシステムの内容が食い違っているのである（このような状態を、ダーティという）。もしこのタイミングでシステムがクラッシュすると、バッファの内容をディスクに反映することができなくなってしまう。その結果、書き込み中のクラッシュと同じように、ファイルシステムに障害が発生してしまうことがあるのだ。

どのブロックの書き込みが完了しなかったのか、あるいは、ファイルシステムのどの部分がダーティなままシステムが止まってしまったのかにより、ファイルシステムにさまざまな障害が発生する。運が良ければ、構造的な不整合は発生せず、書き込みでオープンしていたファイル内容が破損する程度で済む。この場合、fsckはファイルの長さの修正などを行い、整合を取る。

ディレクトリファイルが破損した場合は、かなり不運といっていいだろう。たとえばあるディレクトリファイルが完全に失われてしまった場合、ファイルシステムはどのようになるだろうか？　ルートからディレクトリをスキャンしてファイルをチェックした場合、失われたディレクトリ以下のファイルはスキャンされない。一方、i-nodeテーブルを順に調べていくと、ファイルツリー中のどこにも存在していないファイルやディレクトリが発見される。失われたディレクトリの下にあったファイルやディレクトリである。もはやツリー上に存在していないこれらのファイルは、そのままでは決してアクセスできない。ここでlost+foundの出番である。fsckは、このような孤児のファイルをサルベージし、lost+found以下に収容する。つまり、lost+foundディレクトリ中にファイルエントリを作成し、孤児ファイルのi-node番号を指すようにするのである。これにより、これらのファイルをアクセスできるようになる。ただし、たとえファイルの内容や属性情報が失われていなくても、ファイル名はわからなくなってしまう。ファイル名は親ディレクトリ中で規定されるが、そのディレクトリがなくなってしまっては、名前を調べるすべはない。そのため、lost+found中に作成されるファイルエントリ名は、機械的に生成された名前になる。

Column

サルベージしたファイルをどうするか

lost+found中にサルベージされたファイルは、機械的な名前を付けられているので、それを元の位置に戻すためには、まずそのファイルの正体を調べなければならない。ファイルの属性や所有者、タイムスタンプなどに基づいて類推するのはかなり大変である。テキストファイルは内容を調べることでかなり類推できるが、バイナリファイルはお手上げだ。

バイナリファイルの正体を調べるには、fileコマンド、stringsコマンドが便利である。fileコマンドは、ファイルのマジックナンバーなどに基づいて、そのファイルの種類を識別する。stringsコマンドは、バイナリファイル中の文字列と思われる部分を表示する。

lost+foundは、ファイルシステムの構築時に、i-nodeテーブルやルートディレクトリと共に作成される。また、fsckがファイルをリンクしやすいように、空のディレクトリであるにも関わらず、最初からある一定のサイズにまで拡張されている（既存のディレクトリファイルの書き換えは簡単な作業だが、ファイルの拡張を伴うとなると、処理はかなり面倒だからだ）。

使っていないからといって、lost+foundを消さないこと。後で痛い目にあう可能性がある。また、/tmpをクリーンアップするスクリプトなどで、間違って消さないように注意すること。

■LEGO MindStorms

誌面が少し余ったので、完全に趣味の話。先日、本誌の見本誌が家に届いた。ぱらぱら眺めていたら、MindStormsの記事が載っていた。何を隠そう、筆者もMindStormsを持っている（システム管理者やプログラマーのMindStorms所有率はかなり高いらしい）。以前からLEGOが好きで、10年位前からテクニックシリーズを中心にかなり集めているのだ（遊ぶ暇がないのが悲しい）。そこで、MindStormsファンが興味を持ちそうなパーツをいくつか紹介しよう。これらは、かつて販売されていた（一部は今でも販売されている）テクニックシリーズ（あるいは拡張パーツセット）に入っていたパーツ類である。

・ギヤボックス

入力軸と出力軸が同一軸線にある減速比約20のギヤボックスと、入力軸と出力軸が直交する減速比約20のギヤボックスである。この減速比を通常のギヤで実現しようとすると、かなりの空間が必要になるだろう。もちろん、駆動ロスも少ない。

・デフギヤ

自動車のセットにはいっていたデフギヤで、MindStormsのデフギヤと異なり、外周のギヤがクラウンギヤになっている。

・旋回ギヤ

クレーンやパワーショベルなど、大きな構造物を旋回させるためのベースとして使える大型のギヤである。普通のシャフトとギヤの組み合わせよりかなり強度がある。

・ジンバル

ヘリコプターのメインローターに使われるジンバル部品である。外部からロッドで偏向させられる回転軸だ。

・ユニバーサルジョイント

これは等速ジョイントではないので、大きな角度で曲げるときは注意が必要になる。

・テクニックシリーズの以前のモーター

テクニックシリーズでかつて使われていた9Vモーターで、ケーブルはMindStormsと同じであるが、ギヤは内蔵されていないので、減速が必須である。

・プログラマブルシーケンサー

3個のモーターを制御し、シーケンスを記憶できる。入力ポートはないので、高度なことはできない。

・チェーン

ギヤにかみ合うチェーンである。これが欲しい人は多いだろう。

・キャタピラー

プラスチック製なので、非常によく滑るが、長さは自由にできる。これもギヤにかみ合う。ベルトコンベアーとかにも使える。

・フレキシブルシャフト

自転車のブレーキに使われてるのと同じようなもの。往復運動を自在に伝達できる。

・プロペラ

飛行機に使われていた部品であるが、MindStormsでは使い道はないかも。

・タイヤ

何の変哲もないように見えるが、車軸部分がポイント。レゴでは、軸に対して自由に回転できる大型のタイヤは珍しい。小さいタイヤは、レゴシステムの飛行機などに使われているもの。ターンテーブルと組み合わせれば、コンパクトなキャスターを作れる。

・エアーシリンダーとバルブ

エアーシリンダーには2種類の系列がある。1系統の正圧と負圧で動く単動タイプと、2系統の正圧で動く複動タイプである。電動エアーコンプレッサーを作るためのシリンダーもある。RCX制御の電動バルブがあれば、かなりおもしろいことができるだろう。

・ダブルウィッシュボーン用部品

ナックルアームの上下、タイロッド接続部がボールになっているのがミソである。ユニバーサルジョイントと組み合わせれば、サスペンション付きの転舵可能な駆動輪を作れる。

・人形

GIジョーのように各関節が動く。この人形専用の椅子とかスキー、ヘルメットなどもある。

賢く使うUNIX

これであなたもスマートなUNIX使い！

―シェルスクリプトを書こう(前編)―

Linuxの標準シェルであるbashのコマンドラインを中心として、Linuxの便利な使い方について紹介していく本連載。今回からは、bashのシェルスクリプトについて取り上げ、実用的なシェルスクリプトを作成するためには必須のif、whileといった制御構文やさまざまな組み込みコマンドなどについて解説する。

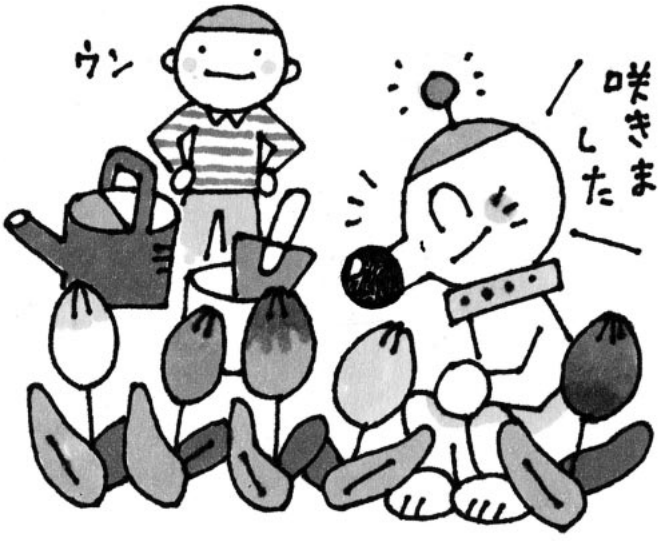


Illustration : Manami Kato

今月のお題

指定したディレクトリ以下に存在する全ファイルから特定の文字列を含む行を探し、ファイル名と行の内容を表示するスクリプトを作成する

文：**大池 浩一**
Text : Kouichi Ooike

これまで、bashのコマンドライン上で複数のコマンドをパイプで接続したり、ifやforといった制御構文を使用することでいろいろな処理を行ってきた。しかし、処理を行うコマンドラインの内容が長くなるにつれ、いちいちキー入力したり、ヒストリーから呼び出して修正したりするのが大変になってくる。

そこで、今回からはbashのもうひとつの使い方である「シェルスクリプト」について取り上げる。シェルスクリプト（単に「スクリプト」と書くこともある）には、実行したいコマンドなどが記述されており、bashがその内容を読み込んで、コマンドラインで入力したのと同様に実行してくれる。複数のコマンドを順次実行するだけでなく、条件判断や繰り返しなどの制御構文を駆使して、プログラム的な処理を行うことも可能だ。

さらに、こうしたシェルスクリプトに実行属性を与えると、C言語などで作成されたコマンドと変わることなくコマンドラインで実行でき、パイプやリダイレクトも利用可能だ。つまり、既存のコマンドを組み合わせるだけで新しいコマンドを作成できるわけだ。実際、あなたが使っているコマンドの中にも、いくつかシェルスクリプトが含まれている。たとえばX Window Systemを起動する際に使うstartxなどがそうだ。

コマンドのように使えるシェルスクリプトを書く

シェルスクリプトをコマンドらしく動作させるには、コマンドラインの内容をそのままファイルに書いただけでは不十分で、いくつか注意すべき点がある。たとえば、コマンドラインでは処理の対象となるファイル名を直接指定したが、シェルスクリプトの場合は、スクリプト実行時に指定されたファイル名を何らかの方法で参照しなければならない。また、ファイル名がひとつも指定されていない場合にも対処する必要がある。

今回は、コマンドラインで実行していた内容をシェルスクリプトにしていく過程を順を追って説明する。実行属性の与え方やスクリプトの実行方法、引数の参照のしかた、条件判断の使い方などだ。今回のお題では、連載の初回に作成した「指定したディレクトリ以下に存在する全ファイルから特定の文字列を含む行を探し、ファイル名と行の内容を表示する」コマンドラインをシェルスクリプト化し、

オプションを解釈して動作を変更する方法について説明する。さらに次回では、配列や関数、数値演算などbashの機能を駆使して、よりプログラムらしいスクリプトを作成する予定だ。

単なシェルスクリプトを作成する

最初に、指定したコマンドのフルパスやファイルサイズなどの詳細な情報を表示することを考えよう。これまでのようにコマンドライン上で実現するには、以下のようにすればいい。

```
$ ls -l $(which rpm)
```

まず、$(…)内に記述されたwhichが実行され、指定したコマンドrpmのフルパス「/bin/rpm」が得られる。続いて、コマンド置換により$(…)部分が「/bin/rpm」で置換された「ls -l /bin/rpm」が実行される。その結果、rpmに関する詳細な情報が表示されるというわけだ。

上のコマンドラインと同様の処理を行うシェルスクリプト「whichcmd」を**リスト1**に示す。テキストエディタなどを使って作成しよう。

スクリプトをbashで実行するには、2つの方法がある。1つは、組み込みコマンドsourceの引数として「source スクリプト名」とする方法。もう1つはスクリプトに実行属性を与えてコマンドラインで直接実行する方法だ。以下では後者の方法を利用する。

ファイルに実行属性を与えるには、chmodを+xオプションとファイル名を指定して実行すればいい。

```
$ chmod +x whichcmd
```

これ以降、whichcmdがカレントディレクトリにあるなら以下のようにして実行できる。

```
$ ./whichcmd rpm
```

先頭に「./」が必要なのは、コマンド検索パス(PATH)にカレントディレクトリが含まれていないからだ。一方、PATHに含まれるディレクトリ(/usr/local/binや~/binなど)にwhichcmdを移動しておけば、

▼リスト1 初期版のwhichcmd

```
1: #!/bin/bash
2: # usage: whichcmd COMMAND
3: ls -l $(which $1)
```

※実際は行番号は付かない

▼リスト2 複数の引数に対応したwhichcmd

```
1: #!/bin/bash
2: # usage: whichcmd COMMAND...
3: ls -l $(which $*)
```

※実際は行番号は付かない

```
$ whichcmd rpm
```

とスクリプト名だけで実行できる。

それでは、**リスト1**の内容を解説しよう。1、2行目はコメントだ。シェルスクリプトやPerlなどのスクリプトでは、「#」以降から行末までの内容はコメントとして無視される。ただし、1行目の「#!」には特別な意味があり、Linuxのカーネルはこの行を参考にしてスクリプトを実行するプログラム（この例では「/bin/bash」）を探す。

結局、スクリプトの実体は3行目の「ls -l $(which $1)」だけだ。コマンドラインとの違いは、具体的なファイル名の代わりに「$1」が使われていること。この部分は、実行時に指定した最初の引数の内容で置換される。たとえば、「whichcmd rpm」とすると、「$1」が「rpm」で置換された「ls -l $(which rpm)」が実行される。

正確には、「$1」は1番目の引数を保持する位置パラメータ「1」の参照だ。シェル変数hogeの内容を「$hoge」として参照するのと同じだが、位置パラメータは参照のみ可能という特殊な組み込み変数なので、「1=rpm」として内容を変更することはできない。

同様に、2番目以降の引数の内容を保持する位置パラメータ「2」…も用意されており、「$2」…で参照できる。さらに、スクリプト名を保持する「0」や、位置パラメータの数を保持する「#」、すべての位置パラメータをスペース区切りで並べた内容を保持する「*」などの組み込み変数も用意されている。

リスト1では、2番目以降の位置パラメータを利用してい

ないため、実行時に複数の引数を指定しても、最初に指定したコマンドの情報しか表示されない。スクリプトで使用しているwhichやlsは、もともと複数の引数に対応しているので、「$1」を「$＊」に書き換えるだけで複数の引数に対応できる(**リスト2**)。

条件判断を含むスクリプトを作成する

リスト2のwhichcmdには、2つの大きな問題がある。1つめの問題は、PATHに含まれないコマンド（たとえばhoge）を指定して、

```
$ whichcmd hoge
```

とすると、「ls: which:: No such file or directory」などのエラーメッセージを表示すること。もう1つの問題は、引数を1つも指定せずに、

```
$ whichcmd
```

とすると、カレントディレクトリの全ファイルの詳細な情報が表示されることだ。

最初の問題は、PATHに含まれないコマンドをwhichの引数として指定すると、「which: no hoge in （PATHの内容）」というエラーが標準出力に出力されることが原因だ。コマンド置換により「ls -l which: no hoge in (PATHの内容)」が実行され、「which:」などのファイルが見つからないためにエラーとなる。

2番目の問題は、引数が1つも指定されなかった場合に「$＊」が空文字列（長さ0の文字列）で置換されることが原因だ。このとき、シェルスクリプトの実行内容は「ls -l $(which)」になる。まず、whichが引数なしで実行され空文字列を出力し、コマンド置換により「ls -l 」が実行される。その結果、カレントディレクトリの全ファイルの情報が表示されるわけだ。

原因が判明したので対策に移ろう。いずれの問題も、特定の場合にwhichやlsを実行しないようにすれば回避できる。そのためには、条件判断を行うif制御構文を使って処理を分岐させる必要がある。

if制御構文の書式は以下のようになる。elif句は複数記述でき、elif句とelse句は省略可能だ。

▼リスト3 存在しないコマンドに対応したwhichcmd

```
1: #!/bin/bash
2: # usage: whichcmd COMMAND...
3: if cmds=$(which $＊); then
4:   ls -l $cmds
5: fi
```

※実際は行番号は付かない

```
if 条件; then
  処理
elif 条件; then
  処理
else
  処理
fi
```

C言語などとの違いは、条件部に真偽式ではなくコマンドを書くことだ。条件部のコマンドが正常終了した場合に限り、対応する処理部のコマンド（複数可）が実行される。正常終了したかどうかの判断には、終了時にシェルに返される「終了ステータス」の値が使われる。0の場合は正常終了、0以外の場合は異常終了だ。

(1) PATHに含まれないコマンドを指定した場合の対応

最初の問題に対応するには、whichがエラーを起こしたかどうかを判別するため、whichとlsを別々のコマンドラインに分ける必要がある。そこで、コマンド置換の結果を適当なシェル変数（たとえばcmds）に格納する。たとえば、コマンドラインで、

```
$ cmds=$(which rpm find egrep)
```

とすると、cmdsには「/bin/rpm /usr/bin/find /usr/bin/egrep」が格納される。一方、

```
$ cmds=$(which hoge)
```

とPATHに含まれないコマンドを与えると、cmdsには「which: no hoge in （PATHの内容）」というエラーメッセージが格納される。ここで、シェルに返される終了ステータスは、前者はwhichが正常終了したので0、後者は異常終了のため1となる。

演算子	真になる条件
文字列1 = 文字列2	文字列1が文字列2と一致する
文字列1 != 文字列	文字列1が文字列2と一致しない
文字列1 < 文字列2	文字列1が文字列2より小さい
文字列1 > 文字列2	文字列1が文字列2より大きい
-n 文字列1	文字列1が空文字列でない
-z 文字列1	文字列1が空文字列である

表1 文字列比較演算子の一覧

上のコマンドラインと同様の内容をif制御構文の条件部で実行すれば、whichが正常終了した場合にだけlsを実行するスクリプトを作成できる（**リスト3**）。3行目のifの条件部でwhichが実行され、正常終了した場合にはcmdsに格納された出力に対してlsが実行される。

(2)コマンドを1つも指定しなかった場合の対応

コマンドの終了ステータス以外の条件、たとえば2つの文字列が一致しているかとか、指定したファイルが存在するかといった条件を評価するには、ifの条件部で[…]制御構造を使用する。[…]の内部には、文字列比較演算子（**表1**）などを用いた条件式を記述し、その評価が[…]制御構造の終了ステータスに反映される。つまり、条件式の評価が真の場合には0（正常終了）、偽の場合には0以外（異常終了）が返されるわけだ。

たとえば、シェル変数hogeの内容がhogehogeの場合にHOGEと表示するには、

```
if [ "$hoge" = hogehoge ]; then
  echo HOGE
fi
```

と書けばいい（「[」の後ろと「]」の前にはスペースを空ける必要がある）。

ここで注目してほしいのは、変数の周囲を二重引用符で囲って「"$hoge"」と記述することだ。「$hoge != hogehoge」と書いてしまうと、hogeの内容が空文字列だった場合は「!= hogehoge」に置換され、「!=」演算子の左側に文字列がないためエラーになる。空文字列になる可能性のある変数参照は、必ず二重引用符で囲むといい。

さて、2番目の問題に対応するには、引数が1つ以上指定された場合にだけwhichやlsを実行するように、ifを使って処理を分岐させる。**リスト4**のスクリプトでは、3行目の「"$＊" != ""」という条件式で、すべての引数の内容を並べたものを「$＊」で参照し、空文字列「""」との不一致を調べている。引数を1つでも指定すれば「$＊」は空文字列ではなくなるので、4行目のwhichが実行されるわけだ（5行目のlsが実行されるかどうかは、whichの終了ステータスによる）。

▼リスト4 引数を指定しない場合に対応したwhichcmd

```
1: #!/bin/bash
2: # usage: whichcmd COMMAND...
3: if [ "$＊" != "" ]; then
4:   if cmds=$(which $＊); then
5:     ls -l $cmds
6:   fi
7: fi
```

※実際は行番号は付かない

▼リスト5 使い方やエラーの表示を追加したwhichcmd

```
 1: #!/bin/bash
 2: if [ "$＊" != "" ]; then
 3:   if cmds=$(which $＊); then
 4:     ls -l $cmds
 5:   else
 6:     echo "whichcmd: No such COMMAND"
 7:   fi
 8: else
 9:   echo "usage: whichcmd COMMAND..."
10: fi
```

※実際は行番号は付かない

引数が1つ以上指定されていることを判断する条件式はほかにも考えられる。たとえば、引数の数を「$#」で参照して「$# != 0」としたり、「-n」演算子を使って「-n "$＊"」としてもいい。また、[…]制御構造では空文字列は偽、空でない文字列は真と評価されるため、シンプルに「"$＊"」と書くことも可能だ。

ところで、**リスト4**のスクリプトでは、PATHに含まれないコマンドを指定したり、引数を1つも指定しなかったりすると、何も画面に表示せずに終了する。これでは不親切なので、ifの条件部のコマンドが正常終了しなかった場合に実効する処理を記述するelse句を追加して、エラーメッセージや使い方（usage）を表示するように改良したスクリプトを**リスト5**に示す。

スクリプトの完成度を高める

リスト5のスクリプトには、まだいくつか細かな問題が残されている。それらを解決して完成版のwhichcmdを作成しよう(**リスト6**)。

(1) エラーメッセージは標準エラー出力に送る

使い方（usage）やエラーメッセージは、**リスト5**のように標準出力に送るのではなく、標準エラー出力に送ったほうがいい。この種のメッセージは常に画面に表示されることが望ましいからだ。具体的には、echoの出力先をリダイレクト（>）して、標準エラー出力のデバイス「/dev/stderr」に切り替える。

(2) エラー時には0以外の終了ステータスを返す

スクリプトがシェルに返す終了ステータスは、特に指定しない限り、スクリプト内で最後に実行されたコマンドの終了ステータスが使われる。一方、終了ステータスの値を指定してスクリプトを終了するには、組み込みコマンドのexitを使って「exit 値」とする。

ほかのコマンドと組み合わせて使うことを考えると、異常終了にあたる場合は、0以外の終了ステータスを返すことが望ましい。その際、引数が指定されていない場合は1、存在しないコマンドが指定されている場合は2といった具合に値を変えれば、終了ステータスによって異常終了の原因を調べられる。

(3) スクリプト名が変更された場合への対応

リスト5では、使い方（usage）やエラーメッセージ内で「whichcmd」というスクリプト名を直接記述している。もし、スクリプトが別の名前に変更されても、これらの表示は「whichcmd」のままだ。

スクリプト名の変更に対応するには、スクリプト名に置換される「$0」を利用する。ただし、そのままだとディレクトリが先頭に付くので、パターン照合演算子「##」を利用してディレクトリ部分を取り除く。具体的には、使い方やエラーメッセージの「whichcmd」を「${0##＊/}」で置き換えればいい（**リスト6**の3、9行目）。

同様に、whichのエラーメッセージから、PATHに含まれないコマンド名をパターン照合演算子「##」と「%%」を使って取り出すこともできる（7、8行目）。

▼リスト6 完成版のwhichcmd

```
 1: #!/bin/bash
 2:  if [ "$＊" = "" ]; then
 3:    echo "usage: ${0##＊/} COMMAND..." > /dev/stderr
 4:   exit 1
 5: fi
 6: if ! cmds=$(which $＊); then
 7:   errcmd=${cmds##＊which: no }
 8:   errcmd=${errcmd%% in (＊}
 9:   echo "${0##＊/}: $errcmd: No such COMMAND" >
10:/dev/stderr
11:   exit 2
12: fi
13: ls -l $cmds
```

※実際は行番号は付かない

(4)エラーを順次処理して見通しを良くする

リスト5では、2つのifが入れ子になっており、正常な処理とエラー処理の区別がつきにくい。エラーになる場合を順次判別し、処理後にexitでスクリプトを終了すれば、もっと見通しのよいスクリプトになる（**リスト6**）。

2行目のifで使われている条件式「"$＊" = ""]は、引数が1つも指定されていない場合に真になる。処理部（3、4行目）では、使い方（usage）を表示し、終了ステータス1を返してスクリプトを終了する。

6行目のifでは「cmds=$(which $＊)」の終了ステータス（whichの終了ステータスと同じ）が、その前に書かれた否定演算子「!」により反転する。つまり、whichが異常終了した場合にのみ以下の処理部（7～10行目）が実行される。whichのエラーメッセージからコマンド名を取り出し、スクリプトのエラーメッセージを表示した後、終了ステータス2を返してスクリプトを終了する。

以上のエラー処理により、lsを実行すると不都合が起きる状況は取り除かれた。12行目では、whichの実行結果（すなわち各コマンドのフルパスを並べたもの）が格納されたシェル変数cmdsを参照して、各コマンドの詳細な情報をlsの-lオプションで出力している。この場合は、lsの終了ステータスがそのままスクリプトの終了ステータスとして使われる。

☑ 指定したディレクトリ以下に存在する全ファイルから特定の文字列を含む行を探し、ファイル名と行の内容を表示するスクリプトを作成する

後半は毎回ひとつのテーマに絞り、それを実現する方法を説明する。今回のお題は、

指定したディレクトリ以下に存在する全ファイルから特定の文字列を含む行を探し、ファイル名と行の内容を表示するスクリプトを作成する

というもの。連載第1回の記事で、これと同じ処理を行うコマンドラインを作成した。その内容は、

find ディレクトリ -type f -print0 | xargs -r0 egrep 'パターン' /dev/null

というもの。なお、第2回の記事ではテキストファイルのみを対象とするための修正を行った。しかし、現在使われているgrepやegrep（バージョン2.3以降）では、バイナリファイルを自動的に判別する機能が追加されているので、上のコマンドラインをそのまま使用できる。

詳細については、本誌1999 No.2に掲載された第1回の記事を参照されたい。バックナンバーが手元にない場合は、「Linux magazine on the Web」(http://www.ascii.co.jp/linux/allascii/linuxmag/no_02.html)から、記事のPDFファイルをダウンロードできる。

さて、上記のような長いコマンドラインを毎回キー入力するのは大変なので、シェルスクリプト「findgrep」を作成する。まずはコマンドラインと同様の処理を行うスクリプトを作成し、さらに検索対象となるファイル名パターンを指定する-fオプションを解釈して動作を変更できるようにスクリプトを改良する。

▼リスト8 初期版のfindgrep

```
1:  #!/bin/bash
2:  # usage: findgrep PATTERN DIR
3:  find $2 -type f -print0 | xargs -r0 egrep "$1"
4:  /dev/null

※実際は行番号は付かない
```

●コマンドラインの内容をそのままスクリプトにする

まずは、コマンドラインの内容をそのままシェルスクリプト化することから始めよう（**リスト7**）。スクリプトに与える引数は「検索パターン」と「検索開始ディレクトリ」の2つで、「findgrep 'パターン' ディレクトリ」という順番で指定する。egrepと形式を似せるため、コマンドラインで指定する順番とは逆になる。

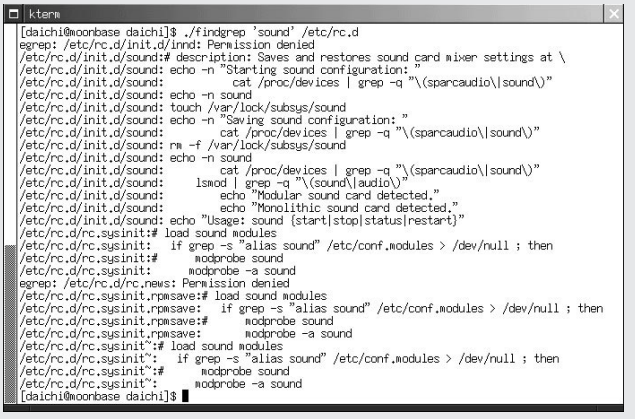
たとえば、

```
$ findgrep 'sound' /etc/rc.d
```

とすると、/etc/rc.d以下の全ファイルを対象として検索が行われ、「sound」を含む行の内容がファイル名とともに表示される（**画面1**）。なお、findgrepがカレントディレクトリにある場合は「./findgrep」として起動することに注意しよう。

リスト7のスクリプトの内容を見ていこう。実体は3行目だけだ。findにより、スクリプトの2番目の引数の内容「$2」を検索開始ディレクトリとして、サブディレクトリ以下も含めた全ファイルのリストが出力される。パイプ（|）で接続されたxargsがそれを受け取り、スクリプトの最初の引数の内容「$1」を検索パターン、受け取ったファイル名を検索ファイルとしてegrepを実行する。

ここで、「$1」が二重引用符で囲まれていることに注目し



画面1 findgrepを実行中の画面

てほしい。検索パターンには正規表現の「*」や「?」がメタ文字として含まれる。二重引用符で囲んでおかないと、シェルがこれらをワイルドカードと見なして展開してしまうので、誤動作の原因になるのだ。

●2個以外の引数を指定した場合に対応する

続いては、想定外の個数の引数が指定された場合、たとえば引数が1つもない、あるいは1つだけ、3つ以上あるといった場合に対応しよう（**リスト8**）。

引数を1つも指定しなかった場合は、whichcmdのように使い方（usage）を表示するのが妥当だろう。**リスト8**では、2〜5行目がこれに該当する。まず、引数が1つも指定されていないか調べ、その場合には使い方を標準エラー出力に送り、終了ステータス1をシェルに返してスクリプトを終了する。

次に、引数を1つだけ指定した場合は「ディレクトリの指定を省略した」、引数を3つ以上指定した場合は「ディレクトリを2つ以上指定した」と判断することにしよう。実際に、findはディレクトリの省略や複数指定に対応している。たとえば、

```
$ find /bin /usr/bin -type f -name rpm
```

とすると、「/bin」と「/usr/bin」の両方が検索開始ディレクトリとなる。また、指定を省略した場合はカレントディレクトリが検索開始ディレクトリとなる。

前節のスクリプト（**リスト7**）でディレクトリの指定を省略すると、「$2」が空文字列に置換されるので、カレントディレクトリが検索開始ディレクトリになる。これは期待通りの動作だ。一方、ディレクトリを2つ以上指定した場合は、最初のディレクトリしか検索されない。3つめ以降の引数がスクリプトで使われていないからだ。

▼リスト8 2個以外の引数に対応したfindgrep

```
 1:  #!/bin/bash
 2:  if [ "$*" = "" ]; then
 3:    echo "usage: ${0##*/} PATTERN [DIR...]" >
 4:/dev/stderr
 5:    exit 1
 6:  fi
 7:  pat=$1
 8:  shift
 9:  find $* -type f -print0 | xargs -r0 egrep "$pat"
10:/dev/null
```

※実際は行番号は付かない

それでは、すべての引数の内容を並べたものに置換される「$*」を使ったらどうだろうか。whichcmdでは、引数の内容はすべてコマンド名なのでうまくいった。ところが、findgrepの場合、最初の引数「$1」は検索パターンで置換されるので、findのディレクトリ指定部分にそのまま「$*」を記述するわけにはいかない。

ここで、組み込みコマンドshiftの出番となる。shiftは、引数で指定した数だけ位置パラメータの内容を前にずらすコマンドだ。引数を省略した場合は内容を1個だけ前にずらす（**図1**）。

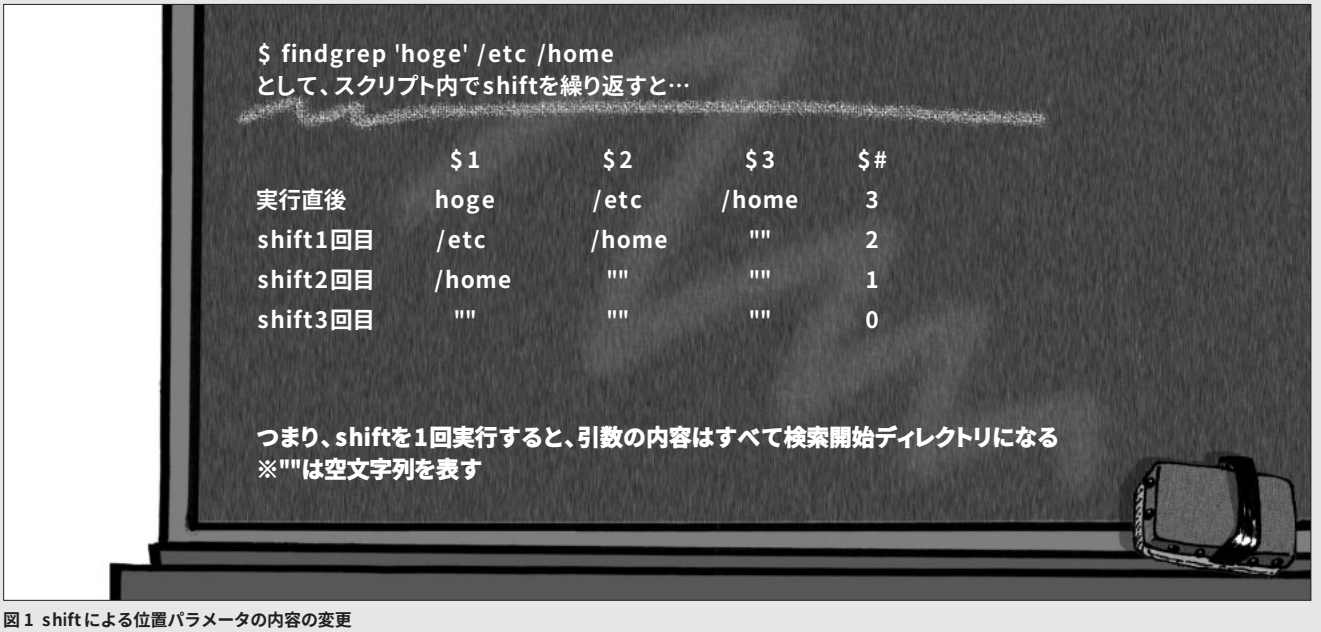


図1 shiftによる位置パラメータの内容の変更

リスト8のスクリプトでは、6行目で最初の引数の内容をあらかじめシェル変数patに保存した後、7行目でshiftを実行している。これで、すべての引数の内容が検索開始ディレクトリになるので、8行目のfindで「$＊」を利用できる。また、最初の引数を保存したシェル変数patにより、egrepの検索パターンは「"$pat"」と書ける。

●オプションの処理を組み込む

最後に、スクリプト実行時にオプションを解釈して動作を変更できるようにスクリプトを改良しよう。具体的には、対象となるファイル名パターンを指定する-fオプションを組み込む。たとえば、

```
$ findgrep -f '＊.c' 'sound' /usr/src/hoge
```

とすると、/usr/src/hoge以下のファイルのうち、「＊.c」にマッチするファイル（C言語のソース）だけを対象に「sound」を含む行を出力する。

bashには、この種のオプション処理用の組み込みコマンドgetoptsが用意されている。書式は「getopts 文字列 変数名」だ。最初の引数の文字列は、オプションから「-」を除いた文字と「:」(コロン)で構成される。コロンは、直前の文字のオプションが引数をとることを示す。また、文字列先頭のコロンは、getoptsのエラー表示を抑制する。findgrepでは、引数をとる-fオプションだけ用いるので、この文字列は「:f:」と書ける。2番目の引数の変数には、現在処理しているオプションから「-」を除いた文字が格納される。このほか、getopts関係の組み込み変数には、オプションがとる引数を格納するOPTARGと、次に処理する位置パラメータの番号を格納するOPTINDがある。

getoptsは一度に1つのオプションしか処理しないので、while制御構文と組み合わせて使うのが普通だ。while制御構文の書式は、

```
while 条件; do
  処理
done
```

で、条件部で実行したコマンドが正常終了すると処理部のコマンド（複数可）が実行され、再度条件部のコマンドの実行に戻る。つまり、条件部のコマンドが異常終了するまで条件部と処理部の実行を繰り返すわけだ。

オプション処理を組み込んだfindgrepの完成版スクリプトを**リスト9**に示す。詳しく内容を見ていこう。まず、getoptsをwhileの条件部で実行すると、処理すべきオプションがなくなるまで処理部（3～8行目）の内容が繰り返し実行される。-fオプションが指定された場合、組み込み変数のOPTARGに格納されている引数をシェル変数filepatに保存する。それ以外のオプションが指定された場合は、エラーメッセージを表示し、終了ステータス1を返してスクリプトを終了する。

続く10行目では、オプションを指定したことにより後ろにずれて格納されている検索パターンやディレクトリを、検索パターンは「$1」、ディレクトリは「$2」以降で参照できるように、shiftを使ってずらす処理を行う。ずらす数は、組み込み変数OPTINDの値から1を引くと得られるので、数値演算を行う$((…))制御構造を利用して「$((OPTIND - 1))」とする。数値演算については、次回の記事で詳しく取り上げる予定だ。

11～16行目は**リスト8**と同じなので省略。17行目でようやく本来の検索処理が実行される。**リスト8**と比較すると、findに判別式「-name "${filepat:-＊}"」が追加されただけだ。ファイルパターンを保持するシェル変数filepatと置換演算子「:-」の組み合わせにより、-fオプションを省略した場合は「-name "＊"」、指定した場合は「-name "ファイルパターン"」となる。

▼リスト9 完成版のfindgrep

```
 1:  #!/bin/bash
 2:  while getopts ":f:" opt; do
 3:    if [ $opt = f ]; then
 4:      filepat=$OPTARG
 5:    else
 6:      echo "usage: ${0##＊/} [-fFILEPAT] PATTERN
 7:[DIR...]" > /dev/stderr
 8:         exit 1
 9:    fi
10:  done
11: shift $((OPTIND - 1))
12: if [ "$＊" = "" ]; then
13:   echo "usage: ${0##＊/} [-fFILEPAT] PATTERN
14:[DIR...]" > /dev/stderr
15:   exit 1
16: fi
17: pat=$1
18: shift
19: find $＊ -type f -name "${filepat:-＊}" -print0 |
20:xargs -r0 egrep "$pat" /dev/null
```

※実際は行番号は付かない

Webサーバ構築術（第8回）

JavaScriptやAppletは、クライアント側で動かすプログラムだ。しかし、そういうWebページをアクセスするのは、IEやNetscapeといったJavaScriptやAppletに対応するブラウザとは限らない。またユーザーによってはその機能をオフしている人もいる。ところがCGIはサーバ側で動作するため、クライアントの動作環境に依存しない。携帯端末でWebを見る人が増えているので、これからもっと多用されていくはずだ。

Apacheで動かすCGI

文：中島昌彦
Text : Masahiko Nakajima

Linuxマシンを構築しているユーザーは、ごく普通にApacheが動いてWebサーバが稼働しているはずだ。PC UNIXの出現により、UNIX環境でいろいろな実験ができるようになっている。

それぞれいろいろな理由でLinuxベースOSを構築していることだろうが、自らサーバを構築して使いたいという理由のひとつに、自分の自由に使えるWebサイトが欲しいという理由がないだろうか。ISPやホスティングの環境では、CGIが使えなかったり、容量の制限があったりと使い勝手が悪い。たとえCGIが使えたとしても、サイト運営側のテストを経過しなければ使えないというケースもある。CGIの実験環境としても、マイWebサイトがあるとないとでは大きな違いにつながってくることだろう。

特定のクライアントだけでの閲覧は時代ではない

これまで、個人ベースでのWebサイトで、少しでも凝ったことをしようとすると、JavaScriptに頼ったり、Appletに頼ることになった。その理由は単純で、ISPがCGIを許さなかったり、自由に実験できる環境がなかったからだといえる。

ただし、これからはちょっと時代の流れが変わってきそうだ。iMODEを始めとして、携帯電話からインターネットアクセスができるようになってきた。Webは画面デザインに凝るという考え方もひとつだが、携帯電話の端末からであれば、画面デザインもなにも存在しない。存在しないどころか、JavaScriptもフレームも受けつけない環境である。

このまま時代が進むと、テレビもゲーム機も電子レンジもインターネットにつながるようになりかねない。そうしたときに、いつまでもJavaScriptやAppletに依存した作りでいては、アクセス数が延びない。特定のクライアントからのアクセスだけを考えていればいいという環境ではなくなってきつつある。

クライアントを選ばずにアクティブなページを作るには、ひとつの方法しかない。サーバ側でプログラムを動かし、その結果を閲覧者のブラウザに返すという仕組みだ。朝ならおはようございますと返し、夜ならこんばんわと返すような仕組みであっても、JavaScriptに依存していてはどんなブラウザからでも見られるものではない。必要なことはサーバ側で処理して、シンプルなHTMLだけで情報を返せば、閲覧者のブラウザがどんなものであっても、コンテンツを表示できる。

サーバ側でのプログラムというと難しそうに感じるが、その仕組みを整えたものがCGIである。Webサーバで共

連載第6回より、Apache 1.3.9をソースからインストールしている環境で説明しています。そのため、httpd.confなどの設定ファイルは、Red Hatのデフォルトでは/etc/httpd/confにありますが、本文では/usr/local/apache/confディレクトリとなっていますので適宜読み替えてください。

Apacheは最新バージョン1.3.11がリリースされています。http://www.apache.org/httpd.htmlより入手できます。

リスト1 1.cgi

```
#!/bin/sh

echo "Content-type: text/html
";

echo "今の時間は<BR>"
echo '<FONT COLOR="#FF0000">'

/bin/date +%r

echo "</FONT><BR>"
echo "です。"
```

画面1 リスト1のCGIの実行結果
右上の画面が閲覧者が1.cgiにアクセスした状態。ブラウザから見たHTMLソースが右下の画面となる。CGIはサーバ側が実行し、その結果をHTMLとして表示するため、どのように記述されているかまったくわからない。ブラウザでソースを表示しても、CGIが返した結果しか見えない。

通のデータ交換インターフェイスに基づいていれば、サーバ上でプログラミングするプログラミング言語は問わない。WindowsやMacという主要なプラットフォームで動き、移植性の高さや手軽にテストができることから、Perlが使われるケースが多いが、Perlに限らず、CやシェルスクリプトでもCGIは作れる。CGIが実行できる環境があるかないかがいちばんの問題だったりする。

/cgi-bin/であれば すぐにCGIが動かせる

通常、Apacheで何か作業をさせようとすると、すぐにhttpd.confとの格闘が始まる。しかし、CGIに関しては/cgi-binに置く限りであれば、特別な設定が不要でCGIが動作する。ディレクトリとしては、/usr/local/apache/cgi-bin/で、ここにスクリプトを置けばいい。

もっとも簡単でありながら、いかにもアクティブな例が**リスト1**だ。わざわざCGIを使わなくてもSSIでも実現可能だが、自分で完全にコントロールできるところが、CGIのいいところだ。

シェルスクリプトを使っているが、仕組みとしては、HTTPヘッダとして、MIMEタイプをCGIが送ることが必須だ。この場合、3行目がそれに該当し、text/htmlを送っている。そして、ヘッダとコンテンツ本文を明示するために、ヘッダ部分は改行2つで終ることになっている。

標準出力に出されたものは、そのままApacheが受け取ってブラウザ側に返すという仕組みだ。つまり、CGI側ではMIMEタイプを渡したあとに、Webブラウザに表示させたいものをなにかしらの形で標準出力されすれば、CGIとしては動作することになる。

たとえば、MIMEタイプとしてtext/plainを送れば、<PRE>を使わなくてもビュアなテキストとして処理できる。このあたりのコントロールが、手軽にできることもひとつの特徴だ。テキストに限らず、JPEGやMPEGのバイナリであっても、正しいコンテンツタイプさえ送り出せば、ブラウザ側では受け取れる。

/cgi-bin/以外のディレクトリにCGIを置く

マイWebサイトであれば、自分の管理下のサーバであるため、自由に/usr/local/apache/cgi-bin/が使える。しかし、会社や学校の部署やゼミ単位で構築しているサーバでは、別途管理者が存在して、自由にcgi-binに書き込めないことがある。

たとえば、そのサーバがWebサイト構築練習用に存在するものであれば、htmlに限らずCGIも自由に試せるような環境であって欲しい。

こんなとき、試す手段は2つある。1つは、cgi-binのパーミッションを公開すること。ただし、完全公開をすると、ほかの人が誤って必要なファイルを削除することもある。また、cgi-binというディレクトリ自体は、/usr/local/apache/htdocsの下にあるわけではなく、/usr/local/apache/cgi-binの下に

ある。使い方によっては危険性を伴うため、cgi-binはScriptAliasによって、ドキュメントディレクトリとは別の場所で運用している。これが逆にアダとなり、ドュメントディレクトリとは違う場所にあるということを理解してもらうにもひと苦労だ。

こうしたことから、テスト用サーバではドキュメントディレクトリ中でCGIを実行できるようにしたり、.htaccessの設定だけでCGIが動かせるような仕組みをとることが多い。もちろん、この設定はデフォルトのままでは動作しない。httpd.confを修正する必要がある。

httpd.confの修正を最低限におさめるとすると、修正場所は次の2カ所になる。httpd.confの675行目にある、

```
AddHandler cgi-script .cgi
```

の行頭にある＃のコメントをはずす。さらに、322行目を、

```
AllowOverride Options
```

と変更する。そして、

```
# /usr/local/apache/bin/apachectl
  restart
```

と入力しApacheを再起動するという手順だ。

最初のAddHandlerの処理が、拡張子cgiのファイルをCGIプログラムとして処理するための手順で、拡張子cgiがCGIスクリプトとして処理されるようになる。次のAllowOverride Optionsは、各ディレクトリで.htaccessによるOptions指定を有効にする設定だ。

CGIを動かしたいディレクトリでは、.htaccessというファイルに、

```
Options +ExecCGI
```

という1行を加えておく。ここから先、各ディレクトリでCGIを動かそうとするならば、自己の責任において.htaccessを作り、上記のような作業をしなければいけない。

誤ってユーザーのドキュメントディレクトリに.htaccessというファイルを作ることはないだろうし、しかもその中にOptions…などという記述は偶然では書けない。あくまでも自己責任のうえでCGIを動かせる環境にしたと考えて、何か問題が生じたとしても、自己責任で解決を求められる。

レンタルサーバだとこうはいかないが、会社や学校のワークサーバならば、これぐらい融通性があってもいい。

アクセスしてきた閲覧者の情報を取得する

閲覧者の情報取得というと、いかにも個人情報を不正に取得しているように思われがちだ。しかし、Apacheはごく普通に閲覧者の情報をログとして記録している。それは、IPアドレスだったり、ブラウザの種別だったりするが、閲覧者の情報を取得していることには違いない。

ログに記録された情報は、cronによ

▼リスト2-1 index.cgi

```
#!/usr/bin/perl

$logfile='/usr/local/apache/htdocs/cgi/logs/log.txt';
$masterhtml='/usr/local/apache/htdocs/cgi/index.html';
$wc='/usr/bin/wc';

$date = `/bin/date`;
chomp $date;

open (LOGWRITE,">>$logfile") || die &erout ("CAN'T WRITE LOGFILE($logfile)");
flock (LOGWRITE,2);
print LOGWRITE "$date\t$ENV{HTTP_USER_AGENT}\n";
flock (LOGWRITE,8);
close LOGWRITE;

$counter=`$wc -l $logfile` || die &erout ("COUNT ERROR($wc)");
$counter=~ s/^ +([0-9]+) .*/\1/g;
chomp $counter;

open (MREAD,"$masterhtml") || die &erout ("CAN'T READ TEMPLATE($masterhtml)");
print "Content-type: text/html\n\n";
while (<MREAD>) {
        s/--DATE--/$date/g;
        s/--COUNTER--/$counter/g;
        s/--BRWSER--/$ENV{HTTP_USER_AGENT}/g;
        print;
}
exit;

sub erout {
        print "Content-type: text/plain\n\n$_[0]";
        exit;
}
```

って1日ごとあるいは1週間ごとに切り出される仕組みになっている。access_logにはすべてのアクセスが記録されるためファイルサイズも大きいし、分割されているため必要な情報を取り出すのは大変だ。そういう時には独自のログを記録する仕組みを自分（＝CGI）で持つようにするといい。

そんなCGIが、**リスト2-1**である。Perlで記述したindex.cgiは、アクセスがあるたびに、ユーザーのブラウザ情報を独自ログに記録して、その行数を元にログ記録を開始してからのアクセス数を割り出す。ブラウザがApacheに渡す環境情報は、すべて環境変数として取り出せるため、手軽に扱える。

これを/usr/local/apache/htdocs/cgi/の下にindex.cgiとして保存する。このスクリプトには実行属性が必要なので、

```
# cd /usr/local/apache/htdocs/cgi
# chmod +x index.cgi
```

として、Perlが起動するようにする。

なお、このディレクトリに先ほどの.htaccessファイルを作っておく。

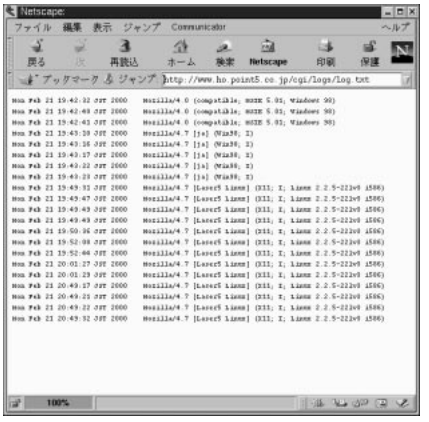
index.cgiはログを、/usr/local/apache/htdocs/cgi/logs/log.txtというファイルに保存する。最初cgiディレクトリの下にlogsはないため、あらかじめディレクトリを作っておく。そして、CGIプログラムが書き込めるようにパーミッションを開けておく。

```
# mkdir logs
# chmod o+rwx logs
```

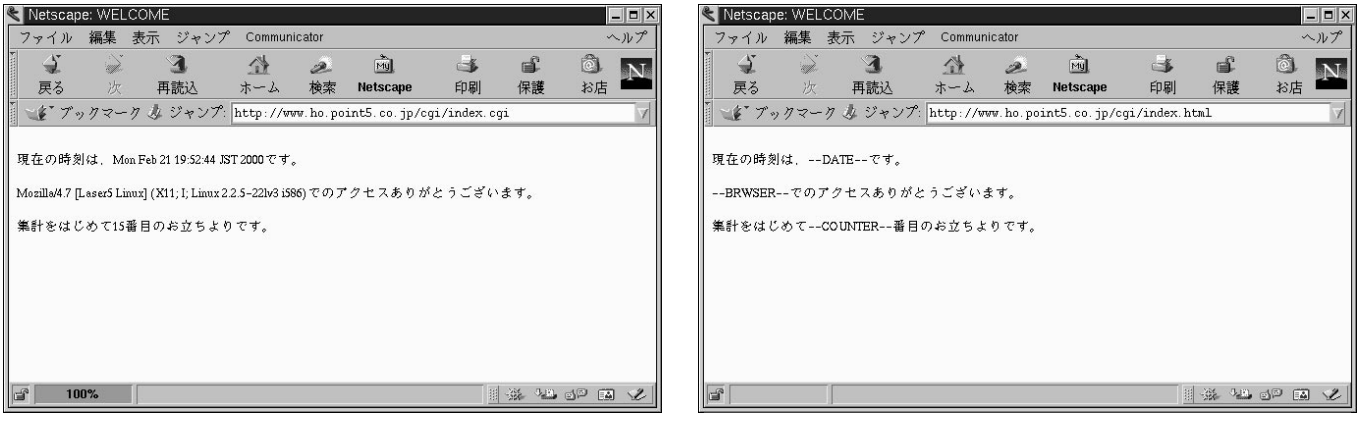
さらに、初期ファイルとして、空のlog.txtファイルを作っておく。

▼リスト2-2 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>WELCOME</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">

<BR>
現在の時刻は、--DATE--です。<BR>
<BR>
--BRWSER--でのアクセスありがとうございます。<BR>
<BR>
集計をはじめて--COUNTER--番目のお立ちよりです。

</BODY>
</HTML>
```



画面2 独自のアクセス記録例（log.txt）
アクセスした時刻とブラウザの種別を1行単位に記録している。真剣にアクセスログを解析するならば、そのほかに直前のURL、アクセス元IPアドレスといった情報も合わせて記録することが多い。アクセス時間を除くアクセス元情報は、すべて環境変数で引き出せるため、容易にログを記録できる。



画面3 リスト2の実行結果
Perlのスクリプトであるindex.cgiにアクセスした結果が左の画面となる。また、ダイレクトにテンプレートファイルであるindex.htmlにアクセスしたときには、右の画面のように表示される。正しくCGIを呼び出していれば、テンプレートを読み込んで置換表示する。しかし、ダイレクトにindex.htmlを呼び出すと、テンプレートそのままが表示されてしまう。

```
# cp /dev/null logs/log.txt
# chmod o+rw logs/log.txt
```

index.cgiは、ログファイルを書き込んだあとに、index.html（**リスト2-2**）を読み込んでブラウザに返す仕組みになっている。そこで、index.html内に、ブラウザ情報を表示する部分とアクセス数を表示する部分を埋め込み、index.htmlをテンプレートとして利用できるような形にした。実際には、テンプレートファイルはindex.htmlではなく、別の名前にしておいたほうがよいだろう。

非常に単純なCGIで、http://（サーバ名）/cgi/index.cgiとしてアクセスすれば、アクセスしてきた時刻とブラウザ情報が、/usr/local/apache/htdocs/cgi/logs/log.txtに記録されるわけだ。

この例ではログファイルがドキュメントディレクトリにあるので、http://（サーバ名）/cgi/logs/log.txtとすることで、結果をブラウザで見ることができる（**画面2**）。ファイル名がわかってしまえば誰でも見ることができてしまうので、セキュリティ上問題がある。そのためCGIで記録するファイルはドキュメントディレクトリ（/usr/local/apache/htdocs）以外に置くようにすることをお勧めする。

nobody/nobodyの実行属性とディレクトリインデックスの調整

CGIでファイルの書き込みをするようになると、必ず作成者パーミッションの問題につきあたる。デフォルト状態では、httpd.confの252、253行目に記述されている、

```
User nobody
Group nobody
```

というユーザーとグループでApacheが実行されている。Apacheのプロセスとして起動したCGIもまたnobody/nobodyで実行されているため、ファイルの書き込みパーミッションには気をつけないと思わぬ落とし穴にはまることになる。

ファイルが書き込めないというエラーが出たとき、それはたいていの場合ディレクトリパーミッションが開いていないときだ。CGIを作るユーザーには、最初からApacheと同じグループメンバーに属させておいたほうが危険が少ない。この場合の危険性とは、ディレクトリパーミッションを完全にオープンにしてしまうことにより、第三者にわけのわからないファイルを書き込まれたりする危険性だ。Apacheと同じグループパーミッションを持っているならば、グループの読み書きまで許すだけでよく、万人が書き込めるディレクトリにする必要はない。

もうひとつ、CGIを扱うようになると、DirectoryIndexの調整が必要になる。たとえば、**リスト2**の例でいくと、/cgi/index.cgiがindex.htmlを読み込む。ところが、/cgi/というURLで呼び出すと、index.cgiではなく、index.htmlが呼び出される。できることならば、ここでindex.htmlを読み込まずに、index.cgiをデフォルトで起動してもらいたいものだ。そこで、httpd.confの358行目を、

```
DirectoryIndex index.cgi index.html
```

というように、デフォルトファイル名としてindex.cgiを先に指定し、それからあとにindex.htmlを指定する。そしてApacheを再起動すれば、ブラウザからディレクトリ指定でアクセスされた場合に、そのディレクトリにindex.cgiとindex.htmlの2つが存在したなら、index.cgiを先に呼び出すようになる。

ただし、本格的にCGIでテンプレートを呼び出す場合、たいていはhtml以外の拡張子を使い、容易にテンプレートにアクセスしにくくする。作成している人によってちがうが、.html.masterとか、.html.tmplという拡張子であったり、単純にtplやtmplという拡張子もある。いずれにしても、自分ならではの拡張子で管理したほうがいい。

商用サーバともなると、ドキュメントディレクトリ以外のところにテンプレートファイルを置き、CGIはテンプレート専用のディレクトリにアクセスするような仕組みをとる。ドキュメントディレクトリ以外にテンプレートを置くことで、Webブラウザ経由ではどんなことをしてもテンプレートファイルにアクセスできないようにしておく。

CGIはサーバ側で動作するアプリケーションである。だからこそ、パーミッションさえ開いていれば、テンプレートにはアクセスできる。安全性を高めようとするならば、基本的にテンプレートはドキュメントディレクトリ以外の場所に置くべきものである。

POST/GETによるコンテンツ間でのデータ交換

ブラウザがApacheに渡すデータだけをCGIがやり取りしているだけであれば、ここまでCGIが着目されることもない。CGIの本当の威力は、閲覧者とサーバがコミュニケーションを取れることである。

サーチエンジンやWebサイト検索エンジンでおなじみのフォームは、まさにPOSTやGETでデータを受け渡している。HTMLから受け渡されるデータを、CGIはPOSTもしくはGETとして

▼リスト3-1 memo.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
        <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>MEMO</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">

<FORM ACTION="memow.cgi" METHOD="POST" ENCTYPE="application/x-www-form-urlencoded">
<P>メモ<BR>
<INPUT TYPE="TEXT" NAME="memo" SIZE="52"><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="書き込み"></FORM>
<BR>
<BR>
これまでのメモ(今の時間は--DATE--)<BR>
<BR>
--MEMO--

</BODY>
</HTML>
```

▼リスト3-2 memo.cgi

```
#!/usr/bin/perl

$memofile='/usr/local/apache/htdocs/cgi/memos/memo.txt';
$masterhtml='/usr/local/apache/htdocs/cgi/memo.html';
$date=`/bin/date '+%Y/%m/%d %H:%d'`;
chomp $date;

open (READMEMO,"$memofile") || die &erout ("CAN'T OPEN MEMOFILE($memofile)");
@memo=<READMEMO>;
close READMEMO;

$i=0;
foreach (@memo) {
        s/^(.*)$/$i:\1\[<A HREF="memod.cgi?no=$i">用件終了<\/A>\]<BR>/;
        ++$i;
}

open (MREAD,"$masterhtml") || die &erout ("CAN'T OPEN TMPLATE($masterhtml)");
print "Content-type: text/html\n\n";
while (<MREAD>) {
        s/--MEMO--/@memo/g;
        s/--DATE--/$date/g;
        print;
}
exit;

sub erout {
        print "Content-type: text/plain\n\n$_[0]";
        exit;
}
```

受け取る。POSTの場合は、標準入力としてフォームデータを受け取り、GETの場合は環境変数QUERY_STRINGを経由して受け取る仕組みだ。

リスト3がその例だ。ToDoリストのWeb版であるが、指定時間になったらアラームが鳴るわけでもない。フォームに書き込んだメッセージが表示され、必要に応じて消せるという簡単なCGIプログラムを作った。処理は3つのCGIと1つのHTML、書き込みファイルを保存するもう1つのテキストファイルという構成だ。

テンプレートとなるHTMLファイルがmemo.html（**リスト3-1**）で、これをmemo.cgi（**リスト3-2**）が読み出して表示する。

これを実行すると**画面4**のようになる。上部フォームで文字を書き込むと、POSTとして書き込んだ文字列がmemow.cgi（**リスト3-3**）に渡され、memow.cgiがmemos/memo.txtに書き込む。メッセージを削除するときに

▼リスト3-3 memow.cgi

```
#!/usr/bin/perl

$memofile='/usr/local/apache/htdocs/cgi/memos/memo.txt';
$masterhtml='/usr/local/apache/htdocs/cgi/memo.html';
$date=`/bin/date '+%Y/%m/%d %H:%d'`;
chomp $date;

if ($ENV{REQUEST_METHOD}=~ m/POST/i) {
        read(STDIN,$buffer,$ENV{CONTENT_LENGTH});
        foreach $qstr (split(/&/,$buffer)) {
                ($qname, $qval) = split(/=/, $qstr);
                $qval=~ tr/+/ /;
                $qval=~ s/%([a-f0-9][a-f0-9])/pack("C",hex($1))/eig;
                $query{$qname}=$qval;
        }
} else {
        &erout ("CAN'T GET POST DATA");
}
open (MEMOWRITE,">>$memofile") || die &erout ("CAN'T OPEN
MEMOFILE($memofile)");
flock (MEMOWRITE,2);
print MEMOWRITE "$query{memo}($date)\n";
flock (MEMOWRITE,8);
close MEMOWRITE;

print "Location: ./memo.cgi?\n\n";

exit;

sub erout {
        print "Content-type: text/plain\n\n$_[0]";
        exit;
}
```

画面4 memo.cgiの画面
memo.cgiはmemo.htmlをテンプレートとして読み込み表示する機能だけを持つ。データの書き込みはmemow.cgiが、データの消去はmemod.cgiが受け持ち、書き込みや消去をしたあとには、memo.cgiに戻る仕組みになっている。

は、用件終了をクリックすると、行番号をGETとしてmemod.cgi（**リスト3-4**）に受け渡し、memod.cgiが該当する行を削除するという単純な仕組みだ。

なお、このCGI一式を動かすには、memo.txtに書き込んだメッセージをすべて記録するため、/usr/local/apache/htdocs/cgi/memos/ディレクトリを作り、Apacheの実行属性nobody/nobodyが読み書きできるようなパーミッションを与え、memo.txtにもnobody/nobodyが読み書きできるようにパーミッションを与える。

```
# cd /usr/local/apache/htdocs/cgi
# mkdir memos
# chmod o+rwx memos
# cp /dev/null memos/memo.txt
# chmod o+rw memos/memo.txt
```

POST／GETでのデータエンコード

POSTは標準入力からURLエンコードした文字列を受け取り、GETは環境変数QUERY_STRINGを通して、URLエンコードした結果を受け取る仕組みだ。基本的に、POST、GETとも、

ネーム=値

という形式で受け渡す。複数のネームがある場合には、

ネーム1=値1＆ネーム2=値2

という形でデータが渡ってくる。受け取り側では、POST、GETを判断したうえで、標準出力、もしくは環境変数から渡された文字列を読み出して、ネームと値を切り出したうえで、URLデコードをかけるという仕組みを作れば

▼リスト3-4 memod.cgi

```
#!/usr/bin/perl

$memofile='/usr/local/apache/htdocs/cgi/memos/memo.txt';
$masterhtml='/usr/local/apache/htdocs/cgi/memo.html';

if ($ENV{REQUEST_METHOD}!~ m/POST/i) {
        $buffer=$ENV{'QUERY_STRING'};
        foreach $qstr (split(/&/,$buffer)) {
                ($qname, $qval) = split(/=/, $qstr);
                $qval=~ tr/+/ /;
                $qval=~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex(\1))/eg;
                $query{$qname}=$qval;
        }
} else {
        &erout ("CAN'T GET DATA");
}

open (READMEMO,"$memofile") || die &erout ("CAN'T READ MEMOFILE($memofile)");
@memo=<READMEMO>;
close READMEMO;

$memo[$query{no}]='';

open (MEMOWRITE,">$memofile") || die &erout ("CAN'T WRITE
MEMOFILE($memofile)");
flock (MEMOWRITE,2);
print MEMOWRITE @memo;
flock (MEMOWRITE,8);
close MEMOWRITE;

print "Location: ./memo.cgi?\n\n";

exit;

sub erout {
        print "Content-type: text/plain\n\n$_[0]";
        exit;
}
```

いい。なお、memo.cgiが出力したHTMLソースを見るとわかると思うが、POST対象となるフォームは、<FORM>～</FORM>の間にあるFORMオブジェクトのみだ。また、GETで渡すときには、データを受け渡す先のURLに続いて、？名前=値&…の形で記述する。

POSTもしくはGETのどちらで渡されたかは、環境変数REQUEST_METHODで判断できる。もしREQUEST_METHODがPOSTであれば、POST処理をすればいい。

POSTを使って値を受け取っている例が、memow.cgiだ。書き込み文字列として、どれだけの長さのデータが渡されるか予想がつかない。こういうものは、POSTで受け渡したほうが安全である。

POSTで受け渡されたメッセージは、memow.cgiの10～15行の処理で分割され、$query{名前}という形で格納される。以後のPerlスクリプト中では、$query{名前}を使って参照する。

標準入力から読み込むときは、普通の書き方だと、EOFが返されないために入力終了を検知できない。そのため、環境変数のCONTENT_LENGTHにPOSTされた長さが格納されている。つまり、この長さだけ標準入力から読み込むという書き方をする。

逆に、GETで受け渡されたときは、memod.cgiの8～13行で分割している。POSTとGETの処理の差は、受け渡された文字を標準入力から取り込むか、環境変数から取り込むかだけの違いしかない。

memow.cgi、memod.cgiともに、書き込み、もしくは行削除の処理をしたあとには、memo.cgiに戻り、データの再描画をする仕組みだ。このときには、Locationを使い、Webブラウザから再度memo.cgiを読み出させる設定になっている。memow.cgiなら25行目、memod.cgiなら30行目にある、

```
print "Location: ./memo.cgi?\n\n";
```

が該当する部分だ。Content-typeではなく、Location:で返すと、そのあとに続いたURLに再度ブラウザはアクセスを試みる。ブラウザ側に再アクセスさせることで、書き込みや削除をした画面を確認させるという仕組みだ。URLの行末に？を付けている理由は、Proxy対策である。キャッシュされた情報を取り出されては書き込み内容が反映されないため、？を付けることでProxyに再度データを読みに行かせる設定になっている。たいていのProxyは、URLに？が付いている場合、キャッシュを読み出さずにダイレクトに情報を読み取りにいくような設定になっているからだ。

Rubyで行こう

プログラムというものには入出力がつきものです。これができなければ、何が起きたかわからないし、何の処理もできません。これではせっかくのRubyも意味なしです。というわけで、今回はRubyの入出力におけるカナメ、ストリーム系クラスについて紹介します。

第4回　ストリーム

文：赤松智也
Text : Tomoya Akamatsu

先月号は休んでしまいました。今後はこのようなことのないように気をつけますので、やっと終わったのかとか、お見限りのないようお願いします。

さて今回は、あらゆる処理の基本「入出力」について解説しましょう。Rubyにおける（というかUNIX、ひいてはC言語における）入出力には、その出入口として「ストリーム」という概念が重要になります。ストリーム（*stream*）というのは、「流れ」という意味で、文字あるいはバイトが「流れてくるように入力」され、「流れていくように出力」されるので、そういう名前になっています。何を今さら当たり前のことを、と思われる方がいるかもしれませんが、ふた昔前の入出力はレコード単位で行われるのが普通で、UNIX流のストリームという概念は画期的なことだったのです。まぁ、このあたりは本で読んだだけで本当は知らないのですが。いや、本当に。

UNIXのストリーム

Rubyがベースにしているの入出力には、ほかにも画期的（もちろん当時は、という意味ですが）なことがあるので、このあたりも紹介しておきましょう。UNIXのストリームは、OSからはファイルディスクリプタ（ファイル記述子ともいいます）として扱われます。このファイルディスクリプタというのは、「プログラムがオープンしているファイル（＝ストリーム）に対応する番号」のことで、単なる整数です。ここで重要なのは、このファイルディスクリプタに対応しているストリームの接続先が、

- **ハードディスク上のファイルだろうが**
- **フロッピー上のファイルだろうが**
- **CD-ROM上のファイルだろうが**
- **最近あまり見ないテープデバイスだろうが**
- **なにか別のプログラムからの出力だろうが**
- **ネットワーク経由の接続（ソケット）だろうが**

すべてまったく同じように入力（または出力）できることです。もちろん、OSの内部処理では、上記のそれぞれに対してまったく違う処理が行われます。ちょっと考えただけでもファイルシステムからのデータの読み出しと、ネットワークを介したデータ通信では処理内容は全然違うことは想像できます。

しかし、ファイルディスクリプタを使う限り、プログラム側が接続先（と、それに対応した処理）を意識する必要はまったくありません。それは、OSが勝手に接続先に合わせた処理を行ってくれるからです。便利なものです。具体的には、読み込みにはreadシステムコールだけ、書き込みにはwriteシステムコールを利用するだけで、すべての入出力先への処理が行えるのです（**図1**）。

これは、オブジェクト指向のポリモルフィズム（動的結合ともいいます）という考え方そのものです。UNIXの開

発が始まったのは1960年代末ごろで、当時はオブジェクト指向という考え方がまだまったく広まってなかった（Smalltalk開発開始は1970年代初頭、公式発表は1984年）ということを考えると、このことはかなり画期的なことだったといえるでしょう。

C言語のストリーム

ファイルディスクリプタのおかげで、各種ファイルおよび入出力先が統一的に扱えて便利なのはうれしいのですが、まだちょっと不満があります。それは性能という点です。

実は、プログラムにおいてシステムコールの呼び出しというのは、比較的コストの高い処理です。UNIXのようにカーネルとユーザープログラムが異なるメモリ空間で動作しているタイプのOSでは、システムコールごとに引数となるデータのコピーやメモリ空間の切り替えなど、かなりコストのかかる処理がてんこもりです。1文字書き込むたびにシステムコールを呼び出していたのでは、プログラムが遅くて使い物にならない場合もあります。

このように、遅くて重いシステムコールの呼び出しを避けるために考えられたのが、ファイルディスクリプタの外側を包むstdioライブラリです。stdioライブラリはファイルディスクリプタを包み込むFILE構造体がバッファを持ち、個別の入出力はただ単にバッファに書き込むだけです。バッファがいっぱいになったときなど、適当なタイミングでシステムコールを呼び出して、実際の出力をまとめて行います。stdioライブラリは、この「適当なタイミング」が絶妙に設計されているので、ふだん使っていてもあまりバッファリングされていることが気になりません。性能は向上させて実装の都合は外に見せない、というのはライブラリの鏡ですね。カッコイイ気がします。

ここでC言語のプログラムでも書いて、このバッファリングの効果を実際に試してみたいところですが、これはRubyの連載ですからやめておきます。ただ、標準入力から標準出力へ1文字ずつコピーするプログラムは、stdioによるバッファリングを使わない場合、10倍近く遅くなったということだけ報告しておきます。

stdioは、C言語の規格に含まれていますから、UNIXに限らずどこでも使えます。C言語でプログラミングをしたことがある方なら、一度は使ったことがあるでしょう（printfなどもstdioの一部です）。しかし、stdioライブラリの最大の利点であるバッファリングも、ネットワーク通信などではかえってジャマになったりします。なぜなら、先方に送ったつもりでいるのに、実はバッファの中に残っているだけで実際には送られていないため、いつまでも返事が来ないというデッドロックが発生したりといった問題の原因になったりするからです。とはいえ、出力するたびに毎回バッファ内容をフラッシュしたのでは繁雑になるばかりです。

結局、ネットワーク通信にはファイルディスクリプタを素のまま使うというのが典型的なアプローチです。まぁ、それはそういうものだとは思うのですが、せっかくいろいろな入出力先を統一的に扱えるというUNIXのポリモルフィズム的、あるいはオブジェクト指向的なうれしさが減ってしまうようで、ちょっと悔しいような気もします。世の中うまいことばかり、というわけにはいかないようです。

Rubyのストリーム

さて、ここからが今回の本題です。では、「カッコよさ」「うれしさ」を追求するRubyの入出力は、どのようになっているでしょうか？

Rubyの入出力は、

- **オブジェクト指向機能によってポリモルフィズムのうれしさを維持**
- **性能を実現するバッファリングはちゃんと行ってくれる**

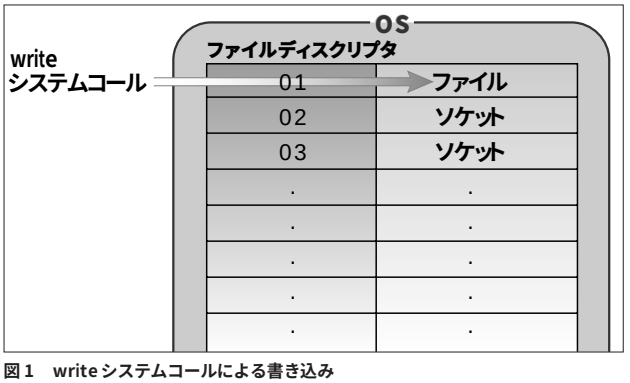


図1　writeシステムコールによる書き込み

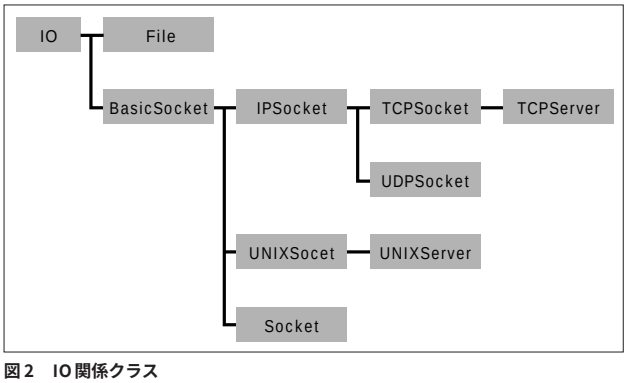


図2　IO関係クラス

が、ネットワーク接続などでも　賢く制御してくれるので、統一的に扱える

という「イイとこ取り」になっています。だからRubyでは、世の中うまいことばかり、になっています。さすが。

Rubyの入出力ストリームは、IOクラスとそのサブクラスによって実現されています。IO関係のクラス階層を**図2**に示します。この中にあるBasicSocketとそのサブクラスはネットワーク接続用のクラスです。ネットワーク接続については、また別の回に説明するつもりなので、今回はIOクラスとFileクラスについてだけ説明します。

IOクラス

IOクラスは、Rubyにおける入出力のカナメのクラスで、UNIXにおけるファイルディスクリプタに相当するものです。このIOクラスのオブジェクトを得る方法はいくつかあります。

- **定数STDIN、STDOUT、STDERR**
  プログラムにデフォルトで与えられているストリームである、標準入出力に相当するIOオブジェクトがあらかじめこれらの定数に格納されています。

- **openメソッド**
  ファイルをオープンする基本的なメソッド。後述。

そのほかにもIOクラスのオブジェクトを得る方法には、

- **IO::newメソッド**
- **IO::popenメソッド**
- **IO::pipeメソッド**

とかいろいろあるのですが、話が高度になってしまうので、今回は説明しません。このIOクラスのメソッドを次ページの**表1**に示しておきます。

ストリームのオープン

というわけで、あらかじめオブジェクトが用意されている標準入出力以外のファイルを使うには、まずストリームをオープンして、対応するIOオブジェクトを入手する必要があります。それを行う基本的なメソッドがopenメソッドです。openメソッドは、次のような書式で呼び出します。

```
open(path[, mode])
```

openメソッドは、パスを指定してストリームをオープンし、それに対応するIOオブジェクトを返します。厳密にいうと、ファイルのパスを指定した場合は、IOのサブクラスであるFileオブジェクトを返すのですが、IOオブジェクトとみなして扱ってかまいません。

modeには、次のうちのいずれかの文字列を指定します。

“r”　読み込み用
“w”　書き込み用
“a”　追加書き込み用

modeを省略した場合には“r”を指定したものとみなします。実は、modeには整数のフラグを指定するやり方もあります。

```
open(path, File::RDONLY|File::APPEND)
```

こちらは、openシステムコールの引数と同じになります。

ファイルからの読み込み

ストリームは、オープンしただけでは何の役にも立ちません。読み込みなり、書き込みをして、初めてオープンした意味を持つわけです。RubyのIOクラスの読み込みメソッドには次のものがあります。

- **gets([sep])**
  ファイルから1行ずつ読み込んで、その文字列を返します。引数sepが与えられると、その文字列を行区切りとします（sepのデフォルト値は“\n”）。ファイルの終端に届く（＝読み込む行がなくなる）と、文字列の代わりにnil（偽）を返します。

```
f = open("/etc/passwd")
while line = f.gets
   ...
end
```

- **readline([sep])**

getsメソッドとほぼ同じですが、ファイルの終端でEOFError例外を発生させる点が異なります。

クラスメソッド	説明
foreach(path[, sep])	各行繰り返し
new(fd[, mode])	ファイルディスクリプタから
pipe	[r,w]のストリーム
popen(cmd[, mode])	コマンドへの入出力
readlines(path[,sep])	各行の配列
select(r[,w[,[e[,timeout]]]])	入出力待ち
インスタンスメソッド	**説明**
self << obj	出力
binmode	バイナリモード
clone	ストリーム複製
close	クローズ
close_read	入力クローズ
close_write	出力クローズ
closed?	クローズされたか
each([sep])	各行繰り返し
each_line([sep])	eachの別名
each_byte	1バイトずつ繰り返し
eof?	ファイル終端か
eof	eof?の別名
fcntl(req[,arg])	fcntlシステムコール
fileno	ファイルディスクリプタ
to_i	filenoの説明
flush	バッファのフラッシュ
getc	1バイト読み込み
gets([sep])	1行読み込み（sep:行区切り）
ioctl(req[,arg])	ioctlシステムコール
tty?	ttyか
isatty	tty?の別名
lineno	行番号
io.lineno=num	行番号設定
pos	ファイル位置
io.pos=pos	ファイル位置設定
print(str,...)	出力
printf(fmt,...)	フォーマット付き出力
putc(c)	1文字出力
puts(str)	1行出力（改行つき）
read(len)	固定長読み込み
read()	全ファイル読み込み
readchar	1文字入力（EOFで例外）
readline([sep])	1行入力（EOFで例外）
readlines([sep])	各行の配列
reopen(io)	再オープン（取扱注意）
rewind	巻き戻し
seek(offset[, whence])	ファイル位置指定
stat	ステータス取得
sync	同期モード
io.sync=mode	同期モード設定
sysread(len)	readシステムコール
syswrite(str)	writeシステムコール
tell	ファイル位置
to_io	内部的に使われる
ungetc(c)	1文字読み戻し
write(str)	1行出力

表1　IOクラスのメソッド

```
begin
  loop do
    line = f.readline
    ...
  end
rescue EOFError
  ..ファイル終端の処理..
end
```

- **readlines([sep])**
  読み込んだ各行を配列に入れて返します。ファイルをすべてメモリ中に読み込むため、あまり大きなファイルには向きませんが、メモリ内に収まるのであれば、たぶんこれが一番速いでしょう。

- **sysread(len)　readシステムコール**
  readシステムコールを呼び、読み込んできた文字列を返します。

- **getc**
  1バイトだけ読み込んで、文字コードに対応する整数を返します。EOFでは、nilを返します。

- **readchar**
  getcメソッドとほぼ同じですが、EOFError例外を発生させる点が異なります。

これらのメソッドは、呼び出すごとに読み込んでくるタイプのものですが、Rubyにはもう1つブロックに対して繰り返すタイプのメソッドがあります。これは、読み込んでくるたびに、読み込んだ文字列をブロックに渡してくれるものです（このようにブロックを使って繰り返すメソッドを「イテレータ」と呼びます。以前にも紹介しましたよね）。
この入力用イテレータには、以下のものがあります。

- **each_line([sep]){|x|..}**
  各行に対して繰り返します。引数sepの意味は、getsメソッドと同じです。使い方は次のような感じです。

```
f = open("/etc/passwd")
f.each_line do |line|
  ...
end
```

getsメソッドを使ったものよりもちょっと気持ちいい感じがしませんか？ このようにブロックを使った方法は、getsメソッドを使うよりも、（わずかですが）高速に動作するようです。

- **each([sep]){|x|..}**
  each_lineの別名。eachメソッドは、for文の内部でも使われますから、次のような書き方もできます。

```
f = open("/etc/passwd")
for line in f
  ...
end
```

- **each_byte{|x|..}**
  ファイルから1バイトずつ読み込んで、繰り返します。

入力だけでいろんな方法があることにびっくりされたかもしれません。Rubyは、便利そうならいろいろなやり方を提供するというのが信条のようです。Rubyを使うなら、「やり方はいろいろある」というスローガン（Perlからの盗作ですが）を覚えておくとよいでしょう。いろんなメソッドがある、つまり語彙が豊富ということは、表現力が高く、プログラムが書きやすく、かつ読みやすくなるということです。このあたりは、自然言語も同じなので、わかってもらえると思います。

ファイルへの書き込み

Rubyは、出力も入力と同様に簡単です。出力用のメソッドには次のものがあります。

- **write(str)**
  実際の出力を行います。他のメソッドは、これを使って出力しています。

- **print(obj,..)**
  引数を順に出力します。

- **printf(fmt,..)**
  C言語のprintfと同様に、引数をフォーマットして出力します。

- **io << obj**
  C++言語的な表記で出力します。

- **puts(str)**
  1行出力します。引数の出力がprintとの違いです。

Rubyでは、明示的に出力先を指定する場合は、次のように、printメソッドなどの「左側」に指定します。

```
print "foobar\n"       # 標準出力へ出力
f.print "foobar\n"     # fに対して出力
```

コマンドへの入出力

コマンドへの入出力が、ファイルに対する入出力と同様の方法でできるのがUNIXのよいところです。Rubyでは、コマンドとの入出力にもopenメソッドを使います。openメソッドに指定されたパスの先頭の文字が“|”なら、それに続くコマンドとの間にストリームを確立します。

```
# カレントディレクトリのファイル名の一覧
file = open("|ls").readlines

# 出力の行数、語数、バイト数を数える
f = open("|wc", "w")
f.print "aaa\n"
...
f.close
```

ストリームのクローズ

ストリームに対する入出力が終了したら、そのストリームをクローズします。もっともRubyでは、ガーベジコレクタがプログラム終了までのどこかのタイミングでストリームを自動的にクローズしてくれますから、クローズするのを忘れてしまってもあまり深刻な問題にはなりませんが。ですから、Rubyでは次のような書き方も気楽にできます。

```
line = open(path).gets # 最初の1行を読み出す
```

このopenメソッドで作られたIOオブジェクトは、いつ

かは自動的にクローズされるという安心設計です。ただし、出力用のストリームでは、明示的にクローズしておかないとバッファにデータが残っていることがありえますから、きちんとクローズするクセをつけることをお勧めします。

「でも面倒だなぁ」とか「C言語では当然かもしれないけど、Rubyではもっとラクしたいよ」という人もいるかもしれません。そんなあなたに朗報が…。実は、Rubyではクローズを忘れない仕掛けが提供されているのです。

まず、openメソッドの別形式から紹介しましょう。

```
open(path) do |f|
  ...
end
```

このプログラムは、ファイルをオープンしてからブロックの中を実行しますが、ブロックを抜けるときには自動的にクローズしてくれます。returnとか例外とかで抜けるときにもちゃんとクローズしてくれます。

次の方法は、「暗黙のオープン」です。これは、名前のとおり、ユーザーが明示的にオープンするのではなく、システムが内部で「こっそりオープン」してくれているというものです。具体的には次のようなメソッドがあります。

```
IO::foreach("/etc/passwd") do |line|
  ...
end

lines = IO::readlines("/etc/passwd")
```

IO::foreachはpathで指定されたファイルを内部的にオープンして、1行ずつブロックに渡します。一方、IO::readlinesは、ファイルを全部一度に読み込んで、1行ごとの配列として返します。これらのメソッドは、処理が終われば当然自動的にクローズしてくれます。

PPPログ解析

いやあ、まるでリファレンスマニュアルのような退屈な説明が続いてしまいましたが、ガマンできましたか？　まぁ、何事も使いこなすようになるためにはガマンも必要です（開き直り？）。

では、ここまでの復習も兼ねて、ちょっと例題を見てみましょう。題して「PPPログ解析」です。これは、ダイヤルアップユーザーならいつも気になる「いつ、どのくらい電話したか」をレポートしてくれるというものです。

このプログラムを作るときの、基本的な考え方は次のような感じです。

■ログの構造

私はダイヤルアップ用のプログラムとして、Linuxの純正PPPプログラムを使っています。純正PPPのログファイルは、**リスト1**のような形式になっています。ただし、このログファイルは、使用しているPPPによって置かれている場所や構造が異なっているかもしれません（PPxPでは、/var/log/ppxp/qdial.logかな？）。ですから、ログファイルの位置や形式についてはスクリプトで調整しやすいようにしておきます。

▼リスト1　純正PPPのログファイル（一部）

```
Feb 14 15:58:49 eevui pppd[13592]: pppd 2.3.10 started by akamatz, uid 1000
Feb 14 15:58:50 eevui chat[13593]: timeout set to 3 seconds
Feb 14 15:58:50 eevui chat[13593]: abort on (\nBUSY\r)
(ここまでpppdのスタート、中略)
Feb 14 15:58:50 eevui chat[13593]: OK
Feb 14 15:58:50 eevui chat[13593]:  -- got it
Feb 14 15:58:50 eevui chat[13593]: send (ATDPxxxx-xx-xxxx^M)
Feb 14 15:58:50 eevui chat[13593]: expect (CONNECT)
(ここまでダイアル開始、中略)
Feb 14 15:59:18 eevui pppd[13592]: Serial connection established.
Feb 14 15:59:18 eevui pppd[13592]: Using interface ppp0
Feb 14 15:59:18 eevui pppd[13592]: Connect: ppp0 <--> /dev/ttyS0
((ここで接続完了、中略)
Feb 14 15:59:49 eevui pppd[13592]: Connection terminated.
Feb 14 15:59:50 eevui pppd[13592]: Hangup (SIGHUP)
Feb 14 15:59:50 eevui pppd[13592]: Exit.
(接続断)
```

■入力

ログからの入力は、ここまで説明してきたとおりです。わからない人は、もう一度読み直しです。答えはもちろん、openメソッドでログファイルをオープンし、1行ずつ読み込む、です。今回の例題ではeach_lineメソッドを使って読み込むことにします。

■ログの解析

接続完了を意味する文字列にマッチするパターンと、接続断を意味する文字列にマッチするパターンの両方を探します。見つかった行は、配列historyに格納しておきます。

■集計と出力

配列historyの中身を解析します。ログファイルの先頭部分の日付をparsedateライブラリを使って解析します。でも、この日付には年の情報がないので、とりあえず今年の値を埋め込んでいます（「start[0] = stop[0] = Time.now.year」の部分）。当然、年末年始にはおかしなことになるでしょうから、そのときまでに解決しておく必要がありますね。このような問題の先送りが2000年問題の本質だったのですが…。まぁ、歴史は繰り返すということで。

それはともかく、parsedateライブラリの結果からTimeオブジェクトを取り出し、時間の計算をします。一度Timeオブジェクトが得られれば、時間の計算は楽勝です。最後に合計を出して終わりです。

この例題の答えとなるプログラム「ppplog.rb」の出力（一部）は、**リスト2**のようになります。これを見ると、接続時間が6時間弱ということがわかります。長いというか、短いというか。

さて、肝心のppplog.rbを**リスト3**に示しておきますので、参考にしてください。

```
$ ruby ppplog.rb
```

とすれば、ログを出力してくれるようになっています。

もしも、このサンプルをそのまま使うなら、先頭のログへのパスと接続の開始と終了を示す正規表現の部分を修正してカスタマイズしてください。

あ、そうそう。PPPのログファイルは、システムによっては一般ユーザーによる読み込みが禁止されている場合があります。そのときは、当然うまく動作しませんから、suコマンドなどでrootユーザーになって

```
# chmod a+r /var/log/ppp.log
```

のようにして、一般ユーザーへの読み込みを許可してやる必要があります。

求む、要望

今回は、駆け足でRubyの入出力クラスについてまとめてみました。いかがでしたでしょうか？　ちょっと地味だったかもしれません。でも、地味がわかってないと派手も始まらないのですよ。

もしも、読者のみなさんで「Rubyのここが詳しく知りたい」「こんなことを取り上げてほしい」という要望がありましたら、編集部まで連絡してください（e-mailは、linuxmag@ml.ascii.co.jpです）。

あと、連載のサブタイトルも募集しています（お気づきでしょうか？　映画のタイトルのもじりになっています）。実はこれが一番たいへんで、前回はいいのが思いつかなくて…、いやそんなことはないです。本当に。

▼リスト2　整形されたPPPのログ（一部）

```
Fri Jan 21 01:54:49 JST 2000 - Fri Jan 21 01:57:17 JST 2000 = 2 min 28 sec
Fri Jan 21 11:32:56 JST 2000 - Fri Jan 21 11:33:47 JST 2000 = 51 sec
(中略)
Tue Feb 15 18:45:56 JST 2000 - Tue Feb 15 18:46:49 JST 2000 = 53 sec
Tue Feb 15 19:31:42 JST 2000 - Tue Feb 15 19:32:19 JST 2000 = 37 sec
Tue Feb 15 22:33:29 JST 2000 - Tue Feb 15 22:34:29 JST 2000 = 1 min 0 sec

from Fri Jan 21 01:54:49 JST 2000
to Tue Feb 15 22:34:29 JST 2000
Total: 5 hour 54 min 11 sec(21251 sec)
```

▼リスト3　PPPログ整形スクリプト「ppplog.rb」

```
#!/usr/bin/ruby

require 'parsedate'
include ParseDate

# この部分をカスタマイズする
Log = "/var/log/ppp.log"
ConnectPattren    = /Connect: ppp0 <-->/
DisconnectPattern = /Exit./

f = open(Log, "r")
history = []
loop do
  entry = []
  # 開始部分の検索
  f.each_line do |line|
    if ConnectPattren =~ line
      entry[0] = line
      break
    end
  end
  # 終了部分の検索
  f.each_line do |line|
    if DisconnectPattern =~ line
      entry[1] = line
      break
    end
  end
  history.push(entry)
  # ファイルの終りまで来たら中断
  break if f.eof?
end

# 秒数から「時・分・秒」表示に
def disptime(sec)
  if sec < 60
    sprintf("%d sec", sec)
  elsif sec < 60*60
    sprintf("%d min %d sec", sec/60, sec%60)
  elsif sec < 24*60*60
    sprintf("%d hour %d min %d sec", sec/3600, (sec%3600)/60, (sec%3600)%60)
  end
end

sum = 0
timerecord = []
history.each do |start, stop|
  # 時刻の解析
  start = parsedate(start[0,15])[0,6]
  stop = parsedate(stop[0,15])[0,6]
  # 年を埋める（苦しい）
  start[0] = stop[0] = Time.now.year
  # Timeオブジェクトの生成
  start = Time.local(*start)
  stop = Time.local(*stop)
  time = stop-start
  STDOUT.printf "%s - %s = %s\n", start, stop, disptime(time)
  sum += time
  timerecord.push([start, stop])
end
# 一番古い記録の開始時からから一番新しい記録の終了時まで
STDOUT.printf "\nfrom %s\nto %s\n", timerecord[0][0], timerecord[-1][-1]
# 集計時間の表示
STDOUT.printf "Total: %s(%d sec)\n", disptime(sum), sum
```

PostgreSQLを極める

さまざまなデータを管理するデータベースでは、データに対するセキュリティが不可欠です。しかし、適切なセキュリティを施すには、そのデータベースシステムが持つセキュリティモデルを正確に把握する必要があります。そこで今回は、PostgreSQLのセキュリティの仕組みと、その設定方法について説明します。

第6回 運用と管理 ―セキュリティ編―

文：片岡裕生
Text : Hiroki Kataoka

Database

PostgreSQLを本格的に利用するようになると、セキュリティも考慮する必要に迫られてきます。場合によっては、権限を持っているユーザー以外に見られたくないデータも扱うこともあるはずです。たとえば、個人のプライバシーに関わる情報などはその最たる例といえるでしょう。

データのセキュリティがしっかり管理されていないシステムでは、いくら機能が充実していてもまったく使い物になりません。そこで今回は、PostgreSQLではどのようにしてセキュリティを実現し、どのように設定すればいいかということを紹介します。

3段階のセキュリティチェック

PostgreSQLでは、大まかに3段階のセキュリティチェックを行っています。具体的には次の3つです。

- **データベースに接続する際にデータベースを利用する権限があるかどうかのチェック**
- **ユーザーIDとパスワードなどによる利用者の確認**
- **利用者にテーブルへアクセスする権限があるかどうかのチェック**

次ページの**図1**に、それぞれのセキュリティチェックの関係を示します。では、これら3種類のセキュリティチェックについてさらに詳しく説明していきます。

ホストベース認証

利用者（もしくはアプリケーション）がデータベースに接続する際の最初のセキュリティチェックのことを、PostgreSQLでは「ホストベース認証」と呼んでいます。ホストベース認証とは読んで字の如く、ホストを基準にしたセキュリティチェックのことで、具体的にはデータベースを利用しようとしているアプリケーションが動作しているホスト（クライアントホスト）が、希望するデータベースに接続する権限があるかどうかをチェックするものです。

一般的なアクセス認証では、利用者に対してアクセス権限のチェックを行うことが多いことと思います。しかし、PostgreSQLの第1段階のセキュリティチェックであるホストベース認証では、クライアントホストそのものがアクセス権を持っているかをチェックします。利用者に対するアクセス権限のチェックは次の第2段階で行われます。

なお、PostgreSQLサーバ内に複数のデータベースがある場合には、それぞれのデータベースごとに接続権限を持つクライアントホストの指定が可能です（**図2**）。

ユーザー認証

クライアントホストがデータベースの利用を許可されている場合、次にPostgreSQLは第2段階のセキュリティチ

ェックである「ユーザー認証」を行います。このユーザー認証では、利用者がPostgreSQLに登録されたユーザーかどうかをチェックします。

次に挙げるように、ユーザー認証の方法にはいくつかの種類が用意されており、管理者は状況に応じてユーザー認証の方法を自由に選択することができます。

- **trust認証**
- **ident認証**
- **password認証**
- **crypt認証**
- **krb4認証およびkrb5認証**

■trust認証

利用者が名乗ったユーザーIDがPostgreSQLに登録済みかどうかだけをチェックする認証方法です。パスワードなどによる信憑性のチェックは行われません。いいかえるなら、利用者が任意のユーザーIDを名乗ることができるということなので、利用者が十分に信頼できるような場合のみ利用します。

■ident認証

ユーザーIDをクライアントホストのidentサーバから取得するという点をのぞけば、trust認証と同じです。利用者がユーザーIDを自由に名乗れないという点ではtrust認証よりも信憑性が高まりますが、クライアントホスト自体のセキュリティが十分でない場合（たとえば、利用者がクライアントホストを自由に操れる場合）には、identサーバから得たユーザーIDの信憑性も十分ではなくなります。ですから、ident認証はクライアントホストのセキュリティが十分に確保されている場合にのみ利用するようにします。

ちなみに、identサーバはすべての環境で利用できるわけではありませんので、利用する際には事前に確認が必要です（Linuxのディストリビューションは、最初から利用できる場合が多いようです）。

■password認証

その名のとおり、ユーザーIDとパスワードにより利用者を確認する認証方法です。これは比較的一般的な認証方法ですが、クライアントホストからPostgreSQLサーバまでのネットワーク回線上をユーザーIDとパスワードが平文のまま流れることになりますので、社内LANなどの信頼できるネットワーク内だけでの利用をお勧めします。社内LANでも素のままのパスワードが流れるのは困るという場合には、次に示すcrypt認証を利用することになります。

■crypt認証

クライアントホストからPostgreSQLサーバまでのネットワーク回線上を流れるパスワードが暗号化されるという点をのぞけば、ほぼpassword認証と同じです。この認証方法を利用すれば、クライアント／サーバ間の通信にインターネットなどの機密性に欠けるネットワークを利用する場合でも、パスワードが漏洩する危険性を低くできます。

しかし、クライアント／サーバ間でやり取りされるデー

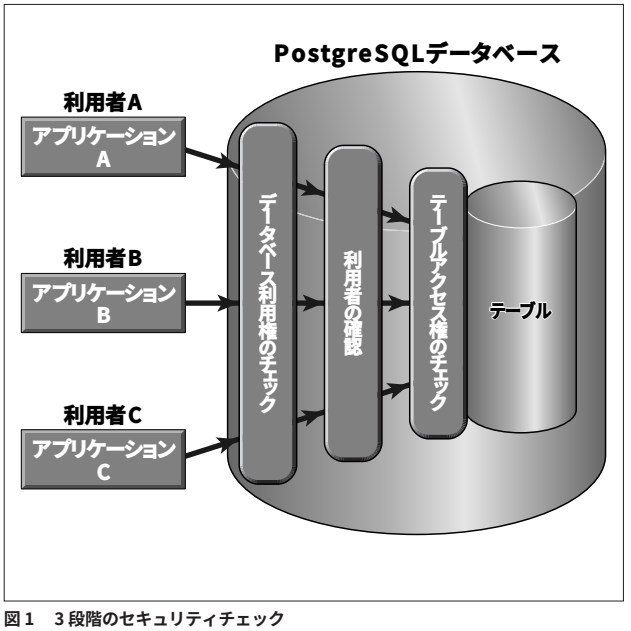


図1　3段階のセキュリティチェック

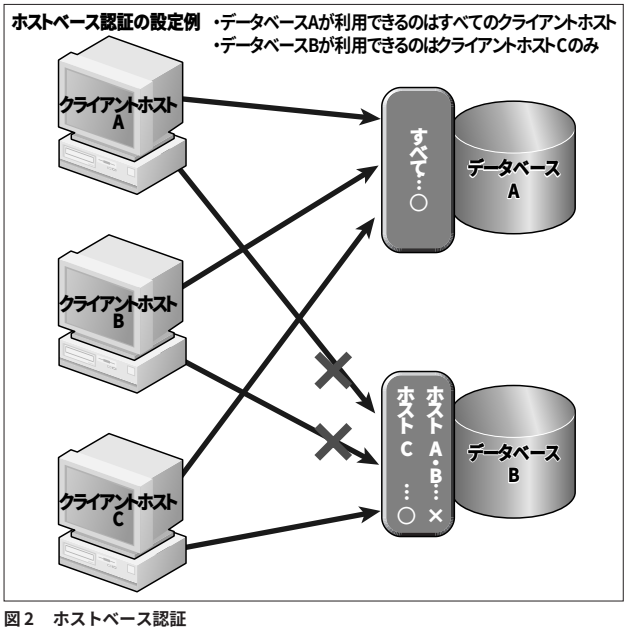


図2　ホストベース認証

タベースの内容そのものについては暗号化されませんので、データそのものに機密性を必要とする場合には、やはり社内LANなどの信頼できるネットワーク内だけで利用することになります。もちろん、別途VPN（仮想プライベートネットワーク）やsshトンネリングなどによる安全な通信経路を確保できるのなら、この限りではありません。

なお、PostgreSQLの関連ツールの中にはcrypt認証に対応していないものもありますので、利用の際には注意が必要です（ODBCドライバなど）。

■krb4認証およびkrb5認証

ユーザーの認証をPostgreSQLで行うのではなく、Kerberosという汎用の認証システムを利用して行う方法です。これに関しては筆者もあまり詳しくないうえに、Kerberosについての解説も必要になってくる可能性がありますので、今回は省略させていただきます。なお、Kerberos認証に興味のある方は、桑村 潤氏のWebサイト（http://www.rccm.co.jp/~juk/）が参考になると思いますので、興味のある方は参照してください。

ホストベース認証の設定方法

PostgreSQLの3段階のセキュリティチェックのうち、最初の2段階に相当する「ホストベース認証」と「ユーザー認証」についての概要を紹介しましたが、ここでは実際にそれらを設定する方法について説明します。

ホストベース認証の設定と、数種類あるユーザー認証の選択は、PostgreSQLのインストールされているディレクトリ（通常は/usr/local/pgsql）の下にあるdata/pg_hba.confという、ホストベース認証の設定ファイルを直接編集することにより行います。

このファイルをLinuxの“ls -l”コマンドで表示した様子を**画面1**に示します。これを見てもわかるように、ホストベース認証の設定ファイルには、オーナー以外に読み書きができないように厳しめのパーミッションが設定されています。セキュリティ情報が格納されているわけですから当然といえば当然ですが、くれぐれも実用システムにおいては設定ファイルの管理にも注意が必要だということを覚えておいてください。

リスト1に設定ファイルの例を示します。“#”で始まる行はコメント行です（空白だけの行も無視されます）。それ以外の行は1行で1つの認証情報を表していて、1つの認証情報にはクライアントホストとデータベース名の組み合わせと、その組み合わせの際に利用すべきユーザー認証の種類が含まれています。たとえば、**リスト1**の例では、“local all”や“host all 127.0.0.1 255.255.255.255”の部分がクライアントホストとデータベース名の組み合わせを表しており、“trust”や“password”の部分がユーザー認証の種類を表しています。

まず、クライアントホストとデータベース名の組み合わせの記述方法について説明します。

リスト1の例では、最初のフィールドが“local”となっている行と“host”となっている行があります。この第1フィールドはクライアントホストのネットワーク種別を表しています。

■ネットワーク種別がlocal

この場合は、クライアントホストはPostgreSQLサーバと同一のマシンであり、かつクライアント／サーバ間の通信にネットワークを利用しないことを意味します。

この行の書式は次のようになります。

```
local  <データベース名>  <ユーザー認証方法>
```

<データベース名>には接続先のデータベース名を指定しますが、ここに“all”を指定することにより、すべてのデータベースが対象になります。<ユーザー認証方法>には、この組み合わせのときに行うユーザー認証の方法を指定します。ここには先ほど紹介したユーザー認証方法のすべての種類が指定できます。また、無条件に接続を拒否する“reject”を指定することも可能です。

▼リスト1　簡単なホストベース認証ファイルの例

```
#
# PostgreSQL host access control file.
#
local  all                              trust
host   all  127.0.0.1  255.255.255.255  trust
host   all  0.0.0.0    0.0.0.0          password
```

```
% ls -l /usr/local/pgsql/data/pg_hba.conf
-rw-------   1 postgres postgres     5311 Jul 29  1999 /usr/local/pgsql/data/pg_hba.conf
```

画面1 ホストベース認証ファイルのパーミッション

■ネットワーク種別がhost

クライアントホストはネットワーク経由でデータベースに接続してくることを意味します。ネットワークを経由する場合はすべてこれに該当しますので、たとえクライアントホストがPostgreSQLサーバと同じマシンであっても、接続先のPostgreSQLサーバとして“localhost”を明示的に指定した場合などは、これに該当しますので注意が必要です。

この行の書式は次のようになります。

```
host   <データベース名>   <ネットワークアドレス>
       <ネットマスク>   <ユーザー認証方法>
```

<データベース名>と<ユーザー認証方法>については、ネットワーク種別がlocalの場合と同じです。

<ネットワークアドレス>と<ネットマスク>には、クライアントホストを識別するためのIPアドレスとネットマスクを指定します。具体的にはクライアントホストのIPアドレスが、<ネットワークアドレス>と<ネットマスク>で定義されたネットワークに含まれているかどうかで判断します。たとえば、192.168.0.0というIPアドレスのクラスCネットワークがあり、このネットワークに含まれるすべてのホストを認証の対象にしたい場合には、<ネットワークアドレス>に192.168.0.0を、<ネットマスク>には255.255.255.0を指定します。あるいは、クライアントホストとしてただ1つのホストを指定したい場合には、<ネットワークアドレス>に許可するクライアントホストのIPアドレスを、<ネットマスク>には255.255.255.255を指定すればいいです。

なお、ネットワーク経由での接続を受けつけるためには、ここでホストベース認証を正しく設定したうえで、さらにPostgreSQLサーバの起動オプションに“-i”オプションを指定しなければなりません。この点は、忘れやすいことですので注意してください。

▼リスト2　詳細なホストベース認証ファイルの例

```
1 #
2 # PostgreSQL host access control file.
3 #
4 local  all                              trust
5 host   all  127.0.0.1   255.255.255.255  trust
6 host   win  192.168.0.3 255.255.255.255  password
7 host   all  192.168.0.1 255.255.255.0    reject
8 host   all  192.168.0.0 255.255.255.0    crypt
9 host   ext  1.2.3.4     255.255.255.255  crypt
```

認証プロセスの流れ

それでは、利用者がデータベースへ接続する際にPostgreSQLが行う認証プロセスを、順を追って見てみましょう。

■ホスト認証

まず、利用者（あるいはアプリケーション）がデータベースに接続しようとすると、PostgreSQLサーバは第1段階のセキュリティチェックであるホストベース認証を行うために、先ほど説明した設定ファイル（pg_hba.conf）から認証情報を読み込みます。そして、利用者の接続要求と一致するクライアントホストとデータベース名の組み合わせを探します。

■ユーザー認証

該当する組み合わせが見つからなかった場合は、その時点でデータベースへの接続は拒否されます。見つかった場合は、その組み合わせに指定されているユーザー認証方法を用いて、今度は利用者の確認を行います。もしも、複数の組み合わせが該当するような場合には、最初の設定（設定ファイル内で、より上の行に書かれた認証情報）が利用されます。

利用者の確認が失敗した場合、当然ながらデータベースへの接続は拒否されます。利用者の確認が成功すれば、晴れてデータベースへの接続が許可されることになります。

以上がデータベース接続時の認証プロセスです。それではここで、設定例を見ながらその内容を解説します。

リスト2を見てください（説明の都合上、行番号を入れてあります）。単純でありながら実用的な設定例を用意しました。設定ファイル内の最初の3行はコメントですから、次の4行目からが有効な認証情報になっています。4行目では次のように設定しています。

・ネットワークを経由せずに自ホストのPostgreSQLサーバを利用する場合には、どのデータベースに接続する場合でもパスワードを必要としない。

5行目は、4行目の設定をネットワーク経由にまで拡大した設定です。127.0.0.1というアドレスはループバックアドレスといって、自分自身のホストを指しているIPアドレスです。“localhost”というホスト名がこれに該当します。5

行目の設定内容は次のようになります。

・ホスト名に“ localhost ”を指定することにより自ホストのPostgreSQLサーバを利用する場合には、どのデータベースに接続する場合でもパスワードを必要としない。

4行目と5行目の設定は、一般にデータベースの管理作業を容易にするために重要です。ただし、このような設定ができるのは、クライアントホストが十分に信頼できる場合に限ります（たとえば、管理者以外はログインできないマシンなど）。

6行目では、次のように設定しています。

・IPアドレスが192.168.0.3のクライアントホストから、“ win ”データベースに接続する場合には、password認証を利用せよ。

これは、たとえば192.168.0.3のクライアントホストからODBCドライバ経由で“ win ”データベースを利用する場合に考えられる設定です。本来は、より安全なcrypt認証を利用したいのだけど、ODBCドライバがcrypt認証に対応していないために限定的にpassword認証を利用しているわけです

7行目は次のような設定になっています。

・192.168.0.1のホストからは、データベースを一切利用させない。

192.168.0.1がルータやファイアウォールマシンなどの、LANの中では最も外側に配置されているマシンで、比較的危険にさらされている場合を想定した安全策の一例です。この設定は、次の8行目の設定よりも前に記述されていなければ意味がなくなってしまいます。

そして8行目は、LAN向けのデフォルト設定です。

・192.168.0のクラスCネットワークからは、crypt認証を用いることによりすべてのデータベースに接続できる。

最後に9行目は、例外的な設定例です。

・1.2.3.4のクライアントホストからは、crypt認証を用いることにより“ ext ”データベースにのみ接続できる。

制限を設けることにより、LAN外からのデータベースアクセスを許可している例です。

以上が、**リスト2**に示した設定例の各行における設定の意味です。特別に難しいことはないと思います。

ユーザー認証の設定方法

ユーザー認証の設定でやらなければならないことは、ここまで来るとあまりありません。ユーザー認証方法の選択は、先ほどのホストベース認証の設定において行ってしまいました。あとはデータベースの利用者を登録して、認証方法によってはパスワードを設定しておくだけです。このあたりのユーザーの管理などに関連する項目については、先月号で解説してあります。

ホストベース認証／ユーザー認証の補足事項

ユーザー認証方法の1つであるtrust認証は、いちいちパスワードを入力しなくてもよいという点では、とても便利なものです。しかし、同時にそれなりのリスクも伴います。少なくとも、管理者以外の一般ユーザーが利用できるクライアントホストに対しては、無条件にtrust認証を指定すべきではありません。

しかし、PostgreSQLの管理用ツールの中には、パスワードが必要な環境では非常に不便になるもの（pg_dumpall後のリストア作業など）もありますので、trust認証を一切使わないというわけにもいきません。

やはり一番よいのは、管理者以外のログインアカウントを一切持たないPostgreSQLサーバ専用マシンを用意し、trust認証はそのマシンの中だけで利用するように設定しておくことです。データベースの管理作業は、PostgreSQLサーバ専用マシンに管理者としてログインしてから行います。そして、ほかのクライアントホストからのデータベースアクセスには、crypt認証あるいはpassword認証などを利用させるわけです。

アクセス認証

第3段階のセキュリティチェックであるアクセス認証では、利用者にデータベース内のテーブルなどをアクセスする権限があるかどうかをチェックします。この権限は、第2段階までのセキュリティチェックと比べると、非常に細かく定義されており、利用者ごと、テーブルやビューごとに設定することができます。また、権限にも種類があり、

PostgreSQLでは次の5種類（ただし、PostgreSQLではUPDATE権とDELETE権の区別がないため、実質は4種類）の権限を指定することが可能です。

- **SELECT権 … 内容を閲覧する権限**
- **INSERT権 … 内容を追加する権限**
- **UPDATE権 … 内容を変更／削除する権限**
- **DELETE権 … 内容を変更／削除する権限**
- **RULE権 … ルールを作成／削除する権限**

また、アクセスの際に権限が必要となる対象は次のものです。なお、PostgreSQLのスーパーユーザーはアクセス認証で拒否されることはありません。

- **テーブル／ビュー**
- **インデックス**
- **シーケンス**

```
% psql template1
Welcome to the POSTGRESQL interactive sql monitor:
        :
template1=> CREATE DATABASE ascii6;
CREATEDB
template1=> ¥c ascii6
connecting to new database: ascii6
ascii6=> CREATE USER user1;
CREATE USER
ascii6=> CREATE USER user2;
CREATE USER
ascii6=> ¥c - user1
connecting as new user: user1
ascii6=> CREATE TABLE table1 (
ascii6->     himitsu1 text, mitene1 text
ascii6-> );
CREATE
ascii6=> INSERT INTO table1 VALUES (
ascii6->     '秘密！', '見てね'
ascii6-> );
INSERT 204480 1
ascii6=> ¥c - user2
connecting as new user: user2
ascii6=> CREATE TABLE table2 (
ascii6->     himitsu2 text, mitene2 text
ascii6-> );
CREATE
ascii6=> INSERT INTO table2 VALUES (
ascii6->     '秘密！', '見てね'
ascii6-> );
INSERT 204481 1
ascii6=>
```

画面2　実験環境の準備

それでは、アクセス認証についてさらに詳しく説明していきましょう。

■オーナーについて

上に挙げたアクセス対象には必ず持ち主が存在します。持ち主とは、アクセス対象をCREATE TABLE文などにより作成した人のことで、「オーナー」と呼びます。オーナーにしかできないことには次のようなことがあります。

- **構造の変更や削除**
- **アクセス権の認可と取り消し**

「構造の変更」や「構造の削除」というのは、アクセス対象そのものの構造の変更（ALTER TABLE文）や削除（DROP TABLE文など）のことです。オーナーは、他人にアクセス権を認めることができますが、構造の変更や削除の権利を認めることはできません。

他人にアクセス権を認めるにはGRANT文を利用します。

■GRANT文

```
GRANT <アクセス権>
ON <アクセス対象>
TO <ユーザー名>
```

＜アクセス権＞には与えたい権限を指定します。権限には“SELECT”、“INSERT”、“UPDATE”、“DELETE”、“RULE”があり、“,”（カンマ）で区切ることにより複数の権限を同時に与えることができます。もしも、一度にすべての権限を与えたいのであれば、ただ単に“ALL”と指定することも可能です。

＜アクセス対象＞には、与えた権限を適用するアクセス対象を名前（テーブル名など）で指定します。アクセス対象にはテーブル、ビュー、インデックス、シーケンスの4種類があり、これも“,”（カンマ）で区切ることによって複数のアクセス対象を一度に指定することが可能です。

＜ユーザー名＞には、権限を与えるユーザー名を指定します。“GROUP”キーワードを前に置くことにより、グループ名も指定することもできます。また、ユーザー名として“PUBLIC”を指定することにより、すべてのユーザーに対して権限を与えることも可能です。

なおPostgreSQLには、一般ユーザーがGRANT文を実行すると、オーナーのアクセス権がなくなってしまうというバグがあります。つまり、持ち主にもかかわらずアクセ

スできなくなってしまうのです。これは困った問題なのですが、こうなってしまってもオーナーであることに変わりはありません。アクセス権を与えることはできますので、すかさず自分にアクセス権を与えてしまいましょう。

それでは、さっそく実験をしてみましょう。まず、**画面2**に示した手順にしたがって、次のような環境を用意します。

- **データベース名は“ascii6”**
- **一般ユーザーは“user1”と“user2”の2人**
- **user1が作成したテーブル“table1”と、user2が作成したテーブル“table2”**
- **アクセス権はデフォルトのまま**

まず、一般ユーザーのuser1として、自分が所有するテーブルであるtable1と他人のテーブルであるtable2を順に表示（SELECT）してみます（**画面3**）。table1を表示したときには問題は起こりませんでしたが、table2を表示しようとしたところでエラー（ERROR: table2: Permission denied.）となりました。この時点ではまだアクセス権の操作はしていないわけですから、他人のテーブルを表示しようとしてエラーになるのは当然です。

次に、GRANT文でuser1にtable2へのSELECT権を与えてから、もう一度同じことをやってみましょう（**画面4**）。GRANT文を実行する前はエラーになっていたtable2の表示が、今度は表示できるようになりました。GRANT文によって、user1にtable2へのアクセス権が認められたということです。

今度は、たった今認めたばかりのSELECT権を冷酷にも剥奪してみましょう。アクセス権を取り消すにはREVOKE文を使用します。

■REVOKE文

REVOKE <アクセス権>
ON <アクセス対象>
FROM <ユーザー名>

取り消したい権限を指定するという点をのぞけば、GRANT文とまったく同じです。GRANT文のところで挙げたバクについても同様です。このREVOKE文を使って、SELECT権を取り消している様子が**画面5**です。再びエラー（ERROR: table2: Permission denied.）が発生するようになりました。

```
ascii6=> ¥c - user1
connecting as new user: user1
ascii6=> SELECT * FROM table1;
himitsu1|mitene1
--------+-------
秘密！   |見てね
(1 rows)

ascii6=> SELECT * FROM table2;
ERROR:  table2: Permission denied.
ascii6=>
```

画面3 アクセス権がある場合とない場合の違い

アクセス権の意味

さて、アクセス権の認可と取り消しができることはわかってもらえたと思いますが、肝心の5種類あるアクセス権については、まだあまり説明していません。以下では、アクセス対象ごとにアクセス権の意味を簡単に説明いたします。

■テーブル／ビュー

テーブルやビューに新しい行を挿入（INSERT文）するにはINSERT権が必要です。また、内容を閲覧（SELECT文）するにはSELECT権が、更新（UPDATE文）や削除（DELETE文）するにはUPDATE権、もしくはDELETE権が必要になります。さらに、ルールやビューを作成するにはRULE権が必要です。なお、ルールやビューに関しては、アクセス権限に特別な規則がありますので、あとで説明します。

■インデックス

インデックスに対するアクセス権限の設定は、最近のPostgreSQLでは必要性ありません。

```
ascii6=> ¥c - user2
connecting as new user: user2
ascii6=> GRANT ALL ON table2 TO user2;
CHANGE                        ↑PostgreSQLのバグ対策
ascii6=> GRANT SELECT ON table2 TO user1;
CHANGE
ascii6=> ¥c - user1
connecting as new user: user1
ascii6=> SELECT * FROM table2;
himitsu2|mitene2
--------+-------
秘密！   |見てね
(1 rows)

ascii6=>
```

画面4　GRANT文によるアクセス権の認可

以前のPostgreSQLでは、テーブルに対するアクセス権限とは別に、インデックスに対するアクセス権限もチェックされる仕様でした。ですから、ほかのユーザーにテーブルアクセスを許可する際には、インデックスにも忘れずに許可を与えなければなりませんでした。しかし、最近のPostgreSQLでは、ユーザーがテーブルにアクセスする権利を持っている限り、インデックスに対するアクセス権限も与えられると考えていて問題ありません。

■シーケンス

シーケンスを通常の使い方で利用（serial型での利用やnextval関数などによる利用）する場合には、SELECT権とUPDATE権（DELETE権でもよい）の両方が必要です。しかし、

```
SELECT * FROM <シーケンス名>
```

のように使用する場合には、SELECT権だけで十分です。

ルールやビューに関する特別規則

ルールというのは、事前に決めておいた条件と一致したアクセスがテーブルに発生したときに、自動的に任意の追加処理（もしくは代替処理）を実行する機能です（1月号でも紹介しましたね）。PostgreSQLのビューは、実はルールによって実現されていますので、ビューもルールの一種だと考えてください。

PostgreSQLでは、ルールによって自動的に実行される処理へのアクセス権限に特例を設けています。これを文章だけで理解してもらうのは難しいと思いますので、先ほどの実験環境を利用して説明しましょう。

画面5では、user1にはuser2が所有しているtable2へのアクセス権がありませんでした。当然、そのままアクセスしようとすればエラーになってしまいます。しかし実は、

```
ascii6=> ¥c - user2
connecting as new user: user2
ascii6=> REVOKE SELECT ON table2 FROM user1;
CHANGE
ascii6=> ¥c - user1
connecting as new user: user1
ascii6=> SELECT * FROM table2;
ERROR:  table2: Permission denied.
ascii6=>
```

画面5　GRANT文によるアクセス権の取り消し

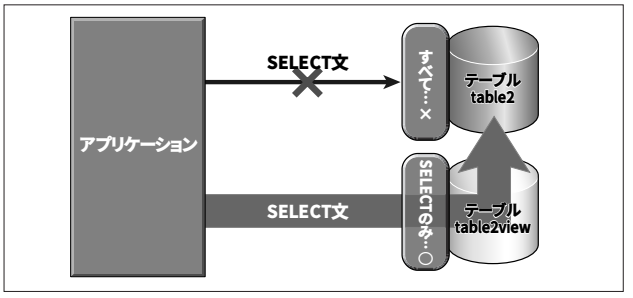


図3　ビューを利用した特例アクセス

user2が秘密にしたいのはtable2の中のhimitsu2カラムの内容だけで、逆にmitene2カラムの内容はuser1にも参照してもらいたいと仮定します。このような場合は、次のようなビューを作成すれば解決します。

```
CREATE VIEW table2view
    AS SELECT mitene2 FROM table2;
GRANT SELECT ON table2view TO user1;
```

要は、公開してもよいカラムだけを含んだビューを別に用意し、このビューに対してアクセス権を与えればいいというわけです（**図3**）。このビューを利用すれば、user1にもtable2のmitene2カラムにアクセスできるのです（**画面6**）。

●

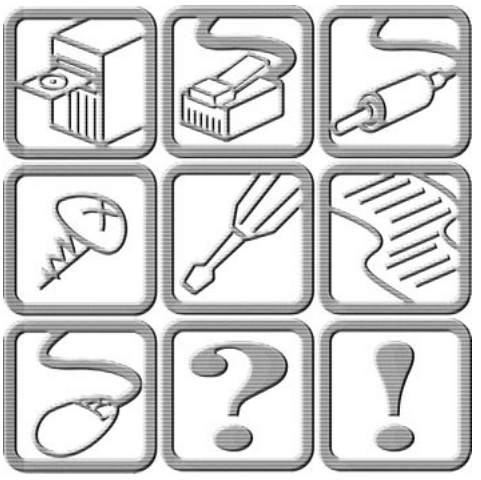
今回もやっぱり定期メンテナンスやバックアップなどの話までたどりつけませんでした。次回こそはこれらについて解説するつもりです。

```
ascii6=> ¥c - user2
connecting as new user: user2
ascii6=> CREATE VIEW table2view
ascii6-> AS SELECT mitene2 FROM table2;
CREATE
ascii6=> GRANT ALL ON table2view TO user2;
CHANGE                          ↑PostgreSQLのバグ対策
ascii6=> GRANT SELECT ON table2view TO user1;
CHANGE
ascii6=> ¥c - user1
connecting as new user: user1
ascii6=> SELECT * FROM table2view;
mitene2
-------
見てね
(1 rows)

ascii6=> SELECT * FROM table2;
ERROR:  table2: Permission denied.
ascii6=>
```

画面6　ビューを利用した特例アクセスの実行例

Try & Try

[trái] [trái]



祝、ノートPC購入

文:政久忠由
Text : Tadayoshi Masahisa

前回やり残した動画（キャプチャデータ）のストリーミング配信をやってみようと思っていたのだけれど、いろいろと忙しかったので満足のいくテストができていない。そこでストリーミング配信はまた別の機会に送るとして、今回は、昨年の暮れに衝動買いをしてしまったノートPCの話をしようと思う。

ノートPCを買う

前にも述べたかもしれないが、僕はモバイルコンピューティングに特別な興味があるわけではないし、出先でカタカタと、どこかしらのネットに接続して何かをする気もない（その必要もないといったほうが正確なんだけど）。スケジュールに関しても暗記で十分対応できるし、そもそも何かしらを持ち歩くことが嫌いなのだ。だから腕時計でさえ身に付けない。でも、さすがに携帯電話を持ち歩かないと多方面に迷惑がかかることがあるので（時計代わりにもなるし）、“なるべく”忘れないようにしている。

こんな僕がノートPCを購入した一番の理由は、家の中で持ち歩きたかったからだ。リビングと自分の部屋は廊下を挟み5m程度離れている。以前は、ネットワークケーブル（最初は10BASE-2の同軸ケーブルだった。今はTだけど）を敷設して、それぞれにデスクトップマシン（ミドルタワー以上のケースしかなかった）を設置していたのだが、リビングでは、ディスプレイ、キーボード、マウスを合わせるとものすごく余計なスペースが必要だし、タバコ（自分の部屋では決して吸わない）や料理調理の影響でキーボードや筐体がすぐにどろどろになるし、お世辞にも快適とはいえなかった。結局、リビングに常設していても、たまにしか使わないので撤収してしまったのだが、なければないで不便さが募るのであった。そこでノートPCというわけだ。

自身でノートPCを所有するのは、10年くらい前のPC-9801 NS/T以来である。当時は競馬で儲かった金の“あまり”で購入したという記憶がある（このマシンは実家で今でもDOS版のLotus 1-2-3の専用機として年に数回稼動しているらしい）。それ以降、ハードウェアの遷り変わりの速さも相まって、使いまわしの利かないノートPCは眼中になかったのだけれど、ここ最近（といっても1年くらい前）、手頃な価格で1024×768表示の可能な製品が発売されていたので、購入に向けて調査を開始したのである。

僕がノートPCに求めることはそれほど多くない。1024×768の表示ができて、そこそこのCPU、そして、それなりのハードディスク容量であれば十分なのである。この要求を満たす機種は各社それぞれ1台以上揃っていた。そこで条件をもう1つ設けることにした。ネットワーク機能を標準で備えていることだ。ネットワーク機能といっても、僕の予定している利用形態ではモデムはどうでもいい、家のネットワーク網（10BASE-T）に接続できることが重要だ。ノートPCの場合、PCカードやUSB機器などでネ

ットワークインターフェイスを追加することができるが、直接内部のPCIバスに接続されたネットワークインターフェイスであったほうが、面倒が少なくて済む。

これらの条件で検討した結果、シャープとソニー、パナソニックなどの機種をピックアップすることになる。しかし、どの機種も決め手を欠いていた。機能的に何かが足りないというわけではないのだけれど、僕の心に“これだ”と決断させるだけの魅力がないのだ。そのため、1年近く悩み続けることとなる。

「欲しいと思ったときが買い時で使い倒すことが大切」とよく言われる。4半期ごとにモデルチェンジが行われるコンピュータハードウェア業界で、待つというのは、あまり得策ではないのだが、僕の場合、新しいハードウェアを待っていたわけではない。たぶんノートPCが必要なかったのだろう。今思うと、そうとしか考えられない。

しかし、その日は突然やってきた。

クリスマスが終わり、お正月まであと数日といったある日、僕は別の用事で外出していた。特に暖かくも寒くもない日で、町はお正月を迎える準備に追われる人々であふれていた。僕は都会の喧騒に逆らうことなく、家の大掃除の時に取り替える電球、蛍光灯などを買いに量販店へと向かった。

量販店は、クリスマス・お正月商戦の真っ只中で、“当たりがでると10万円分が無料になる”キャンペーンなるものが行われていた。取らぬ狸の皮算用。ギャンブルにはめっぽう目がない僕は、瞬間的に当たった時のことを考え、体はすでにノートPC購入へと動いていた。買うのは今しかない、と。

僕にとってギャンブルと高額商品の購入には切っても切れない縁がある。今まで買った高額商品のほとんどすべては、ギャンブルで手にした金によるものだからだ。でも、ギャンブルでプラスになっているわけではない。トータルとしてはトントンでしかない。100万円儲かったとしても、100万円分は負けている。まぁ、コツコツ貯めることに魅力を感じず、いちかばちかの勝負に陶酔していたわけだが、ここ最近、閃きが少なくなってきたのでギャンブルは自重していた。そんなこともあり、いとも簡単に僕のギャンブル魂はくすぐられてしまったわけだ。

クリスマス直後ということもあり、量販店の在庫はかなり少なくなっていた。僕の第一希望と第二希望の機種は入荷待ちで予約しかできない状況だ。仕方なく、その場で持ち帰りができる機種の中から選択することにした。さしたる機種は残っていなかったのだが、僕がピックアップした機種に唯一該当するものがあった。Panasonicの「CF-M1V」だ。

Panasonicは、ノートPCにLet's noteというシリーズを展開している。CF-M1Vはその一員で、比較的薄いながらもCD-ROMドライブ内蔵で重量約1.6kgと持ち運びも楽な機種だ。僕はLet's Noteに良い印象を持っていなかったのだけれど、スペック的にも問題なく、店頭で触ってみると思ったより悪くない。昔、不恰好で、中途半端な設計だなぁと思っていたのは、やはり過去のことのようだ。

早速、CF-M1Vを購入すべくレジに並んだ。もちろん10万円分が無料になる当たりが出ることを信じて。

結果的には当たりは出なかった。会計作業は淡々と行われ、通常の金額が請求された。100人に1人の確率と謳われていたのでかなりの期待で望んだのだが、僕に女神は微笑まなかった。支払いを行っている最中、店内放送ではほかの人の当たりを知らせる威勢のいいアナウンスが流れていた。

支払いを済ませた後、気を取り直し、ついでに買っておくものを考えることにした。特にこのマシン用に必要なものはなかったが、Windows 2000をインストールするので、メモリを64Mバイト追加して128Mバイトにしておくことにした。もちろん増設メモリは、サービスポイントですべてまかなうことができた。

帰路、僕は手荷物の重さを実感しながら、先ほどの決断、衝動買いのことを回想していた。そこには買ったことに対する後悔はなかった。ただ、レジに並ぶタイミングがもう少し早かったり、遅かったりしたらどうだったのだろうかと思ってはみた。しかし、そんなことよりも早く梱包をとき、遊びたいという気持ちが胸いっぱいに広がっていった。我が家へ急いだ。

OSのインストール

梱包をとかれたニューカマー「CF-M1V」は、初期不良もなくご機嫌に動作した。早速、マニュアルにざっと目を通し（本当は電源を入れる前に読まなくちゃね）、プレインストールOSの最終設定を行い、BIOSの調整、メモリの増設も行った。すべては順調だ。

いよいよ、Linux（とWindows 2000）を導入するわけだが、メーカー製マシンは通常プレインストールOS以外の動作保証はない。CF-M1Vはそれほど特殊なハードウェアで構成されていないことは確認済みだが、一応心配なので、プレインストールOS、Windows 98 SEの区画は残

しておくことにした。まずは、インストール済みのさまざまなソフトウェアを削除する（僕にはほとんどのソフトウェアは必要なかった）。そしてさっぱりしたところで、デフラグメントを行いパーティション変更に備える。

CF-M1Vのハードディスク（約8Gバイト）のデフォルトパーティション構成は、ハイバネーション用に200Mバイト、あとはFAT32で通常のCドライブ用となっている。今回はこれを6.7Gバイト（Windows 2000用）、600Mバイト（Linux用）、700Mバイト（Windows 98 SE用）にした。

手順としては、現在のパーティションを縮小し、ハードディスクの後ろのほうに600Mと700Mのパーティションを作成する。ディスクエディタで直接行えるレベルだし、適当なツールを使えば、特に難しい作業ではないので、詳細は省略する。700Mのほうは FAT32でフォーマットし、既存のWindows 98 SEをコピー（Win98自身で自分をコピーさせる）し、ブートできるようにSYSコマンドでシステムを転送する。またそのパーティションは、FDISKなどでアクティブパーティションに設定する。その段階で700Mにコピーした Windows 98での動作確認がとれれば、既存のパーティションを削除して、Windows 2000用の6.7Gバイトのパーティションにするだけだ。

結果としてCF-M1Vのハードディスクのパーティション構成は次のようになった。

```
Device  Boot    Start   End    Blocks    Id   System
/dev/hda1       1       832    6683008+  7    HPFS/NTFS
/dev/hda2       833     906     594405   5    Extended
/dev/hda3 *     907     996     722925   c    Win95 FAT32
                                                (LBA)
/dev/hda5       833     901     554211   83   Linux
/dev/hda6       902     906      40131   83   Linux
```

Linuxのパーティション構成が拡張パーティションで2つに分かれる形になってしまっているが、本当はこの構成にしたかったわけじゃない。たまたま手元にあったRed Hat Linuxがスワップ領域を作成しないとインストールを進めさせてくれないので、一時的に40Mバイトの領域を用意したまでだ。あとで/varの格納先としてその領域を使用している。

今回ブートにはMBRにインストールしたLiloを使用し、LinuxとWindowsを選択できるように設定した。またWindows系OSの選択は、ブートフラグを立てたWindows 98のパーティションでWindows 2000のブートローダが担当することにした。ごくごく一般的な構成である。

■ Linuxの導入

パーティションの構成が終わったらLinuxの導入だ。CF-M1Vの場合、CD-ROMインターフェイスやネットワークインターフェイスが特殊な接続ではないので、インストールに関して特に問題となる部分はない。しかしながら、今回Linuxの領域は600Mバイトしか獲得することができなかった。もう少し大きくしておけばよかったかな、と思う今日この頃だが、今となっては変更するのも面倒だ。とりあえず、この環境でできることを考えることにしよう。なお、これから述べることは、当然、僕のやりたいことであって、Linuxの領域が600Mバイトしかない場合のジェネリックな考え方ではないので注意してほしい。

まず最低限カーネルソースを展開してコンパイルできる環境にしたい。2.3系のカーネルを構築するには、120Mバイト程度の領域が必要となるので、それ以外の作業領域も考えるとソフトウェアは400Mバイト程度に抑えなければならない。最近のディストリビューションは一般的なデスクトップ環境でも1Gバイト近くの容量を必要とするので、泣く泣くベースシステムと開発環境のみをインストールすることにした。それ以外のパッケージは必要に応じて手動で導入している。インストールの詳細で特に説明することもないので割愛する。

カーネルを作り直す

僕の場合、インストール後、真っ先に行うことはカーネルを作り直すことだ。これを行わないとシステムを使う気にならない。早速、カーネル2.3.4xを展開してコンフィギュレーションを行う。

CF-M1Vでは、CPUはCeleron、チップセットは440BX/ZX（PIIX4）、Ethernetコントローラは Intel 82557、カードバスはRicoh RL5c475、グラフィックス／マルチメディアオーディオコントローラはNeoMagic 256AV、あと一応Lucent Microelectronics WinModem（ドライバがまだないので使えないんだけど）というハードウェアを基に設定していく。またパワーマネージメント（APM）も有効で、ハードディスクはUDMA（33）に対応していることも考慮する。作成したカーネルは大体こんな感じになる。

```
Linux version 2.3.47 (gcc version pgcc-2.95.2
Detected 332169837 Hz processor.
Calibrating delay loop... 330.96 BogoMIPS
Memory: 126860k/131072k available (882k kernel code,
3824k reserved, 63k data, 140k init, 0k highmem)
CPU: Intel Celeron (Mendocino) stepping 0a
PIIX4: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0xfcb0-0xfcb7, BIOS settings:
hda:DMA, hdb:pio
    ide1: BM-DMA at 0xfcb8-0xfcbf, BIOS settings:
hdc:pio, hdd:pio
hda: TOSHIBA MK8113MAT, ATA DISK drive
hdc: UJDA140, ATAPI CDROM drive
hda:  TOSHIBA  MK8113MAT,  7815MB  w/0kB  Cache,
CHS=996/255/63, UDMA(33)
Partition check:
 hda: hda1 hda2 < hda5 hda6 > hda3
eth0:  OEM  i82557/i82558  10/100  Ethernet  at
0xc8800000, 00:80:45:11:22:9E, IRQ 9.
  Primary interface chip i82555 PHY #1.
```

今回、PCカードやUSB、IrDAのテストも行ってみたかったのだが、僕が所有しているものといえば、2MバイトのPCMCIA SRAMカードという使い道に困ってしまうものしかなかったので、これらはまた別の機会に行うことにする（一応Natural Keyboard EliteとIntelliMouse ExplorerをUSB接続してみたんだけど、CF-M1Vとの組み合わせでは、LinuxのUSBモジュールを利用するまでもなく、ハードウェアのレガシーモードとして使用できるので深く追求することはやめた）。

パワーマネージメントに関しては、コンソール、X環境ともにきちんとレジューム、そして復帰できることを確認した。ハードウェアスイッチから、apm -sなどのコマンドラインからでも大丈夫だ。ネットワークインターフェイスの復帰に少々不安があったものの、テストした限りでは数秒後に通信は回復している。

CF-M1VはチップセットにはACPI対応なのだが、システムBIOSが今のところ対応していないのでACPIを利用することができない。ACPI対応BIOSは2月下旬に公開されるようなので、それを待ってテストしてみることにする（ただ、どのデバイスがACPI対応なのかよくわからなかったりするのだけれども）。一応、CF-M1VのEthernetコントローラIntel 82557に対応するドライバ（Ether Express Pro 100）には、パワーマネージメントを有効にするルーチンが追加されているので、もしかするとi82557でも利用できるのかもしれない（i82558とかi82559以降でないとだめかも。これもそのうち調べてみる）。

NFSを利用する

さて、カーネルの再構築が終わり、ローカルシステムの設定やコンソール環境で必要なソフトウェアの導入がある程度完了したら、もう少し実用的な環境について考えておこう。今回一番の問題は、ハードディスク容量が足りないことなので、これをどうにかしなくてはならない。ファイルシステムに圧縮機能を追加してもよいが、よくよく考えてみるとすべてをローカルでまかなう必要はどこにもない。何のためにネットワーク機能があるんだ。そう、共有ファイルシステム、UNIXにはネットワークファイルシステム（NFS）があるじゃないか（まぁ最初からその予定であったのだけれど）。

では早速、常設のLinuxマシンのワークエリアを共有することにしよう（今回homeはモバイル時に問題となるので共有しない方針だ）。Linuxは、NFSクライアント、サーバ機能を標準的に備えているので、ほとんどの場合、新たにソフトウェアを追加する必要はないと思う。

一般にディスクレスマシンではすべてを、管理されたサーバ環境では/homeや/usr/bin、/usr/local/binなどを、NFSを利用して共有している。ここでのノートPCのようなモバイル環境では、どこまでのファイルをローカル側に持ち、どこまでをネットワーク側に置くか、よく考えておく必要がある。少なくともネットワークに接続していない状態でも必要な作業が行えるように。

一般にNFSを利用するうえで気をつけておくことは、それぞれのマシンのユーザーID番号（uid）とグループID番号（gid）だ。ファイルシステムでは、通常アクセス制御が行われるが、これはuidとgidを基にしている。つまりシステム的には、名前よりもこのID番号が重要となる。このことは基本的にはNFSでも同様だ。クライアント側のID番号がサーバ側に送信され、サーバ側のファイルシステムのアクセス制御が行われることになる。NFSの場合、双方のuidとgidのマッピングを動的、静的に設定することも可能だが、ここではユーザーの数が1人か2人の場合を想定しているのでクライアント側でID番号を合わせてしまうことにする。

そこで双方のマシンの/etc/passwdと/etc/groupファ

イルを照らし合わせて、名前とID番号の対応が異なっていないかチェックしておこう。まぁ、個人で利用する分にはセキュリティなんてどうでもよかったりするのだけれど、共有したファイルシステム上で自分がオーナーだと思っていたファイルやディレクトリが1001などと番号で表示されたり、アクセスが拒否されたりすると嫌なのできちんと設定しておこう。

なお、ID番号はrootユーザーになり、直接/etc/passwdと/etc/groupを書き換えるか、usermod、groupmodコマンドで変更することができる。また必要に応じて既存のファイルのオーナーも変更しておこう。

・uid、gidともに505のユーザー（user1）をそれぞれ1001に変更する場合

```
# groupmod -g 1001 user1
# usermod -u 1001 -g user1 user1
# chown -R user1.user1 /home/user1
```

■サーバ側の設定

サーバ側では、提供（エクスポート）するディレクトリを設定する必要があるが、これは/etc/exportsファイルに記述する。書式は次のようなものだ。

#/etc/exportsファイルサンプル

```
/mnt/1          (rw,no_root_squash)
/pub            (rw)
/home           *.local.domain(rw)
/usr/local      (ro)
```

1行目では、/mnt/1を読み書き可能で、rootユーザーのuidをnobodyなどにマップしないようにした設定だ。デフォルトでは、rootユーザーのuid 0、gid 0は、そのまま利用できるとセキュリティ的に問題となる可能性が高いため、nobodyなどにマッピングするようになっている。rootユーザーでネットワークログオン（telnetなど）できないようになっているのと同じ理由だ。2行目では、/pubを読み書きできる設定だ。3行目では、/homeへのアクセスをlocal.domainドメインのホストからの要求に限定している。何も設定しないとDNSホスト名での制限は行われない。4行目は/usr/localをリードオンリーモードで公開している。

企業や大学などで本格的にNFSを利用する場合は、セキュリティを第一に考えなければならないが、ここでは完全にプライベートなネットワークでの利用なのでセキュリティ設定について詳しく述べることはしない。

/etc/exportsファイルの設定が終わったら、NFSサービスを開始する。Red Hat系 Linuxでは次のように入力する（常時サービスを提供する場合は、ntsysvコマンドなどでnfsを起動時に有効になるように設定する）。

```
# /etc/rc.d/init.d/nfs start
Starting NFS services:
Starting NFS statd:
Starting NFS quotas:
Starting NFS mountd:
Starting NFS daemon:
```

■NFSクライアント

それでは接続してみることにする。rootユーザーになりmountコマンドを実行する（書式はサーバ名とエクスポート名をコロン（:)で接続した形だ）。

```
# mount server01:/mnt/1 /mnt/1
# mount
/dev/hda5 on / type ext2 (rw)
server01:/mnt/1 on /mnt/1 type nfs (rw,addr=192.168.0.1)
```

mountコマンドのオプションでバッファサイズを指定することも可能だ。デフォルトは1024バイトなので、次のように指定すれば、読み書きそれぞれのバッファサイズを8192バイトに設定することができる。これによりパフォーマンスが向上する場合もある。

```
# mount server01:/mnt/1 /mnt/1 -o rsize=8192,wsize=8192
# mount
/dev/hda5 on / type ext2 (rw)
server01:/mnt/1 on /mnt/1 type nfs (rw,rsize=8192,
                                wsize=8192,addr=192.168.0.1)
```

ちなみに常時接続して利用するのであれば、/etc/fstabに次のような行を追加しておく。

```
server:/mnt/1   /mnt/1   nfs   rsize=8192,wsize=8192
```

さらに、モバイル環境で必要なときにしか接続しない場合で、セキュリティ的に問題なく、接続するのにrootにス

イッチするのも面倒だというのであれば、次のようにnoautoとuserを設定しておくと、各ユーザーレベルでマウントできるようになる。

```
#/etc/fstab
server:/mnt/1   /mnt/1   nfs   rsize=8192,wsize=8192,noauto,
                                                    user

[user01]$ mount /mnt/1（一般ユーザーでマウント可能）
```

また、/etc/exportsで指定したno_root_squashの違いについても確認しておこう。設定の異なる共有ディレクトリをそれぞれマウントし、rootユーザーのままディレクトリなどを作成してみよう。no_root_squashを指定している場合は、問題なく作成できると思うが、指定していない場合は、Permission deniedなどのメッセージが表示され、何もできないと思う。UNIXではrootユーザーの権利は非常に強大なので、このような措置が図られている。

というわけで、ハードディスク容量が足りないという問題は一応の解決をみた。

```
$ df
Filesystem  1k-blocks      Used   Available  Use%   Mounted on
/dev/hda5   545484        473728       44048   91%          /
/dev/hda6   38859          20893       15960   57%        /var
server01:/mnt/1 5668772   428660     4952148    8%       /mnt/1
server01:/home 4047904   2633420     1208856   69%       /mnt/2
```

Xも入れておこう

コンソールだけでは何だか少し寂しいので、Xもインストールしてみることにした。ここでは、XFree86 Version 3.9.17を使用する。ローカルには到底ソースコードを展開できないので、/mnt/1上の適当なディレクトリに展開し、make Worldを実行する。続いてrootユーザーになりmake installを実行する。この時、いくつかファイルを書き換えているようなので、no_root_squashを指定していないとmake installがきちんと完了しないと思う。注意してほしい。インストールが終わったら、xf86configで環境に合わせて設定を行う。テキストベースだが、選択はそれほど難しくない。

まだウィンドウマネージャをインストールしてないけど、Xサーバが動作するかどうか確かめておく。/usr/X11R6/bin/XFree86を実行する。

デフォルトのままだと、たぶん致命的なエラーでXサーバは実行できないと思う。CF-M1Vに搭載されたNeoMagic 256AVだからなのか、NeoMagic全般にいえることなのかはわからないが、プローブでVGAチップしか見つけられないようだ（SuperProbeコマンドを実行してみればわかる）。XFree86はグラフィックスチップセットを自動検出して、それに合ったドライバを読み込むようになっているのだが、そこで失敗している。ただ、PCIバスを検索してNeoMagicの存在には気づいているようだ。

対処としてはドライバを直接指定するしかなさそうなので、/etc/X11/XF86Setupを開き、該当するIdentifierのDriverの指定をvgaからneomagicに変更する。これで問題なく動作するようになると思う。

```
XFree86 Version 3.9.17  / X Window System
(--) PCI: (1:0:0) Neomagic NM2200 rev 18, Mem @
0xde000000/24, 0xdf800000/22, 0xdff00000/20
(II) NEOMAGIC: Driver for Neomagic chipsets:
neo2070, neo2090, neo2093, neo2097, neo2160, neo2200
pciIo_MemAccessDisable: 0x10000
NEOMAGIC instances found: 1
(--) Chipset neo2200 found
```

Xの環境をインストールしたことで、65Mバイトもの容量を消費してしまうことになったが、快適なのでしばらくこのまま使用することにした。ちなみにウィンドウマネージャはtwmに決定した。

それほどの期待をしていなかったこともあって、CF-M1Vの印象は悪くない。でも、すでに少し速いCPUを搭載した新製品がリリースされたので、CF-M1Vは旧モデルということになってしまった。まぁ、そんなことはどうでもよいのだが、唯一、キーボードが気に入らない。僕は、106キーボードの配列は大嫌いだ（なんで変換、無変換など無駄なキーを追加した変なキーボードを日本の標準にしてしまったんだ）。仕方なく刻印を無視して101の配列で使っているのだけれど、バックスラッシュの位置など、結構、頭が痛い。ん〜、CF-M1V用の英語キーボード、誰か知りませんか？　Panasonicさん、101キーボード作って5000円くらいで売ってちょ。

付録CD-ROM Disk2に収録した
TurboLinux 4.5ノンサポート版のインストール

CD-ROMブートができる場合は、付録CD-ROM Disk2をCD-ROMドライブに入れ、CD-ROMから起動します。CD-ROMブートができない場合は、DOSかWindows環境でブートディスクを作成する必要があります。この場合、フォーマット済みのフロッピーディスクをAドライブにセットし、CD-ROMの\dosutilsディレクトリに移動して、ここにあるバッチファイルでブートディスクを作成します。通常はboot.batファイルで作成するブートディスクでよいでしょう。

CD-ROM、またはブートディスクから起動し、“boot:”のプロンプトが表示されたらEnterキーを押します。プロンプトに対し、“einstall”と入力し、Enterキーを押すとメッセージが英語になります。

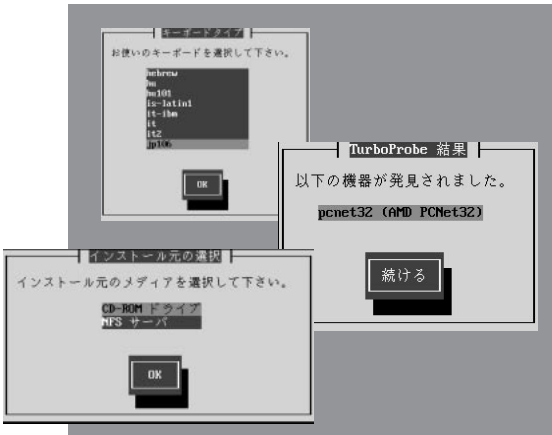
これ以降のインストーラの操作は、[Tab]キーで項目の移動、矢印キーで選択の移動、スペースキーで選択、Enterキーで決定となります。

キーボードタイプからインストール元の選択まで

最初にキーボードの選択をします。国内でよく使われている106キーボードや109キーボードの場合は、「jp106」を選択します。英語キーボードを使っているなら、「us」を選びます。

次に、ハードウェアの自動設定を行うか質問されます。よほど特殊なハードウェアを使っていない限りは、「OK」を選択します。インストーラがハードウェアを発見すると、右の写真のように表示します。

続いてインストール元を選択します。ここでは、「CD-ROMドライブ」を選びます。



パーティションの設定

ハードディスクに、Linuxをインストールするためのパーティションを設定します。

「自動」を選択すると、ハードディスク内のパーティションに割り当てられていない空き領域や既存のLinuxパーティションを使って、「/（ルート）」や/usrのパーティションなどとスワップ用のパーティションを適切なサイズに割り当ててくれます。Windowsパーティション内部に空き領域があるだけでは「自動」は選択できません。また、消したくないLinuxパーティションがある場合には、「自動」を選択してはいけません。

「手動」を選択すると、fdiskコマンドを使って自分でパーティション構成を決定できます。最低限必要なのは、「/」パーティションとスワップ用パーティションです。間違えて必要なパーティションを消さないように、十分に注意して作業を行ってください。

ネットワークの設定

ネットワークカードが自動検出されるか、手動で指定した場合、ここでLANの設定を行います。指定する必要があるのは、IPアドレス、ネットマスク、ゲートウェイ、ネームサーバ、ドメイン名、ホスト名の各項目です。

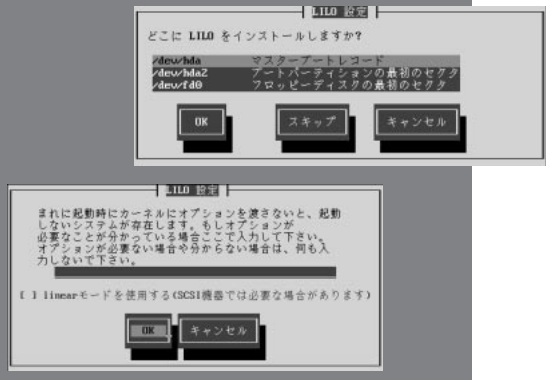
4

インストールタイプ、カーネル選択

インストールタイプでは、「ネットワークサーバ」「ワークステーション」のように用途別にパッケージを選ぶか、「直接選択」でパッケージごとに選択するかを決められます。ハードディスクの空き容量に余裕があるのなら、「すべて」を選択するといいでしょう。

パッケージのインストールは、「すべて」を選んだ場合、相当速いマシンでも30分くらいかかります。しばらく待ちましょう。

その後、カーネルを選択します。

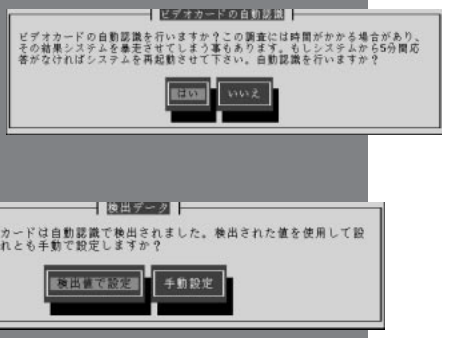


5

LILOの設定

LinuxのブートローダであるLILOのインストール先を設定します。Linuxのみ、またはLinuxとWindows 9xのデュアルブートをさせる場合は、「マスターブートレコード」に、System Commanderなどのブートマネージャを利用している場合は、「ブートパーティションの最初のセクタ」を選びます。Linuxを使う時は、必ずフロッピーディスクで起動したいなら、「フロッピーディスクの最初のセクタ」を選択します。

よほど特別なハードウェア構成でないかぎり、カーネルに渡すオプションを指定する必要はありません。



6

ビデオカードの自動認識

最新のビデオカード以外は、自動で認識させることが可能です。自動認識後、「検出値で設定」を選べば、ハードウェアの設定や、Xサーバの選択まで行われますので、まずは自動認識でやってみましょう。

7

キーボード、マウスの設定

ここから、X Window System関係の設定を行います。キーボードの設定は、最初に行ったものと同じです。

マウスは自動検出に成功していれば、デフォルトとして選択されています。2ボタンのマウスを使う場合には、「3ボタンのエミュレーション」をチェックしておきます。これによって、左右ボタンを同時に押すことで、中ボタンを押したことになります。

8 表示モードの設定

デフォルトの色数、解像度などを設定します。解像度を複数指定しておくと、X Window System使用中に、「Ctrl」＋「Alt」＋「−」、「Ctrl」＋「Alt」＋「＋」で切り替えることができます。次に、フォントの解像度を指定した後、実際にXサーバを起動して、設定が正しいかどうか確認します。

9 ログイン方法、時間帯、プリンタの設定

ログイン方法は、グラフィカルとテキストのいずれかを選択します。グラフィカルログインを選ぶためには、X Window Systemが正しく起動することが確認できていなければなりません。

次に時間帯の設定をします。日本国内なら「Japan」のままにしておきます。「ハードウェアクロック」を「グリニッジ標準時に設定」するのは、あまり一般的ではありません。

「プリンタの設定」では、「追加」を選択すると、プリンタの設定が行えます。

10 起動サービス設定、ウインドウマネージャの選択

起動時に実行させるサーバの指定を行います。設定を終了したら、「Esc」キーを押します。

続いて、デフォルトで使用するウインドウマネージャを指定します。

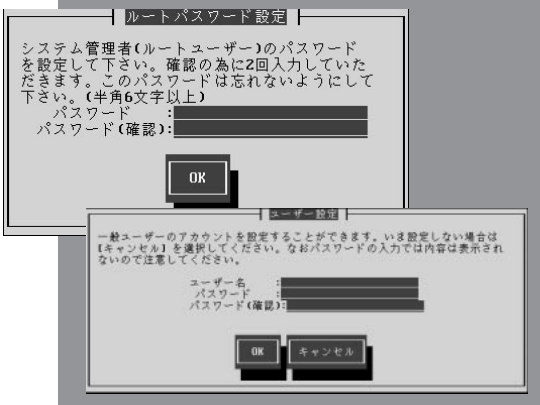
11 パスワードの設定（root、一般ユーザー）

「ルートパスワード設定」では、管理者であるrootのパスワードを設定します。確認のため、2度入力する必要があります。rootは強力な権限を持ち、すべての操作を行えるようになっています。自分では覚えやすく、他人に類推されにくいパスワードを設定しましょう。

続いて一般ユーザーのアカウント作成と、パスワードの設定を行います。繰り返しますが、rootは強力な権限を持っています。そのため、通常の作業をrootで行うと、重要な設定ファイルを消してしまうなどの危険があります。ですから、一般ユーザーを作っておいて、ふだんはこちらでログインする習慣をつけましょう。

なおroot、一般ユーザーのどちらでも、セキュリティの観点からパスワードは画面に表示されません。

以上でインストール作業は終わりです。



Books

ＯＳ戦線 異変あり

ロバート・ヤング、ウェンディ・ゴールドマン・ローム 著 倉骨 彰 訳
日経BP社
四六判／308ページ　　本体価格 1600円

Linuxは今、Windowsに唯一対抗できるOSと考えられているが、それはオープンソースという形で大勢の技術者が無償で開発したものだ。それゆえマイクロソフトは特定の敵が見えない状況を恐れているのである。

そのLinuxをパッケージ化して販売し、付随する各種サービスを提供することでビジネスとして成功したRed Hat社は、昨年、創業から4年でNASDAQに株式を公開した。

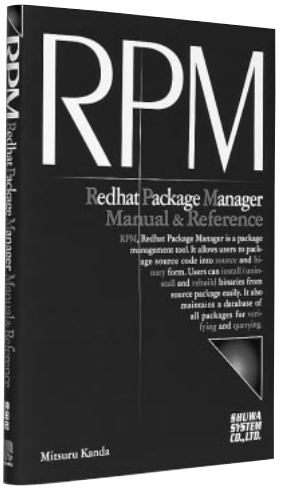
IPOの前にIBMやIntel、Compaq、Dell、Oracleといったビッグカンパニーから出資を受けているが、本書ではそれにまつわる秘話をRed Hat社CEOのロバート・ヤングが自ら語っている。またLinus氏の発言や考え方、Netscape社がWebブラウザのNavigatorをオープンソースに提供するときの具体的な状況などを交え、この新しいビジネスモデルが成立するまでのコンピュータ業界の様子を描いたノンフィクションである。

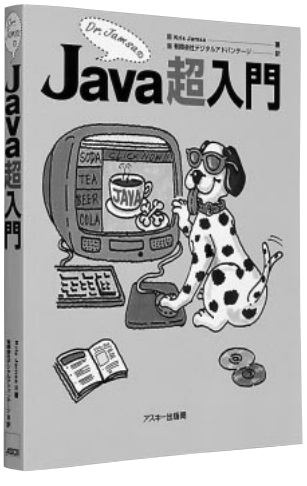
Redhat Package Manager Manual & Reference

神田 充 著
秀和システム
A5判／192ページ　　本体価格 2400円

数多くのディストリビューションがある中で、Red Hat Linuxとそれをベースにした、いわゆるRed Hat系のディストリビューションが主流になってきている。その理由のひとつ、インストール、アンインストールの作業が簡単で、ファイルの依存関係もチェックしてくれるRPMパッケージマネージャによるところが大きいだろう。

しかし、ほとんどのユーザーは、“ rpm -Uvh ＜ファイル名＞ ”としてインストールするためにしか使ったことがないのではないだろうか。違うプラットホームのマシンにインストールするときや、ライブラリのバージョンに違いがあってもSRPMパッケージを使ってうまくインストールできることもある。作ったプログラムをWebで公開する場合にはもちろんだが、自分自身で使うときにもRPMパッケージにしておいたほうが便利である。RPMパッケージの作り方を解説した本書は、中上級者向けだが勉強する価値はあるだろう。





Dr.JAMSAのJava超入門

Kris.Jamsa 著 デジタルアドバンテージ訳
アスキー
B5変形判／362ページ　　本体価格 2200円

JavaはSun Microsystemsが開発したプログラミング言語で、主にインターネットなどWebブラウザ上で、ダイナミックな動きを表現させたり、キーボードやマウスの入力に応じて違うアクションを行わせたりするときに利用されている。クライアント側にJavaの実行環境を備えてあれば、OSやCPUといったハードウェアに依存しないため、WindowsでもLinuxでも、iMacでも携帯端末であっても同様に動作することが特徴である。

本書は、最初にSunのWebサイトからJDK（開発キット）をダウンロードしておけば、あとはWindows上でプログラミングと動作確認を行うようになっていて、ひとつひとつのレッスンに沿って試していくことで、自然にJavaが身に付くようなサンプルが掲載されている。タイトルに「超入門」とあるように、まったくの初心者でプログラミングについての知識がない人にもわかるように構成されているので、Javaを勉強するには最適だ。

DB2ユニバーサル・データベースfor Linux

菅原 香代子 著
トッパン
B5変形判／308ページ／CD-ROM付き
本体価格 2800円

LinuxPPC R5 日本語版入門キット

臼田昭司 著
秀和システム
B5変形判／256ページ／CD-ROM付き
本体価格 2800円

標準Red Hat Linux プログラミング

David Pitts、Bill Ball 他 著 石川直太 訳
インプレス
B5変形判／368ページ
本体価格 2600円

NetatalkでかんたんLAN構築

白井徹也 著
カットシステム
A5判／200ページ／CD-ROM付き
本体価格 2300円

UNIX/Linux デスクトップ・ツール入門

宝剣 純一郎 著
メディア・テック出版
A5判／276ページ
本体価格 2280円

sendmailとqmailによるLinuxメールサーバー構築ガイド

高橋隆雄 著
エーアイ出版
B5変形判／280ページ
本体価格 2600円

最近売れてるLinux書籍は？

今月のキーワードは「停滞」です。今よく売れているコンピュータ書籍は、大きく分けて、「サーバもの」「資格試験もの」「セキュリティもの」などです。この傾向はLinuxにも当てはまるようです。たとえば、「サーバもの」ではインプレスの『できるLinux サーバー構築編』と『できるLinux インターネットサーバー編』の2冊、「セキュリティもの」では、翔泳社の『PC-UNIXサーバのためのクラッカー撃退計画』やセレンディップの『Linuxで構築するファイアウォール』などがよく売れています。「セキュリティもの」に関しては、ここのところまた盛り返す傾向にあるので、先日の「クラッカー騒動」と多少関連があるのかもしれません。また、Linuxそのものの「資格試験もの」は、まだそれほどありませんが、Osborne McGraw-Hillから『RHCE Red Hat Certified Engineer Linux Study Guide』という資格試験関係の洋書が発売されるようです。

さて、ここまで読んで今月のキーワードの意味に気づいた方もいるかもしれません。そうです。私がこのコラムを担当してから、ずっと同じようなものしか売れていないのです。まったく新しい動きが伝わってきません。このコラムのネタも「停滞」しているわけですが、それはとりもなおさず業界そのものが停滞していることを意味します。

これをLinuxに置き換えてみましょう。書店には、あいも変わらずインストール本と紹介本ばかりがあふれている状況で、あまり新しい動きが感じられません。思い起こせば、Linux magazineが初めにムックとして創刊されてから、ちょうど1年がたとうとしています。あれから、Linuxはどう変わってきたのでしょうか？

なんだかインストールと設定ばかりしてきたようで、新しいテクノロジーも、新しいビジネスモデルも、新しいパラダイムも、未だ作り出せずに1年が過ぎてしまったのではないでしょうか？　残ったのはたくさんのディストリビューションだけ、という気さえします。

Linuxは「停滞」しているのでしょうか？　それとも新たな動きへの「産みの苦しみ」なのでしょうか？　「ディストリビューション」ではなく、そろそろ「Linux」を使い始めてほしいと思う次第です。　（書泉ブックドーム 古田島）

Linux Today

米国LinuxToday提携
http://linuxtoday.com/
毎月、米国の人気Linuxサイト「Linux Today」に掲載された記事の要約をお届けします。記事の全文は日刊アスキーLinuxで読むことができます。
http://www.linux24.com/

訳：日下部圭子

Amazonをボイコットせよ！

Text : Richard Stallman
http://linuxtoday.com/stories/13652.html（原文）
http://www.ascii.co.jp/linux/linuxtoday/article/article375360-000.html（邦訳全文）

■

■Amazonで買うのはやめよう

■

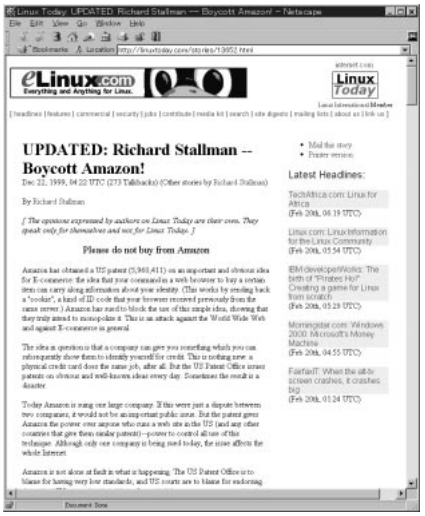
Amazonは電子商取引のための、ある重要で自明なアイデアについて米国特許を取得した。そのアイデアとは、Webブラウザで品物を購入するためのコマンドに、個人の識別情報を含められるようにするというものだ（これは「クッキー」を送り返すことで行う。クッキーとは、ユーザーのブラウザが前もって同じサーバから受け取っている一種のIDコードだ）。Amazonはこの単純なアイデアの利用を妨害する訴訟を起こした。このことはまさに、彼らがそのアイデアを独占しようと考えていることを示している。これはWorld Wide Webや電子商取引全般に対する攻撃だ。

問題のアイデアとは、ある企業が顧客に対し、以降の信用販売で顧客を識別するために提示できる何かを与えることができる、というものだ。これはなんら新しいものではない。なにしろ実際のクレジットカードと同じ機能なのだから。しかし米国の特許庁は、自明でよく知られたアイデアに対して、毎日のように特許を発行している。この決定は災いをもたらし得る。

この件に関して悪いのはAmazonだけではない。米国の特許庁はそのきわめて程度の低い基準に関して咎められるべきだし、また米国の裁判所もこれを是認したことについて責任を負うべきだ。

私たちは法廷がこの特許を法的に無効であると評決することを望んでいるが、彼らがそうするかどうかは、込み入った申し立てとわかりにくい専門用語をどう解釈するかにかかっている。

しかし私たちは法廷が電子商取引の自由に判決を下すのをただ受け身で待っている必要はない。今すぐできることがあるのだ。



Amazonとの取り引きを拒否すればよい。彼らがこの特許を使って他のWebサイトを脅かしたり制限したりするのをやめると約束するまで、Amazonからの購入をいっさい取りやめていただきたい。

Amazonで扱われている書籍の著者ならば、AmazonのWebサイト上のあなたの書籍の「著者のひとこと」欄にこの記事の先頭の題目を入れることで、この運動に対してもっと効果的に協力できる。その際にはamazon@gnu.orgにメールを送り、その後どうなったかを教えていただきたい。

●

Jason Fletcherの反論とそれへの返答（1999/12/19）

Jason Fletcher：「私はStallmanに賛成しない。まず、彼はこの件について誤った説明をしているようだ。Amazonはクッキーの特許を取得していない。クッキーのアイデアについての特許も取得していない。彼らはワンクリックショッピングのアイデアについての特許権を得たのだ」

Stallman：これは本質的には正しい。Amazonはクッキーの特許を取得してはいない。その特許はワンクリックショッピング（これは通常クッキーを使って実装される）に必要な技術に関するものだ。

私が糾弾しているのは、まさにこのワンクリックショッピングの利点を独占しようとする企てについてである。

プロフィール

Richard Stallmanは Free Software Foundation（FSF）の創始者であり、GNU General Public License（GPL）の著者であり、gccやEmacsといった重要なソフトウェアのオリジナルの開発者である。

思いがけない財産

Text : Eric S. Raymond

http://linuxtoday.com/stories/13512.html（原文）

http://www.ascii.co.jp/linux/linuxtoday/article/article366112-000.html（邦訳全文）

数時間前、私は自分が現在（少なくとも理論上は）途方もなく金持ちであることを知った。

私が自分のマシンでプログラミングに取り組んでいたら、VA Linux Systemsの株式公開の成功を知らせるメールが来た。私はこのとき最新の小さなプロジェクトのための作業をしていた。自分の設計したScriptable Network Graphics（SNG）という特殊用途の言語のコンパイラのプロジェクトだ。

「おめでとうございます、だって？ これはおもしろい」私は独り言をいった。「明日までは公開されないんだと思っていたよ」。だが私は知っているべきだったのだ。私はVAの取締役会のメンバーとして、Larry Augustinじきじきに企業の良心の役目となるべく採用されている。それはすなわち私がVAのかなりの株式を所有すると公文書に記載されているということでもあった。VAはNASDAQから実際に株式公開していた。そして私は知らないうちにおよそ4100万ドルの財産を所有していたのだ！

私がこんなことを公言するのを不思議に思う人もいるだろう。ある人が大金持ちになりそうなときに、友人や家族から受ける最初の助言はおそらく「それを秘密にしておけ」だからだ。あなたは自分がさも満足げにほくそえんでいるように思われたくはないだろうし、

慈善事業への寄付の要請や、仕事の企画や、長らく所在不明だった親友からの連絡や、単なるあつかましいたかりなどが、ひっきりなしに殺到するような目にあいたくはないだろうから。

私が「秘密にしておく」方式をとるうえで問題となるのは、私がこの儲けをきわめて公的な方法で得たことだ。あなたがすでにマスコミでよく知られていて、その会社の株価に関与し、社の方針を決定する権限を持つ人間として登録されており、その株が初日収益の記録を破るたびに友人や記者たちから熱狂的なメールが届くようになり始めたら、遠慮がちな態度はとれそうもない。

また、隠しごとはよくないことである。私はコミュニティの要求に応える。私は現在裕福だが、これは実業界に活気を与える助けをし、多くのハッカーたちから尊敬と信頼を得たそのコミュニティの代表として、オープンソースの考え方を世に広めた自分自身の努力によるものだ。Larryはその尊敬はVAの15万株を分与するのに相当すると考えてくれたのだ。私を儲けさせてくれたハッカーたちにフェアに対応するためには、私がこの結果を公に表明し、それが私の人生にどう影響しそうか、また、その金をどう使うつもりかという疑問に公式に答えることが必要だ（いやはや、「Linuxでどうやって儲けるか」が大問題だった頃を覚えているだろうか？）。

私はまず「私は6月までは何もしません」と答えるだろう。というのは、私はVAの役員会のメンバーであり、証券取引委員会の規定による株式に関する6カ月間のロックアウト期間にあるからだ（これは、インチキ株を公開したあとすぐに換金して、株価が暴落する前にアルゼンチンへ逃亡するのを防止するためのものだ）。だから、私が現時点で裕福であるというのは厳密には真実とはいえない。VAや米国の経済が破綻しなければ、私は6カ月のうちに裕福になるだろう。私はVAについては信頼している。アメリカの経済については何とも言えないが。:-)

実際その経済が破綻しなかったとして、6カ月の間にその財産は私の人生にどれほどの影響を与えるのだろう？ 正直に言って、たいした影響はないのではないかと思う。私はこの15年間、金儲けのためにオープンソースに関わってきたわけではない。私はすでに、自分にとって価値のある仕事をして、ほとんど自分の望むとおりのやり方で生活している。お金があることによる個人的な最大の違いは、今後はまったくお金の心配をせずに自分の好きなことを一生続けられるだろうということだ。

よろしい、では細かいことをあげてみよう。おそらく私は、やっと携帯電話を手に入れることになるだろう。そして高速ケーブルネットワークに加入して強烈な速度でWebサーフィンできるようにする。それから新しいフルート。そしておそらく、戦術射撃競技用のよくチューニングされた45口径自動拳銃も。いや実際、私はあまり多くのものを望んだり必要としたりはしていない。その点では私はいわば仏教徒のようなものだ。私は自分の物質的なものへの執着を最低限に抑えることを好む（私の家族は、そのせいで私へのクリスマスプレゼントを買うのにひどく苦労するのだと不満をいう）。

だが私は、自分の財産をすべて喜捨することで、それへの執着を最小化しようとしているのではない。だからきわめて意義深い寄付のための福音伝道者の方々は落ち着いて、私に数100万ドルの寄付をせきたてることはお忘れいただきたい。私は説得されやすいカモになるつもりはなく、しつこい要求はすべてきっぱり拒否するつもりだ。

しかしつまらぬ話はもう十分だろう。私は仕事に戻るとしよう。私はSNGコンパイラの作業をほとんど終えた。次は、pngcheckのコードを再解析してSNG構文を生成するレポート形式のオプションを付けなければいけない……。

プロフィール

Eric S. Raymondは、オープンソースの伝導師として、最も有名な1人。彼の論文『伽藍とバザール』はオープンソース開発モデルの聖書と言っても過言ではない。

読者の声

俺にもいわせろ！

みなさんこんにちは。突然ですが、担当はキーボードを新調しました。最近の日本語キーボードはスペースバーが短いので躊躇していたのですが、ローマ字入力ではキートップのカナ文字表記が不要だと気づき、英語キーボードを選びました。そうです、担当はタッチタイピングができないのです。

それでは、今月のお便りをご紹介しましょう。

隠喩としてのコンピュータ

「隠喩としてのコンピュータ」を読んで、コンピュータの歴史というか展開は、ある種の文化論であり、生物の進化論でもあるという感じを再確認しています。

（鹿児島県　秦 浩起さん）

「隠喩としてのコンピュータ」で豊福氏の言うこともももっともではありますが、MacのGUIが使いやすいのも事実であります。GNOME、KDEも使いやすくはなっていますが、Microsoft Windowsのマネという感じもします。

今年Appleは「Aqua」を発表し、そのベースとなるカーネルはUNIXであります。これから学ぶことも多々あると思います（Linuxがマニア向けを目指すのであれば無用ですけど）。

（宮城県　後藤 淳さん）

コンピュータというと、とても無機質なイメージを持たれますが、それを作ってきたのも、そしてまた使ってきたのも生物である人間です。だからこそ生物の進化にも似た発展の過程を経てきたのかもしれません。コンピュータの世界でも、キーマンと呼ばれる人たちはとても個性の強い方が多いようですしね。また、Appleはパーソナルなコンピュータという部分に重点を置くことによって、独特なコンピュータを世に送り出し、熱狂的なファンに支持されてきました。この点は、Linuxを使うわれわれもよく研究するべきでしょう。

3月号の特集1へのおハガキ

「ホームサーバを立てよう！」はとてもよい記事だと思います。これからも“Windows、Mac→Linuxへ”という読者の味方になってください。

（東京都　山梨智弘さん）

「ホームサーバを立てよう！」かなり参考になりました。Windows 98とLinuxだけのLANなのでMacもほしくなりました。

（兵庫県　西庄一紘さん）

Linuxだけでなく、WindowsやMacintoshにも優れた点がたくさんあります。Linuxはこれらの OS と組み合わせて使うとより便利になりますね。ただ、それぞれ作法が違うので、管理をするのが大変になるのが悩ましいところです。担当も、いつかはMacintoshを買いたいと思っているのですが、現状ではとてもそこまで手が回りません。

3月号ハードウェア特集へのおハガキ

「最新CPU＆マザーボード選び」は参考になりました。そろそろ新しいマシンを組もうかと思っているところなので、現状のチップセット動向なんかがよくわかりましたし。あとは、店巡りして財布と相談（笑）。

（大阪府　向井利宏さん）

ブレーカが落ちる事件があり、稼働していたPCのマザーボードがおかしくなったようです。(;　;)

特集のマザーボード選びは参考になりました。

（静岡県　aitanさん）

今後も、Linux magazineらしいハードウェア特集をお届けできるよう頑張ります。担当も、あの特集で物欲が刺激され、マシンのパワーアップを計画中です。

分散コンピューティング

「Linux日記」の榊さんへ。私はRC5クラッキングにCPUパワーを注いでいます。SUG（Sega Users Group）に参加し、会社の友人のパソコンにもクライアントソフトをインストールして、仕事で使っていないときはもっぱら解

読作業をさせています（もちろんインストールの許可は取っています）。
早く地球外生命体が発見できることを祈っています。

（福岡県　山本栄治さん）

Distributed netは、インターネットを介して世界中のコンピュータをつなぎ、64ビットのRC5暗号解読を行っています。実は編集部にもこのプロジェクトにエントリーしている者が2名います。1人はJLUG（Japan Linux Users Group）に参加しています。そしてもう1人はこのコーナーの担当です（所属チームは秘密です）。

これに対し、編集長は榊さんと同じ、SETI@homeというプロジェクトに加わっています。いつのまにか、編集部中のマシンにSETIクライアントを仕込んでしまいました。

BSDユーザーにもLinux magazine

私はFreeBSDユーザーなのですが、特集がとてもタイムリーだったのと、異文化に触れてみようという思いで初めて買ってみました。とは言っても、内容的にはLinuxに限らず、PC UNIX一般に応用できることばかりなので大変面白かったです。また、こういう雑誌が月刊で多数発行されているLinuxが多少うらやましくも感じました。

（兵庫県　鈴木晋介さん）

BSD系に特化した雑誌は、弊社のBSD magazineくらいしか見あたりませんね。ただ、鈴木さんのおっしゃるとおり、LinuxとBSDに関わらず、知識が共有できる部分もたくさんありますので、BSD magazineともどもLinux magazineもよろしくお願いいたします。

賢く使う旧マシン

企業のリース落ちと思われるPentium 166MHzマシンをほとんどタダ同然で入手。ただいま我が家でリッパにSOHOサーバとして動いております。これぞLinuxの醍醐味……（なのか？）

（愛知県　ぶらふさん）

今月の特集で紹介したように、PentiumマシンはLinuxと組み合わせてまだまだ活用できます。ハードウェアを強化すればGNOMEだってOK。サーバなど用途を限定して負担を軽減すればそのままでも十分現役。工夫次第でPCの活躍場所を拡大できるのもLinuxの醍醐味です。

世界の国からこんにちは

毎月「Free Application Showcase」を楽しみにしています。海外物が多いLinux用フリーソフトにはこういった記事が必要だと常日ごろ思っていました。また、自分で見つけて使い続けていたソフトがここに取り上げられるとホッとします。

（愛知県　藤田善敏さん）

海外で作られるソフトウェアの中には、あっと驚くようなアイデアが盛り込まれているものが数多くありますね。担当は、いろいろなソフトウェアのCHANGE.LOG（改変履歴）を読むのが密かな楽しみです。

カーネルのダイエット

インターネット上で使うサーバ専用にカーネルを作ると、ファイルのサイズがもとの60%ですむ（Red Hat Linuxの場合）。もっと減らせるかも。このへんの記事待ってます。

（神奈川県　kechiさん）

多くのディストリビューションでは、標準でインストールされるカーネルはさまざまな機能を持たせてあるため、そのサイズも大きくなっています。昔とは違って、デバイスのドライバをモジュールにすることで、必要のないモジュールは組み込まれないようになっているのですが、まだまだユーザーがチューニングする余地は残っています。kechiさんのおっしゃるような用途だと、シリアルポートやパラレルポートのサポートすら外すことができるで、かなり小さなカーネルにできますね。

Macintosh de Linux外伝

MacOSのPCエミュレータの上でLinuxを動かし、そこでApacheを走らせています。約10秒と起動が速いので、LASER5 Linuxが常にMacOSで動いています（笑）。

（奈良県　細水康平さん）

斬新な使い方です。Macintosh関連の編集担当も脱帽していました。

やはり気になりますか

私たちはLinuxを「りなっくす」と言うのですが、何年も前から関わっているバリバリのユーザーの方は「りにっくす」と言うんですよね。アスキーさんでは、どう言われてます？　Linux…。

（福岡県　井上由美子さん）

人によってさまざまです。「りなっくす」、「りぬくす」など…。5年ほど前には「らいなっくす」というのも流行りました。決まった読み方はないというのが正解のようです。担当は話をする相手がなんて発音するかに合わせるようにしています。ちなみに、本誌は「リナックスマガジン」と呼んでください。