ディストリビューション新バージョン なぜかこの時期に続々登場！

3月中旬にXFree86 4.0がリリースされた。まだ安定度は不十分のようだが待望のメジャーバージョンアップである。そして、開発版カーネルのバージョン番号が2.3.51から2.3.99（執筆時点では2.3.99-pre3が最新リリース）に一気に上がったところからも、カーネル2.4のリリースが間近に迫っていることがわかる。普通に考えると、ディストリビューターはこれらのリリースを待って、最新ディストリビューションを発表しそうなところだ。

ところが、下の表にあるように、なぜか今月（4月）は毎週のようにどこかのLinuxディストリビューションパッケージが発売される。しかし、これらのディストリビューションは、当然のことながらどれも2.2.13または2.2.14といった2.2系のカーネルのままだ。また、XFree86 4.0を別CDに収録しているものもあるが、標準でインストールされるのはXFree86 3.3.6なのである。

もうすぐ、カーネル2.4が出ると言われているにもかかわらず、このタイミングでバージョンアップされるのはなぜなのだろう。しばらくはどちらも安定しないと見込んでいるのか？

この答えは半年後ぐらいには明らかになるだろうが、それにしても最近のLinuxディストリビューションのバージョンアップの間隔は短い。メジャーどころのディストリビューションは半年に1回のペースでバージョンアップが行われている。この調子では個人ユーザーは最新版のパッケージについてこられないだろう。そう思っていたら、TurboLinuxは初回3万本については9800円というキャンペーン価格を打ち出すし、レッドハットはスタンダード版を3980円と、かなり安い価格で販売する。サポートも、以前はメールだけだったところが多かったのに電話でも受け付けるように充実してきたようだ。

ますます競争が激しくなってきたLinuxディストリビューション業界、当人たちは大変だろうがユーザーにとって見れば常に最新バージョンを安価で入手できるのはありがたいことだ。

ディストリビューション	発売日
Kondara MNU/Linux 1.1	4月1日
TurboLinux Workstation日本語版6.0	4月7日
Vine Linux 2.0 CR Official製品版	4月14日
Red Hat Linux 6.2J	4月21日

Hardware　発売日　2000年3月27日

コンピュータ切替機「PShare Multi4」

URL　http://www.plathome.co.jp/

発売　ぷらっとホーム株式会社
TEL　03-3251-2600
価格　9万9800円

ぷらっとホームは、1組のキーボード、マウス、ディスプレイで4台までのPC/AT互換機、Sunワークステーション、Macintoshを切り替えて利用できるPShare Multi4の販売を開始した。価格は、9万9800円、各機種用の接続ケーブルセットがそれぞれ5000円、フットスイッチが3000円。同社は、年間の販売目標台数を1万台としている。

キーボードとマウスのポートごとに、合計8個のエミュレーション用MPUを備えるため、機種やOSを選ばないうえに、安価な切替機で発生しやすい切り替え時のハングアップを起こすことがないという。

マシンのキーボードポートから電源供給を受けているため、通常は外部電源を必要としない。複数のキーボードを用いたり、カスケード接続時など消費電力が大きい場合には、専用ACアダプタを利用可能。

マシンの切り替えは本体前面のボタンだけでなく、キーボードのホットキー（Ctrl＋Alt＋Shift、または「Scroll Lock」「Num Lock」を2回続けて押す）やオプションのフットスイッチでも行うことが可能で、キーボードから手を離さずにマシンを切り替えられる。

複数のPShare Multi4をカスケード接続することで、最大16台までのマシンを1組のキーボードとマウスで利用できる。またすでに発売されているPShare 4／PShare 8ともカスケード接続が可能なので、必要に応じた拡張が行える。

Software　発売日　2000年3月8日

日本語全文検索ソフトウェア「MitakeSearch」

URL　http://digital.compaq.co.jp/mitake/

発売　コンパックコンピュータ株式会社
TEL　0120-018589
価格　9万円（Linuxモデル）～

コンパックは、インターネット、イントラネッ対応の日本語全文検索ソフトウェアの新バージョン、MitakeSearch V2.0の販売を開始した。最大で10万の検索対象ファイルを扱えるLinuxモデルの価格は、9万円。

Oracleなどのデータベースソフトとのゲートウェイ機能を持ち、リレーショナルデータベース内のデータを検索対象とすることができる。

類語辞書検索機能を持ち、たとえば「PC」というキーワードを入力すると、「パーソナルコンピュータ」「パソコン」などのキーワードもあわせて検索できる。類語辞書は、使用環境に合わせてカスタマイズが可能。

Webインデクサ（インデックス作成機能）が改良され、データの収集が中断された場合は、次回にその地点から収集を再開できるようになった。Tcl/Tkで作成されたGUI検索クライアントが提供されている。このクライアントは、直接 MitakeSearch サーバと通信を行うので、Webブラウザ／Webサーバを持たない環境でも利用可能。

PDF、Microsoft Word、Excel、PowerPointなどさまざまな形式の文書を検索の対象としたり、MP3形式の音声ファイルのタイトル検索なども行える。

Software　発売日　2000年3月27日

インターネット・ウイルス対策ソフト「InterScan VirusWall for Linux」

URL　http://www.trendmicro.co.jp/

発売　トレンドマイクロ株式会社
TEL　03-5334-3650
価格　36万円（30ユーザー）～

トレンドマイクロは、インターネット・ゲートウェイウイルス対策製品のInterScan VirusWall for Linux（以下InterScan）のRed Hat Linux 6.1対応版の販売を3月27日から開始した。価格は、1年間のサポートサービスを含んで、30ユーザーライセンスで36万円から。

InterScanは、各種コンピュータウイルスの媒体となる可能性のあるSMTP／HTTP／FTPプロトコルをサーバ上で監視し、ウイルスの侵入／流出の両方を防ぐためのソフト。

同社が開発したActivePS方式によって、リアルタイムで電子メールの添付ファイルやHTTP、FTPで転送されるファイルの検索を行い、ウイルスやトロイの木馬型プログラムを発見できる。圧縮されたファイルについても、ZIP、LHA、ARJといった主要な圧縮形式に対応しており、圧縮ファイル内のウイルスを検知できる。そのほか電子メールにバイナリファイルを添付する際に用いられる、uuencode、MIMEのBase64、BinHexにも同様に対応している。

ウイルスを発見した場合の処理は、システム管理者への報告だけでなく、送信者と受信者にメールで警告を通知するなど、自由に設定できる。

ウイルスのパターンファイルは、ボタン1つで最新のダウンロードが行える。また、予約アップデート機能を用いて、定期的に自動ダウンロードできる。

Software　　発売日　2000年3月24日

Webアプリケーションサーバ「dbMAGIC 8.2エンタープライズサーバ」

URL　http://www.magic-sw.co.jp/

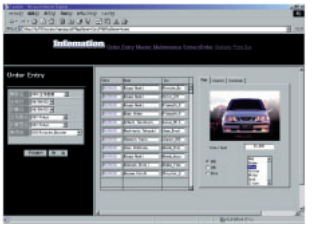
発売　マジックソフトウェアジャパン株式会社
TEL　03-5365-1600
価格　27万円（1インスタンス版）～

マジックソフトウェアは、Webアプリケーションサーバ、dbMAGIC 8.2エンタープライズサーバのLinux版の出荷を3月24日から開始した。価格は、27万円（1インスタンス版）から。

サポートするOSは、Red Hat Linux 6.1日本語版とTurboLinux Server 6.0。データベースは、Oracle 8、Oracle 8i、C-ISAM7に対応している。

Webアプリケーションや、3階層システムのアプリケーションをLinux上で実行するための製品。これらのアプリケーションの開発には、dbMAGIC 8.2エンタープライズ開発（Windows版のみ）が必要になる。

ミドルウェアであるMRB（Magicリクエストブローカー）によって、負荷分散・フェイルオーバーを実現できる。また、dbMAGICエンジンにより、ほかのサーバプラットフォーム用に作成されたプログラムがそのまま利用できる。



Hardware　　発売日　2000年2月27日

単機能サーバ「LogiPROミニサーバ」

URL　http://www.ydc.co.jp/net/mini-server/index.html

発売　ロジテック株式会社
TEL　03-5600-1420
価格　19万8000円～

ロジテックは、単機能サーバの「LogiProミニサーバ」を発売した。Web、Mail、DNS、簡易ファイヤウォールサーバなど基本的な機能を持ったTサーバ、グループウェア機能を持ったBサーバ、ネットワークの監視機能を持ったNサーバ、ワンタイムパスワードを用いた安全性の高い認証サービス機能を持ったUサーバの4モデルがある。価格は、Tサーバが19万8000円から、Bサーバが35万円から、Nサーバが59万8000円から、Uサーバが59万8000円からとなっている。

170（W）×203（H）×248（D）mmと場所を取らないサイズで、重量は4.7kg。100ユーザー以下の環境用の標準タイプは、CPUにAMD 5x86（133MHz）を採用し、32Mバイトのメモリ（最大128Mバイト）、10Gバイトのハードディスク、10BASE-TのLAN×2を装備する。OSはRed Hat Linux 5.2をベースにカスタマイズを加えたもので、カーネルは2.0.36を用いている。

故障時には、送り返して修理または、交換という家電製品的なサポート体制を採用している。また同社では代替機の貸し出しサービス（有償）も行っている。

Software　　発売日　2000年3月1日

アプリケーション開発／サーバ運用ツール「Unify VISION 5.0 for Linux」

URL　http://www.unify.com/jp/

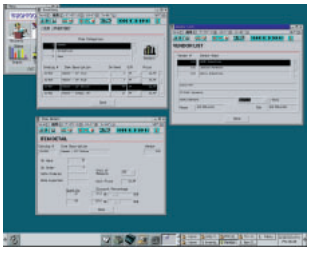
発売　ユニファイジャパン株式会社
TEL　03-5814-3102
価格　88万円～

Webアプリケーション開発／サーバ運用ツールである、Unify VISION 5.0 for Linuxがユニファイジャパンから発売された。

価格は、Unify VISION AppBuilder 5.0 for Linuxが（ビジュアル開発ツール）が88万円、Unify VISION AppServer（Webアプリケーションサーバ）が162万550000円（25同時ユーザー）から。

全社レベルのシステム構築に必要な、アプリケーションの開発、統合、運用、管理機能を統合している。

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TurboLinux 6.0。対応環境は、データベースがOracle、Sybase、Microsoft SQL Serverなど、WebサーバがApache、Netscape Enterprise Server、Netscape Fast Track。クライアントとしては、JDK 1.1に完全対応したWebブラウザが必要。



Software　　発売日　2000年4月1日

Webビジター解析ソフト「WebTrends Enterprise Reporting Server」

URL　http://www.asgent.co.jp/

発売　株式会社アズジェント
TEL　03-5643-2561
価格　49万8000円

アズジェントは、Webサーバのログ（Webビジター）解析ツールWebTrends Enterprise Reporting ServerのLinux版を4月1日から出荷する。対応するディストリビューションは、Red Hat Linux 5.1／5.2／6.0／6.1。64Mバイト以上のメモリと200Mバイト以上のディスク空き容量が必要。

簡単な操作で、さまざまな分類方法のレポートを作成できる。レポートのフォーマットは、用途に応じて、HTML、Microsoft Word、Microsoft Excel、テキストを選択できる。また、定期的にレポートを作成し、電子メール、ファイルへ保存、FTPでの配布を行うスケジューラ機能を持つ。

同社では、WebTrendsシリーズ全体で、年間800本の販売を見込んでいる。

News Headline from 日刊アスキーLinux

www.Linux24.com

米Novell、Linux関連製品の出荷を開始

2000年3月15日

米Novellは、「NDS eDirectory for Linux」、「NDS Corporate Edition for Linux」の出荷を開始した。同社は今後、Linuxに注力してゆくことを発表している。

NDS eDirectory、NDS Corporate Editionは、LDAPに準拠したディレクトリサービスを提供するソフトウェアだ。

ディレクトリサービスは、ネットワーク上の各種リソース（コンピュータ名やプリンタ、ユーザー情報など）を一元的に管理するサービス。LDAP（Lightweight Directory Access Protocol）は、ディレクトリサービスのプロトコルのひとつ。

NDS Corporate Editionは、NDS eDirectoryの基本的な機能のほかに、ユーザーやグループの管理機能が加えられている。

対応OSは、Windows NT、Solaris、Netwareなどで、今回新たにLinuxに対応したことになる。

同社はNDS eDirectoryを独立系ソフトウェアベンダーや開発者に対し、無償で100ユーザーライセンスを提供したり、競合するLDAP製品のユーザーに対して、使用中の製品と同じユーザー数分のライセンスを、アップグレードプロテクションのメンテナンス料のみで提供するといったサービスを行うという（アップグレードプロテクションは、一定期間、ソフトウェアの最新版を提供するサービス。メンテナンス料は、1年間の場合1ユーザライセンスあたり1ドル、2年間の場合は2ドルとなっている）。

価格は、NDS eDirectoryが100ユーザーライセンスで200ドルから、NDS Corporate Editionが、5ユーザーライセンスで130ドルから。

米Novell（http://www.novell.com/）

Loki、オープンソースの3Dオーディオライブラリ「OpenAL」を発表

2000年3月10日

Linuxに商用ゲームを移植していることで知られる米Loki Entertainment Software（Loki）は3月8日、クロスプラットフォームな3Dオーディオライブラリ「OpenAL」を発表した。

OpenALを使うことにより、新しいサウンドカードが備える3Dオーディオ機能を、オープンなAPIで扱うことが可能になる。APIの仕様や実装は、OpenALのWebサイトからダウンロードできる。

ライセンスはGNU LGPL（Lesser General Public License）。これはバイナリと共にソースコードを配布する義務がある点はGPLと同じだが、フリーではないプログラムがLGPLなライブラリをリンクすることを明示的に許可している点が異なり、GPLより制限のゆるいライセンスである。

OpenALはOpenGLのようなクロスプラットフォームなAPIを目指しており、LinuxのほかWindowsとMacintoshにも対応する。

Loki Entertainment SoftwareとCreative Labsが、スポンサーとしてOpenALに参加する。Creative Labsは、OpenALに対応したLinux用ドライバのリリースを予定しているという。

OpenALホームページ（http://www.openal.org/）

デバイス制御用アプリケーション作成ツール「Data Views」がLinuxに対応

2000年3月10日

米Data Viewsは、同社の「ヒューマン・マシン・インターフェイス」開発ツール「Data Views」でLinuxをサポートすることを発表した。

Data Viewsを使って作られたアプリケーションは、たとえば工場やプラントにおいて、タンクなどを制御する場合に使用される。

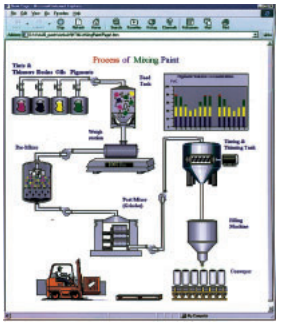
Data Viewsで作成されたアプリケーションは、タンクの中身のようすを画像で表示。タンクに原材料が注入されると、画面上のタンクにも同様に原材料が注入される画像が表示される。仕組みとしては、タンクからの数値をアプリケーションが受け取り、画像を変化させるというもの。

逆に画面（ユーザー）上からタンクを制御することも可能だ。つまり、画面上のバルブをつまんで閉めると、実際のタンクのバルブも閉まる、といったことが実現できる。X端末やWebを使ったネットワークにおける使用も可能。このような人間と機械のインタラクティブなインターフェイスを、同社では「ヒューマン・マシン・インターフェイス」と呼んでいる。

Data Viewsは、こうしたアプリケーションを作成するためのツールで、今まではUNIXやWindowsをサポートしていたが、今回Linuxもサポートするという。対応するディストリビューションは、Red Hat Linux 5.2／6.1で、2000年春にはTurboLinuxもサポートする予定。

価格は 開発ライセンスが165万円、配布用ランタイムライセンスが19万円となっている。

米Data Views（http://www.dvcorp.co.jp/）



統合デスクトップ環境「Helix GNOME desktop」ベータ版がリリース

2000年3月9日

米Helix Codeは、統合デスクトップ環境である「Helix GNOME desktop（以下

HGD)」のベータ版をリリースした。

米Helix Codeは、GNOME Projectのリーダーであるmiguel de Icaza氏が、GNOMEの開発とサポートをするために設立した会社。

HGDは、80以上のGNOME対応のソフトウェアを含んでおり、GNOME環境を管理するために、以下のような機能強化がなされている。(1) Helix GNOME Installer。CD-ROMドライブや、ローカルマシン上のハードディスク、インターネット経由でのインストールをサポートし、GNOME環境を従来よりも簡単にインストールできる。(2) Helix Update。最新のGNOME環境を保つための自動アップデート機能を持つ。

これらの機能を含め、同社の開発したソフトウェアは「GPL (GNU GENERAL PUBLIC LICENSE Version 2)」のもとに公開されるという。

現在のベータ版でサポートされているLinuxディストリビューションは、Red Hat Linux 6.x、Linux Mandrake 6.1／7.0、SuSE 6.3、Caldera OpenLinux 2.3、LinuxPPC 2000である。

2000年4月に予定されているファイナルバージョンでは、FreeBSDなどいくつかのUNIX系OSもサポートされるという。同社ではこのほかに、Excel、Lotus-123などとファイル互換性を持つ、表計算ソフトウェアのGnumericや、メールクライアントやスケジュール管理機能を統合したグループウェアEvolutionなどのオフィススイートソフトウェアの開発を進めている。

米Helix Code
(http://www.helixcode.com/)

日本オラクル、Linuxキャンペーン・コンテストを開始

2000年3月9日

日本オラクルは、Linux上で稼動するOracleを使ったポータルサイトのデザインや、アプリケーションの独創性を競うコンテストを実施する。これは下記の「Linuxキャンペーン製品」の、3月10日からの店頭販売を記念してのもの(キャンペーンそのものの終了日は5月31日)。

応募作品の選定は、日本オラクルと同コンテストのスポンサー企業が行なう。選定にあたっては、ポータルサイトに関しては見た目などのデザイン性、アプリケーションに関してはエンジニアリング的な独創性が重視される。

応募受付期間は2000年6月1日から7月末日まで。優秀作品は2000年9月中旬に発表される。

このほかの詳細は、日本オラクルのWebページ上で発表されている。

応募作品種別と条件は、Linuxポータルサイト・デザインが「Oracle WebDB for Linux」「Oracle8i Workgroup Server for Linux」を使用し、Linux上で動作すること。Linuxアプリケーションが Linux上で動作する「Oracle8i Workgroup Server for Linux」を利用したアプリケーションであること。

Linuxキャンペーンの対象製品は、以下の3種類。(1) Oracle8i開発者キットは、「Oracle8i Workgroup Server for Linux R8.1.5 (開発者ライセンス)」、「Oracle8i Client for Windows NT and Windows 95／98 R8.1.5 (開発者ライセンス)」、「Oracle WebDB for Linux R2.1 90日間トライアル版」の3点がセットになった製品で、3万9000円。(2) Oracle Linuxポータルキットは、「Oracle WebDB for Linux R2.1 (5同時ユーザー)」、「Oracle8i Workgroup Server for Linux R8.1.5 (5同時ユーザー／10クライアント)」の2点がセットになった製品で、19万6000円。(3) Oracle WebDB for Linux R2.1 (5同時ユーザー) は、キャンペーン価格で、9万8000円。

日本オラクル
(http://www.oracle.co.jp/)

Webブラウザ「Mozilla」プレベータ版リリース

2000年3月2日

「Mozilla Milestone 14」がリリースされた。

製品版でない「Mozilla」のバージョンナンバーはMilestoneを前置して表記される。今回の「Milestone 14 (以下M14)」の位置づけは「プレベータ」となっている。

3月2日現在、バイナリが用意されているのは、Linux-i386版、Win32版、MacOS版 (MacOS 8.5以降が必要、PPC版のみ) で、Linux-i386版のファイルサイズは約6Mバイトとなっている。今後、そのほかのプラットフォーム用のバイナリも用意される。ダウンロードは下記のFTPサイトから可能。

Linux版の動作には、glibc 2.1.x環境が必要。また、ソースコードはFTPサイトには用意されておらず、入手したい場合は開発用のCVSツリーから取得する必要がある。CVSによる取得については、www.mozilla.org 内の「source code via cvs」ページを参照のこと。

今回のリリースの変更点としては、Win32版のパフォーマンスの改善、バグフィックスなどとなっている。また、2月25日に発表されたロードマップによると、次回のM15は、いよいよベータ版としてのリリースとなる予定。

Mozilla M14 Release Notes
(http://www.mozilla.org/projects/seamonkey/release-notes/m14.html)
FTPサイト
(ftp://ftp.mozilla.org/pub/mozilla/releases/m14/)

GNU Project、「Emacs 20.6」をリリース

2000年3月2日

GNU Projectは、「Emacs 20.6」をリリースした。

Emacsは、Free Software Foundationを創設したRichard M. Stallmanが中心となって開発しているエディタ。UNIXの使用者にとっては、viエディタとともにスタンダードとなっている。Emacsのほとんどの機能は、elispと呼ばれる内蔵言語により実装されており、エディタとしてだけなく、メーラや、ニュースリーダ、開発環境などとしても機能する。「Emacsは単なるエディタではなく、1つの環境である」といわれるほど拡張性が高く、elispで書かれたアプリケーションが豊富に存在する。

今回のバージョンアップは、ARMプロセッサ上のLinux上で動作可能になった以

外は、バグフィックスのみとなっている。ソースコードのダウンロードは、下記ダウンロードサイトや、GNUアーカイブなどを提供しているRing Serverプロジェクトのミラーサイトなどから可能。Emacsは現在、画像や複数のフォントを扱うなどの新機能を加えたバージョンを開発中で、Emacs 21.xとしてリリースされる予定。

GNU Emacs
(http://www.gnu.org/software/emacs/emacs.html)

MicrosoftがWSU 2.0 日本語版 Beta 2 ダウンロードサービス開始

2000年3月1日

マイクロソフトは、WindowsとUNIXの相互運用環境を実現するためのソフトウェア「Microsoft Windows Services for UNIX 2.0 日本語版」のBeta 2（以下WSU 2.0）のダウンロードサービスを開始した。

WSU 2.0は、Windows NTや、Windows 2000にインストールして使用するもので、以下の特徴を持つ。(1) NFSクライアント／サーバを備え、UNIX系OSからWindows上のディスクをマウントしたり、WindowsからNFSのファイルを参照することが可能になる（ユーザーアカウントに関しては、UNIX系OSのユーザーを、WSU2.0でマッピングすることで処理する）。(2) Windowsにtelnetクライアントを提供（標準添付のものよりも高速で安定しているという）するほか、Windows上で動作するtelnetサーバも提供する。(3) cat、lsをはじめ、tarやsed、psといったUNIXの基本的なコマンド64個を実行できる。これにより、telnetでWindowsに接続すれば、UNIXと同じコマンドによって、Windowsをリモート管理することが可能（用意されているコマンドは、WSU2.0のWebページに掲載されている）。(4) Network Information Service（NIS）からActive Directoryへの移行ウィザードが用意されているほか、Windows 2000のドメインコントローラを、マスタNISサーバとして動作させることが可能。(5) WindowsとUNIX系OSとのパスワード同期機能（サポートされているのは、Solaris 7、HP-UX 10.2、HP-UX 11.0、Red Hat Linux 5.2）。(6) Perlスクリプトを Windows上で実行可能にする「ActiveState ActivePerl 5.6」を提供する。(7)「Microsoft 管理コンソール」により、WSU2.0の機能を統合管理可能。(8) 米Mortice Kern Systems（UNIX—Windows間の相互運用ソフトを製作している会社）のデモソフトが付属。

ダウンロード形態は、上記の特徴をすべて含んだフルインストール版（64Mバイト）と、(6)、(8) を含まない基本インストール版（19Mバイト）の2種類。

WSU 2.0のページ
(http://www.microsoft.com/japan/products/ntserver/sfu/sfu.htm)

無料のグループウェア「TrueOffice」Version-Revision 01-04がリリース

2000年3月1日

QUPA.comは3月1日、無料のグループウェア「TrueOffice Version-Revision 01-04」をリリースした。TrueOfficeは、グループスケジュールやToDoの管理のほか、行き先表示、施設予約、掲示板、お知らせ機能（ログイン時にメッセージを表示する）、WebLink（共有ブックマーク機能）を持つWebベースのグループウェアで、クライアントはWebブラウザしか必要としない。

今回の01-04は前バージョンと比べて、スケジューラやアドレス帳の使い勝手を向上させたほか、条件によっては不安定になるなどのバグが修正されている。プラットフォームは、Linux 2.x（x86版）のほか、FreeBSD、Solaris（SPARCおよびx86版）、Windows 2000／NT／98／95をサポート。Webサーバは、Apacheのほか、IIS（Internet Information Server）、Personal Web Serverなどをサポートしている。

TrueOfficeを使用したいユーザーには、同社のサイトにデモンストレーションのページが用意されている。

なお、TrueOfficeは無料で全機能を使用することができるが、TrueOfficeのインストール作業やサポート作業などで商行為を行う際には、有償となる。ソースコードも有料で提供される。

TrueOfficeの次期メジャーバージョンでは、携端情報端末や携端電話との連携を実現するとしている。

QUPA.com（http://tr.qupa.com/）

「Nautilus」を含む「GNOME」のロードマップ発表

2000年2月29日

GNOME projectは、X Window System上の統合デスクトップ環境「GNOME」のロードマップを発表した。

発表されたロードマップは、ユーザー環境と開発ライブラリについてのもので、同プロジェクトが開発中のオフィススイートソフトウェア「GNOME Office」についての計画は含まれていない。

具体的なロードマップは、(1) 4月に「gnome-core」をアップデートし、初期設定の変更や、より使いやすくなった「panel」、新しい「applet」などを含む大幅な機能追加を行う。(2) 5〜6月に夏のリリースに向けてコードフリーズ。(3) 晩夏に「Nautilus」ファイルマネージャや、新しい開発用ライブラリの追加。年末のリリースのコードフリーズ。(4) 年末にほとんどの「GNOME 2.0」開発用ライブラリをリリース。合わせてユーザー環境もリリース。また、これらのリリースのどれかに「GNOME」デフォルトウィンドウマネージャが含まれる可能性があるという。

この結果、年末には「GNOME」1.0と2.0が併存可能なランタイム環境が提供され、同じマシン上で1.0と2.0のアプリケーションの開発が可能になるという。

「Nautilus」は、米Appleの元在籍者などが設立した米Eazelの開発するファイルマネージャ。

The GNOME project
(http://www.gnome.org/)

Distribution ▶▶▶

Red Hat Linux 6.2J、4月21日発売

レッドハットは、「Red Hat Linux 6.2J」を4月21日に発売する。同製品は、米国で4月10日より発売される「Red Hat Linux 6.2」を日本語化したもので、米国版と同様に「スタンダード」、「デラックス」、「プロフェッショナル」の3種類がある。

Red Hat Linux 6.1日本語版からの主な変更点は、Pentium IIIを搭載したマシンでのソフトウェアRAID 5の性能向上、追加したフォントの自動認識、4 Gバイトまでのラージメモリのサポート、クラスタリングシステム「Beowulf」のサポートなどである。また、samba、bind、dhcpdなどのデーモンを、デフォルトでは起動しないように設定が変更されており、初心者が不用意にインストールした結果、セキュリティに問題が生じるのを防いでいる。

パッケージの内容はスタンダードが、オペレーティングシステムCD 2枚、ドキュメントCD 1枚、90日間のサポート（メール）、30日間の優先オンラインアクセス。デラックスでは、スタンダードの内容に加えて、かな漢字変換システムのATOK12 SE、Wnn6 Ver.3、日本語ワードプロセッサのdp/NOTE、ブートセレクタのSystem Commander Lite、3D CGソフトウェアのShade for Linux Preview Kitなどを含んだワークステーション向け商用アプリケーションCD、30日間の電話によるサポートが提供される。さらに優先オンラインアクセスの期間も90日間に延長される。

プロフェッショナルのみ5月中旬の発売となっており、eビジネスで使用される、Webサーバ構築などの用途に最適化されたパッケージになるという。価格は、スタンダードが3980円、デラックスが1万2800円、プロフェッショナルが予定価格2万9800円。

レッドハット（http://www.redhat.com/jp/）

SuSE Linux 6.4、リリース

ドイツのSuSE Linux社は、「SuSE Linux」の最新版6.4を発表した。4月3日よりドイツで、2週間後に米国で発売する。価格は、49.95USドル。CD-ROM 6枚組または、DVD-ROMで1500以上のプログラムを収録している。

GUIインストーラであるYaST2を改良し、プリンタやサウンドカードなどのハードウェアの自動認識能力を向上させている。

またサーバ向けの機能として、「SuSE Proxy Suite」というファイアウォール＆プロキシツールや、トラブル時にファイルシステムの復旧を容易にするジャーナルファイルシステム「Reiser-FS」が含まれている。

また、他のディストリビューションに先がけて、XFree86 4.0を収録しており、デフォルトではインストールされないが、最新のXFree86を試すことができるようになっている。

SuSE Inc.（http://www.suse.com/）

モトローラ「High Availability Linux」を発表

米モトローラ社は、「High Availability Linux」（HA Linux）を発表した。

HA Linuxは、システム停止が許されない分野の顧客からの要望に応えるために開発されたもので、99.999%の高可用性を実現するという（1年間の停止時間は、5分以下）。

HA Linuxは、Red Hat Linuxへのアドオンパッケージとなっており、同社のCPX8000アーキテクチャと組み合わせて用いる。x86およびPowerPCアーキテクチャをサポートする。ハードウェアのホットスワップ、システムを運用しながらのバックアップ、ディスクレスでの運用、システム異常の通知機能といった機能を持つ。

現在、これらの機能をLinuxカーネルに加えるためのパッチが提供されており、同社のWebサイトからダウンロード可能。HA Linuxの出荷は、2000年の5月に予定されている。

Motorola LINUX SOLUTIONS
（http://www.motorola.com/computer/linux/）

WordPerfect Office 2000、LINUX OS 1.1とともに出荷開始

カナダのCorel社は、同社のデスクトップ用オフィススイート製品である「WordPerfect Office 2000 for Linux」を4月初旬に出荷すると発表した。

パッケージ内容は、ワードプロセッサのWordPerfect9、表計算のQuattro Pro9、プレゼンテーションソフトのCorel Presentation9、PIMソフトのCorelCENTRAL9、データベースのParadox9のオフィススイートに加え、Corel LINUX OS 1.1（以下1.1）が含まれている。

1.1は、昨年末に発売された1.0の改良版で、グラフィックスカードの自動認識能力の向上、サウンドドライバの追加、RPMパッケージのサポート、Windowsとの親和性をより向上したファイルマネージャなどが改善された。同時に、カーネル2.2.14、XFree86 3.3.6などの最新のコンポーネントが採用されている。1.1は単体での販売は行われず、1.0のユーザー向けには、同社のWebサイトからの差分のダウンロードで対応する予定。　なお日本国内でも、Corel LINUX OS 1.0の英語版がメディアビジョンから9800円で発売されている。

Corel Corporation（http://linux.corel.com/）
メディアビジョン（http://www.mvi.co.jp/）

Products

1つのシステムに複数OSをインストールできる

システムコマンダー2000

Linux専用のPCを用意しなくても、たとえばWindowsとLinuxを1台のPCにインストールしておいて、起動時に簡単に切り替えることを可能にするソフトウェアだ。複数のLinuxを切り替えることもできるし、LILOの設定ファイルを書き換えるより使いやすい。

製品名	システムコマンダー2000
価格	1万4800円（キャンペーン価格1万2800円、初回ロット2万本限定）
問い合わせ先	株式会社ソフトボート TEL 03-3256-4711 http://www.softboat.co.jp/

ソフトボートからマルチOSブートソフト「システムコマンダー2000」（以下、SC2000）が発売された。SC2000は、これ1つで、基本的なパーティション操作からブートセレクトまで、マルチブート環境構築に必要とされるあらゆる処理をこなしてくれる万能ツールである。

ブート対象となるOSは、Intelのx86互換CPUを対象とした100種類以上に対応している。Windows 9x／NT／2000をはじめ、各種DOS、Linux、＊BSD、NetWare、BeOS、Bright/Vなどが主な対象OSである。

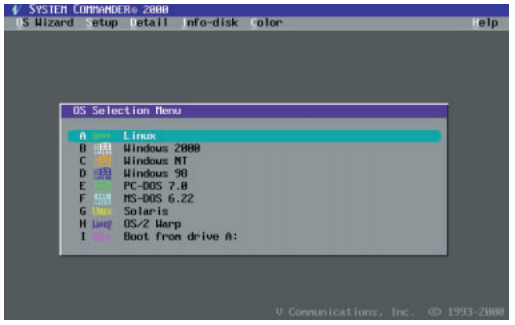
なお、SC2000をインストールしたり起動したりするには、FAT（FAT32）パーティションが必要なため、ハードディスクにDOSまたはWindows 9xがインストールされているか、ハードディスクの先頭から8Gバイト以内の領域にFAT（FAT32）のプライマリパーティションを用意しておく必要がある。Windows NT／2000のNTFSやLinuxだけの環境へは直接インストールできないので注意が必要だ。

OSウィザードで楽々インストール

SC2000をインストールすると、そのシステムのMBR（マスターブートレコード）の内容をファイルに保存し、SC2000のブートローダをMBRに書き込む。システム起動時には、BIOSが、ブートローダを読み込み、制御がSC2000に移る。この時点でパーティションテーブルを参照して、インストールされているOSを表示する。

あとは、上下のカーソルキーで起動したいOSを選び、Enterキーを押せばそのOSが起動する（**画面1**）。

新規にOSをインストールする場合には、同じ画面でALT－Oキーを押



画面1　OSセレクションメニュー
システム起動時に表示される画面。ハードディスクのパーティション内容を調べて、インストールされているOSを自動的に登録する。

画面2　OSウィザード
OSをインストールするときにはOSウィザードを使用する。OSごとにディスク管理ツールは違いものだが、SC2000でインストールするパーティションを用意しておくことで、誤って操作することが防げる。

してメニューにあるOSウィザードを実行する。**画面2**のように3種類のインストール方法が選べる。次にインストールするOSのタイプを聞かれるので、Windows、DOS、UNIXといったメニューを選ぶと、もっと細かく具体的なOS名を要求される。設定が済むとリブートし、実際のインストール作業に入る。

SC2000のOSウィザードと呼ばれる機能を使うと、インストール対象のOSを選択するだけで、最適なパーティションを設定してくれるので失敗が少ないだろう。

LinuxやNTFSのパーティションに対応

パーティション操作では、FAT32、Windows NTのNTFS、Windows 2000のNTFS 5、Linuxのext2、スワップファイルシステムに対応していて、パーティションの作成、削除、移動、コピー、サイズ変更を行うことができる（**画面3**）。

FATとFAT32のファイルシステムでは、クラスタサイズの調節やFATタイプの変更などで、ディスクの浪費領域を削減したり、ディスクアクセスの高速化を実現している。

また、NTFSからFAT32へのコンバート機能があり、圧縮されたNTFSボリュームも処理可能である。

元の環境に簡単に戻せる

SC2000の優れた特徴として、OSのアンインストールを自動化するBackStepウィザード機能がある。

BackStepウィザード（**画面4**）では、OSウィザードやパーティション操作の記録を利用して、元の環境に戻す（OSのアンインストールを行う）ことが可能だ。

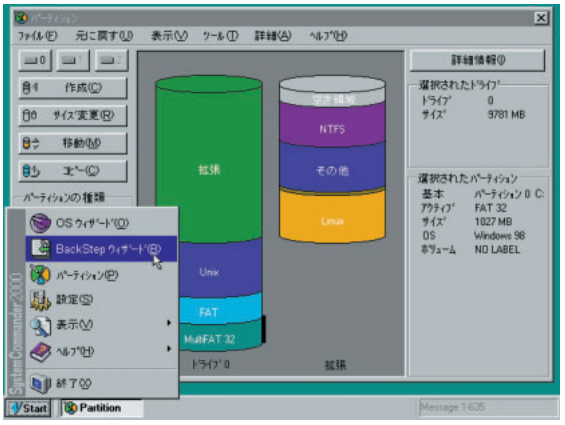
これを利用することで、ベータ版のOSをインストールして試してみることが気楽に行えるだろう。

また、MultiFAT機能によって、DOSやWindows 9x／NT／2000といったFAT対応OSを、1つのパーティションに複数インストールすることも可能だ。

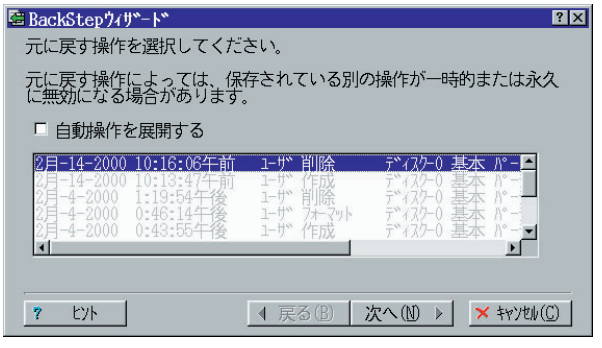
LinuxのブートローダであるLILOをMBRに書き込んでいるシステムに、複数のLinuxをインストールすると、それぞれのlilo.confをシステム構成に合わせて書き換えないと正しくブートできなくなる。

しかし、SC2000を使ったシステムでは、LILOをMBRではなくインストールしたパーティションの先頭に書き込んでおけば、SC2000がそのLinuxを見つけてくれる。

複数のLinuxディストリビューションを１つのシステムにインストールするならば、SC2000の便利さがよくわかるはずだ。



画面3　パーティション画面
ハードディスクのパーティション内容をグラフィカルに表示している。パーティションを操作するには、円筒状の部分を選択し操作すればよい。



画面4　BackStepウィザード
記録された操作情報を元に、パーティションの使用状態を復元できる。ベータ版のOSをインストールする場合に便利。

zxLinux

「ザウルス用のディストリビューションが登場したのか！」と早トチリしてはいけない。あくまで「ザウルスのOS上で」動作するのであって、直接ザウルスを制御するわけではないのだ。とはいえ、Linuxerの好奇心を刺激する注目のプロダクトであることは間違いない。

製品名	zxLinux
価格	無償配布
問い合わせ先	株式会社 アックス TEL 075-724-1966 http://www.axe-inc.co.jp/
配布先	http://www.zxlinux.org/

株式会社アックスは、シャープの携帯端末ザウルス上で動作するLinux「zxLinux」を開発した。Linuxカルチャーの広がりを示すプロダクトとして要チェックだ。

zxLinuxは、同社の提供するWebサイト（http://www.zxlinux.org/）において配布されている（ライセンスはGPL Version 2に準拠）。カーネルのバージョンは2.3.23。動作機種は、ザウルス アイクルーズ EX1（MI-EX1）およびパワーザウルス C1（MI-C1）。専用のカーネル／アプリケーション開発キットとデバッガも用意されており、上記のWebサイトから入手できる。

なお、シャープの協力のもとに開発されたzxLinuxだが、ザウルス上での動作やデータの保全に関しては保証されていない。インストールに際しては、Webサイトに掲載されている使用条件と注意事項に目を通し、慎重を期してほしい。

どうやって動くのか？

zxLinuxは、通常のLinuxカーネルに変更を加えて、ザウルス上で動作するようにしたものである。ザウルスのアーキテクチャに準拠し、可能な限りザウルスの機能をそのまま利用するようにデザインされている。このため、手書き文字認識機能によるLinuxコンソールへの入力や、ザウルスのオプションキーボードの利用が可能で、既存のザウルス環境に影響を与えることもない。

通常のザウルス用アプリケーション（MOREソフト）と同様、zxLinuxはZaurusOSのマイクロカーネルである「XTAL（クリスタル）」のプロセスとして動作する。zxLinux管理下のLinuxアプリケーションは、このzxLinuxプロセスとシステムコールをやり取りしながら動作する。システムコールは、XTALのキュー（XTAL Queue）によるプロセス間通信として実装されている（**図1**）。

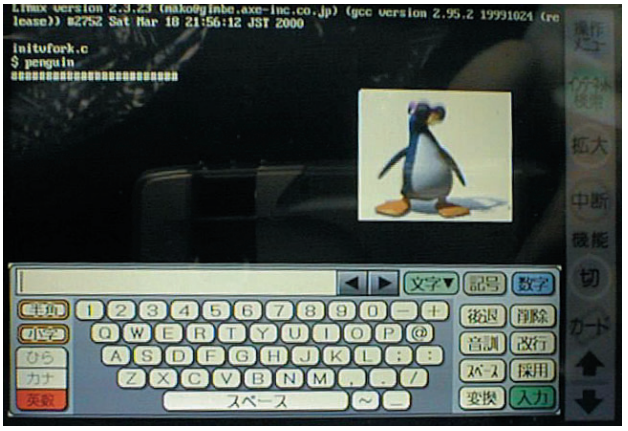


写真1　zxLinuxの動作画面
カーネルのバージョンが「2.3.23」と表示されているのが確認いただけるだろうか？

zxLinuxアプリケーションは、基本的にはLinuxアプリケーションであり、I/O操作も/dev以下のデバイスファイルに対する処理として行われる。デバイスに対する実際の処理は、zxLinuxにリンクされたデバイスドライバ部分が実行する。

入力を例に、処理の流れを図示したものが**図2**である。まず、ザウルスでの入力がZaurusOSのアプリケーション管理モジュールであるAPMに送られる。APMはこの入力を、zlboot（zxLinuxのブートストラップローダ。起動後はMOREソフトとして常駐し、イベント処理を受け持つ）にZaurusOSのイベントとして受け渡す。zxLinuxのデバイスドライバは、XTAL Queueを介してzlbootからこのイベントを取り込む。前述したように、zxLinuxアプリケーションはデバイスファイルへの入力としてこのイベントを扱う。なお現バージョンでは、「文字出力」、「キーボード入力」、「ボタン入力」の3つのI/Oがサポートされている。

ネットワークに関しては、Linuxで

はなくザウルスのネットワークプロトコルスタックを使用する。ネットワーク関連のシステムコールは、ザウルス用に変換されてから実際の呼び出しが行われ、処理はザウルスのネットワーク機能の下で実行される。

ファイルシステムは、ext2を使用している。これは、ファイルシステム自体を格納できる記憶領域がないため、ザウルスのDOSファイルシステム上にext2ファイルシステムのイメージをひとつのファイルとして格納し、それをマウントすることで実現されている。

インストールと起動

zxLinuxをインストールするには、ザウルスのコンパクトフラッシュカードが必要だ。まずはこれを用意しよう。カード内のフォーマットは、DOSのFATファイルシステムに準拠していること。

インストール自体は簡単だ。ダウンロードしたバイナリキットのアーカイブを展開する。展開されたファイルをすべてコンパクトフラッシュカードに転送する。あとはカードを装着して、ザウルスを起動すればいいだけだ。

MOREソフトウェア起動画面に表示される「ZxLinux」（実際にはブートローダプログラム）をタップすると、zxLinuxが起動する。ext2ファイルシステムのイメージ（ZLIXIMG.DAT）は、通常のLinuxと同様にマウントして、操作する。

開発環境

zxLinuxでは、Intel版Linuxでのクロス開発向けに、以下の開発キットが用意されている。

- **カーネル開発キット**
- **アプリケーション開発キット**
- **アプリケーション開発用リモートGDB**

カーネル開発

zxLinuxのカーネル開発では、Linux環境とネイティブのザウルス環境との競合（データ型の相違など）を解決する必要があり、開発環境の構築が難しい。このため、カーネル専用の開発キットが用意されている。

開発時には、ザウルスとのデータ交換にコンパクトフラッシュカードを使用する。このため、開発機にはPCMCIAカードスロットが必要になる。

アプリケーション開発

専用のアプリケーション開発キットも用意されているが、多少の制限にさえ気をつければ、x86 Linuxアプリケーションと同一のソースコードが使用できる。

「アプリケーション開発用リモートGDB」は、Intel版Linux（開発機側）とザウルスをシリアルラインで接続して使用するデバッガ。GDB本体は開発機で動作し、ザウルス上でリモートデバッガを動作させてデバッグを行う。

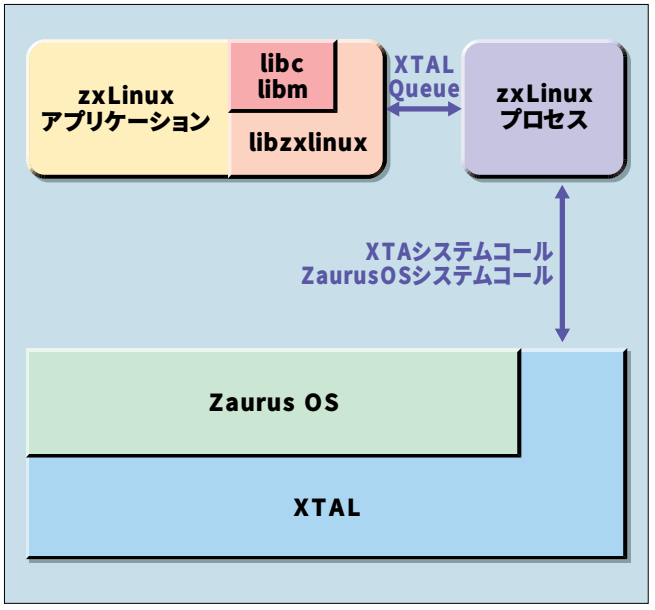


図1　zxLinuxのシステム概念図

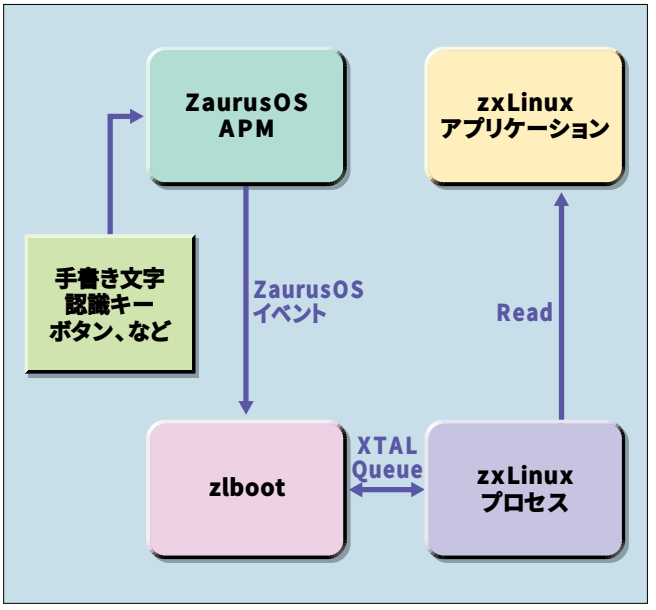


図2　入力イベント処理の流れ

Distribution

新着ディストリビューション

TurboLinux Workstation日本語版6.0

充実かつ安定した日本語ディストリビューションとして、定評のあるTurboLinuxが、カーネルなどの基本的な構成を改め、翻訳ソフトなどの豊富な商用ソフトをバンドルし、デスクトップ用途に的を絞ったTurboLinux Workstation日本語版6.0をリリースしてきた。デスクトップ用途として、どのように手を加えられたかを重点的に紹介していく。

OpenLinux eServer 2.3 日本語版

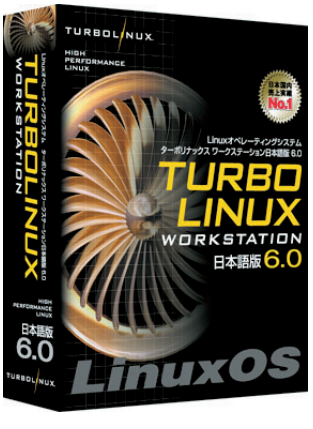
これぞオープンソースの醍醐味。OSの核となるカーネルの拡張、大容量メモリやRaw I/Oのサポートという、大規模サーバ向けのチューニングがほどこされ、OpenLinux eServer 2.3 日本語版がリリースされた。豊富なGUI管理ツールも装備し、サーバ用に独自拡張されたOpenLinux eServer 2.3 日本語版を紹介する。

TurboLinux Workstation日本語版6.0

TurboLinux Workstation日本語版6.0（以下TurboLinux 6.0）はターボリナックス ジャパンが開発、販売するLinuxディストリビューションで、使いやすい日本語デスクトップ環境を提供する。TurboLinux 6.0は、日本語入力ソフトや、翻訳ソフトといった豊富な商用ソフトがバンドルされているほか、同社が独自に手を加えた管理ツールを含め、デスクトップユーザーの使用に配慮して作られている。本体価格は1万2800円（税別）で、正規購入ユーザーは、90日間までのサポートを、Web、E-mail、電話、FAXを通じて受けられる。4月7日より発売され、初回3万本に限り、特典つきで9800円（税別）となる。

最新のパッケージ構成

TurboLinuxは、以前から最新のパッケージ構成と、安定した利用環境を提供している。今回のTurboLinux 6.0も、Xは現時点での安定版では最新バージョンのXFree86 3.3.6が、ライブラリにはデスクトップ用のTurboLinuxとしては、初のglibc2.1を採用している。



製品面　TurboLinux Workstation日本語版6.0
価格　1万2800円（税別）
初回3万本に限り9800円（税別）
問い合わせ先　ターボリナックス ジャパン株式会社
03-5766-1660
http://www.turbolinux.co.jp/

手順の少ないインストール作業

TurboLinux 6.0のインストーラは、インターフェイスに大きな変更はないものの、インストールの手順は大幅に減っている。これは、これまでインストール作業に含まれていた、X、マウス、キーボードの設定をインストール後に分けたためだ。

また、インストール中に選択するインストールタイプも、2種類のワークステーションタイプと、カテゴリー別に分類された一覧から、個々のパッケージをインストールするタイプなど、合わせて4種類しかない。これにより、ユーザーがインストールタイプの選択で迷わないようになっている。

そして、これまでのLinuxディストリビューションでは、バンドルされる商用アプリケーションが、別のCD-ROMに収録され、Linuxのインストール後に、別途商用アプリケーションのインストールが必要であったのが、TurboLinux 6.0では、OSのインストール用のCD-ROMに、商用アプリケーションも収録されているので、OSのインストールと同時に、商用アプリケーションのインストールも行える。この工夫は、インストール作業の手間を減らすという意味で評価できる。

ただ最近では、GUI操作のインストーラを採用するディストリビューションが増えてきているので、TurboLinux 6.0のテキストベースのインストーラにやや古さを感じるのは否めない。このTurboLinux 6.0がデスクトップ用途に作られたことを考えると、初心者ユーザーのためにも、GUIのインストーラをぜひ採用してもらいたいものだ。

また、ディスクパーティション設定ツールに関しても、近頃ではあまり一般的でないfdsikとcfdiskのみと、このあたりも、もっと操作性の良いツールを用意してもらいたい気もする。

セキュリティポリシーに関しても、デスクトップユーザーには、あまり用のないTELNETポートをふさぎ、遠隔

本誌付録CD-ROM Disk.1に、TurboLinux Workstation日本語版6.0 FTP版を収録しています。非商用のソフトだけが含まれますので、本文の表2に掲載されている商用ソフトは含まれていません。また、ターボリナックス ジャパン社のサポートも受けることはできません。

インストールCD	インストール用CD-ROM
ソースCD	ソース収録CD-ROM
コンパニオンCD	XFree86 4.0など未サポートのアプリケーションを収録
TurboLinuxソリューションガイドCD	Linux用商用アプリケーションのデモ版などを収録
インストールディスク（FD）	インストール用フロッピーディスク
System Commander Lite（FD）	デュアルブート環境のためのブートセレクタ
ユーザーリファレンスガイド	インストールガイド
活用ガイド	TurboLinux 6.0の基本的な使い方を説明

表1　製品に同梱される付属品リスト

ATOK12 SE	他のプラットフォームでも定番の日本語入力システム
System Commander Lite	マルチブート用のアプリケーション
RYOBI日本語TrueTypeフォント5書体	美しい日本語フォント
Wnn6 Ver3	UNIX系OSで定番の日本語入力システム
翻訳魂	GUIで操作する和訳、英訳ソフト
Vmware Express（初回3万本購入者に限定）	1台のPCに別の仮想PC環境を実現する ゲストOSはWindows95/98のみと機能制限あり

表2　バンドルされる商用アプリケーションのリスト

操作用にSSHが標準でインストールされるのにも関わらず、一般ユーザーをインストール中に作成できないなど、一貫性がとれてないことも気になる。さらに、インストール作業の最後に、起動時のセキュリティ対策のため、BIOSのパスワード設定を促してくる。BIOSのパスワードは、誤った設定をすると、2度と起動しなくなる場合があるので、このメッセージは初心者ユーザーには無用ではなかろうか。

デスクトップ用ということ

TurboLinux 6.0は、標準のデスクトップ環境にGNOMEを採用し、これまでのTurboLinuxで標準だったAfterStepと比較して、さらに使いやすくなった（**画面1**）。それでは、このほかのデスクトップ用途に工夫された箇所を順に見ていこう。

■豊富な商用ソフトウェア

表2の通り、TurboLinux 6.0には数多くの商用ソフトがバンドルされる。UNIX系OSでは定番のWnn6や、Windows、Mac OSなどでは定番のATOKといった日本語入力システムが含まれ、Linuxをメインのデスクトップ環境として使用するユーザーには、頼もしい限りだ。

また、「翻訳魂」（**画面2**）が今回から初めてバンドルされている。「翻訳魂」はオムロンソフトウェアが開発するGUIの翻訳ソフトで、翻訳したい文章をペーストすると、ボタン一発で変換してくれる優れたソフトだ。英語→日本語の変換だけでなく、日本語→英語の変換も同様の操作で行える。

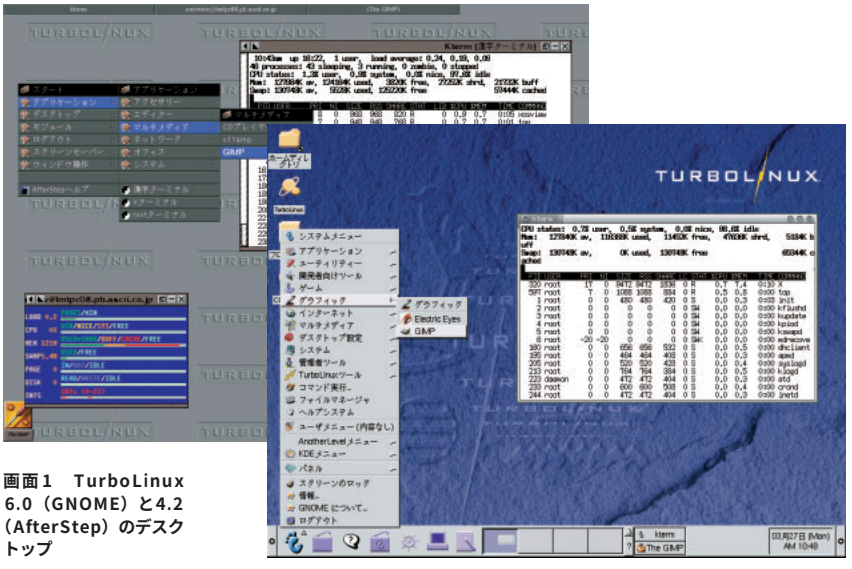
このほかにも、日本語の表示には欠かせないリョービのTureTypeフォント5書体や、マルチブート環境では定番のSystem Commander Liteも同梱している。初回出荷の3万本には、ユーザー登録者にLinux上の仮想コンピュータでWindows 95/98が動かせるVMware Express（機能制限つきのVMware）が無料で配布されるという特典もある。

■カラープリンタにも対応

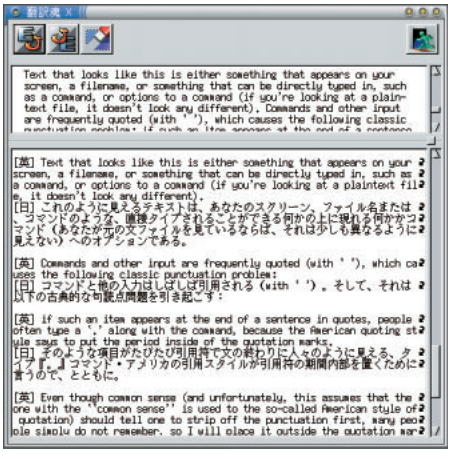
TurboLinux 6.0には、自社スタッフも開発に関わっているPIPSという汎用のカラープリンタドライバが付属している。PIPSはエプソンPM-770CとPM-800C用のプリンタドライバで、独自の設定ツールを利用し、写真画質のカラー出力を実現する。現在のLinuxは、印刷分野で他のOSに遅れをとっているので、TurboLinuxに限らず、今後のLinux普及には、このようなプリンタドライバの開発が、大きな要因になるだろう。このような意味で、今回のPIPS付属は評価に値する。

■マウント不要なCD-ROMとフロッピーディスク

自宅では多くの場合、1人で管理者兼一般ユーザーというLinuxの使い方をする人が多いことだろう。OSとしての性質をUNIXから受け継いだLinux



画面1　TurboLinux 6.0（GNOME）と4.2（AfterStep）のデスクトップ



画面2　バンドルされる「翻訳魂」
ウィンドウ上部に翻訳したい文章を貼り付け、変換のアイコンをクリックすると、ボタン一発でウィンドウ下部に、翻訳された文章が表示される。

は、通常一般ユーザーがCD-ROMなどをマウントできないようになっている。そのため、1度スーパーユーザーでマウントしてからCD-ROMを使うわけだが、ユーザーが1人の場合はあまりに使い勝手が悪い。そこで、TurboLinux 6.0では、使用頻度の高いCD-ROMとフロッピーを、一般ユーザーでもマウントできるよう設定されている。これに加え、デスクトップ上に配置された、CD-ROMアイコンをダブルクリックすることでも、CD-ROMが利用できるようになっている（**画面3**）。これによりWindowsなどを使用してきたユーザーが、違和感なくCD-ROMを使えるよう配慮されている。

■使いやすい設定ツール

TurboLinuxは、以前から独自に開発した管理ツール群が特徴である。これらはTurboToolsと呼ばれ、コンソール上で操作するツールで、Linuxに不慣れなユーザーにはとても使いやすいものであった。今回のTurboLinux 6.0からは、GNOMEから利用するTurboCentro（**画面4**）という管理ツールが追加された。TurboCentroは、TurboToolsを内部で呼び出し、マウスで操作可能なツールで、デスクトップの下段メニューからクリックひとつで起動できる。このTurboCentroでは、ネットワークやプリンタの設定のほか、ブートディスクの作成などが行える。

■ソリューションCD-ROM

TurboLinux 6.0は、今回からの新しい試みとして、ソリューションCD-ROMを添付している。ソリューションCD-ROMは、Linux用に開発された商用アプリケーションを、ベンダー情報とともに紹介するものだ。この中には体験版が収録されているものもあるので、まずWebブラウザでCD-ROMのガイドを読んだあと、各ベンダーから用意されているインストールガイドを参考にして、各アプリケーションを試して欲しい。

■コンパニオンCD-ROM

このCD-ROMには、最新版XFree86 4.0のRPMパッケージ、4Gバイトの物理メモリを使用可能にする拡張されたカーネルや、サーバ用アプリケーションなどが含まれている。ただし、このCD-ROMに収録されるパッケージの使用に関しては、テクニカルサポートの対象外になるので、使用に関しては十分な注意が必要だ。

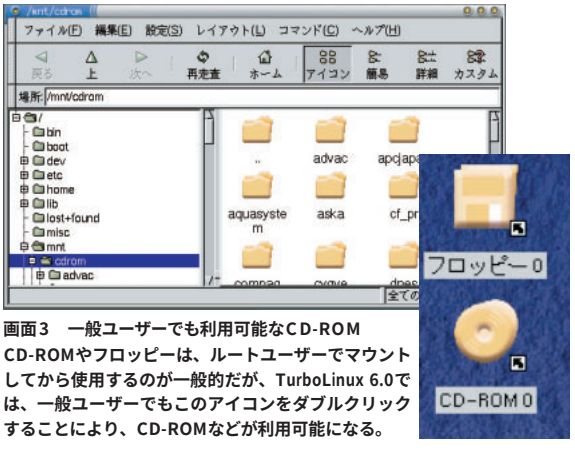
■マニュアル

TurboLinux 6.0にはマニュアル的性格の「活用ガイド」が付属する。このガイドは、Linuxの操作に必要となる基本的なコマンドの解説や、TurboLinux 6.0独自管理ツールのTurboCentro、バンドルされる各種日本語入力システムのほか、ダイヤルアップ接続での、Netscape CommunicatorやNmail4を用いたインターネットの利用法が含まれている。また、ネットワークカード導入の際などに、初心者ユーザーがつまずきがちな、モジュールの組み込み方などを含むFAQもあり、マニュアルとして重宝する。

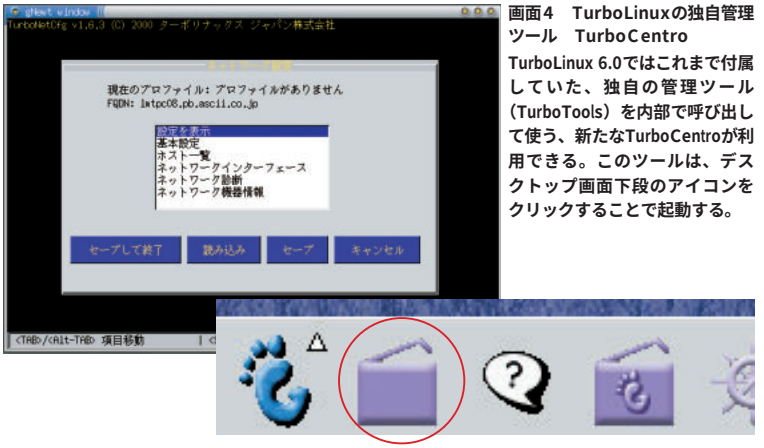
このほかにも、インストール手順を解説した「ユーザーリファレンスガイド」も付属するので、インストールに関しても安心だ。

Linux入門者にお勧め

これまで見てきたように、TurboLinux 6.0は、もともと安定した日本語環境が特徴のTurboLinuxシリーズを、デスクトップ用途として、さらに使いやすいように手を加えたものなので、分かりやすいガイドとあわせて、これからLinuxを使ってみようという初心者ユーザーにお勧めだ。そのうえLinuxユーザーに人気の高い多くの商用ソフトをバンドルしているため、内容と価格を照らし合わせると、かなりお買い得感があると言えるだろう。



画面3　一般ユーザーでも利用可能なCD-ROM
CD-ROMやフロッピーは、ルートユーザーでマウントしてから使用するのが一般的だが、TurboLinux 6.0では、一般ユーザーでもこのアイコンをダブルクリックすることにより、CD-ROMなどが利用可能になる。



画面4　TurboLinuxの独自管理ツール　TurboCentro
TurboLinux 6.0ではこれまで付属していた、独自の管理ツール（TurboTools）を内部で呼び出して使う、新たなTurboCentroが利用できる。このツールは、デスクトップ画面下段のアイコンをクリックすることで起動する。

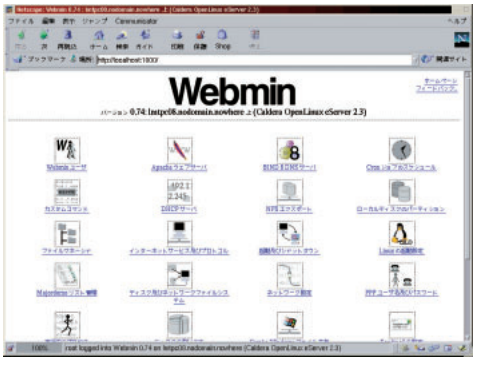
OpenLinux eServer 2.3日本語版

OpenLinux eServer 2.3 日本語版（以下eServer）は、米国Caldera Systemsが開発しているOpenLinux eServer 2.3を、株式会社ネオナジーが、日本語化して販売しているディストリビューションである。Caldera社のLinuxディストリビューションは、RPMのパッケージ管理方式を採用し、KDEをメインのデスクトップ環境としている。また、Caldera社独自の管理ツールCOAS（**画面1**）を組み込み、Linuxの管理を簡単にしているのが特徴だ。本製品はOpenLinux 2.3日本語版（以下eDesktop）のサーバ版に位置付けられ、eDesktopの特徴はそのままに、操作性の良いシステム管理ツールや、拡張されたカーネルの採用により、サーバ用としてチューニングされている。カーネルは2.2.14を採用し、商用ソフトには、IBM WebSphere Application Server Standard（90日間試用版）のほか、IBM VisualAge for Java for Linuxと Fontface4550 R Symon（さいもん）バージョン1.2がフルバンドルされている。価格は2万9800円（税別）で、正規購入ユーザーは30日までの電話でのサポートと、90日までの電子メールでのサポートを、あわせて5件まで無料で受けられる。

ネットワーク経由でeServerを簡単管理

eServerとeDesktopの違いをまとめると、**表1**のようになる。OpenLinuxシリーズには標準でCOASという管理ツールが付属する。このCOASは、主にX上で使用するツールなので、eServerでは、LAN経由での操作も考慮し、Webブラウザから管理可能な、Webminというツールが付属している。この日本語化されたWebminにより、コマンド操作に不慣れなユーザーも、Webブラウザから直感的操作でLinuxの管理が行える。Webminでの設定項目は多く、Webサーバ、DNSサーバ、メールサーバなど、各種ネットワークサーバの設定のほか、ディスクパーティションやユーザーの管理も行える。Webminについて詳しく知りたい読者は、**本誌11月号「ラクラク設定&カスタマイズガイド」**やWebminのサイト（http://www.webmin.com/webmin/）を参考にしていただきたい。

画面1　Caldera社独自の管理ツールCOAS
このCOASでハードウェア、ネットワークの設定などが行える。画面からもわかる通り、COASは複数の設定用モジュールで構成されるため、KDEのメニューから個々の設定ツールを起動することになる。このため、設定項目別にウィンドウが立ち上がる。



画面2
このWebminを使ってLAN経由でサーバの設定を変更できる。日本語化されているので、英語が苦手なユーザーも安心だ。eServerをインストール後、Webブラウザでhttp://hostname:1000/をURLに指定してアクセスする。

	OpenLinux eServer 2.3 日本語版	OpenLinux 2.3 日本語版
パッケージのCPU最適化	Pentium II最適化	i386以上最適化
Webmin	○	×
COAS	○	○
Disk Quota	○	△（設定変更により使用可）
搭載メモリ上限	4Gバイト（カーネル最構築が必要）	2Gバイト
Raw I/O	○	×
Dynamic File Descriptors	○	×
価格（税別）	2万9800円	1万2800円
付属の商用ソフト	IBM WebSphere Application Server Standard（90日間試用版） BM VisualAge for Java for Linux ディスプレイ・フォント「さいもん」	ATOK12 SE for LinuxWnn6 Ver.3 dp/NOTE for Linux ディスプレイ・フォント「さいもん」

表1　OpenLinux eServerとeDesktopの比較
eServerはWebサーバ、データベース運用に必須となる大容量のメモリサポートや、Raw I/Oに対応している。

インストーラもサーバ向け

eServerも、eDesktopと同様に、グラフィカルなインストーラLIZARDで、簡単なインストールが行える。eServerはこのほかにも、多数のマシンにインストールすることを考慮して、LIZARD不在インストールを用意している。LIZARD不在インストールとは、ネットワーク上にインストール用サーバを用意することで、同じ構成のeServerを、バッチ処理でインストールする方法だ。インストールするマシンをインストール専用フロッピーで起動すると、このマシンはインストール用サーバから、DHCPでIPアドレスを取得する。このあとにインストールサーバをNFSでマウントし、あらかじめ作成しておいたインストール用の設定ファイルをもとにして、自動インストールが始まるというわけだ。

拡張されたカーネル

eServerは、大規模なデータベースの運用を考慮してか、大容量のメモリ、Raw I/O、Dynamic file descriptorといった先進的な機能をサポートするために、パッチをあて独自拡張したカーネルを採用している。最大4Gバイトまでの物理メモリの使用には、カーネルの再構築（**画面3**）が必要となる。興味ある読者は、付属のドキュメント（CD-ROM内のルートディレクトリの「BIGMEMja.txt」）を参考にして、この機能を試して欲しい。

ためになる管理者ガイド

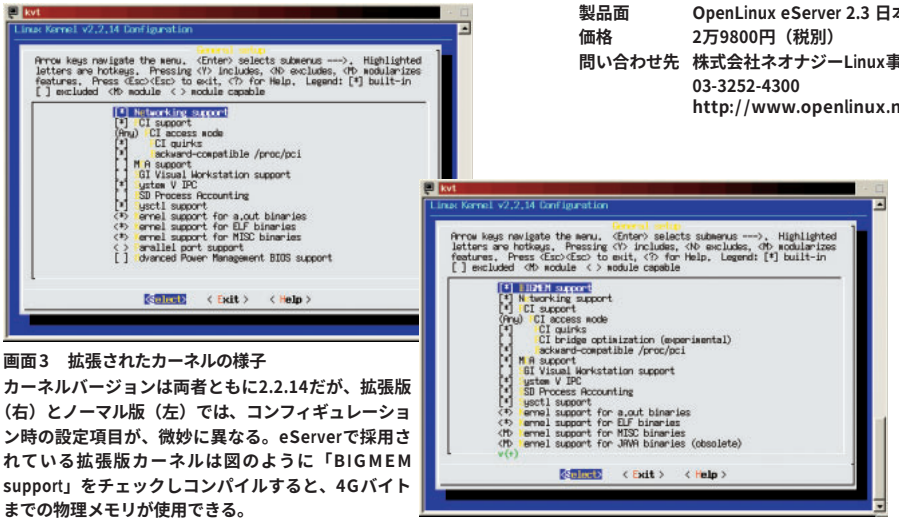
eServerの製品版には「システム管理者ガイド」が付属する。このガイドはインストールの解説、カーネルパッケージの作成方法のほか、管理ツールWebminを用いたネットワークサーバの設定、ユーザー管理など、Linuxシステムの管理法が、多くのページを割いて解説されている。また、Sniffit、Netwatch、tcpdumpを用いたネットワークの監視や、パケットフィルタリングの解説などを通して、基本的なネットワークのセキュリティと管理を学べるよう構成されている。これからLinuxサーバの管理を始める人にとって、非常に参考になるガイドである。

バンドルソフトはWebサーバを意識？

eServerには見やすい日本語表示を考慮した、商用版のさいもんフォントのほか、IBMのWebSphere Application Server Standard（90日間試用版）とVisualAge for Javaという、本格的なWebサーバ向けのJava開発環境がバンドルされている。これらにはインストール用スクリプトが用意されているので、アプリケーションの導入は容易である。またオープンソースのデータベースシステムには、PostgreSQLのほかに、他のLinuxディストリビューションではあまり目にしないMySQLが含まれており、高速さがウリのMySQLの採用は、IBMのJava開発環境とともに、Caldera社が今後eServerを、電子商取引をはじめとするEビジネスのプラットホームとして見据えていることの表れだろうか。

独自性というトレンド

今までのLinuxディストリビューションは、サーバ用途全般という、やや漠然とした使用目的のものが多かった。しかし、このeServerのように、管理ツールやカーネルなどに的を絞った拡張は、ほかのディストリビューションとの差別化を図るという意味で、今後のトレンドになっていくだろう。



画面3　拡張されたカーネルの様子
カーネルバージョンは両者ともに2.2.14だが、拡張版（右）とノーマル版（左）では、コンフィギュレーション時の設定項目が、微妙に異なる。eServerで採用されている拡張版カーネルは図のように「BIGMEM support」をチェックしコンパイルすると、4Gバイトまでの物理メモリが使用できる。

製品面　OpenLinux eServer 2.3 日本語版
価格　2万9800円（税別）
問い合わせ先　株式会社ネオナジーLinux事業部
03-3252-4300
http://www.openlinux.ne.jp/

OpenLinux eServer 2.3日本語版のインストール

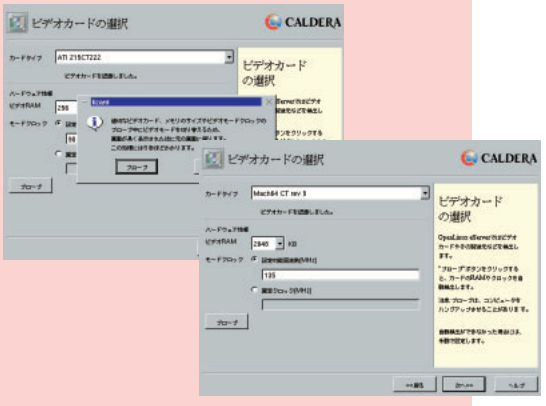
CD-ROMブートが可能であれば、OpenLinux eServer 2.3日本語版（以下OpenLinux）のCD-ROMからインストーラを起動します。CD-ROMブートできない場合は、インストール用のフロッピーディスクを作成します。作成手順は以下の通りです。(1)CD-ROMと空のフロッピーディスクを各ドライブへセット。(2)DOS窓を開きD: [Enter]と入力。CD-ROMドライブが「D:」以外の場合は、そのドライブ名をタイプします。(3)\col\launch\floppyディレクトリへ移動。(4)\col\tools\rawrite\rawrite3.com[Enter]と入力。(5)install.144[Enter]と入力。(6)フロッピードライブがA:ドライブの場合はa[Enter]と入力。(7)もう1度[Enter]を入力するとインストール用フロッピーの作成が始まります。

1 インストーラの起動

ブート方法に合わせ、CD-ROM、またはCD-ROMとフロッピーをドライブに入れて起動します。すると白黒のブート画面が表示されますので[Enter]を押します。カーネルが読み込まれ、しばらくするとXサーバを用いたグラフィカルなブート画面に切り替わります。

2 マウス、キーボードの選択

ブート画面が表示され、しばらくすると各種設定が始まります。まずマウスの設定です。2ボタンマウスを使用している場合は「3ボタンのエミュレーション」をチェックすると便利です。これは2ボタンの左右ボタンを同時に押すことで、3ボタンマウスの真中ボタンの代価をさせる機能です。「マウスのテスト」の箇所にマウスカーソルを持っていき、マウスが正常に動作するかチェックします。設定を終えたら「次へ>>」を押します。次はキーボードの設定です。日本語106、または109キーボードを使用する場合は「モデルの選択」で「日本語106キーボード」を、「レイアウトの選択」で「日本語」を選択して「次へ>>」を押します。

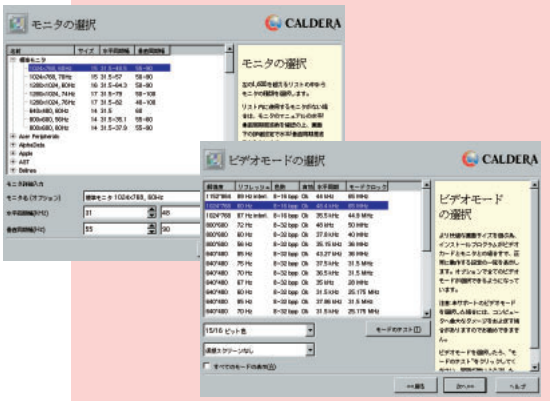


3 ビデオカードの選択

ビデオカードが自動認識されるので、「プローブ」ボタンを押し、ビデオカードのメモリとクロックを自動で検出します。ビデオカードのタイプや、メモリ、クロックが自動検出されない場合は、それぞれを手動で設定します。プローブが終わったら「次へ>>」を押します。

モニタ、ビデオモードの選択

リスト中に、使用しているモニタがあればそれを選択し、リスト中にない場合は、モニタのマニュアルを参考にしながら、「標準モニタ」の欄で適切な解像度、同期幅などの組み合わせを選択し「次へ>>」を押します。画面が変わると使用可能なビデオーモードの一覧が表示されるので、この中から使用するモードを選択します。ビデオモードを選んだら、一覧の下にあるプルダウンメニューで使用する色数を選択し、表示に問題がないか「モードのテスト」を押して確認します。問題がなければ「次へ>>」を押します。

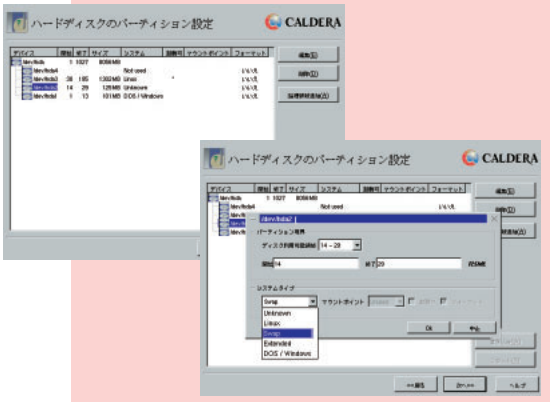


インストール先の選択

OpenLinuxのインストール先を選択します。「ハードディスク全体」にチェックを入れると、ハードディスクのパーティションが自動で切り直され、既存データが全て消去されます。ハードディスクをOpenLinux専用にする場合はこの「ハードディスク全体」を選択すると良いでしょう。「すでにあるパーティション」は、他のLinuxのfdiskや、PartitionMagicのようなパーティション作成ツールで、すでにパーティションが作成されている場合に選択します。「カスタム（エキスパート向け）」は、このあとにハードディスクのパーティション設定を行う場合に選択します。この後の説明では、「カスタム（エキスパート向け）」を選択したものとして進めていきます。

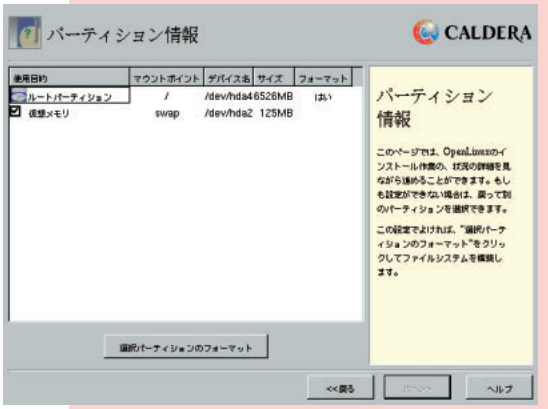
ハードディスクのパーティション設定

この画面では、使用可能なディスク領域が表示されます。ここではOpenLinux用に、Swapパーティションとシステムパーティションを作成します。はじめにSwap領域として使うパーティションを選択し、「編集」を押します。編集用の画面が表示されるので、プルダウンメニューの「システムタイプ」から「Swap」を選択し、実メモリの1～2倍程度の領域を確保して「OK」を押します。システム用パーティションも同様に、使用する領域を選択し「編集」を押します。編集用の画面が出たら、「システムタイプ」に「Linux」を、マウントポイントに「/」を選択します。「起動可」にチェックを入れ、「OK」を押します。パーティション設定の画面に戻ったら「書き込み」を押し、パーティション情報を書き込んで、次へ進みます。



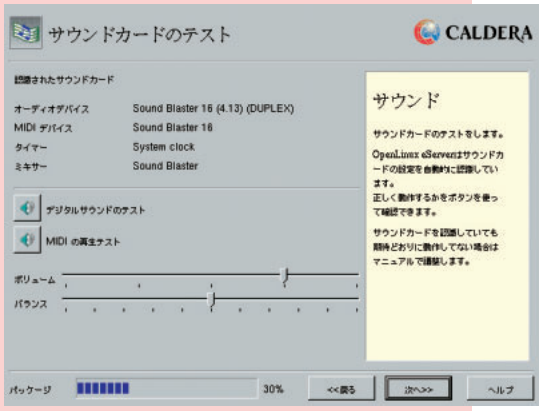
パーティション情報

OpenLinuxで使用するパーティションのフォーマットを行います。フォーマットする領域を確認し「選択パーティションのフォーマット」を押します。フォーマットが終了すると「次へ>>」が有効になるので、これを押して次へ進みます。



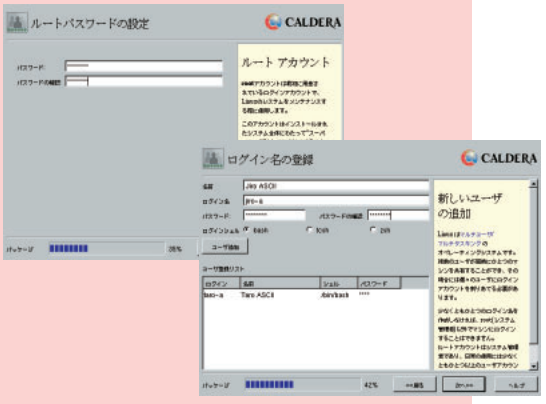
インストールタイプの選択

インストールするパッケージ群を選択します。各パッケージ群をチェックすると、画面の右側にそれぞれの説明が表示されるので、それらを参考にしながらインストールタイプを選択してください。



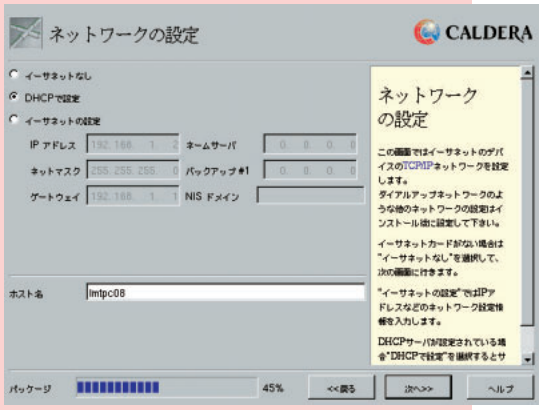
サウンドカードのテスト

自動認識されたサウンドカードについて、ボリューム、バランスなどを適当に設定したあと、サウンドのテストを行います。サウンドカードが自動認識されない場合は、インストール後に手動で設定を行います。



ルートパスワードと一般ユーザーの登録

まず、システム管理者用のアカウントとなる、ルートのパスワードを設定します。各欄に同じパスワードを入力して「次へ>>」を押します。次に一般ユーザーの登録を行います。ユーザーのフルネーム、ログイン名、5文字以上のパスワードを各欄に入力します。ユーザーごとに使用するシェルの選択し、「ユーザ追加」ボタンを押します。「ユーザ登録リスト」に新規ユーザーが登録されているか確認してください。この作業を繰り返すことで、複数のユーザーアカウントが作成できます。ユーザーアカウントを作り終えたら「次へ>>」を押します。

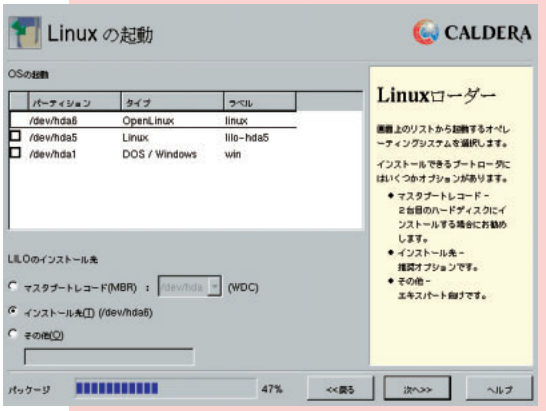


ネットワークの設定

ダイヤルアップでネットワークへ接続する場合や、イーサネットを使用する予定がない場合は、「イーサネットなし」をチェックして次へ進みます。イーサネットを使用して、IPアドレスをDHCPで取得する場合は「DHCPで設定」にチェックを入れ、ホスト名を入力します。固定IPアドレスを使用する場合は「イーサネットの設定」をチェックして、IPアドレスやDNSなどの情報を各欄へ入力します。ネットワークの設定が終ったら、「次へ>>」を押します。

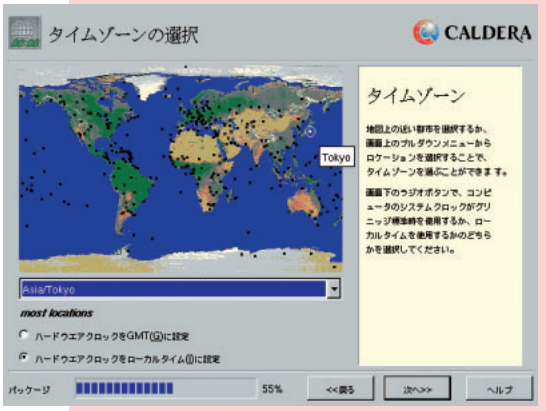
Linuxの起動（LILOの設定）

1台のマシンに、OpenLinuxとWindows 9xを共存させる場合は、LILOのインストール先に「マスターブートレコード（MBR）」を選択します。System Commanderなどのブートセレクタを使用する場合は、「インストール先(/dev/hoge)」をチェックします。他のOSもこのLILOで起動させたい場合は、「OSの起動一覧」でパーティションにチェックを入れます。「その他」はLILOをMBR、ルートパーティション以外にインストールする時に選択します。たとえば、LILOをフロッピーにインストールする場合、「その他」をチェックし、下の欄に「/dev/fd0」と入力します。LILOのインストール先を選択したら「次へ>>」を押します。



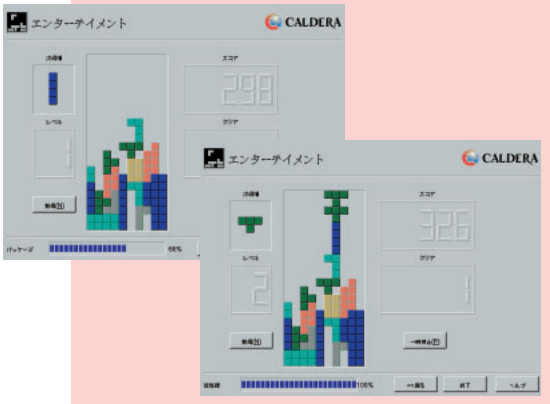
タイムゾーンの設定

タイムゾーンはデフォルトで「Asia/Tokyo」となっています。その他のタイムゾーンを使用するときは、使用したいタイムゾーンを世界地図上、またはプルダウンメニューから選択します。ラジオボタンは「ハードウェアクロックをローカルタイムに設定」をチェックしたままにして「次へ>>」を押します。



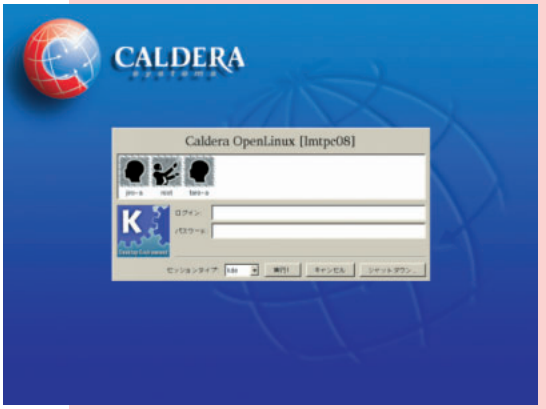
ゲームをする

タイムゾーンまでの設定が終ったら、OpenLinuxではおなじみの、テトリスライクなゲームが始まります。パッケージのインストール終了まで、ゲームをして遊んでください。しばらくして、画面下段にあるインストールの進行状況が、「後処理」に変わり、100%までいくと、「終了」ボタンが有効になります。「終了」ボタンを押すとインストール作業が終了し、自動的にOpenLinuxが起動します。



ログイン画面

OpenLinuxが起動すると右の画面になります。rootユーザーのほかに、インストール時に作成した一般ユーザーも表示されますので、確認してください。



初めてでも大丈夫!

今日から使えるLinux

文 : 竹内充彦
Text : Michihiko Takeuchi
photo : Shuichi Mito(Dee)

第1部

TurboLinux Workstation 6.0のインストール

Linuxが取っつきにくいOSだったのももう昔の話だ。数あるディストリビューションのうちでも、最近人気が高いのは、親切なインストーラやGUIで色々な操作、設定ができるツールを装備したものだ。また、これらのディストリビューションもバージョンアップによって、ますます手軽に使えるようになってきた。今回の特集では、Windowsしか使ったことがないという読者にも実際にLinuxに触れながら理解を深めていただきたい。

この特集では、SONY VAIO PCV-J10とTurboLinux Workstation 日本語版 6.0を例にとってLinuxのインストールから基本設定までを解説する。ほかのマシン、ディストリビューションでも基本は同じだ。

お手軽 VAIO PCV-J10

パソコンの売れ筋ランキングでも常に上位にあるソニーのPCシリーズVAIO。同社の誇るAV機能の統合もさることながら、その洗練されたデザインが人気の的になっているのも事実だ。しかし、価格的には決して安くはない。それでも欲しい。いわゆる憧れのマシンといったところだろうか。そんなVAIOに末っ子が生まれた。それがPCV-J10である（**写真1**）。

このPCV-J10にTurboLinuxのホッカホカの最新版であるWorkstation 日本語版 6.0をインストールしてみよう。

●PCV-J10の仕様

このPCV-J10、いわゆるエントリーモデルとしては若干高めの価格設定なのだが、その分満足度は高い。主な仕様は表1を参照してほしい。

●Linuxインストール時の注意点

PCV-J10にLinuxをインストールする際に、いくつかあらかじめわかっている注意点がある。

ハードディスクは、都合のいいことに、約10Gバイトのドライブが、約4Gバイトと約6Gバイトの領域に分割されている。6Gバイトのほうを丸まるLinuxに使わせてもらおう。

このマシンがVAIOシリーズである証ともいえるのがi.Linkカードだ。しかし、これは、現在のディストリビューションでは、サポートされていない。

また、搭載されているモデムも、残念ながらサポートされていない。モデムが使えないとなると、インターネットにどうやってつなぐか？　ここでは、ISDNを使うという前提で、シリアルポート（COM1）にTAをつなぐことにしよう。外付けモデムを使う場合も同じように設定して使うことができる。

サウンドカードは、Aureal Vortex（AU8810）だ。本誌4月号の「マルチ

写真1　VAIOシリーズの末っ子PCV-J10

本体	
項目	仕様
CPU	Celeron 500MHz
メモリ	64MB
ビデオ	i810チップセットに統合。メモリはメインメモリからシェア
ハードディスク	約10Gバイト（約4Gバイトと約6Gバイトに分割）
CD-ROMドライブ	最大40倍速
キーボード	PS/2
マウス	PS/2、ホイール付き
i.LINK	S400（6ピン）端子×2
サウンド	Aureal Vortex（AU8810）オンボード
モデム	Lucent Winmodem、56Kbps（V.90／K56flex）
ディスプレイ	
項目	仕様
大きさ	15型
CRT	FDトリニトロン管
最大解像度	1280×1024（60Hz）
走査周波数	水平30－70KHz、垂直48－120Hz

表1　PCV-J10の主な仕様

メディアLinuxで遊ぼう」でこのチップのサウンドカードを利用している。実は、サウンドのインストールは若干のスキルが必要になる。入門したてのユーザーには、ちょっと荷が重い。本特集を読み、力をつけてから挑戦してみてほしい。ちなみに、4月号のバックナンバーは在庫僅少なので、早めにゲットしておこう（笑）。

いざインストール開始

いよいよインストールを開始する。PCV-J10のCD-ROMドライブに、今月号付録のCD-ROM Disk 1をセットして再起動しよう。すると、自動的にTurboLinuxインストーラが起動するのだ。

いきなり、大きなTurboLinuxロゴが表示される。よく見ると、画面の下のほうに、「boot：」プロンプトが表示されている。そのまま待っていれば自動的にインストールプログラムが実行される。

●インストールプログラムの操作

ここで、インストールプログラムの基本的なキー操作を知っておこう。

インストールプログラムでは、ボタンやリストボックスが表示される。その中で、黄色で表示されている項目がある。これは、現在そこにフォーカスされて（焦点が合って）いるという意味だ。フォーカスはTabキーで移動することができる。

フォーカスされている選択項目を決定するのがEnterキーだ。［OK］ボタンにフォーカスされているときにEnterキーを押すと、そのボタンが押されたことになる。

この後のインストール解説では、「リストボックスから～を選択し［OK］ボタンを押す」のように表記する。これは、Tabキーでリストボックス内の項目にフォーカスを移し、矢印キーで～を選択し、再びTabキーで［OK］ボタンにフォーカスを移し、Enterキーを押す、という意味である。

TurboLinuxのインストールプログラムでは、デフォルト設定が用意されており、複数の選択肢がある場合でも、かなりの部分で最適な候補が選ばれた状態になっている。ユーザーがいちいち選び直さなくてもいいように配慮されているわけだ。本稿でも、デフォルト設定でいい部分は手順を省略して表記する。

「～」→［OK］とあれば、デフォルトの選択肢のまま［OK］ボタンを押すという意味だ。

なお文中の◇はインストールプログラムの画面のタイトルに対応している。

システムの設定

まずは、システムの基本的なハードウェアに対する問い合わせが続く。

◇キーボードタイプ

「jp106」→［OK］

◇PCMCIAサポート

［いいえ］

◇TurboProbe

［OK］

◇モジュールがロードされていません

［OK］

◇インストール元の選択

「CD-ROM」→［OK］

◇注意

［OK］

◇SCSI設定

［いいえ］

パーティションの設定

ここで、ちょっとした作業が必要になる。パーティション分割である。

PCV-J10では、1台のハードディスクの内容が2つの領域に分けられている。それぞれの領域をパーティションと呼ぶのだが、このうちの6Gバイトのほうのパーティション（Windowsではドライブ Dに相当）を、Linuxで使うために少々いじってやる必要があるのだ。

作業の内容は、その6Gバイトの領域をさらに2つに分けるというもの。1つは、Linuxや、アプリケーションプログラム、データファイルなどを置く領域だ。そして、もう1つが、メモリの内容をディスクに一時的に保存するための領域（スワップ領域）である。

●パーティション操作

◇パーティション設定

［CFDISK］

画面が一転し、CFDISKというプログラムが起動する（**画面1**）。画面を見ると、「hda1」にフォーカスされているのがわかる。これは現在Windowsがインストールされている領域だ（つまりドライブC）。このhda1には絶対に操作を加えてはならない。↓キーで「hda5」にフォーカスを移動しよう。このhda5が、これから操作する領域なのだ。

次に、Tabキーを押して［削除］を選択しEnterキーを押す。すると、フォーカス部分の名前（hda5）が消えてFSタイプに「空き領域」と表示される。

今度は、Tabキーで［新規作成］を選択してEnterキーを押す。すると、

［基本領域］か［論理領域］かを問い合わせてくるので、Tabキーで［論理領域］を選択しEnterキーを押す。領域のサイズを問い合わせてくる。「サイズ：」に続けて、すでに数字が表示されているが、無視してキーボードから、「6100」（単位はMバイト）と入力してEnterキーを押す。これは、スワップ用に130Mバイトほど残すためだ。スワップは実メモリ量と同じか、2倍ほどとればいい。

さらに、その領域をディスクのどちら側から確保するか問い合わせてくるので、Tabキーで［最初から］を選択しEnterキーを押す。

そうすると、今指定した指示にしたがって新しい「/dev/hda5」が作成される。FSタイプがLinuxになっているのも確認できる。これがLinuxをインストールする領域だ。次に余った空き領域をスワップに割り当てる。

またもや、Tabキーで［新規作成］を選択しEnterキーを押す。Tabキーで［論理領域］を選択しEnterキーを押す。

領域のサイズには136.56と表示されるので、今度はそのままEnterキーを押そう。残りすべてがスワップ用の領域として割り当てられる。

画面を見ると、/dev/hda6が作成されているはずだ。しかし、FSタイプがLinuxになっている。スワップとして使いたいので、このままではいけない。Tabキーで［FSタイプ］を選択しEnterキーを押す。

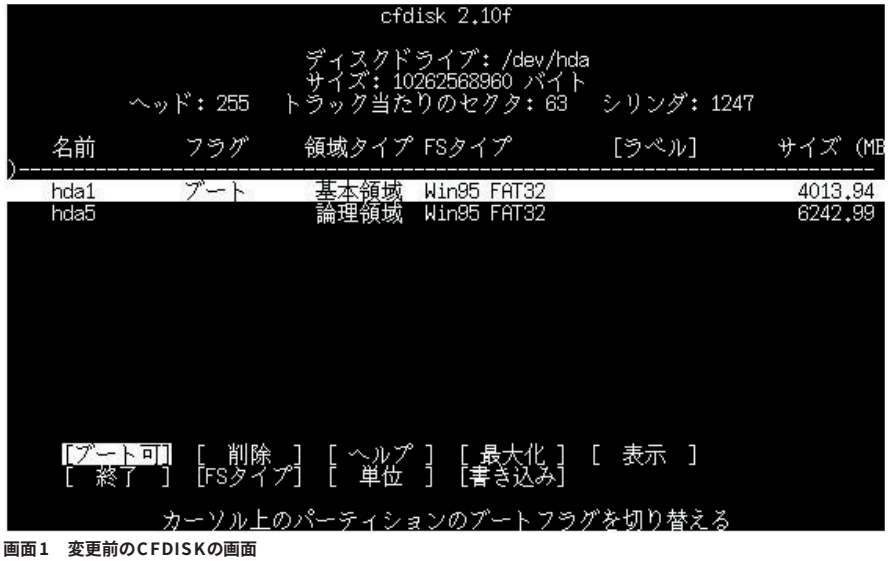
FSタイプ一覧が表示される。Linux swapは82と書かれている。そのままEnterを押すと、タイプ指定画面になるが、すでに「82」が表示されている。そこで、そのままEnterキーを押す。hda6のFSタイプに「Linux swap」と表示されているのを確認しよう。

さて、ここまでの作業を行うと**画面2**のようになるはずだ。もし画面のようになっていなければ、やり直しだ。この段階では、変更結果はディスクに書き込まれていないので、やり直しが可能なのである。フォーカスを移動して、［削除］と［新規作成］でやり直そう。くれぐれもhda1は変更しないように。

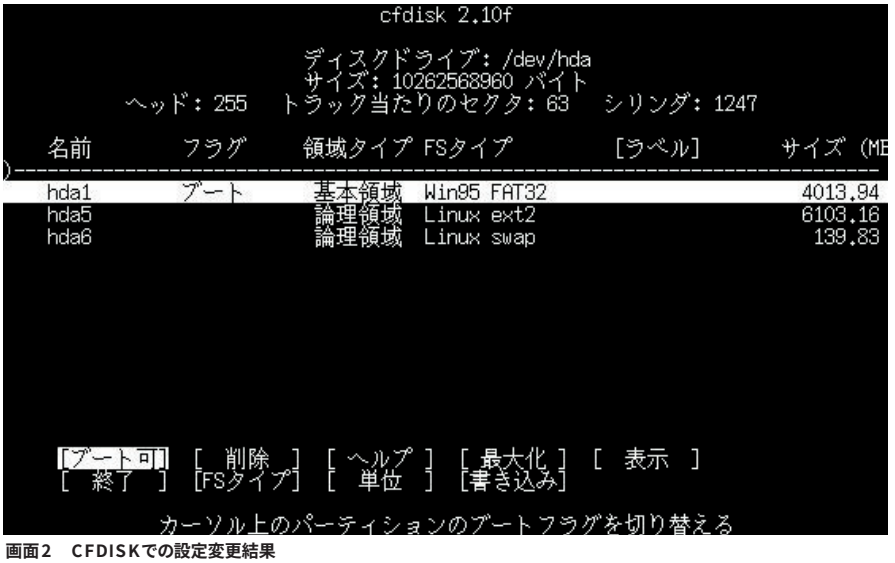
設定できたら、Tabキーで［書き込み］を選択してEnterキーを押す。本当に書き込んでよいかの確認が表示される。ここでの答えは「y」だけではダメ。「yes」と入力し、Enterキーを押す。これも安全のための気遣いなのだ。

CFDISKが終了し、パーティション設定のダイアログに戻る。

［終了］ボタンを押す。



画面1　変更前のCFDISKの画面



画面2　CFDISKでの設定変更結果

●パーティションのフォーマット

◇スワップ領域の設定

［OK］

◇マウントテーブルの設定

「/dev/hda1 Win95 FAT32」→［編集］

マウントポイントの指定ダイアログ

が表示されるので、「/mnt/win」と入力して［OK］ボタンを押す。マウントテーブルの指定画面に戻るので、［OK］ボタンを押す。

◇パーティションのフォーマット
［OK］

◇ネットワーク設定
そのまま［OK］

◇インストールログ
［OK］

インストール開始

◇ようこそ！
［OK］

◇インストールタイプ
「開発ワークステーション」を選択し［インストール］ボタン

◇選択の確認
［続ける］

◇カーネルの選択
「Default Kernel（386 or newer）」→［OK］

◇LILO設定
「/dev/hda マスターブートレコード」→［OK］

◇LILO設定
そのまま［OK］

◇起動可能パーティション
「/dev/hda1 Win95 FAT32」→［編集］

実は、そのまま［OK］ボタンでもかまわない。そうすると、起動オプションが「dos」になる。つまり、PC起動時にLILOが表示する「boot：」プロンプトで、Windowsを起動する場合「dos」と入力することになる。私は、Windowsを起動するのに「win」と入力したいので、ちょっと編集する。

［編集］ボタンを押すと、起動名の修正ダイアログが表示される。起動名に「win」と入力して［OK］ボタンを押す。起動可能パーティション画面に戻るので［OK］ボタンを押す。

◇TurboMkboot
［はい］

緊急時のため、起動用フロッピーディスクを作成しておく。フロッピーディスクをドライブに入れるようメッセージが表示されるのでディスクを入れ［OK］ボタンを押す。起動用フロッピーディスクが完成したというメッセージが表示されたら［OK］ボタンを押す。

◇時間帯の設定
「Japan」→［OK］

◇ルートパスワード設定
パスワード入力欄は2つあるが、2つとも同じものを入力する。入力内容は秘密保持のため画面には表示されない。入力したら［OK］ボタンを押す。

◇完了
［完了］

ここで、システムが再起動するので、CD-ROMとフロッピーディスクを取り出しておこう（そうでないとまたインストールプログラムが起動してしまう）。CD-ROMは、システムが終了するまで取り出しボタンを押しても出てこないかもしれない。そういう場合は、再起動してSONYロゴが表示されているときにでも取り出しボタンを押そう。

X Window Systemの設定

ハードディスクからLinuxが起動すると、引き続きX Window Systemの設定プログラムが自動起動される。X Window SystemはLinuxのGUIを受け持つプログラムだ（単にXと呼ぶこともある）。キーボードやマウスの設定も行う。

◇TurboXcfgへようこそ！
［OK］

◇コンソールキーボード設定
「jp106」→［OK］

◇キーボードモデル設定
「日本語106-key」→［OK］

◇キーボード配置の設定
「日本語」→［OK］

◇マウスの設定
「一般的なPS/2マウス」が選択されているので、「3ボタンのエミュレーション」オプションと、「Wheel（ローラー）付きマウス」オプションにチェックマークを付け、［OK］ボタンを押す。

チェックマークを付けるには、Tabキーでその項目にフォーカスを移動し、スペースキーを押す。

◇ビデオカードの自動認識
［はい］

◇検出結果
［OK］

◇検出データ
［検出値で設定］

◇ビデオメモリ

リストから「4096」（メモリ量・単位はMバイト）を選択して［OK］

◇ディスプレイベンダーの選択

リストから「デフォルトディスプレイ」を選択しEnterキーを押す。

この一覧には、PCV-J10付属のモニタに該当するモデルがないので「デフォルトディスプレイ」を選択した。

解像度を問い合わせてくるので、「1024×768」を選択して［OK］ボタンを押す。

◇デフォルト色数

リストから「24bpp Max resolution 1024x768」を選択して［OK］

◇画面サイズの設定

「1024× 768 @ 76Hz, 62.5kHz hsync」オプションにチェックマークを付けて［OK］

◇デフォルト画面サイズの選択

［OK］

◇フォント解像度

［100 DPI］

ここまでで、基本的なX Window Systemの設定は終わりだ。次は実際に設定が有効かどうかのテストにはいる。

◇設定のテスト

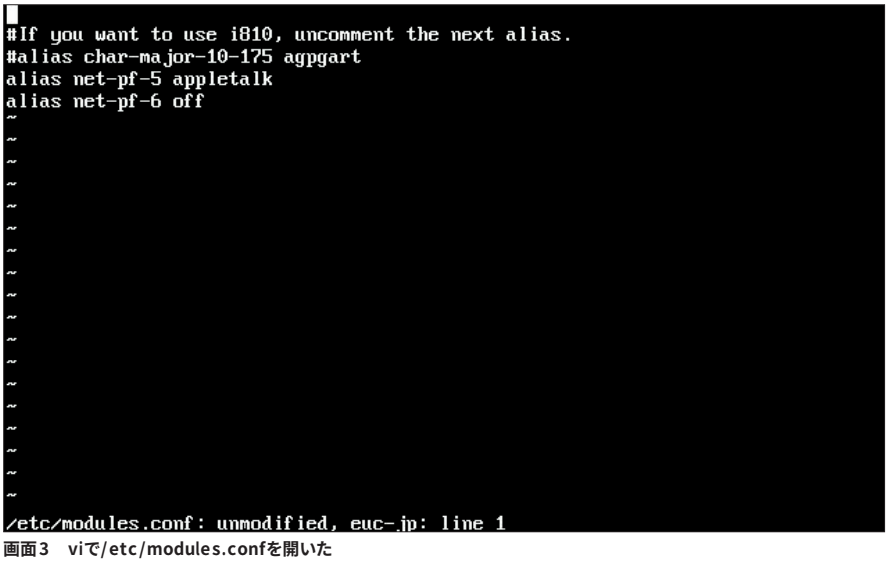
［続ける］

ここまでの設定をテストするのだが、画面には何も表示されない。失敗だ。本当ならばここで、X Window Systemが起動し、各種のチェックを行うのだが、どうやらX Window Systemが正しく起動できていないようだ。

実は、i810チップセットでX Window Systemを起動する場合には、別途設定が必要で、この段階ではそれはできない。この場は、これで済ませておいて、あとで追加設定をしよう。

「正しく表示されましたか？」というダイアログが表示されるので、［いいえ］ボタンを押す。すると、Turbo Xcfgの設定オプション画面になるので、リストから「Write Configuration」を選んでEnterキーを押そう。これまでの設定が書き込まれて、次に移る。

◇ウィンドウマネージャの選択



画面3　viで/etc/modules.confを開いた

リストから［GNOME］を選択して［OK］

◇起動サービス設定

そのまま［終了［ESC］］

以上で、インストールプログラムによるインストール作業は終わりだ。残すは、X Window Systemを正しく起動するための手作業による追加設定である。多少難しいかもしれないが、あと少しだ。

i810チップセットでXを使うために

i810チップセットのビデオチップはXFree86 3.3.6でサポートされ始めた。そのi810用ドライバでは、agpgartというモジュール（ドライバのようなもの）が別途必要である。このモジュール自体はすでに用意されているが、デフォルトでは読み込まれない。そこで、このモジュールを起動時に組み込まれるように設定する。

●まずログイン

さて、インストールプログラムでのインストールを終えた。画面上には何やら文字が表示されている。

画面をよく見てほしい。「localhost login：」と表示されているはずだ。その右にアンダーラインのカーソルが点滅している。

キーボードから「root」と管理者ユーザー名をタイプしてEnterキーを押そう。

すると、次の行に「Password：」と表示され、やはりアンダーラインカーソルが点滅する。そこで、先ほどのインストール時に「ルートパスワード設定」で設定したパスワードをキーボードから入力して、最後にEnterキーを押す。パスワードの入力を間違えた

り、あまり時間がかかり過ぎると、「Password：」表示が消えてしまう。そのときは、「root」の入力からやり直しだ。

正しくパスワードを入力すると、1〜2行のメッセージが表示され、今度は「[root@localhost /root]」と表示され、その右にカーソルが点滅しているはずだ。この表示をコマンドプロンプトと呼ぶ。

●viで設定ファイルを書き換える

いきなり大胆だが、viというプログラムを使う。viについては、特集2でも初歩的な使い方を紹介しているが、ここでは限られた機能しか使わないので、指示どおりキーを打ってほしい。

まずは、以下のようにキーを打って最後にEnterキーを押す。

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

すると、**画面3**のように変わる。もし、この画面のようにならなかった場合は入力を間違えている。また、コマンドプロンプトが表示されたならば、もう一度慎重にキーを打って入力しよう。また、「~」はたくさん表示されているが、画面のような英文の文字が何も表示されない場合は、キーボードから、

```
:q!
```

とキーを打ち、最後にEnterキーを押す。そうすれば、コマンドプロンプトの画面に戻るはずだ。

さて、うまく**画面3**のようになったら、画面をよく見てほしい。左上に白い四角が点滅しているのがわかるだろう。これがカーソルである。現在カーソルは1行目にあり、2行から何やら英文が書かれている。

↓キーを2回押し、カーソルを3行目の先頭の「#」の部分に移動する。「alias」という文字列の直前の「#」だ。

そうしたら、Xキーを1回だけ押す（小文字のxを入力する）。そうすると、先頭の「#」が削除される。

「#」が削除できたら、Shiftキーを押しながら、Zキーを2回押そう（大文字Zを2つ入力）。画面が変わって、コマンドプロンプトに戻るはずだ。

設定が完了したら、コマンドプロンプトに以下のように入力してEnterキーを押す。

```
shutdown -r now
```

これでLinuxが再起動し、変更した設定が有効になる。

●再びX Window Systemの設定

再起動したら、もう一度ログインして、今度はX Window Systemの設定の続きをする。コマンドプロンプトから次のように入力しEnterキーを押す。

```
kon
```

画面が一瞬消えるが、見た目はさほど変わらない。それでいいのだ。そうしたら、次のように入力しEnterキーを押す。

```
turboxcfg
```

ようやく設定オプション画面に戻ってきた（**画面4**）。さて、リストから「Test X Configuration」を選択しEnterキーを押す。X Window Systemが起動し、サンプル画面が表示されるはずだ。マウスで［Quit］ボタンをクリックして、設定画面に戻る。

そうしたら、リストから「Change Login Method」を選択してEnterキーを押す。［グラフィカルログイン］ボタンを押す。問題なく設定変更された旨のメッセージが表示されるので、［OK］ボタンを押すと、設定オプション画面に戻る。

最後にリストから「Write Configuration」を選択しEnterキーを押す。終了のメッセージが表示されるので［OK］ボタンを押す。

これでインストールはおしまいだ。shutdown －r nowと入力して、もう一度Linuxを再起動しよう。

画面4　TurboXconfigが起動する

第2部

Linuxの基本操作

インストールが終了して、自動的に再起動すると、Linuxが起動する。ひょっとして、Windowsは二度と起動しなくなってしまったのではないかと心配するかもしれないが、そんなことはないのでご安心を。ただ、インストール時の初期設定では、起動時に何も指定しないで放っておくと、自動的にLinuxが起動してしまうのだ。

Windowsを起動するためには、「boot：」というプロンプトに対し、インストール時に登録したように「win」と入力しEnterキーを押さなければならない。しかも5秒以内にだ！「まだまだWindowsを使う機会のほうが多いのに、何だか面倒なことになったなぁ…」と嘆くことなかれ。この特集の後半で、ちゃんとWindowsが自動的に起動するように設定を変更する。そのためにも、早くLinuxの操作に慣れてしまおうではないか。

ログイン

Linuxが起動すると、**画面1**のようなログインパネルが表示される。

Linuxは、ある種の機能を除き、基本的にはシステムにあらかじめ登録されているユーザーでないと利用できないようになっている。そのため、Linuxを利用する際には、ユーザー名とパスワードによる認証が行われる。これをログインと呼ぶ。Windowsでも、ネットワークに接続する場合や、複数のユーザーで利用する場合などには、ログオンという認証がある。Windows 95／98のログオンは、キャンセルしても、ネットワークに接続されなかったり、自分用のデスクトップ環境が再現されないだけで、システムの多くの機能は使えてしまう。しかし、Linuxでは、正しくログインしないと、基本的に利用できないと考えよう。

画面1　ログインパネル

ログイン時に必要なのは、あらかじめ登録されているユーザー名とそれに対応したパスワードだ。インストール時にrootというログイン名のパスワードを登録した。したがって、少なくともrootというユーザーは登録されていることになる。

ここでは、rootでログインしてみることにしよう。「Login：」と書かれたテキストボックスにユーザー名を「root」と入力しEnterキーを押す（この時マウスポインタがパネルの上に乗っていないと、入力を受け付けてくれないので注意）。すると入力したテキストボックスのタイトルが「Password：」に変わる。そこにインストール時に設定したパスワードを入力しEnterキーを押そう。この時、入力したパスワードが他人にバレないように、画面には「＊」で表示される。

これでログイン完了である。ログインパネルは消え、晴れてLinuxのデスクトップ環境が使えるようになる。

ここで注意してほしいのは、rootでログインするのは初心者にとっては危険だということだ（コラム参照）。この特集を順に読み進み、できるだけ早く自分用の一般ユーザーを登録し、ふだんは一般ユーザーでログインするようにしてほしい。

ログインをしたからには、利用が終わったら、ログアウトするのだが、その方法は57ページの「ログアウトとシャットダウン」の項で解説する。ここでは、次に進んで、実際にLinuxを利用してみよう。

ウィンドウマネージャSawmillとデスクトップ環境GNOME

さて、ログインすると一気に画面が変わる。どことなく見慣れたような、しかし、見慣れぬような、不思議な画面に感じるかもしれない。この画面こそ、あなたがこれからLinuxライフをエンジョイするためのウィンドウマネージャ「Sawmill」（ソウミル）と、その上で動くデスクトップ環境「GNOME」（グノーム）なのだ（**画面2**）。

画面各部を見ていこう。各部の名称については、**画面2**を参照してほしい。ふだん、Windows 98を使っているユーザーなら、大まかな画面構成は把握できるだろう。SawmillもGNOMEも、Windowsユーザーに直感的にわかりやすいように配慮しているからである。デスクトップはユーザーの好みに応じ

てカスタマイズできる要素を多分に持っている。カスタマイズについては、66ページの「デスクトップのカスタマイズ」を参照してほしい。

●ウィンドウ

画面の真ん中に表示されているのは、いわずと知れたアプリケーションウィンドウだ。最初のログイン時には、rootとしてGNOMEを使おうとしていることへの警告メッセージウィンドウが開いているはず。X Window System対応アプリケーションの多くは、このようなウィンドウ内で実行される（アプリケーションウィンドウは、表示されているメッセージウィンドウとは見た目が多少異なる）。Windowsと同様、バックグラウンドで実行されるようなアプリケーションはウィンドウ表示されない。また、X Window System未対応のアプリケーションは、ターミナルウィンドウや仮想コンソールウィンドウなど、テキストベースのウィンドウ内で実行する。ちょうどWindowsでいうMS-DOSプロンプト（いわゆるDOS窓）だと思えばいい。

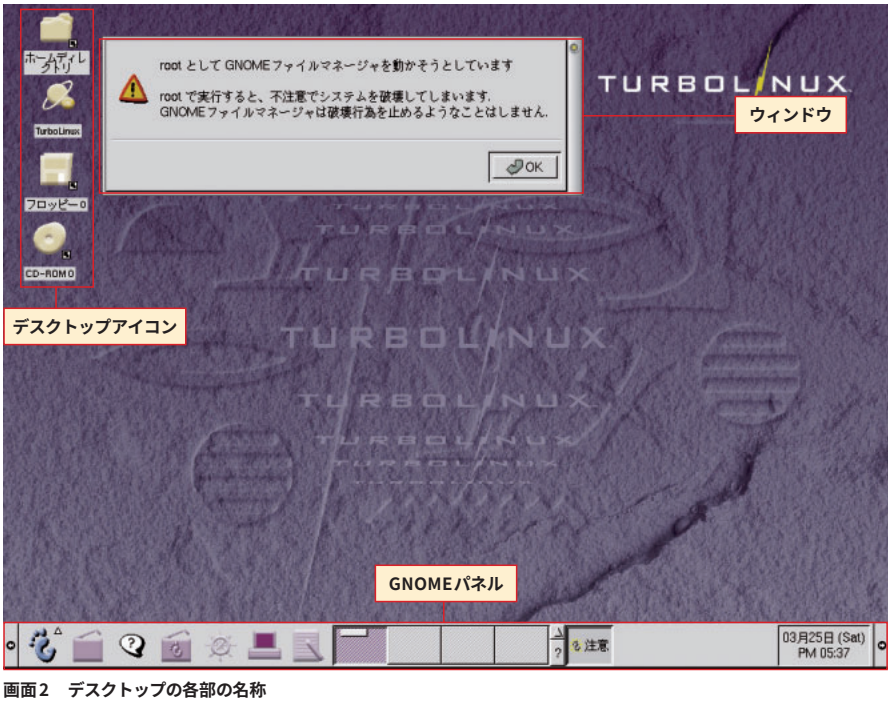
ウィンドウの操作については**62ページ**の「ウィンドウの操作」を参照してほしい。

●デスクトップアイコン

デスクトップに直接置かれているのがデスクトップアイコンで、これをダブルクリックすると、アプリケーションが起動する。画面が広い場合、よく利用するアプリケーションや、ファイル、ディレクトリなどへのリンク（Windowsで言うショートカットに相当）を、このデスクトップアイコンにしておくと能率がいい。Windows同様、このデスクトップアイコンは、ユーザーの好みで追加削除できるようになっている。

●GNOMEパネル

Windowsのタスクバーに相当するのがGNOMEパネルである。GNOMEパネルには、Windowsのスタートメニューに相当するメインメニューや、クイック起動バーに相当するアプリケーションランチャ、タスクや仮想画面を切り替えるページャなど、便利な機能が満載されている。これからLinuxを使っていくうえで、最もお世話になるユーザーインターフェイスといってもいいだろう。このGNOMEパネルの使いこなしの度合いが、Linuxライフの快適性に直結しているというわけだ。また、GNOMEパネルは、Windowsのタスクバーとは比較にならないほど、カスタマイズの自由度が高い。



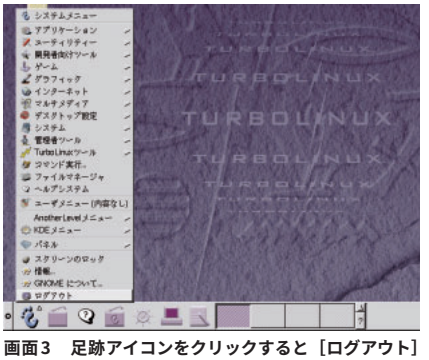
画面2　デスクトップの各部の名称

ログアウトとシャットダウン

ここで、ひとまずログアウトとシャットダウンの方法を知っておこう。

まずは、ログアウトについてだ。Linuxの利用を終えたらログアウトする。これは、システムの安全性の面からも、プライバシー保護の面からも、そしてシステムリソースの効率の面からも、実行するように心掛けたい。まして、rootでログインしたまま、席を空けるようなことは絶対にすべきではない。

ログアウトするには、まずパネルにある足跡マークのアイコンをクリックする。すると、メニューがポップアップするので、その一番下にある［ログアウト］という項目のところでもう一度クリックする（**画面3**）。すると「GNOMEを終了しますか？」と問い合わせて



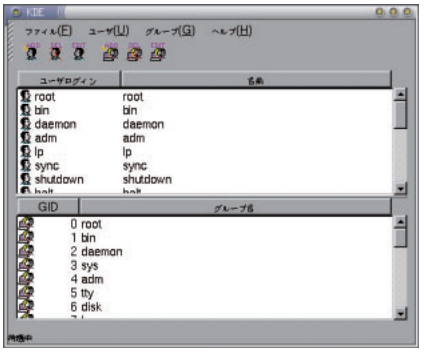
画面3　足跡アイコンをクリックすると［ログアウト］が選べる

くるので［はい］ボタンをクリックする。この時、［現在の設定を保存］オプションが選択できる。オプションボタンは、へこんで見えれば選択されている状態だ。これを選択してログアウトすると、現在のデスクトップの状態が保存され、次回ログインした時に、その状態が再現される。

ログアウトすると、システム起動時と同様のログインパネルが表示される。このログインパネルからメニューを選択することで、システムをシャットダウンしたり、再起動したりすることができる。パネルのメニューから［システム］－［システムの停止...］か、［システム］－［システムの再起動...］を選択する（**画面4**）。

Linuxも、Windows同様、システムが正しくシャットダウン処理をしてから、電源を切らないといけない。たとえば、Linuxのファイルシステムは、動作中は一部メモリ上に展開されている。通常はシャットダウン処理の際に、メモリの内容をハードディスクに書き戻して整合性を保つのだが、いきなり電源スイッチを切ったりすると更新内容が反映されなかったり、悪くするとファイルシステムそのものを破壊しかねない。したがって、この手順を踏まずに、電源を切ることは避けるべきだ。APM対応のPCならば、［システムの停止...］コマンドを実行すれば、自動的に電源も切れる。

画面4　ログインパネルの［システムメニュー］



画面5　KUser

新規ユーザーの作成

さて、rootでログインすることの危険性はコラムにも書いた。それに、ログイン時に毎回GNOMEから警告メッセージが出るのも煩わしいので、ここで、新規ユーザーを作成しておこう。

まずは、うっとうしい、Gnome Hintsと警告ウィンドウをそれぞれの［閉じる］ボタンをクリックして消しておく。

次に、GNOMEの足跡アイコンをクリックしてメニューを表示させ、［KDEメニュー］－［システム］－［ユーザーマネージャ］を選択しよう。するとユーザー管理ツール「KUser」が起動する（**画面5**）。

ウィンドウ内には2つのリストが表示されているのがわかるだろう。上側のリストは、このシステムに登録されているユーザー名で、下側のリストは、同じく登録されているグループ名だ。

現在登録されているものは、いずれもシステムが必要とするもので、インストール時に設定されているものだ。

Column

rootと一般ユーザー

インストール時に必ず作られるrootというユーザーは、システム管理用の特別なユーザーで、システムの重要な部分の変更や、ユーザーの管理など、一般ユーザーには禁じられている操作も行えるのだ。rootはLinuxシステムの中で全権を掌握しているといってもよい万能のユーザーなのである。これは、逆にいえば、rootとしてログインしていると、ちょっと間違えた操作をしただけで、システムに致命的なダメージを与えかねないということだ。ある意味危険なユーザーでもあるのだ。

そのため、たとえあなたが実際にシステムを管理する立場にあるとしても、Linuxのアプリケーションを利用するだけならば、ふだんは一般ユーザーとしてログインしよう。

あなたがrootとしてログインするのは、システムのメンテナンスや復旧をする、アプリケーションをインストール／アンインストールする、ユーザーを追加／削除する、など、システム管理上の実作業時のみにするよう心掛けたい。

ところで、Windows 95／98では、こうしたユーザーによる権限の切り分けがないため、ファイルを削除する方法さえ知ってれば、誰でも簡単にWindowsシステムに致命傷を与えられる。あなたが知らないうちに、誰かがちょっと間違っただけで、システムが起動しなくなってしまう可能性があるのだ。だが、Linuxではそうしたことはめったに起きない。たとえ初心者ユーザーがいたとしても、システムに支障をきたすようなことをする権限を与えはしないからだ。

最近では、家族がそれぞれのメールアドレスを持っていることも珍しくはないと聞く。家族内では、コンピュータスキルのバラつきがあるかもしれない。こういう時にも、システム管理者と一般ユーザーに分かれていることが意味を持ってくる。

家族みんなで1台のPCを使う場合などには、WindowsよりもLinuxのほうが都合がいいこともあるのだ。

このうち、あなたがユーザーとしてログインするのに使うユーザー名はrootぐらいで、あとはシステムが内部で使っていたりする。

グループとは、ユーザー管理上の分類で、ユーザーが所属するグループごとに権限を与えたり、制限したりが可能になる。たとえば、ユーザー名にもグループ名にもrootがある。rootというユーザーはrootグループに属しているのだ。新規ユーザーをrootグループに所属させれば、そのユーザーはrootグループに与えられた権限を獲得する。システムには一般ユーザー用にUsersというグループもすでに用意されているので、ここではそれを使うことにする。

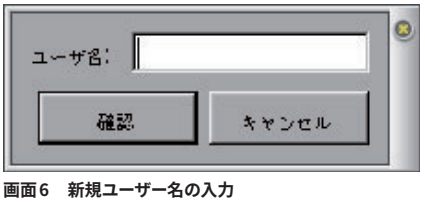
それでは、あなたがこれからふだん使うユーザー名を登録してみよう。KUserのメニューから［ユーザ］—［追加］を選択するか、［ADD］ボタンをクリックしよう。すると、新規ユーザー名入力のための小さなダイアログボックスが表示される（**画面6**）。ここに新規ユーザー名を入力して［確認ボタン］をクリックする。すでに、そのユーザー名が存在する場合は、警告メッセージが表示されるので、別のユーザー名を入力する（**画面7**）。

新規ユーザー名が受け入れられると、設定用のダイアログボックスが表示され、［ユーザ情報］タブが表示されているはずだ（**画面8**）。

順に設定していこう。［フルネーム］には、ユーザーのフルネームをアルファベットで入力しよう。必須というわけではないが、ユーザー名が本名とかけ離れている場合や、ユーザーが増えてきた場合に、管理上わかりやすい。次に［ログインシェル］だ。シェルとは、テキストベースの端末からLinuxを利用する時の、ユーザーインターフェイスである。ここでは、bashに設定しておこう（bashの使い方については**98ページ**からの「シェルを使おう」を参照）。テキストボックスの右にある▼ボタンをクリックして、一覧から［/bin/bash］を選択する。［ホームディレクトリ］は、そのユーザーが自分用のデータファイルや設定ファイルを置いたり、作業に使うディレクトリのことだ。すでに「/home/ユーザー名」と表示されているはずなので、ここはそのままでOKだ。オフィス1、オフィス2、アドレスは、そのユーザーのオフィス（仕事をしている場所）や、連絡先を記入しておく。fingerコマンドなどで参照できるが、必須ではないのでここでは空欄でもかまわない。［ホームディレクトリを作成］をチェックすると、そのユーザー用のホームディレクトリが［ホームディレクトリ］で指定した場所に自動的に作成される。ここではチェックしておこう。［スケルトンをコピー］をチェックしておくと、各種設定ファイルの雛形がホームディレクトリに自動的にコピーされ、シェルやGNOME等の基本的な利用環境が整う。ここもチェックしておこう。［ユーザプライベートグループ］は、そのユーザーをほかのグループに所属させるのではなく、そのユーザー専用のグループに所属させるためのものだ。ここではチェックを外しておこう（**画面9**）。

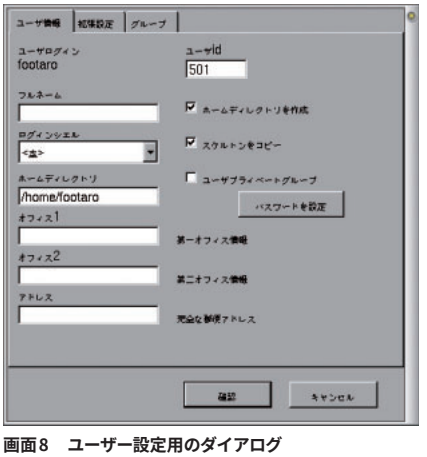
入力が済んだら［パスワードを設定］ボタンをクリックして、このユーザーのパスワードを設定する。インストール時にrootのパスワードを登録したのと同じ要領で、確認のため2つのテキストボックスに同じパスワードを入力し、［確認］ボタンをクリックしよう（**画面10**）。

［ユーザ情報］の入力が済んだら、［グループ］タブをクリックする。そして［第一のグループ］テキストボックスの▼ボタンをクリックして、一覧か



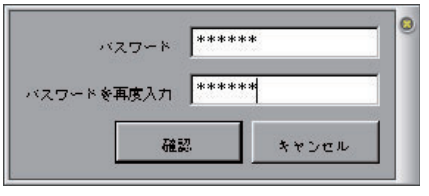
画面6　新規ユーザー名の入力

画面7　すでにユーザー名が使われている場合の警告メ



画面8　ユーザー設定用のダイアログ

画面9　入力後のKUserの［ユーザ情報］パネル

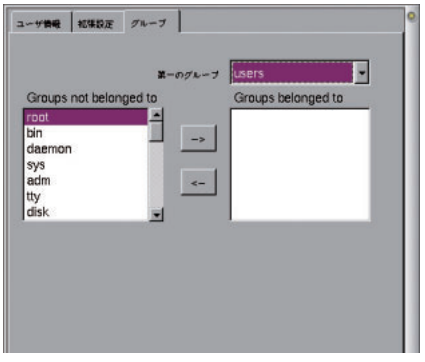


画面10　パスワードの設定

らusersを選択する（**画面11**）。そうしたら［OK］ボタンをクリックしてダイアログボックスを閉じる。

KUserの上側のリストに登録したユーザー名があるのを確認したら、メニューから［ファイル］—［終了］を選択する。変更を保存するかどうか問い合わせてくるので、［保存］をクリックする。保存しないと作成したユーザーが登録されないので注意しよう。

これで新規ユーザーの登録ができた。ログアウトして、登録したユーザー名でログインし直してみよう。



画面11　グループの設定

GNOMEパネルを使おう

一般ユーザーでログインし直したら、GNOMEの機能をいろいろ試してみよう。

まずは、GNOMEパネルの各部を見ていこう（**画面12**）。パネル両端にはパネルかたづけボタンがあり、左から、メインメニューアプレット、アプリケーションランチャ、GNOMEページャーアプレット、時計アプレットが用意されている。ここで言うアプレットとは、パネルの上で機能を提供するプログラムのことだ。

●パネルかたづけボタン

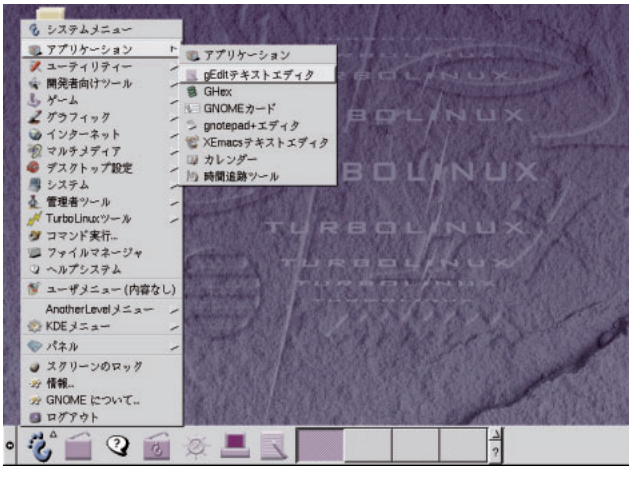
パネルは邪魔なときにかたづけることができる。パネルの両端に付いている、パネルかたづけボタンをクリックすればいい。右端のボタンをクリックすれば、画面の右外に、左端のボタンをクリックすれば画面の左外に、パネルが引っ込む。もう一度クリックすると元に戻る。

Windowsのタスクバーと同様、自動的に隠すこともできる。メインメニューから［パネル］—［プロパティ］—［隠蔽方式］—［自動的に隠す］を選ぼう。すると、マウスポインタが画面下端に来た時にだけパネルが表示され、それ以外の時は自動的に隠れるようになり邪魔にならない。画面の狭いノートパソコンなどでは、便利な機能だ。

●メインメニューアプレット

Windowsのスタートメニューに相当するのが、GNOMEのメインメニューだ（**画面13**）。パネル上の足跡マークのボタンをクリックするとメニューが表示される。このメニューに表示されるコマンドをクリックすることで、アプリケーションを起動したり、システムの設定を変更したり、デスクトップ環境をカスタマイズしたりすることができるのだ。メインメニューは階層メニューになっており、コマンド名の右に三角のマークが付いてる場合は、マウスポインタを持って行くと、その右

画面12　GNOMEパネル



画面13　メインメニューの右に三角マークがある項目はサブメニューになっている

画面14　ウィンドウ切り替えボタン

画面15　仮想デスクトップの切り替えボタン

側にサブメニューが表示される。

●GNOMEページャーアプレット

GNOMEページャーは、アクティブウィンドウの切り替えや、最小化されたウィンドウを元の大きさに戻す、仮想デスクトップの切り替えなどの役割を果たすアプレットだ。

まずは、アクティブウィンドウの切り替えについて見ていこう。Windowsユーザーには、タスク切り替えとしてお馴染みの操作だ。ページャには、現在稼動しているウィンドウ対応アプリケーションのウィンドウ名が付いたボタンが表示される（**画面14**）。このボタンをクリックすることで、アクティブウィンドウを切り替えることができる。また、最小化されているウィンドウを元の大きさに戻す時にも、このボタンをクリックする。Windowsと異なり、このボタンをクリックしてウィンドウを最小化することはできない。

次は仮想デスクトップの切り替えだ。仮想デスクトップとは、モニタに表示できる大きさのデスクトップ（現実のデスクトップ）よりも広大なデスクトップをメモリ上に置き、その一部をモニタに表示する機能だ。仮想デスクトップは4画面分あり、そのうちの1つが実際にモニタ画面に表示されている（**図1**）。表示する仮想デスクトップを切り替えるのが、画面右手にあるページャーアプレットだ。

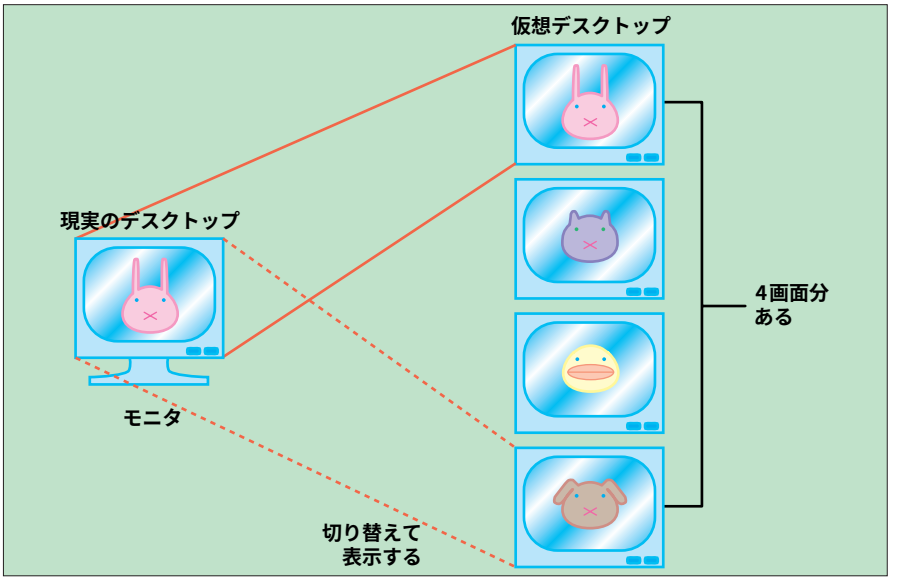


図1　仮想デスクトップの考え方

パネルに、無印のボタンが4つ並んでいる。4つのうち、一番左のボタンだけは押された状態になって（へこんで表示されて）いるはずだ（**画面15**）。その1つ1つのボタンが、それぞれの仮想デスクトップに対応しているのだ。よく観察していると、デスクトップ上に開いたウィンドウの数や形、位置が、一番左の押された状態のボタンの中に投影されているのがわかるはずだ（**画面16**）。では、左から2番目のボタンをクリックしてみよう。今までデスクトップに表示されていたウィンドウが消える。別のデスクトップが表示されたのである。1番目のボタンには、さきほど開いていたウィンドウの情報がちゃんと表示されたままになっている。ここで何かアプリケーションを起動すると、そのウィンドウは現在の（2番目の）仮想デスクトップに表示されるというわけだ（**画面17**）。

画面16　1番目の仮想デスクトップ

画面17　2番目の仮想デスクトップ

このように、仮想デスクトップを切り替えることで、作業単位でデスクトップを切り替えて利用することができるのだ。たとえば、Webをブラウズしていて、それをときどき参照しながら別の作業をするとしよう。Webはできるだけウィンドウを大きくしてブラウズしたい。そんな時は、別の仮想デスクトップで、Netscape Communicatorを最大化して起動し、デスクトップごと切り替えればいいのだ。

●アプリケーションランチャ

アプリケーションランチャは、よく使うアプリケーションを起動するためのものだ。Windowsではクイック起動バーと呼ばれているものがこれに相当する。初期設定では、「ネットスケープ・ブラウザ」(Netscape Communicator)や、「漢字ターミナル」(kterm)などが登録されている(**画面18**)。

アプリケーションランチャのアイコンをマウスでクリックすると、そのアプリケーションが起動する。このアイコンの並び順が気に入らなければ、マウスの中央ボタンでドラッグすることで順番を入れ替えることができる。好みに応じて、アプリケーションアイコンの追加と削除も可能だが、その方法については、66ページの「デスクトップのカスタマイズ」を参照してほしい。

試しに[漢字ターミナル(kterm)]アイコンをクリックして起動してみよう。ktermは、文字ベースのアプリケーションを利用するためのアプリケーションで、起動するとウィンドウ内でbashというシェルが起動する(画面19)。bashについては98ページからの「シェルを使おう」を参照してほしい。ここでは、キーボードから「exit」と入力してbashを終了しよう。bashが終了するとktermのウィンドウも自動的に閉じる。

ウィンドウの操作

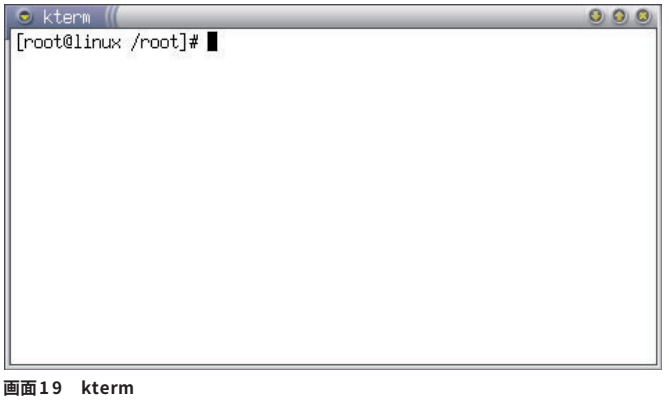
ここで、基本的なウィンドウの操作を説明しておこう。ウィンドウの各部の名称は**画面20**のようなものだ。ふだん、Windowsを使っているユーザーには直感的にわかるようになっているので、とくに戸惑うことはないだろう。

ウィンドウの上側にアプリケーション名が表示されている部分がタイトルバーだ。Windowsと異なるのは、ダイアログボックスの場合、名前が表示されず、ウィンドウの右側に縦に表示されることである。マウスでここをドラッグするとウィンドウを移動させることができる。また、タイトルバーをダブルクリックすると、タイトルバーだけを残してウィンドウが隠れる。もう一度ダブルクリックすると元どおり表示される。

ウィンドウの4辺または4方の角をドラッグすれば、大きさを変えることができる(**図2**)。ただし、大きさを変えられないウィンドウもあるので注意しよう。

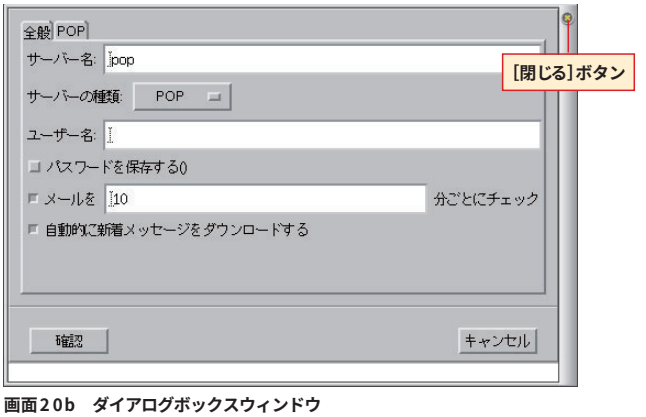
ウィンドウのタイトルバーには、いくつかボタンが並んでいる。左端にコントロールメニューボタン、右端に右

画面18　アプリケーションランチャ



画面19　kterm

画面20a　アプリケーションウィンドウ



画面20b　ダイアログボックスウィンドウ

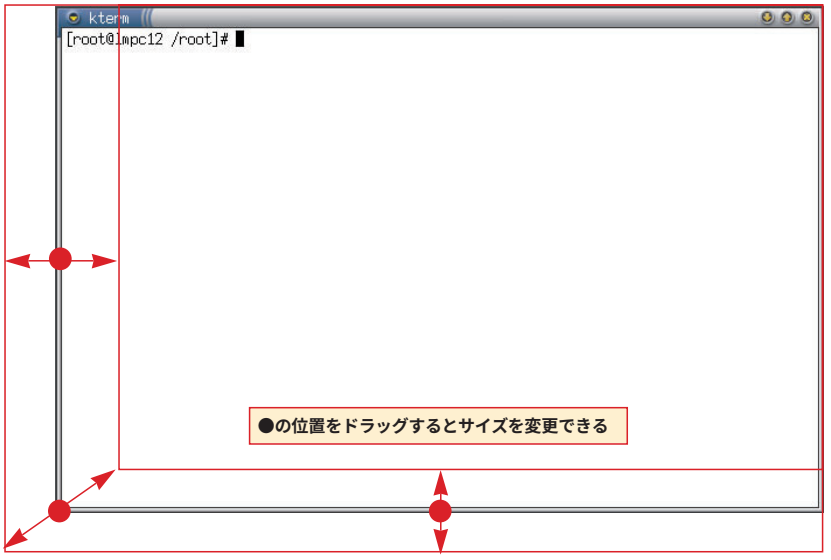


図2　ウィンドウの大きさを変更する

から、［閉じる］ボタン、［最大化／元の大きさに戻す］ボタン、［最小化］ボタンが並んでいる。ダイアログボックスの場合、上端に［閉じる］ボタンしか表示されないこともある。

ファイルマネージャを使おう

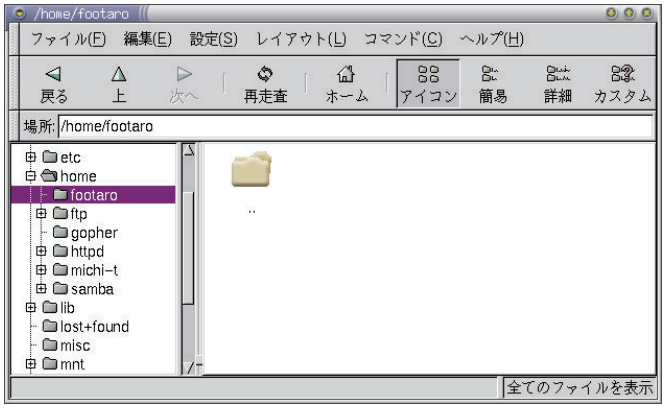
実はGNOMEが提供するパネルは、このファイルマネージャのバックエンドで動いている。このファイルマネージャこそGNOMEの表の顔であるといってもいいのだ。それはさておき、ファイルマネージャは、Linux上のファイル操作を統一されたインターフェイスで行うことができる。また、ファイルマネージャ単体で、ファイル操作や、アプリケーションの起動などを行えるが、他のアプリケーションと併用することで使い勝手がさらに広がる。

では、さっそくファイルマネージャを起動しよう。ファイルマネージャはメインメニューからも起動できるが、デスクトップアイコンの［ホームディレクトリ］をダブルクリックするほうが手っ取り早い。ファイルマネージャの起動時画面は**画面21**のようになる。表示されているのは、あなたのホームディレクトリ「/home/ユーザー名」だ。

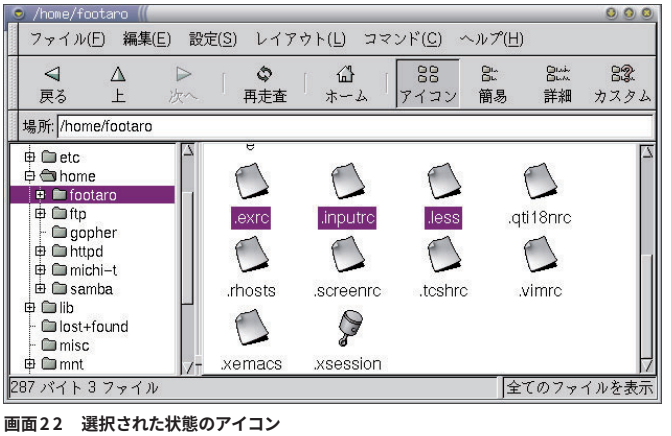
左側にディレクトリツリーが表示されていて、現在のディレクトリが紫のラインで表示されている。右側が現在のディレクトリ内容だ。現在のディレクトリを変更したい場合は、ディレクトリツリーのディレクトリ名をクリックする。あるいは、右側のディレクトリ内容が表示されているところで、フォルダの形をしたアイコンをダブルクリックして移動する。ここで、「..」という名前のフォルダアイコンがあるが、これは現在のディレクトリの1つ上の階層のディレクトリという意味だ。つまり「/home/ユーザー名」から、このアイコンをダブルクリックすると、「/home」に移動するというわけだ。バンドが付いたフォルダアイコンが表示されることがあるが、このディレクトリには移動できない。これは、そのユーザーにディレクトリに移動するアクセス権がないからだ。

●ファイルやディレクトリの選択

操作対象となるファイルやディレクトリを選択するには、ファイルやディレクトリのアイコンをクリックする。アイコンの下の名前が反転表示されれば、選択されている証拠だ（**画面22**）。他のアイコンをクリックしたり、アイコンとアイコンの間の何もないところをクリックすると選択は解除される。



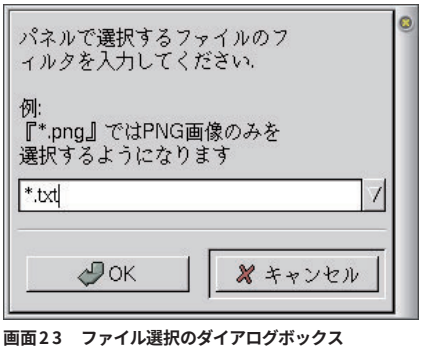
画面21　ファイルマネージャ



画面22　選択された状態のアイコン

複数のファイルを選択したい場合は、最初のアイコンをクリックし、2つめ以降のアイコンをCtrlキーを押しながらクリックする。連続する（左から右、上から下に順に並んで表示されている）アイコンをまとめて選択したい場合は、先頭の（最も左上に位置する）アイコンをクリックし、末尾の（最も右下に位置する）アイコンをShiftキーを押しながらクリックする。マウスをドラッグして範囲を指定することで、指定範囲内のアイコンをすべて選択することもできる。現在のディレクトリにある、アイコンすべてを選択したいなら、メニューの［編集］—［全選択］が便利だ。

ある特定の種類のファイルだけ選択したい場合がある。たとえば、ファイルの拡張子が「.txt」のファイルをすべて選択したいという場合である。そういう時は、メニューの［編集］—［ファイル選択...］を選ぶ。すると**画面23**のようなダイアログボックスが表示される。テキストボックスに「＊.txt」と入力して、［OK］ボタンをクリックすれば、目的のファイルだけが選択できる。このファイル選択で選ばれたアイコンは、すでに選択されているアイコンに加えて選択される。



画面23　ファイル選択のダイアログボックス

ところで、ここで使った「＊」は、ワイルドカードといい、任意の文字列として解釈される。つまり「＊」の部分は「何でもいい」という指示をしたことになるのだ。つまり、「＊.txt」は「何でもいいから末尾に.txtが付く」ものという意味に解釈される。名前のどこかに「turbo」という文字列を含むものだけを選択したければ「＊turbo＊」などと指定できる。名前の先頭が「turbo」と限定するなら「turbo＊」だ。

ワイルドカードは、これ以外に「？」も指定できる。「？」は任意の1文字という意味だ。「mp？」と指定した場合は「mp3」や「mpg」は選択されるが、「mpeg」は文字数が合わないので選択されない。

●ファイルの移動とコピー

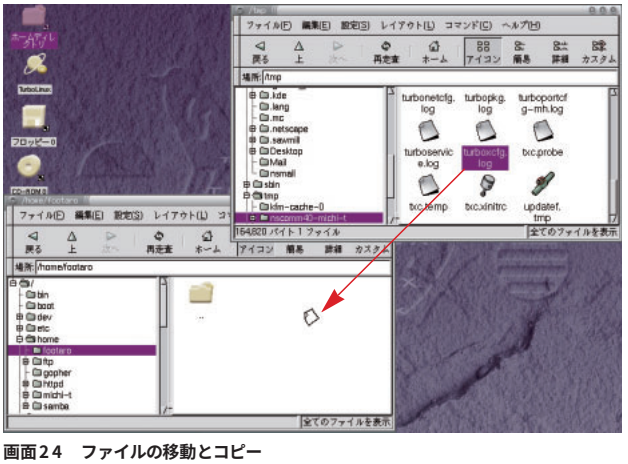
ファイルの移動は簡単だ。選択したファイルを、移動したいディレクトリにドラッグすればいい。移動先は、同じファイルマネージャのディレクトリツリーでもいいし、別のファイルマネージャを起動しておいて、そちらで開いている現在のディレクトリ内容でもかまわない（**画面24**）。

ファイルを別のディレクトリにコピーしたい時は、Ctrlキーを押しながらコピー先のディレクトリにドラッグする。このときShift＋Ctrlキーを押しながらドラッグすると、実体はコピーされず、シンボリックリンクが作成される。シンボリックリンクは、実体を残したまま、リンク情報だけを持つファイルで、表面的には実体と同様に扱える。Windowsでいうところのショートカットと似たようなものだと考えればわかりやすいかもしれない。

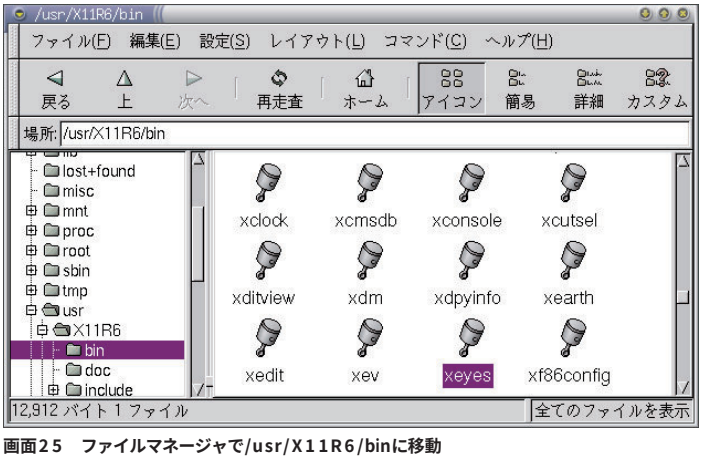
アクセス権によっては、表示はできるが、書き込めないディレクトリというのも存在する。そうしたディレクトリには、ファイルを移動したり、コピーしたりできない。

●ファイルの削除

選択したファイルを削除するには、ファイルマネージャのメニューから［ファイル］—［削除］を選択する。本当に削除するかどうか問い合わせてくるので、［はい］ボタンをクリックすればいい。ただし、Linuxでは、いったんファイルを削除してしまうと、復



画面24　ファイルの移動とコピー



画面25　ファイルマネージャで/usr/X11R6/binに移動

活させることはできないので、ファイルの削除は慎重に行ってほしい。また、ファイルの削除も、アクセス権によっては実行できないものがある。

●アプリケーションの起動

アイコンをダブルクリックするとアプリケーションを起動することができる。ただし、ファイルマネージャから起動できるアプリケーションは、X Window System対応アプリケーションに限られる。

まず、アプリケーションそのもののアイコンをダブルクリックして起動してみよう。ディレクトリを「/usr/X11R6/bin」に移動しよう。エンジンのピストンマークのアイコンがプログラムアイコンだ（**画面25**）。このディレクトリにあるプログラムの多くはX Window System対応のアプリケーションだが、ユーザーが起動するものばかりではないので、むやみに起動しようとしてはいけない。試しに「xeyes」というアプリケーションアイコンをダブルクリックしてみよう。マンガのような目玉が表示されるはずだ（**画面26**）。xeyesは、その視線がマウスポインタを追うというだけのプログラムだ。

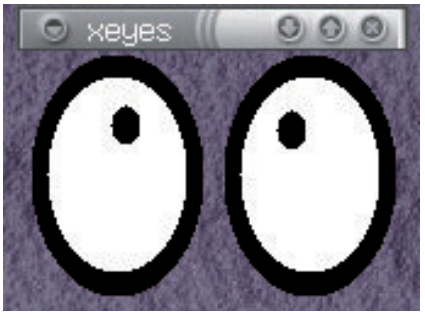
次にデータファイルをダブルクリックして、それに対応したアプリケーションを起動するという方法がある。Windowsなどではお馴染みの方法だ。GNOMEでももちろんサポートされている。まだ、インストールしたてで、データファイルはないが、ファイルの拡張子に応じて、対応したアプリケーションが起動するようになっているのだ。起動するアプリケーションと、ファイルのデータ形式（拡張子）の関連付けは、GNOMEコントロールセンターのMIME型で設定されている。GNOMEコントロールセンターは、パネルのアプリケーションランチャに登録されているので起動してみよう。

起動したら、左側のリストの一番上にある「MIME型」をクリックしよう（**画面27**）。すると、データ形式と拡張子の対応一覧が表示される。試しにリストの中ほどにある「mp3」を選択し、［編集］ボタンをクリックしてみると、**画面28**のようなダイアログボックスが表示される。ここには「.mp3」という拡張子を持つファイルを表示するためのアイコンと、そのデータファイルに対応して起動するアプリケーション「mpg123」が設定されているのがわかる。mpg123はインストールされていないので、ここでは「xmms」（**画面29**）を設定しよう。［Open］テキストボックスに、直接、

```
/usr/X11R6/bin/xmms %f
```

と入力する。これで、「.mp3」という拡張子が付いたファイルをダブルクリックするとxmmsが起動するようになった。

GNOMEでは、メジャーな拡張子については、あらかじめアプリケーションとの関連付けがされている。関連付けされているアプリケーションが正しくインストールされていれば、データファイルをダブルクリックするだけで、自動的に起動する。



画面26　マウスポインタを目で追うxeyes

画面27　GNOMEコントロールセンターのMIME型

画面28　拡張子「mp3」のアプリケーションとの関連付け

画面29　MP3再生アプリケーション「xmms」

第3部 システムを活用する

デスクトップのカスタマイズ

Windowsでもデスクトップを派手に飾る今日、GNOMEでも、デスクトップを好みに応じてカスタマイズしたい（なんのこっちゃ）。とにかく、使いこなしの第一歩は、使いやすい環境作りからだ。ここでは、比較的簡単にできるデスクトップのカスタマイズをしてみることにする。これにより、少しでもLinux環境に愛着が湧いてくれればと思うのであった。

●GNOMEコントロールセンターを使う

アプリケーションランチャから起動できるGNOMEコントロールセンターは、GNOMEのいろいろな設定を変更するためのツールだ（**画面1**）。このツールを使えば、デスクトップのカスタマイズもできるのだ。

スクリーンセーバは初期設定では、ランダム表示されるようになっている。もし、その中で気に入ったものがあれば、設定を変更してお気に入りのスクリーンセーバに固定してしまおう。左の一覧から［デスクトップ］—［スクリーンセーバー］を選ぶ。［デスクトップ］の先頭に「＋」マークが表示されてる時は、［スクリーンセーバー］が隠れているので、その「＋」のところをクリックしよう。

［スクリーンセーバー］を選択すると右側に設定画面が表示される（**画面2**）。利用可能なスクリーンセーバの種類がリストボックスに表示されているので、目的のものを選ぶ。デモを表示させることもできるので一通り見てみるのもいいだろう。スクリーンセーバ実行開始までの時間や、スクリーンセーバ解除時のパスワードを要求するかどうかなども設定できる。また、パワーマネージメントを使用すると、長時間席を空けた時などに、モニタの主電源が切れて節電になる。設定したら［適用］ボタンをクリックしよう。

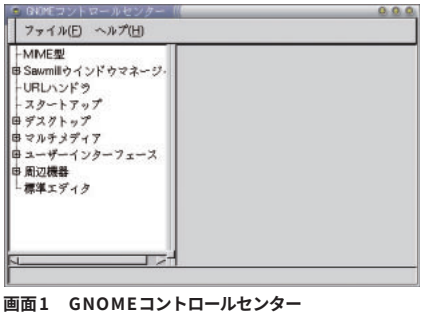
ダイアログや、ボタン、テキストボックスなどのデザインを変更するのが［テーマセレクタ］だ。使用可能なテーマのリストからテーマを選ぶと、試写でどう変わるのかを見ることができる。また、試写では実際にクリックしたり選択したりもできるので試してみよう。「Expensive」などはその名の通り高価なイメージだ（**画面3**）。

［バックグラウンド］はデスクトップの背景色や、壁紙を設定する。初期設定ではTurboLinuxの壁紙が貼られているが、［参照］ボタンをクリックして、画像ファイルを指定すれば好みのものに変更できる（**画面4**）。ちなみに、壁紙がデスクトップより小さい場合や、壁紙を「なし」に設定した場合に、背景の色指定が生きる。こうしてカスタマイズして**画面5**のようなデスクトップが完成した。

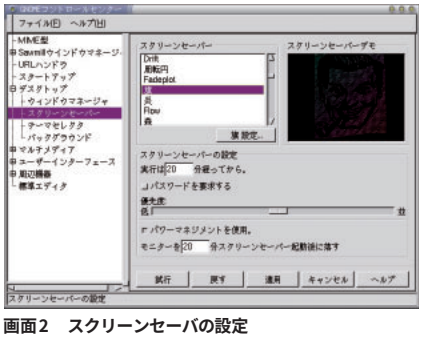
●メインメニューの変更

メインメニューの並びが使いにくかったり、項目の追加／削除をしたい場合は、GNOMEメニューエディタを使おう（**画面6**）。メインメニューの［デスクトップ設定］—［メニューエディター］を選択する。ただし、システムメニューの変更はrootでないとできない。一般ユーザーに許される変更はユーザーメニューのみに限られている。

さて、メニューの並べ替えは簡単だ。左側のメニュー項目のリスト内でドラッグ＆ドロップすればいい。サブメニューの下の項目も同様に並べ替えられ

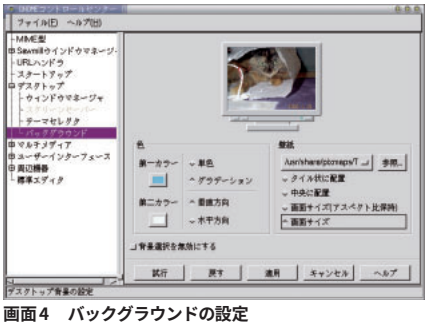


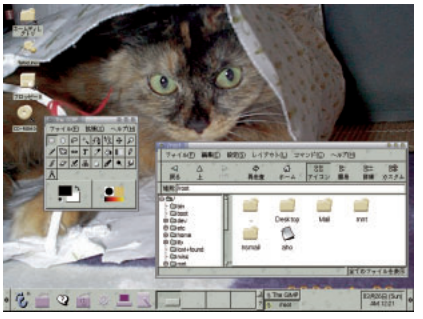
画面1　GNOMEコントロールセンター



画面2　スクリーンセーバの設定

画面3　テーマセレクタの設定


画面4　バックグラウンドの設定


画面5　カスタマイズしたデスクトップ


画面6　GNOMEメニューエディタ

る。サブメニューの項目をメインメニューに格上げ、あるいはその逆もできる（**画面7**）。

項目を選択しておき、［削除］ボタンをクリックすれば、削除される。

メニューに新規項目を追加する場合は、ユーザーメニューに追加するようにしよう。左のリストでユーザーメニューを選択しておき、そこで［新規項目］ボタンをクリックする。右側に「無題」の設定が表示される。メニューに表示される名前と、実行するコマンド、アイコンを選択すればいい（**画面8**）。

サブメニューにしたい場合は、［新規サブメニュー］ボタンをクリックする。「新規フォルダ」の設定が表示されるので、メニューの名前とアイコンを入力しよう。あとはドラッグ&ドロップで既存の項目を追加するか、サブメニューを選択しておいて新規項目を追加しよう（**画面9**）。

●パネルのカスタマイズ

パネルにはさまざまなアプレットが乗っている。アプリケーションランチャは便利だが、自分がよく使うアプリケーションがなかったり、たとえ追加できたとしても、アイコンが大きくて、たくさん並べるのは難しそうである。それに、たくさんウィンドウを開くと［時刻と日付］アプレットは画面から追い出されてしまうし…。そんな悩みを解決すべく、パネルのカスタマイズをしてみよう。

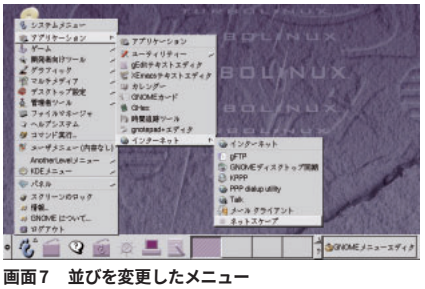
まずは、パネルにランチャやアプレットを追加する方法からだ。パネルによく使うアプリケーションを登録するには、いくつか方法がある。メインメニューから起動しているアプリケーションをパネルに追加したければ、そのメニュー項目をパネルにドラッグ&ドロップすればいい。メニューにないアプリケーションを追加するなら、メインメニューの［パネル］—［新しいランチャを追加］を使う。設定項目はメニュー項目の追加と同様である。

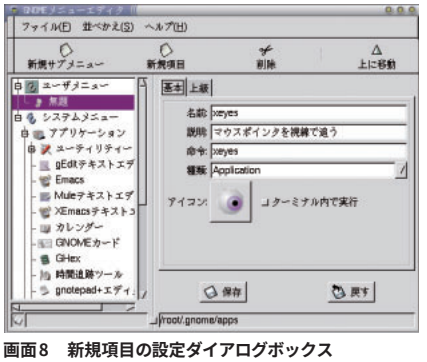
また、パネル上で機能するアプレットも追加できる。［パネル］—［アプレットを追加］のサブメニューには、各種アプレットが用意されているので、好みに応じて追加できる。いくつかアプレットを追加したのが**画面10**だ。CPUの負荷やメモリ利用状況などはリアルタイムに表示が変わるので楽しい。

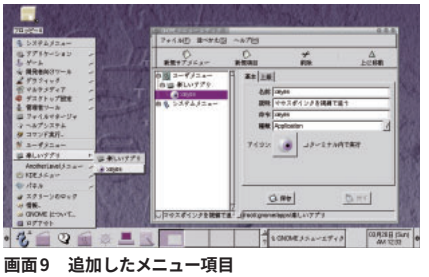
追加したランチャやアプレットはマウスの右クリックで［パネルから削除する］を選択すれば削除できる。

パネルは画面の下にしか表示できないわけではない。画面の上下左右どこにでも配置できる。また、ふだん使わないときには、隠しておくこともできる。［パネル］—［このパネルのプロパティ］で画面内での配置や、隠すかどうかの設定ができる（**画面11**）。

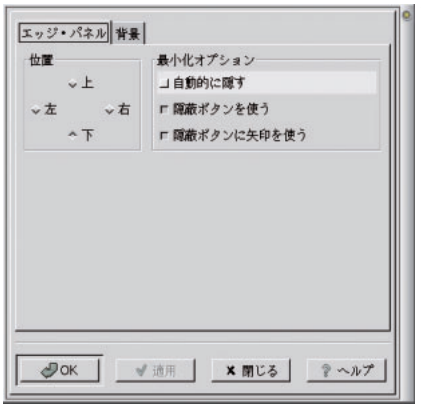
また、別のパネルを作ることもできる。［パネル］—［新規パネルを作成］—［エッジパネル］、または［コーナーパネル］を選択する。エッジパネルとは、画面の端から端まで使うパネルで、コーナーパネルとは画面の角から使っている分しか表示されないパネルだ（**画面12**）。新しいパネルにもメインメニューやアプレット、ランチャなど自


画面7　並びを変更したメニュー

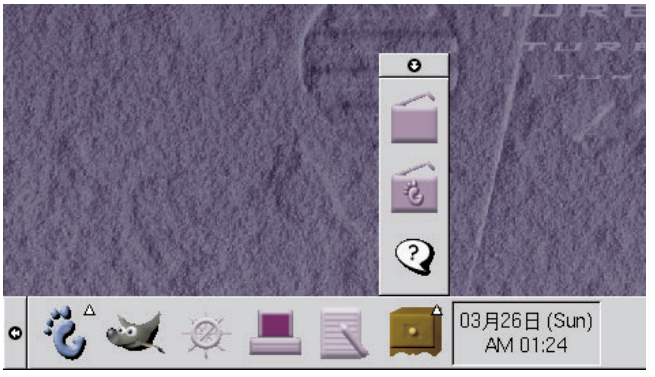

画面8　新規項目の設定ダイアログボックス


画面9　追加したメニュー項目

画面10　追加したランチャとアプレット



画面11
［このパネルのプロパティ］
ダイアログボックス



画面13　引き出しアプレット

由に追加できるので、うまく使い分けよう。

追加したパネルは、メインメニューの［このパネルを削除する］で削除できる。追加したパネルの上でマウスを右クリックすれば、メニューが表示される。もとからあるパネルを削除することはできない。

アプリケーションランチャが増えてきたら、引き出しアプレットを使って整理することも可能だ。引き出しは、パネルから直角に伸びるアイコン収納場所だ（**画面13**）。［パネル］—［引き出しを追加］を選択するとパネルに引き出しが追加される。アイコンを引き出しに収納するには、アイコンをドラッグ＆ドロップすればいい。中央ボタンなら移動、右ボタンならコピーだ。

テキストファイルの表示と編集

Linuxでは設定ファイルの多くがテキストで記述されている。したがって、テキストファイルの編集さえできれば、システム設定のうち、かなりの部分が変更できるのだ。テキスト編集の知識とroot権限があれば、もはやLinuxはあなたの思い通りなのだ（ともいい切れないが…）。もちろんファイルの記述内容についての知識は必要だが、それは今後の本誌の記事でもいろいろと紹介されるはず（たぶん）。また、この特集でも、いくつか設定ファイルの書き換えをやってみるので、挑戦してほしい。

ここでは、そうした時の役に立つ、テキストファイルの表示と編集の方法について解説しておこう。

●テキストファイルの表示

テキストファイルの内容を確認したい場合や、文書を読むだけならば、「テキストファイルビューア（gless）」を使うと便利だ（**画面14**）。起動方法は、メインメニューから、［プログラム］—［ユーティリティー］—［テキストファイルビューア］を選択する。

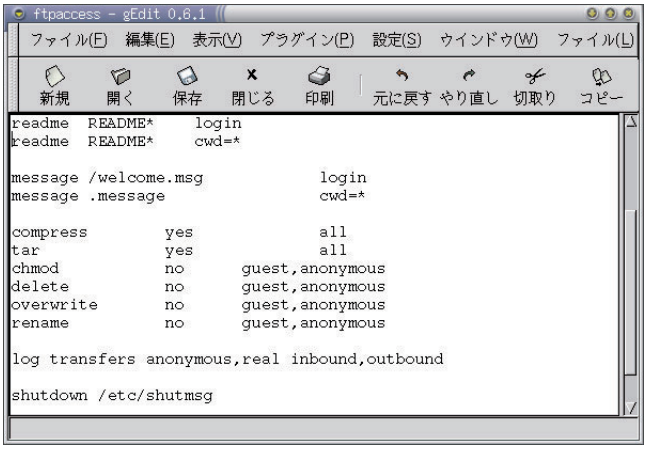
表示させるファイルは、メニューから［ファイル］—［開く］で指定してもいいし、目的のファイルを直接ファイルマネージャからglessのウィンドウへドラッグ＆ドロップしてもいい。テキストが長い場合は、矢印キーを使って表示内容を上下にスクロールさせる。

●テキストファイルの作成と編集

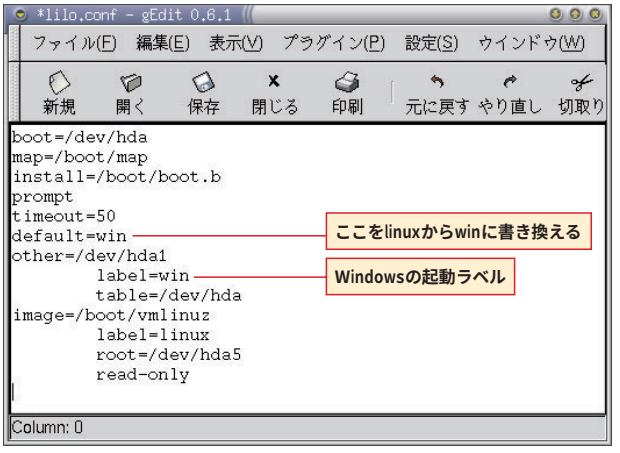
ファイルの作成や編集にはgEditを

画面12　コーナーパネルとエッジパネル

画面14　テキストファイルビューア（gless）



画面15　gEditテキストエディタ



画面16　lilo.confを書き換えた

使おう（**画面15**）。起動方法は、［プログラム］―［アプリケーション］―［gEditテキストエディタ］を選択する。基本的な操作方法は、Windowsのメモ帳とそう大して変わらないが、メモ帳よりは多機能だ。

起動すると、新規文書が作成できる状態になっている。ここで文字を入力して、［ファイル］―［名前を付けて保存］を選べばテキストファイルが作成できる。

すでにどこかにあるテキストファイルを開いて編集するには、［ファイル］―［開く］でファイルを指定するか、ファイルマネージャからgEditのウィンドウへ、編集したいテキストファイルのアイコンをドラッグ&ドロップする。

デフォルトでWindowsが起動するように設定する

さっそくだが、テキスト編集を覚えたので設定変更に挑戦してみよう。起動時の「boot:」プロンプトで、何も指定しない場合には、Windowsが起動するように設定するのだ。この作業は重要なファイルの書き換えを行うので、rootで行う必要がある。ミスは許されないので、くれぐれも慎重に！

rootでログインし、gEditで設定ファイル/etc/lilo.confを開いてみよう。lilo.confはLinuxの起動に関わる各種設定が記述されているファイルだ。ファイルの記述のうち、

```
default=linux
```

というところを、

```
default=win
```

と書き換えよう（**画面16**）。ここで設定する「win」というのは、インストール時に指定したWindowsの起動ラベルであることに注意。ほかの部分は変更しないように注意しよう。正しく書き換えたら、［ファイル］―［保存］で保存する（rootでないと上書きできない）。

保存できたら、次にliloコマンドを実行する（これもrootでないと実行できない）。まず、ktermを起動する。そしてbashに次のように入力してEnterキーを押す。

```
# /sbin/lilo
```

画面17　ktermで/sbin/liloを実行

liloコマンドが正しく設定した旨メッセージが表示される（画面17）。これで、変更したlilo.confの設定が次回起動時から機能する。したがって、次回起動時にLinuxを使いたければ、「boot：」プロンプトに「linux」と入力する。何も入力しないでいると、自動的にWindowsが起動する。

日本語の入力

テキストファイルは作成できるようになったが、まだ日本語の入力方法がわからない。ここでは日本語の入力方法を説明しよう。

Linuxで日本語を入力する場合も、Windowsと同様、かな漢字変換プログラムを使う。Linuxで利用できるかな漢字変換プログラムの代表的なものには、ATOK、VJE、Wnn、Canna、sj3などがある。このうち、ジャストシステムのATOKとバックスのVJEについては、Windows版が有名なので、ご存じの方も多いだろう。基本的にはWindowsで利用されている同名のプログラムをLinuxに移植しているものだ。それに対して、Wnn（「うんぬ」と読む）、Canna（「かんな」）、sj3は、それぞれ、オムロン、日本電気、ソニーの各社が、自社のワークステーションに

搭載していたかな漢字変換プログラムで、生っ粋のUNIX育ちといってよい。ATOKやVJE、Wnnの新しいバージョンは市販の製品だが、Wnnのバージョン4、FreeWnn、Canna、sj3はソースがフリーで公開されているため、Linuxを含む、UNIX系OSで幅広く利用されている。

ところで、日本語入力をする際に知っておいてもらいたいことは、かな漢字変換プログラムを利用しても、入力対象となるアプリケーションが日本語に対応していなければ、正しく入力することができないということだ。これは何もLinuxに限った話ではないが、Linuxのアプリケーションは世界中で膨大な数が開発されているため、必ずしも日本語での利用を考慮して開発されているとは限らないのだ。最近でこそ、国際化という観念が開発者の間にも広まっており、日本語の利用を考慮したアプリケーションも増えてきてはいるが、まだまだ、日本語入力に対応しないアプリケーションも多いという点については、あらかじめ承知しておいたほうがいいだろう。

先ほど使ったgEditは日本語入力に対応しているので、ここで解説するCannaの基本的な使い方は、とりあえずgEditで試してみるといいだろう。

●Cannaを使えるようにする

初期設定ではFreeWnnが使われるようになっている。これはログイン時に実行される設定ファイルの記述によるものだ。TurboLinux Workstation 日本語版 6.0の初期設定では、ATOK→Wnn→Cannaの順番に探していき、先に見つかったものが使われる。製品版パッケージでは、ATOK→Wnn6→FreeWnn→Cannaの順になる。ノンサポート版では、FreeWnn→Cannaの順になるのだ。もちろんFreeWnnが先に見つかる。

別にFreeWnnでも困るわけではない。かな漢字変換の効率については、よほど使い込まない限り、FreeWnnもCannaもさして変わらないだろう。さすがにATOKやWnn6のような最新鋭の機能は持ち合わせないので、多少歯がゆい局面があるかもしれない。しかし、それより何より、とりあえず使うには、Cannaはキー操作が簡単なのである。

では、ここで、かな漢字変換プログラムの検索順序を変更しよう。

/etc/X11/xinitにあるxinitrcというファイルを自分のホームディレクトリにコピーしよう（移動しないように注意）。コピーしたら、ファイル名をxinitrcから「.xinitrc」に変更する（頭にピリオドを付ける）。ホームディレクトリにコピーしたxinitrcのアイコンをいったん選択し、さらにファイル名をクリックして、名前を変更すればいい。頭にピリオドが付いたファイル名を持つファイル（通称ドットファイル）は、ふだんは表示されない。ファイルマネージャの［設定］―［設定］で表示されるダイアログボックスの［隠しファイルを見る］オプションを選択すると表示される（意外と隠しファイルが多いことに驚くかもしれない）。

そうしたら、gEditでその.xinitrcを開く。**リスト1**に示した7行のうち、①の部分4行と②の部分3行を入れ替える。それぞれ「elif」で始まる行から「export」の行までがひとかたまりであることに注意。入れ替えたら保存する。

この.xinitrcはログイン時に読み込まれ実行される。したがって、これを有効にするには、ログインし直すのが手っ取り早い（本当は方法がなくもないが…）。

●Cannaの起動と日本語の入力

まずはCannaを起動しよう（実際には、Cannaの本体はすでにバックグラウンドで起動して待機しており、日本語入力モードに移行するだけ）。Shift－スペースを押す。現在のカーソル位置に［ あ ］と表示されたはずだ（**画面18**）。この表示が日本語入力モードの目印である。この状態でキーを押すと日本語が入力できる。もう一度Shift－スペースを押してみよう。今度は［ あ ］の表示が消えたはずだ。こ

リスト1　ホームディレクトリの.xinitrc

```
elif [ -f /var/lock/subsys/jserver.wnn6 ] \
  || [ -f /var/lock/subsys/jserver ]; then
    /usr/X11R6/bin/kinput2 -wnn -jserver localhost &
    export XMODIFIERS="@im=kinput2"
elif [ -f /var/lock/subsys/canna ]; then
    /usr/X11R6/bin/kinput2 -canna &
    export XMODIFIERS="@im=kinput2"

        ↓①と②を入れ替えた結果

elif [ -f /var/lock/subsys/canna ]; then
    /usr/X11R6/bin/kinput2 -canna &
    export XMODIFIERS="@im=kinput2"
elif [ -f /var/lock/subsys/jserver.wnn6 ] \
  || [ -f /var/lock/subsys/jserver ]; then
    /usr/X11R6/bin/kinput2 -wnn -jserver localhost &
    export XMODIFIERS="@im=kinput2"
```

①　②　②　①

れで、日本語入力モードから抜けたことになる。このようにShift－スペースで日本語入力モードが切り替わるのだ。

では、いよいよ入力してみよう。初期設定ではローマ字かな漢字変換になっている。日本語入力モードに切り替えて、順にキーをM、E、M、O、R、Iと入力してみる。するとローマ字読みの「めもり」が画面に表示される。

ここで、スペースキーを押してみよう。「めもり」が「メモリ」に変換されたはずだ（場合によっては「目盛り」など別の綴りかもしれない）。Cannaの初期設定では変換キーがスペースに割り当てられている。MS-IMEやATOKユーザーにはわかりやすいだろう。また、［あ］の表示が［漢字］に変わっているが、これは漢字変換中であることを示している。

もし目的の語がこの「メモリ」ならば、そこでEnterキーを押して確定する。目的の語が異なる場合は、もう一度スペースキーを押そう。すると、変換候補の一覧が表示される（**画面19**）。この一覧からは、カーソルキーで候補を選択することができる。候補から選んだらEnterキーで確定する。

ひらがなのままでよければ、変換キーを押す前に確定すればいい。カタカナや英数に変換するには、変換キーの代わりに↑キーか↓キーを押す。↑キー押すたびに、半角英数→全角英数→半角カナ→全角カナ→ひらがなという順（↓は逆順）に変換されるので、目的の字種になったところで確定キーを押そう。

●注目文節の移動と範囲変更

変換時にCannaは自動的に文節を解釈するが、それが必ずしもユーザーの意向と一致するとは限らない。そういう時は、注目文節を調整する。「庭には二羽鶏がいる」と変換したいとしよう。「にわにはにわにわとりがいる」と入力して変換する。すると、「庭に埴輪鶏が入ル」と変換される。ここで「庭に」が反転表示されているが、これが注目文節である（**画面20**）。ここでは「庭には」と次の「は」も同じ文節と解釈させたいので、Shift－→キーを押す。すると「庭には」が文節として解釈される。注目文節の範囲変更にはShift－矢印キーを使うのだ（**画面21**）。さて、次の文節に移動しよう。文節間の移動は矢印キーを押す（Shiftは押さない）。→キーで次に移動する。「二羽」としたいところが「庭」になっている。ここで変換キーを押して候補一覧を見ても、「二羽」という漢字は出てこない（**画面22**）。変換候補一覧表示を取りやめるのはCtrl－Gだ（候補一覧表示以外の時に押すと、全入力が変換前に戻るので注意）。注目文節が「にわ」に戻っている。そこで、Shift－←キーを押そう。文節が縮まり「に」と「わ」に分かれ「二」と「和」に自動的に変換されている（**画面23**）。「二」はOKなので、→キーで「和」に移動し変換キーで候補一覧を表示し「羽」に変換する。→キーで最後の「入ル」に移動し、↓キーで「いる」に変換する（**画面24**）。

●その他の文字の入力

記号の入力には、Insertキーを使う。Insertキーを押すと記号候補一覧が表示される。それ以外にも、部首入力やコード入力などの方法があるが、ここでは割愛する。詳しくはCannaのWebサイトを参照してほしい。

Cannaの基本的な機能とキー操作を**表1**にまとめておく。

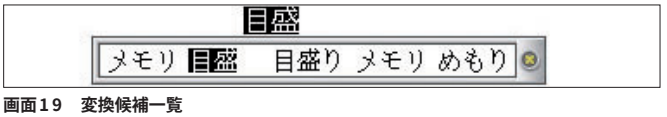
CannaのWebサイト

`http://www.nec.co.jp/japanese/product/computer/soft/canna/`

インターネットアクセス

今や、コンピュータたるもの、イン

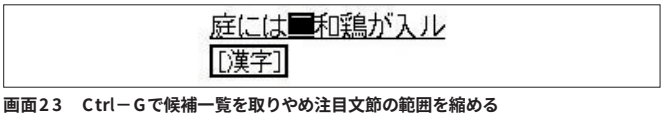
画面18　日本語入力モード（［あ］と表示される）



画面19　変換候補一覧

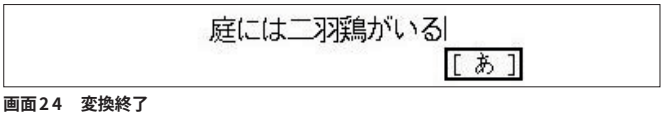
画面22　目的の文字が候補にない

画面20　注目文節



画面23　Ctrl－Gで候補一覧を取りやめ注目文節の範囲を縮める

画面21　注目文節の範囲を広げた



画面24　変換終了

機能	入力キー
起動／終了	Shift－スペース
変換／候補一覧	スペース
確定	Enter
字種変更	↑、↓
記号の入力	Insert
注目文節の移動	←、→
注目文節の範囲変更	Shift－←、Shift－→
取りやめ	CTRL－G

表1　Cannaの基本的な機能とキー操作

ターネットに接続できなければお話にならない。とくにLinuxの情報はインターネット上のあちらこちらに散らばっている。Linux用プログラムの多くも、インターネット上から無料で手に入れることができる。Linuxこそインターネットに接続してその面白味が何倍にも膨れ上がるOSなのだ。

と、意気込んでみたが、サーバ構築とかは難しいので、ここでは、Webブラウジングと、電子メールの読み書きという基本中の基本について説明する。これで、Linuxの情報収集＆交換に励もうではないか。

●PPPの設定

LinuxをモデムやTA（ターミナルアダプタ）経由でインターネットに接続するには、gnome-pppを使うのが簡単だ。［インターネット］－［PPP dialup utility］を選択する。するとgnome-pppが起動する（**画面25**）。

まずは、接続情報を設定しよう。プロバイダのアクセスポイントの電話番号や、接続用のユーザー名、パスワード、ネームサーバアドレスなどが必要になるので、手元に用意しておこう。メニューから［Account］－［新規］

画面25　gnome-ppp

画面26　［ダイアル］タブでの設定

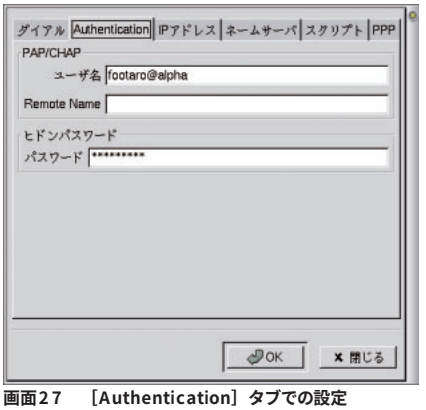
を選択する。設定ダイアログボックスが表示される。［ダイアル］タブでユーザー名（これは、ユーザー名ではなく、実際にはプロバイダ名でいい）と電話番号を入力する。

モデムのところには、モデムやTAが接続されているポートを指定するのだが、ここでは、内蔵モデムではなく、シリアルポートに接続したモデムやTAを使うので「/dev/ttyS0（COM1）」を選択しておこう（**画面26**）。

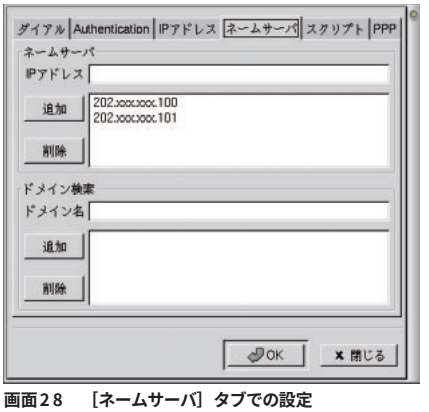
［Authentication］タブで、プロバイダから割り当てられた接続ユーザー名、パスワードを入力する（**画面27**）。［ネームサーバ］タブでは、プロバイダのネームサーバのIPアドレスを入力する（**画面28**）。以上で設定完了だ。［OK］ボタンをクリックしてダイアログボックスを閉じる。

ダイアログボックスを閉じると、［ダイアル］タブで設定したプロバイダ名が表示されているはずだ。では、接続してみよう。接続ボタンをクリックする。接続ボタンの下に現在の状態が表示される。「Dialing」→「Connection established」となれば接続完了である（**画面29**）。

接続を終了するには、［切断］ボタン（［接続］ボタンの名称が変わる）をクリックする。



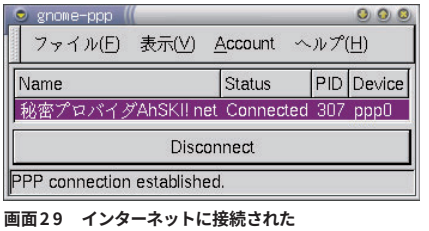
画面27　［Authentication］タブでの設定



画面28　［ネームサーバ］タブでの設定

●Webページのブラウズ

WebをブラウズするにはNetscape CommunicatorのNavigatorを使う。まずはプロバイダに接続しておく。パネルに「ネットスケープ・ブラウザ」というアプリケーションランチャがあるのでそれをクリックすれば起動する。ユーザーごとに、初めて起動する時には、使用許諾についての説明や、ディレクトリを作成する旨のメッセージが表示される。［OK］ボタンをクリックしていくと、Navigatorが起動する（**画面30**）。



画面29　インターネットに接続された

画面30　Netscape Navigator

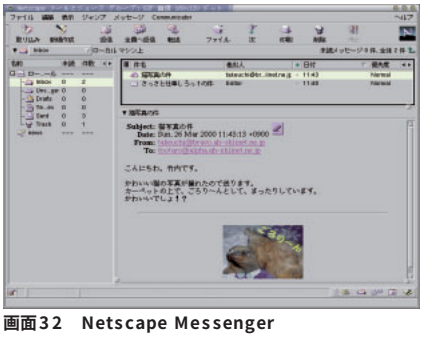
画面31　ウェブコントロールアプレット

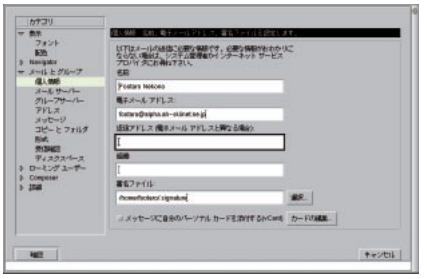
WindowsでInternet Explorer(IE)を使っていれば、基本的な使い方はわかるのではないだろうか。URLをテキストボックスに指定すれば、そのページが表示される。ただし、IEで「お気に入り」と呼ばれている機能は、Navigatorでは「ブックマーク」と呼ばれていたりして、若干異なる部分もあるので注意。

ブラウザが動くようになったら、パネルから直接URLを指定できるウェブコントロールアプレットを追加してしまうのも便利だ（**画面31**）。ここにURLを入力すると、自動的にNavigatorが起動して、そのページを表示してくれる。

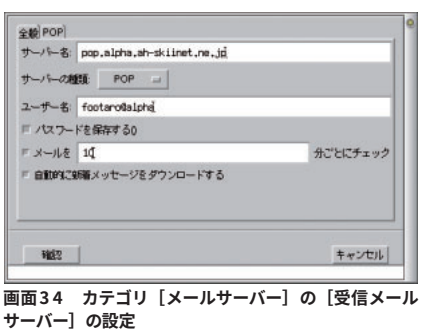
●メールの読み書き

Netscape Communicatorには、Navigator以外にも、Messengerというメールクライアントが装備されている。Navigatorのメニューから［Communicator］—［Messengerメールボックス］を選択すると、Messengerが起動する（**画面32**）。


画面32　Netscape Messenger


画面33　カテゴリ［個人情報］の設定

Messengerでメールを読み書きするには、メールサーバや、メールアドレスなどを設定しなければならない。初めて起動した時には、その設定ダイアログボックス（［編集］—［設定］を選択した時と同じもの）が表示される。［カテゴリ］から［個人情報］を選択し、右側に表示される［電子メールアドレス］の欄に自分のメールアドレスを入力する（**画面33**）。そして、［カテゴリ］から［メールサーバー］を選択する。［受信メールサーバー］のリストボックスにある「pop」を選択して［編集］ボタンをクリックする（**画面34**）。［全般］タブのサーバ名のところに、プロバイダのPOPサーバの名前を入力する。ユーザー名の欄にPOPサーバに接続するためのユーザー名を入力する。［パスワードを保存する］オプションをチェックしておくと、最初の1回だけしかPOPサーバのパスワードを問い合わせないので、煩わしい思いをしなくて済む。［POP］タブでは、サーバにメッセージを残すかどうかのオプションが選択できる。［確認］ボタンをクリックして、［メールサーバー］のダイアログボックスに戻る。次に［送信メール（SMTP）サーバー］にSMTPサーバ名を、そして［送信メールサーバーのユーザー名］にSMTPサーバに接続するためのユーザー名をそれぞれ入力する（**画面35**）。

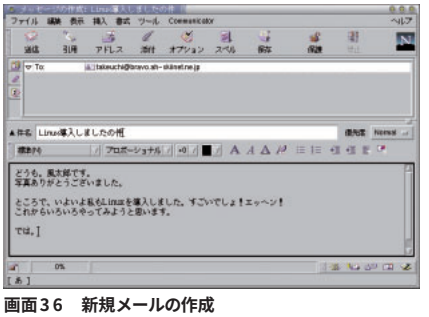


画面34　カテゴリ［メールサーバー］の［受信メールサーバー］の設定


画面35　カテゴリ［メールサーバー］の［送信メールサーバー］の設定

以上の入力が終わったら、［確認］ボタンをクリックして、Messengerに戻ろう。

Messengerの［取り込み］ボタンを押すとPOPサーバからメールを受信する。メールを出したいときは、［新規作成］ボタンをクリックすればいい（**画面36**）。


画面36　新規メールの作成

覚えておきたい必須コマンド30

これだけ使えれば一人前のLinuxer?!
ファイル操作、管理コマンド、viエディタ、bashの使い方を完全マスター！

Windowsを使ううえでコマンド画面を使うことはそう多くないだろう。DOSの時代を経験したユーザーなら、WindowsのいわゆるDOS窓を使いこなしているかもしれないが、最近PCを使うようになったユーザーはまず使わない。Macintoshにいたっては、コマンド画面そのものがないのだ。

一方、LinuxやUNIXではコマンドを多用することが多い。このご時世になんて前時代的なと思うなかれ、UNIXの世界にはDOS環境のCOMMAND.COMなどとは比較にならないほど高機能なシェル、そして、ひとつひとつは単純な機能しか持たないが、組み合わせることで強大な力を発揮する数々のコマンドが用意されている。これらを使いこなせば、GUIでは絶対にできないオペレーションも可能になる。

しかし、そのためにはまず個々のコマンドを知らなければならない。そこで、本特集では使用頻度が高いコマンドを中心に基本的な使い方を解説しよう。

これを読めばあなたはもうコマンドの使い手だ。

ファイルの基本操作

Linux操作の基本、ファイルやディレクトリ操作をコマンドでできるようになろう。

文：中野 賢
Text : Ken Nakano

ディレクトリ位置の確認

コンソールから、ユーザー名とパスワードを入力しログインすると、

```
USER$
```

のようなプロンプトが表示される。

この状態では、ユーザーは**ホームディレクトリ**（ログインディレクトリ）にいる。そして、現在、ユーザーのいるディレクトリのことを**カレントディレクトリ**（作業ディレクトリ）と呼ぶ。つまり、ログインした直後の状態では、ホームディレクトリがカレントディレクトリである。

カレントディレクトリの位置を確認するには、pwdコマンドを実行する。コマンドラインから、

```
USER$ pwd
```

とするだけだ。すると、

```
/home/ken-na
```

のように現在のディレクトリ位置が表示される。ログインした直後では、自分のホームディレクトリがカレントディレクトリになっている。

Linuxシステムは、**図1**のような木構造のディレクトリの上に構築されている。このうち、最上位のディレクトリは根っ子という意味で**ルートディレクトリ**と呼ばれる。すべてのディレクトリはここから始まる。

ディレクトリの中にあるディレクトリは**サブディレクトリ**と呼ばれる。たとえば、homeやtmpなどは、ルートディレクトリのサブディレクトリである。逆に、homeやtmpから見て、ルートディレクトリは**親ディレクトリ**と呼ばれる。同様に、ken-naディレクトリはhomeディレクトリのサブディレクトリであり、ken-naディレクトリの親ディレクトリがhomeディレクトリである。

ディレクトリ階層は各ディレクトリを/で区切って表記する。ただし、ルートディレクトリには名前がないので、単に/とだけ表記する。先頭が/で始まると、それはルートディレクトリからのディレクトリ位置であることを示している。

先ほど、pwdコマンドで表示された/home/ken-naディレクトリは、

ルートディレクトリ内の
homeディレクトリ内の
ken-naディレクトリ

を意味している。先頭に/を付けずに、home/ken-naと指定すると、

カレントディレクトリ内の
homeディレクトリ内の
ken-naディレクトリ

という意味になる。

/がディレクトリの区切りとしての意味と、ルートディレクトリ自身を示すという2つの意味を持つことを覚えておこう。

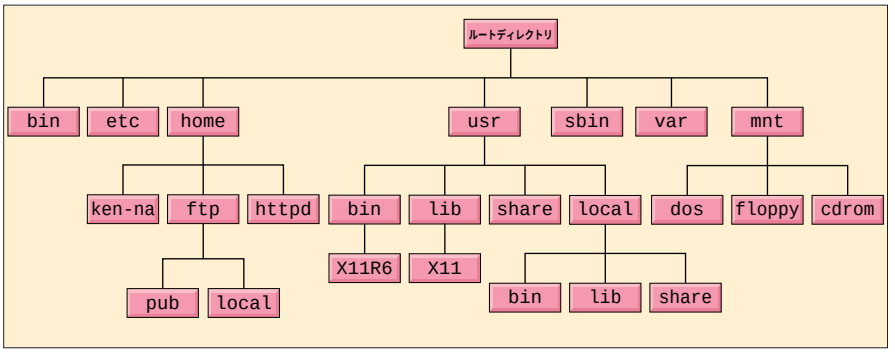


図1　ディレクトリ・ツリー

ディレクトリの移動

ディレクトリを移動するにはcdコマンドを使う。引数には、移動したいディレクトリの名前を指定する。たとえば、/usrディレクトリに移動したいときは、次のようにする。

```
USER$ cd /usr
```

実行すると、カレントディレクトリが/usrディレクトリに移動しているはずだ。pwdコマンドで確認してみよう。

```
USER$ pwd
/usr
```

cdコマンドに移動先を指定しないで実行すると、ホームディレクトリに移動する。

```
USER$ cd
USER$ pwd
/home/ken-na
```

ここで、lsコマンドにaオプションを指定して実行してみよう。lsコマンドはディレクトリの内容を表示するためのコマンドだ。

```
USER$ ls -a
.              .emacs     .xemacs
..             .emacs20   Desktop
.Xdefaults     .gimp      howto
.bash_history  .gtk       memo.txt
.bash_profile  .kde
.bashrc        .kderc
```

表示されたディレクトリの中に、ドット（.）と2つのドット（..）がある。「.」はカレントディレクトリを指し、「..」は親ディレクトリを指している。したがって、

```
USER$ cd .
```

は、カレントディレクトリへの移動を実行している。もちろん、ディレクトリ位置は変わらない。

```
USER$ cd ..
```

とすれば、親ディレクトリに移動する。pwdコマンドを実行してみよう。

```
USER$ pwd
/home
```

カレントディレクトリが/homeディレクトリになったことを確認できた。再び、lsコマンドでディレクトリの内容を表示してみよう。

```
USER$ ls -a
.  ..  ftp  httpd  ken-na  samba
```

当然、ここにも.と..がある。さらに親ディレクトリに移動してみる。カレントディレクトリはルートディレクトリになる。

```
USER$ cd ..
USER$ pwd
/
USER$ ls -a
.   boot home       mnt  sbin var
..  dev  lib        proc tmp
bin etc  lost+found root usr
```

ルートディレクトリにも「.」と「..」が存在する。カレントディレクトリを意味する.は理解できるにしても、親ディレクトリを意味する「..」があるのはおかしいような気がするかもしれない。これは、Linuxのファイルシステムの性質上、すべてのディレクトリが「.」と「..」を持っているという理由で表示されているだけだ。ルートディレクトリの上には何もない。

今度は、ディレクトリ階層を降りていってみよう。サブディレクトリに移動するときには、先頭に/を付けない。カレントディレクトリがルートディレクトリの場合は、/を付けても付けなくても同じだが、ルートディレクトリ以外の場所では指定している位置が異なってしまう。

```
USER$ cd usr
USER$ cd X11R6
USER$ cd lib
USER$ cd X11/locale/C
USER$ pwd
/usr/X11R6/lib/X11/locale/C
```

ディレクトリが存在することを知っていれば、複数のディレクトリ名を/で区切って指定してもかまわない。だが、指定したディレクトリが存在しない場合は、次のようなエラーとなる。

```
USER$ cd X11/locale/japanese
bash: X11/locale/japanese: No such
file or directory
```

Column

X Window Systemでのカレントディレクトリ

コンソールやtelnetでログインしたときには、カレントディレクトリは必ずホームディレクトリになっている。

しかし、X Window System上での、ktermなどの端末エミュレータを開いたときは、ホームディレクトリであるとは限らない。

たとえば、ルートウィンドウから起動したktermではホームディレクトリがカレントディレクトリになっている。けれども、すでに開いているkterm上から、

```
USER$ cd /tmp
```

として、/tmpディレクトリに移動後、

```
USER$ kterm &
```

として起動すると、そのktermのカレントディレクトリは/tmpになっている。

同一のディレクトリで複数の端末エミュレータを開いて作業をしたい場合は、最初に移動しておいて、そこでktermやXアプリケーションを実行すると便利だ。

ディレクトリ	目的
bin	プログラムファイル
boot	システムの起動に用いるファイル
dev	デバイスファイル
etc	システム設定ファイル
home	各ユーザーのホームディレクトリ
mnt	ファイルシステムを一時的にマウントするのに使う
proc	プロセステーブル
root	rootユーザーのホームディレクトリ
sbin	システム管理用途のプログラムファイル
tmp	一時的なファイルの置き場所に使う
usr	ユーザーが日常利用するプログラムやライブラリ
var	ログファイルやスプールなど

表1　Linuxのディレクトリ分類
Linuxのディレクトリツリーは、おおよそ、次のように分類されている。

X11が存在しないときでも、localeでも、japaneseでも、どこかのディレクトリが存在していなければエラーとなる。ディレクトリがあるかどうか、不安な場合は少しずつ移動するほうがよいだろう。

先頭を/で始めない指定での移動では、カレントディレクトリを基準にしている。このようなパス表記を**相対パス**と呼ぶ。対して、先頭が/で始まる、pwdの表示のようなパス表記を**絶対パス**と呼ぶ。

コマンドラインでの操作は、オプションでファイルやディレクトリの位置を指定することが多い。目的のファイルやディレクトリの記述を短くすれば、それだけキー入力を少なくすることができる。特に、「.」と「..」は頻繁に用いるので、いろいろと試して慣れてほしい。

ディレクトリ内容の表示

先ほど、ディレクトリの内容を表示するのにlsコマンドを用いた。ここで、lsコマンドについてもう少し詳しく説明しよう。

lsは引数としてディレクトリを指定しないと、カレントディレクトリの内容を表示する。

```
USER$ ls
Desktop  howto  memo.txt
```

ただし、ドットで始まる名前は表示しない。すべてのファイルを表示するには、aオプションを付ける。

親ディレクトリの内容を表示するのも「..」を用いれば簡単だ。

```
USER$ ls -a ..
.  ..   ftp  httpd  ken-na  samba
```

これまでの実行例では表示される名前がディレクトリなのかファイルなのかがわからない。Fオプションを指定すると、種類に応じて名前のうしろに/や@が付く。/が付いているのがディレクトリで、@が付いているのはシンボリックリンクのファイルを示している。記号の付いていないファイルは、通常のファイルだ。

```
USER$ ls -aF
./              .emacs     .xemacs
../             .emacs20   Desktop/
.Xdefaults      .gimp/     howto@
.bash_history   .gtk/      memo.txt
.bash_profile   .kde/
.bashrc         .kderc
```

ファイルサイズや作成した日付を表示したい場合は、lオプションを用いる。すべてのファイルを表示するにはaオプションも同時に指定する（**図2**）。

lオプションでは、ファイルの種類と許可、リンク数またはディレクトリ数、所有者、グループ、ファイルサイズ、最終修正時刻、名前が出力されている。

ファイルの種類は、通常のファイル(-)、ディレクトリ（d)、シンボリックリンク（l)、デバイスファイル（bまたはc）が区別される。

許可はパーミッションとも呼ばれ、所有者、グループ、その他のユーザーに対して、それぞれ読み取り、書き込み、実行の許可の有無を示している。たとえば、

```
rw-rw-r--
```

のようになっているファイルは、所有者とグループは読み込みと書き込みができるが、その他のユーザーは読み込みしかできないことを意味している。

```
rwxr-xr-x
```

は、すべてのユーザーが読み込みと実行をすることができる。そして、所有者は書き込むことも可能である。

ディレクトリの場合、許可の意味がファイルとは少し異なる。まず、読み取り許可はディレクトリの内容を読み取れるかどうかという意味になる。ディレクトリの読み取り許可がないと、ディレクトリに格納されているファイル名を読み取れなくなる。次に、書き込み許可は、ディレクトリ内容を変更できるかどうかという意味になる。書

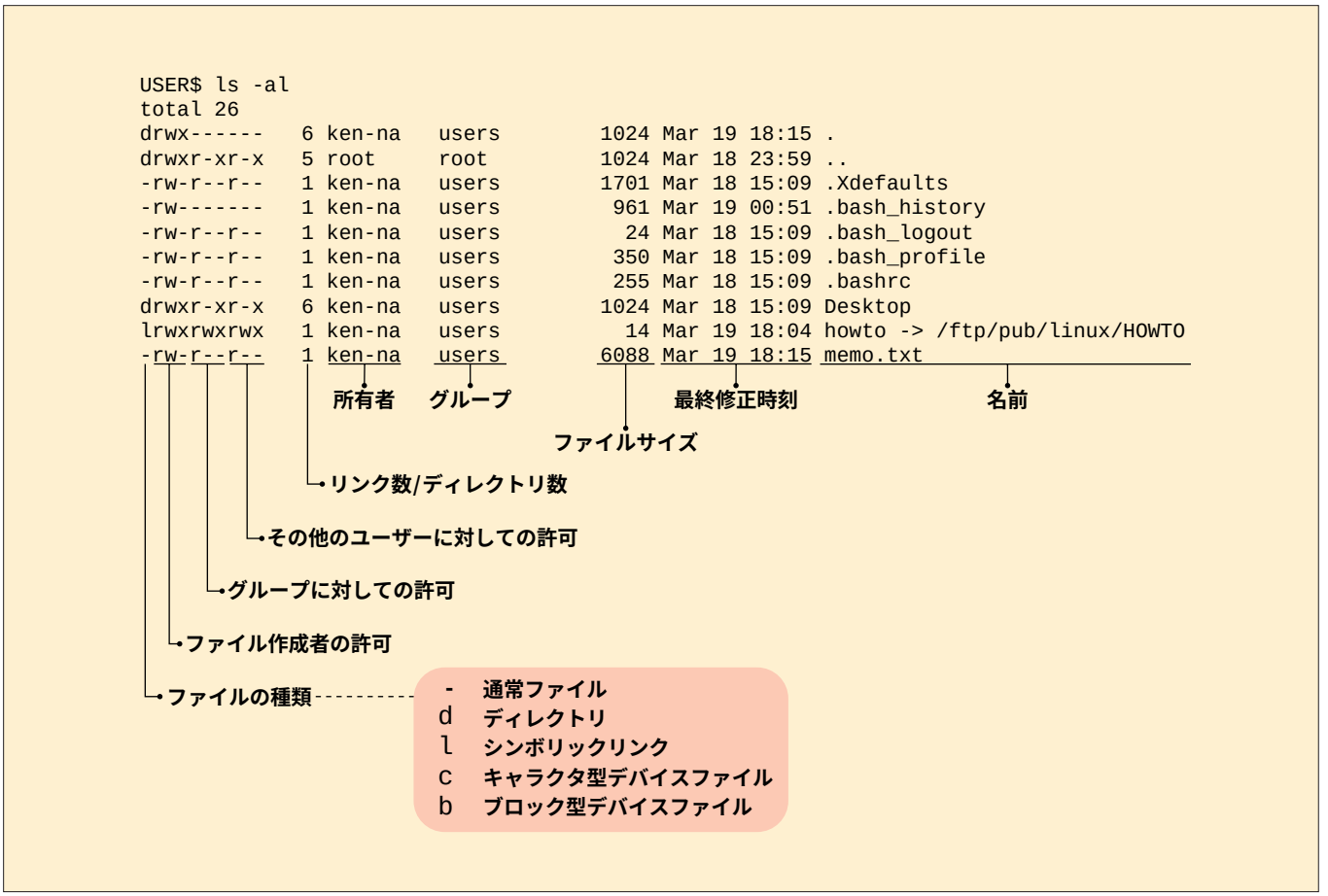


図2　lsの詳細表示

き込み許可がないと、ディレクトリに変更を加えることができない。つまり、ファイルの追加や削除、更新ができなくなる。

そして、実行許可は、そのディレクトリより先に進めるかどうかという意味になる。実行許可がないと、そのディレクトリに含まれているファイルにアクセスできなくなる。

ファイルとディレクトリの属性の変更

ファイルやディレクトリのパーミッションを変更するには、chmodコマンドを用いる。

パーミッションは、読み取り、書き込み、実行の3組を所有者、グループ、その他のユーザーに対してそれぞれ設定する。この指定の方法には記号モードと数値モードの2通りがある。

たとえば記号モードでは、次のように指定する。

```
USER$ chmod ugo+rwx fuga
```

最初のugoは、所有者（user）、グループ（group）、その他（other）に対しての操作であることを指定している。次のプラス記号で許可を与えている。その許可は、読み取り（read）、書き込み（write）、実行（execute）である。

許可をしないようにするにはプラスの代わりにマイナスを指定する。たとえば、その他のユーザー（other）に対して書き込み（write）できないようにするには、次のようにする。

```
USER$ chmod o-w fuga
```

数値モードは、最初は難しく感じるかもしれないが、慣れると非常に使いやすい。このモードでは、それぞれの許可の設定を以下に示す値の合計で指定する。

400	所有者の読み取り
200	所有者の書き込み
100	所有者の実行
40	グループの読み取り
20	グループの書き込み
10	グループの実行
4	その他の読み取り
2	その他の書き込み
1	その他の実行

たとえば、所有者とグループに対し

て読み取りと書き込み、その他のユーザーへは読み取りだけの許可となるようにするには、次のようにする。

```
USER$ chmod 664 fuga
```

所有者だけに読み取りと書き込み、グループとその他は読み取りだけにするには、644となる。所有者だけがフルコントロール、グループとその他は何もなしならば700だ。

ディレクトリに含まれるファイルのパーミッションをすべて変更したい場合は、Rオプションを指定する。

```
USER$ chmod -R go-w fuga
```

ただし、このときにファイルの実行フラグを取ろうとして、

```
USER$ chmod -R go-x fuga
```

のようにしてしまうと、サブディレクトリの実行許可もなくなってしまう。ディレクトリはそのままで、通常のファイルだけ実行許可を変更したい場合は、次のようにする。

```
USER$ find . -type f -exec chmod
go-x {} \;
```

グループの変更は、chgrpコマンドで行う。ただし、新しく設定するグループのメンバーでなければならない。

自分が属しているグループはgroupsコマンドでわかる。

```
USER$ groups
users root
```

この例でのユーザーであれば、グループをusersかrootに変更することが可能だ（**画面1**）。

ディレクトリに含まれるファイルのグループをすべて変更したい場合は、chmodと同じようにRオプションを指定する。

ファイルの所有者の変更は、chownコマンドで行う。ただし、スーパーユーザーだけが変更できる。それ以外のユーザーは変更できないことに注意する（**画面2**）。

chownもRオプションで再帰的に所有者を変更できる。

ディレクトリの作成と削除

ディレクトリの作成はmkdirコマンドで行う。引数には作成するディレクトリ名を指定する。

```
USER$ mkdir linux
USER$ mkdir linux/doc
```

パスを指定してディレクトリを作るときには、上位のディレクトリがすでに存在している必要があることに注意が必要だ。指定したパスに存在していないディレクトリ名が含まれているとエラーになる。

ディレクトリを削除するにはrmdirコマンドを使う。

```
USER$ rmdir linux/doc
USER$ rmdir linux
```

ただし、ディレクトリが削除されるのは、ディレクトリの内容が空になっているときだけである。つまり、指定したディレクトリの中にファイルやサブディレクトリがあると、ディレクトリは削除されない。

ファイルやサブディレクトリの有無に関わらず、ディレクトリを丸ごと削除するには、あとで説明をするrmコマンドを使う。

ファイルとディレクトリのコピー

ファイルをコピーするには、cpコマンドを用いる。

```
USER$ cp linux.txt linux-mag.txt
```

最初の引数にはコピー元となるファイル名、次にコピー先のファイル名を指定する。コピー先のファイルがすでに存在していると、そのファイルの内

```
USER$ ls -l linux.txt
-r--r--r--   1 ken-na   users   15097 Mar 19 18:56 linux.txt
USER$ chgrp root linux.txt
USER$ ls -l linux.txt
-r--r--r--   1 ken-na   root    15097 Mar 19 18:56 linux.txt
```

画面1　ファイルのグループを変更する

```
root# ls -l linux.txt
-r--r--r--   1 ken-na   root    15097 Mar 19 18:56 linux.txt
root# chown www  linux.txt
root# ls -l linux.txt
-r--r--r--   1 www      root    15097 Mar 19 18:56 linux.txt
```

画面2　ファイルの所有者を変更する

容はコピー元の内容に上書きされるので注意をしよう。

iオプションを指定すると、コピー先に指定をしたファイルがすでに存在していると、そのファイルを上書きしてもよいかどうかの確認メッセージが表示される。

```
USER$ cp -i linux.txt linux-mag.txt
cp: overwrite `linux-mag.txt'?
```

ここで“ n ”と答えるとコピーされない。“ y ”と答えればもちろん上書きされる。

コピー先はファイル名でなく、ディレクトリ名でもかまわない。この場合は、指定したディレクトリの中に、コピーするファイルが元と同じ名前で作られる。

```
USER$ cp linux.txt backup
USER$ ls backup
linux.txt
```

rオプションを指定すると、コピー元にもディレクトリを指定できる。このオプションはサブディレクトリに対しても処理を行うという意味を持っている。

```
USER$ cp -r linux Linux
```

cpコマンドでコピーされたファイルは、作成された日付となり、パーミッションもそのディレクトリにあわせて設定される。これらの情報もコピーしたい場合は、pオプションを指定する。

ファイルとディレクトリの削除

ファイルの削除は、rmコマンドで行う。削除したいファイルを引数に指定する。

```
USER$ rm foo bar
```

指定されたファイルの存在するディレクトリに書き込み許可がないと削除できない。ディレクトリ内容の変更ができないためだ。この場合は、chmodコマンドで、ディレクトリに書き込み許可を追加してから削除する。

また、ディレクトリに書き込み許可があっても、削除するファイルに書き込み許可がないと、削除するかどうかの確認メッセージを出力する。fオプションを指定すると、書き込み許可に関わらず、確認メッセージを出さないようになる。

```
USER$ rm -f filename
```

逆に、書き込み許可があっても確認メッセージを出力したい場合は、iオプションを指定すれば、個々のファイルに対して削除するかどうかを聞かれる。

```
USER$ rm -i *
rm: remove `sample.aux'?
```

しかし、慣れてきたり、答える回数が多くなると、勢いで“ y ”を押してしまうこともある。できれば、rmコマンドを実行する前に、lsコマンドで同じ指定をして一致するファイルを確認するほうがよいだろう。

rmコマンドにも、rオプションがある。意味はcpと同じだ。ディレクトリに対して再帰的に操作を繰り返す。

```
USER$ rm -r directory
```

先に説明をしたrmdirコマンドは、対象ディレクトリ内にファイルやサブディレクトリがある場合、ディレクトリを削除しない。しかし、rmコマンドではファイルもサブディレクトリも削除する。そのため、rオプションを使う場面では、特に気をつけて使おう。

ファイルとディレクトリの移動（名前の変更）

ファイルの名前を変更するのには、mvコマンドを使う。最初の引数には元の名前、2番目の引数に新しい名前を指定する。

```
USER$ ls -F
linux.doc
USER$ mv linux.txt linux.doc
USER$ ls -F
linux.doc
```

mvはファイルを別のディレクトリに移動するのにも使える。

```
USER$ ls -F
linux.doc  mag/
USER$ mv linux.doc mag
USER$ ls -F
mag/
USER$ ls -F mag
linux.doc
```

そして、ディレクトリに対しても同様に行うことができる。

mvは、ディスク上のファイルのデータ内容を移動しているのではなく、ディレクトリが保持しているファイル名のデータを書き換えているだけである。そのため、mvを実行するには、移動元ファイルのあるディレクトリと移動先のディレクトリの両方に書き込み許可が与えられていなければならない。

便利なコマンド

知っているとオペレーションが100倍便利になる各種のコマンドを覚えよう。

文：中野 賢
Text : Ken Nakano

入出力の切替え

ディレクトリ内容を表示するのにlsコマンドを利用した。

```
USER$ ls /
```

lsコマンドを次のようにして使うと、その出力が端末でなく、sample.txtファイルに出力される。これをリダイレクトと呼ぶ。

```
USER$ ls / > sample.txt
```

>の代わりに>>を使うと、出力した内容が指定されたファイルの末尾に追加される。

```
USER$ ls /bin >> sample.txt
```

lsコマンドは通常、結果を端末（画面）に出力する。この端末は標準出力と呼ばれる。>と>>は表示先を標準出力から、ファイルへと切替えるための記号だ。この記号はlsが提供する機能ではなく、シェルの機能なので、標準出力に対して結果を書き出すすべてのコマンドに対して利用できる。

標準出力に対応して**標準入力**も用意されている。

catコマンドは、指定されたファイルの内容を標準出力に表示するコマンドであるが、入力ファイルを指定しないと、キーボードを入力とする。このキーボードが**標準入力**だ。標準入力からの入力を終了するには、行頭でCtrl－D（コントロールキーを押しながらDキー）を入力する。

```
USER$ cat
hoge    キーボードから入力した文字
hoge    catコマンドの出力
fuga    キーボードから入力した文字
fuga    catコマンドの出力
        Ctrl-D
USER$
```

<を使うと、標準入力をファイルに切替えることができる。

```
USER$ cat < sample.txt
```

また、|を使うと、コマンドの出力を標準入力として、別のコマンドに渡すことができる。この機能をパイプと呼んでいる。

次の例では、lsで表示した結果を、sortコマンドで並べかえている。

```
USER$ ls -aF1 | sort
../
./
.Xdefaults
.bash_history
.bash_profile
.bashrc
.emacs
   :
   :
```

Linuxで提供されているコマンドを個別に見ると、それぞれは単純な動作しかしないものが多い。しかし、パイプや入出力の切替えをうまく使うと、複雑な処理であっても、簡単に実現できる。これもコマンドによるオペレーションの魅力だろう。

```
USER$ ls -aF1 | sort > sorted.lst
```

ファイルの内容を表示する

ファイルの内容を出力するコマンドはいくつもあるが、catコマンドがもっとも基本的なコマンドだ。

catコマンドは指定されたファイルの内容を標準出力に書き出す。複数のファイルを指定すると、それらをまとめて出力する。

```
USER$ cat a.txt b.txt c.txt > all.txt
```

のように、ファイルを連結するのに便利だ。

nオプションを付けると、各行に番号が付いて出力される。

```
USER$ cat -n smaple.doc
```

ファイルの内容が1画面に収まらないときはlessコマンドが便利だ。端末の画面サイズに応じて表示が止まる。よく使うキー操作を**表2**に示す。

```
USER$ less /etc/services
```

キー	実行内容
スペースバー	1画面下方向にスクロール
b	1画面上方向にスクロール
j	1行下方向にスクロール
k	1行上方向にスクロール
p	ファイルの先頭へ移動
G	ファイルの末尾へ移動
h	使い方の説明
q	終了

表2　lessのキー操作

ログファイルなどは、ファイルの末尾のほうに最近の情報が格納されている。このときはtailコマンドが便利だ。デフォルトでは末尾の10行分だけを表示する。行数を変更するには、その数値をオプションで指定する。

```
root# tail -20 /var/log/messages
```

画面出力も見ながら、結果をファイルに残したいときはteeコマンドが便利である。teeは標準入力を指定されたファイルと標準出力に書き出す。

```
USER$ vmstat 5 10 | tee /tmp/vmstat
.log
```

オンラインマニュアル

コマンドオプションは、プログラムの動作を変更するために設けられている。そのため、プログラムの役割が異なれば、使えるオプションも異なるし、同じオプション名であっても動作が異なる。たとえば、rmやmvではfは強制的に動作させるためのオプションだが、tarのfオプションはアーカイブファイル名を指定するのに用いる。さらに、オプションの指定方法もコマンドごとに異なる。aとbオプションを指定するとき、-abとできるコマンドもあれば、-a -bと個々に指定するしかないのもある。

使い方のわからないコマンドは、manコマンドで簡単に調べることができる。使い方は、次のように、引数にコマンド名を指定するだけだ。

```
USER$ man man
```

kオプションでキーワードを指定すると、関連するページが表示される（**画面3**）。

かっこ内の数値はセクション名であり、次の分類を示している。

1	一般的なコマンド
2	システムコール
3	ライブラリ関数
4	スペシャルファイル
5	ファイルフォーマット
6	ゲーム
7	その他
8	管理コマンド
l	ローカルなコマンド
n	新しいコマンド

同じ名前のページが複数のセクションにある場合は、セクション番号を指定して、目的のページを表示する。たとえば、passwdコマンドではなく、パスワードファイル/etc/passwdの詳細について調べたい場合は、セクション5にあるpasswdを表示するよう指定する。

```
USER$ man 5 passwd
```

オンラインマニュアルには、もうひとつinfoコマンドも用意されている。

```
USER$ info info
```

とすると、infoコマンドのチュートリアルが表示されるので試してみてほしい。

コマンドオプションを調べたいとき、manやinfoでも見つからなければ、--helpや-hを付けて実行してみよう。オプションの説明が表示されることがある。

ファイルの種類

ディレクトリツリー上には、さまざまな種類のファイルが置かれている。そのため、バイナリファイルと気づかずにcatして画面が乱れてしまうこともある。事前にどのようなファイルなのかがわかると便利だ。

```
USER$ man -k passwd
chpasswd (8)     - update password file in batch
gpasswd (1)      - administer the /etc/group file
htpasswd (1)     - Create and update user authentication files
mkpasswd (1)     - generate new password, optionally apply it to a user
mkpasswd (8)     - Update passwd and group database files
passwd (1)       - update a user's authentication tokens(s)
passwd (5)       - password file
pg_passwd (1)    - Manipulate the flat password file
userpasswd (1)   - A graphical tool to allow users to change their passwords.
yppasswd, ypchfn, ypchsh (1)      - change your password in the NIS database
```

画面3　manで“passwd”に関連するページを検索

```
USER$ file /usr/bin/*
/usr/bin/GnomeScott:     ELF 32-bit LSB executable, Intel 80386, version 1,
dynamically linked (uses shared libs), stripped
/usr/bin/Mail:           symbolic link to ../../bin/mail
/usr/bin/Pnews:          Bourne shell script text
/usr/bin/Rnmail:         Bourne shell script text
/usr/bin/SwitchXIM:      Bourne shell script text
/usr/bin/X11:            symbolic link to ../X11R6/bin
/usr/bin/[:              symbolic link to test
/usr/bin/a2p:            ELF 32-bit LSB executable, Intel 80386, version 1,
dynamically linked (uses shared libs), stripped
/usr/bin/access:         ELF 32-bit LSB executable, Intel 80386, version 1,
dynamically linked (uses shared libs), stripped
/usr/bin/aclocal:        perl commands text
                            :
                            :
```

画面4　fileコマンドの実行例

fileコマンドは指定されたファイルの種類を表示してくれる（**画面4**）。

ただし、バイナリ形式に対してはかなり信頼できる報告をしてくれるが、テキストファイルに対しては間違いが多い。これは、fileコマンドがファイル形式を決定するのに、ファイルの先頭付近が特定のパターンになっているかどうかを調べているからである。そのため、テキストファイルのように決定的なパターンが見つからないと、うまく判断ができないのである。しかしながら、画面が乱れる恐れのあるファイルかどうかを調べるには十分だ。

なお、ファイル形式を決めるためのパターンは、/usr/share/magicファイルに記述されている。

漢字コードと改行コードの変換

テキストファイルに使われる漢字コードは、現在、EUC、シフトJIS、JISコードが代表的だ。ネットワークなどによって、異機種間でのファイルのやりとりが多くなると、すべてのファイルの漢字コードを統一するのは難しい。

lessやjvim、muleなどは、使われている漢字コードを自動的に認識し、変換して表示してくれる。しかし、すべてのプログラムがそうなっているわけではない。特に、sedやAWK、Perlなどでフィルタ処理するときには、漢字コードの違いによって、処理結果が異なってくるので注意が必要だ。

漢字コードの変換プログラムには、いくつかあるが、nkfコマンドがよく使われる。nkfが便利なのは、入力ファイルの漢字コードを自動認識してくれることだ。出力コードは、EUC（e）、シフトJIS（s）、JIS（j）を指定できる（かっこ内は指定するときのオプション）。たとえば、

```
USER$ nkf -e infile
```

とするとinfileの内容をEUCコードに変換して標準出力に出力する（画面に表示する）。

nkfは漢字コードは変換してくれるけれど、改行コードは変換してくれない。改行コードを変換する方法もいろいろあるが、今回はtrを使ってみよう。

```
USER$ nkf -s infile | tr \\n \\r
```

とすれば、UNIXの改行コード（\n）をMacintoshの改行コード（\r）に変換する。MS-DOSの改行コード（\r\n）に変換するには、

```
USER$ nkf -e infile | tr \\n \\r \\n
```

のようにする。バックスラッシュを重ねているのは、シェルに特殊文字だと解釈されないようにするためである。次のように引用符で囲んでもよい。

```
USER$ nkf -e infile | tr '\n' '\r\n'
```

なお、trコマンドの入力は標準入力

Column

コマンドラインでの一括処理

変換するファイルがいくつもあるときは、シェルの機能を使って、一気に変換すると便利だ。たとえば、bashの場合は、次のようにする。>のプロンプトはfor文の続きを入力するためのプロンプトである。

```
USER$ mkdir sjis
USER$ for f in *.doc ; do
> nkf -s $f > sjis/$f
> done
```

もし、カレントディレクトリにa.doc、b.doc、c.docというファイルがあれば、

```
nkf -s a.doc > sjis/a.doc
nkf -s b.doc > sjis/b.doc
nkf -s b.doc > sjis/c.doc
```

をしたことと同じになる。

だけである。したがって、nkfやcatを用いて、trにデータを渡すか、リダイレクトする必要がある。

文字列の検索

ファイルの数が多くなると、ファイル名だけでは、どこに何があったのかわからなくなってくる。あるキーワードが含まれるファイルを探したいときに、エディタの検索機能を用いるのも不便だ。grepコマンドはファイルから、指定したキーワードを探してくれる。

たとえば、linux.txtから“ linux ”という文字列を探すには、次のようにする。

```
USER$ grep linux linux.txt
```

キーワードの大文字／小文字を区別しないで探すのならiオプションを付ける。nオプションは一致した行の番号を表示する。

```
USER$ grep -in linux linux.txt
4: % ねらいとしては、Linux（UNIX）を
8: % 操作を通して、Linuxを理解してもらう
32: Linuxシステムは図1のような木構造
123: $ mkdir linux
```

キーワードの指定には**正規表現**も指定できる。正規表現とは、文字列のパターンを指定する記述法のことだ。**表3**によく使われる正規表現を示す。egrepではより強力な正規表現を使える。

正規表現を用いると、空行を探すことも簡単にできる。

```
USER$ grep -n '^$' linux.txt
10:
28:
44:
```

また、次のようにして、行頭がUSERまたはrootで始まる文字列を検索することも簡単だ。かっこや|の表記はegrepでしか使えない。

```
USER$ egrep -n '^(USER|root)' linux.txt
14:USER$
20:USER$ pwd
80:USER$ cd /usr
160:root rootユーザーのホームディレクトリ
```

fgrepを用いると、複数個のキーワードを記述したファイルに基づいて検索することができる。

```
USER$ cat key
hoge
fuga
USER$ fgrep -ni -f key linux.doc
```

この場合は、keyファイルに記述されている“ hoge ”と“ fuga ”という文字列の含まれる行を表示する。

正規表現	意味
^	行の先頭
$	行の終り
.	任意の1文字
[...]	...のうちの任意の1文字。a-zや0-9のような範囲指定も有効
[^...]	...にない任意の1文字。範囲指定も有効
r*	ゼロ回以上の、rの繰り返し
r+	1回以上のrの繰り返し
r?	ゼロまたは1回のr
r{n,m}	n回以上m回以下のrの繰り返し
r1|r2	r1またはr2（egrepのみ）
(r)	rの正規表現（egrepのみ）

表3　grepとegrepの正規表現

Column

バーチャルCD-ROM

CD-ROMのバックアップを取りたい場合、catコマンドの引数にデバイスファイルを指定すると、そのデータを読み込み、標準出力に出力する。そこで、次のようにするだけで、簡単にCD-ROMの内容を取り出すことができる。

```
USER$ cat /dev/scd0 > /home/data/foo|.img
```

作成したファイルは、ループバックデバイスを経由して、ファイルシステムにマウントできる。

```
root# mount -t iso9660 -o loop /home/data/foo.img /mnt/cdrom
```

ファイルの検索

ファイルが多くなってくると、どこに何のファイルを置いたか忘れてしまうことがある。また、エディタが作ったバックアップファイルだけを削除したいというようなこともある。findコマンドは、ディレクトリツリーの中から指定された条件に合うファイルを見つけてくれる。さらに、そのファイルに対して操作をすることもできる。

拡張子が.bakの名前を持つファイルを探したいときは、次のようにする。最初の引数のドットは検索をするディレクトリ位置である。この場合は、カ

レントディレクトリ以下を探すよう指定している。-nameはファイル名で探すという指示であり、名前の条件は「*.bak」に一致するファイルとしている。ファイル名を引用符で囲んでいるのは、アスタリスクをシェルに解釈されないようにするためだ。

```
USER$ find . -name '*.bak'
./2000-05/linux.txt.bak
./2000-05/lists.txt.bak
```

execオプションを用いると、探し出した各ファイルに対して、削除やパーミッションを変更したりといった操作をすることができる。たとえば、バックアップファイルを削除するには、次のように指定する。execオプションはセミコロン（;）までの間をコマンドとして実行する。セミコロンにバックスラッシュを付けているのも、シェルに解釈されないためである。

```
USER$ find . -name '*.bak' -exec
rm -f {} \;
```

{} の部分が見つかったファイルの名前に置換される。上記の例は、次のように実行されることに等しい。

```
rm -f ./2000-05/linux.txt.bak
rm -f ./2000-05/lists.txt.bak
```

ディレクトリ単位でファイルのパーミッションを変更するようなとき、chmodのRオプションでは、ファイルもサブディレクトリもすべて変更してしまう。しかし、設定するパーミッションによっては、ディレクトリのパーミッションは変更したくない場合がある。このようなときにもexecオプションは便利だ。

次の例では、ファイルのパーミッションを644、ディレクトリを755にしている。

```
USER$ find . -type f -exec chmod
644 {} \;
USER$ find . -type d -exec chmod
755 {} \;
```

findは時刻をもとに検索することもできる（**画面5**）。

mtimeオプションはファイルが最後に修正された時刻で検索する。オプションのうしろの数値は日数を示している。日数の前にプラスを付けると、最後に修正された時刻がその日数よりも前のファイルが報告される。マイナスの場合は指定した日数が経っていないファイルが報告される。何もつけないとその日数分だけ経過したファイルになる。

mminオプションもmtimeと同様だが、指定する数値の意味が分単位になる点だけが違う。

newerオプションは数値でなく、ファイルを指定する。そして、指定されたファイルが最後に修正された時刻よりも、最近修正されたファイルが報告される。

mtime／mmin／newerはファイルが最後に修正された時刻をもとにしたが、同様のものとして、最後にアクセ

```
USER$ date
Wed Mar 22 15:07:46 JST 2000
USER$ ls -ltr
合計 228
-rw-r--r--   1 ken-na   staffs       2909 Mar 13 15:27 from-kinoshita.txt
-rw-r--r--   1 ken-na   staffs       1958 Mar 14 20:15 options.txt
-rw-r--r--   1 ken-na   staffs       2643 Mar 15 17:42 lists.txt
-rw-r--r--   1 ken-na   staffs       2643 Mar 15 17:42 lists.txt.bak
-rw-r--r--   1 ken-na   staffs      58572 Mar 22 15:11 genko.txt
-rw-r--r--   1 ken-na   staffs      58655 Mar 22 15:11 genko.txt.bak
USER$ find . -mtime +6
./from-kinoshita.txt
./options.txt
USER$ find . -mtime -6
.
./genko.txt
./genko.txt.swp
./genko.txt.bak
USER$ find . -mtime 6
./lists.txt
./lists.txt.bak
USER$ find . -mmin -10
.
./genko.txt
./genko.txt.bak
USER$ find . -newer options.txt
.
./genko.txt
./genko.txt.bak
./lists.txt
./lists.txt.bak
```

画面5　findの実行例

スされた時刻をもとにするatime／amin／anewer、最後に属性が変更された時刻をもとにするctime／cmin／cnewerもある。

ファイルの圧縮と展開

ソースファイルや文書ファイルを配布するときに、バラバラに渡すと必要なファイルがなかったりして混乱しやすい。そこで複数のファイルをまとめることになる。この用途のためのプログラムはアーカイバとか書庫プログラムと呼ばれる。アーカイバにもいろいろあるが、Linuxではtarがよく使われている。

アーカイブファイルを作成するには、作成を指示するためのcオプションと、出力ファイルを指定するためのfオプションを指定する。そして、含めるファイルを指定する。

```
USER$ tar cf ciao.tar ciao.c ciao.1
Makefile
```

ディレクトリを指定した場合は、ディレクトリ情報も保存される。

```
USER$ tar cf hoge.tar ciao
```

作成されたtarファイルの内容を確認するには、cの代わりにtオプションを使う。

```
USER$ tar tf ciao.tar
ciao/.exrc
ciao/Makefile
ciao/ciao.c
ciao/ciao.1
```

tarファイルを展開するには、xオプションを指定する。ディレクトリ付きで作成されているtarファイルの場合は、そのディレクトリもあわせて復元される。

```
USER$ tar xf ciao.tar
USER$ ls -a ciao
.    .exrc      ciao.c
..   Makefile   ciao.1
```

パス名を含めてtarファイルを作成するときは、少し注意が必要だ。絶対パスで保存をしてしまうと、パーミッションの関係から、スーパーユーザーでないと展開できなくなる可能性があるからだ。GNU版ではPオプションを付けない限りは、先頭の/を取り除いてtarファイルを作成するようになっている。だが、他のOSが提供しているtarコマンドもそうだとは限らない。間違いを減らすためにも、つねに相対パスで指定するクセを付けるほうがよいだろう。

tarコマンドは、ファイルをまとめるだけで圧縮はしない。そのため、ファイルの大きさを小さくするには、別に圧縮をする必要がある。以前は圧縮形式として、compress形式が使われていたが、最近はより圧縮率の高いgzip形式が使わることが多い。

tarファイルを圧縮するには、次のようにする。

```
USER$ gzip ciao.tar
```

すると、カレントディレクトリにciao.tar.gzが作成される。tarでまとめたものをパイプでgzipコマンドに渡すこともできる。tarコマンドの－は標準出力を示している。gzipコマンドの-cも標準出力へ出力するオプションである。

```
USER$ tar cf - ciao | gzip -c >
ciao.tar.gz
```

逆に、tar＋gzipで作られたアーカイブのリスト一覧を見るときや展開するときも同様に行える。gzipのdオプションは、展開をするためのオプションだ。

```
USER$ gzip -cd ciao.tar.gz | tar tf -
```

GNU版のtarでは、tarコマンドのzオプションで同様のことができる。

```
USER$ tar zcf ciao.tar.gz ciao
USER$ tar ztf ciao.tar.gz
USER$ tar zxf ciao.tar.gz
```

tarコマンドは、ファイルシステムを越えて、ディレクトリを移動するときにも便利だ。次の例の後半は、ディレクトリを移動して、そこで標準入力から受け取った内容を展開している。後半部分をかっこで括らないと正しく動作しないことに気をつけてほしい。

```
USER$ tar cf - hoge | (cd
/export/share/fuga ; tar xf - )
```

また、rshコマンドを利用できれば、ネットワークを越えてファイルコピーするのにも使える。

```
USER$ tar cf - hoge | dd bs=1024 |
rsh リモートホスト名 dd bs=1024
of=hoge.tar.gz
```

2番目のddコマンドは出力するブロックサイズを指定するためだけに用いている。そして、リモートホストのddでは、ofオプションでhoge.tar.gzに書き出している。ofの指定をテープデバイスに指定すれば、ローカルホストに接続されていないデバイスにバックアップすることができる。

管理コマンド

管理者たるもの、システム管理はコマンドを駆使してきめ細かく行いたい。

文：中野 賢
Text : Ken Nakano

ここでは、システム管理に関連するコマンドについて説明をする。そのため、一般ユーザーでは実行できないコマンドもいくつかあるが、スーパーユーザーが何をしているのかをかいま見ることができるだろう。また、Linuxシステムを理解する一端となると思う。

スーパーユーザー

Linuxでは、管理者ユーザーのことをスーパーユーザーと呼んでいる。アカウント名がrootであることからルートとも呼ぶ。

スーパーユーザーは、ファイルに設定されている読み取りや書き込みといった許可に関係なく、ファイル操作をすることができる。たとえば、一般ユーザーの所有するファイルのパーミッションが600であっても、スーパーユーザーは何の制限もなくファイルを読み、修正することができる。削除してしまうことも可能だ。そのため、スーパーユーザー権限でコマンドを実行するときは、特に注意深く行わなければならない。もし、カレントディレクトリが/であるのに気づかず、うっかりとrmコマンドをfrオプション付きで実行してしまうと、無条件にすべてのファイルが削除され、悲惨な状況に陥ってしまう。

スーパーユーザー権限で作業をするのに、再度ログインをする必要はない。コマンドラインから、

```
USER$ su
```

と実行し、rootのパスワードを入力するだけだ。これだけでスーパーユーザーになる。逆に、一般ユーザーレベルに戻るには、

```
root# exit
```

とする。なお、環境変数などもrootとしてログインした場合と同じにする場合は、“ su - ”を実行する。

suコマンドを使うとスーパーユーザーになるだけでなく、任意のユーザーにもなることができる。たとえば、PostgreSQLではデータベース管理者用に特別なアカウントpostgresを設けている。そのユーザーになるにも、

```
USER$ su postgres
```

として、パスワードを入れる。パスワードが正しければ、PostgreSQLのデータベース管理者となることができる。

そこで、パスワードを知らないはずのユーザーが、rootや他のユーザーになっていないかを時おり、確認しなければならない。

誰がsuコマンドを実行して、どのアカウントになろうとしたのかは、/var/log/messagesファイルに記録されている。このファイルは、通常のテキスト形式なため、lessで表示したり、grepで必要な部分だけを抽出することができる。suコマンドの部分だけなら、

```
root# grep '(su)' /var/log/messages
```

とすればよい。クオーテーション（'）で囲っているのは、かっこがシェルの特殊文字であるためだ。なお、messagesファイルは他にも多くの記録が残されているため、rootユーザー以外は読めないようにしておこう。

パスワード

銀行のキャッシュカードやクレジットカードの暗証番号は極めて重要な役割を持っている。パスワードを知っていれば、本人になりすましてたいていのことができてしまうからだ。現代社会ではそれなりに認識されていると思う。Linuxのパスワードも同じだ。

パスワードを忘れないようにと手帳に残しておいてはいけない。ましてや、それをディスプレイに貼っておいたり、机の引出しにいれておくことは論外だ。また、パスワードを入力するときは、周りに人がいないことを確認したり、背中で手もとを隠して、他人に見られないように気を配ろう。逆に、他人がパスワードを入力するときには、認証が終るまで、うしろを向いたりして画面や手もとを見ないようにするのが礼儀だ。

銀行口座の入出金を記帳したり、クレジット明細書を見て、おかしな取り引きがないかを確認するように、自分のログインした日付をときどき確認しよう。

```
USER$ last | grep ken-na
```

lastコマンドは、/var/log/wtmpファイルに保存されているログイン記録を表示している。ただし、ファイルサイズが大きくなりがちなため、1カ月程度でファイルをリセットして運営することも多い。このとき、前月分を/var/log/wtmp.1、前々月分を/var/log/wtmp.2ファイルとして残すようになっていれば、fオプションで、前月分の記録を参照することができる。

```
USER$ last -f /var/log/wtmp.1 |
grep ken-na
```

もしも不審な時間にログインしているようであれば、すぐにパスワードを変更し、管理者に連絡をする。

パスワードの変更はpasswdコマンドで行う。コマンドを実行すると、現在のパスワードを求められる。変更しようとしているユーザーが本人かどうかを確認するわけだ。入力したパスワードが正しければ、新しく設定するパスワードの入力が求められる。そして、新パスワードの内容を確認するために再び入力を求められる。この2度のパスワードが一致して初めて、パスワードが更新される（**画面6**）。

新パスワードに変更を終えたら、一度ログアウトして、新パスワードでログインできるかを確認したほうがよいだろう。

パスワードを設定するときに、たまたま2度同じタイプミスをしたりすると、ログインできないかもしれない。あるいは他人によって変更されていたり、忘れてしまってログインできない場合もある。このときはスーパーユーザーに頼むしかない。

一般ユーザーは自分のログイン名のパスワードしか変更することができないが、スーパーユーザーは誰のパスワードでも変更することができる。スーパーユーザーが他人のパスワードを設定するときは、引数に変更するユーザーのログイン名を指定する。

```
root# passwd ken-na
```

スーパーユーザーが実行したときは、現在設定されているパスワードの入力を求められない。そこで、新パスワードを適当に設定して、それをユーザーに伝える。もちろん、受け取ったユーザーはすぐにパスワードを変更する。

それでは、スーパーユーザーがrootのパスワードを忘れてしまった場合はどのようにするか。シングルモードで起動をし、passwdコマンドで設定をするだけだ。

シングルモードで起動するには、LILOで複数のカーネルを切り替えて起動するようにしてあるのならば、起動ラベル名の引数として、1またはsを指定する。

```
linux 1
```

LILOを使っておらず、オプションを指定するチャンスのない場合は、レスキューディスクやブートディスクで起動するしかない。そして、passwdファイルのあるファイルシステムをマウントし、パスワードを変更する。

ユーザーとグループ

ディストリビューションによっては、システムをインストールするときに、一般ユーザーも追加することができる。しかし、この段階では管理者自身のアカウントを作成するだけにとどめておき、その他のユーザーは、少なくとも、システムが正しくインストールされ、必要なサービスも稼働していることを確認してから作成するほうがよいだろう。システムのインストールもうまくいくかわからないし、インストーラがどのように設定するのかもわからないからだ。

ユーザーを作る前に、まずはグループの確認をしておこう。Linuxではファイルのパーミッションを所有者、グループ、その他に対して設定する。1人のユーザーは複数のグループに属せるが、1つのファイルには1つのグループしか設定できない。そのため、あまりに細かくグループを分けても、全員が同じグループに属するのでも、複数ユーザーでのファイルの共有が面倒になってしまう。基本的な方針としては、共同作業を行うグループ単位とすることが多い。たとえば、開発者チームはkaihatsuグループ、広報グループはkouhouグループに分けるイメージだ。

グループを追加するには、groupadd

```
USER$ passwd
New UNIX password:          新しく設定するパスワードを入力する
Retype UNIX password:       再度、新しく設定するパスワードを入力する
Sorry, passwords do not match     2つのパスワードが一致しない場合
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully      成功
```

画面6　passwdコマンドの実行例

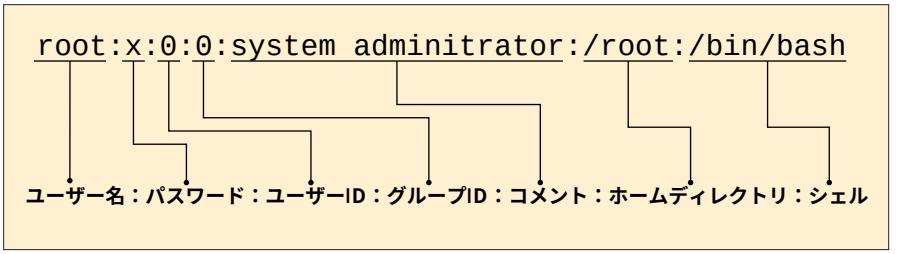


図3　/etc/passwdファイルの内容

コマンドにグループ名を指定して実行する。グループ名は既存のグループ名と重複してはいけない。/etc/groupファイルに記述されている名前と異なる名前を考えよう。実行後、/etc/groupファイルを開いてみれば、指定されたグループが追加されていることがわかる。

```
root# /usr/sbin/groupadd kaihatsu
```

ユーザーを追加するには、useraddコマンドを用いる。引数には、属するグループとユーザーのログイン名を指定する。実行後、指定したユーザーの設定が/etc/passwdファイルに追加され、/homeディレクトリに、そのユーザーのホームディレクトリが作成される。

```
root# /usr/sbin/useradd -g kaihatsu
ken-na
```

そして、passwdコマンドでパスワードを設定する。これでユーザーの作成が終了だ。

ちなみに、ユーザー情報は**図3**のように/etc/passwdファイルに格納されている。もちろん、パスワードは暗号化されて記述されている。だが、ファイル自体は誰でも読める。現在のマシン性能からすると簡単にパスワードを知られてしまう。そこで、パスワードを保護するために、多くのシステムではシャドウパスワードを用いている。シャドウパスワードでは、暗号化されたパスワードは/etc/shadowに保存され、rootだけが読むことができる。

ユーザーを削除するには、userdelコマンドを用いる。

```
root# /usr/sbin/userdel ken-na
```

ただし、そのユーザーのホームディレクトリは残されたままになっている。もし、ホームディレクトリも削除するのなら、rオプションを指定する。

```
root# /usr/sbin/userdel -r ken-na
```

最初にオプションを付けないで実行してしまったときは、rオプションを付けて再度、実行しようとしても、すでにユーザー情報が削除されているのでエラーになる。このときは、

```
root# rm -fr /home/ken-na
```

として、ホームディレクトリを削除すればよい。

ただし、userdelコマンドは、ユーザーがホームディレクトリ以外に作成したファイルを削除しない。このようなファイルは、ls -lで表示したとき、ユーザー名でなく、そのユーザーに割り当てられていた番号が表示される。このようなファイルは、同グループに属するユーザーか、rootを所有者にしておこう。

変更作業は簡単である。もちろん、ユーザーを削除する前に行ってもかまわない。

```
root# find / -uid 101 -exec chown
fuga {} \;
```

組織の形態が変更になったりして、既存ユーザーのグループを変更したいときは、usermodコマンドを用いる。

```
root# /usr/sbin/usermod -g kouhou
ken-na
```

usermodコマンドも、ユーザーの作成したファイルのグループは変更されないので、必要に応じて、findコマンドを用いて変更する。

ファイルシステム

LinuxではハードディスクもCD-ROMもフロッピーディスクもネットワーク上のファイルシステムも、/から始まるディレクトリツリーのどこかにマウントして使う。どのようにマウントされているのかを知るにはmountコマンドを実行する。

```
USER$ mount
/dev/hda5 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/hda7 on /home type ext2 (rw)
/dev/hda6 on /usr type ext2 (rw)
none on /dev/pts type devpts
(rw,gid=5,mode=620)
/dev/scd0 on /home/ftp/pub/cdrom
type iso9660 (ro)
/dev/hda1 on /mnt/dos type vfat
(rw)
```

この例では、hda5が/にマウントされ、hda7が/home、hda6が/usr、

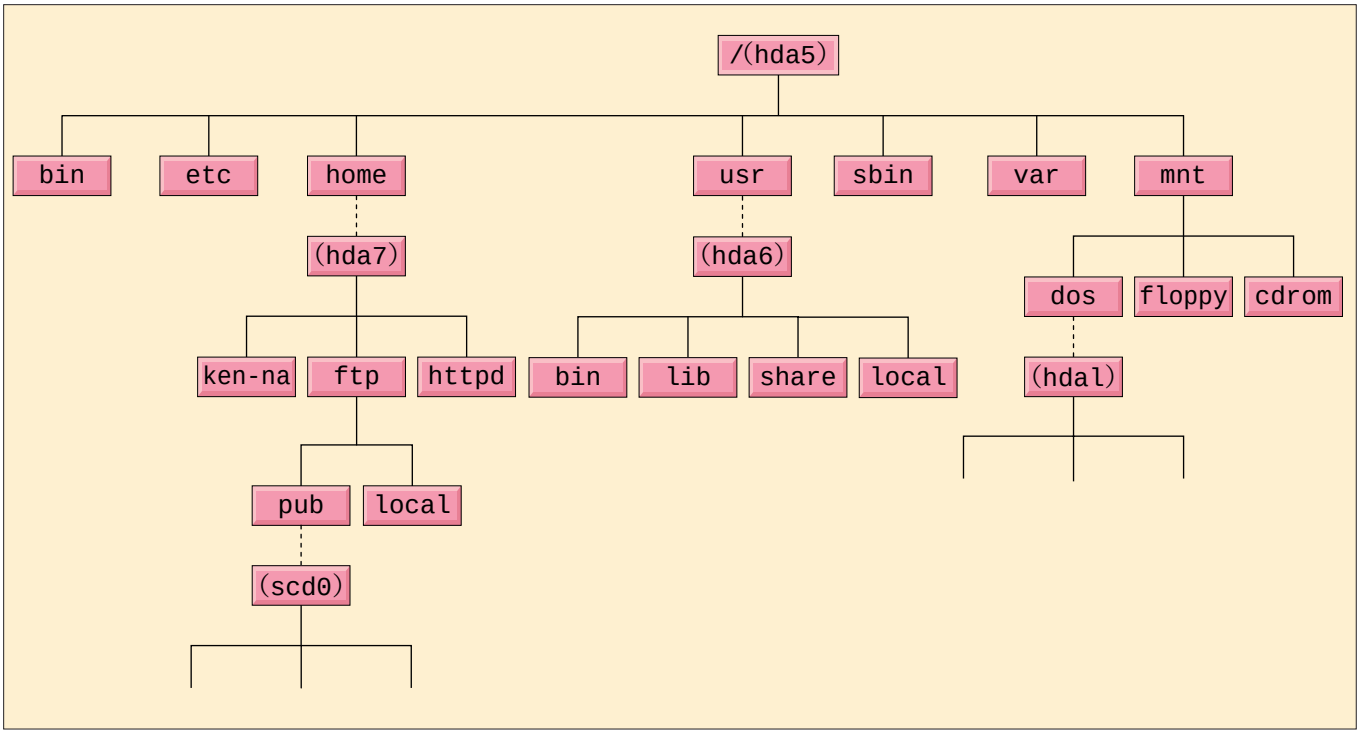


図4　複数のファイルシステムにまたがったディレクトリツリー

scd0がcdromディレクトリにマウントされていることがわかる（**図4**）。

各ファイルシステムのマウント方法は、すべて同じである。mountコマンドを用い、tオプションでファイルシステムのフォーマットタイプを指定し、マウントするデバイス名と位置を指定する（**画面7**）。

tオプションに指定できるフォーマットタイプには、**表4**のようなものがある。ただし、全部のファイルシステムが標準で使えるとは限らない。カーネルに組み込まれていなければ使えないので注意が必要だ。

マウントしてあるファイルシステムを取り外すにはumountコマンドを用いる。引数へ指定するのは、ファイルシステム名でもマウントポイントでも、どちらでもかまわない。

```
root# /bin/umount /dev/scd0
```

umountコマンドを実行したとき、誰かのカレントディレクトリがそのデバイス上にあったり、ファイルをアクセスしている場合、次のようなメッセージが表示され、アンマウントできない。

```
umount: /dev/scd0: device is busy
```

このときは、fuserコマンドで誰が使っているのかを確認する。

```
root#/sbin/fuser -vm /dev/scd0
                     USER      PID ACCESS COMMAND
/dev/scd0ken-na 10089 ..c..  bash
```

そして、そのユーザーと協議して、別のディレクトリに移動してもらったり、処理を中断してもらわねばならない。あるいはユーザーの処理が終るのを待ってから取り外すようにする。

```
root# mount -t ext2 /dev/hda7 /home      ハードディスクの場合
root# mount -t iso9660 /dev/scd0 /mnt/cdrom         CD-ROMの場合
root# mount -t vfat /dev/fd0 /mnt/floppy            フロッピーディスクの場合
root# mount -t nfs 192.168.10.1:/export/share /usr/local/share        NFSの場合
```

画面7　mountコマンドの実行例

ディスクスペース

本格的にLinuxを利用し始めると、追加インストールしたプログラムや、ユーザーの作成したファイルによって、ディスクの空き容量に余裕がなくなってくる。使い切ってしまうと、ユーザ

名前	意味
ext2	Linux ext2
iso9660	ISO9660
nfs	Networkファイルシステム
vfat	Microsoft Windows
msdos	MS-DOS
hfs	Macintosh HFS

表4　フォーマットタイプの種類

```
USER$ df
ファイルシステム  1k-ブロック  使用済     使用可     使用率%  マウント場所
/dev/hda5         147766      56248      83889      40%      /
/dev/hda7         1304468     596244     641960     48%      /home
/dev/hda6         1517920     926340     514472     64%      /usr
/dev/scd0         598872      598872     0          100%     /home/ftp/pub/cdrom
/dev/hda1         3204644     2520180    684464     79%      /mnt/dos
USER$ df -h
ファイルシステム  サイズ      使用済     使用可     使用率   マウント場所
/dev/hda5         144M        55M        82M        40%      /
/dev/hda7         1.2G        582M       627M       48%      /home
/dev/hda6         1.4G        905M       502M       64%      /usr
/dev/scd0         585M        585M       0          100%     /home/ftp/pub/cdrom
/dev/hda1         3.1G        2.4G       668M       79%      /mnt/dos
```

画面8　dfコマンドでディスクの利用内容を表示する

ーの作業ができなくなるだけでなく、ログファイルなどの書き込みもできなくなってしまうので、システムにとっても具合が悪い。

ディスクの空き容量を調べるには、dfコマンドを用いる（**画面8**）。すると、接続されているファイルシステムごとに容量が報告される。数値がわかりにくければ、hオプションを付けてみよう。わかりやすいサイズで表示してくれる。

ファイルやディレクトリをコピーするとき、どのくらいの容量が必要なのかを調べたいことがある。このような用途にはduコマンドを使う。

duコマンドをオプションを付けないで実行すると、全てのサブディレクトリごとに使用サイズが表示され、最後に合計が出力される。

```
USER$ du /etc
5       /etc/profile.d
15      /etc/X11/WindowMaker
33      /etc/X11/wmconfig
4       /etc/X11/twm
          :
          :
2310    /etc
```

この例では、/etcディレクトリ全体で2310ブロック使われている。1ブロックは1Kバイトなので、全体で約2.3Mバイトとなる。

引数に指定をしたディレクトリの合計だけが必要な場合は、sオプションを指定する。さらにhオプションも指定をすると、わかりやすい単位で表示してくれる。

```
USER$ du -s /bin
4828    /bin
USER$ du -sh /etc
2.3M    /etc
```

サブディレクトリごとに合計を出力したい場合は、次のようにディレクトリの指定をスラッシュで終らせる。

```
USER$ du -sh /etc/*/
158k    /etc/X11
204k    /etc/charsets
41k     /etc/codepages
2.0k    /etc/cron.d
8.0k    /etc/cron.daily
     :
     :
```

プロセス

Linuxはマルチプロセス・システムである。プロセスとはプログラムの実行単位のことだ。誰かがプログラムを実行すると、それはひとつのプロセスとして動作をする。同時に同じプログラムを実行しても、システムはそれぞれを別個のプロセス単位として扱う。

現在実行中のプロセスを調べるには、psコマンドを用いる。何もオプションを指定せず実行すると、psコマンドを実行した端末で実行中のプロセスだけが表示される。

```
USER$ ps
  PID TTY          TIME CMD
10085 ttyp0    00:00:00 bash
10092 ttyp0    00:00:04 vi
10276 ttyp0    00:00:00 ps
```

xオプションを指定すると、バックグラウンドで実行中のプロセスや、別の端末で実行しているプロセスも含め、自分が起動しているプロセスがすべて表示される。さらに、aオプションを付けると、すべてのプロセスが表示される（**画面9**）。

稼働中のプロセスがハングアップしてしまった場合は、killコマンドでプロセスを終了させることができる。

```
USER$ kill -9 10274
```

最初の引数はプロセスを終了させるためのシグナルを意味している。2番目の引数は終了させたいプロセス番号だ。psコマンドのPIDの欄に表示されている。

一般ユーザーも自分の実行したプロセスに対してならば有効だ。スーパー

ユーザー以外は、他のユーザーのプロセスを終了させることはできない。

なお、lpdやhttpdのようなデーモンを停止したい場合は、Red Hat系のディストリビューションならば、

```
root# /etc/rc.d/init.d/lpd stop
```

のようにして停止する。

システムの終了

最後に、Linuxシステムを停止するときの手順を簡単にまとめておこう。

・ログインしているユーザーを確認する

```
root# who  または
root# w  または
root# last | grep still
```

・稼働中のプロセスを確認する

```
root# ps aux
```

・バッファの内容をファイルシステムに反映する

```
root# sync
```

・シャットダウンする

```
root# /sbin/shutdown -h now
```

・haltと表示されたら、電源を切る。

なお、「パスワード」の項の最後で、シングルユーザー・モードで起動するのに、

```
linux 1
```

ランレベル	動作
0	停止
1,s	シングルユーザー
2	マルチユーザー（NFSサポートなし）
3	マルチユーザー
4	未使用
5	X11
6	リブート

表5　標準的なランレベル

とするように説明をした。この1はLinuxシステムのランレベルだ。ランレベルは通常、**表5**のように設定されている。

実はshutdownコマンドも、ランレベルを0にすることでシステムの停止処理を行っているのだ。スーパーユーザーはtelinitコマンドでランレベルを変更できる。そこで、shutdownコマンドを用いなくても、

```
root# sync ; /sbin/telinit 0
```

として、ランレベル0に移行すればシステムは停止する。同様に、telinitコマンドに1を指定すれば、シングルユーザーモードに移行する。そして、3を指定して実行（またはCtrl－D）すれば、再びマルチユーザーモードに移行する。

```
USER$ ps x
 PID TTY      STAT   TIME COMMAND
10051 tty1     S      0:00 -bash
10062 tty1     S      0:00 sh /usr/X11R6/bin/startx
10069 tty1     S      0:00 xinit /home/ken-na/.xinitrc -- :0 -auth /home/ken-
na/
10074 tty1     S      0:02 /usr/bin/enlightenment
10075 tty1     S      0:05 kinput2 -canna -cs localhost
10085 ttyp0    S      0:00 bash
10089 ttyp1    S      0:00 bash
10092 ttyp0    T      0:04 vi genko.txt
10153 ttyp2    S      0:00 bash
10274 ttyp1    T      0:00 fuga > hoge
10277 ttyp0    R      0:00 ps x
USER$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:02 init
    2 ?        SW     0:00 [kflushd]
    3 ?        SW     0:00 [kupdate]
    4 ?        SW     0:00 [kpiod]
    5 ?        SW     0:00 [kswapd]
  286 ?        S      0:00 portmap
  341 ?        S      0:00 syslogd -m 0
  352 ?        S      0:00 klogd
  368 ?        S      0:00 /usr/sbin/atd
  384 ?        S      0:00 crond
                 :
                 :
```

画面9　psコマンドの実行例

コマンドリファレンス

コマンドによっては数多くのオプションを持つ。このリファレンスを参考にしよう。

文：中野 賢
Text : Ken Nakano

[] で囲まれた部分は省略可能であることを示している。ドット（...）は複数個指定できることを示している。

●man

目的：オンラインマニュアルを表示する。
構文：man ［オプション］［セクション］ 名前...

オプション

-k	whatisデータベースから文字列を検索する。

●ls

目的：ディレクトリの内容を一覧表示する。
構文：ls ［オプション］［ファイル名...］

オプション

-1	垂直方向に1列で表示する。
-a	ドット（.）で始まるファイルも表示する。
-c	最後にファイルのステータスを変更した時間でソートする。詳細表示のときは、時刻の欄が、最後にファイルのステータスを変更した時間になる。
-C	垂直方向に並べて表示する。
-d	ディレクトリの中身でなく、ディレクトリ自身を一覧表示する。
-F	名前のうしろにファイルの型を示す文字を付ける。 通常のファイル なし 実行可能ファイル ＊ ディレクトリ / シンボリックリンク @ FIFO | ソケット =
-g	無視（詳細表示のとき、グループ情報を表示する）。
-G	詳細表示のとき、グループ情報を表示しない。
-h	ファイルサイズをわかりやすい単位で表示する。
-i	ファイルのinode番号を表示する。
-l	詳細形式で表示する。
-L	シンボリックリンクを、リンクされているファイルの情報で表示する。
-n	所有者とグループを名前でなく、番号で表示する。
-r	ソートの順番を逆にする。
-R	ディレクトリの内容を再帰的に表示する。
-s	ブロック単位でのファイルサイズを表示する。
-S	ファイルサイズでソートする。大きなファイルが先頭にくる。
-t	修正時刻順でソートする。新しいファイルが先頭にくる。
-u	最後にファイルをアクセスした時間順でソートする。詳細表示のときは、時刻の欄が、最後にファイルにアクセスした時間になる。
-x	水平方向に並べて表示する。
-X	拡張子の順番でソートする。

●mkdir

目的：ディレクトリを作成する。
構文：mkdir ［オプション］ ディレクトリ名

オプション

-m《モード》	ディレクトリのアクセス権を《モード》で作成する。
-p	引数に指定した各ディレクトリで、存在しないディレクトリも含めて作成する。

●rmdir

目的：空のディレクトリを削除する。
構文：rmdir ［オプション］ ディレクトリ名

オプション

-p	引数に指定したディレクトリに存在しているディレクトリも含めて削除する。

●cp

目的：ファイルをコピーする。
構文：cp ［オプション］ コピー元 コピー先

オプション

-a	元ファイルの構成と属性を保ったままコピーする。-dpRと同じ。
-d	シンボリックリンクをコピーするとき、リンクされているファイルでなく、自身をコピーする。ハードリンクされているものをコピーした場合は、コピー先でもハードリンクの状態を保つ。
-f	強制的にコピーをする。
-i	上書きをするかどうかの確認をする。
-p	コピー元のファイルの所有者、グループ、パーミッション、タイムスタンプを保持する。
-r	ディレクトリを再帰的にコピーする。ディレクトリ以外のファイルはすべてファイルとしてコピーされる。
-R	ディレクトリの内容を再帰的にコピーする。
-u	コピー先ファイルのタイムスタンプが同じか新しい場合はコピーしない。
-v	コピーする前に、そのファイル名を表示する。
-x	異なるファイルシステム上のサブディレクトリはコピーしない。

●rm

目的：ファイルを削除する。
構文：rm ［オプション］ ファイル名...

オプション

-d	ディレクトリを削除する。スーパーユーザーのみで使用可能。
-f	強制的に削除する。存在しないファイルがあっても通知しない。
-i	ファイルを削除するかどうかの確認をする。
-r	ディレクトリの内容を再帰的に削除する。
-v	削除する前にファイル名を表示する。

●mv

目的：ファイル名を変更する。ファイルを移動する。
構文：mv ［オプション］ 移動元 移動先

オプション

-f	強制的に移動する。ファイルが既に存在しても通知しない。
-i	ファイルを移動するかどうかの確認をする。
-v	移動する前にファイル名を表示する。

●chmod

目的：ファイルのアクセス権を変更する。
構文：chmod ［オプション］ アクセス権 ファイル名...

オプション

-R	ディレクトリを再帰的にたどってアクセス権を変更する。

記号モードと数値モードのいずれかで指定をする。

記号モードでは、「誰」「操作」「権利」の組み合わせを文字で記述して指定をする。複数指定する場合はカンマ(,)で追加指定できる。

誰	所有者(u)、グループ(g)、その他のユーザー(a)、すべてのユーザー(a：ugoと同じ)
操作	追加（+)、削除（-)、設定（=)
権利	読み込み(r)、書き込み(w)、実行(x)、実行時にユーザーIDまたはグループIDを追加(s)、スティッキビット(t)

数値モードでは、次に示す数値の合計で設定をする。

4000	実行時にユーザーIDを設定
2000	実行時にグループIDを設定
1000	プログラムコードをスワップに維持
400	所有者の読み取り
200	所有者の書き込み
100	所有者の実行
40	グループの読み取り
20	グループの書き込み
10	グループの実行
4	その他のユーザーの読み取り
2	その他のユーザーの書き込み
1	その他のユーザーの実行

●chgrp

目的：ファイルのグループを変更する。
構文：chgrp ［オプション］ グループ名 ファイル名...

オプション	
-f	グループを変更できなかったときのエラーを表示しない。
-h	シンボリックされているファイルでなく、自身を変更する。
-R	ディレクトリを再帰的にたどってアクセス権を変更する。
-v	実行状況を詳細に表示する。

●chown

目的：ファイルの所有者を変更する。
構文：chown ［オプション］ 所有者名 ［:グループ名］ ファイル名...

オプション	
-f	所有者を変更できなかったときのエラーを表示しない。
-h	シンボリックされているファイルでなく、自身を変更する。
-R	ディレクトリを再帰的にたどってアクセス権を変更する。
-v	実行状況を詳細に表示する。

●cat

目的：ファイルの内容を標準出力へ出力する。
構文：cat ［オプション］［ファイル名...］

オプション	
-A	-vETと同じ。
-b	空白でない行に番号を付ける。
-e	-vEと同じ。
-E	各行の最後に`$'を表示する。
-n	すべての行に番号を付ける。
-s	連続した空行を1つの空行にまとめる。
-t	-vTと同じ。
-T	タブ文字を`^I'と表示する。
-v	改行とタブ文字を除く、非表示文字を`^'表記で表示する。

●head

目的：ファイルの先頭の部分を表示する。
構文：head ［オプション］［ファイル名...］

オプション	
-《行数》	先頭から《行数》分を表示する。
-c 《数値》	先頭の《数値》バイト分を表示する。b(512バイト)、k (1Kバイト)、m (1Mバイト) を追加して、単位を変更可能。
-n 《行数》	先頭から《行数》分を表示する。
-q	ファイル名を表示しない。
-v	ファイル名を表示する。

●tail

目的：ファイルの末尾の部分を表示する。
構文：tail ［オプション］［ファイル名...］

オプション	
-《行数》	末尾の《行数》分を表示する。
+《行数》	先頭から《行数》分以降を表示する。
-c 《数値》	末尾の《数値》バイト分を表示する。b（512バイト）、k（1Kバイト）、m（1Mバイト）を追加して、単位を変更可能。
-f	ファイルの末尾まで読み込んでも終了しないで読み続ける。
-n 《行数》	末尾から《行数》分を表示する。
-q	ファイル名を表示しない。
-v	ファイル名を表示する。

●tee

目的：標準入力を読み、標準出力とファイルに出力する。
構文：tee ［オプション］［ファイル名...］

オプション	
-a	ファイルを上書きせずに、追加する。
-i	割り込みシグナルを無視する。

●nkf

目的：漢字コードの変換をする。
構文：nkf ［オプション］［ファイル名...］

オプション	
-e	EUCコードを出力する。
-E	入力コードをEUCコードとする。
-j	JISコードを出力する。
-J	入力コードをJISコードとする。
-s	シフトJISコードを出力する。
-S	入力コードをシフトJISコードとする。

●tr

目的：文字を変換、削除する。
構文：tr ［-cst］ 《文字列1》 《文字列2》
tr -s 《文字列1》
tr -d 《文字列》

オプション	
-c	《文字列1》に指定された文字以外の文字列を置換の対象とする。
-s	複数個の《文字列1》をひとつにまとめる。
-t	System V版trのような動作をする。
-s	複数個の《文字列1》をひとつにまとめる。
-d	指定された《文字列》を削除する。

●grep／fgrep／egrep

目的：パターンに一致する行を表示する。
構文：grep/fgrep/egrep ［オプション］［ファイル名...］

オプション	
-《行数》	一致した行の前後の《行数》も出力する。
-c	行の内容を表示しないで、一致した行数だけを表示する。
-e《パターン》	《パターン》を指定する。
-f《ファイル名》	パターンを《ファイル名》から読み取る。
-i	パターンの英大文字と小文字の区別をしない。
-n	行番号を表示する。
-v	パターンに一致しなかった行を対象にする。

●diff

目的：2つのテキストファイルの違いを出力する。
構文：diff ［オプション］ ファイル名1 ファイル名2

オプション	
-b	空白文字の違いを無視する。
-B	空行の違いを無視する。
-C 《行数》	一致しない行とともに、前後《行数》分も出力する。
-e	edのスクリプト形式で出力する。
-i	英大文字と小文字の違いを無視する。
-P	ディレクトリ単位で比較するとき、1番目のディレクトリにないファイルは、空のファイルとして扱う。
-q	違いの詳細を表示しないで、違っていることだけを報告する。
-r	ディレクトリ単位で比較するとき、再帰的に比較をする。
-s	内容が同じことも報告する。
-w	空白を無視して比較をする。

●sort

目的：テキストファイルの行をソートする。
構文：sort ［オプション］［ファイル名...］

オプション	
-b	先頭の空白を無視する。
-d	英文字、数字、空白以外の文字を無視する。
-f	英小文字を大文字として扱う。
-n	先頭の文字列を数値として扱う。
-r	比較の結果を逆順にする。
-t 《文字》	フィールドの区切り文字を《文字》にする。
+《位置1》 ［-《位置2》］	ソートするのに用いるフィールドを指定する。《位置1》から《位置2》の直前までがソートキーになる。《位置2》を省略した場合は行末までがソートキーとなる。
-k 《位置1》 ［,《位置2》］	ソートキーを指定する別の表記。

●uniq

目的：ソート済みのファイルから、内容の同じ行を削除する。
構文：uniq ［オプション］［ファイル名...］

オプション	
-u	重複していない行だけを出力する。

-d	重複している行だけを出力する。
-c	出現回数も表示する。

●find

目的：ディレクトリツリーの中からファイルを探し出す。

構文：find ［パス...］［オプション］［判別式］［アクション］

オプション

-daystart	-amin, -atime, -cmin, -ctime, -mmin, -mtimeの時間の基準を24時間前でなく、コマンドを実行した日の0時にする。
-follow	シンボリックリンクの参照先を検索する。
-mount	他のファイルシステムにあるディレクトリを探索しない。

判別式

+nはnより大きい、-nはnより小さい、nはちょうどnの指定。

判別式は以下の演算子で条件を追加できる。

(expr)	カッコの内部が先に評価される。
! expr	exprが偽の場合、真となる。
-not expr	! exprと同じ
expr1 -a expr2	expr1が偽の場合はexpr2は評価されない。
expr1 -o expr2	expr1が真の場合はexpr2は評価されない。
-amin n	最後にアクセスされたのがn分前であれば真。
-anewer《ファイル名》	最後にアクセスされたのが、《ファイル名》が修正された時刻以降ならば真。
-atime n	最後にアクセスされたのがn×24時間前であれば真。
-cmin n	最後にステータスが変更されたのがn分前であれば真。
-cnewer《ファイル名》	最後にステータスが変更されたのが、《ファイル名》が修正された時刻以降ならば真。
-ctime n	最後にステータスが変更されたのがn×24時間前であれば真。
-gid n	ファイルのグループID番号がnならば真。
-group《グループ名》	ファイルのグループID番号が《グループ名》ならば真。
-mmin n	最後にファイルが変更されたのがn分前であれば真。
-mtime n	最後にファイルが変更されたのがn×24時間前であれば真。
-newer《ファイル名》	最後にファイルが変更されたのが、《ファイル名》が修正された時刻以降ならば真。
-nouser	ファイルのユーザーID番号に対応するユーザーがいなければ真。
-nogroup	ファイルのグループID番号に対応するグループがなければ真。
-size n	ファイルがn分の領域を使用していたら真。
-type c	ファイルタイプがcならば真。 b ブロック型スペシャルファイル c キャラクタ型スペシャルファイル d ディレクトリ p 名前付きパイプ f 通常のファイル l シンボリックリンク s ソケット
-uid n	ファイルの所有者ID番号がnならば真。
-user《ユーザー名》	ファイルの所有者が《ユーザー名》ならば真。
-exec《コマンド》;	《コマンド》を実行する。
-ls	ファイル名を「ls -dils」形式で表示する。
-print	ファイル名をフルパスで表示する。

●file

目的：ファイルの種類を調べる。

構文：file ［オプション］［ファイル名...］

オプション

-L	シンボリックファイルの場合、リンクされているファイルについて調べる。
-z	compressで圧縮されたファイルの中身についてファイルタイプを調べる。

●tar

目的：tar形式のアーカイブファイルの作成と展開をする。

構文：tar ［オプション］［ファイル名...］

オプション

-c	新しいアーカイブを作成する。
-C《ディレクトリ名》	《ディレクトリ名》にcdしてから、動作を行う。
-t	アーカイブ内容の一覧を表示する。
-x	アーカイブからファイルを抽出する。
-f《出力先》	アーカイブファイルを《出力先》で示されるファイルまたはデバイスにする。
-p	許可情報をすべて抽出する。
-P	ファイル名の先頭の/を取り除かない。
-s	抽出するファイル名をソートする。
-T《ファイル名》	抽出または作成するファイルの名前を《ファイル名》から読み込む。
-v	処理したファイルの一覧を詳しく表示する。
-Z	アーカイブをcompressにフィルタする。
-z	アーカイブをgzipにフィルタする。

●gzip

目的：gzip形式での圧縮と伸長をする。

構文：gzip ［オプション］［ファイル名...］

オプション

-c	結果を標準出力に出力する。
-d	ファイルを伸長する。
-f	強制的に動作をする。
-l	圧縮ファイルの内容を表示する。
-n	元のファイルの名前とタイムスタンプの保存／復元をしない。
-N	元のファイルの名前とタイムスタンプの保存／復元をする。
-v	作業状態の詳細を出力する。
-r	再帰的に処理を行う。

●w

目的：ログインしているユーザーと、そのユーザーが実行中のコマンドを表示する。

構文：w ［オプション］［ユーザー名］

オプション

-f	リモートホスト名の表示をするかしないかのスイッチ（デフォルトと逆の状態にする）。
-h	ヘッダを表示しない。
-s	ログイン時刻、JCPU（そのttyで実行した全プロセスが使ったCPU時間の合計）、PCPU（カレントプロセスが使ったCPU時間）を表示しない。

●ps

目的：プロセスの状態を報告する。

構文：ps ［オプション］［端末番号］［プロセスID］

オプション

a	自分以外のユーザー名についても表示する。
u	ユーザー名と開始時刻を表示する。
x	制御端末のないプロセスについても表示する。
w	出力の1行あたりの幅を広げる。
h	ヘッダを出力しない。
r	実行中のプロセスだけを表示する。
l	詳細を表示する。
f	プロセスの親子関係をツリー形式にして表示する。
端末番号	txxで指定をする。xxに端末番号を指定する。
プロセスID	表示するプロセスID

●whoami

目的：現在のユーザー名を表示する。
構文：whoami

●last

目的：ログインしたユーザーの記録を一覧表示する。
構文：last ［オプション］［ユーザー名...］

オプション

-《数値》	表示させる行数。
-n《数値》	表示させる行数。
-f《ファイル》	/var/log/wtmpでなく、《ファイル》から記録を読み取る。

●vmstat

目的：仮想メモリの使用状況を報告する。
構文：vmstat ［-n］［間隔［回数］］

オプション

-n	最初の1回だけヘッダを表示する。
間隔	更新するまでの秒数
回数	表示する回数

●df

目的：空きディスク領域を報告する。
構文：df ［オプション］［ファイル名］

オプション

-i	ブロック単位でなく、inodeの使用状況で表示する。
-k	1Kブロック単位で表示する。
-h	わかりやすい単位にして表示する。

●du

目的：ディスクの使用量を報告する。
構文：du ［オプション］［ファイル名］

オプション

-a	ディレクトリだけでなく、すべてのファイルについて表示する。
-b	バイト単位で表示する。
-c	すべての引数について集計したあと、合計を出力する。
-k	Kバイト単位で表示する。
-h	わかりやすい単位にして表示する。
-s	引数で指定したファイルの総計だけを表示する。
-x	別のファイルシステムにある分を集計しない。

●kill

目的：プロセスにシグナルを送る。
構文：kill ［オプション］ プロセスID

オプション

-s《シグナル》	《シグナル》で指定したシグナルを送る。

●mount

目的：ファイルシステムをディレクトリツリーにマウントする。
構文：mount ［オプション］ デバイス名

オプション

-a	/etc/fstabに記述されているすべてのデバイスをマウントする。
-n	マウントするとき/etc/mtabファイルに書き込まない。
-t《タイプ》	ファイルシステムのタイプを指定する。
-r	リードオンリーでマウントする。-o roと同じ。
-w	書き込み可能なモードでマウントする。-o rwと同じ。
-o《オプション》	マウントオプションを《オプション》で指定する。

●umount

目的：ファイルシステムをディレクトリツリーから取り外す。
構文：umount ［オプション］ デバイス名

オプション

-a	/etc/mtabに記述されているすべてのファイルシステムをアンマウントする。
-n	取り外すときに/etc/mtabファイルに書き込まない。
-r	リードオンリーで再マウントをする。
-t《タイプ》	-aオプションを指定したとき、アンマウントするタイプを限定する。

●useradd

目的：ユーザーを追加する。
構文：useradd ［オプション］ ユーザー名

オプション

-u《数値》	ユーザー番号を《数値》で作成する。
-g《数値》	グループ番号を《数値》で作成する。
-G《グループ》	指定された《グループ》に登録する。カンマ（,）で区切って複数のグループを指定できる。
-d《ディレクトリ》	ホームディレクトリを《ディレクトリ》で作成する。
-s《シェル》	ユーザーが使うシェルを《シェル》で作成する。
-c《コメント》	コメントフィールドを《コメント》で作成する。

●groupadd

目的：グループを追加する。
構文：groupadd ［オプション］ グループ名

オプション

-g《数値》	グループ番号を《数値》で作成する。
-o	gオプションで指定した《数値》の重複を許可する。

●usermod

目的：ユーザー情報を変更する。
構文：usermod ［オプション］ ユーザー名

オプション

-u《数値》	ユーザー番号を《数値》にする。
-g《数値》	グループ番号を《数値》にする。
-G《グループ》	指定された《グループ》に登録する。カンマ（,）で区切って複数のグループを指定できる。
-d《ディレクトリ》	ホームディレクトリを《ディレクトリ》にする。
-s《シェル》	ユーザーが使うシェルを《シェル》にする。
-c《コメント》	コメントフィールドを《コメント》にする。
-l《名前》	ユーザー名を《名前》に変更する。

●userdel

目的：ユーザーを削除する。
構文：userdel ［オプション］ ユーザー名

オプション

-r	ユーザーのホームディレクトリも削除する。

●groupmod

目的：グループ情報を変更する。
構文：groupmod ［オプション］ グループ名

オプション

-g《数値》	グループ番号を《数値》にする。
-o	gオプションで指定した《数値》の重複を許可する。
-n《名前》	グループ名を《名前》に変更する。

●groupdel

目的：グループを削除する。
構文：groupdel グループ名

●dmesg

目的：システム起動時のログの表示。
構文：dmesg

●shutdown

目的：システムのシャットダウンをする。
構文：shutdown ［オプション］［時間］［メッセージ］

オプション

-h	システムを停止する。カーネルがAPM機能をサポートしている場合は、電源も切る。
-r	システムをリブートする。
-f	リブートするとき、ファイルシステムをチェックしないようにする。
-q	デフォルトのメッセージをユーザーに送る。
-s	シングルユーザーモードでリブートする。

停止するまでの時間を指定する。省略した場合は2分後に実行される。

now	すぐに実行する。
hh:ss	hh時ss分に実行する。
+n	n分後に実行する。

現在ログインしているユーザーに出力するメッセージを指定する。

シェルを使おう

コマンドを使うために、まずはシェルのbashを使いこなせるようになろうではないか。

文：山岸典将
Text : Norimasa Yamagishi

最近のLinuxはセットアップ完了後にリブートするとX Window Systemが立ち上がり、シェルをまったく意識しなくてもある程度使えるようになっている。しかし、ちょっと高度な操作を行おうとした場合には、どうしてもシェルを利用せざるを得ない。ここでは、Linuxのほとんどのディストリビューションで標準として採用されているbashの使い方を中心に、解説していく。

シェルとはなにか

読者の中には、今までWindowsやMacintoshしか使ったことがなく、Linuxを使い始めたばかりの方、LinuxもXからログインし、すべてをX上のGUIで行っている方もいるかもしれない。このような、GUIベースの操作ができるようになったことが、Linuxが普及した理由のひとつであることは間違いない。もちろん筆者もGUIの便利さを否定する気はないし、これからもGUIはどんどん使いやすくなっていくだろう。しかし、Linuxの本来の力を引き出すには、シェルの活用は欠かせないものなのだ。

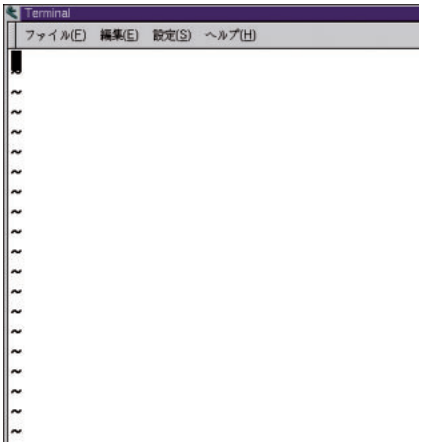
では、シェルとはなんだろうか。簡単にいえば、シェルはプログラムの実行のためのインターフェイス、そして簡単なプログラミング言語というのがその答だ。ユーザーによるコマンドラインからの入力は、まずシェルが受取り、それを解釈して、コマンドを実行する。

UNIXのシェルといえば、昔はshかcshだったが、今はそれらのシェルを元にしたbash、tcsh、zshといった高機能なシェルが開発されている。ここでは、Linux標準のbashを例にとって解説する。

コマンドライン編集

グラフィカルログインをしている場合など、Xでシェルを利用するには、ターミナルエミュレーションソフトを使う。メニューの中から「Xterm」、「kterm」、「Gnomeターミナル」、「日本端末」といったものを実行すればよい（**画面1**）。テキストログインならば、ログインすればもうそこがシェルの画面だ。

シェルが立ち上がると、最初に表示されるのは“［nor@mr2 nor］$”といった文字だ。これは、シェルが「コマンドの入力を受け入れる」ことを意味しているプロンプトだ（**画面2**）。プロンプトは現在の状況を簡単に示している（**図1**）。実際には、これらの文字列は設定によって変更することができる。



画面1　メニューからターミナルエミュレーションソフトを選ぶ
Laser5 Linuxではメニューの「ユーティリティ」から「日本端末」を選択するとktermが起動する。

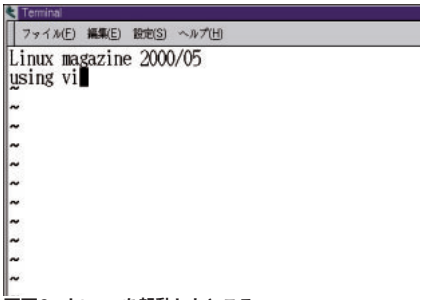
ここで、さまざまなコマンドをタイプするわけだが、シェルにはいくつかの編集機能が用意されている。BackSpaceキーで直前の文字を消し、カーソルキーでコマンドラインに打ち込んだ文字列の中を移動することができることに気が付いている人も多いだろう。しかし、コントロールキー（Ctrlキー）やメタキー（Altキーになっていることが多い）との組み合わせで、さらに強力な編集機能を使うことができる。

特に覚えておくと便利なのは、**表1**の機能だ。

これらの機能を使えば、タイプミスやコマンドの引数の順番を間違えたときも、簡単に修正できる。このほかにも、たくさんの編集コマンドが用意されており、それらは“man bash”とタイプして読むことができるオンラインマニュアルに詳しく記載されている。

また、Tabキーによるファイル名補完機能も、便利な機能のひとつだ。まず、コマンドラインで“ls -l .bash_p”までをタイプしてみよう。

```
$ ls -l .bash_p
```



画面2　ktermを起動したところ
シェルが立ち上がるとまずプロンプトが表示される。

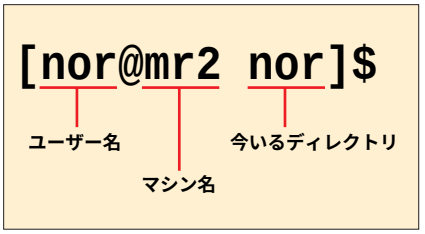


図1　プロンプトの意味する内容の例

ここで、Tabキーをタイプする。

```
$ ls -l .bash_profile
```

“ rofile ”という文字列が補完されたのがわかるだろう。これは、“ .bash_p ”で始まるファイルがそのディレクトリには.bash_profileしか見つからなかったので、Tabキーを押すことによってbashがファイル名を補完してくれたのだ。

では、複数のファイルが見つかった場合はどうなるのだろう。“ ls -l . ”とタイプして、Tabキーをタイプしてみてほしい。

```
$ ls -l .
```

なにも起こらない。そこでもう1度、Tabキーをタイプしよう。“ . ”で始まるファイルがすべて表示されたはずだ。Tabキーによる補完機能は**図2**のようになっている。

コマンド履歴

実際にシェルで作業をしていると、以前と同じ入力を繰り返したいときがよくある。bashは以前入力したコマンドを回数分覚えているので、そのようなときは、履歴機能を使うと、すばやくコマンドを入力することができる。

筆者の場合、もっともよく使うのが、Ctrlキーと組み合わせた履歴の呼び出しだ。コマンドラインで、Ctrl－Pをタイプしてみよう。直前に入力したコマンドが表示されるはずだ。ここで再度、Ctrl－Pをタイプすると2回前に実行したコマンドが表示される。こうして、以前に入力したコマンドを呼び出すことができる。もし、いきおい余って過去に戻りすぎてしまったら、Ctrl－Nをタイプすれば履歴を1回分ずつ戻ってくる。↑↓（上下カーソルキー）でも、履歴を呼び出すことができるので、こちらを利用しても構わない。

オプション名	説明
Ctrl－B	カーソルを左に移動
Ctrl－F	カーソルを右に移動
Ctrl－A	カーソルを行の先頭に移動
Ctrl－E	カーソルを行の終りに移動
Ctrl－H（Back Space）	カーソルの左の1文字を削除
Ctrl－W	カーソル位置の単語を削除
Ctrl－K	カーソル位置から行末までの文字列を削除
Ctrl－U	行頭からカーソル位置までの文字列を削除
Ctrl－Y	直前に削除した文字列を貼り付け

表1　bashのコマンドライン編集機能

しかし、しばらく前に入力したコマンドを呼び出すには、もっとよい方法がある。履歴の内容を検索するCtrl－Rコマンドだ。

コマンドラインでCtrl－Rをタイプすると、プロンプトが以下のように変わる。

```
(reverse-i-search)`'):
```

ここで、以前入力したコマンドの一部の文字列をタイプしてみよう。以下のように以前に入力されたコマンドが表示されるはずだ。ここで、Enterキーをタイプすれば、そのコマンドを実行できる。もし、利用したい履歴が別のものであれば、さらにCtrl－Rをタイプすることによりさかのぼって検索することができる。

```
(reverse-i-search)`mv'): mv foo.txt
/tmp
```

さらに、ここでCtrl－PやCtrl－Rで呼び出した履歴は、前に書いたコマンドライン編集機能で編集することもできる。

また、“！”を利用した、簡単な履歴機能もある。以前使ったコマンドをキチンと覚えているならば、

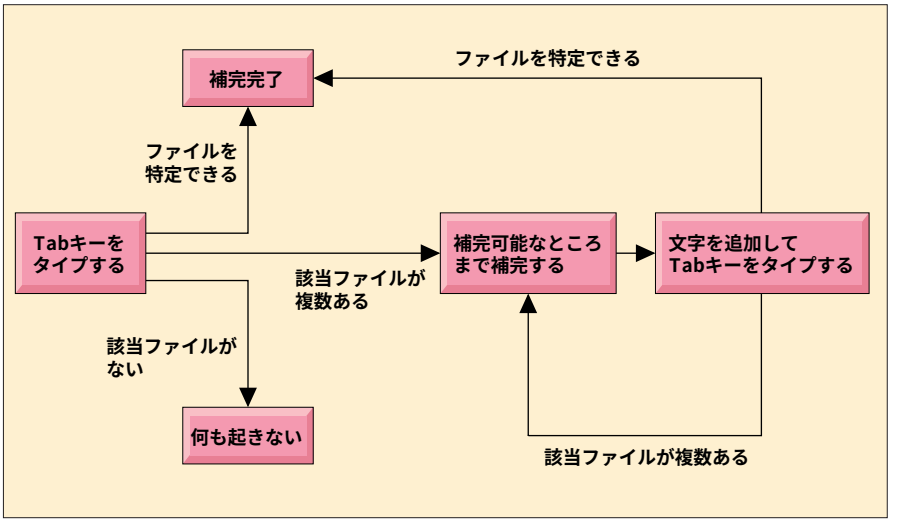


図2　Tabキーによる補完機能

```
$ !str
```

とタイプすることで、履歴の中にある“ str ”から始まるコマンドを再び実行する。直前のコマンドに限っては“ !! ”とタイプするだけで実行する。

また、シェルで作業中には、直前のコマンドの引数をもう1度、別のコマンドの引数として利用したいときもあるだろう。そんな場合には“ !* ”を使う。たとえば、あるファイルを編集して、それをコンパイルしたいときなどは、まず、

```
$ vi foo.c
```

で、編集し

```
$ cc !*
```

とタイプすることにより、“ !* ”の部分が“ foo.c ”に展開され、“ cc foo.c ”というコマンドを与えたのと同じになる。これにより、foo.cをコンパイルすることができる。

エイリアス

シェルでは、あるコマンドに対して別の名前を割り当てることができる。これをエイリアス（別名定義）といい、指定は以下のようにする。

```
$ alias lsl='ls -alF'
```

これにより“ lsl ”とタイプすることにより“ ls -alF ”が実行されるようになる。現在定義されているエイリアスをすべて表示するには、なにも引数をつけずにただ“ alias ”とタイプすればよい。

もし、定義されているエイリアスを無効にしたい場合には、“ unalias ”コマンドで無効化できる。

```
$ unalias lsl
```

実際には、エイリアスの設定は、あとで述べる初期設定ファイルに記述しておくことがほとんどだろう。

変数とはなにか

変数は、シェルやその他のプログラムが実行時に利用するもので、ふつうはユーザーの名前や、ターミナルの設定、言語環境の設定、そして個別のプログラムのための設定など、さまざまな情報が数値や文字列としてあらかじめ設定されている。また、変数にはシェル変数と環境変数の2つがある。

まず、シェルから“ set ”とタイプしてみよう。そのシェルで設定されている変数が大量に表示されるはずだ。

シェル変数の定義方法は、単に変数とその値を“ = ”で結べばよい。

```
$ FOO=txt
```

これで、変数FOOに“ txt ”が設定される。そして、その変数は、変数名の前に“ $ ”をつけることで利用できる。引数に指定された文字列を出力するechoコマンドで試してみよう。

```
$ echo $FOO
txt
$ ls
aaa.txt  bbb.tif  ccc.txt  ddd.tif
$ ls *$FOO
aaa.txt  ccc.txt
```

最後のlsコマンドでは、$FOOが“ txt ”に展開され、ls *txtというコマンドになったのがわかるだろう。

最低限必要な変数に関してはシステムに最初から設定されている。シェルを使い始めたばかりのうちはあまり変数を使うこともないだろうが、覚えておいたほうがよい変数もある。

そのひとつが、PATH変数だ。このPATH変数は、入力されたコマンドをシェルが探しに行く場所（パス）を設定している。つまり、ここで設定されていない場所にあるコマンドを起動するには、コマンドがある場所も指定しなくてはならないということだ。存在するはずのコマンドを入力したのに、「command not found」となってしまう場合には、そのコマンドが存在する場所がPATH変数に設定されていない可能性がある。

また、このPATHに関しては、現在いるディレクトリ（カレントディレクトリ）は含めないのが一般的なので注意しよう。特にMS-DOSをよく知って

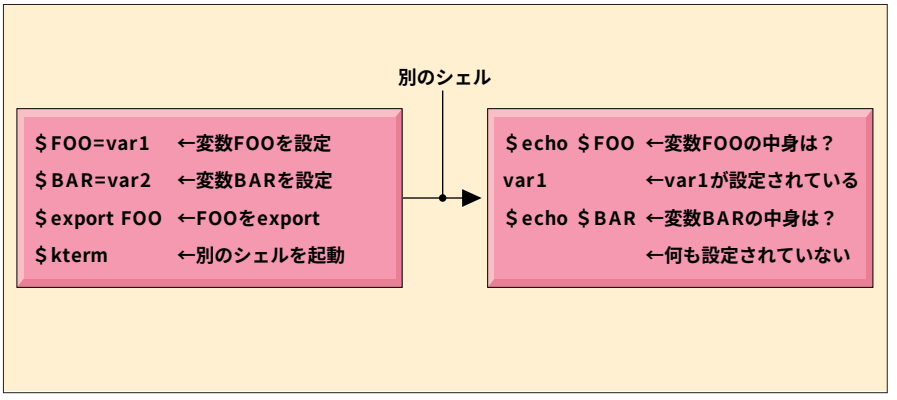


図3　変数とexport

いるユーザーが引っ掛かりやすい。カレントディレクトリにあるコマンドを実行するには、コマンド名の前にカレントディレクトリ意味する“ ./ ”をつける。

```
$ ./command
```

なお、基本的にシェル変数の設定は、そのシェルのみで有効となり、別のシェルには影響しない。ということは、X上で複数のターミナルを開いていた場合、あるターミナルで変数を変更しても、別のターミナルではその変数は変更されない。そのうえ、そのシェルから起動したプログラムも、シェル変数を参照することはできない。

しかし実際には、変数を利用してさまざなプログラムの設定を行いたい場合が多い（日本語環境の設定も変数で行っている）。そこで、シェル変数をそのシェルから起動したプログラムに引き継がせるための、“ export ” というコマンドが存在する。

```
$ export FOO
```

exportされたシェル変数は、環境変数となり、そこから起動された（シェルを含む）プログラムに引き継がれることになる（**図3**）。実際にこのexportがどのように使われるかについては、次の「bashの初期設定」で解説しよう。

bashの初期設定

最後にシェルの初期設定の方法について解説しよう。

bashの初期設定ファイルは、/etc/profileと、各ユーザーのホームディレクトリにある.bachrc、.bash_profileの3つだ。この3つのファイルの違いは、まず、/etc/profileにはシステム全体としての設定が書かれており、残りの2つは、ユーザーによる設定ファイルだということ。そして.bashrcがシェルの起動時に読み込まれ、/etc/profile、.bash_profileがログイン時に読み込まれるというところにある（**図4**）。

つまり、Xでターミナルを開くたびに.bashrcは実行されるが、/etc/profile、.bash_profileはログイン時にのみに実行される。ということは、ユーザーは.bashrcにはエイリアスのようなシェルの基本的な動作の設定を書いておき、.bash_profileにはログイン時にのみ行いたい処理を書いておけばよいわけだ。なお、実際には最初から用意された.bash_profileには.bashrcの設定もログイン時に読み込むように設定されている。

.bash_profileに登場するのが、先ほどのexportだ。一般に変数に関しては.bash_profileで設定され、exportされることになる。.bash_profileはログイン時に読み込まれるので、それ以降に起動されたプログラムに関しては、すべて.bash_profileでexportされた変数が引き継がれることになるからだ。

・bashの初期設定のポイント

1. エイリアスは.bashrcに書く
2. 環境変数は.bash_profileに書いて、exportする。

以上、簡単なシェルの使い方について解説してきた。シェルのさらなる活用法については、ぜひ本誌連載の「賢く使うUNIX」を読んでいただき、「スマートなUNIX使い」を目指してほしい。

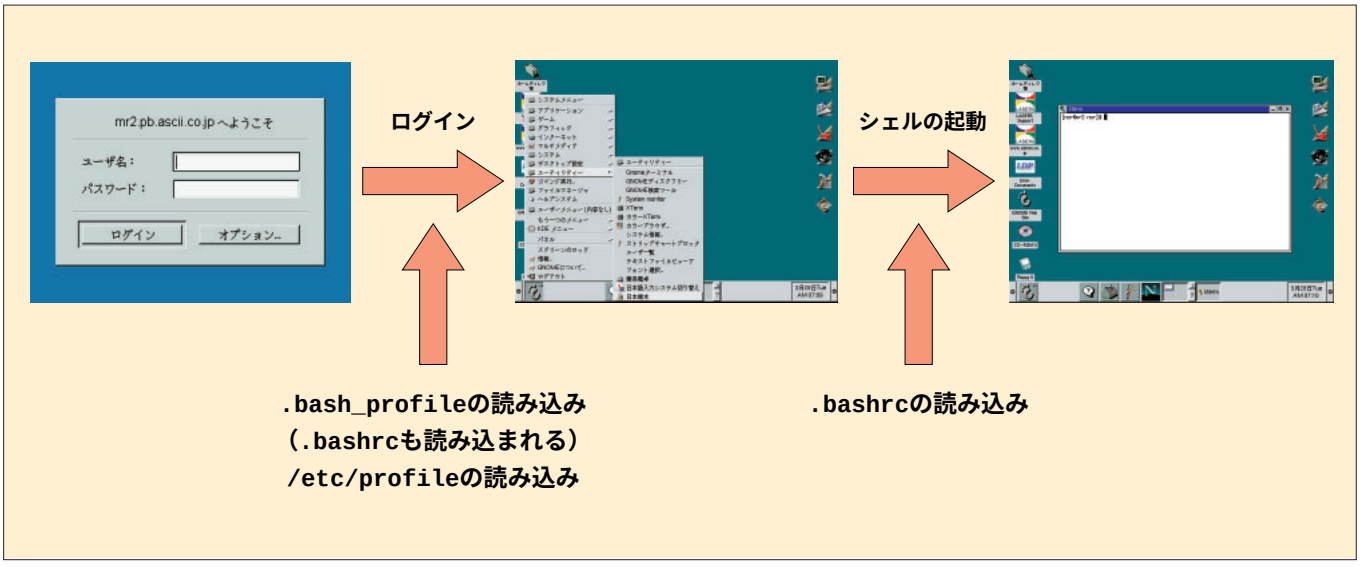


図4　bash設定ファイルの読み込みタイミング

vi、最初の1歩

UNIX界でもっともベーシックなエディタvi。これを覚えれば設定ファイルももう恐くない。

文：山岸典将
Text : Norimasa Yamagishi

Linuxを触り始めると、まずやってみたくなるのが、さまざまな設定ファイルの書き換えだろう。そこで必要になるのがエディタだ。もちろん、ディストリビューションや、デスクトップ環境によっては、オリジナルのエディタが入っていたりすることもあるだろうが、Linux上で使われる代表的なエディタといえば、viとEmacsだ。

ここでは、そのうちのひとつ、viの使い方を説明する。

なぜviを使うのか

まず、Emacsはインストール時に選択されていないと導入されないこともあるが、viが入っていないシステムというのはまずない。また、最近のRed Hatなどでは、viは/binの下に入っていることもあり、/usrパーティションがマウントできないような緊急時にも使うことができるのだ（Emacsは/usr/binの下に入っている）。さらに、Emacsなどに比べると起動が速いので、ちょっと作業をしたくなったときなどに便利だ。

といっても、viは、メニューは出ないし、何か間違った操作をしても、最低限の警告しかしてくれない。もともとのUNIXの寡黙さを体現しているといってもよいエディタだ。しかし、実は慣れるとその独特の操作体系がとても使いやすく感じるようだ。

ただ、ここではviのすべてを紹介することはとてもできない。便利な使い方を追究するというよりも、最低限必要な操作方法を覚えるのが目的だ。

モードという概念

今までWindowsのエディタを使ってきた人の多くが、viは非常にとっつきにくいエディタだと感じるようだ。それはviにはモードの概念があるからだろう。viには文字を入力するための「入力モード」と、編集コマンドをタイプするための「コマンドモード」がある。入力モードでは、単に文字を打ち込んでいくこと以外はできず、カーソルの移動、文字の消去といった作業は、コマンドモードで行うことになる（正確には［Ctrl］－［h］で現在の入力モード内で入力した直前の文字だけは消去することができる）。

入力モードへ入るには［i］キーをタイプし、コマンドモードに戻るにはEscキーをタイプする（**図1**）。

それでは、早速viを立ち上げてみよう。

起動と終了

viの起動方法は、想像通り、コマンドラインでviのあとに編集するファイル名をタイプするだけだ。

```
$ vi foo.txt
```

すると、**画面1**のような状態で立ち上がる。

立ち上げてすぐの状態は、コマンドモードになっている。そこでまず、［i］をタイプして入力モードに入り適当な文字をタイプしてみよう。気をつけてほしいのは、viはコマンドの大文字と小文字を区別するということ。つまり［i］と［I］は別のコマンドになってしまうということだ。

入力モードの時は最下行に、

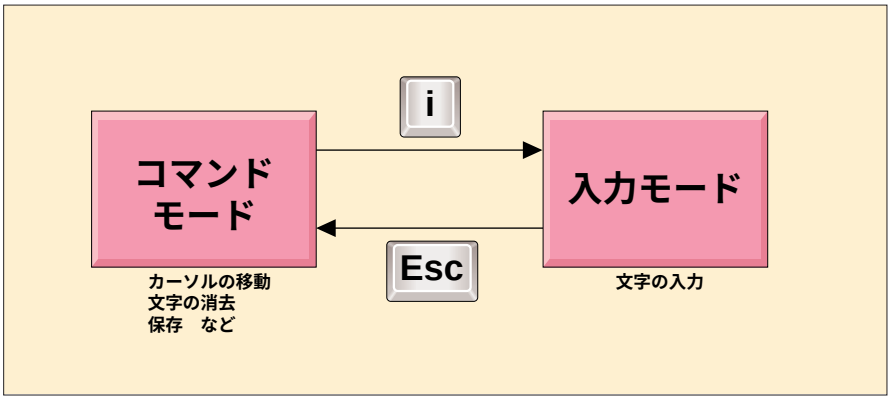
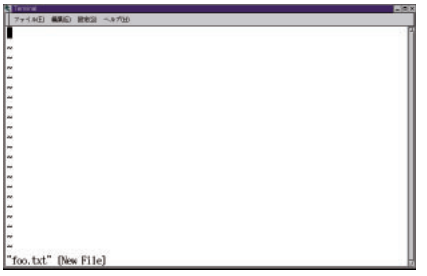
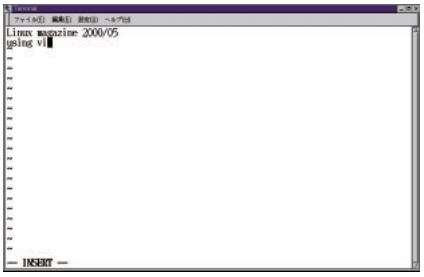


図1　入力モードとコマンドモード



画面1　立ち上げ直後のvi



画面2　入力モードのときは最下行に［INSERT］と表示される

“INSERT”の文字が表示される（**画面2**）。もっとも、これは本来のviの機能ではない。実はこのviはVIMというviクローンで、モードの表示はVIMの拡張機能だ。Red Hat Linux 6.0、LASER5 Linux 6.0 Rel.2、Turbo Linux 4.5などではこのVIMをviとして使っている。もし使っているviがVIMでない場合、モードの表示はされない。

では、できあがったファイルを保存しよう。今は入力モードなので、Escキーをタイプしてコマンドモードに戻る。そして保存するには、［:w］とタイプしよう（**画面3**）。

保存したら、続きを打ち込むことにしよう。［i］とタイプして、なにか1文字タイプしてみると、最後の1文字の前にタイプした文字が挿入されてしまう。実は、ここで続きをタイプするのに［i］コマンドは使えない。

先ほど説明した［i］コマンドは“insert”の略で、カーソルのある部分に文字を挿入するコマンドなのだ。それでは、文章の最後に文字を追加するにはどうすればよいのだろう。追加は“append”、そう、コマンドモードで［a］をタイプすればカーソルの後に文字を追加できる（**図2**）。

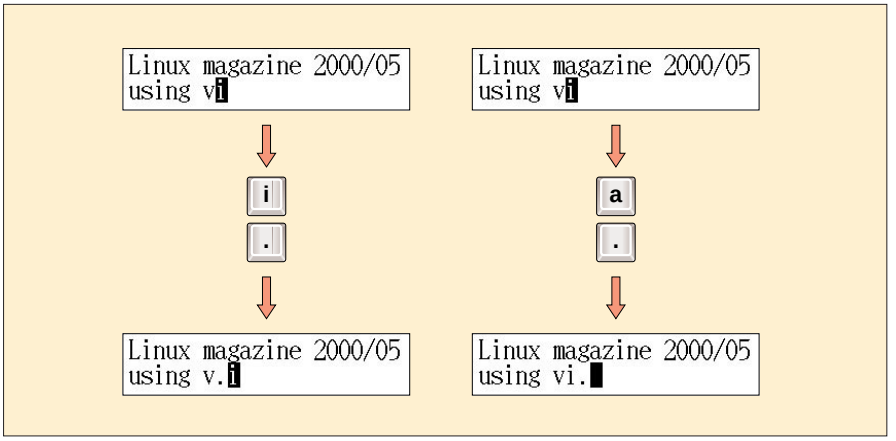


図2　［i］コマンドはカーソル位置に挿入、［a］コマンドはカーソル位置のうしろに追加する

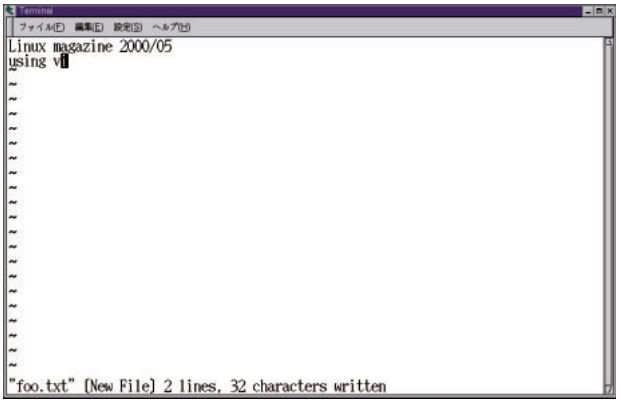
では、まず今間違って入力してしまった文字を消そう。まだ、入力モードにいるなら、［Ctrl］－［h］をタイプすると直前にタイプした文字を消すことができるが、ここでEscキーをタイプして、コマンドモードに移ってしまった場合は、どうするか。カーソル位置にある文字を消すには［x］をタイプすればよい。

間違った文字を消したら、コマンドモードで［a］を押して、続きを打ち込もう。

さて、なにか打ち込んだら、今度は、いったん終了する。まず、コマンドモードに戻る。そして今まで編集したファイルをセーブして終了するなら［:wq］＋Enter、編集結果を破棄するなら［:q!］＋Enterとタイプすればよい。もし、なにも編集していなければ［:q］＋Enterだけで終了する。

カーソルの移動

さて、起動と終了方法がわかったら、次は、カーソルの移動方法だ。コマンドモードで［h］［j］［k］［l］の4つのキーを押すと、それぞれ「左」「下」「上」「右」への移動になる（**図3**）。これは、頭で覚えようとしてはいけない。



画面3　画面の下に保存したファイルに関する情報が表示される

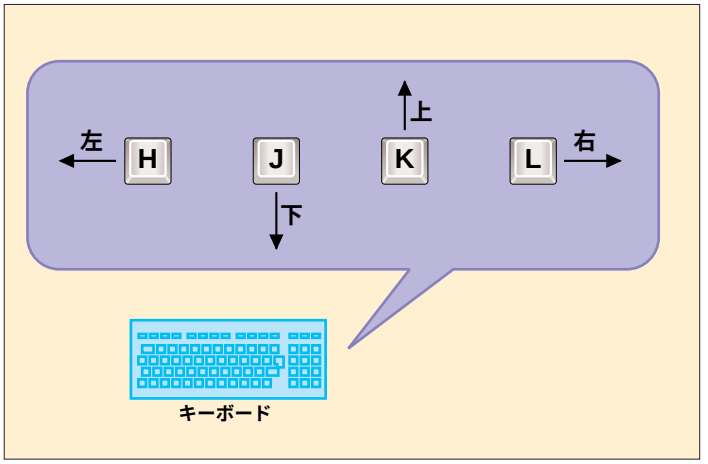


図3　［h］［j］［k］［l］キーによるカーソル移動

実際にキーボードを触ってみれば、この4つのキーが並んでいて、すぐに覚えられるはずだ。ただし、やってみればわかるが、この移動は行頭、行末を越えて移動することはない。行頭で、いくら［h］をタイプしても、カーソル位置はそのままだ。

なお、viの行とは、ワープロなどとは違い、改行位置までを1行としている。表示が2行でも、その間に改行が入っていなければ1行ということだ。

とりあえず適当なファイルを作って練習してみよう。

```
$ ls -l / > root.txt
$ vi root.txt
```

もっとも、これだけしか知らないと、移動の距離が長いときにはキーを連打しなくてはならなくなってしまう。そんなときは移動コマンドを使おう。移動コマンドの前にどれだけ移動したいかを指定できるので便利だ。たとえば、［10h］とタイプすれば、10文字分左に移動する。このコマンドの前の数字の指定は、［10x］とタイプすると10文字消去されるといったように、移動以外のコマンドにも有効だ。

残った必修コマンド

さて、カーソルの移動方法がわかれば、あとは［i］［a］で文字を追加し、［x］で消去していけば、ある程度の編集ができるはずだ。しかし、今まで覚えたコマンドではどうしてもできないことが、いくつかある。

まず、2つの行をつなぐコマンドだ。ほかのエディタでは、カーソルが行末の文字がない部分に移動するので、そこを削除すれば、次の行がつながってくれるが、viではカーソルをそこに移動することはできない。

そこで、使われるのが大文字の［J］コマンド（join）だ。［J］をタイプすると改行が削除され、次の行が行末にくっつくことになる。この時注意してほしいのは、今いる行と、追加された行の間にはスペースが1つ入るということだ。このスペースは適宜［x］で削除しよう。

さらに、日常的にエディタを使ううえでは、カット＆ペーストも必修項目だろう。

行を削除する場合は、［dd］コマンドを使う。そして［dd］コマンドをタイプすると、削除された行はviの内部に保存される（Windowsのカット、［Ctrl］－［x］と同じだ）。その行を別のところに貼り付けたいならば、その場所にカーソルを移動して大文字の［P］をタイプすると、カーソルのある行に削除した行が挿入される。ここで［P］の代わりに小文字の［p］を使うと、カーソルのある行の下に挿入されることになる。

もし、もとの行を削除せずにコピーしたい場合は、［yy］コマンドを使う。コピーしたい行で［yy］をタイプし、貼り付けは同じく［P］か［p］を使う（**図4**）。

これら［dd］［yy］のコマンドも、コマンドの前に削除、コピーしたい行数を指定することができる。そして複数の行が削除されれば、もちろん［P］／［p］コマンドで複数の行が貼り付けられる。

なお、［P］／［p］コマンドでは［x］で削除した文字を貼り付けることもできる。この場合、［P］コマンドではカーソル位置に、［p］コマンドではカーソル位置の後に貼り付けられることになる。

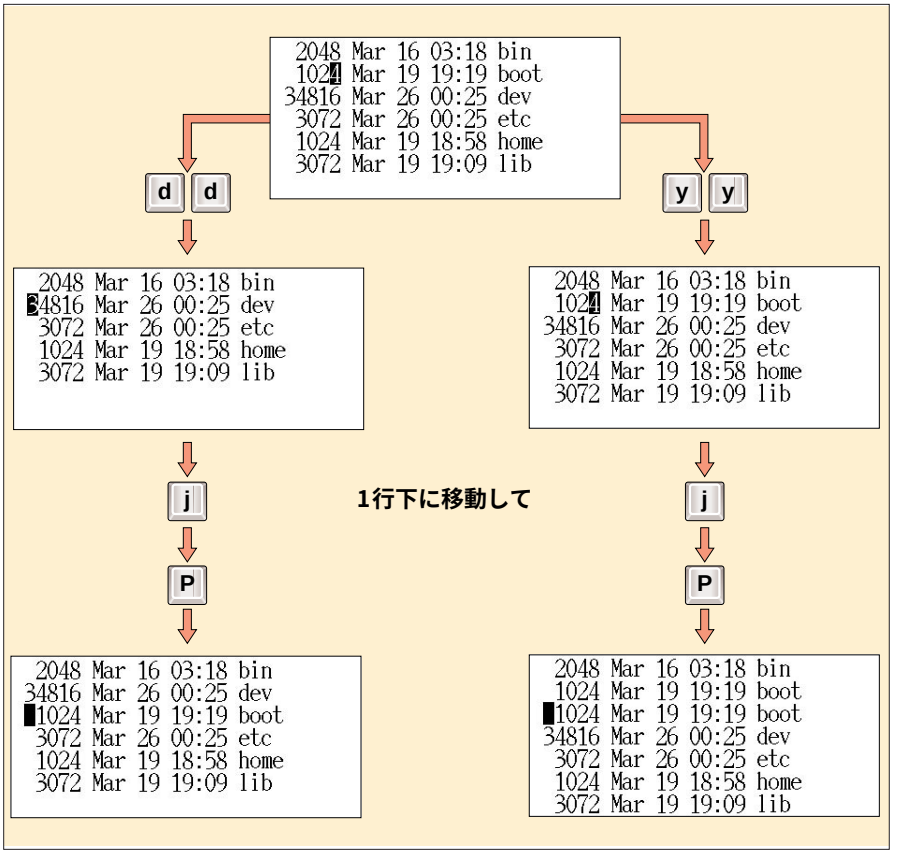


図4　カット／コピー＆ペースト

アンドゥと繰り返し

ここまでで覚えた操作を組み合わせれば、自由自在にとまではいかないだろうが、なんとか必要なファイルを編集することはできるはずだ。といっても、これだけではとても効率的に編集はできない。

次に覚えたいのが、アンドゥと繰り返しコマンドだ。

［u］コマンド（undo）を使うと直前の操作を取り消すことができる。［u］が有効なのは今まで出てきた中では、［x］、［J］、［dd］、［y］、［P］、［p］コマンド、そして直前の入力モードでの入力だ。さらにある行を編集していて、その行から別の行に移動していない場合は、［U］コマンドでその行に対して行った変更のすべてを取り消すことができる。

アンドゥがあれば、繰り返しコマンドもある。［.］をタイプすると直前に行った操作を繰り返すことができる。［.］コマンドが効く操作も［u］コマンドと同じだ。「12345」と入力し、コマンドモードに戻って、［.］を4回タイプすれば「1234512345123451234512345」と入力される。

検索

設定ファイルなどを編集していると、設定項目の文字列を検索したくなることは多い。

viの検索コマンドは［/］だ。［/］をタイプすると最下行に“ / ”が表示されるので、それに続けて検索したい文字列とEnterキーをタイプする（**画面4**）。もし、文字列が存在しない場合は“ Not found ”と表示される。同じ文字列を続けて検索したい場合は［n］をタイプする。逆方向に検索したい場合は［N］をタイプすればよい。最初から逆方向に検索したい場合には［?］をタイプして、検索したい文字列そしてEnterキーをタイプする。

文字列の置換は、［:］、置換開始行番号、置換終了行番号、検索する文字列、置換する文字列の指定、オプションの指定で行う。**図5**のコマンドではファイル中のすべての“ foo ”に対して“ bar ”への問い合わせ置換を行う。

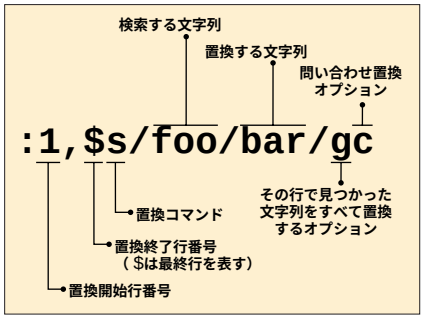
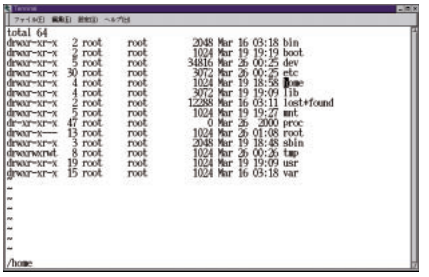


図5　問い合わせ置換のコマンド例

高度な移動

大きく移動したいときには数字のあとに移動コマンドをタイプする、と書いた。が、実際にはファイルの内容を眺めながらスクロールさせていきたい、



画面4　［/］で“ home ”という文字列を検索する

という場面も多い。スクロールさせるためのコマンドは、Ctrlキーとアルファベットを組み合わせたもので、そのスクロール量に応じていくつかある。**図6**を見てほしい。

移動したい行番号がわかっているときには、［G］コマンドで移動できる。10行目に行きたいなら［10G］とタイプすればよい。単に［G］とだけタイプすると、ファイルの最終行にジャンプする。そして、その行の番号を知りたい場合には、［Ctrl］－［g］をタイプする。

さて、以上駆け足だったが基礎的なviの使い方を紹介した。これらは、viの機能のほんのさわりだけだ。しかし、ここで書いたことを覚えていれば、ちょっとしたファイルの編集をするには十分だろう。

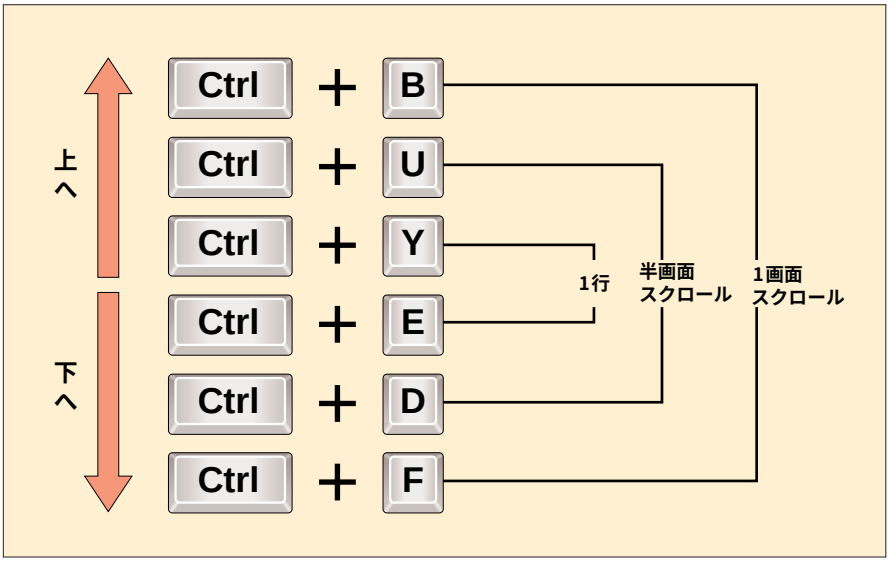


図6　スクロールコマンド

ディスク環境大強化！

0円でできるRAIDハードディスク

文：村田　勉/BiTmark
Text : Tsutomu Murata/BiTmark
photo : Shuichi Mito(Dee)

ハードディスク安くなってます！

メモリやハードディスクなど、パソコンの周辺機器や本体の価格の低下は驚くほどの速さで進んでいる。私がコンピュータに興味を持った時は、まだまだNECの勢力が強くてPC-9801はビジネスユーザー向けで、パーソナルユーザーはPC-8801を何とか買えるという感じだった。メモリも4Mバイトは夢のよう、ハードディスクなどはとんでもないという、そんな時代だ。

ちょうど10年前の資料を見てみると、1Mバイトのメモリが7万5000円、40Mバイトの外付けハードディスクが20万円を超えている。現在では、40Mバイトというのはメモリの量としても少なくなってしまい、最近のOSでは、もはや40Mバイトのメモリでは起動するのがやっとだったりする。私がコンピュータを買ったときは32Mバイトのメモリが（DIMMだったが）11万5000円した。それもたった4年前だ。それを考えるとこの数年で一気に安くなったのだろう。

■安いんだから、使わなきゃ

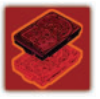
まぁ、そんな時代もあったんだぁというぐらいで気持ちを切り替えるとして、今のハードディスクの安さと来たら、店で見ても目を疑うほどになっている（表1）。安いだけではない、その容量の大きさも「だれが、そんなに使うんだよ〜」と言いたいほど大きくなっている。

しかし、容量の大きさ、値段の安さにつられてハードディスクを載せ換えてみたものの、その大きな容量のバックアップを取ろうにも、テープデバイスぐらいしか選択肢がないのが現状である。そして、バックアップを容易にとれないと知った時につきまとい始める、システムクラッシュの恐怖（そんな馬鹿な）。必要なファイルだけ取っておけばいいじゃんと言われてしまうと、それまでなのだが……。やはりシステムのバックアップも取りたいのが人情だろう。

では、考え方を変えてクラッシュしても立ち直りの早い、もしくは比較的楽なシステムはないもんだろうか？そんな都合のいい話あるわけないじゃん、と考えてみると……。スピード優先のハイエンドユーザーが、もしくは24時間止めることができないサーバなどを運用している企業などが使っているものがある。そう「RAID」（レイド）である。随分遠回りになったが、今回は「スピード優先！」「バックアップ命！」など、いろんな要望に応えることができる「RAID」システムを、最近本当に安くなったIDE ハードディスクを使って試してみようというわけである。

メーカー名	型番	容量	販売価格（単位：円）
IBM	DPTA-351500	15Gバイト	1万4400〜1万6800
	DPTA-352250	22.5Gバイト	1万7500〜1万9450
	DPTA-353000	30Gバイト	2万5100〜3万800
	DPTA-371360	13.6Gバイト	1万4700〜1万5400
	DPTA-372050	20.5Gバイト	1万7700〜1万7900
	DPTA-372730	27.3Gバイト	2万5800〜2万6800
Maxtor	91531U3	15.3Gバイト	1万3600〜1万4800
	91728D8	17.2Gバイト	1万6500〜1万7000
	92041U4	20.4Gバイト	1万5800〜1万8000
	92049U6	20.4Gバイト	1万8200〜1万8800
	92732U8	27.3Gバイト	2万1900〜2万5300
Quantum	FBLCT13.0AT	13.0Gバイト	1万2000〜1万3000
	FBLCT17.3AT	17.3Gバイト	1万4300〜1万6500
	FBLCT20.4AT	20.4Gバイト	1万6900〜1万7800
	FBLCT26.0AT	26.0Gバイト	1万9400〜2万1500
Seagate Technology	ST313620A	13.6Gバイト	1万3480〜1万3900
	ST320430A	20.4Gバイト	1万7200〜1万7900
	ST328040A	28.5Gバイト	2万900〜2万1980
Western Digital	WD136AA	13.6Gバイト	1万2500〜1万6000
	WD172AAF	17.5Gバイト	1万5800〜1万6500
	WD205AAN	20.5Gバイト	1万6500〜1万9000
	WD273BA	27.3Gバイト	2万4800〜2万6800

表1　市販されている主なハードディスク（1999年3月秋葉原にて調査、比較的大容量なものだけ）



RAIDとはなんぞや？

そもそもRAIDとは、ハードディスクなどの記憶装置を複数台用いてアクセスを分散させることにより、高速、大容量で信頼性の高いディスク装置を実現させるための技術である。

RAIDは1987年、アメリカはカリフォルニア大学バークレー校のDavid A. Patterson氏らによって提唱された。当時のハードウェアの技術者達はCPUやメモリの性能が飛躍的進歩を遂げるのに対してI/Oの性能がいっこうに向上しないことに不満を抱いていた。

特に、ハードディスクの速度を上げるための技術は、頭打ちの状態だった。ヘッドを所定の位置まで移動させるシークタイムや、回転数に依存する回転待ち時間などの機械的な動作を向上さ

せることは簡単ではなく、最速のハードディスクとローエンドのハードディスクの性能差はあまりなかった。

その後、ハイエンドハードディスクの容量が急速に増大したにも関わらず、期待するほどの速度の向上は得られなかった。そんな時、速度改善の壁を打ち破るアイデアとして提案されたのが「RAID」である。パソコン用のローエンドハードディスクをアレイ状（整列）に構成することによって、性能向上を図ったのである。

RAIDは何かの頭文字だろうということは想像していることだと思う。「Redundant Arrays of Independent (Inexpensive) Disks」が正式名称である。正式名称の分際でRAIDの「I」の部分が2通りある。もともとRAIDの論文が発表されたとき「I」は「Inexpensive」の略であった。これは、安価な（Inexpensive）なハードディスクを複数台使うことで、高価なディスクと同等の性能や信頼性を持たせようということが目標だったからだ。

しかし、RAIDが世に広まるに連れ、単体の高価なディスクはRAIDによるディスクアレイに取って代わってなくなってしまった。そのため「Inexpensive」では意味が通りにくくなり、代わりに「Independent」（独立した、自立した）が「I」の略として使われることが多くなってきている。

■どんな種類があるの？

RAIDはそれぞれの機能によって、いくつかの段階に分けることができる。最も簡単なものは複数台のディスクを1台のディスクとして認識させる（リニアともいう）。次は複数台のディスクにデータを分散させてアクセス速度を向上させる、といったものから高度なものは、書き込むデータにパリティやエラー訂正データを付加して、さらにそれらを複数台のディスクに分散させ、ディスクが1台故障してもデータを復旧してしまうものまである。

それぞれ、「RAID 0」から「RAID 5」という6段階のレベルが定義されている。機能的には「ストライピング」「ミラーリング」「パリティチェック」「ECC」「データとパリティ、ECCの分散書き込み」といった技術を組み合わせたものだ（**表2**）。

次に各レベルを説明しようと思うのだが、ここでは、よく使われるレベル「0」「1」「5」には**図**を付けておいたので参考にしてほしい。

用語	説明
ストライピング	データを小さいブロックに分割し、複数のドライブに分散させ、同時に読み書きすることによりアクセス速度の向上を図る
ミラーリング	同一データを複数のドライブに書き込む。1台が故障しても残ったドライブからデータを読み出すことができる
パリティチェック	データの排他的論理和の情報を記録しておき、読み出し時にデータとパリティを付き合わせてエラーを検出する
ECC	誤り訂正コード（ECC）によってビット単位でデータの信頼性を高める
データとパリティ、ECCの分散書き込み	データを元にして計算したパリティまたはECC情報と、データブロックを複数のハードディスクに分散して記録することで、故障してもデータを再現できる

表2　RAIDで使われる機能の説明

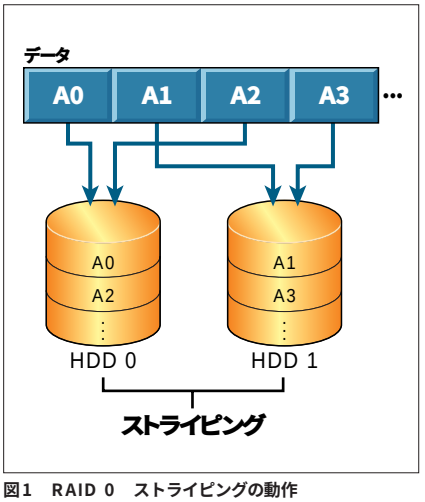


図1　RAID 0　ストライピングの動作

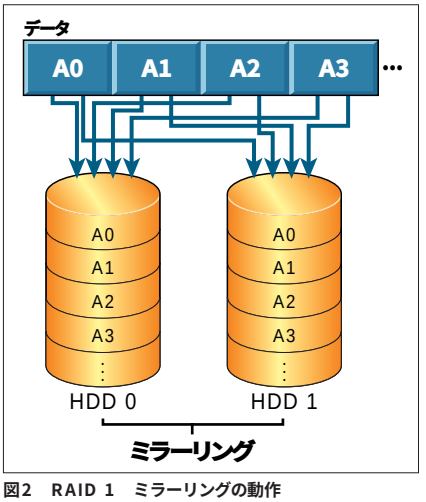


図2　RAID 1　ミラーリングの動作

• RAID 0

データをストライピング（分散化）して複数のハードディスクに記録する。連続するデータを書き込む際にブロックごとに別のハードディスクに分散するようにして、アレイを構成するハードディスクに対して同時にアクセスすることにより速度向上を図る。搭載しているハードディスクの容量すべてを使用することができるが、ブロック単位で分散されているためハードディスクの一部に障害が起きたときでも、すべてのデータが再現できなくなる可能性がある。エラー検出／データ回復のための冗長情報を持たないため、信頼性においては低いものとなる。

• RAID 1

一般にミラーリングと呼ばれている方式。アレイを構成する2台以上のハードディスクに、同一のデータをそれぞれ記録する。どちらかのハードディスクが故障しても、正常なハードディスクからデータを読み出すことが可能

である。読み書きの速度は通常、ハードディスク1台の場合と同じかそれよりも遅いことが多い。構造の単純さが好まれ、比較的多用されている方式なのだが、実効容量が1台分にしかならないことと、速度の向上が期待できないことが欠点として挙げられる。

・RAID 0+1

RAIDレベル0とRAIDレベル1を組み合わせて使う方式。RAID 0（ストライピング）のよってI/Oの性能を高め、RAID 1（ディスクミラーリング）によってデータの信頼性を高めている。4台以上のドライブで構成することによりコストはかかるものの、速さも信頼性も確保できることになる。

・RAID 2

メインメモリで一般的になっているエラー検出／訂正の技術（ECC）を応用したもので、データとは別にハミングコードと呼ばれるECCコードを生成し、ビット（もしくはバイト）レベルで複数のハードディスクに分散して記憶し、同時にそれらのデータに対してECCを付加。それらも同様にECC用のハードディスクに記憶する。これらECCを含めたすべてのハードディスクは同時かつ並列に読み書きされる。データ用のディスクのいずれかが故障した場合には、ECCによりデータをリアルタイムに再生するので、あたかも正常なように動作する。しかし、現在一般的に使用されているハードディスクには、それ自身に強力なエラーの検出／訂正機能が付加されているのと、後述するRAID 3～5と比較してECCのオーバーヘッドが大きいため、あまり採用されていない。

・RAID 3

データを複数のディスクに分割して書き込み、データ修復用に専用のパリティディスクを使用する。データは1セクタなどの大きさの単位に分割され、複数のデータ専用ディスクに書き込む。各ディスクの回転待ち時間が不均一のため最も回転待ちの長いディスクの処理時間に左右される。これを克服するために同期回転技術が必要になってくる。パリティはデータから生成され、1台のパリティ専用ディスクに書き込む。

・RAID 4

データの修復情報にはパリティ方式を採用し、パリティは、1つのパリティ専用のディスクに記録する。データの分散化をビット単位ではなくブロック単位（たとえば“nセクタ”）で行う。アレイ上のハードディスクは、それぞれ独立して動作可能。RAID 3が常に全部のハードディスクにアクセスするのに対し、RAID 4は必要とされるハードディスクのみにアクセスする。これにより、あらゆる大きさのデータ読み込み時において高速化が期待できる。しかし、書き込みの際には、書き込みが発生したハードディスクとパリティ用ハードディスクから古いデータを読み出し、更新パリティを作成後、これを書き戻すという余分なアクセスが必要である。このためRAID 3よりも処理時間が長くなる。

・RAID 5

データの修復情報にはパリティ方式を採用。データはブロック単位などで記録され、パリティもデータ用ディスクに分散して記録される。これは、RAID 4においてパリティ用ディスクがパフォーマンスのボトルネックとなっていることを解消するために考えられた機構である。複数のハードディスクに同時に読み出し、書き込みを行うため高速で、パリティによって信頼性も確保している。

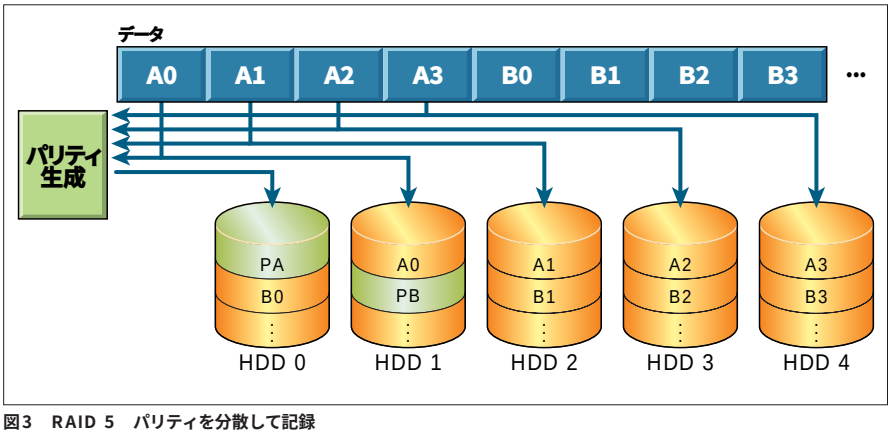


図3　RAID 5　パリティを分散して記録

RAIDの実現手法

ひとくちにRAIDといっても、大きく分けると2通りの手段が存在する。ハードウェアRAIDとソフトウェアRAIDだ。

■ハードウェアによる実現

ハードウェアRAIDとは、ホストのCPUとは別に、ディスク装置に対するインターフェイスと、コントローラで構成され、コンピュータとは独立してRAID処理を実行する。

ハードウェアRAIDにはさらに、ホストコンピュータ内のPCIバスに接続するカードタイプと、ディスクアレイ

のようにディスクドライブと一体化され、ホストとはSCSI（Ultra SCSIのようなパラレルSCSIや、ファイバーチャネル、IEEE1394のようなシリアルSCSI）で接続するタイプがある。

RAIDコントローラカードとしてはMylex、DPTなどのいくつかのメーカーがLinuxのサポートをしている（詳しくは、http://linas.org/linux/raid.htmlを参照）。最近のLinux、特にサーバ向けでは、これらのRAIDコントローラカードのドライバを標準搭載しているディストリビューションもある。しかし、自力でセットアップするには、自分の知識や技術力と相談して導入する必要があるだろう。

同じハードウェア制御でも、SCSI経由で利用できるRAIDコントローラを使った外付けRAIDシステムは、たいていの場合、コンピュータから外付けの単体のSCSIハードディスクとして見えるものが多いので、ハードウェアの取り扱いを覚えればそれほど難しいことはない。その分といってはいけないのかもしれないが、それなりの値段が付いている。

・ハードウェアRAIDの長所と短所

【長所】

1. **コンピュータからは1台のハードディスクと同様に見えるので、起動ディスクとして使用できることはもちろん、コンピュータ側で特殊な処理を必要としない。**
2. **RAIDコントローラとドライブが一体化しているので、全体としての信頼性が確保される。ホットスワップや自動的なリカバリの堅牢さが実現されている。**
3. **専用のハードウェアを持っているので、処理速度が高速。**
4. **RAIDユニットに障害が発生した場合でも、故障したドライブを新しいドライブに入れ替えることにより、ほとんどの場合、継続使用が可能。**

【短所】

1. **専用のコントローラが必要になるため高価になる。**
2. **単体のドライブを購入して増設や交換するといったようなことが制限される。**

■ソフトウェアによる実現

方法としてはSCSI、もしくはIDEインターフェイスなどを通してコンピュータに直接接続されている複数のハードディスクを、1つの記憶装置とみなしてアクセスするソフトウェアをコンピュータ上で実行させる方式である。Windows NTのソフトウェアRAIDなどが有名だ。LinuxでもソフトウェアRAIDをサポートしているので、別にソフトウェアを買ってこなくても使用することができる。

・ソフトウェアRAIDの長所と短所

【長所】

1. **ソフトウェアだけで構成できるため、専用のコントローラが不要で安価。**
2. **IDEやSCSIのハードディスクドライブを複数導入するだけで、RAIDシステムの構成が可能。**
3. **ドライブを並べて信頼性を確保する単純なミラーリングや、画像データなどの大容量データの転送速度を向上させるストライピングの構築は比較的簡単。**

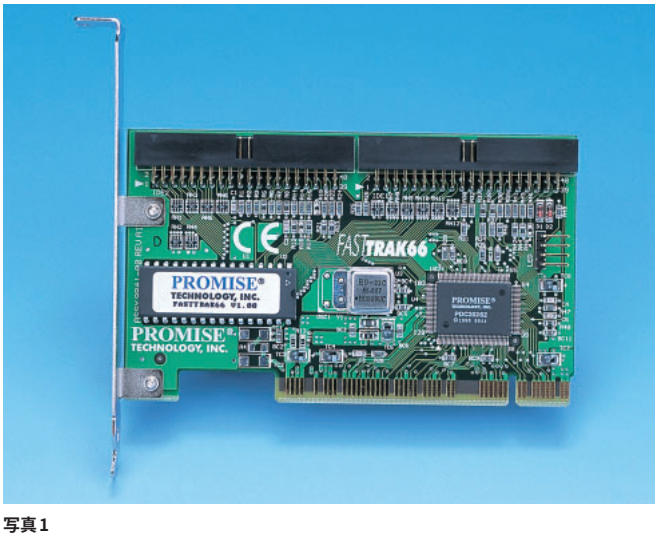


写真1
UltraATA/66インターフェースを2チャンネル備えたFASTTRAK66。外部用のインターフェイスはない。搭載チップはPDC20262で、同社の「Ultra66」カードでも同じチップを使っている。

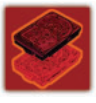
写真2
今回使用したIBMのDPTA-351500。15Gバイトの容量を持ち、回転数は5400rpmのタイプ。

【短所】

1. **OSが起動するまでRAIDが機能しないので、起動ディスク全体をRAID化することができない。起動パラメータなどの設定をRAIDで保護することができない。**
2. **システムの構成状態によって動作の安定度が決定されるため、RAID全体の信頼性を確保するために、ある程度（というかかなりの）知識が必要。**
3. **ホストコンピュータのCPUでディスク処理を行うため、アプリケーションの実行速度が低下する。ソフトウェアですべての処理を行うために、ハードウェアRAIDに比べ低性能。**

RAIDの導入を考えているのなら、それぞれの特徴と導入のしやすさ、サポートなども含めて検討しよう。高価なハードウェアRAIDをホームユースで導入する人は少ないと思うが、ソフトウェアRAIDで、まず試してみるのもよいではないだろうか。

安価なハードディスクを複数購入して自分でソフトウェアRAIDを構築すれば、外付けRAIDシステムを買うより遥かに安い。



IDEハードディスクで試しましょう

「FASTTRAK66」（**写真1**）というIDE RAIDカードがある。これはUltraATA/66に対応したIDEバスをカード上に2チャンネル持っていて、今使っているマザーボードに挿すだけで、UltraATA/66に対応したハードディスクを最大4台使うことができる。ちょっと魅力的な話だ。おまけにこのFASTTRAK66はBIOSで設定してやることで、カードによる制御という形でハードウェアRAIDを実現できる。

表3　FASTTRAK66を使ったWindows上でのベンチマーク（HDBENCH 2.61）

HDDの数	Read(KB/s)	Write(KB/s)
1台	19606	19820
2台（RAID 0、ストライピング）	37934	29290
2台（RAID 1、ミラーリング）	19731	15680

PROMISE Technology	http://www.promise.com/techsupport/Trouble/Guides.htm
kernelパッチ	http://www.kernel.org/pub/linux/kernel/people/hedrick/old/ide.2.2.14.20000124.patch.gz
kernelソース	http://www.kernel.org/pub/linux/kernel/v2.2/linux-2.2.14.tar.bz2
HDBENCH clone	http://www.enjoy.ne.jp/~gm/program/hdbench/downenjoy.html

表4　本文中で参照したWebページ

UltraATA/66対応のハードディスクでも私の今の環境より速くなるのに、そのうえRAIDを組んだらどうなるのだろう？　私だけではなく読者の方も気になるところだろう。それでは、早速試してみよう。

■FASTTRAK66をWindows 98で使ってみる

まずはFASTTRAK66とハードディスクのセットアップをする。

そして、皆さんおなじみのHDBENCHでベンチマークを計ってみた（**表3**）。ミラーリングは2台のハードディスクに同じデータを書き込んでいるだけあって、Writeが1台よりも遅くなってしまうのはしかたないことなのだろう。しかし、バックアップを取ってくれてることを考えると我慢できる範囲だろう。

ストライピングはきれいに倍とはいかないが、しっかりと速くなってくれている。こんな魅力的なカードをWindowsだけで使うのはもったいない。当然、Linuxで使いたいと思うのは当たり前のことだ。

■Linuxでは使えない？

実はFASTTRAK66を挿していればLinuxでも、Windowsのように簡単にハードディスクが見えると思っていた（恥）。しかし、やっぱりというか当たり前だよというかドライバが必要だ。そこでドライバを捜すためにメーカーであるPROMISEのホームページ（**表4**）を見てみるが、悲しくも「Red Hat Linux 6.1（Coming Soon）」と書いてあるだけだった。

どうしたものかと、いろいろ探し巡っていると、FASTTRAK66で使われているチップ「PDC20262」に対応させるパッチが配布されていた（ide2.2.14.20000124.patch.gz、**表4**）。これは！と思い早速ダウンロードしようとしたところ、LASER5 Linux 6.0で使っているカーネル2.2.5といった古めのカーネル用のパッチが見あたらない。

しかたがないので新しいカーネルソース（linux-2.2.14.tar.gz、**表4**）と一緒にパッチをダウンロード。パッチを当てコンパイルして再起動したところ、見事に認識されたのだが、何かがおかしい。2チャンネルあるはずのIDEバスが1チャンネルしか認識されない。FASTTRAK66のチャンネル2側に繋いだハードディスクが認識されないのだ。

これでは、2台のハードディスクをマスタに設定して、RAIDを組むことができない（FASTTRAK66のマニュアルにはハードディスクの設定は2台ともマスタにして、1つのバスに1台のハードディスクを繋ぐように書いてある）。そこで、残念ながらFASTTRAK66によるRAIDは、メーカーからドライバがリリースされるのを待つことにして、今回は見送ることにした。

LinuxでソフトウェアRAID

さて、FASTTRAK66でのテストに使うために、UltraATA/66に対応したハードディスクを2台用意していた。そこで、同容量のハードディスクが2台あることだし、コンピュータに最初から付いているIDEのインターフェイスを使って、RAID 0（ストライピング）を試してみた。

ご存じの方も多いと思うが、通常DOS/V機にはIDEバスが、プライマリとセカンダリの2チャンネル用意されている。メーカーもののマシンだと、両方のIDEバスにハードディスクとCD-ROMドライブをそれぞれマスタに設定して繋いでいるものなどもあるが、ここでは1つのバスにハードディスクとCD-ROMドライブがマスタ、スレーブとして繋がっていて、もう1チャンネルのIDEバスは空いていると想定している。

▼リスト1　RAID 0の/etc/raidtab

```
raiddev  /dev/md0
raid-level    0
nr-raid-disks   2
persistent-superblock  1
chunk-size  4
device  /dev/hdc1
raid-disk  0
device  /dev/hdd1
raid-disk  1
```

▼リスト2　RAID 1の/etc/raidtab

```
raiddev /dev/md0
raid-level      1
nr-raid-disks   2
nr-spare-disks  0
chunk-size      4
persistent-superblock   1
device  /dev/hdc1
raid-disk       0
device  /dev/hdd1
raid-disk       1
```

ストライピングで高速アクセス

それではとりかかろう。用意するのは（できれば）同じ容量のハードディスクとLinuxが正常に稼働しているコンピュータだ。今回私はIBMのDPTA-351500（15Gバイト）を2台用意した。今となってはそんなに大容量でもないが、8Gバイトや10Gバイトのハードディスクより容量あたりの単価に割安感がある。ソフトウェア処理なのだから、できればCPUは速いほうがよいだろう。ちなみに私のマシンは以前から使っているものだが「Pentium II/266MHz」だ（遅いじゃん！）。

■意外と簡単？

空いているIDEバス（セカンダリ）にマスタ、スレーブに設定した2台のハードディスクを繋ぐ。そして、Linuxを起動する。この起動に使ったシステムは新たにRAID用に繋いだ2台のハードディスクではなく、プライマリに繋がっているハードディスクから起動している。その後、RAID 0の設定をしよう。

セカンダリIDEのマスタ、スレーブに2台のハードディスクを繋いだので、デバイス名はhdcとhddになる。ここを間違えてしまうといくら試しても動かないことになってしまうので気を付けてほしい。

まず、fdiskコマンドでRAIDに使い

Column

/etc/raidtabファイルの設定項目

Linuxのソフトウェアを利用するときに設定するraidtabファイルで使われている設定項目について説明しておこう。

詳細は「Software-RAID-HOWTO」（http://www.linux.or.jp/JF/JFdocs/The-Software-RAID-HOWTO-1.html）に目を通していただきたい。

■persistent-superblock

昔のraidtoolsは/etc/raidtabを参照してRAIDを初期化していた。しかしこれは、/etc/raidtabが存在するファイルシステムがマウントされていることが前提になる。それではRAIDからブートを試みるときあまりよい構造とはいえない。さらにいえば、RAIDデバイスをマウントするとき、通常使われる、/etc/fstabに書いて済ますというやり方ができず、初期化スクリプトでマウントしなければならなかった。

そこでpersistent-superblockという1行を加えてRAIDを初期化すると、アレイに参加しているすべてのディスクのスーパーブロックに、RAIDの情報が書き込まれ、カーネルがRAIDのデバイス構成をディスクから直接読めるようになる。

■chunk-size

デバイスに書き込む最小のデータ量を、Kバイト単位で定義する。これにより、各ディスクへのアクセスをなるべく同時にして、スピードの向上を図ることができる。このサイズを大きくすることで大量の書き込みが発生した場合、オーバーヘッドを減らすことができるが、小さいファイルが多い場合などは、かえって1台のディスクにかける時間が長くなるので、あまり大きな設定はしないほうがよい。

■nr-raid-disks

RAIDを組むハードディスクの数を設定する。

■nr-spare-disks

RAIDレベル1以上で、冗長情報用のディスクが何台あるかを記入する。

たいパーティションを用意する。今回は2台とも1パーティションにした。まぁ、テストだからね。あとは/etcディレクトリにraidtab（**リスト1**）というファイルを用意して、mkraidコマンドを叩くだけである。

```
# mkraid /dev/md0
```

さて、どうなっただろうか？　“ガ、ガ、ガ”などと言いながらRAIDがスタートしたと思う。どうやって確かめたものか？　便利なことにこのようすはちゃんと/proc/mdstatに記録される。/procディレクトリ内のファイルでステータスがわかるというのはLinuxではお決まりの構造だ。早速mdstatを覗いてみよう。

```
# cat /proc/mdstat
```

RAIDのレベルや使用しているディスクの数が出ていると思う。では、何かを書き込んでみようか？　もちろんその前にフォーマットが必要である。RAIDだからといって特別なことはない。いつもの通りファイルシステムを作ってやればいい。そしてマウントして、容量を見てみよう。

```
# mke2fs /dev/md0
# mkdir /raid
# mount /dev/md0 /raid
# df -k
```

今回使ったハードディスクは1台が約15Gバイトで、2台のハードディスクをRAID 0（ストライピング）で1つのパーティションとして設定しているので、30Gバイトほどになるはずだ。

物理的には2台のハードディスクの2つのパーティションをマウントしているのだが、マウントポイントに、1つのデバイス/dev/md0が1つのパーティションとしてマウントされている。ストライピングされているので当たり前なのだが、2台あっても1台と認識される（というかさせている）。もちろん、パーティションを区切って使うこともできるし、マウントしてしまえば普通のハードディスクとなんら変わりはない。

ミラーリングでデータは安全

次にさっきの状態のままRAID 1（ミラーリング）を試してみよう。まずは/etcディレクトリのraidtabを書き換える（**リスト2**）。書き換えるところは、ほんの少しなので、そんなに手間はからない。さて再びmkraidコマンドを叩こう。

```
# mkraid /dev/md0
```

ここでエラーメッセージが出力された。「このディスクには、もうRAID用の設定がしてある」というものだ。同じハードディスクだから先ほどのRAID 0の設定が、まだ生きているのである。これを書き換える必要があるのだが、この場合は、mkraidコマンドにオプションを付ければよいのだが、このオプションがすごい！

```
# mkraid --really-force /dev/md0
```

Column

ソフトウェアRAIDで使うコマンド

私が確認した限りではRed Hat Linux 6.1とLASER5 Linux 6.0では、インストール時に「ワークステーション」か「サーバー」を選択していれば、raidtoolsがデフォルトでインストールされる。

「カスタム」でLinuxをインストールした場合で、raidtoolsがインストールされていない時には、バイナリCD-ROMのRedHat/RPMS/raidtools-X.XX-Y.i386.rpm（X.XXはバージョン番号、Yはリリース番号）を、

```
# rpm -i ファイル名
```

としてインストールする。

raidtoolsに含まれている主なコマンドを説明しておこう。

■mkraidコマンド

mkraidコマンドは、RAIDハードディスクの初期化を行うコマンドである。あらかじめ、/etc/raidtabファイルに、RAIDの設定を書いておく。raidtabファイルの書式は、/usr/doc/raidtools-0.90/raid＊.sampleファイルを参考にするとよい。

通常は、以下のようにraidtabに記述したデバイス名を書くだけだ。なお、「mdX」のXには数字が入る。

```
# mkraid /dev/mdX
```

■raidstart／raidstopコマンド

読んで字のごとく。RAIDのスタート／ストップを制御する。基本的な使い方は以下の通り。

```
# raidstart /dev/mdX
# raidstop /dev/mdX
```

■raidhotadd

本文中にも出てきたが、障害のあったハードディスクを取り替えた時に使用する。

```
# raidhotadd /dev/mdX /dev/YYY
```

YYYには、取り替えたハードディスクのhdaやsdaといったデバイス名を記述する。このコマンドによって、付け加えられたディスクは再構築され、新たにRAIDアレイの仲間入りをする。

このオプションを付けて実行してみても、さらに「キャンセルするなら、5秒以内にCtrl－Cを押せ」というメッセージが出て、5秒後に実行される。RAIDの初期化コマンドだけに慎重になっているのだろう。

RAIDができたなら、mke2fsでファイルシステムを作成する。マウント後、容量を見てみると、2台合わせると30Gバイト近いのだが、領域としては半分の約15Gバイトになっている。これは1台のハードディスクの1つのパーティションと同じものが、もう1台のハードディスクに作られているからである。

そして片方のハードディスクが万一トラブルに見舞われても、もう1台に内容をコピー（ミラー）してあるから復旧できるという仕組みになっている。

■Linux版HDBENCH

それにしても、RAIDの速さは気にならないだろうか？　私は気になる……。というわけでここでもハードディスクのベンチマークをとっておこうと思う。

正確には「HDBENCH clone」である。**表4**のURLよりダウンロード後、Linuxではおなじみのやり方でバイナリができあがる。この動作は覚えておきましょうね。

```
# ./configure
# make
# make install
```

実行ファイルは/usr/local/binにインストールされる。それでは、どきどきしながら、試してみよう。先に断っておくが、X Window Systemを立ち上げておくことを忘れないように。

```
# /usr/local/bin/hdbench &
```

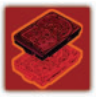
立ち上げた後、「DRIVE」と書かれた覧にテストしたいハードディスクのマウントポイントを書き込んでテストする。今回私は「/raid」という場所にマウントした（**画面1**）。ここでおまじないというか、転送方法を変えるために次のコマンドを打っておこう。これによりDMA転送を実現してくれる。xには変更したいデバイスを入れていただきたい。

```
# hdparm -t -d1 /dev/hdx
```

結果は確かに速くはなったのだが、RAIDを組んでいない1台のハードディスクでの結果も、良い数字を出している。ミラーリングのテストとともにストライピングのテストもしたので、併せて載せておく（**表5**）。本当ならばRAID 0（ストライピング）が、飛び抜けて速くなるはずなのだが、今回はテストの都合上、1つのIDEバスに2台のハードディスクを繋いでテストしたのとインターフェイスがUltraDMA/33のせいか、期待通りの結果にはならなかった。

本来は1つのIDEバスに1台のハードディスクを繋いでRAID 0を組むことによりIDEバス上のお互いの信号の干渉による待ち時間を少なくすることができ、転送レート的には1台の時の倍近い数値になるはずである。

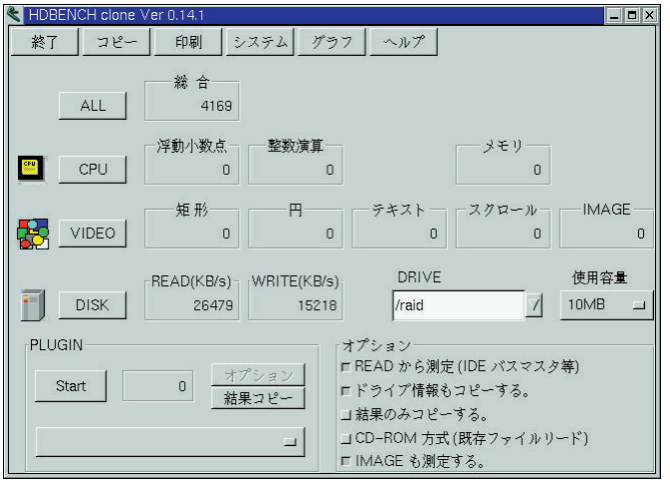
RAID 1（ミラーリング）では読み込みは1台と変わらないか速いくらいになったが、ミラーリングをするため、同じ内容をもう1台のハードディスクに書き込んでいる時間が余計にかかるので、1台の時の約半分程度になっている。これは、信頼性を高めるためにはしかたがないことといえるだろう。



RAIDに対応しているLinuxインストーラ

ほとんどの最新ディストリビューションでは、標準でソフトウェアRAIDに対応している。起動時に、**画面2**のように「md driver」行や、「auto detecting RAID arrays」といったメッセージが出力されていることでも確認できる。

しかしソフトウェアRAIDの設定、特にraidtabファイルの設定は面倒だ。



画面1　HDBENCH cloneベンチマークの定番ソフト、HDBENCHでハードディスクの転送時間を計測した。

表5　LinuxでのRAID使用時のベンチマーク（HDBENCH clone）

HDDの数	Read(KB/s)	Write(KB/s)
1台	18172	17392
2台（RAID 0、ストライピング）	26479	15218
2台（RAID 1、ミラーリング）	20623	9216

インストール時に設定できたら楽なのにと誰でも思うところだ。

現在のところ唯一、Red Hat Linux 6.1のグラフィカルインストーラ（Anaconda）だけが、ソフトウェアRAIDをサポートしている。Kondara MNU/Linuxもインストーラに Anacondaを採用しているのでソフトウェアRAIDに直接インストールできる。ソフトウェアRAIDを試してみるのなら、これらのディストリビューションを利用すると便利だろう。

■Red Hat Linux 6.1で簡単に

それでは、Red Hat Linux 6.1改訂日本語版を使って、実際にソフトウェアRAIDを設定し、そこにインストールしてみよう。

インストールCDから起動し、いくつかの質問項目に答えていくと、パーティションの設定画面になる。「追加」ボタンを押すとダイアログボックスが表示されるので、パーティションタイプで「Linux RAID」を選択すると、マウントポイントには「＜RAIDパーティション＞」が表示される（**画面3**）。サイズや選択可能なドライブは、使用しているハードディスクにあわせて適宜入力する。ここでは、hdaに3000Mバイトを設定した。

RAIDを構成するためには複数のパーティションが必要なので、同様に、hdcにも3000MバイトのRAIDパーティションを設定すると、hda6とhdc1の2つのRAIDパーティションができた（**画面4**）。

先ほどまで薄くなっていた「RAIDデバイスの作成」ボタンが選択できるようになるので、それを選び、2つのパーティションを/dev/md0という1つのRAIDデバイスに設定していく（**画面5**）。

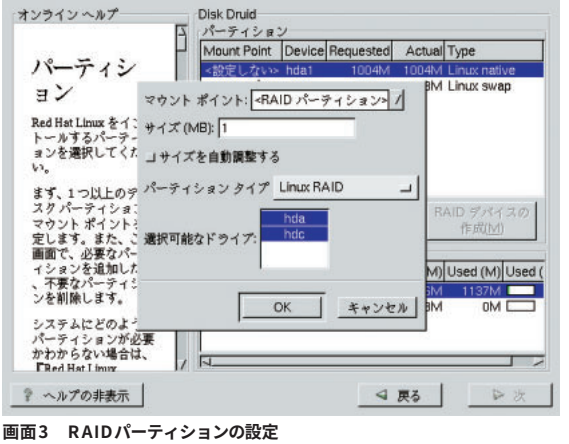
RAIDタイプには、RAID 0のほかにRAID 1やRAID 5も選べる。RAID 1（ミラーリング）の場合には、そのRAIDパーティションからブート可能なので、マウントポイントに「/」を設定できるが、RAID 0やRAID 5にすると「/」は選べなくなる。

ここでは、RAID 0（ストライピング）の設定を行っているため、マウントポイントを「/home」とし、「/」用に別のパーティション（hda1）を割り当てている。

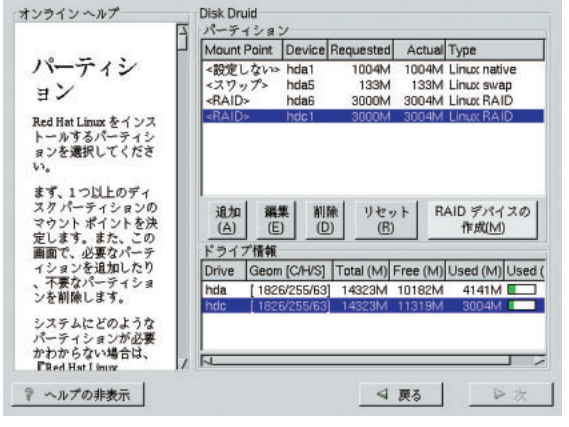
ほかの設定は、通常のインストールと変わらない。ファイルシステムの作成の際に、RAIDパーティションとして作成され、/dev/md0といったデバイスにインストールされる。

```
md driver 0.90.0 MAX_MD_DEVS=256, MAX_REAL=12
raid5: measuring checksumming speed
   8regs     :   168.402 MB/sec
   32regs    :   148.590 MB/sec
using fastest function: 8regs (168.402 MB/sec)
scsi : 0 hosts.
scsi : detected total.
md.c: sizeof(mdp_super_t) = 4096
Partition check:
 hda: hda1 hda2 < hda5 hda6 hda7 hda8 hda9 >
autodetecting RAID arrays
autorun ...
... autorun DONE.
```

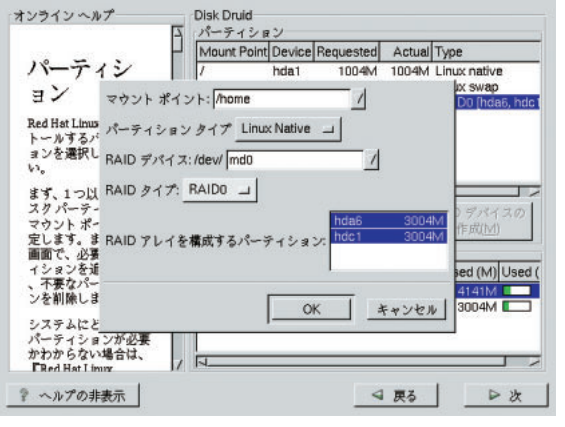
画面2　LASER5 Linux 6.0の起動時に出力されるメッセージ



画面3　RAIDパーティションの設定



画面4
2つのRAIDパーティション



画面5
RAIDデバイスを構成

もしもハードディスクにトラブルが発生したら

スピード優先でRAID 0を組んだ場合は、冗長情報がないので復旧は不可能で、クラッシュした場合は泣くしかない。安全性、信頼性優先でRAID 1を組んだ時、万一ハードディスクがお亡くなりになったとして、どのようにして、こっち（管理者、ユーザー）に伝わるのだろうか。

■本当に写してますか？

それでは、1台のハードディスクを壊してみよう（！）。というわけにもいかないので、一度シャットダウンして、片方のハードディスクの電源を抜いて起動してみよう。私の場合は本体のフタを開けっ放しだから楽だが、皆さんは大変かもしれない。さて、どうなっただろうか？

最初私は何も考えず、IDEバスのセカンダリに繋がっている、マスタのほうのハードディスクを止めたのだが、起動途中にメンテナンスモードに入ってしまった。

そして、何も設定を変えず、抜いたハードディスクも戻さずに再起動をすると、今度は起動はしたのだが、RAID自体は「hdcがない！」といわれて動かすことができなかった。次に、抜いた/dev/hdcは戻して、/dev/hddを抜いてみた。今度は何事もないように起動する。そして、raidstartコマンドで、RAIDをスタートさせてみる。RAIDが動き始めるときに「スペアディスクがない」という警告が出た。

■復旧はできるのか？

さて、RAID 1（ミラーリング）の設定をしたまま、再び/dev/hdcの電源を抜いてハードディスクが壊れていることにして起動してみよう。/dev/hdcがないのでメンテナンスモードに入ってしまうと思う。少なくとも私の場合は入ってしまう。

そんな突然の事故のためのミラーリングだが、起動してくれなくては、システム復旧もへったくれもない。こんな時は、まず起動させることを考える。この時の復旧方法として考えられるのは2つある。1つは/etc/raidtabを消すか名前を変えてRAIDを組んでいなかったことにする。しかし、これでは、何のためのミラーリングだかわからなくなってしまう。

それではもう1つ。/dev/hddに繋がっているハードディスクの設定をスレーブからマスタに変える。先ほど、/dev/hdd側のハードディスクを壊れた状態にしたときは何事もなく起動し、RAIDもスタートできたが、/dev/hdcがない警告が出ると書いた。これを逆手にとって、/dev/hdcはあることにしてしまうのである。そうすれば起動もできるし、仮にRAID上に大事なファイルがあってもミラーリングされているので、ほかのデバイスにバックアップが取れるってもんだ。

しかし、このままではいけない。ちゃんとハードディスクを交換して、またミラーリングを始めてもらはなければ。

■REBUILD（再構築）してみる

さて、hdd側のハードディスクが壊れたことにして、新たに別のハードディスクを繋げることを想定しよう。

ここではRAIDを初期化をしてしまうわけではないので、先ほどのmkraidコマンドではなくraidhotaddコマンドを使用する。これにより加えられたハードディスクはアレイの仲間入りをして、自動再構築を始める。

```
# raidhotadd /dev/md0 /dev/hdd1
```

画面にハードディスクの複製を作っているようすが流れて、再構築が始まる。

mdstatの状態を見てみると、「[2/1] [U_] recovery=94% finish=0.2min」といったメッセージで再構築の状態がわかる（**画面6**）。

同じ行の出力が、「[2/2] [UU]」のように変われば、再構築が終了していることになる（**画面7**）。

```
# cat /proc/mdstat
Personalities : [raid1]
read_ahead 1024 sectors
md0 : active raid1 hdc1[2] hdd1[0] 2048192 blocks [2/1] [U_]
recovery=94% finish=0.2min
unused devices: <none>
```

画面6　REBUILD中のmdstat情報

```
# cat /proc/mdstat
Personalities : [raid1]
read_ahead 1024 sectors
md0 : active raid1 hdc1[1] hdd1[0] 2048192 blocks [2/2] [UU]
unused devices: <none>
```

画面7　REBUILD終了後のmdstat情報

革命的大変化！

XFree86 4.0の衝撃！

速報！

「UNIXでGUIなんて邪道だ」という一部の人を除けば、多くのLinuxユーザーがX Window Systemを使っている。そんなLinuxユーザーにおなじみの、XFree86が4.0にバージョンアップした。3D描画機能の大幅な強化など、新機能満載のXFree86 4.0の実力をレポートする。

CD-ROMにソース&バイナリ収録！

XFree86 4.0とは？

長く待ち望まれていたXFree86 4.0（以下4.0と表記）が、開発元であるXFree86プロジェクト（**画面1**）から発表された。

XFree86は、X Window Systemを＊BSDやLinuxなどのPC UNIX系OSに移植したものである。PC UNIXの普及状況を考えれば、世界で最も多く使われているX Window Systemと言えるだろう。

現在X Window Systemを管理しているのは、The Open Group内のX.Orgという組織である。The Open Groupは、特定のメーカーによらない中立的な機関だ。4.0は、X Window Systemの最新版である、X11R6.4をベースに開発されたものだ。

4.0は、今まで開発が続けられてきたXFree86 3.3.x系列とは内部構造が大幅に異なっており、一から作り直したといってもいいほどである。また新たな機能も多く付加されており、野心的な新バージョンだ。

その一方で、現状ではドライバが十分に揃っていないこともあり、すべての人に推奨できるレベルには到達していない。大規模な公開ベータテストが始まった段階と思えばいいだろう。

XFree86 4.0の構成や、新機能を詳細に見ていこう。

画面1
XFree86プロジェクトのWebサイト。

XFree86 4.0の新機能

X Window Systemは、最初のバージョンが発表されてから10年以上もたっているシステムだ。基本設計が優れているため、今でもUNIX系OSの標準的ウィンドウシステムとして用いられているが、近年のハードウェア／ソフトウェア事情に適応していない部分があることも事実だ。

そこでXFree86 4.0では、主要プラットフォームであるx86 PCに合わせて、機能が追加された（**表1**）。

Xサーバの一本化

4.0の配布セットを見て最初に気がつくことは、Xサーバで「XFree86」という名前のものが、ただ1つしかないことだ。3.3.x系列では、新しいグラフィックスカード用のXF86_SVGA以外に、XF86_S3、XF86_Mach64などアクセラレータチップ固有のXサーバが存在していた。

モジュールのダイナミックロード

「Xサーバが1つしかないということは、XFree86はとても巨大なのでは？」と心配する方がいるかもしれない。だが4.0では、機能やハードウェア別に分けられたモジュールを、必要に応じてロードするダイナミックロードの仕組みを備えている。実際「XFree86」のファイルサイズは、2.2Mバイト程度と3.3.x系列のXF86_SVGAよりも小さいくらいだ。

モジュールのダイナミックロードは、次に述べるマルチヘッドのサポートや、新機能の追加を破綻することなく行うために、必須だといえるだろう。

また最近は減少してきたが、ハードウェア情報を公開しないメーカーの製品でも、バイナリのモジュールのみ配布することで、オープンソース環境での利用が可能になる。

このモジュールローダは、商用のXサーバであるMetro Xで知られている米Metro Link社（**画面2**）が提供したものだ。OSの機能とは独立したものであるため、モジュールのダイナミックロードをサポートしていないOSでも利

名称	目的
Xサーバの一本化	機能追加の簡易化
モジュールのダイナミックロード	保守性の向上
XFree86 Acceleration Architecture（XAA）の作り直し	2D描画性能の改善
マルチヘッド／Xinerama	複数ディスプレイの有効利用
VESA DDCサポート	設定の簡易化
フォントフォーマット追加	国際化
Unicodeサポート	
GLX／DRIサポート	3D描画性能の改善

表1　XFree86 4.0の新機能

用可能だ。

またOS間の差異は、メインの「XFree86」サーバが吸収するため、CPUが同じなら、OS間でモジュールは共通で利用できる。たとえばx86のLinuxでコンパイルしたモジュールは、FreeBSDの「XFree86」サーバでも利用できる。

XFree86 Acceleration Architectureの作り直し

アクセラレータチップを利用して、2Dの描画を高速化するXFree86 Acceleration Architecture（XAA）は、XFree86 3.3.x系列でも利用されていた。4.0では、新たにコードを書き起こしており、まったくの新バージョンになっている。XAAはほとんどのドライバで利用されており、描画速度の改善が実現している。

そのほかDirect Graphic Access（DGA）、X-Video（XV）、XFree86-Misc、XFree86-VidModeExtensisionといった描画高速化の仕組みも、それぞれバージョンアップされている。

マルチヘッド／Xinerama

MacintoshやWindows 98以降では、複数のグラフィックスカードを使って複数のディスプレイを利用できる。また、商用のXサーバもこの機能を持っている製品が多い。XFree86 4.0でもマルチヘッドと呼ぶ、複数ディスプレイの利用が可能になった。多くの作業領域が必要な、グラフィック系のアプリケーションや、GUIプログラムの開発環境で、威力を発揮するだろう。また、入力デバイスも同種のものを複数利用することができるようになった。1台のPCにマウスを2つ接続してもあまり実用的ではないが、絵を描くためのゆっくり動くトラックボールと、デスクトップを操作するための速いマウスの両方を使うといったことが考えられるだろう。

Xineramaは、4.0のベースとなったX11R6.4に付け加えられた機能で、単に複数のディスプレイを使うのではなく、複数のディスプレイを1つのデスクトップとして利用できるものだ。以前のX Window Systemでは、ディスプレイを2つ接続しても、2つの独立したデスクトップとして利用するしかなかった。Xineramaではこの制限がなくなり、1つのウィンドウを複数のディスプレイにまたがって表示させられるようになった。

VESA DDCサポート

今まで、XFree86のインストール時には、ディスプレイの水平同期周波数やリフレッシュレートといったスペックを自分で入力する必要があった。

4.0では、Video Electronics Standards Association（VESA）という標準化団体が策定した、Display Data Channel（DDC）という規格をサポートした。この機能を利用したドライバを用いることで、ディスプレイの情報をシステムが取得することが可能になり、最適な解像度やリフレッシュレートを自動的に設定できるようになった。

フォントフォーマット追加

日本国内で扱われているLinuxディストリビューションの多くが、X-TrueTypeを組み込んだXサーバを用いているが。4.0は、標準でX-TrueType、またはxfsftのどちらかを組み込むことで、TrueTypeフォントを利用することができる。

また、米アドビシステムス社のCIDフォントもサポートされるようになった。この機能は、米SGI社が提供している。

Unicodeサポート

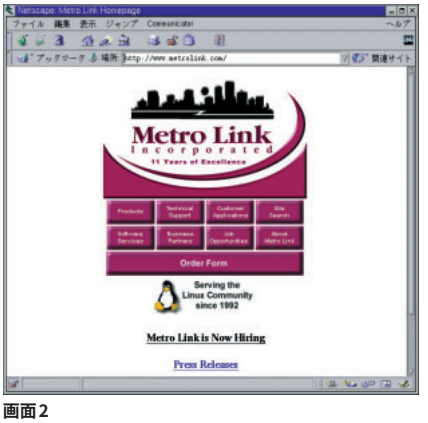
Unicodeサポートが行われ、Unicodeエンコーディングのフォントセットが利用できるようになった。付属するフォントは、ヨーロッパ系の文字が中心だ。

同時にxtermも改良が加えられ、これらヨーロッパ系の文字を表示できるようになった。

GLXとDRI

XFree86 4.0の新機能のうちで、エンドユーザーへの影響が大きいのは、このGLX／DRIのサポートであろう。今までWindows 98に比べて、明らかに劣っていた3Dアプリケーションのパフォーマンスの改善が期待される。

UNIX系のOS上で3Dグラフィックスを扱う際には、米SGI社が開発したOpenGLか、その互換品であるMesa3Dを用いるのが一般的だ。GLXは、OpenGLの表示をX Window



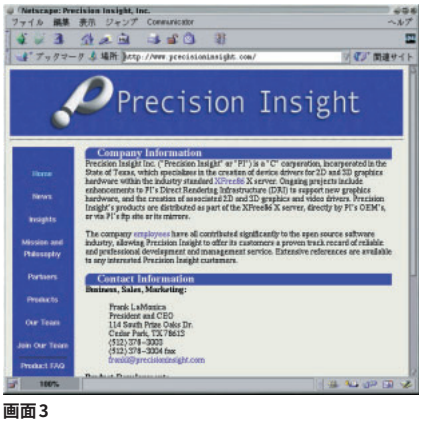
画面2
米Metro Link社のWebサイト

System上で行うための仕組みである。SGI社は、GLXをオープンソースとして提供している。

XFree86 4.0では、GLXと3Dグラフィックシステムである Mesa3DをXサーバに統合している。さらにハードウェアの3Dアクセラレーション機能を直接利用するための仕組みである Direct Rendering Infrastructure（DRI）を付加することで、X Window System上でも高速の3Dグラフィックス環境が利用できるようになった。

上記の作業を実際に行ったのは、米Precision Insight社である（**画面3**）。同社はハードウェアベンダーと協力して、X Binary Free（XBF）、XFree86 Compatible（XFCom）と呼ばれるXサーバを提供していたことで知られている。XBFやXFComはバイナリのみで提供されるため、ハードウェアベンダーにとっては、自社の技術を公にせずオープンソースコミュニティへドライバを提供できるという利点がある。またユーザーにとっても、X Window Systemを利用できるハードウェアが増える点で、ありがたいものだ。

Linuxの場合、XFree86 3.3.x系列でも、米3dfx社のVoodoo3／Bansheeではハードウェアの機能を用いた高速な3D描画ができた。これは、同社が専用の3Dライブラリである GlideのLinux版を提供していて、OpenGLはGlideライブラリへのラッパーとして実現することができたからだ。当然ほかのベンダーはこの方法を用いることはできない。



画面3
米Precision Insight社のWebサイト

WindowsにおけるDirect3Dに似た、DRIが提供されることで、UNIX系OS上での3Dグラフィックスを扱う機会が増えるだろう。もちろん、3Dゲームも多くのリリースが期待される。

DRIの詳細

高速な3Dグラフィックス環境を実現するためには、ハードウェア、ドライバ、OSが密接に連携していることが必要だ。Windowsの場合は、マイクロソフトが「これからはDirect3D」と決めれば、簡単に実現できることだが、OS本体、Xサーバがそれぞれ別のグループで開発されているオープンソースのOSにとっては、とても困難なことだ。DRI（Direct Rendering Infrastructure）は、これを実現するために、Xサーバ用のモジュール、カーネル組み込み用モジュールをそれぞれ用意している。

今のところ、DRIを利用できる環境は限定されており、LinuxはDRIを利用できるただ1つのOSである。またグラフィックスカードも米3dfx社のVoodoo3／ Voodoo Bansheeと 、3Dlabs社のOxygen GMX 2000に限られている。Oxygenは、業務用で高価なグラフィックスカードであるため、事実上Voodoo3／Voodoo Bansheeしか選択肢がない。

しかしIntel i810、Matrox G400、ATI Rage 128、 3dfx Voodoo4／ 5（未発売）といった比較的新しいアクセラレータも開発が進められている。特にi810は、安価なグラフィックスとメモリコントローラハブの統合チップであり、主にローエンドPCで利用されているチップだ。安価なPCでも高速な3Dグラフィック環境が得られるようになる意義は大きい。

DRIのインストール

ここでXFree86 4.0をインストールしたPCへのDRIのインストール方法を紹介しよう。グラフィックスカードは、Voodoo3を用いているとする。XFree86 4.0のインストールについては、123ページを参照してほしい。

繰り返しになるが、高速な3Dグラフィックス環境を得るためには、ハードウェアとOSやドライバが密接に連携していなければならない。そのためDRIを用いて、3Dアクセラレータを利用する際には、カーネルにもDRI用のモジュールを組み込む必要がある。

このカーネルモジュールは、バイナリパッケージには含まれておらず、ソースから作成しなければならない。まず付録CD-ROMに収録されているX400src-1.tgzを解凍し、xc/programs/Xserver/hw/xfree86/os-support/linux/drm/kernelディレクトリに移動する。余談だが、このディレクトリ内にはi810やmgaで始まるファイルが多数置かれており、これらのアクセラレータでDRIが利用できる日も近いのではないかと期待させる。

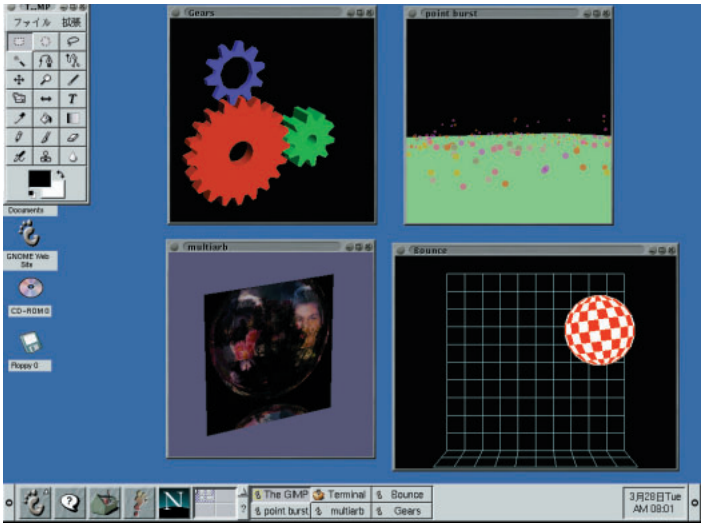
次にmakeコマンドで、カーネルモジュールを作成する。

```
# make -f Makefile.linux tdfx.o
```

終了したら、/lib/modules以下の適切なディレクトリにtdfx.oをコピーし、insmod tdfx.oとしてロードする。その後Xを起動して、ログファイル（/var/log/XFree86.0.log）に、

```
(II) TDFX(0): direct rendering enabled
```

のメッセージがあったら、インストール成功だ。いくつかデモプログラムを起動してみたが、ハードウェアによる高速化が実感できた（**画面4**）。



画面4
Mesa3D付属のデモプログラム。

XFree86 4.0の今後

多くの新機能を持ったXFree86 4.0だが、現状ではまだまだ完成度の低い部分が多い。これらの問題点は、古いハードウェアのドライバ不足と、新しいハードウェアや機能への未対応に大別される。

古いハードウェアのドライバ不足については、現状ではあきらめるしかないだろう。利用者の少ない古いグラフィックスカードについては、3.3.x系列で使い続けるというのが、正しい道だ。

新しいハードウェアに関しては、人気の高い製品ならば、じきにサポートされるようになっており、心配はないだろう。また最近では、ベンダー自身によるサポート例も増えている。モジュールのダイナミックロード機能を利用することで、バイナリのみの配布もしやすくなっており、このような例はますます増えてくると思われる。

新しい機能についても同様で、ある程度時間が経過すれば、解決していく問題だ。

多くのドライバが書き直された結果、現在のXFree86 4.0の安定性は、デスクトップとして常用するにはちょっときついレベルになってしまっている。だが、現在十分に安定している3.3.x系列も3.0が出たのは、5〜6年も前のことだ。時が経過するにつれて、安定性も向上していくだろう。

XFree86 4.0は、今ちょうど公開ベータテストが始まったところと言える。実験マシンを用意できるなら、チャレンジしてみるといいだろう。

XFree86 4.0のインストール

この原稿を書いている段階では、XFree86 4.0を含んだLinuxディストリビューションはないが、4月7日に発売されるTurboLinux Workstation日本語版6.0のコンパニオンCD-ROMに、収録されることになっている。またSuSE Linuxも次バージョンでの収録を予定している。

インストール前に

マニュアルを読めばわかるように、現状の4.0は、まだ安定しているとは言いがたい状態だし、古いグラフィックスカードは、サポートされていないものが多い。

また、XFree86プロジェクトから提供されているバイナリは、tarボール形式であるため、RPMやdeb形式でパッケージ管理を行っているディストリビューションにインストールすると、データベースの整合性が失われてしまうことになる。

しかし、それでも新しいものを試してみたいという読者のために、付録CD-ROMにXFree86 4.0のバイナリパッケージとソースを収録した。バイナリパッケージは、それぞれglibc 2.0、2.1に対応した2種類がある。Vine Linux の1.1以前、TurboLinuxの4.5以前は、glibc 2.0用を、LASER5 Linux、Red Hat Linux、TurboLinux Workstation 6.0はglibc 2.1用を用いる。CD-ROM容量の都合で、libc 5用のバイナリパッケージは収録できなかった。

編集部でも、このバイナリパッケージを用いて、既存のディストリビューションに4.0をインストールしてみたので、その際の手順を以下に記す。

使用したディストリビューションは、LASER5 Linux 6.0 Rel.2である。この手順だけが「正解」というわけではないし、編集部のお勧めでもない。この点を了解したうえで、先に進んでいただきたい。

必須	オプション
Xbin.tgz、Xlib.tgz、Xman.tgz、Xdoc.tgz、Xfnts.tgz、Xfenc.tgz、Xetc.tgz、Xvar.tgz、Xxserv.tgz、Xmod.tgz	Xhtml.tgz、Xnest.tgz、Xflat2.tgz、Xprog.tgz、Xfnon.tgz、Xjdoc.tgz、Xprt.tgz、Xps.tgz、Xf100.tgz、Xfscl.tgz、Xfcyr.tgz、Xfsrv.tgz、Xvfb.tgz

表2　バイナリパッケージ一覧

準備

可能ならば実験用の専用マシンを用意したい。そこまでできなくても、専用のパーティションを切るなどして、ふだん使っている環境と切り分けたところを使おう。重要なデータが入っているようなマシンは、避けたほうがよいだろう。

まず用意した環境に、通常どおりにLinuxをインストールする。グラフィカルログインではなく、コンソールで起動するように設定しよう。RPM系のディストリビューションでは、ランレベルを3にしておけばよい。

○既存のX環境の待避

以下の作業は、rootで行うものとする。まず、今あるX Window System関係のファイルをディレクトリごと待避する。

```
# cd /usr
# mv X11R6 X11R6.335
# cd /etc
# mv X11 X11.335
```

○パッケージのコピー、パーミッションの変更

CD-ROMをマウントし、必要なパッケージをコピーする。

```
# cd ~
# cp  /mnt/cdrom/Linuxmag/Linux-ix86-glibc21/ .
```

次にインストーラスクリプトと、スクリプトが呼び出すプログラムのパーミッションを変更して、実行可能に設定する。

```
# cd Linux-ix86-glibc21
# chmod 755 Xinstall.sh extract
```

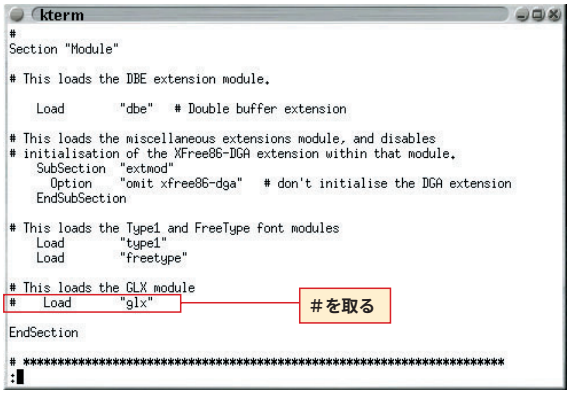
○インストール

インストーラスクリプトを起動して、パッケージをインストールする。**表2**に示した必須のパッケージは、自動的にインストールされる。

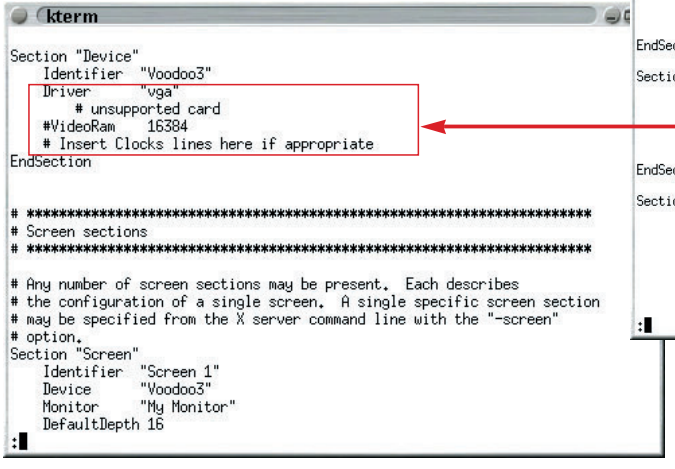
```
# ./Xinstall.sh
```

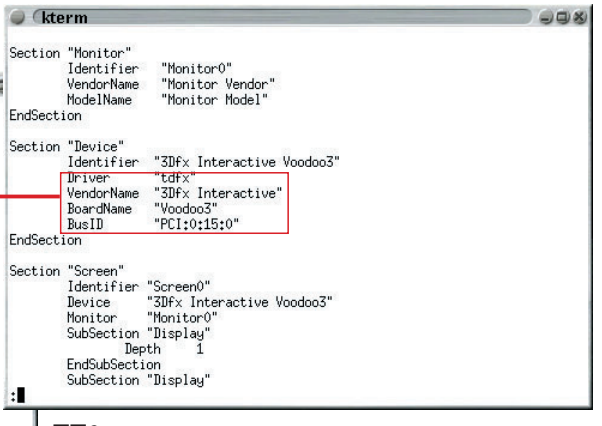
○設定ファイルの作成

今のところ、4.0には3.3.x系列に用意されていたXF86Setupのような設定ツ



画面5
XF86Configの修正（その1）
「Module」セクション内のGLXモジュールを有効にする。





画面6
XF86Configの修正（その2）
「Device」セクションの内容を「Identifier」以外、すべてXF86Config.newのものに置き換える

ールがないため、テキストベースのxf86configを用いて設定を行う。

しかしxf86configは、あまり使いやすくないうえに、グラフィックスカードのリストに比較的新しい製品が載っていない（古いデータベースをそのまま使い回した?）。そのため、Voodoo3やMatrox G400などのグラフィックスカードでは、正しい設定ファイルが出力されない。

そこで、4.0からXサーバに組み込まれた設定ツールを用いて、ハードウェア情報を得、xf86configが出力した設定ファイルを修正するといいだろう。コマンドラインから、

```
# XFree86 -configure
```

とすることで、ホームディレクトリにXF86Config.newというファイルが作成される。これを参考に、xf86configが出力した/etc/X11/XF86Configをまず「Module」セクション内を修正する。GLXを使用するなら、コメント（#）を外してGLXモジュールが読み込まれるようにしよう（**画面5**）。Voodoo3なら、もちろんGLXは必要だ。

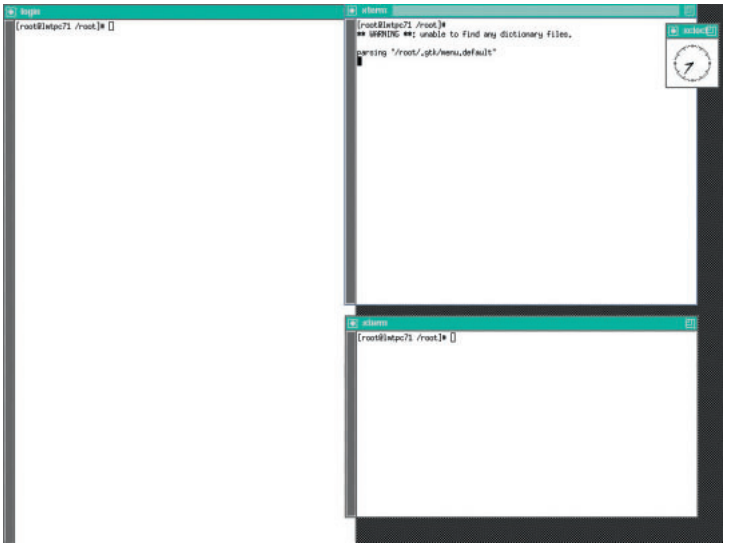
続いて、「Device」セクションだ。**画面6左**は、xf86configが作成したXF86ConfigファイルのDeviceセクションだが、xf86configのデータベースにVoodoo3が載っていないため、Driverが「vga」となっている。そこで、**画面6右**のXF86Config.newのDeviceセクションを利用する。XF86Config.newは、Xサーバによるハードウェアの検出結果に基づいて出力されるため、より正確である。画面6のように、Identifier以外の4行をXF86Configにコピーしよう。Identifierをコピーすると、そのほかのセクションも変更する必要が出てくるので、そのままにしておくほうがよい。

○起動の確認

設定ファイルができたら、X Window Systemが起動するかどうか確認しよう。

```
# startx
```

シンプルなtwmが表示されれば、成功だ（**画面7**）。うまく起動しなかった場合は、ログファイル（/var/log/XFree86.0.log）を参照して、間違っている部分を修正しよう。



画面7
4.0インストール成功！

○待避していたディレクトリを書き戻す

最初に待避しておいたディレクトリの中身を書き戻す。4.0のファイルに上書きしてしまわないように注意しよう。

```
# cd /usr
# cp -ai X11R6.335/* X11R6/
# cd /etc
# cp -ai X11.335/* X11/
```

以上の作業で、XFree86 4.0に含まれているファイルを除いて、元の状態に戻ったことになる。インストールされていたGNOMEやKDEも起動することを確認した。そのまましばらく使ってみたが、特に大きな問題は起きなかった。比較的よくサポートされているVoodoo3というグラフィックスカードを用いているうえに、4.0の新機能であるXineramaやマルチヘッドを使わないければ、それほど不安定になることはないようだ。

ディストリビューションの対応状況

「世界で初めてXFree86 6.4を収録したディストリビューション」の栄誉に輝くのは、4月3日にドイツで発売されるSuSE Linux 6.4のようだ。また4月7日発売のTurboLinux Workstation 6.0日本語版が、僅差で続いている。どちらも標準では、3.3.x系列がインストールされるようになっており、4.0は新しいソフトを試してみたいユーザー用のサービスと言えるだろう。

対応ドライバの少なさから考えて、インストーラのデフォルトで4.0が用いられるようになるのは、しばらくかかりそうだ。

XFree86 4.0対応アクセラレーター一覧表

ドライバ名	ベンダー名	対応アクセラレータ／グラフィックスカード
apm	Alliance Pro Motion	AT3D、AT25、AT24
		XAAに対応しており、ハードウェアアクセラレーションが可能
ati	ATI	VGAWonder、
		Mach32（Graphics Ultra Pro、Graphics Wonder）、
		Mach64（Graphics Pro Turbo、3D Xpression、All-In-Wonder）
		上記各シリーズのアクセラレータに対応する。（）内は代表的なグラフィックスカード名。ほかにも多数のカードがある。
		現在のバージョンでは、ハードウェアアクセラレーションを用いていない。
		ATI Rage128シリーズは、別のドライバが対応
chips	Chips & Technologies	Basic（ct65520、ct65525、ct65530、ct65535、ct65540、ct65545、ct65546、ct65548）
		WinGine（ct64200、ct64300）
		HiQV（ct65550、ct65554、ct65555、ct68554、ct69000、ct69030）
		Basic、WinGine、HiQVの各シリーズのアクセラレータに対応する。
cirrus	Cirrus Logic	CL-GD5420、CL-GD5422、CL-GD5424、CL-GD5426、CL-GD5428、CL-GD5429、CL-GD5430、CL-GD5434、CL-GD5436、CL-GD5440、CL-GD5446、CL-GD5462、CL-GD5464、CL-GD5465、CL-GD5480、CL-GD6205、CL-GD6215、CL-GD6225、CL-GD6235、CL-GD6420、CL-GD6440、CL-GD7541、CL-GD7542、CL-GD7543、CL-GD7548、CL-GD7555
		XAAに対応しており、ハードウェアアクセラレーションが可能
cyrix	Cyrix	MediaGX、MediaGXi、MediaGXm
		CPU＋チップセットを統合したMediaGXシリーズに対応する。
		現在は、8ビット（256色）モード固定。
glint	3DLabs	GLINT 500TX、GLINT MX plus Delta、GLINT MX plus Gamma、Permedia、Permedia 2（classic、2a、2v）
		XAAに対応しており、ハードウェアアクセラレーションが可能
i740	Intel	i740
		XAAに対応しており、ハードウェアアクセラレーションが可能
i810	Intel	i810、i810-dc100、i810e
		アクセラレータとメモリコントローラハブの統合チップであるi810シリーズに対応する。
		XAAに対応しており、ハードウェアアクセラレーションが可能
		現在は、x86版のLinuxのみに対応。Linuxカーネル2.3.42以降に付属しているカーネルモジュール（agpgart.o）が必要。
mga	Matrox	Millennium、Millennium II（RAMDACがTI TVP3026のカードのみサポート）
		Mystique、Mystique II、Millennium G200、Mystique G200、Productiva G100
		G400（Dual Headは、プライマリコネクタのみ使用可）
		TV出力は使用不可。
		mgaドライバは、非常に開発が進んでおり、XFree86環境における最速Xサーバである。
neomagic	NeoMagic	NeoMagic 2200、NeoMagic 2160、NeoMagic 2097、NeoMagic 2093、NeoMagic 2090、NeoMagic 2070
		ノートPC用のアクセラレータ。外部ディスプレイにも対応。
		XAAに対応しており、ハードウェアアクセラレーションが可能（8、15、16ビットモードのみ）

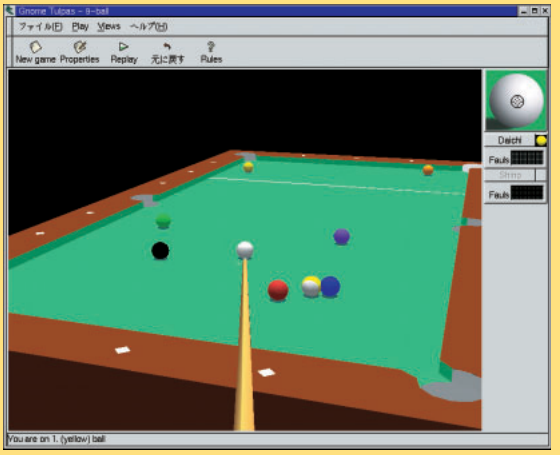
ドライバ名	ベンダー名	対応アクセラレータ／グラフィックスカード
nv	nVIDIA	Riva 128、Riva 128ZX、Riva TNT、Riva TNT2（Ultra、Vanta、M64）、
		GeForce256、Quadro
r128	ATI	ATI Rage 128を利用したグラフィックスカード（Rage Fury AGP、XPERT 128 AGP、XPERT 99など）に対応する。
		XAAに対応しており、ハードウェアアクセラレーションが可能
rendition	Rendition	V1000、V2100、V2200
		現在のバージョンでは、ハードウェアアクセラレーションを用いていない。
s3virge	S3	ViRGE、ViRGE DX、ViRGE GX、ViRGE MX、ViRGE MX+、ViRGE VX、Trio 3D、Trio 3D/2X
		XAAに対応しており、ハードウェアアクセラレーションが可能
sis	SiS	SiS6326、SiS5597、SiS5598、SiS530、SiS620
		上記の各アクセラレータや、アクセラレータ＋チップセット統合チップに対応する。
		検証が不十分だが、SiS300、SiS540、SiS630にも対応している。
		XAAに対応しており、ハードウェアアクセラレーションが可能
tdfx	3dfx	Voodoo3、Voodoo Banshee
		XAAに対応しており、ハードウェアアクセラレーションが可能
tga	DEC（現Compaq）	DEC 21030 TGA
		Linux/Alpha上のみで動作が確認されている。
		XAAに対応しており、ハードウェアアクセラレーションが可能
trident	Trident Microsystems	TVGA8900D、TGUI9420DGi、TGUI9440AGi、TGUI9660、TGUI9680、ProVidia 9682、ProVidia 9685、Cyber9320、Cyber9382、Cyber9385、Cyber9388、Cyber9397、Cyber9520、Cyber9397/DVD、Cyber9525/DVD、3DImage975、3DImage875、Blade3D、CyberBlade/i7、CyberBlade/DSTN/i7、CyberBlade/i1
		XAAに対応しており、ハードウェアアクセラレーションが可能
tseng	Tseng Labs	ET4000AX（アクセラレータではない）
		ET4000/W32、ET4000/W32i、ET4000/W32p、ET6000、ET6100
		ET4000AX以外はXAAに対応しており、ハードウェアアクセラレーションが可能

主な非対応アクセラレータ

ベンダー名	アクセラレータ
Compaq	AVGA
IBM	8514/A、XGA-2
IIT	AGX-014、AGX-015、AGX-016
Number Nine	Imagine128（シリーズ1、2、4）、Revolution 3D（T2R）
nVIDIA	NV1
S3	86C911、86C924、86C805、86C805i、86C928、86C864、86C964、86C732、86C764、86C765、86C767、86C775、86C785
Tseng	ET3000
Weitek	P9000、P9100
Western Digital	WD90C00、WD90C10、WD90C11、WD90C24、WD90C24A、WD90C30、WD90C31、WD90C33、

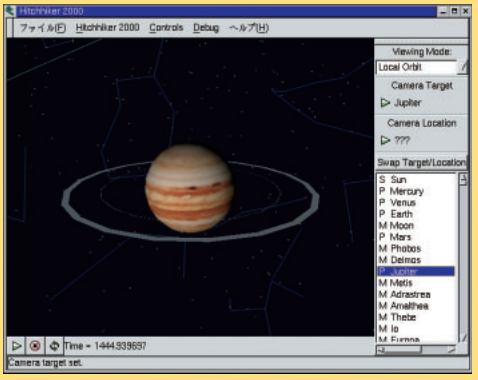
Free Application Showcase

文：出井 一
Text : Hajime Dei



Gtulpas　P.136

Free Amp　P.127



Hitchhiker 2000　P.139

※紹介したソフトは、すべて付録CD-ROMに収録されています。

テーマにも対応した高速MP3プレーヤ

FreeAmp

バージョン：2.0.5　　ライセンス：GPL

http://www.freeamp.org/

FreeAmpは、LinuxやWindows 9x／NTなどで動作する完全にフリーなMP3プレーヤだ。元々はクールな外見と安定性で定評のあるプレーヤだったが、バージョン2では外見を変化させる「テーマ」、ディスク上のMP3ファイルやプレイリストを一括管理する「My Music」、ダウンロードマネージャなど大幅に機能が強化された。EsounDサウンドデーモンや、ALSAサウンドドライバにも対応している。

●ビルドと日本語への対応

FreeAmpは、tarボールのほか、RPM／DEBパッケージでも配布されている。RPMパッケージはRed Hat 6.x用で、LASER5などでは、これを利用するのが簡単だろう。コマンドラインから、「rpm -ivh freeamp-2.0.5-1.i386.rpm」とすればいい。

tarボールからビルドする場合、egcs 2.91以降とNASM（http://www.web-sites.co.uk/nasm/）が必要だ。Vine 1.1ではegcsの入れ替えなど大掛かりな作業になる。ビルドの手順そのものは、「./configure」→「make」→「make install」という一般的なものだ。

なお、日本語環境で起こるダイアログの文字化けを解消するパッチ（freeamp-locale.patch）を用意した。ビルドする前に、tarボールの展開先で「patch -p1 < freeamp-locale.patch」とすればいい。

このほか、Vine 1.1やTurboなどglibc 2.0系のディストリビューションでは、base/unix/src/bootstrap.cppの76行目「union { int val; } unsem;」を「union semun unsem;」に修正する必要がある。

画面1
標準テーマのウィンドウ。サイズは3段階に変更できる。

●起動と初期設定

FreeAmpを起動するには、ktermなどのコマンドラインから「freeamp」とする。RPMパッケージを導入した場合は、「LANG=C freeamp」と英語環境で起動すれば、ダイアログの文字化けを避けられる。

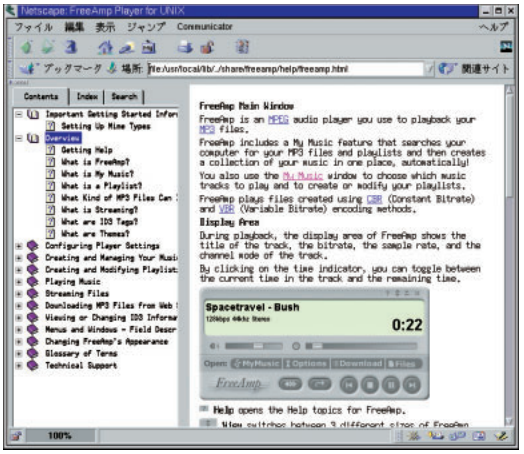
ウィンドウのデザインはシンプルなパネル風だ（**画面1**）。再生などのボタン類は説明するまでもないだろう。右上の小さなボタンをクリックすると、HTML形式のヘルプの参照や（**画面2**）、ウィンドウサイズの変更が可能だ。

なお、サウンドデーモンEsounD（esd）を利用している場合は、FreeAmpの出力モジュールを変更する必要がある。中央部の[Options]ボタンを押して設定ダイアログを開き、[Plugins]ページの[Audio Output]を、[esound.pmo]に変更すればいい（**画面3**）。

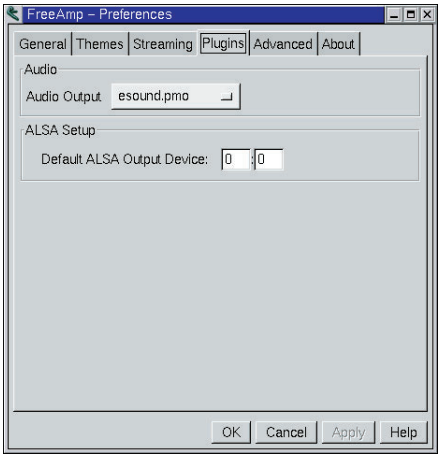
また、ALSAサウンドドライバを使っていない環境にRPMパッケージを導入すると、起動時に警告メッセージが表示される。これを消すには、/usr/lib/freeamp/plugins内にあるALSA用の出力モジュール（alsa.pmo）を削除する（拡張子を変更してもOK）。

●My MusicでMP3ファイルを管理

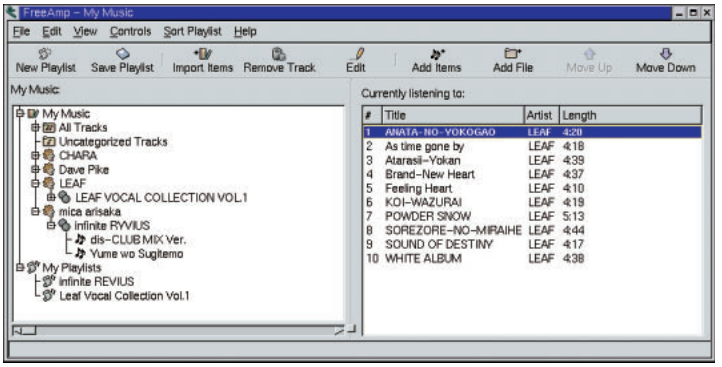
FreeAmpでは、ディスク上に散在するMP3ファイルやプレイリストを一括管理する「My Music」機能が用意されている。具体的には、各MP3ファイルのID3タグの情報に基づいて、「ア



画面2
左上の？をクリックするとHTML形式のヘルプが表示される。



画面2
EsounDの出力モジュール「esound.pmo」に変更する。



画面4
MP3ファイルやプレイリストを一括管理するMy Musicウィンドウ。

画面5
ディスク上からMP3ファイルを検索して自動登録する。

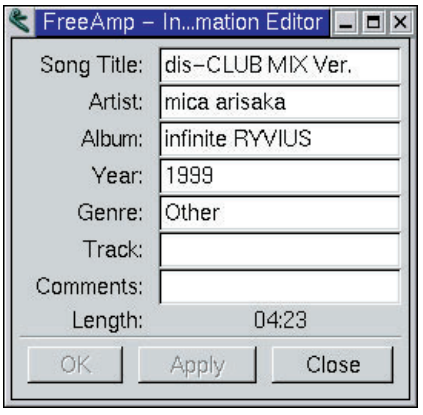
ーティスト」→「アルバム」というツリーで分類する。

まずは、中央部の[My Music]ボタンを押してウィンドウを開こう。初めて開いた場合は、ディスク上からMP3ファイルを検索するためのウィザードが一緒に開くが、この機能は不安定なことがあるため、[Cancel]ボタンを押して消しておく。

ウィンドウは2ペイン構成で、左にはMP3ファイルやプレイリストがツリー表示され、右には現在のプレイリストの内容が表示される（**画面4**）。なお、GTK+ 1.2.3以前には左のペインが隠れてしまうバグがあるので、左下の四角いつまみをドラッグして左のペインを表示させよう。

MP3ファイルを登録する方法は、手動と自動の2通り用意されている。手動での登録は、ツールバーの[Import Items]ボタンを押してダイアログを開き、登録するMP3ファイルやプレイリストを指定すればいい。

一方、自動登録のほうは、ウィザード形式で検索開始ディレクトリを指定すると、MP3ファイルやプレイリストをすべて登録してくれる（**画面5**）。ただし、現状ではスレッドによる検索処理が不安定で、FreeAmpが操作不能になる場合がある。



画面6
ID3タグの内容（曲目やアーティスト名など）を変更できる。

●プレイリストを作成して再生

いったんMP3ファイルやプレイリストがMy Musicに登録されれば、再生の操作はきわめて簡単だ。

たとえば、既存のプレイリストの内容をそのまま再生するには、ツリー中のプレイリストをダブルクリックすればいい。右側のペインにプレイリストの曲目が追加され、演奏が開始される。同様に、ツリー中のMP3ファイルをダブルクリックすると、現在のプレイリストの末尾にその曲目が追加される。プレイリストを白紙の状態に戻すには、[Edit]－[Clear Playlist]を選択すればいい。

さらに、[Move Up／Down]ボタンで曲順を変更したり、[Remove Track]ボタンで曲目をプレイリストから削除したりして、プレイリストの内容を自由に変更できる。作成したプレイリストは、[Save Playlist]ボタンで名前を付けて保存しよう。

プレイリストで選択されているMP3ファイルのID3タグは、[Edit]ボタンで表示・変更が可能だ（**画面6**）。FreeAmpは日本語タグに対応していないため、日本語の曲名などを入力しないようにしよう。

●テーマを導入して外見を変える

FreeAmp標準のウィンドウのデザイ

Column

起動時のSegmentation falutを回避するには

現状のFreeAmpには不具合があり、一般ユーザが初めてFreeAmpを起動した際、My MusicにMP3ファイルをひとつも登録しないで終了すると、次からは「Segmentation fault」を起こして起動に失敗する。

この問題を回避するには、初めて起動した際に手動でMP3ファイルをMy Musicに登録し、いったん終了すればいい。すでに「Segmentation falut」を起こす状態になっている場合は、「rm -rf ~/.freeamp」として、ホームディレクトリの「.freeamp」ディレクトリ以下をいったん削除してから起動すれば、初めて起動したのと同じ状態になる。

画面7
テーマを導入することで外見をさまざまに変化させられる。

画面8
こちらは、米コカ・コーラ社が提供するテーマ。

ンは、誰にでも分かりやすいものだが、面白みには欠ける。バージョン2では、「テーマ」（いわゆる「スキン」）を切り替えることで、ウィンドウの外見を簡単に変えられるようになった。

FreeAmpのテーマは拡張子fatの単独ファイルで、FreeAmpのWebサイトに用意されている（**画面7**）。また、米コカ・コーラ社のWebサイトからもクールなテーマ（**画面8**）を入手できる（http://www.coca-cola.com/moments/turn_up/mp3_index_frm.html）。

取得したテーマは、/usr/share（RPMパッケージの場合）か/usr/local/share（tarボールの場合）以下のfreeamp/themesディレクトリにコピーしておくと、FreeAmpが自動的に利用する。

テーマの切り替えは、設定ダイアログの[Themes]ページで行う（**画面9**）。一覧表示されているテーマの中から切り替えたいものを選択して、[OK]ボタンを押せばいい。

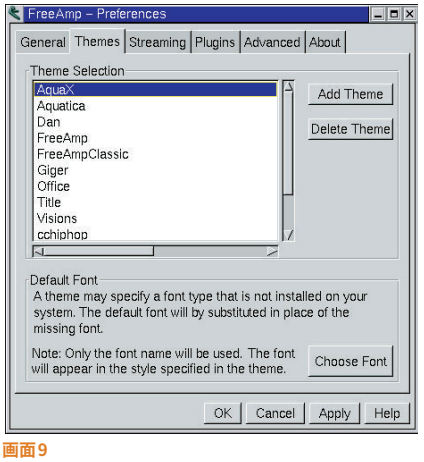
なお、上記のディレクトリ以外に置かれたテーマも、[Add Theme]ボタンで追加できる。ただし、この部分の処理に不具合があり、追加後のリストの表示が乱れてしまうので注意されたい（テーマの追加自体は正常に行われている）。

●Webからダウンロードする

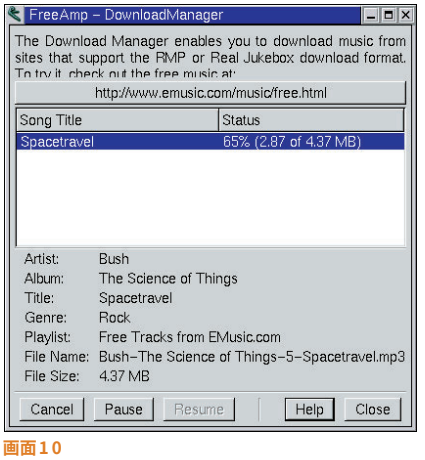
FreeAmpは、Real Jukeboxのダウンロード用形式RMP（Real Music Package）に対応しており、Real Jukebox用の合法的なMP3ファイルが置かれたWebサイト（http://www.emusic.com/music/free.htmlなど）からMP3ファイルをダウンロードできる。

こうしたページをブラウザで閲覧し、Real Jukebox用のリンクをクリックすると、FreeAmpのダウンロードブラウザ（**画面10**）が起動してMP3ファイルをダウンロードできる。ダウンロード中は曲の情報が表示され、一時停止や再開も可能だ。

FreeAmpとブラウザを連動させるには、ブラウザのヘルパーアプリケーションとしてFreeAmpを登録する必要がある。たとえば、Netscape Navigatorの場合は、設定ダイアログの[Navigator]－[アプリケーション]ページで[新規]ボタンを押し、MIMEタイプを「application/vnd-rn-rn_music_package」、拡張子を「rmp」、アプリケーションを「freeamp %s」とすればいい。



画面9
テーマの切り替えは、設定ダイアログの[Themes]ページ行う。



画面10
WebからRMPを利用してMP3ファイルをダウンロード。

viライクなキー操作の2画面ファイルマネージャ

Vide

バージョン：0.2.2　　ライセンス：GPL

http://vide.sourceforge.net/

Videは、2つのディレクトリのファイル一覧が表示され、コピーや移動の結果を確認できる2ペイン構成のファイルマネージャだ。viライクなキー割り当てを採用していることが最大の特長で、viのキー操作になじんでいる人にお勧めだ。hjklキーによるカーソル移動をはじめ、「:」コマンドによるファイル処理など、キーボードから手を離すことなくあらゆる処理を行える。vimとの連動もバッチリだ。

●ビルドと日本語への対応

Videは、ファイル一式をtar＋gzipしたtarボールで配布されている。実行にはGTK+ 1.2以降が必要だ。「./configure」→「make」→「make install」という一般的な手順でビルドとインストールを行う。

なお、日本語環境では、ダイアログの表示などが文字化けしてしまうので、ビルドする前に次のように修正しておこう。ソースファイルsrc/vide.cの188行目「gtk_init（&argc, &argv）;」の前に、「gtk_set_locale（）;」という行を追加する。これで、GTK+の設定ファイル（/etc/gtk/gtkrc.jaなど）で設定したフォントセットが使われるようになる。

●起動と初期設定

ktermなどのコマンドラインで「vide」とすると、左右に2つのファイルリスト、中央部にコマンドリストを持つウィンドウが開く（**画面1**）。

最初に、Videから起動するビューアと端末ソフトの設定を行う。[Options]－[Configure]で設定ダイアログを開いて、左のリストの[General - Page 2]をクリックする（**画面2**）。初期設定では、ビューアは「rxvt -e vim」（rxvtでvimを実行）、端末ソフトは「rxvt」だ。ktermでjvimを実行するなら、ビューアの設定を「kterm -e jvim」に変更すればいい。

●カーソル移動はhjklキーでしょ

Videではマウスを使った操作も可能だが、以下ではVideの最大の特長であるviライクなキー操作（リスト1）に基づいて説明しよう。

まず、カーソル移動にはhjklキーが割り当てられている。たとえば、カーソルを1行下に移動させるにはjキーを押せばいい。メニュー項目も同様にhjklキーで移動できる。

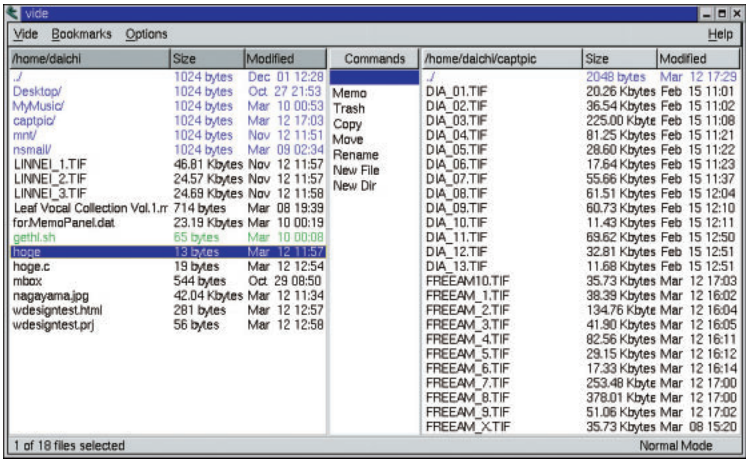
なお、カーソルを持つ「アクティブ」なリストは、インデックス部分が濃い灰色で表示される。カーソルの切り替えには、Tab（あるいはスペース）キーを利用する。

●Enterキーでファイルを処理

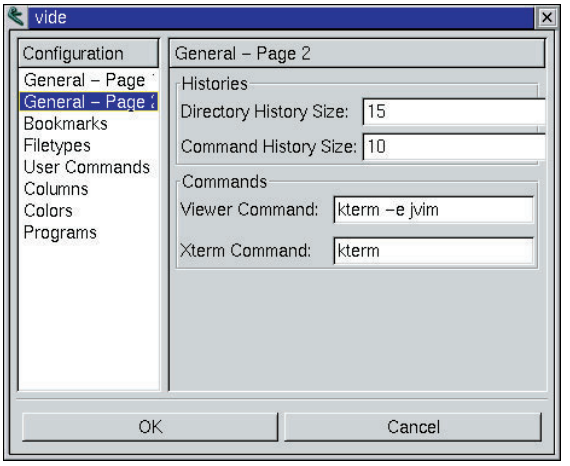
カーソルをテキストファイルまで移動させてEnterキーを押すと、さきほど設定したビューア（初期値はrxvt上のvim）が起動する（**画面3**）。

一方、画像ファイルなどの場合は、ファイルタイプに応じたコマンドが実行される。ファイルタイプに属する拡張子と実行するコマンドは、設定ダイアログの[FileTypes]ページで設定されている（**画面4**）。

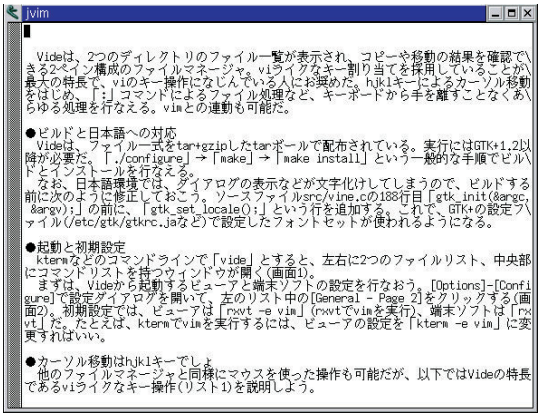
ユーザーが新たなファイルタイプを追加したり、既存のファイルタイプに属する拡張子やコマンドの内容を変更することもできる（**画面5**）。拡張子とコマンドは複数登録できるので、柔軟



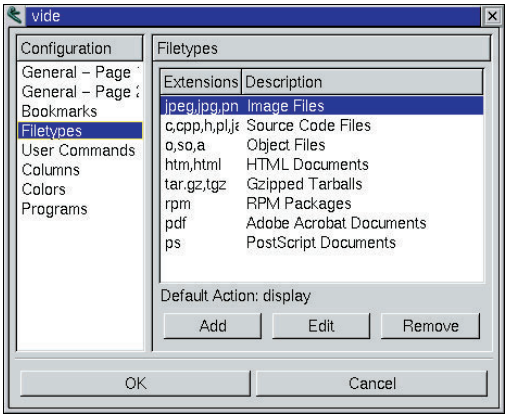
画面1
2つのディレクトリのファイル一覧を同時に参照できる。



画面2
設定はすべてこのダイアログで変更でき、即座に有効になる。



画面3
ビューアの初期設定はvim。ユーザーが変更することも可能。



画面4
ひとつのファイルタイプに複数の拡張子を登録できる。

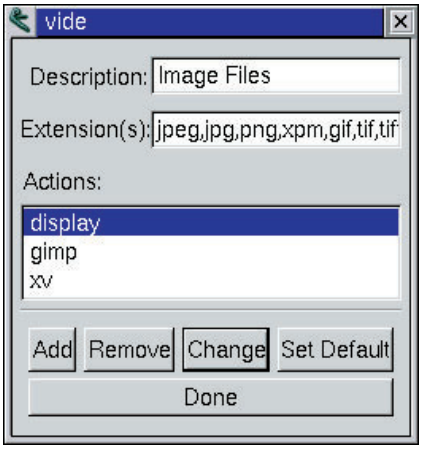
な設定が可能だ。なお、Enterキーで実行されるのは「デフォルト」設定のコマンドだけで、残りのコマンドは、pキーで開くポップアップメニューから実行する。

●ファイル処理を行う

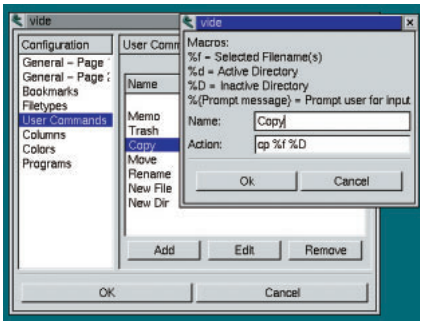
コピーや削除などのファイル処理は、コマンドを直接入力する方法と、コマンドリストから選択する方法がある。通常はカーソルのあるファイルが対象だが、／キーに続けてパターンを指定して複数ファイルを処理対象にすることも可能だ。

コマンドを直接入力する場合は、コロン（：）に続けてコマンドを入力し、最後にEnterキーを押す。たとえば、ファイルを削除するには、「:e」とEnterキーを押せばいい。一方、コマンドリストを利用する場合は、Tab（またはスペース）キーを押してコマンドリストにカーソルを切り替え、目的のコマンド上でEnterキーを押す。

コマンドリストの設定は、設定ダイアログの[User Commands]で変更できる（**画面6**）。アクション設定では、選択されたファイルを表わす「%f」などのマクロを利用可能だ。



画面5
拡張子や実行するコマンドの登録はこのダイアログで行う。



画面6
コマンドリストの内容はユーザーが自由に変更可能だ。

Tab	ファイルリスト・コマンドリスト間の移動
u	1つ上のディレクトリのファイルリストを表示
Enter	ファイルタイプに応じた処理を実行（ファイルの場合）
	ディレクトリのファイルリストを表示（ディレクトリの場合）
p	ファイルパターンごとのメニューをポップアップ
P	汎用のプログラムメニューをポップアップ
j	カーソルを下に移動
k	カーソルを上に移動
h	カーソルを左に移動
l	カーソルを右に移動
n	カーソルをパターンにマッチする次のファイルに移動
Ctrl－f	カーソルを1ページ下に移動
Ctrl－b	カーソルを1ページ上に移動
g	カーソルをリスト最上行に移動
G	カーソルをリスト最下行に移動
／	パターン パターンにマッチする全ファイルを選択
:q	Videを終了
:d	選択したファイルを削除
:m	選択したファイルを移動
:co	選択したファイルをコピー
:sh	カレントディレクトリで端末ソフトを起動
:! cmd	シェルコマンドを端末ソフトで実行
Ctrl－C	ノーマルモードに戻る

リスト1 Videのキー割り当て一覧

viライクなキー操作の2画面ファイルマネージャ

Downloader for X

バージョン：1.13　　ライセンス：フリー

http://www.krasu.ru/soft/chuchelo/

Downloader for Xは、HTTP／FTPプロトコルを利用してファイルのダウンロードを行う自動巡回ソフトだ。指定されたURL以下のファイルを再帰的に取得したり、ファイルの種別を限定することも可能だ。このほか、中断した個所から受信を再開する、指定した時間に受信を開始するといったさまざまな機能を備えている。日本語カタログが付属しており、日本語表示で使える点もうれしい。

●ビルドとインストール

Downloader for Xは、tarボールとRPMパッケージの両方で配布されている。RPMバイナリパッケージはRed Hat 6.1用なので、他のディストリビューションではソースパッケージからバイナリパッケージをリビルドしよう。

「rpm --rebuild nt-1.13-1.src.rpm」とすると、/usr/src/redhat/RPMS/i386にnt-1.13-1.i386.rpmが作成されるので、「rpm -ivh nt-1.13-1.i386.rpm」としてこれをインストールすればいい。

一方、tarボールからビルドする場合は、展開先のmain/face/log.ccの278行目「gdk_font_load」を「gdk_fontset_load」に修正すると、ログ表示ウィンドウの文字化けを解消できる。ビルドとインストールは、mainディレクトリに移動して「make」→「make install」とすればいい。

●基本的な使い方

ktermなどのコマンドラインで「nt」として起動すると、上下2ペイン構成のウィンドウが開く（**画面1**）。上のペインにはダウンロード待ちリスト（キュー）、下のペインには各種メッセージがそれぞれ表示される。

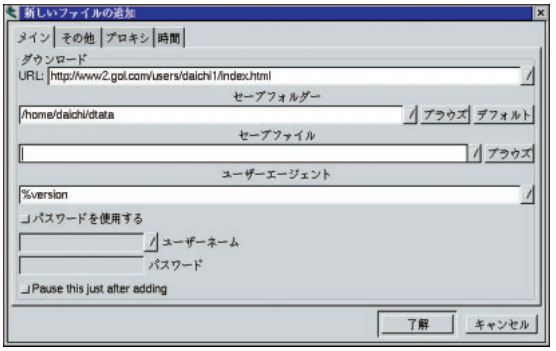
ファイルをダウンロードするには、ツールバー左端の[新しいファイルの追加]ボタンを押してプロパティダイアログを開き、ファイルのURLを入力する（**画面2**）。ファイルを保存する「セーブフォルダ」や、URLとは別の名前で保存する「セーブファイル」なども設定可能だ。

このほか、ブラウズ中のWebページのリンクをドラッグして、Downloader for Xのウィンドウや、アイコン風の「ドラッグ&ドロップトラッシュ」にドロップしてもプロパティダイアログが開く。クリップボード監視機能を利用して、他のアプリでURLをクリップボードにコピーしてもOKだ。これらの場合、プロパティダイアログのURLは自動的に設定される。

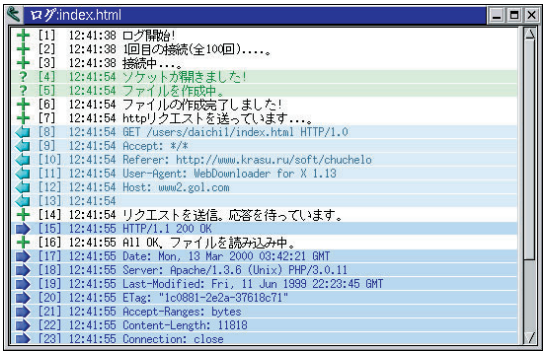
[了解]ボタンを押すと、ダウンロード待ちリスト（キュー）の末尾にファイルが追加され、キューの先頭から順番に、一度にひとつずつファイルがダウンロードされる（変更可）。

ツールバーのボタンで、キューの順番変更や削除、ダウンロードの停止・再開などを行える。また、各キューの右クリックメニューでは、ダウンロード時のログの参照（**画面3**）や、プロパティの変更が可能だ。

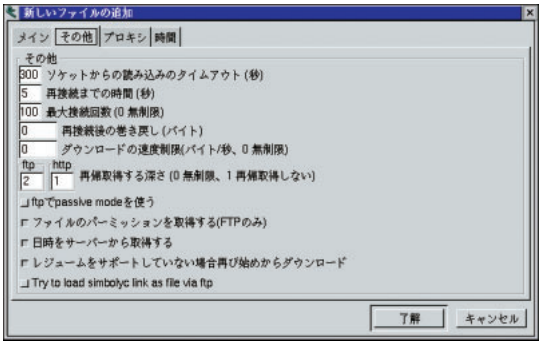
画面1
上のペインにダウンロード対象のファイルが並ぶ。



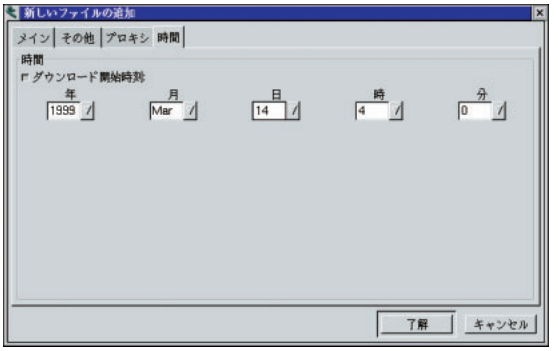
画面2
ファイルのURLなどを入力するプロパティダイアログ。



画面3
各キューごとにダウンロード時のログが保存されている。



画面4
再帰取得の設定はプロパティの[その他]ページで行う。



画面5
ダウンロードを開始日時を設定して、無人運転を行う。

●再帰取得を行う

Webページに含まれる画像やリンク先のページを取得したり、FTPサイトのディレクトリの内容をまるごと取得するには、プロパティダイアログの[その他]ページで、[再帰取得する深さ]を変更する（画面4）。設定値は0以上の整数で、1段階の再帰を行う場合は「2」を設定する。「0」は制限なしの再帰取得、「1」は再帰取得しないことを意味する。

[http]の初期値は1なので、URLで指定したページのHTMLファイルしか取得しない。ページに含まれる画像やリンク先ページを取得するには、[http]を2以上に設定する必要がある。

一方、[ftp]の初期値は2なので、1段階の再帰取得が行われる。つまり、URLにディレクトリを指定すると、そのディレクトリに含まれるファイル（サブディレクトリ以下のものを除く）をすべて取得する。サブディレクトリ以下も含めた全ファイルを取得するには、[ftp]を0に設定すればいい。

なお、再帰取得では、取得先のディレクトリ構造がセーブフォルダ以下に反映される。FTPでは指定したURL部分を除いたディレクトリ構造が作成され、HTTPでは指定したURL（ドメイン名を除く）を含んだディレクトリ構造が作成される。

たとえば、URLに「http://www.hoge.com/users/hogehoge/index.html」を指定して再帰取得を行うと、セーブフォルダ以下にusers/hogehogeディレクトリが作られ、index.htmlなどが格納される。このファイルをブラウザに読み込めば、オフライン状態での閲覧が可能だ。

●無人運転やファイル種別の指定

プロパティダイアログの[時間]ページでは、受信の開始日時を設定できる（**画面5**）。この設定を有効にして[了解]ボタンを押すと、そのキューは待機状態になり、指定された日時になるとダウンロードが実行される。昼間のうちにファイルのキューを作成しておき、テレホーダイ時間に自動受信するといったことが可能だ。

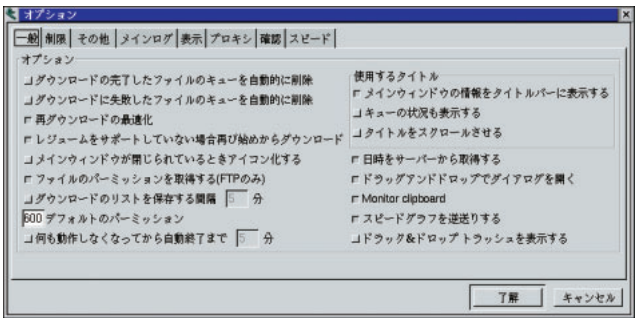
FTPの場合は、URLの末尾にワイルドカードを含むファイル名を指定することで、取得するファイルの種別を制限できる。たとえば、URLを「ftp://ftp.hoge.com/pub/*.jpg」とすると、pubディレクトリで「*.jpg」にマッチするファイル（JPEG画像）だけが取得対象になる。

●設定変更と受信速度制限

プロパティダイアログで設定した内容は、そのキューに対してのみ有効だ。全体の設定を変更するには、[オプション]−[一般]を選択してオプションダイアログを開く（**画面6**）。

設定項目は多岐にわたる。[制限]ページでは同時にダウンロードするファイルの最大数、[スピード]ページでは受信速度の制限値を設定可能だ。このほか、プロパティダイアログで個別に設定する内容、たとえばセーブフォルダやプロキシなどの初期値もこのダイアログで設定できる。

受信速度の制限は、大きなファイルをダウンロードしながら、Webページを快適に閲覧したい場合に便利な機能だ。ツールバー中央部の[スピードレベル1]ボタン（赤）や[スピードレベル2]ボタン（黄）を押すと、オプションダイアログの設定値が受信速度の上限になる。一方、ファイルのダウンロードに全力を注ぐなら、初期設定である[スピード制限なし]ボタン（緑）を押せばいい。



画面6
設定ダイアログには、さまざまな設定がジャンル分けされている。

カラー構文表示のHTMLエディタ

Web Designer

バージョン：0.0.19　　ライセンス：フリー

http://webdesigner.linuxbox.com/

Web Designerは、カラー構文表示に対応したタグ挿入型のHTMLエディタだ。ツールバーのボタンを使ってタグを効率よく入力でき、内蔵のプレビューアや外部のブラウザで実際の表示を確認できる。編集中のタグの文法が表示されるヘルプバーや、複数のファイルをまとめて扱うプロジェクト機能も用意されている。なお、ソースを一部変更することで、日本語の入力・編集が可能になる（日本語EUCのみ）。

●ビルドと日本語対応

Web Designerは、tarボールでのみ配布されている。tarボール内のパスが「../webdesigner-0.0.19/…」となっているため、「mkdir webdesigner-0.0.19; cd webdesigner-0.0.19」として、展開用のディレクトリに移動してから、「tar xzf webdesigner-0.0.19.src.tar.gz」としてtarボールを展開するとよいだろう。

ビルドとインストールは「./configure」→「make」→「make install」という一般的な手順だ。なお、tarボールにconfig.cacheが含まれているので、ビルド前に「rm config.cache」として削除しておこう。

日本語EUCで書かれたテキストを入力・編集するには、ビルド前にソースを修正する必要がある。この修正を行うパッチ（webdesigner-ja.patch）を用意した。tarボールの展開先で「patch -p1 < webdesigner-ja.patch」とすればいい。

なお、Web Designerの動作には、GTK+1.2.2以降とGNOME（内蔵プレビューアが、GNOMEのGtkXmhtmlライブラリを使用するため）が必要だ。

●基本的な使い方

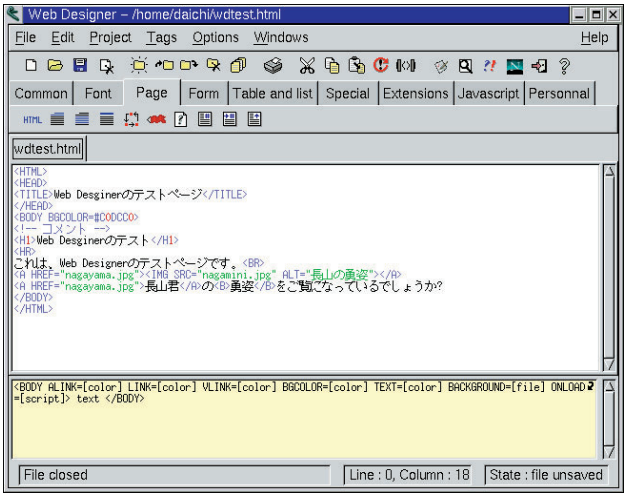
ktermなどのコマンドラインで「wdesign」として起動すると、2つのペインと、数多くのボタンが並んだツールバーを持つウィンドウが開く（**画面1**）。タグ挿入用のボタンはツールバーの2段目にあり、タブ付きページでジャンル分けされている。

上のペインはファイル編集用の領域で、HTMLのタグやC言語の予約語などがカラー表示される。複数ファイルを同時に編集でき、その場合はペイン上部のタブでファイルを切り替える（**画面2**）。このほか、[Windows]－[New window]で複数のウィンドウを開いて編集することも可能だ。

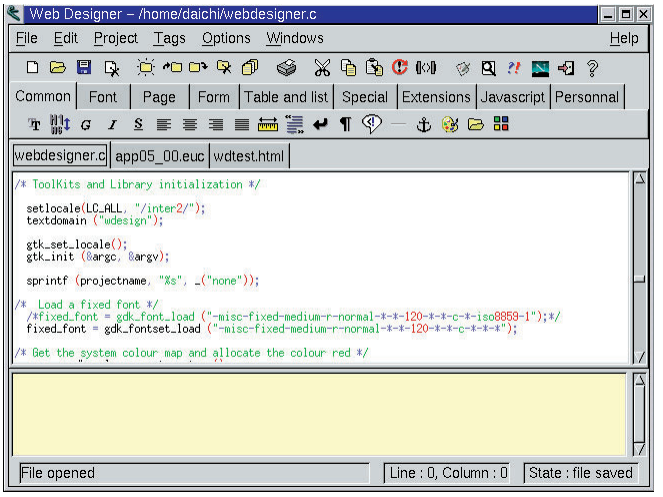
一方、下のペインはタグの文法が表示される「ヘルプバー」で、追加できる属性などが一目でわかる。ヘルプバーが必要ない場合は、ツールバー右の[Toggle Help bar]ボタンで消すことも可能だ。

使い方は一般的なタグ挿入型のHTMLエディタと同じで、ツールバーのボタンを押したり、[Tags]メニュー以下の項目を選択すると、対応するタグがカーソル位置に挿入される。<B>、</B>のようにテキストを囲むタイプのタグは、テキストを選択した状態でボタンを押せばいい。

日本語入力の切り替えはShift－スペースキーで行う（Ctrl－Oキーはファイルオープンダイアログにバインドされている）。入力・編集ともに問題なく行える（**画面3**）。



画面1
Web Designerのウィンドウ。HTMLのタグがカラー構文表示される。



画面2
複数のファイルを同時に編集可能。切り替えはタブをクリック。

画面3
日本語の入力や編集も問題ない（日本語EUCのみ対応）。

画面4
内蔵プレビューアによる確認はセーブ前の状態でも可能だ。

●内蔵プレビューアと外部ブラウザ

作成したHTMLファイルは、内蔵のプレビューアと外部のWebブラウザ（初期値はNetscape Navigator）で見栄えを確認できる。

ツールバー右の[Internal preview]ボタンを押すと、内蔵プレビューアが起動してページを表示する（**画面4**）。起動は高速で、セーブする前の内容もプレビュー可能だ。ただし、GtkXmhtmlライブラリによるページレイアウトは、一般的なブラウザとはかなり異なる。文字主体のページならそれほど気にならないが、画像を多用したページではあまりあてにできない。

一方、[View in netscape]ボタンを押すと、Netscape Navigatorが起動して、セーブ時点のページの内容を表示する（**画面5**）。なお、外部ブラウザ実行中はWeb Designerの操作は行えず、画面表示も更新されない。

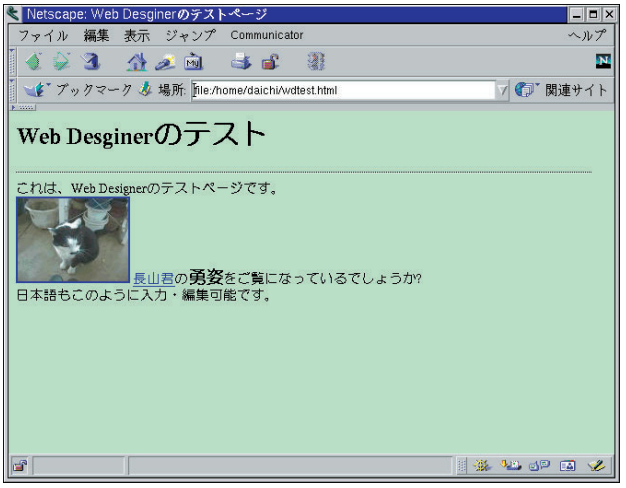
外部ブラウザのパスは、[Options]－[Browser path]で変更できる（**画面6**）。初期値は「/usr/bin/netscape %s」だ。なお、LASER5ではこれを「/usr/X11R6/bin/netscape %s」に修正する必要がある。

●プロジェクト機能を利用する

一般に、Webサイトは複数のHTMLファイルで構成されているので、Web Designerには、こうした複数のファイルを「プロジェクト」としてまとめて扱う機能が用意されている。

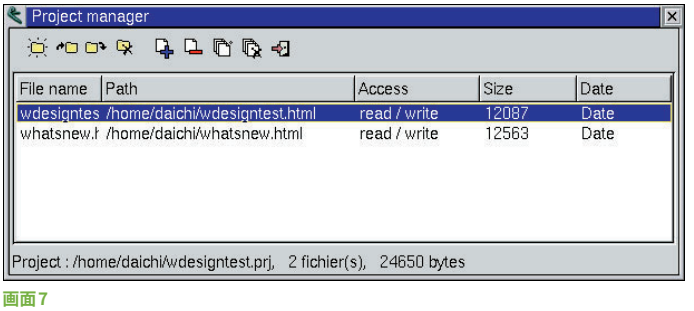
ツールバー中央部の[Project Manager]ボタンを押すと、プロジェクトの管理を行うプロジェクトマネージャが開く（**画面7**）。新たにプロジェクトを作成するには、ツールバー左端の[New project]ボタン、既存のプロジェクトを読み込むには[Open project]ボタンを押せばいい。

プロジェクトには、[Add file]／[Remove file]ボタンでファイルを追加・削除できる。プロジェクトに含まれるファイルをWeb Designerで編集するには、リスト上で右クリックし、メニューから[Edit]を選択すればいい。さらに、[Open all files]／[Close all files]ボタンで一括オープン・クローズすることも可能だ。



画面5
外部ブラウザのNetscape Navigatorで実際の表示を確認する。

画面6
外部ブラウザのパスはこのダイアログで変更できる。



画面7
複数のファイルをまとめて扱うプロジェクトマネージャ。

リアルな挙動の3Dビリヤードゲーム

Gtulpas

バージョン：1.0.0　　ライセンス：GPL

http://www.suse.cz/gtulpas/
http://www.mesa3d.org/ （MesaGL）
http://www.student.oulu.fi/~jlof/gtkglarea/ （GtkGLArea）
http://www.gnu.org/software/guile/ （guile）

Gtulpasは、9ボールやスヌーカー、キャロムをプレイできる3Dビリヤードゲームだ。ビリヤードゲームでは、ボールの挙動をどれだけ本物らしくできるか重要だが、Gtulpasの物理エンジンは優秀なほうで、捻りの影響やクッション際での動きはなかなかリアルだ。ウィンドウを分割して複数の視点からテーブルを眺めることもできる。動作には、GNOME、MesaGL、GtkGLArea、guileが必要だ。

●ビルドとインストール

Gtulpasは、ファイル一式をtar＋gzipしたtarボールで配布されている。ビルドはconfigureスクリプトを使う一般的なものだ。ただし、「./configure --prefix=/usr」→「make」→「make install」として、/usr以下にインストールする必要がある。--prefixオプションを付けずに/usr/local以下にインストールすると、起動時に初期化ファイルなどの読み込みに失敗して異常終了してしまう。

このほか、テンキーのないキーボードで視線を切り替えられない、ヘルプブラウザでヘルプが表示されないという不具合がある。この2点を解決するパッチ（gtulpas-lm.patch）を用意した。ビルド前にGtulpasの展開先で、「patch -p1 < gtulpas-lm.patch」としてソースを修正してほしい。

動作に必要なライブラリは、ビルド前にあらかじめインストールしておく必要がある。特に、3Dライブラリの MesaGLは、最新の3.1（3.1beta3以上）が必要なので、3.0や3.1beta1などを使っている場合は入れ換えよう。以下では、MesaGL 3.1、guile 1.3.4、GtkGL Area 1.2.2を使用している。

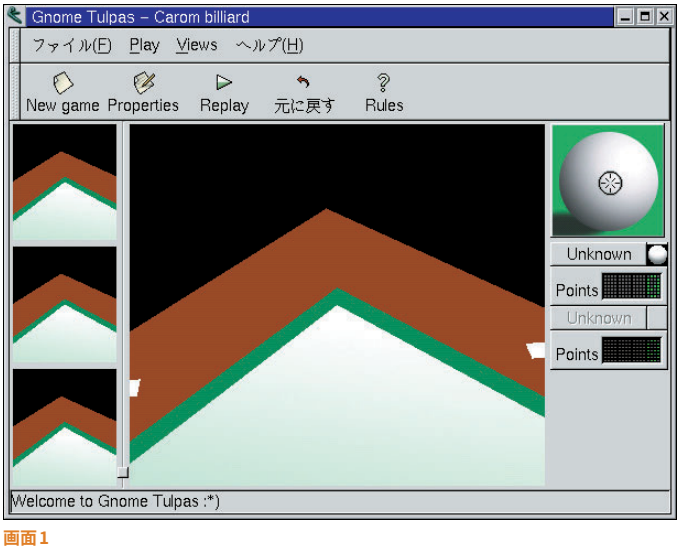
●画面表示とゲームの切り替え

ktermなどのコマンドラインで「gtulpas」とすると、スプラッシュが表示されてウィンドウが開く。ただし、これだと端末画面にボールの座標などの情報が大量に表示されるので、実際には「gtulpas > /dev/null」としたほうがよいだろう。

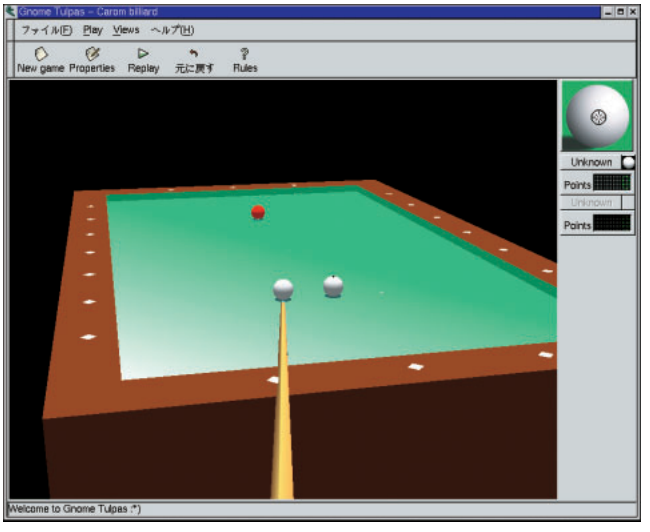
ウィンドウには、中央のメインビューのほか、左側に3つのミニビューが用意され、さまざまな角度からテーブルを眺められる（**画面1**）。ミニビューが必要ないなら、[Views]－[Show mini views]で消してしまおう。ウィンドウのサイズは自由に変更可能だ。

プレイできるゲームは、ポケット（穴）のないテーブルで行う三つ球のキャロム（**画面2**）、ポケット台で行うおなじみの9ボール（**画面3**）、少し大きめのポケット台と小さな球で行うスヌーカー（**画面4**）の3種類だ。ゲームの切り替えは、[File]－[Select game]で行う。

ゲームが始まったら、[ファイル]－[Players]でプレイヤー名を登録しよう。現時点では、ローカルマシンで人間同士の対戦のみ可能だ。2人のプレイヤー名や得点、現在の順番、ファウルの回数といった情報は、メインビューの右側に表示される。



画面1
初期設定ではウィンドウに4つのビューが表示される。



画面2
三つ球のキャロム。手球以外の2つのボールに当てれば得点だ。

●英字キー＋左ドラッグで操作

ゲーム中の操作はマウスとキーで行う（**リスト1**）。基本的な操作は、「英字キーを押しながらマウスを左ドラッグする」というもの。たとえば、Cキーを押しながらマウスを上下にドラッグすると、キューのストローク（手球を撞く動作）になる。

なお、プレイヤーの視線は、マウスの左・中ドラッグで変更できるが、思い通りのアングルにすることはなかなか難しいため、代表的な視線に切り替える0〜9キーが用意されている。

お勧めは、キューの向きに視線を連動させる0キーだ。左ドラッグだけでキューの向きを変えられるし、手球が常にビューの中心に位置するので狙いをつけやすい。微妙な調整が必要なときはCtrlキーを併用する。右上のボールの撞点を変更すると、手球に捻りや押し引きを加えられる。

キューの向きと撞点が決まったら、C＋左ドラッグ（または右ドラッグ）で手球を撞く。的球がポケットに入るか（9ボール、スヌーカーの場合）、手球が2つの的球に当たる（キャロムの場合）と、背景が青くフラッシュして続けてプレイできる。そうでなければ、もう一人のプレーヤに交代だ。なお、ファウルを犯した場合は、画面が赤くフラッシュする。

9ボールやスヌーカーのブレイク時や、9ボールで相手がファウルした後は、手球がフリーボールとなり、M＋左ドラッグで自由に位置を変更できる。また、スヌーカーでは、赤球をポケットした後、次に狙う球（赤以外）をクリックで指定する必要がある。このほか、直前のショットを取り消したり、リプレイすることも可能だ。

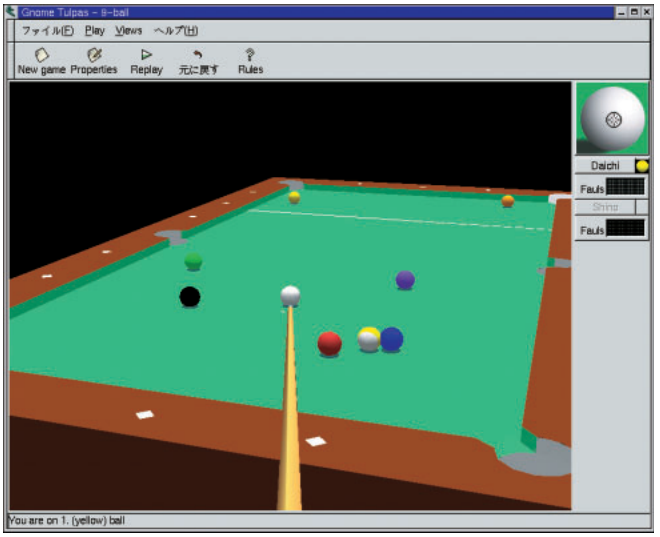
C＋左ドラッグ	キューをストローク
右ドラッグ	（同上）
V＋左ドラッグ	キューを回転
M＋左ドラッグ	ボールを移動（フリーボール時）
0	キューの向きに連動した視線に切替
1〜9	テーブルを八方から見た視線に切替
Z＋左ドラッグ	ズーム
＋／-	ズームイン／アウト
P＋左ドラッグ	平行移動
中ドラッグ	（同上）
R＋左ドラッグ	回転
左ドラッグ	（同上）
S＋左ドラッグ	自動回転
Ctrl－S	自動回転の停止
H	ヒント表示（トグル）

リスト1 マウス、キー操作一覧

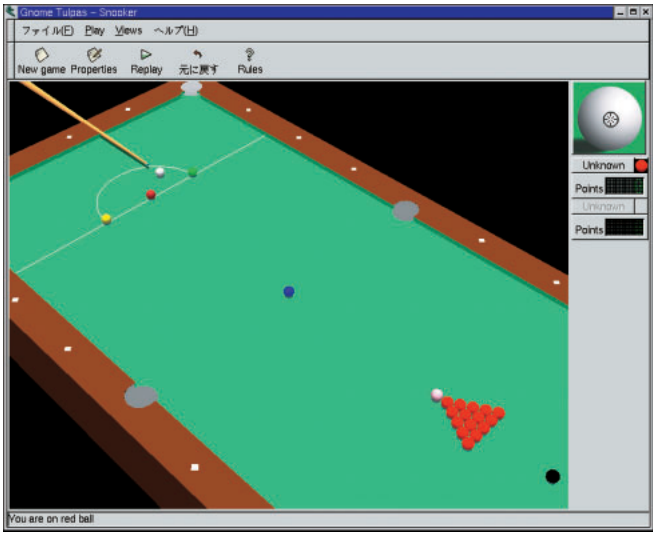
●ゲームの追加も可能

Gtulpasは、各ゲームのルールや、テーブルの大きさなどの設定を、/usr/share/gtulpas/rulesおよびschemeディレクトリに置かれたguileスクリプトから読み込む。

ルールや設定を追加して、新しいゲームを組み込むことも可能だ。たとえば、三つ球のキャロムのルールを少し修正すれば、四つ球のキャロムのルールを作成できる。腕に自信のある人は挑戦してみよう。



画面3
おなじみ9ボール。ファウル後は手球がフリーボールになる。



画面4
赤球→他の色球という順番でポケットするスヌーカー。

Xのルートウィンドウに常駐するカレンダー

X Desktop Calender

バージョン：0.9c　　ライセンス：GPL

http://www.shiratori.riec.tohoku.ac.jp/~jir/linux/products/xdkcal/index-j.html

X Desktop Calender（以下xdkcal）は、Xのルートウィンドウ（背景）にカレンダーを貼りつけるソフトだ。常駐タイプなので自由に取り外し可能だ。また、表示に使われる文字のフォント、各部の色などを自由にカスタマイズできる。なお、週や月の名前はロケールデータベースから読み込まれるが、Linuxには対応する日本語の文字列データベースが用意されていないため、英語表記で表示される。

●ビルドと実行

xdkcal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を行うと、/usr/local/binに実行ファイル（xdkcal）がコピーされる。

ktermなどのコマンドラインで「xdkcal &」として起動すると、Xのルートウィンドウの左上隅に、今月の小さなカレンダーが表示される。ルートウィンドウに描画されるため、他のウィンドウの邪魔にならない。

●起動時オプションで設定変更

カレンダーの表示位置や使用する文字のフォント・色などの設定は起動時オプション（**リスト1**）で設定する。文字の色は細かく設定可能だ。たとえば、座標（512,16）にフォントセット「-alias-*」を使って、デフォルト色を黒で表示するなら、

```
$ xdkcal -x512 -y16 -f '-alias-*'
-C black &
```

とすればいい（**画面1**）。このほか、月や週の名前を表示しない、横一列に日付を表示する（**画面2**）といったオプションも用意されている。

一度オプションで指定した内容は設定ファイル（~/.xdkcalrc）に保存され、次回からは自動的に読み込まれる。フォントセットを細かく指定するなど、コマンドラインでは面倒な場合、設定ファイルを直接エディタで書き換えてもいい。

なお、いったん好みの設定が決まった後で、別の設定を試してみるようなときは、設定内容をファイルに保存しない-Nオプションをあわせて指定するとよいだろう。

画面1
Xのルートウィンドウにカレンダーが表示される。

−h	ヘルプを表示
−x X	X座標を指定
−y Y	Y座標を指定
−f	フォント フォントセットを指定
−C 色	デフォルト色を指定
−H 色	休日の色を指定
−S 色	土曜の色を指定
−T 色	今日の色を指定
−M 色	月の色を指定
−t on|off	透明化（初期値off）
−p 値	パディング値を指定
−s 値	スタイル値を指定 （0:通常、1:ライン）
−m on|off	月表示（初期値on）
−w on|off	週表示（初期値on）
−n 値	週の文字数（初期値2）
−N	設定を保存しない

リスト1 xdkcalの起動時オプション一覧

画面2　文字のフォントや色は自由にカスタマイズ可能だ。

3D表示された太陽系をヒッチハイク

Hitchhiker 2000

バージョン：0.3　　ライセンス：GPL

http://hirame.hiram.edu/~worleyam/
http://www.mesa3d.org/ （MesaGL）
http://www.student.oulu.fi/~jlof/gtkglarea/ （GtkGLArea）

Hitchhiker 2000（以下hh2000）は、太陽や惑星、衛星、彗星、小惑星などが3D表示された宇宙を眺めるシミュレーションソフトだ。カメラの位置や表示の中心となる物体をリストから選択でき、マウス操作で表示を拡大・回転させられる。表示日時を変更したり、時間の進み方を一時停止（あるいは加速）することも可能だ。なお、動作にはGNOME、OpenGL（またはMesaGL）、GtkGLAreaが必要だ。

●ビルドとインストール

hh2000は、ビルド済みのバイナリとソースのtarボールがそれぞれ配布されている。バイナリのtarボールは、展開先のディレクトリで「sh setup.sh」とすればインストールが行われる。ソースのtarボールは、「./configure」→「make」→「make install」という一般的な手順でビルドとインストールを行う。なお、必須ライブラリのMesaGLとGtkGLAreaは、hh2000をビルドする前にインストールしておく必要がある。

●3つの表示モードで太陽系を散策

ktermなどから「hh2000」として起動すると、太陽系を外宇宙から見た眺めがウィンドウに表示される。ウィンドウサイズの初期値は少し小さめなので、適当な大きさに広げるといいだろう（**画面1**）。

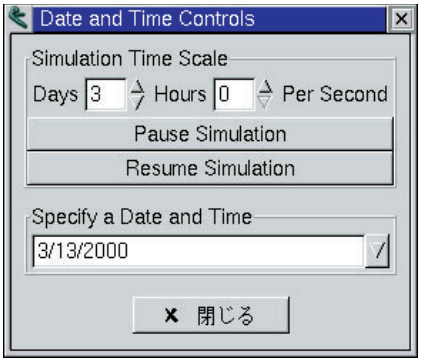
表示モードは、天体を外宇宙から眺める「Deep Space」（初期値）、その天体の軌道上から眺める[Local Orbit]（**画面2**）、指定した天体から眺める[Sky View]の3種類。表示の中心となる[Camera Taerget]を切り替えるには、右下のリストから適当な天体を選択し、[Camera Target]の三角ボタンをクリックすればいい。

Deep Space／Local Orbitモードでは、左右ボタンのドラッグにより表示の回転や拡大縮小が可能だ。一方、Sky Viewモードでは、こうした操作の代わりにカメラの位置を[Camera Location]で指定する。指定方法は[Cameta Target]と同じだ。

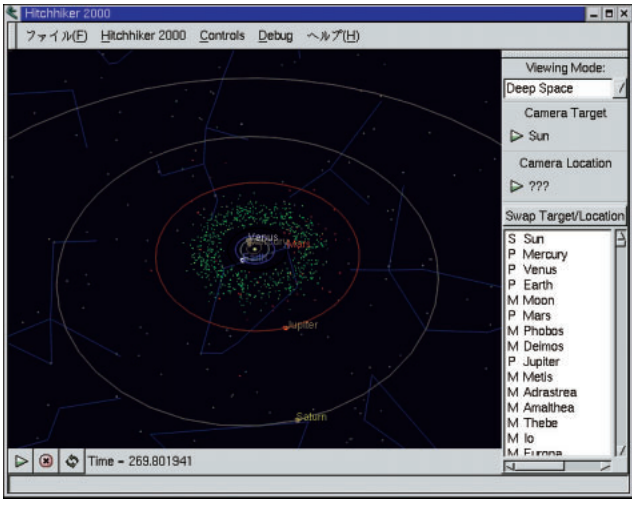
●表示する日時なども変更可能

[Controls]－[Time and Date]で開くダイアログでは、1秒あたりの表示で経過させる時間、表示する日時の設定のほか、時間経過の一時停止や再開も可能だ（**画面3**）。

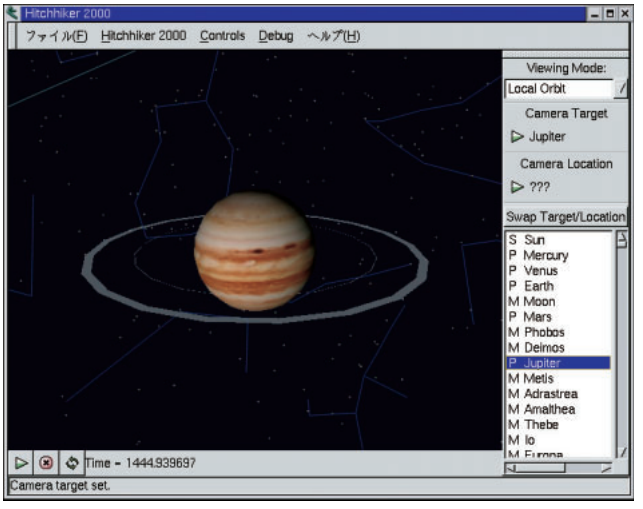
また、[Controls]－[Display Options]で開く設定ダイアログでは、惑星の軌道を表示するか、惑星を実際より大きく表示するか、惑星の名前を表示するかといった、細かな設定を行える。このほか、表示される天体のデータベースを別ウィンドウに表示させることも可能だ。



画面3
表示する日時や、1秒あたりに経過させる時間などを設定。



画面1
Deep Spaceモードで外宇宙から太陽系を一望する。



画面2
Local Orbitモードで、木星を至近距離から眺める。

GNOMEのパネルにメモを貼りつける

MemoPanel

バージョン：1.5　　ライセンス：GPL

http://www.geocities.co.jp/NeverLand/1645/memopanel.html
http://www.geocities.co.jp/NeverLand/1645/HtmlHeadLine.html（HtmlHeadLine.sh）
http://www.eecs.umich.edu/~dhelder/misc/gnet/（GNet）

MemoPanelは、GNOME環境で画面下に表示されるパネルにメモを貼りつけるアプレットだ。日本発の国際化アプリで、ソースを変更することなく日本語の文章をメモできる。複数行にわたるメモを残したり、文字や背景の色を指定することも可能。さらに、ファイルの内容を1行ずつ表示する機能も持つため、他のツールと組み合わせるとパネルにヘッドラインニュースを表示できる。

●基本的な使い方

ビルドとインストールは、「make」→「make install」とするだけでいい。コマンドラインで「memopanel」とするか、パネル上で右クリックし、ポップアップの[アプレットの追加]－[ユーティリティ]－[MemoPanel]を選択すると、パネル上にメモが表示される（**画面1**）。

メモ上で右クリックして[Properties]を選択するとプロパティダイアログが開く（**画面2**）。ここでは、メモの内容のほか、文字色や背景色を設定可能だ。なお、複数行にわたるメモを書くには、改行の代わりに「__」（アンダーバー）を利用する。設定内容は自動的に保存され、次にGNOMEを起動した際には、同じ位置に同じ内容のメモが表示される。

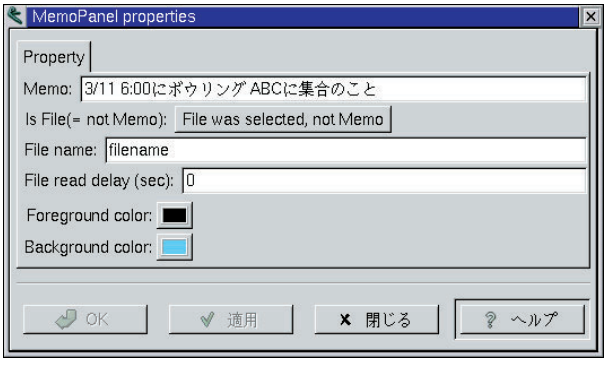
●ファイルの内容を更新表示する

MemoPanelでは、ファイルの内容を1行ずつ、時間をおいて表示する機能も用意されている。これを利用する典型的な例が、同じ作者の「HtmlHeadLine.sh」というシェルスクリプトとの組み合わせだ。

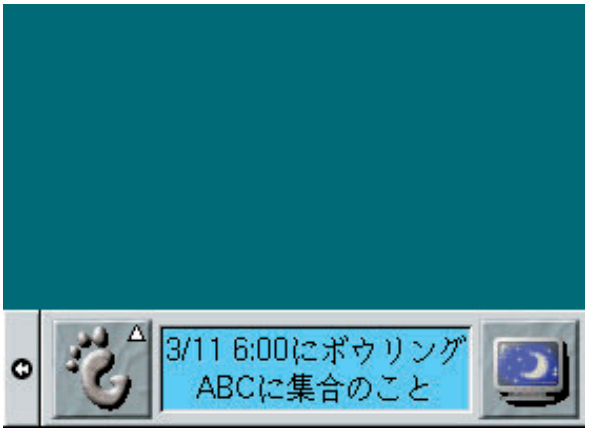
HtmlHeadLine.shは、freshmeatやslashdotなどのニュースサイトから情報を取得し、自動的にニュースヘッドラインを作成する。実行にはネットワークライブラリの「GNet」が別途必要だ。初期設定ではHTML形式でヘッドラインを作成するが、スクリプト中で「FILE_ONLY=YES」とすると、プレーンテキストのヘッドラインを~/for.MemoPanel.datに出力する。

そこで、MemoPanelのプロパティの[File was selected, not Memo]ボタンを押し、ファイル名を「~/for.MemoPanel.dat」、更新間隔を「5」（秒）に設定する。これで、ヘッドラインの内容が5秒ごとに1行ずつ、GNOMEのパネルに表示される（**画面3**）。

さらに、HtmlHeadLine.shを一定時間ごとに繰り返し実行することで、ヘッドラインを最新の状態に保つことができる。具体的には、crontabに登録するか、whileとsleepを使ったシェルスクリプト（Webサイトを参照）と組み合わせればいい。



画面2
メモの内容や文字色・背景色をプロパティダイアログで設定。



画面1
GNOMEのパネルにメモが表示される。日本語も使用可能だ。

画面3
HtmlHeadLine.shと組み合わせてヘッドラインを表示する。

xsetの設定をGUIで行う

GTK xset

バージョン：0.3　　ライセンス：GPL

http://www.seindal.dk/rene/software/gxset/

GTK xsetは、X Window Systemの各種設定を行うxsetのフロントエンドだ。マウスやキーボード、スクリーンセーバ、フォントパスなどの設定がジャンル別に分類され、マウスなどを使って設定を変更できる。日本語カタログが最初から用意されている点もうれしい。設定内容はシェルスクリプトとして保存でき、他のシェルスクリプトから容易に呼び出せる。動作にはGTK+ 1.2以降が必要だ。

●ビルドから起動まで

GTK xset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は、「./configure」→「make」→「make install」という一般的な手順だ。内部でxsetを実行して処理を行うため、xsetがPATHの通ったディレクトリに置かれている必要がある。たいていの場合、xsetはXFree86のパッケージに含まれているので、特にインストールする必要はないだろう。

ktermなどのコマンドラインで「gxset」として起動すると、ジャンル別のタブが並んだウィンドウが開く。日本語環境では、付属の日本語カタログ（ja.po）を利用して、項目名などが日本語で表示される。

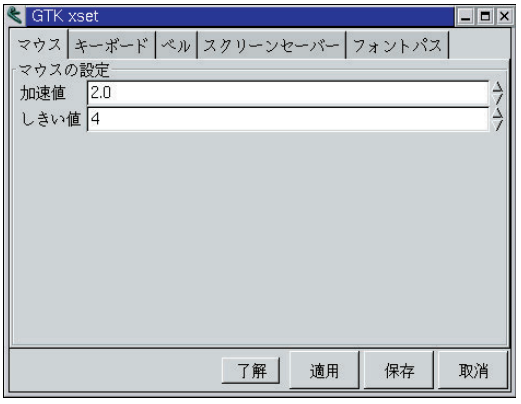
●マウスの加速などを変更する

GTK xsetでは、xsetのコマンドラインオプションの代わりに、マウスやキーボードを使って設定を行う。設定項目は「マウス」「キーボード」「ベル」「スクリーンセーバー」「フォントパス」の5種類にページ分けされており、タブで切り替えられる。また、項目上にカーソルをしばらく停止させると、内容に関するチップヘルプが表示される。

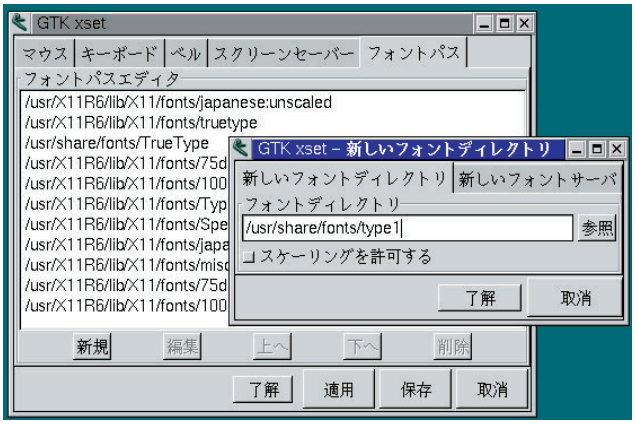
マウスページ（**画面1**）では、カーソルの加速に関する設定を行う。設定項目は、加速の度合いを表わす「加速値」と、加速させるかどうかの境界となる「しきい値」の2つだ。また、フォントパスページ（**画面2**）では、X用のフォントが置かれたディレクトリや、フォントサーバの追加・編集・削除・順序変更が可能だ。

こうした設定は、[了解]／[適用]ボタンで有効になる。終了後も設定は変更されたままだが、Xを再起動すると元に戻る。現在の設定を保存するには、[Save]ボタンでダイアログを開き、ファイル名（初期値は~/.xset.sh）を指定すればいい（**画面3**）。

このファイルの内容は、現在の設定に相当する起動時オプションを付けてxsetを実行するシェルスクリプトだ。つまり、GTK xsetの代わりにこのスクリプトを実行するだけで、保存した設定に変更できる。



画面1
設定内容はジャンル別のページに分かれている。



画面2
Xが参照するフォントパスも変更可能だ。

画面3
現在の設定内容をシェルスクリプトとして保存できる。

隠喩としてのコンピュータ

インターネットにおける「現実界」

文：豊福 剛
Text：Tsuyoshi Toyofuku

GUI以後、スクリーンの背後を問わない態度が一般化し、コンピュータはスクリーンと見なされるようになった。シェリー・タークルは、これをインターフェイス・バリューと呼び、ポストモダンと関連づけた。同様の態度変更は、インターネットにおいても反復されたように思われる。WWW以後、インターネットはブラウザが表示される外見に還元されるようになり、GUIと同様、ユーザーはその背後にあるものを問わなくなっているようだ。

もちろん、ブラウザが表示するページの外見は、HTMLファイルをレンダリングした結果であり、このHTMLファイルはユーザーには隠されていない。ユーザーは、ページという外見の背後にあるHTMLによる記述を読むことができる。この点はGUIと異なり、むしろ外見の背後は積極的に問われているといえるだろう。

さらに、ユーザーは、WWWの外見をブラウズするだけでなく、自ら記述したHTMLファイルをISP（インターネットサービス・プロバイダ）などが提供するサービスを利用して、情報発信することができる。ISP側にあるサーバの然るべきディレクトリにHTMLファイルを転送すれば、情報発信する側に立つことができる。このようなユーザーに対する公開性から、WWWは透明なメディアであるかのように見える。

rootという権力

しかし、サーバにおいては、ユーザーに対するアクセス権の制約というかたちで、権力関係が現実に存在している。あらゆるサーバにはrootという特権的なユーザーが存在する。rootはサーバ上のあらゆるリソースに対してアクセス可能であるのに対して、一般ユーザーのアクセス権は限定されている。

この権力関係の内実は、ユーザーには不透明である。仮にrootがサーバ上で悪質な行為を行ったとしても、一般ユーザーには、それをチェックすることが技術的にできない。rootの良心と倫理を信じるしかないのだ。

このようなサーバにおける権力関係は、HTMLファイルを置くだけであれば、問題にならなかった。しかし、スタティックなHTMLファイルの公開から、さらに一歩進んで、CGIなどを利用するWWWに拡張しようとした途端に、直面せざるをえない現実となる。

ちなみに、最近のISPでは、PerlによるCGIプログラミングが可能なところは珍しくない。グローバルオンライン

というISPでは、MySQLというフリーソフトウェアのデータベースも利用できるようだ。このようにMySQLやPostgreSQLなどのデータベースを利用できるISPは、今後増えていくに違いない。またサーバサイドのプログラム実行環境として、Perlだけでなく Java をサポートするISPも出てくるだろう。

ISPが提供するWWWのバックエンド環境は、今後ますます充実していくと思われる。しかし、ISPが所有するサーバをレンタルで利用する以上、そのサーバにrootが存在することは避けられない。これは賃貸マンションに住んでいる以上、マスターキーをもった管理人が存在するのと同じことだ。映画『硝子の塔』では、マンションの住人のプライベートな生活は、管理人のモニターテレビによって常時覗き見されていた。rootが存在することは、このような覗き見が技術的に可能であることを意味する。

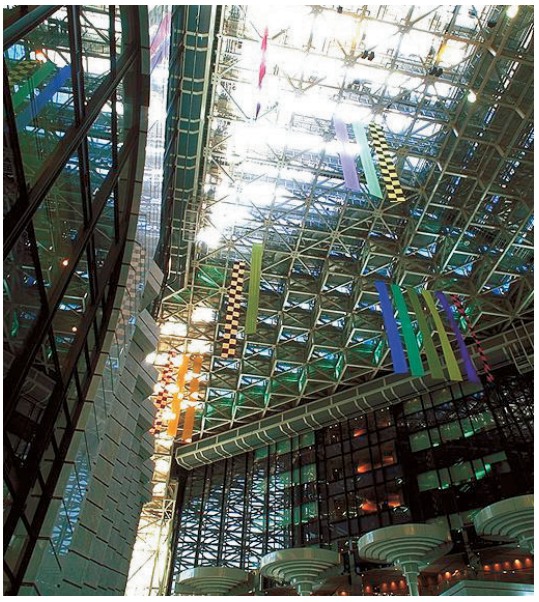
セキュリティ技術の限界

もちろん、サーバ上のデータと現実のプライベートな生活は同じではないし、覗き見される可能性を考慮したうえで、他人に見られてまずいデータは一切置かないという判断は、とりあえず賢明だ。

しかし、たとえば通販サイトを立ち上げるとしたら、顧客からの注文データの扱いには、慎重にならざるをえないだろう。そこには、顧客の名前や電子メールアドレス、場合によっては、住所や電話番号など、プライバシーにかかわるデータが含まれている。これらのデータを平文のままサーバにファイルとして保存した場合、rootによるアクセスを回避することはできない。

MySQLやPostgreSQLなどのデータベースに格納するとしても、そのデータベースの管理者がいる以上、悪意によってデータが盗まれる可能性は否定できない。暗号化を施すとしても、暗号化処理をPerlで記述していたら、解読されるのを防げない。注文データをメールで送信するとしても、メールプログラムに手が加えられていれば、データが読まれてしまう。公開鍵方式の暗号を使って、サーバ側で公開鍵で暗号化し、それをメールで転送して、秘密鍵で復号化するしかないだろう。

このように考えると、インターネットにおけるセキュリティ技術には盲点があったように思われる。ファイアウォールに代表されるようなセキュリティ技術は、サーバに対する外部からのアクセスを制限する。サーバに至るまでの



通信に対するセキュリティとしては、たとえばSSLなどを使うことができる。しかし、エンドにあるサーバで処理される受信データは無防備である。

つまり、UNIXにおけるrootユーザーにかぎらず、これまでのコンピュータは特権ユーザーによる超越的な管理を前提としているのだ。rootの存在によって帰結する不透明な権力関係を完全に解決するには、サーバを自ら所有するしかない。

インターネットの商用サービスが開始された'95年当時、それまでスタンドアローンで使っていたパーソナルコンピュータが、世界規模のコンピュータ集合体に接続可能になることを想像するとき、ある種エロス的な興奮を覚えたものだ。とりあえず電話回線を利用したダイヤルアップ接続を利用せざるをえないとはいえ、2005年くらいになれば、自宅からでも大容量の通信回線による常時接続が実現されるだろうと期待した。そしてLinuxの成長とともに、フリーソフトウェアによるサーバ構築が現実となったとき、残された問題は通信回線のコストだけであるように思われた。

しかし、この5年間のインターネットの変化を振り返ってみると、クライアントとサーバの格差をなくす方向よりも、その格差を増大させる方向に進展してきたように思われる。自前のサーバを立ち上げるために、たとえばOCNなどの比較的低価格なサービスを利用したとしても、それらの回線容量にはおのずと限界があり、アクセスが急増した場合には、その負荷に耐えられなくなるだろう。そうすると、どうしても高速バックボーンに直結できるサービスを利用せざるをえない。そのようなサービスが、ディスクスペースのレンタルである以上、どうしてもrootの存在は避けられない。

rootにとっての倫理

3月11日の朝日新聞は、大塚製薬のホームページに登録された個人情報の流出について報道している。「カロリーメイトバランスチェック」に登録された約9900人分の自宅住所、電話番号、身長、体重、妊娠に関する情報といった個人情報が、閲覧できる状態になっていたらしい。IDとパスワードを入力する仕組みになっていたらしいが、このIDとパスワードの一覧表がアクセスできたという。

大塚製薬は、サイトの管理を業者に委託していたそうで、2000年問題への対応における業者の作業ミスによって、こ

れらのデータが「防護システム」（ファイアウォールのことか？）の外側に出たのが原因らしい。ということは、個人情報は平文のまま、CSVなどのテキスト形式で保存されていたのだろう。まったく、rootの権力性を云々する以前の問題だ。大手製薬会社にして、この現実だから、この手のサービスの舞台裏は、およそ疑ってみたほうがいい。

大塚製薬の管理のずさんさには、昨年のJCO臨海事故と似たものを感じる。また、インターネットとは直接関係ないが、電話会社が管理する顧客データが外部に漏れる事件が、たびたび新聞などで報道されている。モラルハザードといってしまえばそれまでだが、技術は信用できても、それを扱う人間が信用できないの現実は悲しい。WWWによって、簡単に個人情報を扱えるようになったのに比例して、個人情報に対する感性が恐ろしく鈍化していないだろうか。

本誌の読者のような、本格的Linuxユーザーの多くは、インターネットサーバを運営するrootである方も少なくないだろう。社会的責任といえば、ちょっと大袈裟に聞こえるかもしれないが、rootには今後ますます社会的責任が問われるようになるだろう。そして、もしLinuxやGNUの思想を深いレベルで理解するならば、そこからrootとしての倫理が展開できるように思われる。

ソースコードを公開する行為は、さまざまな意味付けが可能であるが、あえて強調したいのは、それが倫理的な行為である点だ。それはプログラミングのレベルにおける隠し立てをしない態度であり、プログラムにおいて何を実行するのかをすべて開示する行為である。それは、情報の隠蔽による権力の行使をあらかじめ放棄することを意味する。

このような倫理的態度をrootとしてのオペレーションに拡張できないだろうか。rootがオペレーションとして何をどのように実行したのかを、隠し立てせずに開示すべきである。あるいは、ユーザーがrootの操作を常に監視できる仕掛けが実現されなければならない。

また、個人情報を扱うWWWにおいては、CGIの内容を公開すべきである。もし、それを公開することがセキュリティ上の問題になるのであれば、そのような中途半端な技術を使うべきではないのだ。

Profile

とよふく つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

オープンソース
・
アクティビズム

ソフトウェアに自由を！ 政府にオープンソースを！

文：安田幸弘
Text : Yukihiro Yasuda

あぶないコード

ちょっと前の話だけれど、官公庁で使っているソフトの製造者が、ちょっとアブないカルト教団の信者だったということが判明したとかで、そのソフトを作り直すというニュースがあった。

ソフト発注の過程で、いろんな内部情報がカルト教団側に流れたとしたらマズいとは思うけれど、だからといってソフトを作り直すってのも何だかヘンだ。そのソフトがどんなソフトなのかしらないけれど、起動時に怪しげなマントラが流れるわけでもないだろうし、コードそのものが健全なら問題はないんじゃないかしらん。もっとも、その昔、この教団が売ってたパソコンのBIOSに、わけのわからないコードが入ってるという噂もあった。今回問題になったソフトにも妙なトラップが仕掛けられているかもしれないという心配があるのかもしれない。

ソフトにトラップなんて、いかにもカルト教団がやりそうなことだけど、しかしソフトにトラップをかけたがるのはカルト教団だけではないだろう。下心の有無はともかく、どこかのフラグを立てておくと、何かの情報をダンプするとか、どこかのリソースに触るとどこかにメールが飛ぶようになっているとか、そんな程度の仕掛けは珍しくもなんともない。そこらで売られている市販のソフトにだって、何が仕組まれているかわかったもんじゃない。

しかし、さらに危ないのは官公庁自身、特に警察かもしれない。何しろ昨今のていたらくである。あの盗聴法という悪法を名目に、あの警察が作った盗聴ソフトが仕様書通りに動作するなんて、誰が信じるだろう。

こうなると、信用できるコンピュータなんてなくなってしまう、という人がいるかもしれない。でもそれはソースコードが付属しないソフトを動かすことだけを考えるからだ。これまでソースコードは企業秘密に属すると考えられてきたけれど、本当にそうなのだろうか。もう一度、ソースコードは企業秘密、という「常識」を見直してみてもいいのではないだろうか。

ぼくは、少なくとも官公庁に納入するソフトは、すべてソースコードを添付することを条件にすべきだと思う。特に、官公庁で使うソフトや公共性の強い事業で使われるソフト、社会的な影響の大きいソフトでは、ただひたすら製造業者の善意を信じるだけというのは、あまりにも素朴というしかない。例のY2Kのときの騒動だって、悪意でも故意でもなかったとはいえ、ソースコードがないことが騒動を大きくした原因のひとつだった。もし、ソースコードがきちんと参照できれば、対策はずいぶん違ったものになっていただろう。

ついでに言えば、官公庁が税金で買うソフトは、社会に公開してもらいたいものだ。利用ライセンスをどうするかという問題は残るかもしれないが、少なくともソースコードの閲覧が自由にできるようにすれば、さすがの日本の警察だって、盗聴ソフトに妙な仕掛けを組み込んだりはできないだろうし、カルト信者が作ったプログラムだからといって、むやみに疑心暗鬼になる必要もなくなる。

今はまだソースコードの添付なんて、夢のまた夢みたいな気もする。だけど、数年前にソフトメーカーがソースコードを公開するなんて本気で想像していた人はどれぐらいいただろう？ いますぐは無理でも、その気になれば決して夢物語ではないと思う。

赤旗Linux

たとえば中国では「アメリカ帝国主義のOSなんて使えるか」とばかりに「赤旗Linux」とやらを開発し、かなり広範囲に利用されているらしい。サイバー戦争の研究を進めているアメリカと対峙するアジアの大国である中国が、アメリカ製のソフトウェアに頼り切るというわけにはいかないと

判断したとすれば、それは合理的な判断だ。ただ、利用するソフトにGPLで保護されたコードが含まれていれば、少なくとも理屈では彼らが開発したソフトを「これは国家秘密だ」といって秘密にすることはできなくなる。これぞ改革・開放だ。

世界に広がるオープンソースムーブメント

中国のほかにも、最近はソースコードが利用できないOSに疑問を持ったり、政府レベルでオープンソースに興味を抱き始めた国がぽつりぽつりと現れている。どんな国の政府でも、政府は政府。「マニアのもの」としか思われていなかったオープンソースに対する世間一般の認識が変わってきたことを示しているのだろう。たとえば、この6月に韓国でリーナス・トーバルズやリチャード・ストールマンを呼んで開かれるGlobal Linux 2000は、情報通信部という日本でいえば通産省と郵政省を足したような役所の主導で開かれるイベントだ。

このイベントの目的は、オープンソース・ビジネスの活性化なのだそうだが、オープンソースに疑問を持っていた人も、政府省庁のお墨付きのイベントが開かれるとなれば、韓国内でのオープンソースの位置づけは飛躍的に高まるだろう。共産主義の嫌いな韓国が、どことなく左っぽいリチャード・ストールマンを呼ぶというのが面白いところだが、どうやら彼が「コピーレフト」の主唱者だからということらしい。何事にもご先祖様を大切にする韓国では、若輩者のオープンソースという概念よりも、オープンソースのご先祖様格のコピーレフトという概念を大事にしている……というのは冗談だけど、韓国では日本よりも「コピーレフト」という言葉の知名度が高いという。

また、しばらく前にメキシコが公共教育のコンピュータOSにLinuxを採用することに決定したという話を聞いた。その後の消息は不明だが、Linux育ちの若者が大量に育成されるというのは頼もしい。Linuxならいくらでも無料でコピーでき、さまざまな開発ツールも利用できる。しかも古い386機でもそれなりに動いてくれる。考えてみれば、オープンソースはコンピュータ教育に最適なプラットフォームだろう。

そして、最新の情報分野でもオープンソースを評価する政府レベルの担当者も少なくないらしい。この3月にマレーシアで開かれたGlobal Knowledge IIという会議では、NGO側の参加者から提案された（といっても、正式な議題にはしてもらえなかったらしい）フリーソフトウェアの利用について、マレーシアの国策会社で準政府組織ともいえるMIMOSの担当者が非常に積極的な反応を示していたそうだ。

マレーシアの指導者のマハティール氏は、先のアジア通貨危機のときもIMFの処方を蹴って独自路線を貫くなど、欧米諸国の押しつけに唯々諾々とは従わない人物だという。もっとも、市民的な自由といった西洋的な人権意識も持ち合わせていないというのは困ったところだが、ともあれオープンソースを使えばコンピュータソフトの米国支配から脱却できるとすれば、マハティール氏としても悪い気はしないのではないだろうか。

●

世界のあちこちで、オープンソースに対する認識は高まっている。ソースコード添付が政府機関の納入条件になるまでには、まだいくつも難しい問題が残っているだろうが、オープンソースが信頼できる健全なソフトウェアとしての地位を確立するのは決して夢物語とばかりも言えないだろう。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ドクターShiodaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shioda

2・17 Windows 2000発売
3・2 Cray売却される
3・4 PlayStation2
3・6 AMD 1GHz CPUを発表
3・6 モトローラ、高可用性の独自Linuxを開発
3・10 Microsoft X-Box発表

原稿書きのためのマシンで、筆者は、カレンダー表示用に自作のプログラムを使っていたのだが、ふと気が付くと、そのカレンダーには2月29日がなかった。しまった、こんなところにY2Kが……。そういえば、閏年の判定を単に4年に1回という計算で作ったような記憶が、たぶん、これを作ったのは、1990年頃なので、「こんなもんでいいだろう」と思っていたのだが、まさか、10年近く同じプログラムを使うことになるとは、夢にも思わなかった……。

インテル Pentium III

インテル対AMD

この業界には、いろいろと競合する会社の組み合わせがあるが、その競合のうち最も熱いもののひとつが、このインテルとAMDの組み合わせである。386互換CPUからスタートしたAMDのx86互換CPUビジネスだが、最近は、かなりインテルを苦しめるところまで来ているし、一時は、低価格PCはほとんどAMDといわれていたこともある。

AMD Athlon

AMDが、低価格PCを席巻するのに対し、インテルは、Celeronを登場させるものの、最初はなかなか苦しかった。しかし、キャッシュをオンチップにしたCeleronが登場するや、低価格戦略が功を奏して、かなり盛り返した。しかし、AMDは逆にAthlonでインテルのメインストリームともいえるPentium IIIに挑むという戦略で、巻き返しにかかった。

独自バスであることや、当初AMDのチップセットしかないことで、導入が危ぶまれたAthlonだが、かなり健闘し、多くのメーカーが採用することになった。ひとつには、以前に公約したロードマップを前倒ししてでも、低価格PC分野で巻き返しをはかったインテルが、生産体制に狂いを生じて、昨年後半、CPUの品不足を引き起こしてしまい、多くのメーカーが迷惑を被ったとのこと。ゲートウェイなんかは、一時、「もうAMDは使わず、インテル一筋」なんてことを言っていたけれど、結局Athlonを採用することになった。

この競争だが、AMDに有利と言われている。というのは、現在のインテルのCPUは、P6という基本アーキテクチャを使ったもので、すでに高速化に限界が見え始めている。プロセスを変えるだの、命令を追加するだのして、延命策を図っているが、そろそろ速度的な限界に達しつつある。というのも、こうした基本アーキテクチャを作り替えるにはかなり時間がかかることもあり、インテルは、AMDとの競争がこれほど激しくなかった時に計画を立ててしまった関係で、すぐには新しいアーキテクチャに変更できないのである。一応、次世代のIA-32（つまり、いまのPentium IIIと命令レベルで互換性があるもの）については、Willametteと呼ばれる新しいアーキテクチャが準

備中なのだが、これは今年後半までは出てきそうにない。

昨年末のPentium III 800MHzについても、ほとんどモノがなく、かえってメーカーに悪い印象を与えたようである。こうした事情もAMDにとっては有利なんでしょうね。

さて、一応、世界で初めて1GHz CPUを発表したAMDだが、道はそんなにラクでもない。前述のWillametteが出てくるとなると、今度は、Athlonのほうが苦しくなるからだ。AMDも2次キャッシュをオンチップにするなどの計画はあるようだが、これも、結局、去年インテルが使った手で、あまり長いことは優位を保てそうにない。さて、この2社がどうなるか？　夏ぐらいにもう一波乱ありそうな気がするのだが……。

Microsoft対ソニー

ある意味、インテルとAMD以上に熱いのが、この2社の対決。こと、ゲーム機となると、一般家庭にも及ぶ分野なので、こちらのほうが世間一般には話題になる対決かもしれない。

すでに何年も遅れ続けていたWindows NT 5.0こと、Windows 2000が2月17日にようやく出荷されることになった。一応、お約束の深夜販売が行われたが、まあ、夜にしては人が多いなという感じで、Windows 95のときほどの盛り上がりはなかった。

これよりも盛り上がったのが、3月4日のPlayStation2の発売日。オンライン販売も行われ、店頭販売では、徹夜組も出てアッという間に98万台を売ったわけだ。このPS2、やはり1年も前に発表されており、一応は予定通りなのだが、発表から出荷まで長い時間がかかったのは、Windows 2000と同じ。

MicrosoftのX-BOXオフィシャルサイト。まだ情報はほとんどない（http://www.xbox.com/）

ソニーのPS2オフィシャルサイト。こちらはさすがに充実（http://www.scei.co.jp/）

ただ、ユーザーを待たせた時間は、Windows 2000のほうが長かったので、第一ラウンドはMicrosoftの勝ちか。

さらに、Microsoftは3月10日に、X-Box（仮称）と呼ばれるゲーム機を発表する。Windows 2000のカーネルを使い、インテルの600MHzクラスのCPUと、nVIDIAのグラフィックチップを使うもの。しかも、本体ハードウェアは、Microsoft自身が販売を行い、ビジネスモデルとしては、ゲーム専用機と同じく、ソフトハウスからライセンス料を取る形式になるという。

はっきりいって、これは、完全なPS2対抗製品である。いままで、ゲーム専用機については、セガのドリキャスと、任天堂のDolphin（仮称）の三つ巴だったのが、ここにMicrosoftが加わるわけだ。もっとも、X-Boxは来年秋と1年以上先のことなので、そのときにどうなっているかはわからないが……。

このX-BOXは、去年の夏あたりからウワサが出ていたのだが、結局PS2の発表を見て、慌てて発表した感じか、PS2のやり方をまったく踏襲したかどちらかであろう。ほんとうは、裁判が落ち着いてからと思っていたのではないでしょうかね？　発表直前にCPUがAMDからインテルに、グラフィックチップがギガピクセル社からnVIDIAに変更になるなど、なにかドタバタしたものが感じられるからだ。まあ、ここで発表しておいて、買い控えを起こそうという魂胆なんでしょうかね。

ただ、ゲーム機の普及ということを考えると、1年間の差はかなり大きいと思うのだがどうだろうか？　実際、セガはセガサターンをドリームキャストに切り替え、任天堂も後追いで64を出したけど、結局PSにはなかなか追いつけていない（その他の理由もあるんだろけどね）。だいたい、今ゲームがしたいのに来年の秋まで買うの待つ？　難しいと思うけどねぇ。

この戦いに比べると、インテルとSunのItaniumを巡る対立とかなーんか、せこいって感じがしちゃうのだけど……。すでに数からいえば、SolarisはLinuxに負けちゃってるし、MicrosoftもWindows 2000の後継で対応っていってるので、IA-64は、最初はLinux一色って感じになるんじゃないでしょうか。

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台
～TransmetaとLinux 2.4の関係～

日刊アスキー Linuxでは、米国Linux Todayと提携し、コラムを翻訳して掲載している。最近人気があったのが、Joe Pranevich氏執筆の「Linux 2.4の素晴らしき世界－再検討版」だ。しかし、この文書のCrusoeのくだりを見ていくと、Transmetaが開発しているCrusoeとの、意外な接点が見つかっ

モバイルLinuxが大挙登場か？

2000年春から夏にかけての最大の楽しみといえば、やはりLinuxのカーネル2.4の登場ということになるだろう。今回は、日刊アスキー Linuxでもアクセスが集中した、Joe Pranevich氏執筆の「Linux 2.4の素晴らしき世界―再検討版」（http://www.ascii.co.jp/linux/linuxtoday/article/article396128-000.html　原文タイトル：Joe Pranevich ― Wonderful World of Linux 2.4 ― “ Back on the Table Edition ”）をもとに、Linux 2.4の特徴を見やすく再整理したい。同文書は、Linux 2.4のホワイトペーパーともいえる内容であり、日本語訳で2万字近い大作だ。そこで、主な特徴を表にまとめてみた（**表1**）。

読んでみて、ふと気になることがあった。同文書の中のCrusoe関連の記述である。Transmetaはご存じのとおり、「Mobile Linux」という、Crusoe用にカスタマイズされたLinuxを利用したWebデバイス（Transmetaでは「Web Pad」と呼んでいた）をデモンストレーションしている。日刊アスキーLinuxでは、Transmetaの日本担当者・和田 信氏にMobile Linuxとはいかなるものなのか、インタビューをしたのだが、未だ発表前の段階であり、x86用のLinuxを改良したものであること以外は不明な点が多かった。インタビューを抜粋したので、コラムをお読みいただきたい。

この時点では、Mobile Linuxは、Linuxカーネルに手を加え、Crusoe用の機能を追加した“だけ”の存在と思われた。言ってみれば、カスタマイズされたLinuxカーネル（バージョンは不明だった）を元にし、さらにCrusoeの能力を引き出すためのディストリビューション（？）であって、本家のLinuxとは別のものというイメージだ。

しかし、その印象はおそらく間違いだ。「Linux 2.4の素晴らしき世界―再検討版」をもう一度よく見てみると、「Linux 2.4のリリース版にCrusoe用の各機能が入ってくることが期待される」と記述されている。つまり、Mobile LinuxとLinux 2.4は、単に「公式のカーネルとディストリビューション」もしくは「公式のカーネルをカスタマイズしたカーネル」という以上の関係があるはずなのだ。

ここで気づくのは、Mobile Linuxの発表時期が第2四半期―今年の半ばであり、Linux 2.4の発表時期も同じ時期といわれていることだ。

もしかしたら、Mobile Linuxとは、Linux 2.4そのもの（に近い存在）なのではないか？ そして、Linux 2.4が登場した直後には、Mobile Linuxを搭載したWeb Pad第1号が出荷される……。考えてみれば、Linux関連の新しいフィーチャーがカーネルに統合されるという流れは、きわめて自然な話であり、何ら驚くことではない。

そしてこのMobile Linuxは、オープンソースとなってさらに多くの開発者の手に渡る。それはすなわち、Mobile Linux＝Linux 2.4登場を契機として、メーカー製のハンドヘルドLinuxマシンが多く登場するような状況を作り出すかもしれない。Webで配付されているTransmetaのホワイトペーパーを見ると、CrusoeとMobile Linuxを搭載したハンドヘルドマシン（Mobile Gearのような、キーボード付き）が登場し、外出先では小回りが効いてWebもブラウジングできるノートとして、家やオフィスに帰ってくればドッキングステーションと合体し、オフィスツールも使える強力なデスクトップとして機能するようすが描かれている。この夏が今から楽しみである。

プロセッサ	64ビットプロセッサへの対応	Itanium（原文ではMerced）への移植準備は、AlphaやSparc64への移植作業により難しい部分の下準備はできている
	Crusoe	Quakeテストによって、確証を得ている状態。Crusoe用の各機能は、リリース版カーネルに入っていることが期待される
	i386以前のIntel製チップ	ELKS（Embedable Linux Kernel Subset）やuLinuxといったプロジェクトはあるが、現時点では対応していない
バス	ISAのプラグ＆プレイ	カーネルレベルでサポート。ISAPnP IDEコントローラからの起動などが可能になる
	I2O（Intelligent Input/Output）	カーネルレベルでサポート。I2Oとは、OSに依存しないドライバ作成を目標としたPCIのスーパーセット
外部デバイス	PCMCIA	インストールおよび設定が簡単に（外部デーモンやコンポーネントは従来どおり必要）
	USB	対応は初期段階ではあるが、キーボード、マウス、スピーカといった一般的なUSBハードウェアはカーネルレベルでサポート
	Firewire	広帯域幅デバイスでよく使われるFirewireをサポート
ブロックデバイス	IDE	IDEコントローラ数の最大値が従来の4つから10まで増えた。IDEドライバにも変更が行なわれ、PCI、PnPのIDEコントローラ、IDEのフロッピードライブ、テープ、DVDもしくはCD-ROMチェンジャーへのサポートが改良された
	rawデバイスサポート	キャッシング層を介さず、直接低レベルのデバイスにアクセスすることが可能。ミッションクリティカルな用途で使用される
	ファイルシステム	DVDで使われるUDF（Universal Disk Format）、IRIXで使われるefsがサポートされた。ReiserFSが組み込まれるかもしれない（3月25日時点では2.4リリース時点では入らないことがほぼ確定している）。LinuxからOS/2への書き込みが可能になる
	SMB（Server Message Protocol）	リモートシステム形式の自動検出による、複数バージョンのWindowsが混在したネットワークでの利便性向上
	NFS（Network File System）	NFSv3に対応
	ビデオカード	DRM（Direct Rendering Manager）のサポート
キャラクタデバイス	キーボードやマウス	USB接続のサポート。キーボードがBIOSによって初期化されないシステムや、キーボードが接続されているかどうか判断できないようなシステムにおけるキーボードのサポート。デジタイザのサポートと、デジタイザをマウスとして使えるようにするエミュレーションオプション
	シリアルポート	マルチポートのシリアルカードのサポート数を増やした
	パラレル	パラレルポートのサブシステムの改良。“汎用の（generic）”パラレルデバイスのサポート。これにより、対象ポートのPnP情報を探ったり、UDMAの利用を含んだパラレルポートの拡張モードすべてにアクセス可能になる。また、すべてのコンソールメッセージをプリンタなどのパラレルポートデバイスに出力できる
	ユーザー補助	音声合成カードサポート。視覚障害のあるユーザーは、起動プロセスの初期のメッセージを含めて音声で聞くことができる
	マルチメディア	サウンドやビデオカードへの対応。サウンドの全二重通信の対応
ネットワーク／プロトコル	ネットワークソケット	従来はネットワークソケットからのイベントを待っているプロセスが多い場合、ソケットからのイベントにすべてのプロセスが反応してしまい、オーバーヘッドがかかっていたのを改良。イベントが発生した場合、関係のあるプロセスしかアクティブ化しないようになった
	ネットワーク層	スケーラビリティの向上。Windowsを含む、一般的なOSのネットワーク処理用スタックの癖に対応し最適化した。DECNet、ARCNetのプロトコル／ハードウェア用コードを追加し、これらのシステムとの相互運用性を高めた
	PPP	主要なコードの書き直しとモジュール化。ISDN層やシリアルデバイスPPP層からの、PPP層への結合
実行形式	プログラムローダ	「ヘルパ」アプリケーションと実行形式を結びつけるモジュール。WINEやDosemuが、WindowsアプリケーションやDOSアプリケーションを、実行させる命令を、カーネルから“ネイティブ”に発行できる。これにより、たとえばマシン上のすべてのWindowsアプリケーションをWINEと結びつけ、「./notepad」と入力したときに適切に処理される
	カーネルWebデーモン「khttpd」	カーネルが、Apacheなどのユーザー空間のサーバと通信せずにhttp要求を処理できる機能

表1 Linuxカーネル2.4の特徴

※「Linux 2.4の素晴らしき世界―再検討版」より抜粋

Column

インタビュー
Transmeta・和田 信氏

日刊アスキー：Mobile LinuxについてTransmetaのWebページには「Web Padに使われる」「オープンソースである」ということしか出ていませんでしたが、具体的にどういった存在のものなのでしょう？

和田：Mobile Linuxは、Transmetaの社内で開発しているものです。基本的にx86のLinuxコードのものです。具体的にいえば、カーネルに加えて、パワーマネージメントのような、Crusoeの長所を生かす部分をあわせて開発しています。Linuxベースの携帯端末で、Windowsでは電池が8時間持って、Linuxでは12時間持つ、そういうものができたら面白いなと。

Transmetaがお見せしたWeb Padの端末には、Mobile Linuxのプロトタイプが載っています。端末にはハードディスクがなくて、ROMベースで、LinuxとX Window SystemとNetscape Navigator、バーチャルキーボードが入っている、というものでした。

日刊アスキー：組み込みのLinuxというと、カーネルから必要ない機能を削ってサイズを小さくするイメージがありますが、パワーマネージメントということは、カーネルに手を加えているということですか？

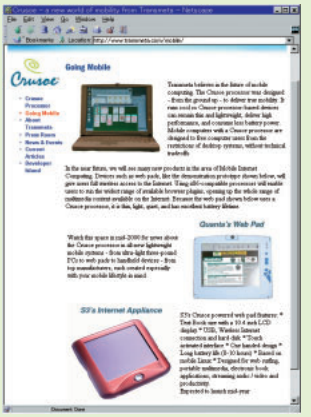
和田：詳細は今のところあまりお話しできない状態なのです。カーネルのバージョンは、すごく新しい番号がついていたとは記憶しているのですが、私のほうからは何とも言えないんです。

日刊アスキー：新しいというのは、2.2系で新しいということですか？

和田：そうですね。完成の予定は第2四半期と言われています。オープンソースで出してしまおうと考えています。

日刊アスキー：いままでのお話ですと、「パワーマネージメント＝Crusoe用」という印象を受けるのですが、x86のコードの中でパワーマネージメントをやるとすると、どうやるのでしょうか？ Crusoeの特別な機能に対する、たとえばCrusoeネイティブのコードが、Mobile Linuxに含まれているのでしょうか？

和田：いまのところはx86としか言ってませんね。



こんなLinuxマシンが続々登場するか（http://www.transmeta.com/mobile/）

正しきタコへの道

初級Linuxer養成講座

第8回　ひとり管理者の心得（3）～ファイルのUIDとGID

MS-DOSやWindows 98などの個人用のOSにはないが、LinuxやWindows NTなどのマルチユーザーを対象としたOSに存在するのが「ファイルの所有権」という概念だ。Linuxでは、ファイルの所有権を「UID」や「GID」という数値で管理している。ふだんはまったく無視できるものだが、ほかのマシンとのファイルのやり取りの場面では、その性質を知っていないと混乱させられることがある。

文：竹田善太郎
Text : Zentaro Takeda

今年、コンピュータ関連業界（最近は「IT業界」といったほうが格好いいらしい）では、年末年始の2000年問題からこちら、**話題の新製品**の相次ぐ登場などもあって、公私を問わず忙しい日々を過ごした関係者も多かったことだろう。中でも、Windows 2000と**某ゲーム専用機**の発売は、まったく性質の異なるものであるにもかかわらず、一般マスコミでも大きく取り上げられていた。Windows 2000はともかく、某ゲーム専用機の発売にいたっては、販売用Webサイトにおけるトラブルや、発売日に小売店の前にできた行列などが仰々しく報道されたものだ。

筆者も、パソコン業界の隅っこで生活している以上、**コンピュータゲームも必須科目である**という名目をつけて家内を納得させ、件のゲーム専用機をWebサイト経由で予約し、（発売日には間に合わなかったものの）比較的早期に入手することができた。

ところが、届いた品物が**不良品**でまともに動かない。すでにインターネットの関連サイト上で指摘されていたようなさまざまなトラブルを、**すべて一身に抱え込んでいる**というとんでもない代物だったので、返品交換をお願いしようと購入先のサポートセンターに問い合わせた。

取扱説明書にあるような注意事項はすべて**厳守**して、トラブルの原因となるような取り扱い上のミスがないことを確認したうえ、症状を詳しく述べて製品の不良であるということを**きちんと伝えたつもり**だった。ところが、サポートから返ってきた返事はそっけないもので、「電源が入らない」とか「外観上の破損個所がある」などの場合以外は対応できないから、製造メーカーに問い合わせてくれという。伝えたはずの不具合についても「取扱説明書をよく読め」といったような返答しかなく、まったく困惑してしまった。

おそらく、発売直後でサポート宛ての問い合わせが膨大な数になっていて、ていねいな対応ができない状態だったのだろう。筆者も以前、ユーザーサポート業務を担当した経験があるので、担当者のご苦労は想像できるのだが、訴えを**右から左へ聞き流す**ような対応をされてしまったような気がして、どうもすっきりしない。とはいえ、ここで相手に激しくかみついて、昨年話題になった某家電メーカーにまつわる**暴言事件**の二の舞になるのはごめんなので、ちょっとだけ苦言を記したメールを送る程度にとどめて、指示されたとおりに製造メーカーのサポートに問い合わせた。結局、初期不良交換として対応してもらうことができたのだが、この原稿を書いている時点で、まだ代替品は届いていない。仕事や生活に必需のものではないので、遅れたところで一向に構わないのだが、ちょっと不安である。

奇妙なユーザー名

さて、筆者が以前やったことのあるサポート業務の中で、UNIX関連のソフトウェアに関するものがあった。プログラムに対するパッチファイルや追加のコマンドファイルなどを**tarファイル**の形式で1つのファイルにまとめて、ユーザーに送るといったものだったのだが、ユーザーからの問い合わせの中に、「もらったファイルを展開すると、**へんなユーザー名**が表示される。ファイルが壊れているのでは？」というものがあった。よく話を聞くと、「ユーザー名とグループ名が意味不明の数字になっている」というのだ。つま

り、展開したファイルをlsコマンドなどで見てみると、ユーザー名の部分が数字になってしまっているということだ（**画面1**）。

UNIXに慣れた人なら、**なんだ、そんなことか**とばかにするような問題なのだが、初心者が不安に思うのはもっともだし、「ファイルのユーザーIDの設定は、セキュリティ上重要である」なんて文言を聞いたことがあればなおさらだろう。結論からいえば、個人がスタンドアロン環境で使っているLinuxマシンにおいては、ファイルのユーザーIDについて気にする必要は、それほどない。しかし、一部のプログラムを動かす場合には、ファイルのユーザーIDを適切に設定しないといけない場合もあるので、ユーザーIDについての知識とその操作の方法について、最低限の知識はもっていないといけないだろう。

いずれにせよ、ファイルのユーザーIDやアクセス権の設定がどうなっているかを気にするのは、悪いことではない。たとえ1人でしか使っていないマシンであっても、将来、Linuxマシンを**ホームサーバ**にするときには、家族全員でサーバマシンを共用することになるだろうし、そのようなときは自分の**秘密のファイル**のアクセス権設定を、きちんとできるかできないかは重要な問題になる（とはいえ、rootユーザーの権限を持つ「お父さん」あるいは「お母さん」なら、**子供の日記ファイル**をこっそり盗み読むこともたやすいのだが、そのへんをどうするかはご家族同士の信頼関係しだいだろう）。

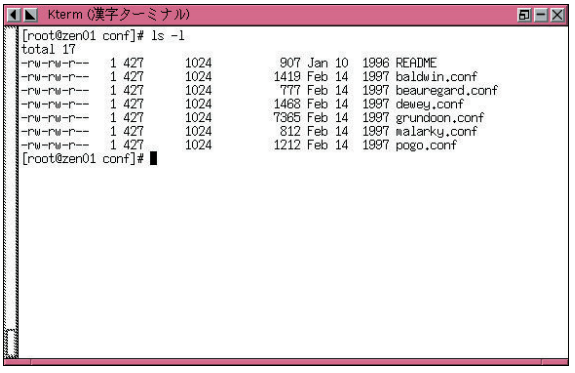
UIDとGID

最近になってLinuxを使い始めた初心者ユーザーにとって、UID（ユーザーID）やGID（グループID）という言葉はなじみが薄いかもしれない。Linuxのディストリビューションをインストールするときにも、昔ならUIDやGIDを「自分で」設定する必要があったのだが、現在のディストリビューションの多くでは、ユーザー名を入力するだけで**インストーラが勝手に**UIDやGIDの設定をしてくれるので、知らなくても困ることはないからだ。

ユーザーIDとは、Linuxの利用者（ユーザー）1人ごとに割り当てられる識別番号のことで、0から60000までの間の整数のどれかが割り当てられることになっている（この数値は、システムの設定によって異なるが、Red Hat系のLinuxではこのように設定されている）。このうち、0から99までのUIDは、rootユーザー（UIDは0）やその他のシステム管理上必要となる特別なユーザーに割り当てられるのが慣例となっている（これらの多くは**実体のないユーザー**で、そのアカウントでログインすることはできなかったり、その必要がなかったりするものばかりだ）。

ところで、「ユーザーID」という言葉を聞くと、Linuxマシンにログインするときに入力する英数字からなる「ユーザー名」のことではないか、と考える人もいるかもしれない。一般的なコンピュータ用語で「ユーザーID」と言ったときは、このような「ユーザー名」のことを指す場合が多いので混乱しやすいのだが、Linux（UNIX）の世界で「ユーザーID」と言った場合、ユーザー識別用の数値のことを指し、「ユーザー名」（user name）と言った場合、英数字から構成されていて**人間にも意味のわかる**ログイン用IDのことを指す（**図1**）。この違いはぜひ覚えておいてほしい。

Linuxにログインするときには、数字のUIDではなくログイン名を入力するし、さまざまなコマンドでも数字のUIDを直接扱う場面はほとんどない。



画面1　奇妙なユーザー名
よそからもらってきたtarファイルを展開してls -lコマンドで表示すると、ユーザー名の部分がこのような数字になってしまうことがある。

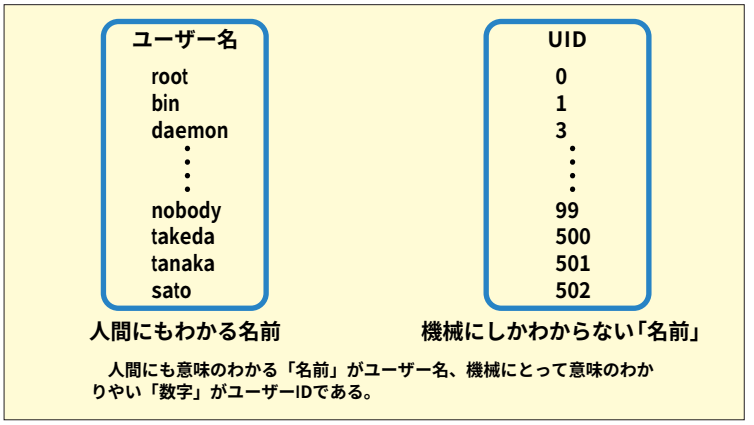


図1　「ユーザー名」と「ユーザーID」の関係

しかし、Linuxの内部では、すべてのファイルやプロセス（プログラム）の管理にはログイン名ではなくUIDを使っている。極端に言えば、Linuxというコンピュータシステムから見ると、人間にとって意味のある「ユーザー名」は、逆に何の意味も持たないのだ。このため、ユーザーが入力したユーザー名のデータは、かならずUIDに変換されてから内部で処理されることになっている。

Linuxマシンに新たにユーザーを追加する場合、本来ならユーザー名と一緒にUIDも設定する必要があるのだが、多くのディストリビューションに付属しているuseradd（あるいはadduser）コマンドでは、UIDを指定しなくても、適当に未使用のUIDを割り当ててくれるようになっている。また、前述のようにLinuxのインストール時にユーザーアカウントを作成する場合にも、UIDの入力は必要ない。このため、UIDの存在を一生知らずに過ごしてしまうLinuxユーザーがいても不思議ではない。

しかし、前回説明したtarコマンドを使ってファイルのバックアップを行ったり、他人との間でtarファイルの受け渡しをするときに、前記のようにUIDの存在がちらっと顔を出すことがある。そのときになって慌てないように、そして、将来LinuxをLANやインターネットなどのネットワーク環境で使いたいと思っているのなら、そのときに困らないように、UIDやGIDとはどんなものなのか、少しでも知っておいたほうがよいだろう。

ファイルの所有者とパーミッション

以前にもこの連載で説明したと思うが、Linuxはその手本となったUNIXと同様に、マルチユーザー環境のOSとして作成されている。つまり、1台のマシンを、複数の人間が同時にログインして使えるようになっているわけだが、そのような使い方をする場合に、あるユーザーが別のユーザーのファイルを勝手に書き換えたり消去してしまったりできるようだと、いろいろと問題が生ずる。このため、Linuxではすべてのファイルごとに、それがどのユーザーの管理下にあるものなのかを示す所有権の情報として、UIDが割り当てられている。いや、実際にはファイルだけでなく、メモリ領域、プロセス、デバイスなど、OSが管理しているありとあらゆるものにUIDが割り当てられているのだ。いわば、すべての持ち物について、所有者の名前シールが貼り付けられているようなものと思えばよい。

ファイルの所有者のUIDは、そのファイルを新たに作成したとき、たとえばmuleなどでテキストファイルを保存したり、コマンドの出力をファイルにリダイレクトしたときなどに自動的につけられる。当然、そのときにつけられるUIDは、その時点でログインしているユーザーのUIDである。

「名前シール」をつけるのに、人間にもわかりやすい「ユーザー名」を使わずに、一見すると意味不明なUIDを使っているのは、コンピュータにとってはそのほうが都合がよいからにすぎない。たとえば、8文字のユーザー名で名札をつければ、8バイトの記憶領域が必要だが、60000以下の整数を名札に使うのなら、たかだか2バイトの領域で済む。

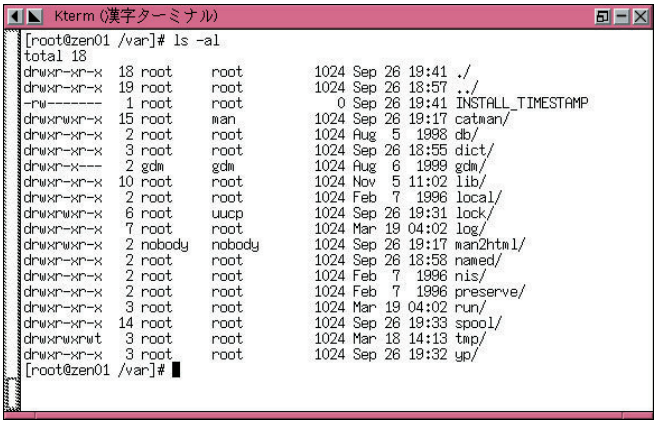
では、ユーザー名とGIDの間の関係はどこで定義されているかというと、/etc/passwdというファイルに、各ユーザーに関するさまざまな情報と一緒に記録されている（**リスト1**）。昔のUNIXでは、このpasswdファイルをテキストエディタなどで編集して、ユーザーの追加やユーザー情報の変更を行っていたので、UIDの割り当てなども人間（管理者）が行う必要があったのだが、現在ではuseraddなどの便利なコマンドがあるので、その必要はほとんどない。

ただし、複数のLinuxマシンをLAN接続して、お互いのディスクをネットワーク経由で共有しようとする場合などは、それぞれのマシン上のユーザー名とUIDを同じにしておく必要がある。そのような場合は、管理者が自分でUIDの指定をする必要がある。あるいは、「NIS」という機構を使って、ネットワーク上のすべてのLinuxマシンの

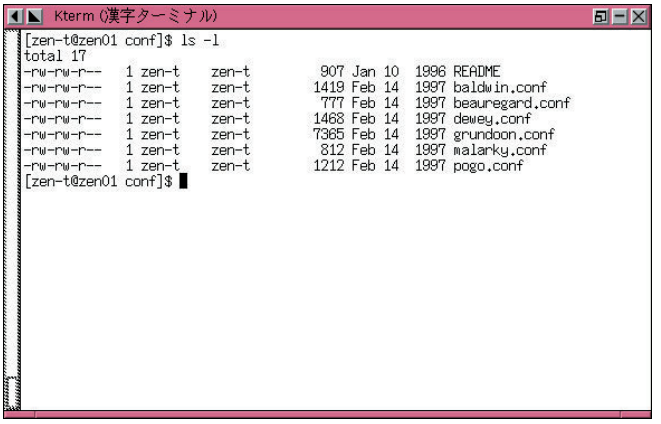
▼リスト1　/etc/passwdファイルの例

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
   （中略）
takeda:x:500:500:TurboLinux User:/home/takeda:/bin/bash
tanaka:x:501:501:TurboLinux User,,,:/home/tanaka:/bin/bash
sato:x:502:502::/home/sato:/bin/bash
```

現在では、このファイルを管理者が直接いじることはほとんどないが、Linuxマシンに登録されているすべてのユーザーに関する情報が記述されている。



画面2　ユーザー名とグループ名
ls -lコマンドでユーザー名の次に表示されるのがグループ名である。



画面3　一般ユーザーの状態でtarファイルを展開
画面1と同じファイルを、一般ユーザー（ここでは「zen-t」）の状態で展開した結果。ファイルの所有者はすべて「zen-t」になっている。

ユーザー情報を一元的に管理することもできる。ただ、これらについてはもっと専門的な話になってしまうので、ここでは触れない。

グループIDは必要か？

ところで、Linuxでは各ユーザーのログインアカウントには、UIDのほかに**グループID（GID）**というものも割り当てられている。「ls -l」コマンドなどでファイルを表示させると、ファイルの所有者のユーザー名の隣に、もうひとつユーザー名のような文字列が表示されるのに気がついただろうか（**画面2**）。これが**グループ名**（group name）で、ユーザー名の場合と同じように、このグループ名に対しても整数の値であるGIDが割り当てられているのだ。

GIDは、複数のユーザーをまとめたユーザーのグループを定義して、そのグループに属するユーザーの間でだけ、ファイルなどを共有できるようにするために存在する。たとえば、企業などでLinuxを使う場合は、各部署ごとに1つずつGIDを定義して、その部署に所属するユーザーのGIDをそれに統一する、といった使い方ができる。ほかの部署の人間には見られたくないが、部署内の人間とは共有したいようなファイルについて、GIDが同じユーザーだけ読み出しできるが、GIDが異なるユーザーは読み出しできない、というような設定ができるのだ。グループ名とGIDの定義は、前出の/etc/passwdとよく似た/etc/groupというファイルで行われている。

個人的にLinuxを使う場合、GIDとグループ名が必要になるような場面は、UIDよりさらに少ないだろう。また、TurboLinuxなどのディストリビューションを使っている場合は、ユーザーを追加するときに、そのユーザーのUIDと同じ値のGIDが自動的に割り当てられる。つまり、1つのGIDに属するユーザーは1人だけ、という状態になる。このような使い方をしているLinuxでは、GIDは**ほとんど存在意義がない**と言ってしまってよいと思う。

したがって、Linuxを使い始めたばかりのユーザーは、とりあえずGIDの存在についてはあまり気にしなくてもよいだろう。ただし、特別なアプリケーションをインストールする場合には、ファイルのGIDを適切に設定しなければならないこともある。もっとも、このような場合はインストールのドキュメントに手順が記されていることが普通だし、気の利いたプログラムなら、インストール時にある程度の設定は自動的に行ってくれるはずである。

tarファイルの中身の所有者は？

ところで、やっと今回の本題に入るのだが、ネットワーク（インターネット）経由で別のマシンからファイルをコピーしたtarファイルを、自分のLinuxマシンで展開（解凍）した場合、展開後のファイルのUIDはどのようになるのだろうか？

まず、tarファイルには、そのtarファイルを作成した元のファイルの内容と同時に、そのファイルの**UID**（およびGID）の値が記録されている。**ユーザー名**がそのまま記録されているのでは**ない**ことに注意する。

rootユーザー以外の**一般ユーザーとしてログイン**している場合は、tarファイルの内容にかかわらず、展開したファイルはすべてログインしているユーザーの所有権になる。だから、あまり問題にはならない（**画面3**）。しかし、**rootユーザーとしてログイン**している状態でtarファイルの

展開を行うと、展開されたファイルのUIDは、大元のファイルと同じ値になる。**UIDの値がそっくりそのまま**復元されるのである。

ところで、ユーザー名とUIDの関係は、1台のLinuxマシンの中でしか意味を持たない。たとえば、あるLinuxマシンの中で、「takeda」というユーザーのUIDが123番だったとしても、別のマシンではUIDが123番のユーザーは「jacob」だったりするのだ。

あるいは、123番のUIDにはどのユーザーも割り当てられていない場合もある。このような場合、rootユーザーの状態でtarファイルを展開すると、展開されたファイルのUIDは**まったく意味を持たない**ことになる。このようなファイルをlsコマンドで表示すると、そのUIDに該当するユーザー名を見つけることができないために、前述の**画面1**のように、ユーザー名の部分にUIDの数字が表示されることになるのだ。

同一のマシン上であっても、たとえば、OSの再インストールを行ったり、ユーザーアカウントの追加や削除を行うと、UIDの割り当て状況が以前とは変わってしまうこともある。よくあるのが、ホームディレクトリの内容をtarファイルとしてバックアップしておいて、OSの再インストール後に復元したのに、そのユーザーとしてログインしたり、ホームディレクトリ中のファイルが読み書きできなくなる、というトラブルだ。

これは、まさにtarファイルへのバックアップ前と、復元時でUIDの割り当てが変わっていて、復元後のホームディレクトリ中のファイルの所有者が、**まったく別のユーザー**や**存在しないユーザー**のものになっているからだ。

tarファイルで配布されているようなソフトウェアをインストールする場合なども、手順の最後にプログラムファイルを適当なディレクトリにコピーする作業以外は、**一般ユーザーの状態で作業せよ**と指示されることが多いが、このような**問題**が起こるのを避けるという意味もある。

とはいえ、個人で使っているLinuxマシンでは、いちいちsuコマンドなどでrootユーザーと一般ユーザーの状態をいったりきたりするのは面倒なので、rootユーザーのままで作業をする人も少なくないだろう。このような使い方をいちいちあげつらうのは**余計なおせっかい**でしかないし、個人用のマシンならば、それほど実害があるわけでもない。

でも、マシン上に存在しないUIDをもつファイルは、一般ユーザーのアカウントからは読み書きできなかったり、コマンドファイルなどの場合は実行できなかったり、実行時に思わぬ誤動作をする可能性もある。rootユーザーの状態でよそからもらってきたtarファイルを展開して、ファイルの所有権がおかしくなっているのに気づいたら、次に述べるchownコマンドを使って、ファイルの所有権を適切なUIDに設定し直せばよい。

chownコマンド

「chown」とは「Change Ownership」を略したもので、その名のとおり、ファイルなどの所有権（ownership）を変更するためのコマンドだ。詳しい使い方はマニュアルなどを読んでもらうことにして、ここではファイルの所有権を変更する方法だけ紹介しておこう。

1つのファイルの所有権を変更するには、rootユーザーの状態で、次のようにchownコマンドを使う。

```
# chown username filename
```

たとえば、「strayfile」という名前のファイルの所有権を、ユーザー「takeda」の所有権に変更したければ、次のようにchownコマンドを使う。

```
# chown takeda strayfile
```

あるディレクトリ以下のすべてのファイルの所有権を変更したければ、「-R」オプションを使う。

```
# chown -R username directory
```

たとえば、ディレクトリ「straydir」以下のすべてのファイル（サブディレクトリ中のファイルも含む）の所有権をユーザー「sato」に変更したければ、次のようにすればよい。

```
# chown -R sato straydir
```

chownコマンドでは、ファイルのUIDだけでなくGIDも同時に変更できる。strayfileのUIDを「takeda」に、GIDを「users01」にしたければ、

```
# chown takeda:users01 strayfile
```

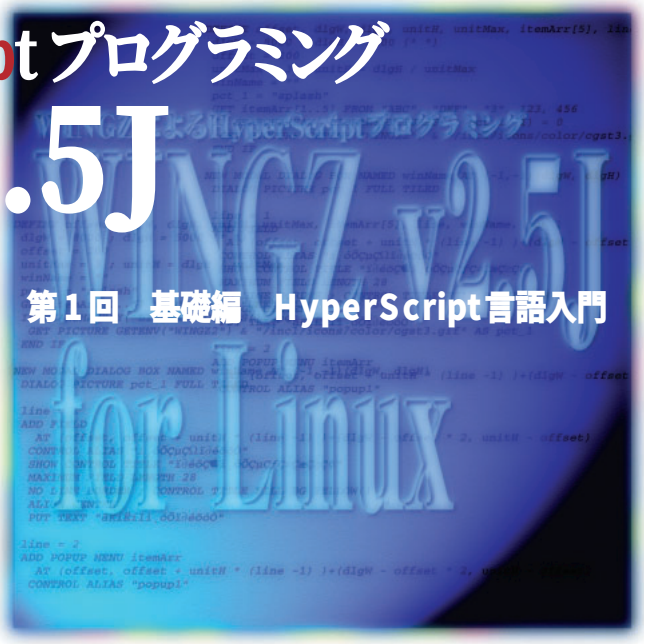
のように、ユーザー名の後ろに「:」（コロン）をつけ、続けてグループ名を指定すればよい。

さて、Linuxのファイルにまつわるトラブルでは、今回解説したUIDやGIDのほかに、最初にちょっと触れた**アクセス権**にまつわるものが多い。次回は、今回の続きとして、ファイルのアクセス権について説明してみることにしよう。

新連載

WINGZによるHyperScriptプログラミング

WINGZ v2.5J for Linux



WINGZは、スプレッドシートをベースとした簡易GUI構築を行えるアプリケーションで、国内販売開始から10年になります。Macintosh、Windows、OS/2、NEXTSTEP、Motif、OPENLOOKといった各種プラットフォーム展開を行ってきたWINGZですがLinux版についても開発が進んでいます。販売にさきがけ、これから3回にわたり、GUI構築、イベント処理、RDBとの通信処理など、WINGZのHyperScriptを使ったプログラミングを紹介していきます。WINGZ／HyperScriptを使った簡易GUIプログラミングの世界をご体験ください。

文：株式会社アイフォー　久米 繁之
Text：i4 CORPORATION　Shigeyuki Kume

はじめに

WINGZはスプレッドシート（表計算）機能をベースとしたソフトウェアで、オリジナルは米国Informix Software,Inc.が開発、国内では1989年から販売を開始し、その後ソースコードを取得して現在は株式会社アイフォーで独自に開発、販売を行っています。昨年秋で国内での販売開始10周年になりました。

WINGZは、搭載しているHyperScript言語を使ってGUI（グラフィックユーザーインターフェイス）ベースのアプリケーションを作成することができ、またアドインのDataLink機能を使えばRDBとの接続や問い合せ処理を行うことも可能なため、社内業務システムで使用するデータベースのフロントエンドなどとして多く利用されています。

■WINGZ v2.5J for Linux

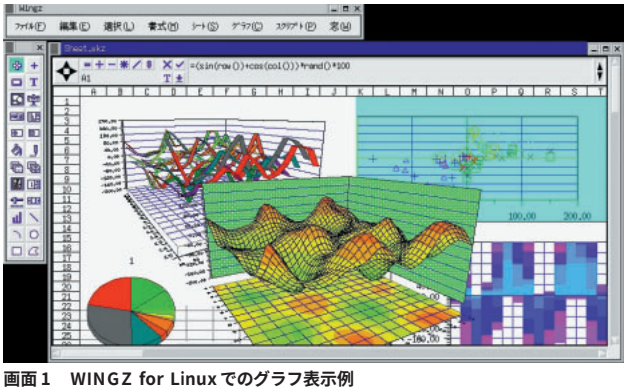
これまでにWINGZは、Macintosh、Windows、OS/2、UNIX（NEXTSTEP、Motif、OPENLOOK）などのプラットフォームへ対応してきました。Linux版についても開発を進めており現在β版まで完成しています（**画面1**）。

現在急速に市場が広がりつつあるLinuxプラットフォームですが、サーバ用途以外のデスクトップ・クライアント用途のアプリケーションの少なさは否めないのではないでしょうか。

一方、Xの各種ツールキットを使ってGUIアプリケーションを組むのは結構面倒で簡単にはダイアログボックスやコントロールを使った処理は作れない、メンテナンスが大変、という現状もあることと思います。

WINGZのHyperScript言語を使うことで、GUIベースのアプリケーションの作成やリレーショナルデータベースのフロントエンド作成が、より簡単になります。たとえばWINGZのダイアログボックスビルダー（**画面2**）を使えば、ツールパレットからコントロールを雛型に置くだけで、ダイアログボックスとコントロールを生成するスクリプトを作成することができます。

ダイアログボックスやワークシート上で各種コントロール（**画面3**）を使った処理、イベント処理、RDBとの接続処理程度であれば、HyperScriptを使った処理でも十分やっていけることがこの連載でわかっていただけると思います。



画面1　WINGZ for Linuxでのグラフ表示例

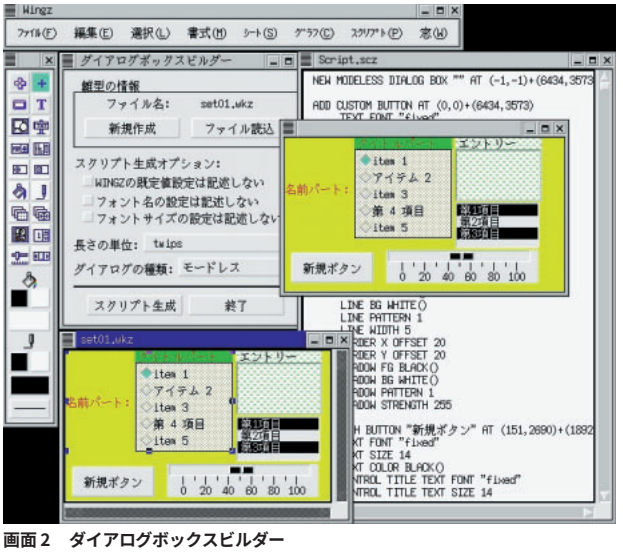
■HyperScriptについて

HyperScriptはBASICに似た言語形態のインタープリタです。実行環境としてWINGZ自身が必要となりますが、インタープリタであるため試作～実行のサイクルが短くてすむなど、一般のツールキットと比べて比較的気軽にプログラミングすることができます。実際には、WINGZ自身もHyperScriptで書かれたひとつのアプリケーションにすぎません。

HyperScriptの処理においては、スクリプトを直接実行するだけでなく、対象（ワークシート、ダイアログボックス、コントロールなど）にスクリプトを添付（アペンド）することで、たとえばワークシートがアクティブになった際にスクリプトを実行する、プッシュボタンが押されたときに実行する、コントロール上でマウスをある領域にドラッグした場合に実行する、などといったイベント依存型の処理も行えます。

HyperScriptで扱う変数には基本的に型の制限がありません。255バイトまで格納できる変数、3次元まで定義できる配列変数を、グローバル変数やユーザー関数内のローカル変数として使い分けることができます。またこの変数に加えて、ワークシートのセル座標や範囲、セル範囲名を使ったプログラムも可能です。また、業務アプリケーションの作成を前提としていますので、一般のGUI作成やイベントのハンドリングはもとより、入力オペレータの誤操作を防ぐうえで必要になる、コントロールやウィンドウの使用制限属性の設定も行えます。

利用可能なファイル形式としては、WINGZ専用のデータ形式のWKZ（ワークシート）、SCZ（スクリプト）以外にも、テキスト（EUC/SJIS）、ピクチャーデータ（JPEGなど）、一般的なスプレッドシート形式（SLK、WJ2など）があり、それぞれのデータ形式を指定したファイル制御（読み込みや保存）もHyperScriptから行えます。



画面2　ダイアログボックスビルダー

WINGZ v2.5J for Linux 体験版について

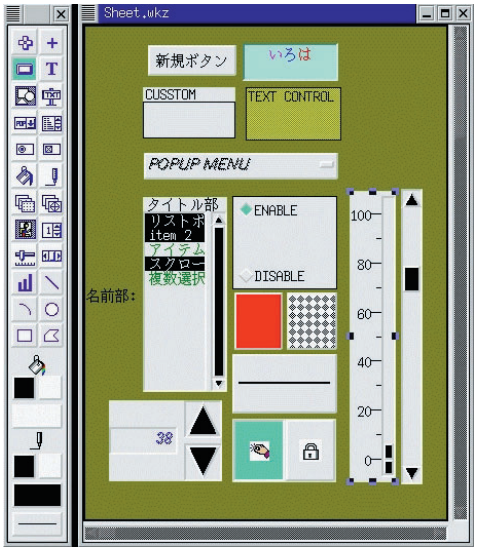
本号の付録CD-ROMにWINGZ v2.5J for Linuxの体験版と本文中で使用するサンプルスクリプトを収録しています。体験版は、インストールから3か月間の使用が可能です。ただし、最終使用期限は西暦2000年11月末日までとさせていただきます。動作環境、インストール方法の詳細については、付録CD-ROMのreadme_wz25demo.eucをお読みください。

WINGZ v2.5J for Linuxのマニュアルは、WINGZのインストール先のmanual/ディレクトリに展開されます。オンラインマニュアルはPDF形式ファイルです。必要に応じてPDFファイルリーダーをインストールのうえ、ご覧ください。

基礎演習 HyperScript関数、HyperScriptコマンド

では、これからHyperScriptを使った演習に入っていきます。

例題ごとにスクリプトとその解説を行っています。例題は個々に完結していますので、興味のある部分から読み進んでください。一部、長いスクリプトについては本文中に掲載していないものがあります。この場合、ファイル名をもとに付録CD-ROMに収録のスクリプトファイルを参照してください。



画面2　WINGZで使用できるコントロールの一部

スクリプトの実行にあたっては、まず新規のスクリプト・ウィンドウを用意します。WINGZの［ファイル］メニューから［新規］メニューアイテムを選ぶと、サブメニューが表示され［ワークシート］と［スクリプト］の2つのメニューアイテムが並んでいます（**画面4**）。ここで［スクリプト］を選ぶと、新しいスクリプトが表示されます。

スクリプトを実行するには、次のいずれかの操作を行います。

●スクリプトパレットのRunアイコンを押す
●Ctrlキーと.(ドット)キーを同時に押す
●Ctrlキーと2キーを同時に押す

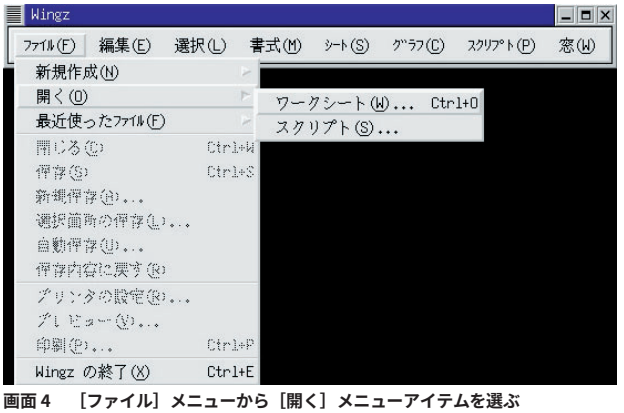
テキストファイルをスクリプトとして開くこともできます。他のエディタで作成したテキストファイルをスクリプトとして開く場合は、WINGZの［ファイル］メニューから［開く］―［スクリプト］を指定して表示される、オープンダイアログボックスでテキストファイルを選択、指定してください。

ワークシートのセルにスクリプトを記述してそのセルを選択しCtrl＋．を押すことで、短いスクリプトを単独で実行することもできます。

■イントロダクション ―メッセージボックスの表示―

新規のスクリプト・ウィンドウを作成して、例題に記述したスクリプトを打ち込み、実行させてみてください。ここでは、HyperScriptで扱える各種ウィンドウを作成する簡単な例を紹介します。

```
例01 - sample01.scz -
MESSAGE "Hello world"
```



画面4　［ファイル］メニューから［開く］メニューアイテムを選ぶ

HyperScriptのスクリプティングの第一歩として、メッセージダイアログを表示させてみました。HyperScriptを使えば、このように1行でメッセージダイアログを表示できます。この場合、ウィンドウの表示位置は画面中央、ウィンドウサイズは表示されるメッセージ文字列に従って自動調整されます。この例では、「MESSAGE」の部分がHyperScriptコマンド、""（ダブルクォート）で括った文字列の部分がコマンドの引数になります。HyperScriptではこのように、文字列をダブルクォートで括って扱います。文字列を連結する場合は、&（アンパサント）を使います。HyperScriptのコマンド、関数、変数の記述では、半角アルファベットの大文字・小文字の違いはありません。

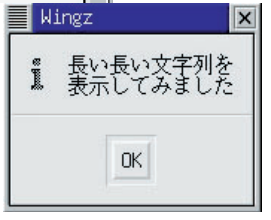
■文字列の加算と文字列関数の使用

```
例02 - sample02.scz -
MESSAGE REPEAT( "長い", 2 ) & "文字列を" & CHAR(13)
& "表示してみました"
TITLE ADATE( NOW(), "元号g年m月d日")
```

MESSAGEコマンドにはTITLEオプションがあります。このオプションを使って、タイトルバーに指定した文字列を表示できます。指定しなければWINGZのデフォルトの文字列が入ります。また、ここで使用しているREPEAT( )関数は、文字列を扱うHyperScript関数のひとつで、引数に文字列と繰り返す回数を指定します。CHAR( )関数は、引数で指定したコードに相当する文字列を戻します。例02ではCHAR( )関数の少し変わった使い方をしてます。CHAR(13)は改行コードです。

HyperScriptにはじめから用意されているHyperScript関数には、用途によって文字列関数、数式関数、HyperScript制御用の関数などがあります。目的に合ったHyperScript関数が存在しない場合は、既存の関数や制御

画面5　例01のスクリプトを開いたところ



画面6　例02の実行で表示されるメッセージウィンドウ

処理を組み合わせて、独自のユーザー関数を作って使用することもできます。サンプルで使用しているコマンドと関数の詳細については、オンラインマニュアルの関数/コマンドリファレンスを参照してください。

■モードレスダイアログボックスの作成

HyperScriptで扱えるダイアログボックスは、モーダルダイアログボックスとモードレスダイアログボックスの2つです。モードレスダイアログボックスは、同時に複数のダイアログボックスを表示したまま、使用するダイアログボックスに制御を切り替えて、処理を進めることができます。モーダルダイアログボックスは、いったんダイアログボックス内の処理を終えてダイアログボックスを閉じないと、呼び出し元のウィンドウに制御を戻すことができません。ユーザーが必要項目を入力し終えないと次の処理に進めない、といった処理に使用します。ここではまずモードレスダイアログボックスを生成してみましょう。

```
例03 - sample03.scz -
NEW MODELESS DIALOG BOX "サンプル" AT (-1,-1)
(5000,3000)
USE DIALOG BOX
```

これはまったく素のダイアログボックスを作成した例です。「サンプル」という名称のモードレスダイアログボックス（**画面7**）が画面中央に表示されます。

◇ウィンドウのロケーションとサイズの指定方法

HyperScriptにおけるウィンドウの座標指定は、用途によって絶対指定と相対指定を選べます。

●絶対指定：AT (2000,2000)(3000,4000)
ウィンドウの矩形の起点(左上位置)と終点(右下位置)を指定
●相対指定：AT (0,0) + (3000,2000)
ウィンドウの矩形の起点とウィンドウサイズを指定

画面7　例03の実行で表示されるモードレスダイアログボックス

開始位置に(-1,-1) を指定するとウィンドウの中心を画面中央に合わせて表示します。

◇長さの指定方法

HyperScriptにおける長さの単位はtwipsです（1twipsは1/20ポイントで、1/1440インチに相当します）。長さを表すHyperScript演算子（inches、millimeters、etc.）を使うこともできますが、1cmは約567twipsに相当しますので「3 ＊ 567」といったように定数や変数で直接指定することもできます。たとえば縦10cm、横15cmのウィンドウであれば、「AT (0,0)+(10 ＊ 567, 15 ＊ 567)」と指定します。

■コントロールの入力情報を取得する

```
例04 - sample04.scz -
DEFINE offset, dlgW, dlgH, unitH, unitMax, line
    dlgW = 8000 ; dlgH = 5000 ; offset = 200
    unitMax = 2 ; unitH = dlgH / unitMax

NEW MODAL DIALOG BOX NAMED "コントロールの値を取得する
例" AT (-1,-1)(dlgW, dlgH)

line = 1
ADD FIELD
    AT (offset, offset + unitH * (line -1) ) +
    (dlgW - offset * 2, unitH - offset)
    CONTROL ALIAS "入力した文字列"

line = 2
ADD PUSH BUTTON "文字を入力した後ボタンを押してください"
    AT (offset, offset + unitH * (line -1) ) +
    (dlgW - offset * 2, unitH - offset)
    SCRIPT  "MESSAGE CTSTRING(""入力した文字列"",0)"

USE DIALOG BOX
```

この例では、モーダルダイアログボックス上にテキストフィールドとプッシュボタンを配置し（**画面8**）、プッシュボタンを押すとテキストフィールドに入力されている文字列を獲得して表示する（**画面9**）、というスクリプトを記述しています。

このスクリプトから変数を使い始めました。もちろんこ

れまでの例で記述してきたように、変数を使わずに数値などの定数で指定していくこともできますが、将来のメンテナンスの便を考えると、変数を利用してスクリプトを記述したほうが便利な場合があります。

◇変数の定義

変数はDEFINEコマンドで定義します。HyperScriptにおいては、変数を定義する位置でグローバル変数とローカル変数に扱いを分けることができます。グローバル変数は、他のワークシートやスクリプトから値を参照することができる変数です。ローカル変数は、後で記述するユーザー関数の中で定義した変数のことで、定義した関数以外の場所で参照したり、使用したりすることはできません。

このスクリプトではコントロールの数（実際にはコントロールを並べる行数）をダイアログボックスの高さで表すようにしました。これで、たとえばダイアログサイズを微調整することになったとしても、スクリプト記述の先頭部分にあるダイアログボックスのサイズだけを変更すれば、ほかのサイズはダイアログボックスのサイズを元に計算された値で表示されます。

もし変数を使わずに値を直接指定していると、サイズの微調整などの際に、もれなく座標を見直さなければならず、手間がかかりミスの元となることがあります。実際のスクリプトを組むにあたって、変数を採用する、しないの判断は、作成物の規模やメンテナンスの度合いを考慮して決定することになります。

◇CTSTRING( )とCTVALUE( )

コントロールの内容を獲得する時は、CTVALUE( )とCTSTRING( )を使用します。

この関数を使うことで、テキストフィールドに入力された文字列を取り出したり、ポップアップメニューの何番目が選択されたかなど、コントロールの状態を得ることができます。CTSTRING( )の構文は以下のとおりです。

CTSTRING（対象コントロールの指定，取り出す情報の指定）

CTSTRING( )は、第1引数で指定されたコントロールについて、第2引数で指定された状態を、文字列で戻します。情報の指定は、コントロールの種類によって使い分けます。

対象コントロールの指定は、コントロールIDまたは、コントロール名、コントロールタイトル、コントロールエイリアスで指定します。コントロールIDはコントロールを作成した順番に自動的に一意に振られる番号です。この例では、テキストフィールドのコントロールIDに1が、プッシュボタンには2が割り振られます。

コントロールに、番号ではなく名称を定義する場合は、コントロール名やコントロールタイトル、コントロールエイリアスを定義します。コントロール名やコントロールタイトルをつけると、コントロール上に定義した文字列が表示されます。コントロール名やコントロールタイトル、コントロールエイリアスは、わかりやすい意味のある名前で設定しておくとメンテナンスが楽になるでしょう。

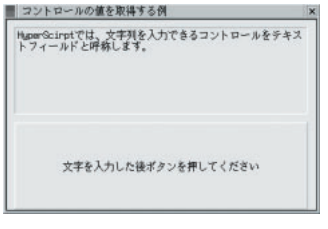
◇SCRIPTコマンド

生成したコントロールにスクリプト機能を加えるには、SCRIPTコマンドを用います。SCRIPTコマンドはコントロールだけでなくダイアログボックス自身やワークシートなどでも使用することができます。SCRIPTコマンドの引数は文字列として指定し、SCRIPTコマンドの中でさらに文字列を指定するなど、文字列表現中の文字列もダブルクォートで括って表現します。

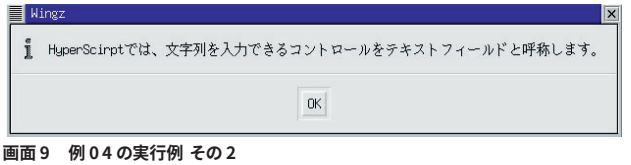
SCRIPTコマンドで指定したスクリプトは、1個の独立した新しいスクリプトとして定義され、動作します。コントロールなどには貼り付けず単体のスクリプトを定義する場合は、RUNコマンドを用います。RUNコマンドもSCRIPTコマンドと同様、スクリプトを文字列の引数として渡して定義します。このように、SCRIPTコマンドやRUNコマンドなどを使うことで個々の独立したスクリプトを文字列として必要に応じて別途定義できる点が、HyperScriptの真骨頂です。

■コントロールの属性設定

次に、前の例で使ったスクリプトを拡張してみましょう。



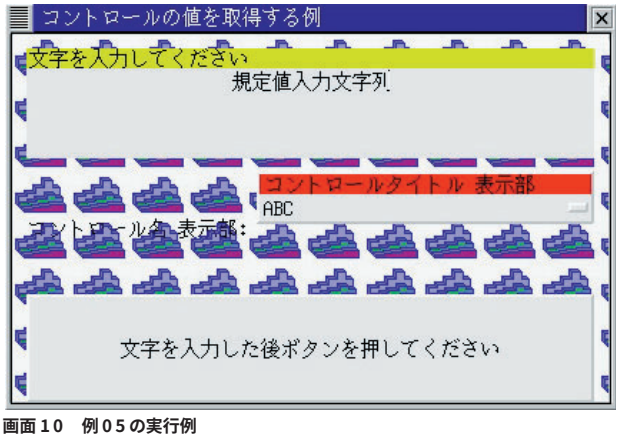
画面8　例04の実行例 その1



画面9　例04の実行例 その2
画面8のダイアログボックス上のコントロール情報をHyperScript関数で取得し、メッセージウィンドウに表示させています。

新しいコントロールとしてポップアップメニューを追加し、また、ダイアログボックスやコントロールの属性を指定してみます（**画面10**）。

```
例05 - sample05.scz -
DEFINE offset, dlgW, dlgH, unitH, unitMax, itemArr[5],
    line, winName, pct_1

   dlgW = 8000 ; dlgH = 5000 {* ダイアログボックスのサイズ *}
    offset = 200
    unitMax = 3 ; unitH = dlgH / unitMax
    winName = "コントロールの値を取得する例"
    pct_1 = "splash"
    GET itemArr[1..5] FROM "ABC", "DEF", "3番目のアイテム", 123, 456
    IF SEARCHARRAY( LIBRARYPICTURES(), pct_1, 1) = 0
        GET PICTURE GETENV("WINGZ2") &
        "/incl/icons/color/cgst3.gif" AS pct_1
    END IF

NEW MODAL DIALOG BOX NAMED winName AT (-1,-1)(dlgW, dlgH)
    DIALOG PICTURE pct_1 FULL TILED

    line = 1
    ADD FIELD
        AT (offset, offset + unitH * (line -1) ) +
        (dlgW - offset * 2, unitH - offset)
        CONTROL ALIAS "入力した文字列"
        SHOW CONTROL TITLE "文字を入力してください"
        MAXIMUM FIELD LENGTH 28
       NO LINE BORDER ; CONTROL TITLE FILL BG YELLOW()
        ALIGN CENTER
        PUT TEXT "規定値入力文字列"

    line = 2
    ADD POPUP MENU itemArr
        AT (offset, offset + unitH * (line -1) ) +
        (dlgW - offset * 2, unitH - offset)
        CONTROL ALIAS "popup1"
        SHOW CONTROL NAME "コントロール名 表示部"
       SHOW CONTROL TITLE "コントロールタイトル 表示部"
        CONTROL TITLE FILL BG RED()

    line = 3
   ADD PUSH BUTTON "文字を入力した後ボタンを押してください"
        AT (offset, offset + unitH * (line -1) ) +
        (dlgW - offset * 2, unitH - offset)
        SCRIPT
        "MESSAGE CTSTRING(""入力した文字列"",0) &
        REPEAT( CHAR(13), 2) &" , """ポップアップメニューは ""
        & CTVALUE(""popup1"",0) & ", """番目のアイテムが選択されています"""

USE DIALOG BOX
```



画面10　例05の実行例
ダイアログボックスの背景部分は、着色したり、ピクチャーで表示させたりすることもできます。

例05のスクリプトを例04のスクリプトと見比べてみてください。たとえば、「line = 1」で記述しているテキストフィールド（FIELD）の部分に、次の6つのコマンドを追加しています。

- **SHOW CONTROL TITLE：コントロールにタイトルを定義しています**
- **MAXIMUM FIELD LENGTH：テキストフィールドに入力可能な文字の長さの上限を定義しています**
- **NO LINE BORDER：コントロールの外周の枠線を非表示にしています**
- **CONTROL TITLE FILL BG：タイトル表示部分の背景色を指定しています**
- **ALIGN：テキストフィールドに入力する文字列の配置を指定しています**
- **PUT TEXT：テキストフィールドにあらかじめ配置しておきたい文字列を指定しています**

このようにダイアログボックスの生成やコントロールの

属性設定などは、上記のようなスクリプトを記述していくことになりますが、前述のHyperScriptアプリケーション・ダイアログボックスビルダーなどを使えば、ダイアログボックスの生成、コントロールの配置、属性設定について、いっさいスクリプトを記述することなく、マウス操作だけで行うことができます。ダイアログボックスビルダーを使った実習と、コントロールの属性を設定するコマンドについては、第2回で解説する予定です。

このサンプルのトピックスは次のとおりです。

◇コメントの指定

HyperScriptでは、{ }（中括弧）で囲った部分がコメント部となります。

◇配列

HyperScriptでは1次元から3次元の配列を扱うことができます。

配列の添え字は[ ]（大括弧）で括り、要素は,（カンマ）で区分けします。処理の途中で属性や要素数を変更する場合は、REDIMENSION コマンドを使います。ここでは、5つの要素からなる1次元の配列itemArrを使ってポップアップメニューコントロールの要素を定義しています。

◇環境の取得

GETENV( )関数を使うと、画面解像度などといったWINGZの環境の取得や、WINGZを起動したシェルの環境変数を取得することができます（**表1**）。サンプルでは、環境変数WINGZ2を取得しています（「GETENV("WINGZ2")」）。シェルの環境変数を取得する場合は、取得する環境変数をダブルクォートで括って指定します。

◇ピクチャー情報

HyperScriptではピクチャーデータも扱えます。手順としては、ピクチャーファイルの情報をいったん内部メモリ（ピクチャーライブラリ）に格納した上で利用することになります。

例05では、9行目でGET PICTUREコマンドを使って、ピクチャーファイルを読み込んでいます。「AS pct_1」はWINGZで使用する名称の指定です。読み込んだピクチャーの名前は、LIBRARYPICTURES( )関数の戻り値の配列で取得できます。ピクチャーデータのファイルパス、幅などの情報は、必要に応じてLIBRARYPICTUREINFO( )関数で取得します。

MESSAGE GETENV("HOME")	環境変数HOMEを取得してメッセージボックスに表示
GETENV("SHELL")	環境変数「SHELL」の取得
GETENV(5)	使用しているスクリーンの幅の取得
GETENV(6)	使用しているスクリーンの高さの取得
GETENV(16)	プラットフォームで使用されているファイルセパレータの取得

表1　GETENVの使用例

◇コントロール属性

次に、コントロールを生成している部分を注目してみましょう。たとえばテキストフィールドに対しては、NO LINE BORDERコマンド（枠線指定）、ALIGN CENTERコマンド（文字列配置指定）、PUT TEXTコマンド（文字列配置）などを指定しています。作成したコントロールの規定値を設定したり、他のコントロールの状態に応じて設定内容を変更することができるよう、HyperScriptにはコントロールごとに多様な属性設定コマンドが用意されています。ダイアログボックスとコントロールの各種設定方法については、次回で説明します。

■ワークシートの作成

```
例06 - sample06.scz -
NEW WORKSHEET "sample06.wkz"
    LOCATION (0,0)(8000,5000)
COLUMN WIDTH RANGE A1..AVLH32768 TO 1953

PUT "=" & "NOW()" INTO A1
SELECT RANGE A1
    FORMAT DATE 4

ADD FIELD RANGE FRAC( A2..B3, 50, 50, 200, 200)
SELECT FIELD 1 FROM 0 TO 0
    PUT TEXT "(=ADATE(A1,""元号g年m月d日""))"
    ALIGN CENTER
    LOCK FIELD TEXT

ADD PUSH BUTTON "+" RANGE A4
    SCRIPT "PUT A1+1 INTO A1"

ADD PUSH BUTTON "-" RANGE B4
    SCRIPT "PUT A1-1 INTO A1"

UNSELECT
```

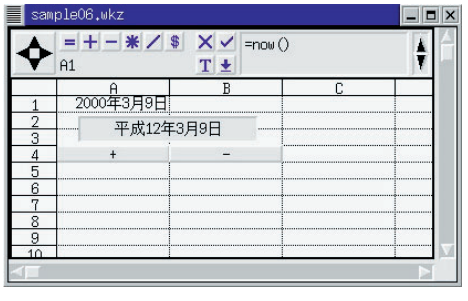
次に、ワークシートを使ったスクリプトを紹介します。

ワークシートはNEW WORKSHEETコマンドを使って生成し、シート名を引数として使用します。シート名にNULL（空白文字列）を指定した場合はユニークな名前で生成されます。LOCATIONはNEW WORKSHWTコマンドのオプションで、不要の場合は省略することができます。その場合、WINGZのデスクトップいっぱいの大きさでシートが生成されます。既存のワークシートを開く場合は、NEW WORKSHEETコマンドの代わりにOPENコマンドを使用します。

◇ワークシートセルへオブジェクトを配置する際の指定方法

ワークシートにもオブジェクト（コントロールやグラフ、ピクチャー、矩形など）を配置することができます。

通常は、ツールパレットから生成するオブジェクトツールを選択して、ワークシート上にマウスでマークすることで配置できますが、これをスクリプトで記述すると、例06のようになります。この例では、テキストフィールドとプッシュボタンをワークシートに配置しています（**画面11**）。ワークシートにオブジェクトを配置する場合は、セル領域上に配置する指定を行うことになります。このためワークシート上に配置させる場合は、ダイアログボックス上にコントロールを配置する際に使用した領域指定のATの代わりに、RANGEを使用します。また、指定したセルの領域の一部分にコントロールを配置する場合は、FRAC( )関数を使用します。FRAC( )関数は第1引数で指定したセル範囲を縦横それぞれ255等分割した単位で扱います。第2～5引数でセル範囲中の使用領域を指定します。以下にRANGEとFRAC( )関数の使用例を示します。



画面11　例06の実行例
ワークシート上のコントロールもダイアログボックス上のコントロールと同様に、スクリプト処理を設定できます。

画面12　例07のswitch.wkzワークシート
TARGET WINDOWコマンドで対象ウィンドウを切り替え、ウィンドウの表示／非表示を行います。

```
ADD PUSH BUTTON "新規ボタン" RANGE A1..B5
{＊A1..B5のセル領域いっぱいにプッシュボタンを生成＊}

ADD OVAL RANGE FRAC( A1..B5, 128, 50, 100, 255 )
{＊A1..B5のセル領域の一部分に円弧オブジェクトを生成＊}
```

■ワークシート属性の設定

例07のスクリプト（sample07.scz）は付録CD-ROMを参照してください。

WINGZ／HyperScriptの環境では、ダイアログボックスと同様にワークシートも、ユーザーインターフェイスのベース画面として利用することができます。ただし規定値設定のままでは、コントロールを消したりウィンドウの位置やサイズを変えることができてしまいますので、こういった事故を防ぐためのウィンドウやコントロールの属性を設定するコマンドもHyperScriptには存在しています。例07では、switch.wkz上のプッシュボタン［HIDE/SHOW］（**画面12**）を押すことで、対象シートのtargetWin.wkzの表示／非表示を切り替える処理を行っていますが、その際にswitch.wkzのウィンドウ属性を変更しているため、このウィンドウの位置とサイズはマウス操作では変更できないようになっています。また、セルやエントリーバーも通常のワークシートとは違って、表示されていません。さらにワークシートのスクロールバーも扱えず、セルを移動できる範囲にも制限を設けています。実際のプログラムでは、処理の段階の応じて制限を緩和したり、操作可能な領域を変更したりします。

■グラフの描画

基礎演習の最後のサンプルは、ワークシート上のデータを元にグラフを生成し装飾する例です（例08のスクリプトは付録CD-ROMのsample08.sczを参照してください）。この例を応用してDataLink機能やピクチャーデータへの保存機能と組み合わせれば、定期的にデータベースから取り出したデータをもとにグラフを作成し、Webサーバ上にピクチャーファイルとして保存、グラフイメージをブラウジングする、などの処理も可能となります。

スクリプト中、画面描画機能を一時中断させている個所（「INVALIDATE ON/OFF」）に注目してください。INVALIDATE ONコマンドが実行されると、オブジェクトなどの描画が行われなくなります。その後、実行中のスクリプトが終了するか、INVALIDATE OFFコマンドが実行されると、必要な描画が1度だけ行われます。グラフや

コントロールに対して複数の表示属性を設定する際などに、これらのコマンドを使って描画を一時中止すると、処理時間を大幅に短縮することができます。HyperScriptで扱える描画関連のコマンドは、このほかに次のものがあります。

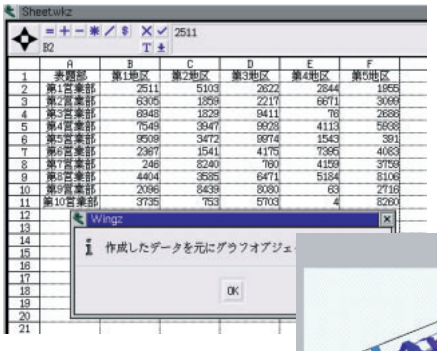
●REPAINT ON/OFF
対象ウィンドウの再描画を制御します。
●REPAINT SELECTIONS ON/OFF
ワークシートを対象とした再描画制御コマンドです。選択セル領域の反転表示、選択オブジェクトのハンドル表示を制御します。
●REPAINT CONTROL
ダイアログボックス上のコントロールなどを個々に選択して再描画を強制します。
●REPAINT WINDOW/REPAINT ALL WINDOWS
再描画の制限を解除した後などにウィンドウの描画を強制します。前者は対象ウィンドウのみの描画を、後者はすべてのウィンドウが対象となります。

応用演習 ユーザー関数変数と参照

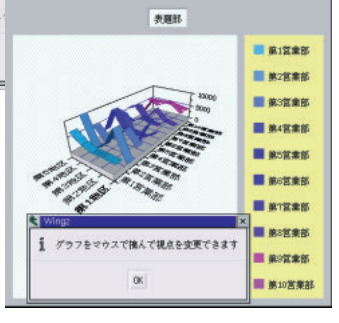
これまでの演習でHyperScriptの概要はおわかりいただけたのではないでしょうか。ここからは応用演習として、HyperScriptの機能を組み合わせて記述し、全体でひとつに完結できる処理を見ていくことにします。

■ユーザー関数の例 ～素数検索～

HyperScriptの分岐処理とユーザー関数の例として、素数を検索する例を記述してみましょう。



画面13　例08の実行例 その1
ワークシートのデータをもとにグラフの生成と装飾を自動で行います。



画面14　例08の実行例 その2
グラフオブジェクトの各部品ごとに、装飾を加えてみました。3Dグラフはマウスを使って直接視点を変更できます。

まずはじめに、単純な仕様でとにかく動くレベルのものを作成し、その後、効率を少し考慮したものを作成します。

作例1
単純な計算で、はじめの素数25個をメッセージボックスに表示する

```
DEFINE n, pn[25], pnCount
    pn[1] = 2 ; pnCount = 1

    n = 3
    WHILE    CheckNumber( n )
    n = n + 1
    END WHILE
    MESSAGE COMMALIST( pn ) TITLE "素数: はじめの 25 個"

FUNCTION CheckNumber( chk )
    DEFINE k

    FOR k = 1 TO pnCount
        IF MOD( chk, pn[ k ] ) = 0
            RETURN 1
        END IF
    END FOR
    pnCount = pnCount + 1 ; pn[pnCount] = chk
    RETURN IF( pnCount < 25, 1, 0 )
END FUNCTION { CheckNumber }
```

検査する数（変数n）をWHILEループで加算して調査し、見つかった素数は配列pnに格納していきます。はじめの素数2はあらかじめ配列pnの先頭の要素pn[1]に格納しておきます。

実際のチェックはユーザー関数CheckNumber( )で行っています。FORループの個所で、引数で与えられた数をすでに見つかっている全素数で割って調べています。効率悪いですね。まずは変数の総数を少なくして、とにかく動くものを作ってみました。

割り切れる場合は、CheckNumber( )は数値1を戻して関数を終了します。最後まで割り切れなかった場合は、素数と判断して配列pnの新しい要素として格納します。素数として判断した場合の関数の戻り値については、見つかった素数の総数が25個に満たない場合は数値1を戻し、25個まで見つかった場合は、数値0を戻して関数を終えます。

WHILEループはCheckNumber( )の戻り値で継続する

か否かを判断します。真（戻り値=1）の間は、調査する数（変数n）をひとつ増やして、再びCheckNumber( )を呼び出します。素数の総数が25になった場合だけ0が戻りますので、その場合はWHILEループを抜けて結果をメッセージボックスに表示します。

◇ユーザー関数

ユーザー関数は、FUNCTION～END FUNCTIONで括って定義します。関数名にはHyperScriptの予約語、セル座標名、すでに定義済みの変数名や関数名に重複しない名前を定義できます。ユーザー関数で利用できる引数は50個までで、引数の個数を可変にすることはできません。また、ユーザー関数内でDEFINEした変数、配列はユーザー関数内だけで使用できるローカル変数となり、ユーザー関数の引数も同じ扱いとなります。ユーザー関数の戻り値を設定する場合は、RETURNコマンドを使用します。

ユーザー関数の呼び出しは、CALLコマンドを使用することで可能ですが、単に変数への代入式やセルの数式として設定しておいても、再計算時などの適当なタイミングでユーザー関数は呼び出されます。また、ユーザー関数は再帰的に呼び出すことも可能です。条件によって関数の処理を中断する場合には、EXIT FUNCTIONコマンドを使用します。

◇分岐処理

分岐処理の例としてここではIF～ENDIF、WHILE～END WHILEを使用しています。中断する場合はユーザー関数と同様、EXIT IF、EXIT WHILE を使用します。またIF分岐ではELSEやELSEIFも使用できます。分岐処理としてはこのほかにCASE文を使用することもできます。

では次に、作例1を少し発展させてみましょう。

作例2
求める素数の数を可変にする
判断効率を上げる
奇数だけチェックする
調査する数の平方根の数まで調べても割り切れない場合は素数と判断する
結果をワークシートに表示する
計算終了時には、計測にかかった時間を表示する

効率を上げる措置は、結構簡単に追加できると思います。HyperScriptでは、kの平方根はSQRT(k)またはk^(1/2)で求めることができます。作例1のスクリプトを元に作例2の条件を検討してみてください。作例2の実例として 例09を記述して付録CD-ROMに収録してあります（sample10.scz）ので、参考にしてください。

例09のスクリプトでは、はじめに探し出す素数の個数を指定し、見つけた素数を順番に配列pnに格納、指定した個数まで素数が見つかったら、結果をワークシートに出力して検索にかかった時間を表示して処理を終了します。

結果をテキストファイルに保存したい場合はPUT pn INTO A1を実行した後で、SAVE TEXTコマンドを使うとよいでしょう。このスクリプトでは2つのユーザー関数を使用しています。SetPn( )は、はじめの素数2を格納するのにも使用しています。このように、ユーザー関数を使って処理をより汎用化させていくこともできます。

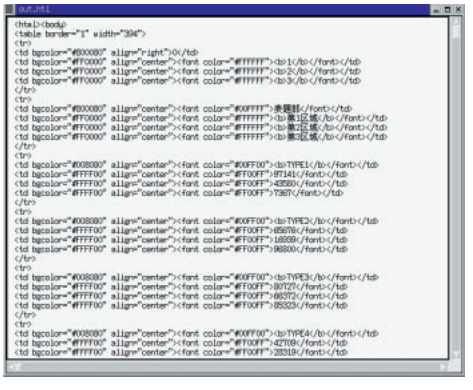
■スプレッドシートの表示をHTMLへ出力

第1回の演習の集大成として、少し大きなスクリプトの例を作成してみました（例10のスクリプトは付録CD-ROMのsample10.sczを参照してください）。このスクリプトでは、ワークシート上で選択しているセル領域、データを取得し、内容を変数に格納してから、HTMLタグを付けたデータを生成し、最終的にテキストファイルに出力します。

扱うウィンドウが増えたりスクリプトが大きくなると作成も解析も大変になりますが、このスクリプトのように、ユーザー関数内だけで処理できる変数はローカル変数で対応するなどして取り扱うグローバル変数の数を減らすように記述すれば、解析やメンテナンスも比較的楽に行うことができます。

スクリプトの内容を見ていきましょう。

メインの関数であるCreatehtml( )は、ユーザ関数CellrangeInfo( )を呼び出し、この関数の戻り値が1の場



画面15　ワークシート情報をもとに生成されたHTMLタグの例

合は、ユーザー関数GetData( )、CreateText( )を呼び出しています。

ユーザー関数CellrangeInfo( )ではエラーチェックを行っています。関数の動作としては、有効なワークシートがなかったりセル領域が正しく選択されていない場合は数値-1を戻り値として返し、処理を終えます。単一のセル領域が選択されている場合は、正常動作として、選択情報となるデータを配列で戻しています。

◇選択情報を取得する関数NRSELECTIONS( )とSELECTIONTYPE( )

このユーザー関数CellrangeInfo( )ではNRSELECTIONS( )、SELECTIONTYPE( )といったセルや選択項目に関するHyperScript関数を利用することで目的の動作を実現させています。

●NRSELECTIONS( )：選択中の対象物(セル領域、オブジェクト、コントロール)が何個あるかを数値で戻します。
●SELECTIONSTYPE( )：選択対象の種類が何かを示す数値を戻します。選択物が複数の時は何番目に選択したものを対象にしているのかを、引数で指定します。

ユーザー関数GetData( )ではCellrangeInfo( )で得られたデータ範囲に関する各種情報を取得し、配列データとして戻しています。FORMAT( )、TEXTCOLOR( )、TEXTSTYLE( ) などの書式情報を扱うHyperScript関数を、HTMLタグに出力するという目的に合わせて、適切な文字列に置換しているのがお分かり頂けると思います。

◇ユーザー関数AlignStat( )の例

このユーザー関数では、引数valの値に相当する配置状態を文字列で戻しています。

HyperScriptではセルの配置情報をFORMAT( )関数の戻り値と12288（10進）の論理和で表しています。

```
BITAND( FORMAT(), 12288 )
```

このままではHTMLのタグにはなりませんので、配置状態を表す値を該当する文字列に置換しています。このように、条件によって分岐する場合はCASE文を利用します。

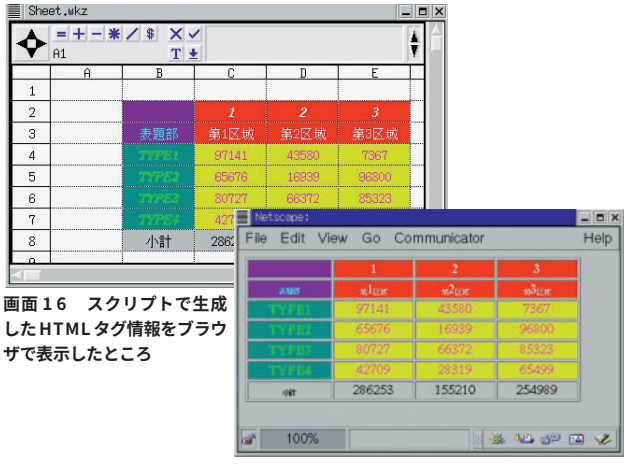
```
FUNCTION AlignStat( val )
    DEFINE rt
    CASE val
        WHEN 0
            rt = ""
        WHEN  8192
            rt = "center"
        WHEN 12288
            rt = "right"
        OTHERWISE
            rt = "left"
        END CASE
    RETURN rt
END FUNCTION { AlignStat }
```

ColorStat( )では、別のユーザー関数をさらに呼び出している個所がありますが、このように個別処理については別途ユーザー関数にして利用すると、再利用できて便利です。

最後にユーザ関数CreateText( )では、引数で与えられた情報をもとにHTMLタグを生成しています。まず、NEW DOCUMENTコマンドを使って出力用のウィンドウを生成していますが、はじめは非表示オプションを使って、処理が終わるまでデータシートを画面に表示しないようにしました。このように、中間処理のデータなどのウィンドウを非表示のまま扱うこともできます。

ユーザ関数CreateText( )の処理内容は単純にデータを並べていっているだけですので、取得するデータに対応するタグを出力している個所を更新すれば、扱える属性を簡単に増やすこともできます。

今回は第1回目としてHyperScript自身の解説を中心に説明しました。次回はコントロールを多用したダイアログボックスの生成と処理、イベント処理の実習を行います。



画面16　スクリプトで生成したHTMLタグ情報をブラウザで表示したところ

瑞穂の国の人なれば

Emacsはじめました

第3回　日本語とEmacs

Linuxをはじめとするフリー UNIX系OSで、Emacsを使う大きな理由となっているのが日本語の読み書きです。日本語の入力方法にはいろいろありますが、今回はとくに、Linuxの多くのディストリビューションでよく使われているかんなとWnn4に焦点を当てて、上手な使い方をみていきましょう。

文：佐々木太良
Text : Taroh Sasaki

文字とコンピュータ

コンピュータは欧米で開発されたので、おもに英語を喋ります……というのは冗談ですが、アジア諸国など表現に数多くの文字を必要とする国では、欧米に比べて、入力・表示とも恵まれない環境にあります。これは文字の表現方法に理由があります。

コンピュータ上では文字の図形（フォント）とその図形を表現するための番号（文字コード）を対応させる必要があります。現在のコンピュータにとって都合のよい1バイトで取り扱える文字は256種類です。英語圏ではこれでも十分で、英文字（大文字・小文字）、数字、記号などをこの範囲で割り当てています。ヨーロッパの他の言語はその他にも特殊な文字（ウムラウトなどがついた文字）が必要ですが、それでも256種類を超えることはないようです。

一方日本や中国では使用する文字の種類がとても多く、われわれが小・中学校で習う漢字だけでも2000以上あります。日本語は歴史的に、1文字を表現するために2バイトを使用してきました。理論的にはこれで6万5536種類の文字まで表現できるわけです。ハングル語や中国語（北京系および台湾・香港系）など多国語対応については、折をみて紹介していこうと思います。これらの言語は1文字を1バイトでは表せないので、ここではまとめて「マルチバイト文字」と呼ぶことにします。

Emacsにかぎらず、ソフトウェア開発者にとってマルチバイト文字の取り扱いはなかなかやっかいなことです。UNIX系のOSでは、ソフトウェアを各国語用にローカライズするのではなく多国語対応をめざす動きがあり、Emacsもそうした試みの結果、日本語が使えるようになっているわけです。

インプットメソッド

さてマルチバイト文字は、一般にキーボードから直接入力することができません。この連載の第1回でメタキーを使って漢字を直接入力する裏技（バカ技?）を紹介しましたが、まさかこの方法で漢字のすべてのコード（入力できる2つのキーの組み合わせ）を覚えるわけにはいかないでしょう[*1]。

そこでインプットメソッドの登場となるわけです。インプットメソッドは、アプリケーションソフトウェアからいったんキーボード入力を奪い、なんらかの加工をしてから渡すソフトウェアです（**図1**）。インプットメソッドはキー入力を奪う必要があるため、X上の機能として実現されて

[*1]：じつはかつて（今もあるかな?）、すべての漢字文字の入力方法を丸暗記してしまおうという体育会系2ストローク入力メソッドというのが実在したのです。

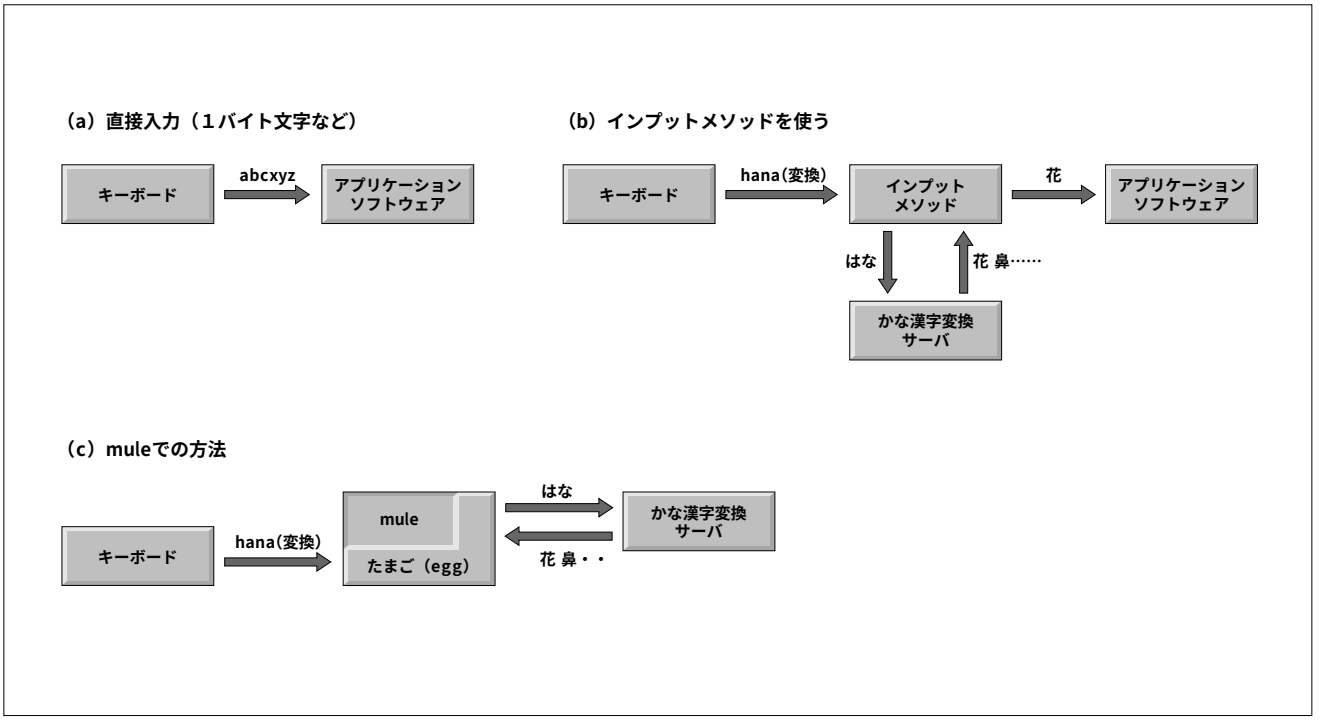


図1　インプットメソッド

います。これはWindowsのかな漢字変換でもおなじみの方法ですが、それらがひとつのソフトウェアでキー入力の横取りから漢字変換までを行うのに対し、Linuxでよく使われるかな漢字変換ソフトウェアのかんなやWnnにはキーボードからの入力を奪う機能はなく、Linuxマシン内あるいはネットワーク上でデーモンプログラムとして動作し、通信プロトコルを使ってインプットメソッドからかな文字列を受け取り、変換した漢字文字列を送り返します。しかしユーザーにしてみれば、インプットメソッドがそれ自身で変換しているのか、変換サーバと通信しているのかはこの際問題ではないでしょう。

viや日本語対応のXアプリケーション、Turbo LinuxやLaser5 Linuxに含まれているXEmacsはこの方法を使い、**図1-b**のようにkinput2などのインプットメソッドとWnnやかんな（Canna）などのかな漢字変換サーバを組み合わせて日本語入力を実現しています。

インプットメソッドを使えば、日本語に対応しているソフトウェアならどれでも共通のかな漢字変換が利用できることになり、したがって、

- **インプットメソッドに好きなものを使える**
- **いったん習熟したインプットメソッドをどのソフトウェアにも使える**
- **アプリケーションソフトウェアの開発者は、インプットメソッドに気を遣う必要がない**

などなどの利点があります。現在、インプットメソッドの代表的なものとしてkinput2があげられますが、将来あらたなものが開発されれば置き換えることも可能です。

一方muleの場合は、Emacs lispでプログラムされた「たまご」（egg）というマクロが組み入れられており、Wnnを組み合わせて使う場合には、kinput2のようなインプットメソッドを使わずに直接Wnnサーバと通信してかな漢字変換を行うようになっています（**図1-c**参照）。また、かんなもライブラリを使って同様のインターフェイスを組み入れるようになっています。

すると、muleで使用されているeggはなにかインプットメソッドの独立性という利点を損なっているような気もします。しかし、eggは非常によくできていて、インプットメソッドを使ってEmacsに漢字を渡すよりも機能が高く、軽快なので、文章の執筆などには実用的です。また各種の設定を自分で変える場合、カスタマイズ性もたいへん高いのです。と書くとkinput2の作者に失礼ですが、これはC言語でX上のアプリケーションを書くよりもEmacs lispの柔軟性が幸いしたのではないでしょうか。現状ではkinput2の安定性があまりよくないこともあって、本気で文章を書きたい人にはkinput2でXEmacsに入力する方法はあまりおすすめできません。

日本語読むならEmacs、日本語書くならEmacs

Emacsというエディタだけがeggやかんなのライブラリの恩恵を受けられるわけで、使用範囲が限定されているように思われるかもしれません。しかし、Emacsの使い途はエディタだけにとどまりません。Emacs lispでマクロを書けば、メールの読み書きやチャットなど、外部のアプリケーションを特別に用意する必要がないほどです。

というわけで、日本語を扱うような局面をすべてEmacs（mule）ですませられるようになり、とにかく「なにか字を書く仕事ならEmacs」と固く心に誓ってしまって一生やめられなくなるのです（筆者のこと）。

Wnnについて知る

Wnn（うんぬ）は、京都大学、オムロン、アステックが共同で開発したかな漢字変換プロセッサです。連文節変換すらめずらしかった当時、「わたしのなまえはなかのです」の文章をまるごと一発で変換できる賢いものを目標に開発されたため、文節の先頭文字を取ってWnnと名付けられました。

かな漢字変換プロセッサをデーモンプログラムによるサービスとして動作させ、ネットワーク上で利用するという枠組みは、きわめてクールでネットワーク的なものだと思います。ネットワークで接続された複数のコンピュータで辞書のみを共用する、というやり方もあるでしょうが、かな漢字変換を専用にやってくれるデーモンがいるというのはなかなか楽しいものではないでしょうか。

Wnnは1987年にフリーソフトウェアとして配付されて以来、バージョンアップを重ね、現在では同じ技術をもとにして中国語（香港・台湾バージョン）のtWnn、中国語（大陸バージョン）のcWnn、ハングル語バージョンのkWnnなどが開発さています（Windows版のcWnn98、kWnn98がオムロンソフトウェアから販売されています）。

Wnnがフリーソフトウェアとして配付された最後のバージョンはWnn4ですが、そのソースコードはGPLに基づいて再利用が自由にできるようになり、改良はFreeWnnプロジェクト（http://www.tomo.gr.jp/FreeWnn/）に引き継がれています。

その一方で、文節間の関係も学習する、より高度な学習機能をもったWnn6がオムロンソフトウェアから発売されています（http://www.omronsoft.co.jp/）。Wnn6はLinux／FreeBSDなどで使えるので自分でもふだん使っていますが、学習機能だけでなく、外来語を入力して英語の綴りに変換してくれる辞書なども備わっていて、個人的には一押しのお勧めです。Windows向けのWnn98も販売されているので、マルチOSユーザーにとってもありがたい製品だと思います。

Wnnの特徴をひとことでいうと次のようになります。

- **さまざまなOS、環境で使える。日本語だけでなく中国語や韓国語でも使える**
- **変換効率、学習機能が優れている**
- **重い**

Column

muleの歴史

Emacsは、Free Software Foundationが開発と配布をしているGNUのソフトウェアのひとつですが、日本では日本語を扱えるように拡張したNemacsが開発され、1992年ごろまで利用されていました。その後、日本語だけでなく各種の中国語やハングル語など、多数のマルチバイトコードが扱えるmule（multi-lingual Emacs）が開発されてNemacsにとって代わり、さらに改良が加えられています。

マルチバイトコードへの対応は、基本的に本家GNUの開発とは独立に進められているので、muleの最新版は、Emacsの最新版よりメジャーバージョン番号にして常に1〜3版程度遅れています。TurboLinux日本語版4.0やVine Linux 1.1CRに付属しているmule 2.3は、Emacs19ベースのものです。muleにはバージョン番号とバージョン名があって、mule 2.3にはSUETSUMUHANAというバージョン名がついています。

本家Emacsでもバージョン20以降、マルチバイト文字に対応しようという動きがありますが、mule（Emacsバージョン19以前）で培われた技術を導入しようという試みもなされており、XEmacs + muleというバージョン番号を持ったものもあります。

ともあれ、現在はこれらの過渡期にあり、ディストリビューションに付属しているEmacsは二派に分かれているようです。ユーザーから見ると、mule（Emacs19）とXEmacs + mule（Emacs20/21）ではたいへんな違いがあります。外観もさることながら、近ごろのXEmacs対応のmuleでは多国語対応のしかたも変わってきているようです。本連載ではこれ以降も、問題となりそうな差異についてはその都度記していくことにします。

変換効率については、使い方や書く文章によって異なると思いますので、あくまでも筆者の主観ということになります。だた、自分自身ではまとまった文章を大量に書くことが多く、句点までいっきにひらがな入力をし、一回で変換するのが習慣になっているのですが、そうした経験からほかのWindows用インプットメソッドと比べても、Wnn4／6はかなり時間の節約になると実感しています。

かんなについて知る

「かんな」（Canna）は、NECのワークステーション用インプットメソッドに端を発しています。「かな」を意味する古語の「かんな」に由来しているそうです。欧文表記では“Canna”となります。こちらもWindows用が使用可能です。

個人的にはなかなか連文節の学習をしてくれない（ようにみえる）ので、かんなはあまり使っていませんが、その軽快なところが気に入っている人、またLinuxの多くのディストリビューションに最初から付属している（Emacsがかんなを使うよう設定されている）ことから、最近ではメジャー派になってきました。

また、各種入力方法で操作性が統一されている、というのも魅力のひとつでしょう。かんなはEmacsのインターフェイス（Emacsにかんなの機能が組み込まれたもの、ライブラリ）のほか、kinput2などでも利用することができます。かんな派の人には、Netscape Navigatorなどの他のアプリケーションにkinput2で漢字を入力するとき、Emacsと同様の操作性にしようとすると、Wnnのローマ字かな変換や編集操作があまりにもたまごに頼りすぎているので設定が二度必要（mule上のかんなの設定と、kinput2の設定）で面倒くさい、という意見もあります。

日本語入力のいろは

この雑誌の読者で、今までかな漢字変換を使ったことがない人はいないでしょう。ここではまず、WnnやCannaを使ったかな漢字変換の基本操作を並行して説明していくことにします。

■かな漢字変換のon/off

さてかな漢字変換の開始と終了です。いきなりWnn、かんな共通でない操作がでてきてしまいました。かな漢字変換をoffにしている状態では、キーボードから入力したものが直接Emacsのバッファに渡されます（**図1-a**参照）。この状態からかな漢字変換をonにするには、Wnnの場合は**C-**、かんなの場合は**C-o**をタイプします。ステータス行の左側は、次のように変わります。

Wnnの場合：

```
      [--]
C-¥   [あ]
```

Cannaの場合：

```
      -
C-o   -[ あ ]
```

ただし、C-oはopen-lineコマンド（第2回目の操作一覧表を見てください）とかち合っていますので、かんなを使うように設定すれば当然open-lineコマンドが使えません（あるいは他のキーに割り当てられています）。

また、かんなを組み込んだ状態のmuleであっても、M-x cannaとしないとかんなが使えない場合があります。こ

Column

たまご

muleからWnnを利用する場合に用いられているのが「たまご」（egg）というインターフェイスです。たまごが備えている機能は次のとおりです。

- ローマ字入力をかな文字に変換
- 入力文字列をWnn4に渡し、漢字を受け取る
- 中国語、ハングル語の文字単位の変換（ITS）

本文でcWnn、tWnn、kWnnを紹介しましたが、中国語・ハングル語については、単漢字変換程度と思ってください。日本語の場合、今や携帯電話ですら単語変換や文節変換が当たり前なので、ひらがなから漢字への変換は変換サーバに任せています。

たまごはmuleとともに育ってきたため、muleに組み込まれたたまごの機能を切り分けることはむずかしいようです。

たまごが使うかな漢字変換サーバはjserver（Wnnのプログラム名）と決まっているわけではないので、たまご forかんなとか（ちなみにかんなの変換サーバはcannaserverという名前です）、たまごfor SJ3などがあってもよさそうですが、たまご（たかな）と呼ばれているバージョンではWnn4を使うことになっているようです（商用のWnn6には、Wnn6と通信できるバージョンも付属しています）。

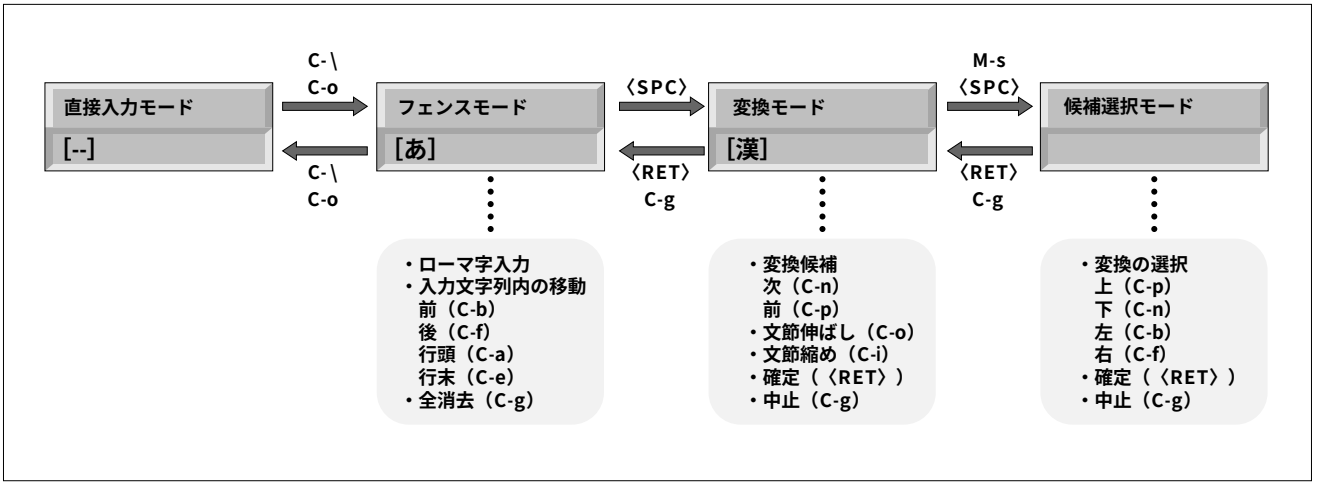


図2　モードの変化

のような場合は、スタートファイル~/.emacsに1行、

```
(load-library "canna") (canna)
```

と書いておけばOKです。はじめからかんなが使える場合は、あらかじめこのように記述されているスタートファイルが用意されていると思われます。

■モード

Wnnでもかんなでも、今なにができる状態なのかのモードというものがあります。入力操作のあいだ、どのモードなのかを無意識のうちにも指で使い分けることがたいせつです。**図2**の説明は、Wnnとかんなの両方をまとめて説明する都合上、それぞれのもとのモード名とやや異なっています。

■よみがなの入力

かな漢字変換をonにした直後は、入力したローマ字が次々とひらがなに変換されます。この状態では、ひらがらの両側に“｜”が表示され、ローマ字かな変換中であることを示します。この状態はフェンスモードと呼ばれます。

■変換

入力したひらがな文字列が正しいことを確認してから〈SPC〉を押すと、かな漢字変換作業に入ります（変換モード）。モードの表示は“［漢］”または“ -［漢字］”に変化し、フェンスの内側は文節ごとにスペースが挿入され、候補が確定していないことを示します。

この状態で何度か〈SPC〉を押すと、最初の文節の候補を選ぶことができます。候補の上手な選び方および文節の伸ばし方、縮め方については、次に見てみることにしましょう。とりあえず候補を確認して〈RET〉を押すと、フェンスがとれて文節間のスペースも詰められ、はじめてバッファに挿入されます（ホントのことを言うと、ここまでの操作はすべてバッファ上の文字列を書き換えることでユーザに提示しているのですが）。

■かな漢字変換を抜ける

かな漢字変換に用がなくなり、直接入力モードに戻したくなったら、再度C-\またはC-oを押すだけです。以上の一連の操作とモード行の変化を**図3**に示します。

■気に入らない変換結果を直す

変換モードにいるとき、最初に提示された文節の区切り方が適切とはかぎりません。えっ？「単語ごとに変換・無変換するから文節なんて関係ないよ」？　まあそういわずに、長い文を打って一発変換してみてください。きっと入力効率が上がります。文節の関係を学習してくれるかな漢字変換プロセッサであれば、「お湯は熱い」「友情は厚い」の変換も夢ではありません。

■文節を伸ばしたり縮めたり

注目する文節を移動（カーソルをその文節の上に移動）させる場合は、**C-b**（左方向）、**C-f**（右方向）、**C-a**（先頭文節へ）、**C-e**（最終文節へ）などが使えますから、コントロールキーでカーソルを移動することに慣れているあなたは、もはや何も覚えることがありません。と言いつつ、X環境ならカーソルキーでも注目文節は移動できるのでし

キー操作	バッファとモード行の表示
かな漢字変換OFF	-
C-o	- [あ]
o t a n……	| おたのしみはまたあした |
s h i t a	- [あ]
〈SPC〉	| お - 楽しみは またあした |
	- [漢字]
〈SPC〉	| 御 - 楽しみは またあした |
	- [漢字]
〈RET〉	御楽しみはまた明日
	- [あ]

図3　かんなのモード行の変化

キー操作	バッファとモード行の表示
かな漢字変換OFF	[--]
C-\	[あ]
o t a n……	| おたのしみはまたあした |
s h i t a	[あ]
〈SPC〉	| お - 楽しみは またあした |
	[漢]
〈SPC〉	| 御 - 楽しみは またあした |
	[漢]
〈RET〉	御楽しみはまた明日
	[あ]

図4　Wnnのモード行の変化

た（マウスでクリックする方法だけは使えません）。

前にも述べたとおり、注目文節の上で〈SPC〉を何度か押すと、変換候補が選べます。また**C-p**（前候補）、**C-n**（次候補）で（〈↑〉〈↓〉でもOK）候補を選べるので、**図4**のように上下左右に文節ごとの候補が並んでいると考えれば簡単です。

文節を伸ばす場合は、候補となる文節を選んだうえで**C-o**、文節を縮めるなら**C-i**です。注目している文節の末尾が伸び縮みし、その後ろの文節は再度、最適と思われる変換がなされて表示されます。

	| おたのしみはこれからだ |
〈SPC〉	| お-楽しみは これ 空だ |
C-f（2回）	| お-楽しみは これ 空だ |
C-o	| お-楽しみは これか 羅だ |
C-o	| お-楽しみは これから 田 |
C-o	| お-楽しみは これからだ |

ところで文節ってどうやって区切るか知ってますか? 中学の国語の先生が教えてくれたのですが、「ね、」を挿入できるところが文節の切れめだそうです。「私はね、かつてね、Emacsをね、……」。

■上手な候補の選び方

文節選びを説明したのでここに補足しますが、じつはWnnとかんなで候補の選び方が（デフォルトの設定で）若干異なっています。

Wnnでは、〈SPC〉を何度か押しても候補の一覧が現われることはありません。これに対して、かんなでは〈SPC〉を2回以上押すと候補一覧がミニバッファに出て、そのなかから選択できるようになります（C-nやC-pを何度押しても候補一覧にならないのは、Wnnと同じです）。

Wnnでは、**M-s**をタイプするとはじめて候補一覧が表示されます。人名など、候補が大量に出てくると予想される場合は、変換モードに入ってすぐにM-sをタイプするとよいでしょう。逆に2～3回で出てきそうなら〈SPC〉で選んだ方が簡単です。

Wnn、かんなとも、候補一覧でのミニバッファの操作方法はほぼ同じです。C-b、C-fでミニバッファ中の左右の候補を選択し、C-p、C-nで上下の候補を選択します（**図4-c**）。候補が1行単位で出てくると考えれば、これもまたカーソル移動と同じです。

かんなでは次候補の選択に〈SPC〉も使うことができます。つまり変換モードからいつのまにか候補一覧モードに変化しているので、同じ操作で次々候補を選んでいけるようになっています。

どの候補を選ぶかが決定したら、〈RET〉を押します。候補一覧のメニューが消え、変換モードに戻ることができます。

■無変換ってどうするの?

Wnnもかんなも同様ですが、無変換やカタカナ変換操作がほしいことがあります。もちろん、フェンスモードからいきなり〈RET〉を押せば、ローマ字かな変換されたひらがながそのまま登場しますが、「点・丸まで打って一発変換」主義にはなじみません。また専用の「ひらがな変換」（Wnnでは**M-h**）、「カタカナ変換」（Wnnでは**M-k**）などもあるにはあるのですが、いちいち覚えるのも面倒くさい

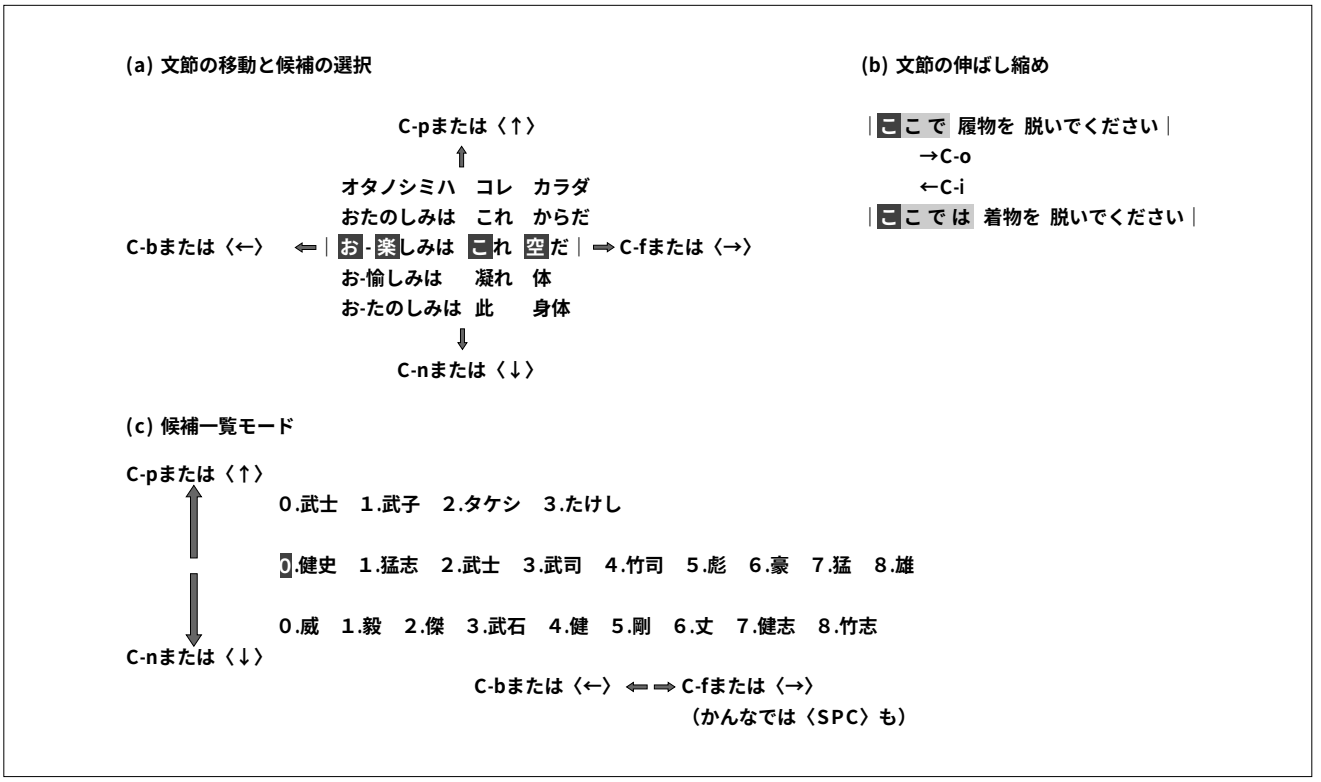


図5　変換操作

でしょう。

こんなとき、Wnnもかんなも候補の並べ方は「最初の候補の直前がひらがな、その前がカタカナ」だという大原則を覚えておくと便利です。**図4-a**でわかるように、候補はリング状に並んでおり、最後の候補の次は最初の候補になります。つまり、ひらがな変換をしたいときは注目文節でC-pを1回、カタカナ変換なら2回タイプすればよいわけです。

■困ったときはC-g

連載第1回にでてきた、「困ったときはC-g」の大原則はたいがいの場面で生きています。かな漢字変換でも、候補一覧モードならば変換モードに、変換モードならフェンスモードに、フェンスモードならばフェンス内を全クリア、というようにひとつ前の段階に戻れるようになっています。また後述するWnnの各種入力方法（部首入力など）では、選択に選択を重ねているうちに訳がわからなくなったりするのですが、そんなときも前の段階に1ステップだけ戻せます（状況によって異なります）。

■さらに困ったときは

細かい説明を読みたいときは、連載の第2回目で説明した、ヘルプの一種「info機能」が役立ちます。Wnnにもかんなにも、ともに日本語のinfoが用意されているので、さらなる使い方は**C-h i**（または**C-H i**）からegg-jp（Wnnの使い方はWnnそのものの使い方というよりたまごの使い方ですので、こういうタイトルになっています）やCanna-jpの項目を選んでみてください。

Wnnを使ってみたい!!

さて、かんなはディストリビューション標準の漢字入力方法として使える場合が多いので、ここからはWnnを偏愛する者としてWnnを使う場合を紹介しましょう。筆者の一押しはWnn6ですが、パッケージソフトの購入がままならない（あるいは日本語入力の効率にそれほどの価値を見いだせない）人もいるでしょう。

Wnn4の導入手順は、ディストリビューションによって異なります。まず、Emacs（muleまたはXemacs）がWnnやかんなを使えるようにコンパイルされていることが必要です。Wnnとかんなは併用可能ですが、ディストリビューション付属のEmacsがそうなっていない場合、両方使えるEmacsのRPMを入手して入れ替えます。http://www.linux.or.jp/jrpm/rpms/には、使用する環境

とかな漢字変換ソフトウェアの組み合わせで各種のRPMがありますが、ぴったりのRPMが見つからないときにはむずかしくなります（むろん、自分でEmacsの再コンパイルができればOKです）。対応ずみのEmacsのパッケージが入手できれば、あとは変換サーバ本体を同じようにパッケージで入手するだけです。

Wnn6であれば、製品にインストーラスクリプトなどが付属しているため、手順にしたがってインストールするだけです。製品の紹介はhttp://www.omronsoft.co.jp/を参照してください。

おわりに

さ〜て、次回は?（揉み手）　いよいよEmacsを各種のアプリケーションとして使ってみましょう。その第1回として、電子メールの読み書きというテーマでお贈りします。メールなら専用のメーラを使ったらいいじゃんという人も、まずはお試しください。

また、連載で取り上げてほしいテーマや意見、質問、率直なご感想は、taroh@taroh.orgあてにお寄せください。

では、Happy Hacking!!

Column

Wnnの上手な使い方

Windowsのインプットメソッドを使ったことがある人ならば、使用するキーに多少の違いがあるにせよ、漢字入力の操作自体は理解できるでしょう。ここでは、マニュアル類にはなかなか書いてないことを紹介しましょう。

■各種入力方法

C-^をタイプすると、各種入力方法が使えます。ここでは、メニューの選択などを繰り返していけば、読みのわからない字や、特殊な文字に到達できます。

C-^をタイプしてみましょう。最初のメニューの「JIS入力」では、漢字コード（JISコード）による入力ができます。次の「記号」「英数字」「ひらがな」「カタカナ」「ギリシャ文字」「罫線」「第1水準」「第2水準」は、漢字を分野別に並べただけで、漢字コードの一覧表を細かく分けたものと考えてください。これでも記号やギリシャ文字の入力などには便利です。選択方法は通常の候補一覧モードと同様です。

「部首入力」や「画数入力」は、漢和辞典を引ける程度の知識がないときついかもしれませんが、漢字コード表が手元になくても読み方のわからない漢字を捜せるので、筆者は便利に利用しています。なお部首入力では、部首の画数を入力したあと、部首を除いた画数ではなく総画を入力します。これは漢和辞典と同じですね。

■語の登録

よく使う語が辞書にない場合は、登録することになります。登録には、語の始点（または終点）にマークをセットし、終点（または始点）でM-x toroku-region（〈ESC〉〈x〉tor〈TAB〉r〈TAB〉〈RET〉でOK）とするのですが、頻繁に登録する人は、

```
(global-set-key "\C-xt" 'toroku-region)
```

と~/.emacsに書いておけば、C-x tで登録できて便利です。このとき、まず辞書を選びます。辞書はユーザーアカウントごとに用意され、学習結果も個人別に記録されるので、個人的に使うのであればudで十分です。しかし、大学の研究室など共用辞書に専門用語を登録しておけば、他の人も恩恵を受けられます。この場合、一人でも品詞の分類に無頓着な人や、短縮語をがんがん登録してしまう人がいると、周りの人は迷惑をこうむります。

次に品詞を選ばなければならないのですが……品詞は大体、高校までの国語で習う国文法のとおりです。なかには国文法以上の細分化をしている品詞もありますが、変換の効率を上げるためにいろいろ試してみるといいでしょう。

間違って登録してしまった場合は、M-x edit-dict-itemとすると、dでマークをつけてxで削除することができます。

■特殊な記号

ところで記号を入力するには、C-^をタイプしたあと「記号入力」を選んで、候補一覧モードのようなミニバッファの表示から選んでもよいのですが、たいていの記号は読みでも検索できます。たとえば“△”という記号は、「さんかく」を変換すればいくつめかの候補に挙がっていることでしょう。ただしこれを多用すると、特殊文字の読み方が上位候補に出るようになって閉口します。“□”を「しかく」で変換した直後に「資格検定試験」を変換すると「□検定試験」になってしまう……（笑）。

一部の記号は、かな漢字変換デーモンに渡さずともローマ字かな変換の規則で出すことができます。このために使われるプリフィックス文字（最初にタイプする文字）はzです。もちろん〈z〉〈a〉は「ざ」に割り当てられているのが普通ですから、それ以外の組み合わせになります。

筆者が多用するのは、たとえば“…”（〈z〉〈.〉）や2重かぎカッコの“『”、“』”（〈z〉〈[〉、〈z〉〈]〉）、矢印の“←”“→”“↑”“↓”（〈z〉〈h〉、〈z〉〈l〉、〈z〉〈k〉、〈z〉〈j〉）などです——おや、これはviのカーソル移動キーの配列と一緒ですね（笑）。やはりviは押さえておけ、ということでしょうか。変わったところでは“〒”（〈z〉〈p〉）や“♀”“♂”（〈z〉〈0〉、〈z〉〈9〉）、さきほどの“□”も〈z〉〈4〉で出すことができます。ちょっと連想記憶法が入っています。

これらの特殊記号を含むローマ字かな変換規則は、自分でカスタマイズできるので、多用する記号は辞書に登録するのではなく、ローマ字かな変換で出すのもアイデアです。

Linux日記

第8回　名前解決（1）

Webやメールなどインターネットを利用しているとさまざまな「ドメイン名」を目にします。このドメイン名とは切っても切れない関係にあるのがDNSです。さて、今回からはDNSによる名前解決のお話です。

illustration : Aki

文：榊 正憲
Text : Masanori Sakaki

前回は誌面が足りなかったので、レゴの部品を説明したものの、どんなものかを見せられなかった。そこで、唐突ではあるが、次ページに写真を載せておこう。

Mind Stormsを買って2カ月以上経ったものの結局じっくりいじれないでいる。かつて購入したテクニックシリーズとともに、これもコレクションになってしまうのだろうか……。

さて、前回の原稿を書き上げた頃に、OCNが開通した。開通したとは、すなわち、128kbpsの常時接続データ回線でOCN側のルータと我が家のルータの間が接続され、我が家のDNSのアドレスがJPNICなどのネームサーバに登録されたということである。我が家のインターネット環境が完全に常時接続状態になったということではない。この時点では、とりあえずルータを接続し、ルータとOCN側のネームサーバの間でpingが通ったという程度である。その後、何台かのマシンを常時接続セグメントにつなぎ換えて、ちょこちょこっとルータやネームサーバの設定を行い、ぱらぱらとWebを眺めたり、FTPでファイルを拾ってこれる程度にはしたものの、ほかの仕事の締め切りがあったり、風邪をひいて寝込んだりといったことで、未だ完全開通には至っていない。いくつかの締め切りをクリアし、どうにかちょっと時間ができたので、今はsendmailと戯れている。今回の分の原稿が終わったら、少し腰を据えていろいろ設定しようと思っている。

と、この辺までは遅れながらも順調にいっていたのであるが、大問題が露見した。OCN用のサーバに割り当てた古いPentium 133MHzマシンが思いっきり2000年問題にひっかかっていたのである。我が家はいまだに486マシンが現役であるというくらい物持ちがいい。そんな状況で、今回のFreeBSDには、贅沢にもPentium 133MHzを奢ってやった。しかし、こいつのマザーボードが悪かった。リアルタイムクロック（RTC）がタコなのである。こいつがどうしても2000年になってくれない。BIOSセットアップで2000年と設定しても、それを覚えていてくれないのだ（ほかの古いボードは、多少問題があっても、BIOSセットアップで2000年に設定すればどうにかなった）。2000年に設定してBSDをインストールして、ふと気付いたらすべてのファイルのタイムスタンプが1994年に戻っていたのである。とりあえず、dateコマンドで正しい日付を設定すれば問題なく動くのだが、リブートしたら1994年に逆戻りだ。リブートのたびに日付設定を行えばいいのだが（まるで初代IBM-PCだ）、へたすると（うまく動いていれば）リブートなんて2年に一度だ。日付設定が必要だなんてことはすっかり忘れているだろう。今はまだこのマシンがテスト用サーバとして動いているが、結局、2000年であることを覚えていられるPentium 120MHzのマシンをサーバにすることにした（世間に6年遅れのメールを送るわけにはいかない）。

おかげで我が家のマシン構成の予定が狂ってしまい、実験機として新たに

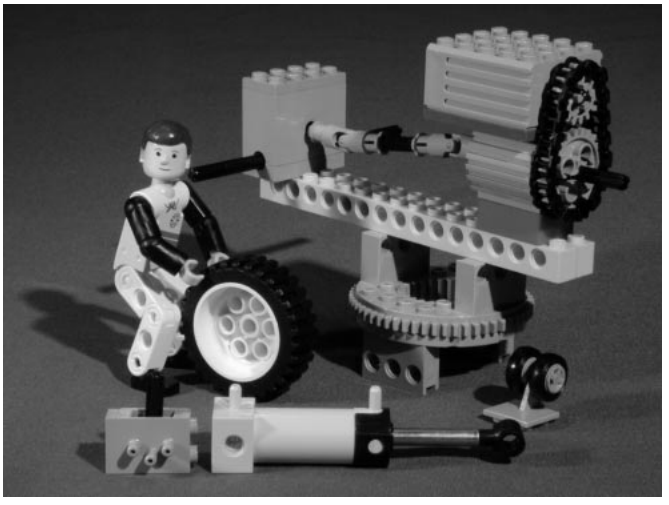


写真1　レゴの部品　その1

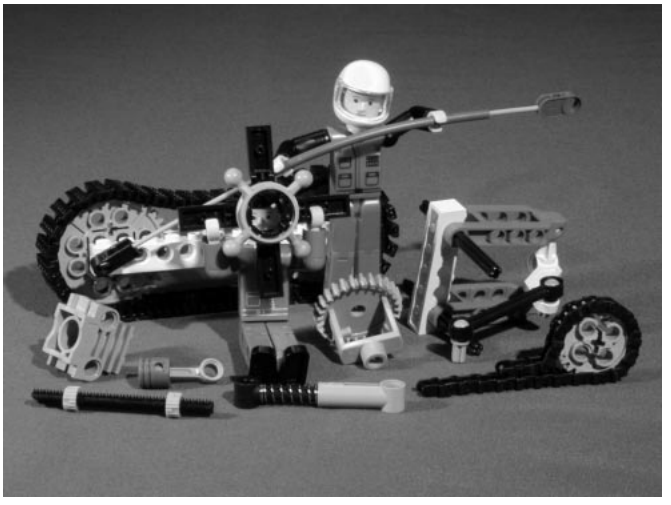


写真2　レゴの部品　その2

Pentium III 700MHzを導入した。予想外の痛い出費である（実際には、実売10万円以下のCeleronマシンで十分だったのだが、どうせ投資するからには、SETIのパフォーマンスを少しでも上げたかったのである。暮れに導入したPentiumIII500MHz×2の威力もあり、どうにか上位1パーセントに食い込めたのだが、この先が険しい。少しでも上位に上がるためには、とにかく速いマシンが必要なのだ）。

マシンを増やした結果、今度はディスプレイが足りなくなった。とりあえず、サーバなど、コンソールの使用頻度が低いマシンについては切り換え器でごまかしているが、画質がかなり劣化するので、近いうちに新しいディスプレイを導入することになるかもしれない。筆者の性格からして、どうせ買うならということで、きっと20インチクラスのものを購入するに違いないのだ。問題は、それをどこに置くかということだが、今までも、もうこれ以上マシンやディスプレイは増やせないと思いつつ、なんだかんだいってどうにか収めてしまえたので、新しいディスプレイもどうにかなるだろうと思っている（おかげで仕事部屋は、電源ケーブルの嵐である。最後に数えた時は、AC100Vに接続する家電製品、電子機器が60台弱あった。100台を越えるのは時間の問題だろう。ネットワークなどの弱電ケーブルの数なんか、見当もつかない）。

RTCがタコなおかげでOCN環境がどーのこーのという話を編集部でしていたら、編集担当のA氏が、「まだLinux magazineではDNSの話を載せたことがないんで、DNSについて書いてくれませんか？」というのであった。うちのDNSは、前にも書いたように、FreeBSDの上で動いている。『Linux日記』というタイトルの記事で、BSDの上で動くDNSの話も何だなぁとは思ったのであるが、まぁ、ネームサーバ自体はBSDでもLinuxでも違わないし、ほかのネタも思い付かないし、それでいくかと思ったのであった（数回前の記事で、インターネット接続の話は当分しないといったのに、この体たらくである）。

■DNSの役割と必要性

DNSとは、Domain Name Systemの略語である。一言でいってしまえば、インターネットドメイン名からIPアドレスを調べる名前解決システムである（XXを評価して、対応するYYを求めることを、XXを解決（Resolution）するという）。プロバイダにダイヤルアップ接続の契約をしたときに指定される

Column

インフルエンザかもしれない

咳がひどくなり、熱が上がり始めたころに医者に行ったら、「インフルエンザかもしれない」といわれた。うちの奥さんもその数日前に同じ医者から同じことをいわれたそうだ。インフルエンザの流行のおかげで、ワクチンどころか、検査薬まで底を尽いているらしく、正確な診断もできないらしい。結局、抗生物質と解熱剤、咳止めなどの薬をもらっただけである。

そうこうしているうちに、熱がどんどんあがり、最高で39.6度までいった。さすがに、40歳を越えてこの体温は辛いものであった（でも、「おおすごい」とも思ったのであった）。しかし、そんな熱の合間に子供の幼稚園の学芸会に、30kgあまりのビデオ機材を抱えて撮影に行ったのだから、実はそんなにひどくなかったのかもしれない。

いくつかのパラメータのひとつに、「ネームサーバのIPアドレス」という項目があるが、そのネームサーバの話だ。

ネームサーバという名称が出てくることからもわかるように、DNSはサーバ／クライアントシステムである。名前の解決を求める各コンピュータ（クライアント）は、ネームサーバに問い合わせを送り、名前に対応するIPアドレスを取得するというシステムだ。ネームサーバは、全世界に広がるインターネットの莫大な数のホストの名前解決を単独で行うことはできないので、ほかのネームサーバにも問い合わせて名前解決を行う。つまりネームサーバは、分散データベースサーバなのである。

さて、個人でちょこちょこっとインターネットにつなぐという時に、ネームサーバ自体を自分で設定する必要があるかというと、もちろんそんなことはない。自分のコンピュータに、プロバイダのネームサーバのアドレスを登録するだけでいい。では、ちょっとした規模の組織や会社を常時接続する場合にネームサーバが必要かというと、やはりそんなことはない。最近は、各種インターネットサーバをプロバイダ側が適当に用意してくれるからだ（自前で用意すると便利なこともあるが）。従って、かなり大規模な組織であるとか、自分でプロバイダみたいなことをやるとか、あるいは単にサーバマニアであるといったユーザー以外は、DNSの仕組みなんて知らなくてもまるで困らない。

さて、運悪くどうしてもネームサーバをいじらなければならない、あるいは趣味でサーバを立ち上げたい場合はどうすればいいか。簡単だ。オライリー・ジャパンから発行されている『DNS＆BIND 第3版』を読めばいい。バッタの表紙の本だ。必要なことはすべてこの本に書かれている。悪いことは言わない。実際にネームサーバの設定を行わなければならないのであれば、ケチケチしないでこの本を買うべきだ。ネームサーバを、Windows NTなどの非UNIX環境で動かす時でもこの本は役に立つ。筆者は以前にDNSの設定をしたときに、この本の最初の版を買って読んだ。そして今回、ネームサーバのバージョンが上がり、セキュリティ機能の追加とか設定ファイルの記述の変化など、かなり中身が変わったので、改めて第3版を買った。初版を読んでいたので、今回は拾い読みであるが、それでも役に立ったことは確かだ。

『DNS＆BIND 第3版』の紹介で今回の記事をおしまいにしてもいいのだが（いつの世もポインタ渡しは効率的だ）、それでは編集氏がいい顔をしないだろうから、もう少し書かねばなるまい。しかし最初にはっきり断っておく。これから書くことは、あくまでも教養というか、雑学としてのDNSの知識である。この記事だけ読んでネームサーバの設定をやろうなんて考えないこと。

■インターネットドメイン名

インターネットドメイン名は、日ごろWeb閲覧で使っているURLや、電子メールアドレスなどに使われている。たとえば“ http://www.ascii.co.jp/index.html ”といったURLの“ www.ascii.co.jp ”の部分、“ linuxmag@ascii.co.jp ”といっ

Column

今風のやり方

小規模な組織が小規模なネットワークをインターネットに接続し、電子メールを使ったり、Web閲覧を行ったり、あるいはWebサーバを用意したいといった場合、今や自前で各種サーバを用意する必要はない。必要なサーバは全部プロバイダが用意してくれる。

かつては、組織のネットワークをインターネットにつなぐということは、独自のドメイン名を登録し、グローバルIPアドレスの割り当てを受け、接続先を探し、その組織のドメインのためのネームサーバ、メールサーバ（必要であれば、WebやFTPのサーバなども）を準備するということであった（今でも必要な手続きはいくつかあるが、ほとんどプロバイダが代行してくれる）。このような環境を構築するためには、どうしてもシステムを熟知した管理者が必要になる。

以前は、インターネットにつなぐということ自体、関連業界の会社だとか大学とか研究所とかしか行っていなかったので、このような人材に困ることはなかった。今は、そんなことができる社員がいる会社のほうが珍しい。

インターネットに常時接続し、組織を表すドメイン名を使って電子メールを使ったり、Webサーバを用意するということの敷居を低くするために、多くのプロバイダは、管理サイドの業務を代行するサービスを提供している。これにより、“ ユーザー名@会社名.co.jp ”というメールアドレスを使用し、“ www.会社名.co.jp ”というWebサーバを運用できるようになった。実際には、会社側にサーバはないし、管理者もほとんど不要である（メールアドレスの登録を行うとか、Webコンテンツを作成するといった業務はあるが）。また、サーバがプロバイダ側に置かれることで、Webサーバが大容量回線で接続されるといったメリットがある。

どうやれば、プロバイダ側でこのようなサービスを提供できるかを理解するためには、DNSやメールサービスの仕組みを詳しく知る必要がある。ちゃんとしたプロバイダには、やはり秘儀を習得できるだけの管理者が必要なのである。

たメールアドレスの「ascii.co.jp」という部分がインターネットドメイン名である。

インターネットドメイン名は、階層化された名前システムで、ピリオドで区切られたいくつかの名前から構成される。ファイルシステムでもドメイン名でも、階層化することによって、比較的簡単に莫大な数の要素の中から特定のものを識別できるようになる。また、途中のパスが異なれば（すなわち、別のドメインであれば）、同じ名前のホストを使うことができる。どこの組織でも“www”というホスト名が使えるのは、それを含むドメイン名が異なるものだからだ。ドメイン名は、階層化ファイルシステムのパス名とは異なり、右に行くほど上位の名前である。“www.ascii.co.jp”というドメイン名であれば、右から順に、日本、会社組織、アスキー、ホスト名という形になっている。

一方、“www.microsoft.com”というように、最上位に国名ではなく、組織種別が付くドメイン名もある。これは、インターネット（正確にはその前身）がアメリカでしか使用されていなかった時代からあるドメイン名である。一貫性に欠けるように思えるかもしれないが、国際的な企業などにとっては便利であるし、何よりも短いドメイン名を取得した組織がその名前を手放すとは思えないから、今後も組織別、国別の2本立ての状態が続くことだろう。

そういうわけで、最上位に使われるドメイン名は、組織種別と国別のものがある。組織別のドメイン名と、日本を表すjpドメインの下位レベルドメインを**表1**に示しておく（jpドメインには、これとは別に、地名に基づいたサブドメインもある）。組織別ドメイン名のうち、arpaは特殊なものである。これについては次号以降で解説する。

jpドメインがどのように管理されているかについては、http://www.nic.ad.jp/を見てほしい。簡単に説明すると、goは政府関係の組織、coは企業法人、orは会社以外の各種団体、法人である。プロバイダはneかadである。adはJPNICの会員か、日本国内のインターネット運用において重要な組織と認められた法人、団体（管理組織）である。grは2名以上から構成される任意の団体で、クラブや家族などでも取れる（orのほうが条件が厳しい）。acはおおむね大学以上の高等研究機関、edは小中学校などの初等、中等教育機関である。

プロバイダの個人加入者のメールやホームページのアドレスは、たいていはプロバイダのサブドメイン名（“user@XX.provider.ne.jp”とか“http://www.provider.ne.jp/~user/”など）を使っているが、プロバイダによっては、or.jpの名前を使っている場合がある。これは歴史的経緯によるものだ。neドメインは比較的新しい種別で、これが制定される以前は、プロバイダの契約者は、プロバイダ契約者の団体としてor.jpドメインを使っていた。その後、neドメインの制定により、プロバイダの契約者はneドメインを使うように推奨され、多くのプロバイダのユーザーのアドレスが変更された。しかし、メールアドレスが変わるというのは、引越しのようなものである。過去のしがらみを捨てるにはいいが、そういった事情がない場合はうっとうしいことだ。機械的にorがneに変わるだけとはいえ、仕事などで頻繁にメールを使うユーザーにとっては面倒なことであった。また、メーリングリストの管理者なども、ユーザーアドレスの変更などで結構な手間がかかった。現在or.jpを使っているプロバイダは、この時にneに変更しなかった業者である。管理上はneが好ましいものの、ユーザーの利便を取ったという形である（筆者のメールアドレスは今でもor.jpで、neへの変更騒ぎの影響は受けなかった。IIJがneに変更しなかったおかげだ）。

■ドメイン名は何を表すのか？

最初に説明したように、インターネットドメイン名サービスの最大の目的は、ドメイン名から特定のホストのIPアドレスを求めることである。IPアドレスがわかることで、さまざまなアプ

Column

一子相伝の秘儀

今は、UNIXシステム管理に関する解説書は、オライリーを始めとして多数出版されている。安くない書籍ではあるが、詳細な解説を日本語で読むことができる。ありがたいことである。今ほど解説書が充実していなかった頃（というか、そんなものがなかった頃）、UNIXのシステム管理者は、man、docなどの英語のドキュメントを読み、システムの設定を行わなければならなかった。わからないことは自分でドキュメントやソースを読んで調べるか、知り合いの管理者に聞かなければならなかった。

それでも、DNSの設定はまだ簡単な部類の作業だった。最難関だったのが、sendmailの設定である。sendmailの設定ファイルであるsendmail.cfの記述はほとんど誰にもわからず、代々の管理者の間で、一子相伝で伝えられる秘儀のようにいわれていたものである。今では解説書（コウモリの本）も出ているが、その内容はやはり一子相伝級の複雑さである。筆者は、sendmailの解説記事だけは書きたくないと思っている。

編：そうですか…ふふふ。

リケーションがインターネットを使って通信を行うことができる。たとえばネームサーバに“www.ascii.co.jp”という名前を問い合わせると、アスキーのWebサーバホストのIPアドレスが得られる。ブラウザはこのIPアドレスを使って、Webサーバと通信を行う。Webサーバに限らず、FTPサーバ、データベースサーバなど、みな同じようにドメイン名からIPアドレスを求め、ネットワークコネクションを確立する。

では、ドメイン名はインターネット上のホストを識別する名前といい切ってしまっていいのだろうか？

メールアドレスを考えてみよう。たとえば、あるプロバイダと契約すると、“user@XXX.provider.ne.jp”というメールアドレスが使えるようになるとしよう。この時、“XXX”は適当な名前である。同じプロバイダでも、地域ごとに変わるとか、契約内容によって変わるといったことがあるだろう。ユーザーは、自身のメールクライアントで、SMTPサーバ、POPサーバに“XXX.provider.ne.jp”というホストを指定する。この時、“XXX.provider.ne.jp”は、メール送受のためのメールサーバのホスト名である。ユーザー宛てのメールはこのホストでスプールされ、随時クライアント側にダウンロードされる。そしてユーザーが送ったメールは、まずこのホストに中継され、その後宛先のメールサーバに転送されることになる（**図1**）。これが、基本的なインターネット電子メールの送受の仕組みである。

会社など、独自のドメイン名を持っている組織のユーザーのメールアドレスは、これとはちょっと違う形式になっていることが多い。たとえばアスキーの社員のメールアドレスは、“user@ascii.co.jp”という形式である。先ほどのメール送受の仕組みからすると、このようなアドレスを有効なものとするためには、“ascii.co.jp”という名前のメールサーバが必要になる。しかし実際には、“mail0.ascii.co.jp”（仮名）といったメールサーバが存在し、メールの送受はこのサーバが受け持つようになっている。どのような仕組みにより、このようなメールアドレスが使えるようになるかは、あとで説明する（そう、これはネームサーバが実現している仕組みなのだ）。

そもそも“ascii”という名前以前に、“co”だとか“jp”といった名前がドメイン名の要素として存在していることを考えれば、ドメイン名が、常に対応するIPアドレスを持っているわけではないということがわかるだろう。

ここで重要な点は、インターネットドメイン名は、単にインターネット上のホストを識別するだけでなく、組織や、組織内のサブグループなどの識別にも使用できるという点である。ホストのIPアドレス以外の情報をネームサーバで調べた時に、どのような情報が得られるかということについては、あとで解説する。

同じように階層的な名前を使っているファイルシステムとインターネットドメイン名で、もっとも異なっているのがこの部分だろう。階層化ファイルシステムでは、すべての名前要素はファイルに対応している（ディレクトリもファイルである）。同じように考えると、すべてのドメイン名は特定のホスト（IPアドレスを持つ）に対応し、そして中間ドメイン名に対応するホスト（ファイルシステムにおけるディレクトリに相当するもの）は、必ずネームサーバでなければならない（ネームサーバは、その下位にサブドメインを定義できる）ということになる。だが実際には、DNSはそんな単純な仕組みではない。

ドメイン名	種別
arpa	逆引き用ドメイン
com	営利組織、個人、任意のグループ
edu	教育機関*
gov	政府機関*
int	国際機関
mil	軍組織*
net	ネットワークサービスを提供する組織（プロバイダや管理組織など）
org	非営利団体

*はアメリカの組織しか使用できない。

表1　組織別トップレベルドメイン名

サブドメイン名	種別
ac	大学、研究機関
ad	管理組織
co	企業
ed	教育機関
go	政府機関
gr	任意団体
ne	ネットワークサービス組織
or	各種団体

表2　jpドメインの組織種別

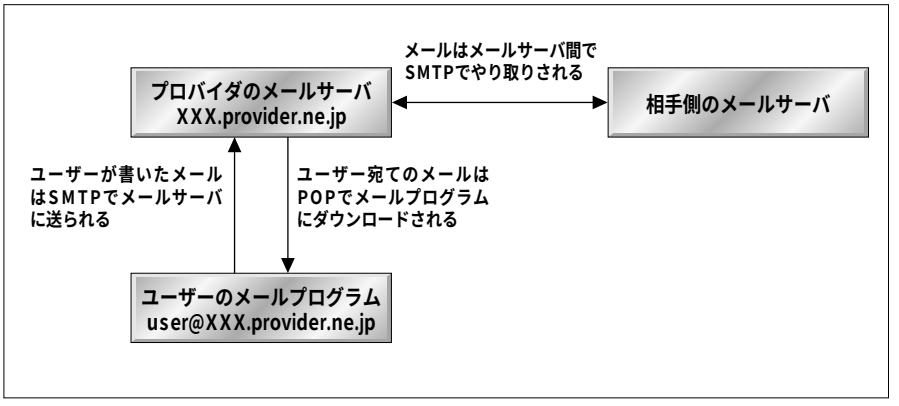


図1　メールの送受信

ではどうなっているかという話をこれからしていこう。DNSを理解していけば、なぜこのようになっているのかといったことがわかってくるだろう。ファイルシステムと似ているのは名前の構造だけであり、実装はまったく異なっているのである。

■ルートドメイン

ドメイン名は階層的な名前である。そして階層構造にはルートがつきものである。UNIXファイルシステムであれば、すべてのツリーはルートディレクトリの下に構成される。そして、ルートディレクトリには名前がない。1つしかないものなら、名前を付ける必要はないからだ。インターネットドメイン名の場合も同様である。もっとも上位にルートドメインがあり、それには名前がついていない。“ jp ”や“ com ”などは、ルートドメインの下位に位置する名前なのである。しかし、実際のところ、“ jp ”や“ com ”が最上位のドメイン名として扱われることが多い（確かに、名前を持っているドメイン名としては最上位である）。そのため、“ jp ”や“ com ”などは、トップレベルドメイン名と呼ばれる。単にネームサーバを使うだけであれば、ルートとトップの違いなどどうでもいいことなのだが、実際にネームサーバを設定するとなると、トップレベルドメインの上位に名前のないルートドメインがあるという点が重要になってくる。

階層化ファイルシステムに、相対指定、絶対指定のパス名があるように、ドメイン名にも絶対指定、相対指定がある（とはいっても、ファイルシステムのように、相対指定で親までさかのぼることはないが）。たとえば“ www.ascii.co.jp ”というドメイン名は絶対指定である。また、同一ドメイン内のホストにtelnetやFTPでつなぐという場合であれば、“ db-server ”といった名前だけ指定して“ db-server.ascii.co.jp ”に接続することができる。これは、着目しているドメイン名（たとえば、そのマシンが所属しているドメイン名）に対する相対指定と見ることができる。ネームサーバの設定を行う際は、こういった点の区別が重要だ。ネームサーバの設定では、相対指定と絶対指定を厳密に区別している。そのため、絶対指定の場合は、“ ascii.co.jp. ”というように、最後にピリオドを付ける。ファイルパスの絶対指定が、“ /usr/bin/vi ”というように、名前のないルートからのパスを列挙する（すなわち、パスが“ / ”で始まる）のと同じである。名前のないルートドメインまで列挙することにより、最後にピリオドがくる形式になる。この形式のドメイン名をFQDN（Fully Qualified Domain Name、完全限定ドメイン名）という。最後にピリオドがない形式は相対指定とみなされる。たとえば“ ascii.co.jp. ”ドメインに着目しているときに、“ www ”と指定すれば、“ www.ascii.co.jp. ”というドメイン名として解釈される。同じ状況で、“ www.microsoft.com ”というドメイン名を指定するとどうなるか。普通に考えればこれは絶対指定のように見えるが（なぜなら、最上位ドメイン名まで含んでいるからだ）、ネームサーバの設定という状況においてはそうはならない。“ www.microsoft.com.ascii.co.jp. ”というドメイン名になってしまうのである。本来の指定を行いたいのであれば、“ www.microsoft.com ”ではなく、“ www.microsoft.com. ”とFQDNで指定しなければならない。

■さて次回は

ここまで書いたところで誌面が尽きてしまった。次回は、実際にネームサーバと対話して、どのように名前を解決しているのかを見ていく。その後、UNIXで使われている標準的なDNSの実装であるBINDの紹介、設定について説明していくつもりだが、たぶん次回では終わらないだろう。

Column

ホスト

インターネットについて解説する際には、接続されているコンピュータをホストと呼ぶことが多い。たとえば、UNIXでコンピュータの名前を設定、参照するコマンドはhostnameである。なぜホストなのか？

ホストコンピュータという用語は、大型コンピュータの時代によく使われていた。多くのユーザーに同時にサービスを提供するTSS（Time Sharing System、時分割システム）環境などを運用していた時代である。多くのお客にサービスを提供するといった意味で、ホストだったのである。当初UNIXもこのような使い方が多かった。大きなコンピュータでUNIXが動作し、多数のユーザーが、シリアル回線を使ってそれに端末を接続し、マルチユーザーオペレーティングシステムとして使っていたのである。このようなUNIXマシンは、明らかにホストコンピュータである。その名前を指定するコマンドがhostnameであることに何の不思議もない。

インターネットの初期の時代は、このような運用形態のコンピュータが相互に結ばれていた。パソコンやワークステーションなど、個人ベースのコンピュータが主流になる以前のことである。結局、パソコン、ワークステーション、サーバといった分散環境に移行した後も、ホストという名前だけは残っている。ダイヤルアップでインターネットにつないだマシンで、他人に何もサービスを提供していなくても、それはインターネット上のホストコンピュータなのである。

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(後編)―

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。今回は、前回に引き続いてbashのシェルスクリプトについて取り上げ、よりプログラム的なスクリプトを書くために、スクリプト内での数値演算や配列、関数などの機能について解説する。

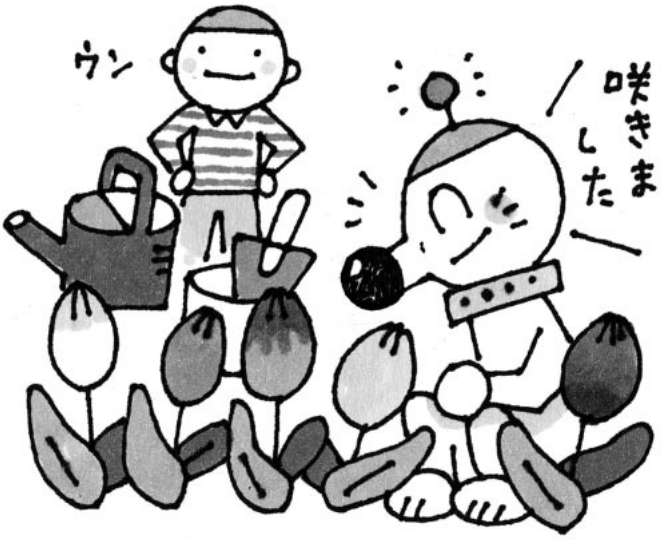


Illustration : Manami Kato

今月のお題

ファイルをキーワードを使って暗号化・復号化するスクリプトを作成する

文：**大池 浩一**
Text : Kouichi Ooike

今回は、前回に引き続いてbashの「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。単に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といった制御構文を駆使して、より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。

bashは、数値演算(ただし整数のみ)を外部コマンドexprを呼び出すことなく行ったり、スクリプト内で関数を定義して呼び出せるなど、プログラム的なスクリプトを書くのに適した特性を持っている。さらに、bash2.0以降では、シェル変数の配列などの機能強化が行われた。今回は、これらの機能を解説し、より複雑なスクリプトの作成に挑戦してみよう。

数値演算、配列、関数を扱う

C言語などのプログラミングの経験がある人がシェルスクリプトを書く際に陥りがちなワナが、「シェル変数の演算は文字列ベースである」ことをうっかり忘れてしまうことだ。たとえば、以下のシェルスクリプトの出力はどのようなものになるだろうか。

```
1: #!/bin/bash
2: i=0
3: while [ $i != 10 ]; do
4:   echo $i
5:   i=$i+1
6: done
```

一見、これは「シェル変数iの値が0～9の間、iの値を1つずつ増やして表示する」スクリプトのように思われる。5行目の「i=$i+1」で、シェル変数iの値を1つずつ増やしていき、10になった時点で[…]内の条件式が成立しなくなっ

てループを抜けるはずだ。

●スクリプトで数値演算を行う

それでは、実際の結果をお見せしよう(**画面1**)。このように、「0+1+1+…」という文字列が表示され、Ctrl-Cで中断するまで停止しない。このような結果を生んだ原因は、シェル変数iの値に1を加えるつもりで「$i+1」と書いてしまったことにある。

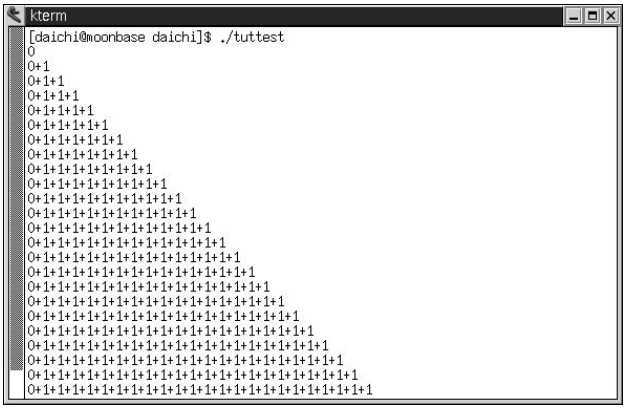
シェル変数の演算は文字列ベースなので、「$i+1」はiの値と「+1」という文字列を並べた文字列にほかならない。つまり、実際のiの値は「0」から始まり、「0+1」「0+1+1」「0+1+1+1」…という文字列になる。これらは、いずれも「10」と等しくないので、[…]内の条件式は永久に成立しつづける。

「シェル変数iの値が0～9の間、iの値を1つずつ増やして表示する」という処理を正しく行なうスクリプトを以下に示す。違っているのは問題の5行目だけだ。

```
1: #!/bin/bash
2: i=0
3: while [ $i != 10 ]; do
4:   echo $i
5:   i=$((i+1))
6: done
```

bashは、$((…))の内部を数値演算式と見なし、$((…))全体を演算結果で置換する。上の例では、$((i+1))が「シェル変数iの値に1を加えた数値」で置換される。よって、iの値は初期値0から始まり、1、2、3、…という数値になる。10になった時点で[…]内の条件式が成立しなくなってループから抜けるので、それまで0～9の数値が1行に1つずつ出力されることになる。

bashには、C言語と同様の数値演算子(**表1**)が用意されており、ひととおりの整数演算が可能だ。演算子の優先順位や組み合わせ、カッコの使用方法などもC言語と同じだ。なお、数値演算式の内部では変数の参照に「$」をつける必要はなく、余分なスペースは無視される。たとえば、「$(($i+1))」「$((i+1))」「$((i + 1))」はすべて同じ結果になる。また、変数の参照やコマンド置換と同様に、二重引用符「"」で囲まれた(クォーティングされた)文字列の内部でも$((…))を使用可能だ。



画面1 文字列が永久に出力され続ける

+	加算
-	減算
*	乗算
/	除算（小数点以下切り捨て）
%	剰余
<<	左ビットシフト
>>	右ビットシフト
&	論理積（AND）
|	論理和（OR）
~	論理否定（NOT）
!	（同上）
^	排他的論理和（XOR）

表1 数値演算式で利用可能な数値演算子

●ifやwhileの条件部で数値演算式を使う

「変数の値がある数値より大きい(あるいは小さい)」といった数値の比較をifやwhileで行う場合、数値演算式を直接指定する((…))制御構造を使うといい(bash2.0以降でサポート)。C言語と同じ関係演算子(**表2**)を利用して、直感的な表現で条件部を記述できる。

たとえば、「変数iの値が10より小さい」という条件は「((i < 10))」と書け、「変数iの値が10以上で、かつ20より小さい」という条件は「((i >= 10 && i < 20))」と書ける。[…]制御構造と同様に、「((」の後と「))」の前にはスペースが必要だ。

((…))制御構造の終了ステータスは、指定した条件が成立する場合は0(正常終了)、成立しない場合は1(異常終了)になる。これはC言語や数値演算式で使われる真偽値(1を真、0を偽とする)とは逆なので気をつけよう。

((…))制御構造を使って、「シェル変数iの値が0～9の間、iの値を1つずつ増やして表示する」スクリプトを書きなおすと、以下のようになる。

<	より小さい
>	より大きい
<=	以下
>=	以上
==	等しい
!=	等しくない
&&	論理積
||	論理和

表2 数値演算式で利用可能な関係演算子

```
1: #!/bin/bash
2: i=0
3: while (( i < 10 )); do
4:   echo $i
5:   i=$((i+1))
6: done
```

●常に数値として扱われる整数型変数

シェル変数の属性を「整数型変数」に変更すると、文字列ではなく整数値として処理されるようになる。属性変更には、組み込みコマンドdeclareの-iオプションを使用する。対象となる変数名を指定して「declare -i 変数名」とすればいい。「declare -i 変数名＝値」という書式で、属性変更と同時に値を代入することも可能だ。

整数型変数に対する代入は、$((…))をつけなくても数値演算式と見なされる。たとえば、整数型変数iに対して「i=i+1」とすると、「i+1」という文字列ではなく、「iの値に1を加えた数値」が代入される。

「シェル変数iの値が0～9の間、iの値を1つずつ増やして表示する」スクリプトを書きなおしてみよう。iを整数型変数に属性変更することで(2行目)、iの値を1ずつ増やす記述(5行目)が簡潔になる。

```
1: #!/bin/bash
2: declare -i i=0
3: while (( i < 10 )); do
4:   echo $i
5:   i=i+1
6: done
```

●組み込みコマンドreadを使う

bashには、標準入力の内容を1行ずつ読みこむ組み込みコマンドreadが用意されている。readの書式は「read [-r] 変数名…」で、標準入力の内容を1行ずつ読み込み、ワードに分割して、引数で指定したシェル変数に順番に格納する。指定された変数の数がワードより少ない場合は、残りの内容を最後の変数にまとめて格納する。なお、-rオプションを付けると、行末の「\」による継続行処理が無効になる。

たとえば、標準入力の内容をそのまま出力するスクリプトは次のようになる。

```
1: #!/bin/bash
2: IFS=""
3: while read -r line; do
4:   echo "$line"
5: done
```

IFSは入力行をワードに分割する区切り文字を保持する環境変数で、初期値は「スペース・タブ・改行」だ。初期値のままだと、行頭の空白文字をreadが取り除いてしまうため、readを実行する前に空文字列に変更する(2行目)。

続いて、whileの条件部では、readで読み込んだ行の内容がシェル変数lineにまとめて格納され(3行目)、echoでそのまま出力される(4行目)。読み込む行がなくなるとreadは終了ステータス1を返し、whileループを抜ける。

次に、標準入力の内容の行頭に行番号と「: 」を付けて表示するスクリプトを示す。

```
1: #!/bin/bash
2: IFS=""
3: declare -i num=1
4: while read -r line; do
5:   echo "$num: $line"
6:   num=num+1
7: done
```

基本的な構造は上のスクリプトと同じなので、異なる部分だけ説明しよう。行番号を格納する整数型変数numに初期値1を設定し(3行目)、行の内容($line)の前に行番号($num)と「: 」をつけて出力する(5行目)。その後、行番号の値を1ずつ増やしている(6行目)。

●配列を使う

bash2.x以降では、シェル変数の「配列」も利用できる。配列の要素は「変数名[インデックス]」と表記する。インデックスは0以上の整数だ。

通常のシェル変数と同様に、「変数名[インデックス]=値」として各要素に個別に値を代入できるほか、「変数名 =(値 値…)」とすることで、複数の要素に値をまとめて代入することも可能だ。たとえば、「hoge=(あうー こくこく フッ)」とすると、hoge[0]に「あうー」、hoge[1]に「こくこく」、hoge[2]に「フッ」がそれぞれ代入され、その他の要素の内容はすべて削除される。

配列の要素を参照する際は、「${変数名[インデックス]}」と中カッコで囲む点に注意しよう。たとえば、hoge[0]の値をechoで出力するには、「echo ${hoge[0]}」とすればいい。このほか、特殊なインデックス「@」により、要素の値すべての並びは「${変数名[@]}」、要素数は「${#変数名[@]}」でそれぞれ参照できる。

それでは、配列を使って、標準入力を行単位で逆順に(最後の行からさかのぼって最初の行まで)表示するスクリプトを作ってみよう。

```
 1: #!/bin/bash
 2: IFS=""
 3: declare -i num=1
 4: while read -r line; do
 5:   lines[num]="$line"
 6:   num=num+1
 7: done
 8: num=num-1
 9: while (( num > 0 )); do
10:   echo "${lines[num]}"
11:   num=num-1
12: done
```

4～7行目では、読み込んだ行の内容(line)を、行番号(num)をインデックスとして、配列(lines)の要素に格納する。最後の行まで読み込んだら、9～12行目で行番号を減らしながらインデックスに指定して、配列の要素に格納された内容を出力する。

●関数の定義と利用

bashには、スクリプトによく似た機能を持つ「関数」も用意されている。内容の定義がメモリ内に格納されるため、呼び出しが高速なのが特長だ。また、スクリプト内で同じような処理が繰り返される場合、処理を行なう関数をスクリプト内で定義すると見通しが良くなる。

関数の定義は、「function 関数名 { コマンド... }」という書式で行う。コマンド部分には、ifなどの制御構造や組み込みコマンド、外部コマンドを記述できる。実行時の引数は$1、$2、…で参照可能だ。呼び出した側に処理結果を返すにはシェル変数を利用する。

たとえば、さきほどの逆順表示スクリプトを関数で記述してみよう。

```
 1: function reverse
 2: {
 3:   local num line lines ifs
 4:   ifs="$IFS"
 5:   IFS=""
 6:   declare -i num=1
 7:   while read -r line; do
 8:     lines[num]="$line"
 9:     num=num+1
10:   done
11:   num=num-1
12:   while (( num > 0 )); do
13:     echo "${lines[num]}"
14:     num=num-1
15:   done
16:   IFS="$ifs"
17: }
```

中心となる処理はスクリプトと同じだ(5～15行目)。その前に、関数内でのみ利用する変数num、line、lines、ifsを、関数外に影響を与えないローカル変数に指定している(3行目)。また、IFSの内容はifsに保存して(4行目)、関数を終了する際に元に戻している(16行目)。

現在のシェルで関数を定義するには、コマンドラインで1行ずつ入力するか、関数を定義したファイルを組み込みコマンドsourceの引数に指定する。また、「declare -f」とすると、現在のシェルで定義されている関数の一覧が定義内容とともに表示される。

関数の実行は、単に「関数名」とすればいい。たとえば、上の関数を定義してから「reverse < hoge」とすると、hogeの内容が逆順に表示される。

今月のお題

☑ ファイルをキーワードを使って暗号化・復号化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、

ファイルをキーワードを使って暗号化・復号化するスクリプトを作成する

というもの。ファイルの内容とキーワードをXOR(排他的論理和)するだけの簡単な暗号だが、キーワードが長いほど破られにくい。暗号化と復号化の流れ(イラスト)を簡単に解説し、必要な処理を実現していこう。

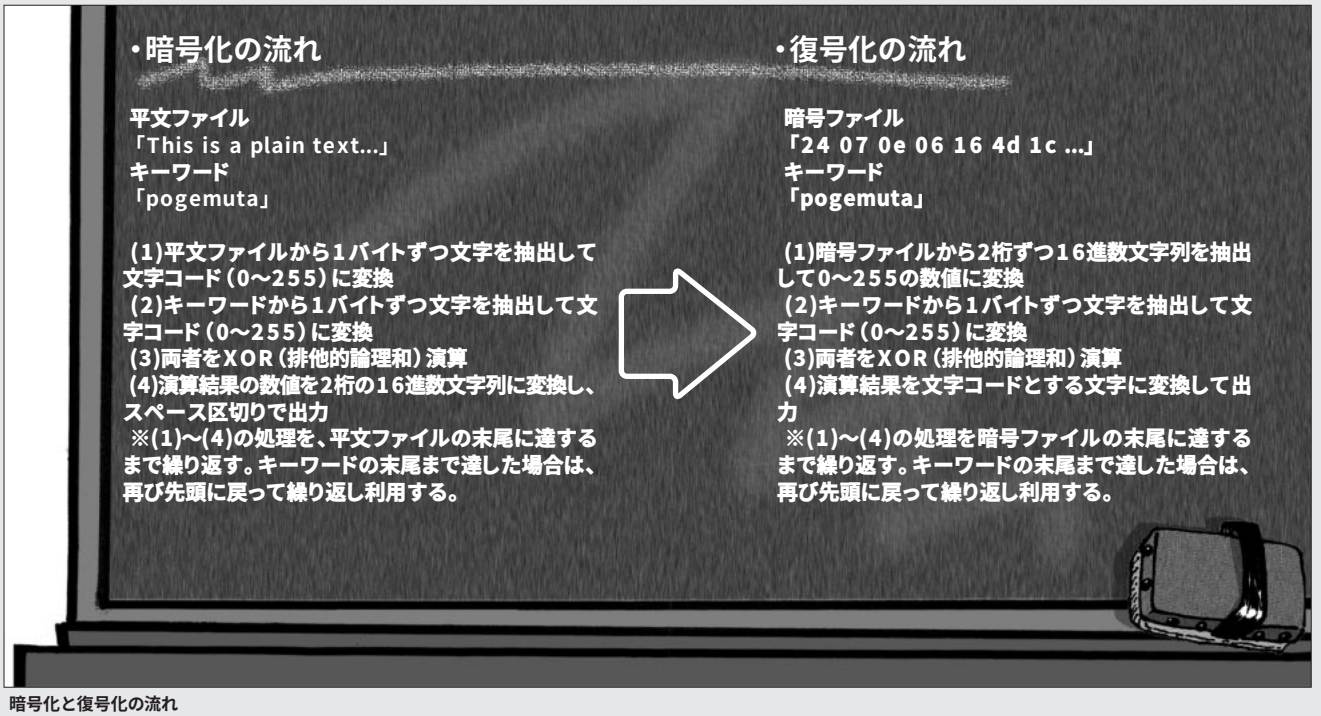
暗号化のスクリプトでは、平文ファイルとキーワードからそれぞれ文字を1バイトずつ取り出して文字コード(0～255)に変換し、両者をXOR(排他的論理和)演算する。演算結果は0～255の数値になるので、それを2桁の16進数文字列に変換し、スペース区切りで出力する。以上の処理を平文ファイルの末尾に達するまで繰り返す。なお、キーワードは繰り返し利用する。

復号化のスクリプトでは、暗号ファイルから2桁の16進数文字列を取り出して0～255の数値に変換し、キーワードから文字を1バイトずつ取り出して文字コードに変換した数値とXOR演算する。演算結果は0～255の数値になるので、それを文字コードとする文字に変換して出力する。この処理を暗号ファイルの末尾に達するまで繰り返す。キーワードを繰り返し利用する点は暗号化と同様だ。

以下では、

- **ファイルから1バイトずつ文字を抽出**
- **キーワードから1バイトずつ文字を抽出**
- **文字を文字コードに変換する関数**
- **文字コードから文字に変換して出力する関数**
- **数値を16進数文字列に変換して出力する関数**
- **16進数文字列を数値に変換する関数**
- **起動時オプションの処理**

といった処理を実現するスクリプトを作成し、最終的にはこれらを組み合わせて、暗号化・復号化を行なうスクリプト「mycrypt」を完成させる。



暗号化と復号化の流れ

●ファイルから1バイトずつ文字を抽出する

ファイルから1バイトずつ文字を抽出するには、readを使って各行の内容をシェル変数に読み込み、サブ文字列展開(${変数名:オフセット:バイト数})を利用して、シェル変数の先頭から1バイトずつ切り出せばいい。

具体的なスクリプトを以下に示す。

```
 1: #!/bin/bash
 2: IFS=""
 3: while read -r line; do
 4:   while [ -n "$line" ]; do
 5:     c="${line:0:1}"
 6:     line="${line:1}"
 7:     echo -n "[$c]"
 8:   done
 9:   echo -ne "[\n]"
10: done
```

内側のwhileループでは、行の内容が格納されたlineから、サブ文字列展開により先頭1バイトの文字を抽出してcに格納し(5行目)、lineからそれを取り除く(6行目)。本来なら暗号化を行なう部分は、単に抽出した文字を大カッコで囲んで出力するだけだ。これらの処理を、lineが空文字列になるまで繰り返す。なお、readは行末の改行を取り去るので、内側のwhileループを抜けた9行目では、改行を大カッコで囲んで出力する。

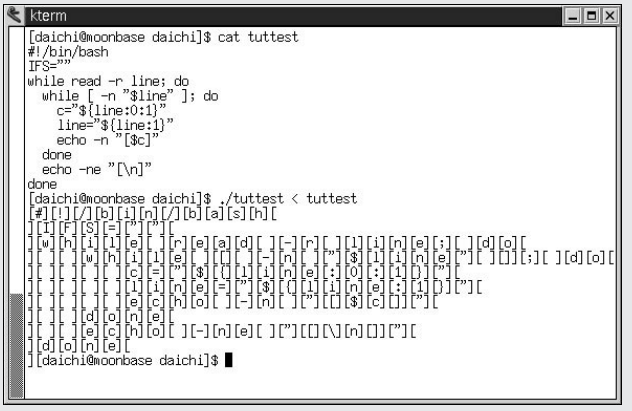
このスクリプトを実行すると、各文字が1バイトずつ大カッコで囲まれて出力される(**画面2**)。日本語など2バイト文字を含む場合は表示が乱れるので注意されたい。

●キーワードの処理と組み合わせる

キーワードの処理は、1バイトずつ文字を抽出する点はファイルの場合と同じだが、末尾に達したら先頭に戻って繰り返し抽出する点が異なる。なお、キーワードはすでにシェル変数keywordに格納されているものとする。

具体的なスクリプトは以下のようになる。

```
 1: #!/bin/bash
 2: IFS=""
 3: keyword="pogemuta"
 4: function getkey
 5: {
 6:   key="${keyword:$keyidx:1}"
```



画面2 1バイトずつ抽出し、大カッコで囲んで出力

```
 7:   keyidx=$(((keyidx+1) % ${#keyword}))
 8: }
 9: declare -i keyidx=0
10: while read -r line; do
11:   while [ -n "$line" ]; do
12:     c="${line:0:1}"
13:     line="${line:1}"
14:     getkey
15:     echo -n "[$key:$c]"
16:   done
17:   getkey
18:   echo -ne "[$key:\n]"
19: done
```

キーワードから1バイトずつ文字を抽出する処理はスクリプト中に2回登場するので、getkeyという関数を定義する(4～8行目)。keywordから抽出する位置(先頭を0とする)は整数型変数keyidxに保存されており、抽出された文字はkeyに格納される(6行目)。keyidxの値は1ずつ増やされ、keywordの長さ(${#keyword})に達すると、剰余演算子(%)により0に戻される(7行目)。つまり、keyidxの値は、「0」(キーワード先頭)から「keywordの長さ-1」(キーワード末尾)の間を動くわけだ。

メインのwhileループでは、14行目でgetkeyを呼び出しており、キーワードから抽出したkeyを、ファイルから抽出したcと一緒に大カッコで囲って出力する(15行目)。また、行末処理の際にもgetkeyを呼び出し(17行目)、keyと改行(\n)を大カッコで囲んで出力する(18行目)。つまり、暗号化の際に組み合わせる2つの文字を、このスクリプトでは単に並べて表示しているわけだ。

●各種の関数を作成する

文字を文字コードに変換する関数getordと、文字コードを文字に変換して出力する関数putchrを、それぞれ以下のように定義する。

```
function getord
{
  eval $1=$(perl -e "print ord(\"$2\")")
}
function putchr
{
  echo -ne "\\$(($1/64*100+$1%64/8*10+$1%8))"
}
```

getordでは、引数で指定した内容をシェルに処理させる組み込みコマンドevalを活用している。2番目の引数($2)の文字をperlを使って文字コードの数値に変換し、コマンド置換により最初の引数($1)で指定したシェル変数に格納する。たとえば、「getord hoge A」とすると、「A」の文字コード(65)が変数hogeに代入される。

putchrのほうは、-eオプション付きのechoが「\8進数」で任意の文字を出力できることを利用する。$((…))内は、最初の引数($1)で与えた文字コードを、対応する8進数の値に変換する演算だ。

続いて、0～255の数値を2桁の16進数文字列に変換し、スペース区切りで出力する関数puthexと、2桁の16進数文字列を数値に変換する関数getdecを、それぞれ次のように定義する。

```
xd='0123456789abcdef'
function puthex
{
  echo -n "${xd:$(($1/16)):1}${xd:$(($1%16)):1} "
}
function getdec
{
  local i h1 h2
  h1=${xd%${2:0:1}*}
  h2=${xd%${2:1:1}*}
  eval $1=$((${#h1}*16+${#h2}))
}
```

puthexでは、16進数を構成する数字記号列xd(0～9､a～f)から、サブ文字列展開を利用して、それぞれの桁で使う文字を取り出して出力する。

一方、getdecでは、パターン照合演算子(${変数名％パターン})を利用して、16進数文字列の各桁で使われている文字以降をxdから取り除き、残りの文字列の長さを使って10進数に変換する。変換した値は、evalを利用して最初の引数($1)で指定したシェル変数に格納する。

●暗号化処理のプロトタイプ

これまでのスクリプトや関数を組み合わせ、いくつか修正を加えると、暗号化処理のプロトタイプができあがる。スクリプトの主要部を以下に示す。

```
 1: function getkey
 2: {
 3:   getord key "${keyword:$keyidx:1}"
 4:   keyidx=$(((keyidx+1) % ${#keyword}))
 5: }
 6: declare -i keyidx=0
 7: while read -r line; do
 8:   while [ -n "$line" ]; do
 9:     getord ord "${line:0:1}"
10:     line="${line:1}"
11:     getkey
12:     puthex $((ord ^ key))
13:   done
14:   getkey
15:   puthex $((10 ^ key))
16: done < "${1:-/dev/stdin}"
```

getkeyは、キーワードから抽出した1バイトを文字コードに変換してからkeyに格納するよう変更された(3行目)。暗号化処理の中心となるwhileループでは、lineの先頭1バイトをgetordで文字コードに変換してordに格納する(9行目)。このordとkeyをXOR演算し、puthexで2桁の16進数文字列に変換してスペース区切りで出力する(12行目)。行末処理では、改行コード(10)とkeyのXOR演算と変換・出力を直接行なう(15行目)。

16行目の「< "${1:-/dev/stdin}"」で、このwhileループの標準入力を、最初の引数($1)で指定したファイルにリダイレクトする。$1が空文字列の場合は「/dev/stdin」(標準入力)から読み込む。

●復号化処理のプロトタイプ

復号化処理のプロトタイプとなるスクリプトの主要部は、以下のようになる。

```
1: while read -r line; do
2:   while [ -n "$line" ]; do
3:     getdec ord "${line:0:2}"
4:     line="${line:3}"
5:     getkey
6:     putchr $((ord ^ key))
7:   done
8: done < "${1:-/dev/stdin}"
```

復号化処理の中心となるwhileループでは、lineの先頭2桁の16進数文字列をgetdecで10進数に変換してordに格納する(3行目)。lineからは区切りのスペースも含め、先頭3バイトが取り除かれる(4行目)。このordとkeyをXOR演算した数値を文字コードとし、putchrで文字に変換して出力する(6行目)。平文ファイルの改行も含めて暗号化されているので、行末に関する特別な処理は必要ない。

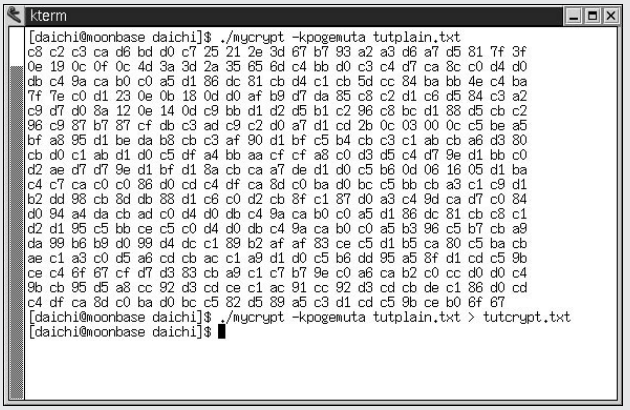
●オプションの処理などを加えて完成だ

残った処理は起動時オプションだ。通常は暗号化処理、-dオプション指定時には復号化処理を行なう。また、キーワードは-kで指定でき、省略時はキーボードから入力することにしよう。以下に、オプションと入力ファイル名、キーワード処理部分のスクリプトを示す。

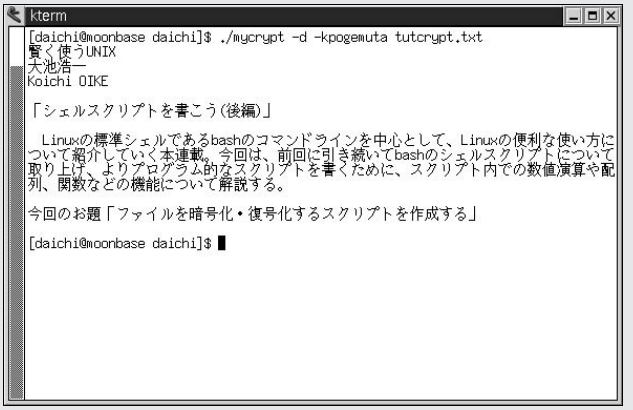
```
 1: while getopts ":dk:" opt; do
 2:   case $opt in
 3:     d ) decode=yes ;;
 4:     k ) keyword="$OPTARG" ;;
 5:     * ) echo "usage: ${0##*/} [-d] [-kKEYWORD] infile" > /dev/stderr
 6:         exit 1 ;;
 7:   esac
 8: done
 9: shift $((OPTIND - 1))
10: if [ -z "$1" ] && [ -z "$keyword" ]; then
11:   echo "usage: ${0##*/} [-kKEYWORD] infile" > /dev/stderr
12:   exit 1
13: fi
14: while [ -z "$keyword" ]; do
15:   echo -n "keyword: " > /dev/stderr
16:   read keyword
17: done
```

最初のwhileループでは、getoptsを使ってオプション処理を行なっている(1～8行目)。オプションは復号化用の-dと、キーワードを指定する-kの2つだ。続いて、コマンドラインで入力ファイルとキーワードをどちらも指定しなかった場合の処理(10～13行目)、キーワードが指定されなかった場合にキーボードから入力する処理(14～17行目)をそれぞれ行なっている。

すべての処理を組み込んだ完成形の暗号化・復号化スクリプト「mycrypt」を本誌CD-ROMに収録した。「mycrypt -kキーワード 平文ファイル名」とすると、キーワードに基づいて平文ファイルを暗号化して標準出力に出力する(**画面3**)。復号化するには、「mycrypt -d -kキーワード 暗号ファイル名」とすればいい(**画面4**)。



画面3 ファイルの内容を暗号化して出力



画面4 暗号化されたファイルを元に戻す

Webサーバ構築術（第9回）

最近、ITブームのためか、Webをデータベースと連携して使うというのが、このところ着目されているWebコンテンツのテクノロジーである。単なるHTMLではなく、データベースと組み合わせることでWebサーバのパブリッシング技術と表現能力は大きく向上する。今回は、ApacheからフリーのRDBMSであるPostgreSQLを利用する方法を紹介しよう。

ApacheとPostgreSQLの連動

文：中島昌彦
Text : Masahiko Nakajima

修正、パブリッシングが即時にできる

ユーザー管理や膨大なデータ管理といえば、データベースとすぐに思い浮かぶはずだ。ユーザー管理ならば、フィールドは定型になっているので、データベース向きだからである。しかし、ユーザー管理に限らず、Webサーバで公開されているコンテンツの中には、データベースで管理したほうが数段楽なものもある。たとえば、会社のプレスリリース、製品カタログ、更新履歴、といったものだ。Webコンテンツのデザインは定型でありながら、テキストの中身がちがう。もし1つ1つHTMLで起こしていたら、デザインリニューアルのときに、すべてのHTMLファイルを書き換えなければならない。実に手間がかかる作業だ。

しかし、プレスリリースや製品カタログ、更新履歴をHTML単位ではなく、中身のテキストだけをデータベース（以下DB）で管理すると、デザイン変更にも即座に対応でき、しかもDBに登録したときから、パブリッシングができる。修正も、該当するテキストデータを変更するだけで、HTMLエディタを使って修正、変更、Webサーバにアップロードという作業が不要だ。

さらにDB管理をすると、アクセス数が増大したときでもしっかりと対処できる。バックエンドのDBは1つであっても、Webサーバは複数に分散できる。これからのWebパブリッシングには、もはやDBでの管理機能が必須項目ともいえる。

PostgreSQLのインストール

今回は、ApacheをPostgreSQLとPerlと組み合わせて使用する。Red Hat LinuxやLASER5 Linuxなら、PostgreSQLのRPMファイルが用意されているので、CD-ROMなどからインストールする。

```
# rpm -i postgresql-6.5.2-1.i386.rpm
# rpm -i postgresql-perl-6.5.2-1.i386.rpm
# rpm -i postgresql-server-6.5.2-1.i386.rpm
```

Red Hatの場合、PostgreSQLの起動は以下のコマンドで行う。

```
# /etc/rc.d/init.d/postgresql start
```

RPMファイルからのインストールで提供されていない場合には、ちょっと面倒だがソースからインストールする手順をまとめておこう。

RINGサーバ（http:/www.ring.gr.jp/）などから、PostgreSQLのtarボール（http://www.ring.gr.jp/pub/misc/db/postgresql-jp/6.5.3/postgresql-6.5.3.tar.gz）を入手し、/usr/local/srcディレクトリでインストール作業を行う。

```
# cd /usr/local/src
# tar xvfz postgresql-6.5.3-3.tar.gz
```

としてソースを展開する。このままmakeにいきたいところだが、以後の安全のために、PostgreSQL専用のアカ

ウントを作り、そのアカウントを使って作業をする。理由は単純で、PostgreSQLの管理者を特定個人にしてしまうと、その利用者でなければDBをコントロールできない。学校や会社など、複数人でDBを管理するようなときには、特定個人ではなく、数人で管理にあたることがある。そんなときに、個人アカウントとままだと、su -したときに、個人のセキュリティが保てなくなる。DB管理用アカウントを使って、それで作業をするほうがいい。

ユーザーアカウントはなんでもかまわない。ここでは、

```
# adduser postgres
# chown -R postgres postgresql-6.5.3
# su - postgres
```

とpostgresユーザーを作り、postgresユーザーを使って作業を進める。

```
$ cd /usr/local/src/postgresql-6.5.3/src
$ ./configure --with-mb=EUC_JP
```

ソースの準備はここまでだ。configureのときに、マルチバイトコードとしてEUC_JPを指定しているが、UNICODE、Muleの内部コードなど、ほかにもいくつかのマルチバイトコードを指定できる。ただし、SJIS、JISはサポート外なので、たいていはEUC_JPでの指定となるはずだ。

```
$ make all
$ su -
# mkdir /usr/local/pgsql
# chown -R postgres /usr/local/pgsql
# exit
$ make install
```

という手順でPostgreSQLがインストールできる。

これに加えて、最後にPerl用のインターフェイスをmakeする。

```
$ cd interface/perl5/
$ perl Makefile.PL
$ make
$ su
# make install
# exit
```

この手順で、PostgreSQLのPg.pmが組み込まれる。これで、ようやくPostgreSQLがPerlから利用できるようになる。

さて、PostgreSQLはインストールできたものの、この段階ではテーブルを作る準備ができていない。そこで、postgresユーザーの環境変数に次の値をセットする。

```
$ export PATH="$PATH":/usr/local/pgsql/bin
$ export POSTGRES_HOME=/usr/local/pgsql
$ export PGLIB=$POSTGRES_HOME/lib
$ export PGDATA=$POSTGRES_HOME/data
$ export LD_LIBRARY_PATH="$LD_LIBRARY_PATH":"$PGLIB"
```

ログインシェルがbashならば、.bashrcにこの値を書き込んでおくとよいだろう。あとは、

```
source ~/.bashrc
initdb --pgencoding=EUC_JP
postmaster -S
```

として、データベース領域の初期化と、postmasterをデーモンで起動しておく。再起動したときに、postmaster -Sが動くように、/etc/rc.d/rc.localに、

```
su - postgres -c "/usr/local/pgsql
```

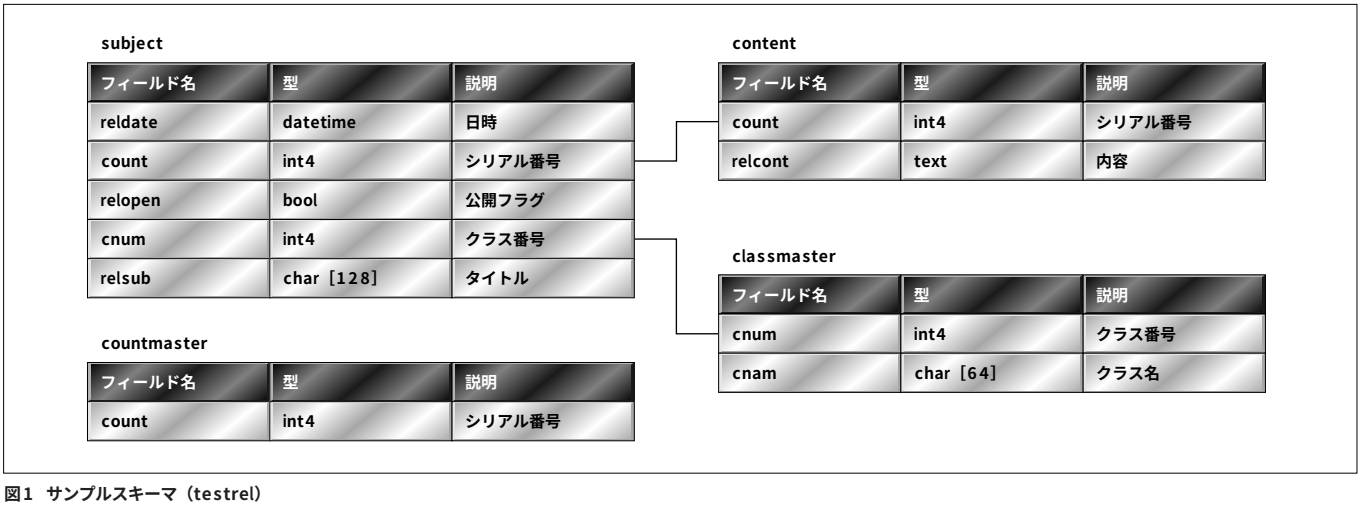


図1 サンプルスキーマ（testrel）

```
/bin/postmaster -S"
```

の1行を加えておくといいだろう。

テーブルを作成する

ここまでの手順で、DBが動くようになった。ここから、基本となるテーブルを作る。

まず、**図1**のようなテーブルを作りDBを構築する。ニュースリリース用のDBもどきだが、この一連のDBをtestrelという名称で作業をしよう。

postgresユーザーになって、

```
$ createdb -E EUC_JP testrel
```

という作業で、testrelというデータベースが構築される。あとは、testrel中で図1に合わせてsubject、content、classmaster、countmasterのテーブルを作っていく。

今回のようにテーブル数が多くないものは、PostgreSQLのSQLインタープリタを呼び出して手作業で追加していってもかまわないが、別サーバでの再構築、全体の構成を見直すときに、sqlファイルとして別に持っておくといい。そこで、testrel.sqlとしてリスト1のファイルを作り、これを読み込ませる。作成したtestrel.sqlは、

```
$ psql testrel <testrel.sql
```

とすることで、subject、content、classmaster、countmasterの4つのテーブルができあがる。

リスト1 テーブル作成のためのtestrel.sql

```
create table subject (
        reldate datetime,
        count int4,
        relopen bool,
        cnum int4,
        relsub char(128)
);
create table content (
        count int4,
        relcont text
);
create table classmaster (
        cnum int4,
        cnam char(64)
);
create table countmaster (
        count int4
);
```

テスト版リリース管理DBの仕組み

簡単にテスト版リリース管理DBの仕組みを解説しよう。

テーブルclassmasterは、マスターテーブルだ。クラス番号とクラス名で管理しており、クラス名はクラス番号から引用する。

テーブルsubjectは、タイトル名だけを管理するテーブルで、このテーブルがベースとなっている。テーブルcontentは、subjectのcountフィールドで引かれ、該当するリリース文章の中身が入っている。データ型としてtextを指定しているため、可変長テキストを扱える。ただし、PostgreSQLでは、8Kバイトを越えるデータは扱えない。

あまり美しい作りではないが、countをキーにcontentを引けば、その内容が取り出せる。subject中のcnumをベースにclassmasterを引けば、クラス名が取り出せるという仕組みだ。classmasterを別テーブルで管理しているため、新たなクラス増設が必要になったら、classmasterに追加すればいい。

Apacheがこの全体のDB構成に対して、何をするのかといえば、DBへの入出力インターフェイスがすべてWeb経由のものになる。データ入力、データ取り出し、といった機能を、Perl経由のCGIで動かすことで、クライアントOSを問わずにデータ入力、表示ができるシステムを組み上げられる。

前回、Perlで作成したTodoリストではテキストファイルとしてデータを管理したが、今回の例もそのようなレベルで十分に対応できる。ただし、DBで管理していることで、データが膨れあがったときの対応、書き込んだ内容の修正、複数ユーザーの同時書き込み処理といった日常のメンテナンス作業をCGI側で吸収する必要がなくなる。

CGIからPostgreSQLにアクセスする

Apache側の設定としては、前回のPerlでCGIが動く設定になっていればいい。逆に、PostgreSQLのほうにはユーザー登録作業が必要だ。

Apacheでは、CGIを特定ユーザーで動作する。/usr/local/apache/conf/httpd.confの中で、

```
User nobody
Group nobody
```

というようにApacheが動作しているときのユーザー属性を登録してある。そこで、このnobodyユーザーがPostgreSQLのデータベースを操作できるようにする。

```
# su - postgres
$ createuser nobody
```

として、対話的に問い合わせに答えていく。問合わせの内容は、nobodyにデ

ータベースを作らせるか、nobodyはスーパーユーザーか、nobody用のデータベースを作るかの3点だ。**リスト2**のように、どれも“n”と入力する。

これで、nobodyユーザーが登録できたが、nobodyユーザーはsubject、content、classmaster、countmasterのテーブルにアクセスできないという問題が残っている。そこで、各テーブルにアクセスし、insert、updateできる権限を、nobodyに与える作業をして、ようやくDBにアクセスできるようになる。

```
$ psql testrel
```

として、SQLインタープリタを実行してから、**リスト3**を実行し、ユーザーnobodyに対して、select、insert、updateの権限を与えておく。また、カウントの初期値として、0をcountmasterに定義する。

この一連の作業で、ようやくCGIからPostgreSQLへアクセスできるようになる。ただし、この段階ではテーブルが存在しているだけであり、データの中身がない。

最初に作らなければいけない部分は、テーブルclassmasterを管理するツールだ。Webブラウザを使って、クラスのデータも修正できるようにしておくと手間がかからない。これを構成するスクリプトが、class.cgi、classw.cgiである。class.cgiからclass.htmlというテンプレートファイルを読み出している(**リスト4**)。

どちらも、共通するスクリプトとして、psqldb.plをrequireしている。また、psqldb.plから日本語文字コード変換ライブラリjcode.pl-2.11をrequireするため、jcode.pl-2.11をFTPサイト(ftp://ftp.iij.ad.jp/pub/IIJ/dist/utashiro/perl/jcode.pl-2.11)から入手して、ほかのスクリプトと同一の場所に入れておかないと動作しない。

あとは、Webブラウザからclass.cgiを呼び出し(**画面1**)、クラス番号とクラス名を登録していけば、テーブルclassmasterへデータを登録できるようになる。

classmasterを管理するツールができたところで、テーブルsubject、contentの2つが書き込めるようになる。

そのためのcgiが、content.cgi、contente.cgi、contentew.cgi、contentw.cgiの4つだ。さらに、content.html、contente.htmlという2

リスト2 WebからアクセスできるようにPostgreSQLにnobodyアカウントを追加

```
$ createuser nobody
Enter user's postgres ID or RETURN to use unix user ID: 99 ->
Is user "nobody" allowed to create databases (y/n) n
Is user "nobody" a superuser? (y/n) n
createuser: nobody was successfully added
Shall I create a database for "nobody" (y/n) n
```

リスト3 nobodyアカウントがテーブルにアクセスできるように権限を渡す

```
grant select,insert,update on subject to nobody;
grant select,insert,update on content to nobody;
grant select,insert,update on classmaster to nobody;
grant select,insert,update on countmaster to nobody;
insert into countmaster values (0);\
```

画面2
新規クラスを登録することで、クラス分け部分ができあがる。そこで、本格的な中身が登録できるようになる。このページは、http://.../content.cgiとダイレクトに呼び出す

画面1
新規クラス登録のページ。ダイレクトにhttp://.../class.cgiとURLを指定することで呼び出す

画面3
実際の登録内容修正。DBに入っているデータを呼び出して表示し、修正が終ったら必要な部分のみ書き戻すという作業のため、ファイル全体のロックが不要となる

つのHTMLファイルをテンプレートとして呼び出していて、合計6つのファイルを使い、テーブルsubjectとcontentを管理している（**リスト5**）。

content.cgiを実行して、新規データの入力を行う（**画面2**）。修正ボタンで登録した内容の訂正ができるようになっている（**画面3**）。

スクリプトの中身を見てもらうとわかるが、テキストを管理するよりも、むしろスクリプト自体は短くなる。特に、データ修正とデータ書き込みに関わる部分はデータベースにデータを渡すだけなので、DBさえ構築できてしまえば、CGIにかかる負荷が大幅に減少する。

また、PostgreSQL自体が書き込み時に対象フィールドのみロックをかけるため、ファイル全体にロックをかける必要がなくなる。また、同時書き込みがほとんどゼロと考えられるならば、contentw.cgiで処理しているように、テーブルcountmasterのcountの値をインクリメントし、その直後に値を読み出すことで、シリアルを得られる。

読み込み、書き込み中にはcountmasterのcountにはロックがかかっているはずなので、連続した処理中に新たな書き込みが生じるようなことがなければ、こんな仕組みでも同時書き込みに対しての対策となる。ファイル全体をロックしないために、DBのパフォーマンスさえうまくコントロールして

リスト4-1 class.cgi

```
#!/usr/bin/perl
use Pg;
require "psqldb.pl";
&opendb ('testrel');
$max=&sqlout ('select * from classmaster order by cnum');
if (!$max) {
    $gettable="--ありません--";
} else {
    $gettable ="<TABLE BORDER=\"1\"><TR><TH>クラス番号
</TH><TH>クラス名</TH>\n";
    for ($i=0;$i<$max;++$i) {
         $getname=$result->getvalue($i,1);
         $getname=~ s/ +$//g;
         $gettable .='<FORM NAME="'.$i.'"
ACTION="classw.cgi" METHOD="POST" ENCTYPE="application/x-
www-form-urlencoded">'."\n";
         $gettable .='<TR><TD><INPUT TYPE="HIDDEN"
NAME="cnum" VALUE="';
         $gettable .=$result->getvalue($i,0);
         $gettable .='"><INPUT TYPE="text" NAME="cnum2"
VALUE="';
         $gettable .=$result->getvalue($i,0);
         $gettable .='" SIZE="4"></TD>'."\n";
         $gettable .='<TD><INPUT TYPE="TEXT" NAME="cnam"
VALUE="';
         $gettable .=$getname.'"
MAXLENGTH="64"></TD>'."\n";
         $gettable .='<TD><INPUT TYPE="HIDDEN" NAME="com"
VALUE="update"><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="
修正"></TD></TR></FORM>'."\n";
    }
    $gettable .="</TABLE>\n";
}
print "Content-type: text/html\n\n";
open (READ,"class.html");
while (<READ>) {
    s/--gettable--/$gettable/g;
    print;
}
```

リスト4-2 class.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
    <TITLE>クラス登録/修正</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">

<FORM ACTION="classw.cgi" METHOD="POST"
ENCTYPE="application/x-www-form-urlencoded">
<P>新規登録<BR>
<BR>

<TABLE BORDER="1">
    <TR>
         <TD>クラス番号</TD>
         <TD><INPUT TYPE="TEXT" NAME="cnum" SIZE="4"
MAXLENGTH="4"></TD>
    </TR>
    <TR>
         <TD>クラス名</TD>
         <TD><INPUT TYPE="TEXT" NAME="cnam" SIZE="64"
MAXLENGTH="64"></TD>
    </TR>
    <TR>
         <TD COLSPAN="2"><INPUT TYPE="HIDDEN" NAME="com"
VALUE="insert"><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="
新規登録"></TD>
    </TR>
</TABLE>
</FORM>
<BR>
登録内容<BR>
--gettable--<BR>

</BODY>
</HTML>
```

リスト4-3 classw.cgi

```
#!/usr/bin/perl
use Pg;
require "psqldb.pl";
&getquery;
&opendb ('testrel');
if ($QUERY{com} eq 'insert') {
    $querycom="$QUERY{com} into classmaster values
('$QUERY{cnum}','$QUERY{cnam}')";
    if (&sqlout ("select cnum from classmaster where
cnum='$QUERY{cnum}'")) {
        &errorout ("すでに番号が登録されています");
    }
} else {
    if ($QUERY{cnum} != $QUERY{cnum2}) {
        if (&sqlout ("select cnum from classmaster where
cnum='$QUERY{cnum2}'")) {
            &errorout ("すでに番号が登録されています");
        }
    }
    $querycom="$QUERY{com} classmaster set
cnam='$QUERY{cnam}', cnum='$QUERY{cnum2}' where
cnum='$QUERY{cnum}'";
}
&sqlout("$querycom");
print "Location: $ENV{HTTP_REFERER}\n\n";
exit;
```

リスト5-1 contentw.cgi

```
#!/usr/bin/perl
use Pg;
require "psqldb.pl";
&getquery;
if (!$QUERY{cnum}) {
    &errorout ("クラス名は必須です");
}
&opendb ('testrel');
if (!$QUERY{relopen}) {
    $boolval='False';
} else {
    $boolval='True';
}
&sqlout("update countmaster set count=count+1");
&sqlout("select count from countmaster");
$count=$result->getvalue(0,0);
$querycom="insert into subject values
(CURRENT_TIMESTAMP,$count,$boolval,'$QUERY{cnum}','$QUERY{subject}')";
$querycom2="insert into content values ('$count','$QUERY{relcont}')";
&sqlout("$querycom");
&sqlout("$querycom2");
print "Location: $ENV{HTTP_REFERER}\n\n";
```

リスト4-4 psqldb.pl

```
require "jcode.pl-2.11";
$gettable='';
sub sqlout {
    $result= $conn->exec ("$_[0]");
    return ($result->ntuples);
}
sub opendb {
    $conn=Pg::connectdb("dbname=$_[0]");
    if (PGRES_CONNECTION_OK ne $conn->status) {
        $ermsg=db.$conn->errorMessage;
        print "Content-type: text/plan\n\n$ermsg";
        exit;
    }
}
sub getquery {
    read(STDIN, $buf, $ENV{'CONTENT_LENGTH'});
    foreach $pair (split(/&/, $buf)) {
        ($name, $value) = split(/=/, $pair);
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",
hex($1))/eg;
        &jcode'convert(*value,'euc');
        $QUERY{$name}=$value;
    }
}
sub query {
    foreach $pair (split(/&/, $ENV{QUERY_STRING})) {
        ($name, $value) = split(/=/, $pair);
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",
hex($1))/eg;
        &jcode'convert(*value,'euc');
        $QUERY{$name}=$value;
    }
}
sub errorout {
    print "Content-type: text/plain\n\n$_[0]";
    exit;
}
```

おけば、多数のアクセスがあったとしてもレスポンスの低下を防ぐことができる。

DBでデータを管理することで、タイトル検索、絞り込みの作業の処理が手軽に装備できる。数個の情報であれば、わざわざ検索機能や絞り込み機能を加える必要はないが、公開情報が増えていけば、タイトルの検索や絞り込みの機能が必須である。

リスト5-2 content.cgi

```
#!/usr/bin/perl
use Pg;
require "psqldb.pl";
&opendb ('testrel');
$max=&sqlout ('select * from classmaster order by cnum');
if (!$max) {
    $getclass='<OPTION VALUE="">--ありません--';
} else {
    for ($i=0;$i<$max;++$i) {
        $getclass .='<OPTION VALUE="'.$result-
>getvalue($i,0).'">'.$result->getvalue($i,1)."\n";
    }
}
$max=&sqlout ('select
subject.count,subject.relopen,subject.relsub,classmaster.c
nam from subject,classmaster where subject.cnum =
classmaster.cnum order by subject.count');
if (!$max) {
    $getsubj='--ありません--';
} else {
    $getsubj ="<TABLE BORDER=\"1\"><TR><TH>タイトル
</TH><TH>クラス</TH><TH>フラグ</TH></TR>\n";
    for ($i=0;$i<$max;++$i) {
        $getsubj .='<TR><TD>'.$result-
>getvalue($i,2).'</TD><TD>'.$result-
>getvalue($i,3).'</TD><TD>'.$result-
>getvalue($i,1).'</TD><TD><FORM ACTION="contente.cgi"
METHOD="POST" ENCTYPE="application/x-www-form-
urlencoded"><INPUT TYPE="HIDDEN" NAME="count"
VALUE="'.$result->getvalue($i,0).'"><INPUT TYPE="SUBMIT"
NAME="Submit" VALUE="修正"></TD></FORM></TR>'."\n";
    }
    $getsubj .='</TABLE>';
}
print "Content-type: text/html\n\n";
open (READ,"content.html");
while (<READ>) {
    s/--getclass--/$getclass/g;
    s/--getsubj--/$getsubj/g;
    print;
}
```

リスト5-3 contentew.cgi

```
#!/usr/bin/perl
use Pg;
require "psqldb.pl";
&getquery;
&opendb ('testrel');
if (!$QUERY{relopen}) {
    $boolval='False';
} else {
    $boolval='True';
}
$querycom="update subject set
relopen=$boolval,cnum=$QUERY{cnum},relsub='$QUERY{relsub}'
where count=$QUERY{count}";
$querycom2="update content set relcont='$QUERY{relcont}'
where count=$QUERY{count}";
&sqlout("$querycom");
&sqlout("$querycom2");
print "Location: content.cgi?\n\n";
```

リスト5-4 content.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
    <TITLE>リリース登録/修正</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<FORM ACTION="contentw.cgi" METHOD="POST"
ENCTYPE="application/x-www-form-urlencoded">
<P>新規登録<BR>
<BR>
<TABLE BORDER="1">
    <TR>
        <TD>タイトル</TD>
        <TD><INPUT TYPE="TEXT" NAME="subject" SIZE="64"
MAXLENGTH="128"></TD>
   </TR><TR>
        <TD>クラス</TD>
        <TD>
            <SELECT NAME="cnum">
            <OPTION VALUE="" SELECTED>クラス名選択
--getclass--
            </SELECT>
        </TD>
    </TR><TR>
        <TD>公開</TD>
        <TD><INPUT TYPE="CHECKBOX" NAME="relopen"
VALUE="True"  CHECKED>公開します</TD>
    </TR><TR>
        <TD>内容</TD>
        <TD><TEXTAREA NAME="relcont" ROWS="8"
COLS="64"></TEXTAREA></TD>
    </TR><TR>
        <TD COLSPAN="2"><INPUT TYPE="HIDDEN" NAME="com"
SIZE="-1" VALUE="insert"><INPUT TYPE="SUBMIT"
NAME="Submit" VALUE="新規登録"></TD>
    </TR>
</TABLE>
</FORM>
<BR>
登録内容<BR>
--getsubj--
</BODY>
</HTML>
```

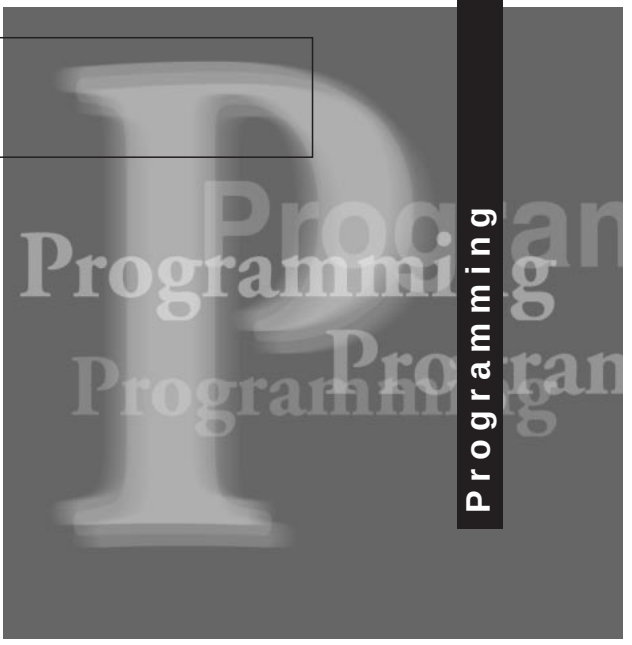
リスト5-5 contente.cgi

```
#!/usr/bin/perl

use Pg;
require "psqldb.pl";

&getquery;
&opendb ('testrel');

$max=&sqlout ('select * from classmaster order by cnum');
if (!$max) {
    $getclass='<OPTION VALUE="">--ありません--';
} else {
    for ($i=0;$i<$max;++$i) {
        $getclass .='<OPTION VALUE="'.$result-
>getvalue($i,0).'">'.$result->getvalue($i,1)."\n";
    }
}


$max=&sqlout ("select * from subject,classmaster,content
where subject.cnum=classmaster.cnum and
subject.count=$QUERY{count} and
content.count=$QUERY{count}");

if (!$max) {
    $getclass='<OPTION VALUE="">--ありません--';
} else {
    $subject=$result->getvalue(0,4);
    $subject=~ s/ +$//g;
    $relcont=$result->getvalue(0,8);
    $cnum=$result->getvalue(0,3);
    $getclass=~ s/(VALUE="$cnum")/\1 SELECTED/;
    $relopen=$result->getvalue(0,2);
    if ($relopen eq 't') {
         $relopen='CHECKED';
    } else {
         $relopen='';
    }
}


print "Content-type: text/html\n\n";
open (READ,"contente.html");

while (<READ>) {
    s/--subject--/$subject/g;
    s/--relcont--/$relcont/g;
    s/--getclass--/$getclass/g;
    s/--relopen--/$relopen/g;
    s/--count--/$QUERY{count}/g;
    print;
}
```

リスト5-6 contente.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>

<HEAD>
    <META HTTP-EQUIV="Content-Type"
CONTENT="text/html;CHARSET=EUC-JP">
    <TITLE>リリース修正</TITLE>
</HEAD>

<BODY BGCOLOR="white">

<FORM ACTION="contentew.cgi" METHOD="POST"
ENCTYPE="application/x-www-form-urlencoded">
<P>登録内容修正<BR>
<BR>

<TABLE BORDER="1">
    <TR>
         <TD>タイトル</TD>
         <TD><INPUT TYPE="TEXT" NAME="relsub" SIZE="64"
MAXLENGTH="128" VALUE="--subject--"></TD>
    </TR>
    <TR>
         <TD>クラス</TD>
         <TD>
                  <SELECT NAME="cnum">
--getclass--
                  </SELECT>
         </TD>
    </TR>
    <TR>
         <TD>公開</TD>
         <TD><INPUT TYPE="CHECKBOX" NAME="relopen"
VALUE="True" --relopen-->公開します</TD>
    </TR>
    <TR>
         <TD>内容</TD>
         <TD><TEXTAREA NAME="relcont" ROWS="8" COLS="64"
CALUE="relcont">--relcont--</TEXTAREA></TD>
    </TR>
    <TR>
         <INPUT TYPE="HIDDEN" NAME="count" VALUE="--
count--"><TD COLSPAN="2"><INPUT TYPE="SUBMIT"
NAME="Submit" VALUE="修正"></TD>
    </TR>
</TABLE>
</FORM>
</BODY>

</HTML>
```

プログラミング工房



一部のマニア向けOSであったLinuxが、ここまで注目を集めるようになったのは、X Window Systemの存在が大きい。Xがなければ、Microsoft Windowsと比較されることもなかったはずだ。今回から、X Window Systemのプログラミングに挑戦する。まずは、基本的な機構について見ていこう。

第6回　Xのプログラミング（1）

文：藤沢敏喜
Text : Toshiki Fujisawa

今月号から3回にわたり、X Window Systemのプログラミングを取り上げる。一般にウィンドウシステムのプログラミングは難しい概念が多く、解説すべき内容は膨大である。

そのため、本連載ではXの実践的なプログラミングを解説するのではなく、Xの偉大な思想とその基本概念を理解することを主眼として解説してみたい。第1回目の今月はX Window Systemの操作自体があまり詳しくない人にもXの機構がわかるように進めていきたい。

X Window Systemの名前の由来

X Window System は、1980年ごろスタンフォード大学で開発されたウィンドウシステム「W」の次の文字である「X」から命名された（ちなみにC言語は、その祖先であるB言語の次の文字である「C」から名付けられた）。

オンラインマニュアルで「X」を参照すると（man Xを実行する）、このウィンドウシステムは、次の5つのうちのいずれかで呼ばなくてはならないと書かれている。

- **X**
- **X Window System**
- **X Version 11**
- **X Window System, Version 11**
- **X11**

ネットニュースなどではX Windowsというような、とんでもない呼ばれ方もされているようであるが、読者の皆さんにはぜひ上記5つのうちのどれかの正式な名称を用いるようにしていただきたい。

■X Window Systemの歴史

マサチューセッツ工科大学（MIT）で、分散型教育環境構築プロジェクト（Atena Project）の中の1つのテーマとして、1984年に開発されたウィンドウシステムであるXは、現在に至るまで次のように開発が行われている。

1985年 X9　　ライセンスフリーとして公開
1986年 X10　　Xlibのみをサポート
1987年 X11R1　Xプロトコルの大幅変更。GCの導入
1988年 X11R2　ツールキットの追加
1988年 X11R3　フォントフォーマットを追加
1990年 X11R4　サーバの高速化
1991年 X11R5　フォントサーバ。国際化。3D機能
1994年 X11R6　国際化に関する機能の充実

最近になって「オープンソース」という言葉が流行しているが、Xはその誕生時からソースコードが広く公開されたため、非常に幅広く使われている。もしXの存在がなければLinuxやFreeBSDが、Microsoft Windowsと比べられるようなことにはならなかっただろうと思われる。

PC-UNIXにおいては非常に重要なソフトウェアである。

X Window Systemの特徴

現在のLinuxディストリビューションでは、インストール終了時にはKDEやGNOMEなどが既に使えるようになっている。ユーザーの目からは、Microsoft Windowsと同じように見えるかもしれないが、その内部構造は大きく異なっている。

特に、UNIXの文化を受け継いでいるためか、ウィンドウシステムを構成するレイヤが非常に洗練された構成となってる。そのため多種多様なOSやマシンに実装することが容易で、どんなネットワーク環境でも高い実行効率を得ることができる。また、GUI部分を分離しているため、さまざまな見かけや操作性を簡単に実現できる。以下では、これらの特徴について少し詳しく述べてみる。

1. ネットワーク透過

Xでは、プログラムを実行するマシンと、表示やキー入力を行うマシンを分離することができる。ここでは遠方にあるFreeBSDマシン（bsd.fujisawa.gr.jp）上で実行されている電卓プログラムを、手元にあるLinuxマシン（lnx.fujisawa.gr.jp）に表示する手順について説明する。もし手元に2台のマシンがあるならば、ぜひ実際にやってみてほしい。

さて、まずLinuxマシンでXを起動して、ktermなどのターミナルエミュレータを実行する。次に、このLinuxマシンにFreeBSDマシンとのXでの接続を許可するため、xhostコマンドを入力する。

```
$ xhost +bsd.fujisawa.gr.jp
```

続いて遠方にあるFreeBSDマシンに次のようにしてロ

Column

Xサーバと Xクライアント

インターネットの普及にともない「サーバ」という言葉は、テレビCMなどでも盛んに耳にするようになった。

ある日、電車に乗っていると、吊広告の中の「ホームサーバープレゼント」という文字が目に飛込んできてちょっとびっくりしたが、コンピュータではなく、生ビールを注ぐ装置をプレゼントするというものであった。

メールサーバやHTTPサーバは、メールやWebというサービスを提供するが、ビールサーバの場合は家に遊びにきたお客さん（クライアント）にビールを注ぐサービスを提供するというわけである（図）。

さて、「Xサーバ」という用語であるが、これはグラフィックスの描画やキーボードからの入力といったサービスを提供するサーバである。つまり、電卓や時計などのプログラム（Xクライアント）から、ある座標位置に文字や図形を描画するようにXサーバに依頼すると、Xサーバはその依頼に応じてスクリーンに描画を行う。通常は、ホストがサーバであり、端末がクライアントになる。しかし、Xに限っては、ホスト側が「Xクライアント」に、クライアント側が「Xサーバ」になるのだ。

Xサーバは、ネットワークのポート番号6000番を監視し、Xクライアントからの依頼を待ち受ける。描画などのすべての依頼は、ネットワーク経由で送られる。そのフォーマットはOSやCPUに依存しないように定められている。したがって、どのようなOSでもXサーバを実装することが可能となっていて、Sunのワークステーションなどの UNIXマシンだけでなく、Microsoft WindowsやOS/2などでもXサーバを動作させることができる。

図　Xサーバ

グインして、電卓プログラムであるxcalcなどを実行する。

```
$ telnet bsd.fujisawa.gr.jp
Trying 10.6.72.17...
Connected to bsd.fujisawa.gr.jp
Escape character is '^]'.

FreeBSD/i386 (bsd.fujisawa.jp) (ttyp1)

login: fujisawa
Password:
....
$ export DISPLAY=lnx.fujisawa.gr.jp:0.0
  ↑cshの場合はexportでなくsetenv
$ /usr/X11R6/bin/xcalc &
$ /usr/X11R6/bin/xclock -update 1 &
```

このようにすると、手元のLinuxマシンに電卓プログラムや時計が表示されるはずだ。ここでDISPLAYへ設定する文字列は、

「マシン名:ディスプレイ番号.スクリーン番号」

という形式であるが、多くの場合ディスプレイは1つしかないので、「マシン名:0.0」となる。

ちなみに、この状況下では次のようにしてプログラムが実行されていることになる。

(1) 手元のLinuxマシンのマウスの動きがネットワークを通じて、遠方のFreeBSDマシンへ送られる。
(2) FreeBSDマシン上で動作している電卓プログラムは、ネットワークを通して受け取ったマウスが押された位置座標を調べる。そして、そこに「数字ボタン」や「÷ボタン」がある場合はそれに応じた演算を行い、最終的に画面に表示する情報を作成する。そしてその情報をネットワークを通して、Linuxマシンへ送り返す。
(3) Linuxマシンでは、FreeBSDマシンから受け取った表示情報に基づき、画面に描画を行う。

このように、表示と演算を別々の計算機で行えるとさまざまなメリットがある。たとえば、教育機関では1台のコンピュータにプログラムを入れておくだけで、多数の学生がそれを使用することができる。

もし、非力なノートマシンしか手元にない場合でも、1Gバイトのメモリが搭載されているマシンがネットワークで使えるならば、GIMPを実行して数100Mバイトの巨大画像も軽やかに編集することができるのである。

また、Microsoft Windows上でXサーバプログラムを動作させれば、Microsoft WORDから、Linux上のmuleへの双方向コピー&ペーストも可能となる。

ちなみに今書いているこの原稿は、自宅の2階に置いてある完全無音なディスクレスマシン（ネットワークブート）を使って執筆している。ところが、このマシンはファンなしで動作させるために非力な486CPUを使っているので、muleなどのエディタやかな漢字変換プログラムを満足に動作させることができない。そこで、エディタなどは1階にあるコンピュータで動作させ、Xを用いて表示のみを行っているのである。

以上のように、ネットワークを経由して自由自在に表示が行えるという点が、X Window Systemの非常に大きな特徴となっている。これに慣れてしまうと、ネットワーク透過でないウィンドウシステムはもう使うことができない。

2. オープンな環境で高パフォーマンスで動作可能

Microsoft Windowsでは、Win32やMFCのようなライブラリを用いてさまざまなウィンドウ操作が行われるが、そのライブラリの中身は完全にクローズドである。

一方、Xの場合は、前述のようにネットワークを通して公開されたプロトコルによって種々のウィンドウ操作を行うことができる。プロトコルが公開されているだけでなく、Xサーバの実装など、すべてのソースコードが公開されているため、完全にオープンなガラス張りの世界である。

また、そのプロトコルは世界中から非常に優秀な人材が集まることで知られるMITのメンバーなど、多くの人たちが考えたものであるため、たいへんよくできている。いろいろなCPUやさまざまなOSに簡単に移植できるうえ、非常に効率よく動作するように考えられていて、将来の拡張に関しても考慮されている。

そして、Xプロトコルは共有メモリやTCP/IPだけでなく、どのような伝送路を使っても通信可能であるように設計されている。

Xが開発されたのは15年以上も前であり、当時のCPUやメモリ、ビットマップディスプレイの性能を考えると、その先進性には驚かされる。筆者が最初にXプロトコルの話を聞いたときには、そんなやり方で実用的なものになる

ものかと疑った。特に、自分自身のビデオメモリに描画する場合も、ネットワークを経由するのと同じプロトコルを使うという部分には、パフォーマンスが出るはずがないと思ったのである。しかしながら、Xプロトコルの詳細を知るにしたがって、たいへん巧妙な方法でネットワークトラフィックを減らすように工夫されていることがわかった。また、一般にはXプロトコルのオーバーヘッドよりもグラフィックスの描画時間のほうが長いため、高いパフォーマンスで描画ができることもわかり、その斬新なアイデアに感動した。

3. さまざまなユーザーインターフェイスを作成可能

Xプロトコルでは、ウィンドウシステムにおいて普遍的で基本的な操作のみを提供している。つまり、ウィンドウを階層的に開く機構や、線や円弧を描いたり、領域を塗りつぶすなどの基本機構（メカニズム）は提供するが、ウィンドウを閉じるボタンの位置や形などの見かけ（ポリシー）は提供していない。

本連載の「Mingw32」を用いたプログラミングで解説したように、Win32環境では、CreateWindow関数を用いるとウィンドウが開かれ、上部にはタイトルバーが付き、タイトルバーの右端にはウィンドウを閉じる「×」ボタンが、ウィンドウシステムによって自動的に付けられる。

一方、Xではタイトルバーを付けたりするのはウィンドウマネージャと呼ばれる一般ユーザーアプリケーションで行われ、Xの下層部分で行われるわけではない。

つまり、メカニズムを提供するレイヤと、ポリシーを提供するレイヤが明確に別れているのである。

Microsoft Windowsでは、タスクバーやタイトルバーの見かけや機能を変更することはたいへん困難だが、Xではさまざまなウィンドウを開いたまま、ウィンドウマネージャを入れ替えるだけで、見かけを変更することが可能で

Column

tcpdumpによるXプロトコルの観測

Xのプログラムが、ネットワークを通して実行されていることを実感するには、ネットワークをモニタしてみるとよい。

ネットワークをモニタするには「tcpdump」を使うのが便利だ。tcpdumpはrootユーザーでないと使えないようになっている場合が多いので、スーパーユーザーでログインしてから実行する。 Red Hat Linux6.1の場合、tcpdumpは/usr/sbinにあるので、パスが通っていることを確認して、

```
# tcpdump
```

というように実行する。コマンドラインオプションを付けない場合は、デフォルトのEthernetをモニタするようになっているので、なにか通信をするとその様子が表示されるはずだ。

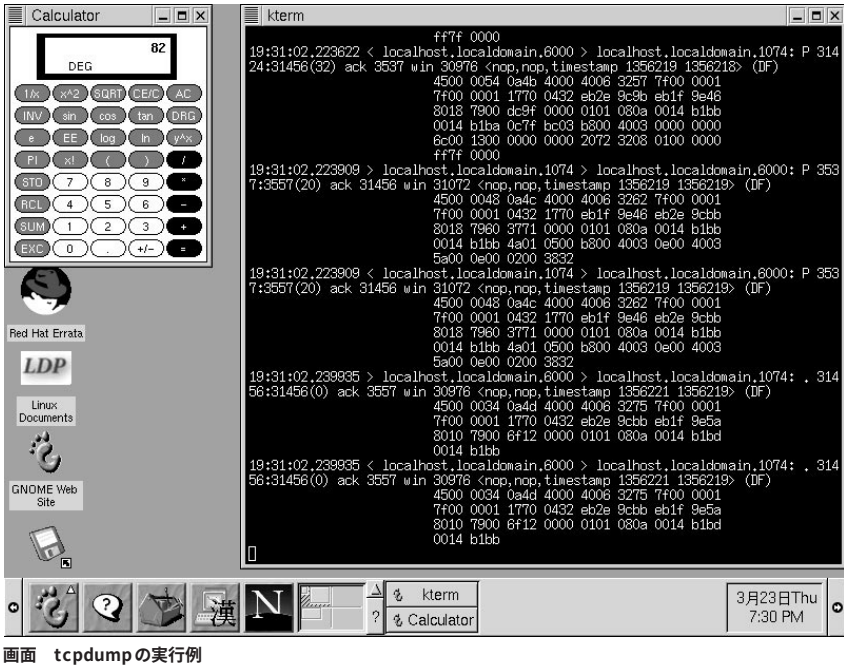
さて、Xプロトコルを観測するために、環境変数DISPLAYに「127.0.0.1:0.0」をセットして、電卓プログラム(/usr/X11R6/bin/xcalc)などのXクライアントを実行してみよう。Red Hatの場合、loという名前のループバックインターフェイス（127.0.0.1）が使用されるので、-iオプションでインターフェイスを指定する。また、送り先（dst）が127.0.0.1で、ポート番号が6000番のXプロトコルのみを表示するためのオプションも指定しておこう。

```
# tcpdump -i lo dst 127.0.0.1 and port 6000
```

プロトコルのすべての内容を見るためには、

```
# tcpdump -l -s 512 -x -i lo dst 127.0.0.1 and port 6000
```

とするとよい。ここでは、-xオプションで16進数表示の指定を行い、-sオプションで、512バイト表示するようにしている。



画面　tcpdumpの実行例

ある。

Windows 3.1からWindows 95へとOSがバージョンアップする際に、デスクトップが大きく変更されたように、時代とともに見かけ（ポリシー）の流行は変わる。しかし、基本機構（メカニズム）は変らない。KDEやGNOMEなどの新しいデスクトップ環境を見るたびに、Xのウィンドウシステムの設計が正しかったことを思い知らされる。

Xlibとツールキット

さて、XではXプロトコルと呼ばれる通信方法で描画などが行われ、基本機構を提供する下層レイヤと、見た目（ポリシー）を決定するレイヤが別になっていることは理解できたかと思う。

次は、アプリケーションプログラムがどのようにそれを使っているのかを見てみよう（**図1**）。

1. Xプロトコルを直接扱う

Xプロトコルは、単に情報の送り方を決めたものであるため、それを実際に送るにはなんらかのプログラムを作成しなくてはならない。このプログラムはたとえば**リスト1**に示すように、ネットワークにデータを送出する関数を用いて作成できる。**リスト1**のputN()関数は、ネットワーク上にNバイト送信する関数で、getN()関数はNバイト受信する関数である。このプログラムでは接続を確立するだけであるが、ウィンドウを開く関数や円弧も同様にして作成することができる。しかしながら、単に接続するのにいちいちこれだけのソースコードを作成するのは非常にたいへんである。この手間を削減するためのライブラリが「Xlib」である。

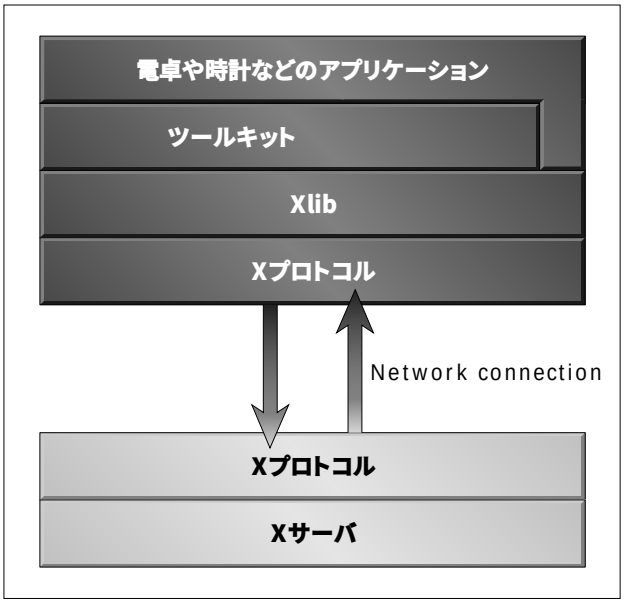


図1　Xでのレイヤ構成

2. Xlib

Xlibを使うと、**リスト1**に示す内容と同じことを、

```
XOpenDisplay("localhost:0.0");
```

という1行で実現できる。Xプロトコルを詳細に理解しなくても、プログラミングをすることができるため便利だ。

ただし、Xlibはあくまでも Xプロトコルを簡易に扱うライブラリである。そのため、ボタンを作るにしてもいちいち外枠を描いたり、中の文字を描画したりしなければならない。そして、ボタンが押されたかどうかを検出するにも、マウスのボタンがクリックされた座標がボタンの内部の座標であるかどうかを判断する必要があり、Xlibだけでプログラミングをするのは、かなりの忍耐が必要になる。

3. ツールキット

前述のようにXlibだけを使ったプログラム開発は、アセンブリ言語でプログラムをするようなもので、開発効率はかなり落ちてしまう。そのため通常は「ツールキット」と呼ばれるライブラリが使用される。

このツールキットには、MITからサンプルとして提供されている「アテナウィジェット」と「Xt」を組み合わせたものやMotifベースのものなど多数のものがあるが、最近ではGIMPやGNOMEで使われる「GTK+」が有名である。

これらのツールキットライブラリでは、ボタンやスクロールバーなどの部品が用意されていて、簡単なプログラムでその部品を使用することができるようになっている。

ツールキットを使う場合、電卓や時計などのアプリケーションは**図1**のようにして、Xサーバと通信することになる。

来月号の予告

今月は、プログラミングをする観点からのXの思想とその概要を見てきた。来月は、Xlibを使って、Xがどのようにしてネットワークトラフィックを節約しているかなどを、実際のプログラミングを通して解説してみたい。

リスト1　Xのコネクションを開く例

```
wint X11_OpenDisplay(char *hostname)
{
        long            len;
        char            vender_buf[4096];
        char            *p;
        unsigned long   id_base;
        unsigned long   id_mask;
        unsigned long   root_window;
        unsigned short  vender_len;
        unsigned char   n_format, n_screen, n_depth, n_visual;
        int             i, j, k;

        if( (fd=os_net_client_open(hostname, 6000 )) == -1 ){
                os_message("Can't open to server");
                return -1;
        }

        put1( 0x6c );   /* LSB */
        put1( 0 );      /* not use */
        put2( 11 );     /* X major Version 11*/
        put2(  0 );     /* X minor Version  0 */
        put2(  0 );     /* auth name length */
        put2(  0 );     /* auth data length */
        put2(  0 );     /* not use */
        put_flush();
        if( get1() == 0 ){
                os_message("Can't connect to X server");
        }
        (void)get1();
        os_message("X major version = %d", get2() );
        os_message("X minor version = %d", get2() );
        len = get2();
        os_message("X Release = %d", get4() );
        id_base =               get4();
        id_mask =               get4();
        /* buffer size */       get4();
        vender_len =            get2();
        /* max request len */   get2();
        n_screen =              get1();
        n_format =              get1();
        /* image byte order */  get1();
        /* format-bit-order */  get1();
        /* scaleline-unit */    get1();
        /* scaleline-pad */     get1();
        /* min-keycode */       get1();
        /* max-keycode */       get1();
        /* not use */           get4();
        for(p=vender_buf, i=0; i<vender_len; i++){
                *p++ = get1();
        }
        *p = '\0';
        for(i=0; i< ( ( 4 - ( vender_len % 4 ) ) % 4 ); i++){
                get1();
        }
        /* LISTofFORMAT */
```

右ページへ続く↗

```
        for(i=0; i<n_format; i++){
                int depth, bits;
                depth =                 get1();
                bits =                  get1();
                /* scanline-pad */      get1();
                /* not use */           get1();get1();get1();
                /* not use */           get1();get1();
                os_message("depth=%d bits=%d", depth, bits );
        }
        /* LISTofSCREEN */
        for(i=0; i<n_screen; i++){
                root_window             = get4();
                /* default_colormap */    get4();
                /* white_pixel */         get4();
                /* blacke_pixel */        get4();
                /* input mask */          get4();
                /* width  (pixel) */      get2();
                /* height (pixel) */      get2();
                /* width  (mm) */         get2();
                /* height (mm)*/          get2();
                /* min-installed-maps */  get2();
                /* max-installed-maps */  get2();
                /* root-visual */         get4();
                /* backing-stores */      get1();
                /* save-unders */         get1();
                /* root-depth */          get1();
                n_depth =                 get1();
                for(j=0; j<n_depth; j++){
                        /* depth */               get1();
                        /* not use */             get1();
                        n_visual =                get2();
                        /* not use */             get4();
                        for(k=0; k< n_visual; k++){
                                get4(); get1(); get1(); get2();
                                get4(); get4(); get4(); get4();
                        }
                }
        }
        os_message("vender=[%s]", vender_buf );
        os_message("root_window=%08lx id_base=%08lx(%08lx)",
                                root_window, id_base, id_mask );
        return root_window;
}
```

※ 上記のプログラムは、あくまでもXプロトコルの理解を助けるためのサンプルプログラムです。

※ プログラムを実行すると、Xプロトコルのバージョン(Version 11)や、リリース番号そして、Xサーバのベンダ名などが表示されます。

※ このプログラムで、ネットワークにアクセスするために使用しているos_net_client_open関数や、メッセージを表示するために使用しているos_message関数は、本誌3月号の本連載で使用したものと同じです。

特別講座

ステップアップC言語

インクルードとライブラリ

Xのプログラムをコンパイルする場合、各種のヘッダファイルをインクルードしたり、さまざまなライブラリをリンクする必要がある。今回はこのような場合どのようにしてインクルードやリンクをすればよいかを見てみる。

■インクルードパスの指定

たとえば、本文で解説したXlibを使う場合には、そのヘッダファイルを次のようにインクルードする必要がある。

```
#include <X11/Xlib.h>
```

このファイルは、Linuxディストリビューションや FreeBSDなどの場合、

```
/usr/X11R6/include/X11/Xlib.h
```

に置かれることが多い。gccは、デフォルトでは/usr/includeしか検索しない場合が多く、何も指定を行わないとXのインクルードファイルを見つけることができない。

これを解決するためには、gccのコマンドラインオプションとしてインクルードパスを追加する。「-I」オプションを使って、

```
$ gcc -I/usr/X11R6/include  ...
```

というように指定すれば、標準でインクルードされるパス（/usr/include）だけでなく、/usr/X11R6/includeも検索対象となる。

■ライブラリパスの指定

さて、インクルードの指定をしてコンパイルが成功しても、今度はリンクの段階でエラーが出る。たとえば、

```
$ gcc -I/usr/X11R6/include sample.c
```

を実行すると、リストのようにXlibにある関数がリンクできないというエラーメッセージが出力されるはずだ。

Xlibはlibx11.aというファイル名になっていて、

```
/usr/X11R6/lib/libX11.a
```

にある。そこでgccの「-L」オプションを使い、

```
-L/usr/X11R6/lib
```

というように、リンクするライブラリがあるディレクトリを指定する。また、そのディレクトリの中のどのライブラリを使うかを、「-lX11」というオプションで指定する。ここで「-l名前」のように指定すると、「lib名前.a」というライブラリがリンクされることになる。たとえば、libXt.aをリンクしたい場合は「-lXt」とすればよい。

なお、このような指定をいちいちコマンドラインから行うのが面倒な場合は、次のようなMakefileを作成しておくとよい。

```
INCDIR=/usr/X11R6/include
LIBDIR=/usr/X11R6/lib
CC=gcc -Wall -I$(INCDIR)
LINK=gcc -L$(LIBDIR)
LINK_OPT=-lX11

a.out : sample.o
                $(LINK)  sample.o
$(LINK_OPT)
sample.o:  sample.c
        $(CC) -c sample.c
```

■ライブラリの作成方法

さて、ついでにどのようにするとライブラリを作成できるかも見てみよう。

たとえば、xxx.cというファイルで、次のような関数が定義されていたとする。

```
char xxx(void){ return 'a'; }
```

同様に、yyy.cというファイルでも

```
char yyy(void){ return 'b'; }
```

という関数が定義されている場合を考える。この場合、

```
gcc -c xxx.c ;  gcc -c yyy.c
```

とすることにより、xxx.oとyyy.oという2つのオブジェクトが作成されるが、この2つのオブジェクトをlibab.aというライブラリにまとめるには、arコマンドとranlibコマンドを使って、

```
ar r libab.a xxx.o yyy.o
ranlib libab.a
```

とする。なお、このライブラリを使用するmain.cをコンパイルしてリンクするには、

```
gcc main.c libab.a
```

というようにすればよい。

リスト　Xlibにある関数がリンクできないというエラーメッセージ

```
/var/tmp/ccl113211.o: In function `main':
/var/tmp/ccl113211.o(.text+0x2f): undefined reference to `XOpenDisplay'
/var/tmp/ccl113211.o(.text+0xcf): undefined reference to `XCreateSimpleWindow'
/var/tmp/ccl113211.o(.text+0xe4): undefined reference to `XMapWindow'
/var/tmp/ccl113211.o(.text+0xf0): undefined reference to `XFlush'
```

PostgreSQLを極める

前回までの2度の解説で、PostgreSQLの運用管理の基礎とセキュリティに関することを説明した。とりあえずPostgreSQLを起動していろいろ試してみる分には、特に必要なことはないと感じられるかもしれない。しかしPostgreSQLを継続的に活用するつもりなら、大事なことがまだ残っている。メンテナンスだ。

第7回　運用と管理 ―メンテナンス編―

文：片岡裕生
Text : Hiroki Kataoka

PostgreSQLを実務などに応用しようとすると、長期的に安定した環境を維持する必要があります。一般的にはバックアップなどの障害対策などが考えられますが、PostgreSQLの場合には、バックアップのほかに“VACUUM”（バキューム）という作業も必要です。

今回はこのVACUUMや、バックアップの方法などを紹介します。

レコード更新のメカニズム

PostgreSQLではレコードの内容が更新される場合、レコードの内容を物理的に上書きしたりはしません。つまり更新後のレコードの内容は、ディスク上の新しい位置に追加して書き込まれます。このため、更新前の内容は無駄な領域として残ります（**図1**）。

この仕様は、現在のPostgreSQLにとっては避けられないことの1つなのですが、残念ながら次のような大きな副作用を持っています。

・レコードを更新するだけでも、物理的なディスクの使用量が増加していく

1レコードを更新するたびに物理的に新しいレコードを1つ費やしていくわけですから、当然の副作用だとおわかりいただけると思います。つまり、たとえばたった5レコードしかないテーブルでさえも、更新を繰り返すうちに膨大なディスク容量を消費するようになってしまうということです。

もちろん、無駄な領域を自動的に再利用してくれればこのような副作用はないのですから、疑問に思う方もいるかと思いますが、今のPostgreSQLではそのようになっていません（近いうちに実現されるかもしれません）。このままでは、無駄な領域は永遠に利用されずにディスク領域を占有し続けます。

では、無駄な領域を処分してディスク領域を解放させる方法はないのでしょうか。

実は“VACUUM”（バキューム）がこれを行います。

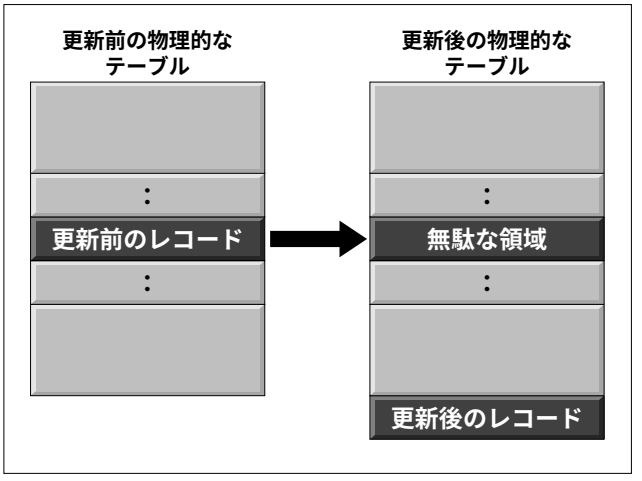


図1　レコード更新のメカニズム

VACUUMとは

VACUUMとは、更新が起きるデータベースに対して、定期的に必ず行わなければならないPostgreSQL特有の管理作業のひとつです。

VACUUMには、使用されていない無駄な領域を開放するという大切な役目（**図2**）がありますが、それ以外にも重要な役目があります。それは、問い合わせの最適化を行うのに必要な情報を集めることです。

ここで、問い合わせの最適化について簡単に説明しておきます。

問い合わせの最適化

一般にデータベースシステムでは大量のデータを扱うことができます。しかし、大量のデータを扱えるというだけでは十分とは言えず、その中から目的のデータをいかに効率よく見つけ出すかということも重要になります。どのように検索したら最も効率よく目的のデータを見つけ出せるかを考えるのが、問い合わせの最適化です。たとえば、大勢の人の情報を集めた“person”というテーブルがあり、以下のような3つのカラムを持っているとします。

- **番号**
  カラム名：id
  データ型：INTEGER型
  データサイズ：4バイト
  データのばらつき：重複なし
- **名前**
  カラム名：namae
  データ型：TEXT型
  データサイズ：主に8バイト以上
  データのばらつき：重複なし
- **性別**
  カラム名：sex
  データ型：TEXT型
  データサイズ：2バイト
  データのばらつき：“男”50%、“女”50%

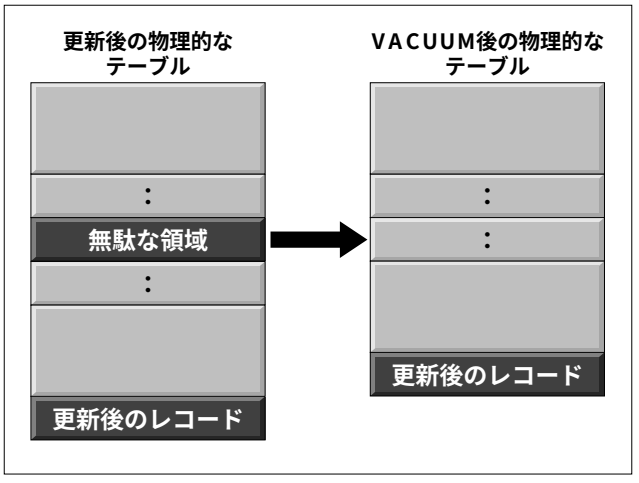


図2　VACUUMによる無駄な領域の開放

そして、このテーブルからある人のレコードを探し出すために、次のようなSQL文が与えられたとします。

```
SELECT * FROM person
WHERE
    id = 9 AND
    namae = '片岡 裕生' AND
    sex = '男';
```

なお、問い合わせの最適化の説明をわかりやすくするために、上のSQL文ではすべてのカラムに条件を与えています。

もしも、どのカラムにもインデックスがないとしたら、テーブル内のすべてのレコードを順々に調べていく以外に目的のデータを見つける方法はありません。したがって、この場合には問い合わせの最適化などという話は出てきません。

しかし、もしもインデックスがあったならどうでしょう。インデックスを使ったほうが目的のデータを効率よく探せるということはだれにでも容易に想像できますね。ここで、もう少し話をおもしろくしてみます。もしも、3つのカラムそれぞれにインデックスがあった場合にはどうなるでしょうか。インデックスはその性質上、すべてのインデックスを同時に使うということはできませんので、どれかひとつを選んで使うことになります。そして、どのインデックスを使うかによって、検索の効率も変わってきます。このような場合にどのインデックスを使うか決定することを、問い合わせの最適化といいます。

ではなぜ、利用するインデックスによって検索の効率が変わって来るのでしょうか。

先のSQL文では、番号が9で、名前が“片岡 裕生”で、性別が“男”のデータを探そうとしています。ここでもしも番号カラムのインデックスを利用したならどうなる

でしょうか。

まず番号カラムのインデックスによって、番号が9のレコードが高速に見つけ出されます。そして番号が9のレコードが1件見つかったとすれば、そのレコードの名前と性別が“片岡 裕生”と“男”であるかどうかをチェックして終わりです。つまり、インデックスを利用することによってアクセスすべきレコードが1件に絞られたわけで、テーブルに対してはたった1レコードのアクセスで済んでいます。

もしも名前カラムのインデックスを利用したならどうなるでしょうか。

まず名前カラムのインデックスによって、名前が“片岡裕生”のレコードが高速に見つけ出されます。そして名前が“片岡 裕生”のレコードが1件見つかったとすれば、そのレコードの番号と性別が9と“男”であるかどうかをチェックして終わりです。この場合も、テーブルに対してはたった1レコードのアクセスで済んでいます。

それでは、性別カラムのインデックスを利用した場合にはどうなるでのしょうか。

まず性別カラムのインデックスによって、性別が“男”のレコードが高速に見つけ出されます。といっても性別が“男”のレコードは全体の50％もありますので、残りの条件である番号と名前を調べなければならないレコードが大量に残ってしまったという状態です。そして、これら残ったレコードのすべてに対して、番号と名前が9と“片岡 裕生”であるかどうかをチェックして行かなければなりません。つまりこの場合は、インデックスを利用しても全体の2分の1にしか絞ることができなかったわけで、結局その後にテーブル全体の半数のレコードに対してアクセスしなければならなくなっています。

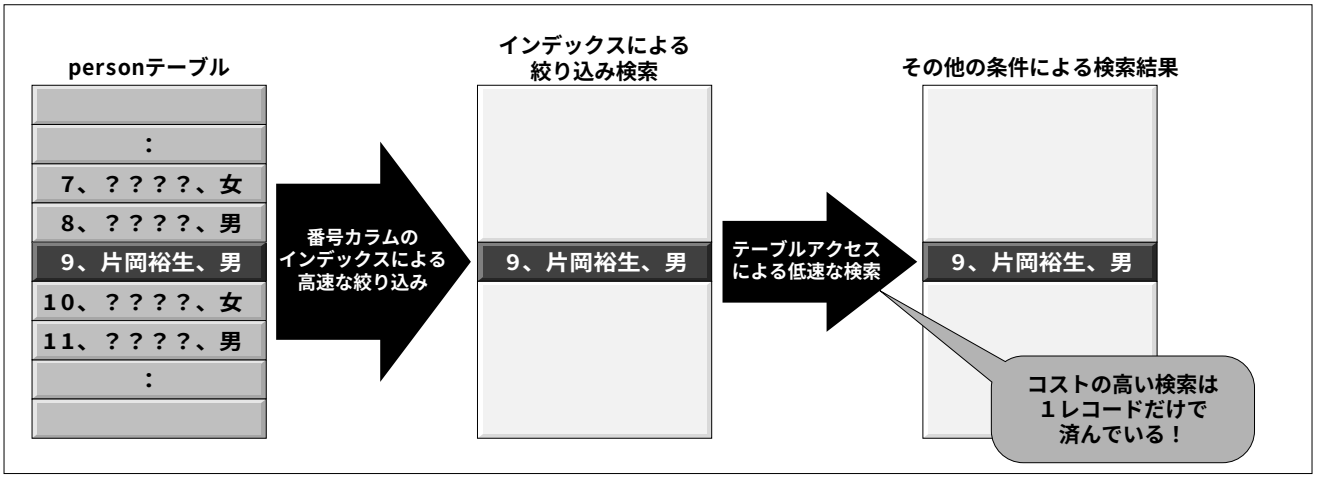


図3　効率的なインデックスの使用例

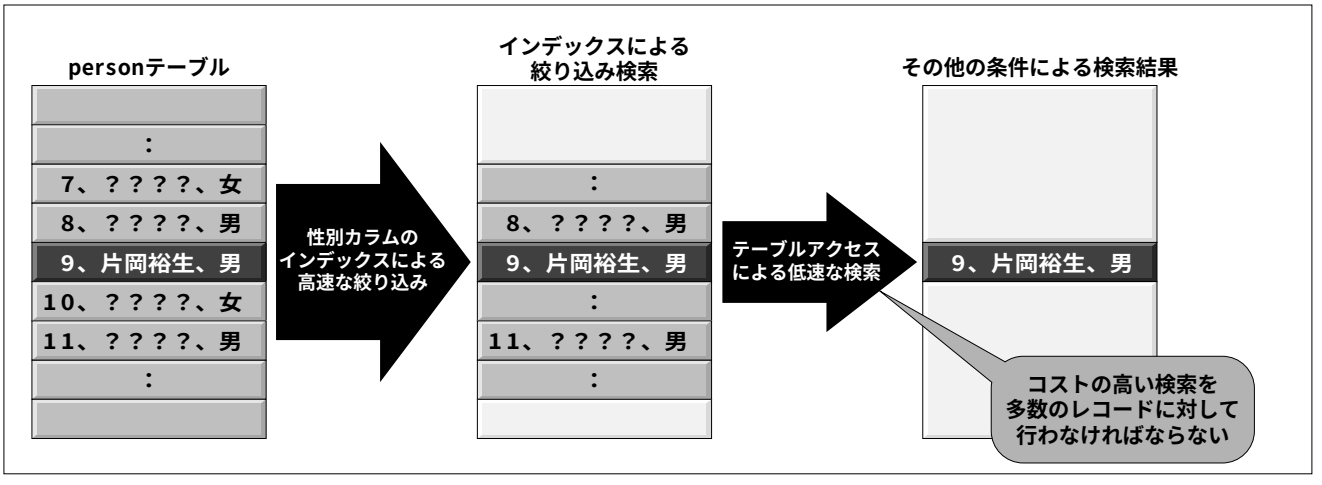


図4　非効率的なインデックスの使用例

データベースの検索にかかるコストのうちの代表的なものは、テーブルに対するアクセス量です。番号カラムのインデックスと名前カラムのインデックスを利用したそれぞれの場合では、テーブルに対してはたった1レコードのアクセスしか必要でありませんでした（**図3**）。それに対して性別カラムのインデックスを利用した場合では、テーブル内のレコードの半数にアクセスする必要がありました（**図4**）ので、この場合の検索効率は非常に悪いといえます。

ちなみに、番号カラムのインデックスと名前カラムのインデックスを利用したそれぞれの場合でも、厳密には優劣を付けることができます。番号カラムのデータサイズが4バイト、名前カラムのデータサイズが8バイト以上。つまり、同じ件数のデータを演算する場合であっても、名前カラムは番号カラムの2倍以上のデータ量を必要としていることになります。データ量が多ければディスクI/Oに費やす時間も多くなりますので、よりパフォーマンスが悪くなるといえます。ただし、この点はPostgreSQLによる問い合わせの最適化には加味されないようですので、データベースを設計する時点で、できるだけ小さなサイズのインデックスを作成するように心がけることも重要です。

今説明した例は、ひとつのテーブルに効率のいいインデックスと効率の悪いインデックスがあった場合という、比較的説明しやすいというだけの例ですので、これが最適化のすべてではありません。もっと複雑で現実的な例としては、複数のテーブルを結合した問い合わせなどが考えられます。この場合は、どのテーブルから検索を開始したらいいのかなどが、重要な最適化の要素になります。

VACUUMの必要性

話を戻しますが、VACUUMの重要な機能として、問い合わせの最適化に必要な情報を集めることを挙げました。問い合わせを最適化するためには効率の良いインデックスを利用することが必要ですし、複数のテーブルを結合した問い合わせなら、どのテーブルから検索を開始するべきかも選択する必要があります。しかし、いったいどれを選択すれば本当に効率が良いのかということは、実際のテーブルのレコード件数やデータのばらつきなどを調べないことには困難です。そこでVACUUMの出番です。

VACUUMを行うと、使われていない無駄な領域を開放すると同時に、テーブル内のレコード件数なども調査します。また、データのばらつき具合を予想するための情報も集めます。本来なら、テーブル内のレコード件数などは常にカウントされているべきとは思いますが、現在のPostgreSQLではVACUUM時にのみ、これらの情報を調査する仕様になっています。ですから、テーブルの内容がある程度更新されるたびに、定期的なVACUUMを実行する必要があるわけです。

では、VACUUMを実行しなかった場合にはどうなるのかを説明しましょう。無駄な領域が開放されないのは当然として、問い合わせの最適化はいったいどうなるのでしょうか。残念ながらPostgreSQLは、最適ではない手順で問い合わせを実行することが多くなります。たとえば先ほどの例を実際に試してみると、VACUUM（正確には“VACUUM ANALYZE”）をしなかった場合には性別カラムのインデックスを使いました。

VACUUMの必要性をおわかりいただいたところで、使い方を説明します。

VACUUMの使い方

VACUUMはSQL文のひとつとして、psqlなどのSQLインターフェイスから実行できます。VACUUM文の書式は次のとおりです。

```
VACUUM
    [VERBOSE] [ANALYZE]
    [<テーブル名>[(<カラム名>, ...)]]
```

“VERBOSE”キーワードを指定すると、VACUUMの実行状況を表示するようになります。しかし、それ以上

```
% psql ascii7
Welcome to the POSTGRESQL interactive sql monitor:
        :
ascii7=> VACUUM VERBOSE ANALYSE person;
NOTICE:  --Relation person--
NOTICE:  Pages 34: …
NOTICE:  Index person_sex_ind: Pages 14 …
NOTICE:  Index person_namae_ind: Pages 22 …
NOTICE:  Index person_id_ind: Pages 17 …
VACUUM
```

画面1　VACUUM文の実行例

```
% vacuumdb -v -z -t person ascii7
NOTICE:  --Relation person--
NOTICE:  Pages 34: …
NOTICE:  Index person_sex_ind: Pages 14 …
NOTICE:  Index person_namae_ind: Pages 22 …
NOTICE:  Index person_id_ind: Pages 17 …
```

画面2　vacuumdbコマンドの実行例

には一般向けの有用な情報は表示しませんので、実行状況を眺めたい場合以外には必要ないと思います。

VACUUMはデフォルトでは、テーブル内の無駄な領域を開放し、レコード数などのテーブルの大きさに関する情報のみを調査しますが、“ANALYZE”キーワードを指定すると、カラム内のデータのばらつきぐあいを推測する情報も調査するようになります。このときに<カラム名>を指定していれば、指定したカラムについてのみデータのばらつきぐあいを調査し、<カラム名>の指定がない場合には、テーブル内のすべてのカラムについて調査します。PostgreSQLが問い合わせの最適化をより正確に行うようにするために、できるだけこの“ANALYZE”キーワードは指定したほうがいいようです。

なお<カラム名>は、“ANALYZE”キーワードと同時に指定しなければなりません。

<テーブル名>には、VACUUMを実行したいテーブル名を指定します。テーブル名を指定しなかった場合には、接続しているデータベース内のシステムテーブルを含んだすべてのテーブルが対象になります。

VACUUMの定期的な実行は、テーブルが更新されない限り必要ありません。無駄なメンテナンス作業をできるだけ避けたいと考えるのであれば、VACUUMコマンドにきちんとテーブル名を指定して、更新されたテーブルだけをVACUUMすべきですが、データベース自体のサイズが比較的小さいのであれば、テーブル名を省略して、すべてのテーブルをVACUUMしてしまってもかまわないと思います。

画面1は、“ascii7”テーブルに接続して“person”テーブルにVACUUMを実行しているようすです。ここではあえて、“VERBOSE”キーワードも指定してみました。“NOTICE:”で始まっている行が、“VERBOSE”キーワードにより表示された情報です。この例ではテーブル“person”について処理した後、引き続き3つのインデックスについても処理していることがわかります。

VACUUMの実行は、OSのコマンドラインから直接行うことも可能です。このために“vacuumdb”というコマンドが用意されています。これを利用すれば、cronによる深夜の自動VACUUMなども可能になります。vacuumdbコマンドの書式は次のとおりです。

```
vacuumdb [<オプション>] [<データベース名>]
```

<オプション>には**表1**に挙げたものが指定できます。これらのうち、データベース接続に関するオプションは、psqlコマンドなどの他のコマンドと同じ仕様になっています。

<データベース名>には接続するデータベースを指定します。省略した場合には、現在のOS上のユーザーと同じ名称のデータベースに接続を試みます。

画面2は、先ほどの**画面1**のVACUUM文と同じことをvacuumdbコマンドで行ったようすです。

なおVACUUMは、データベースに接続さえできればだれでも実行することができます。テーブルのオーナーである必要はありません（この仕様が妥当かどうかは別ですが）。

VACUUMの定期的な実行

PostgreSQLデータベースのパフォーマンスを維持するためには、定期的なVACUUMが必要なことは説明しました。ここでは、UNIX系OSにおけるVACUUMの定期的な実行方法を紹介します。

UNIX系OSであれば、コマンドを定期的に起動する“cron”という機能が搭載されています。これはVACUUMの定期的な実行にも利用することが可能です。

なお以下の説明では、筆者の手元にあるVine Linuxの場合を基にしています。

まず、定期的にVACUUMを実行させるために、vacuumdbコマンドの起動命令をcronに登録します。cronの登録内容はOSのユーザー単位で管理され、登録されたコマンドは登録したユーザーの権限で起動されます。ですからcronの設定を行う前に、vacuumdbコマンドを

データベース接続に関するオプション	
-h <ホスト名>	接続先のPostgreSQLサーバのホスト名を指定する。省略した場合は自ホストのPostgreSQLサーバに接続する
-p <ポート番号>	接続先のPostgreSQLサーバのポート番号を指定する。省略した場合は5432
-u	ユーザー名とパスワードの入力を促すプロンプトを表示する。省略した場合は現在のOS上のユーザー名で接続しようとするが、パスワードが必要な場合にはエラーとなる
データベース接続に関するオプション	
--table <テーブル名>	VACUUMするテーブルを指定する。このオプションを省略した場合には、データベース内のすべてのテーブルが対象となる。カラムも指定したい場合には、<テーブル名>として“テーブル名(カラム名, ...)”という形式を指定する
-z --analyze	VACUUM文の“ANALYZE”キーワードと同じ
-v --verbose	VACUUM文の“VERBOSE”キーワードと同じ

表1　vacuumdbコマンドのオプション

起動したいユーザーでログインし直します。ここではPostgreSQLのスーパーユーザー（例として“ postgres ”）でログインしているものとします。

cronの設定を行うにはcrontabコマンドを利用します。

```
% crontab -e
```

cronコマンドに“ -e ”オプションを指定して実行すると、cronの設定を行うためにviなどのテキストエディタが起動し、テンポラリファイルが開かれます。このテンポラリファイル内に定期的に起動したいコマンドを記述してエディタを終了すれば、自動的にcronに登録されます。

テンポラリファイルに記述する書式は次のようになっています。

```
<分> <時> <日> <月> <曜日> <コマンド>
```

＜分＞、＜時＞、＜日＞、＜月＞、＜曜日＞には、コマンドを起動したい日時を指定します。すべての日時や曜日に該当する“ ＊ ”（アスタリスク）も指定できます。たとえば毎日9時00分にコマンドを起動したい場合には“ 0 9 ＊ ＊ ＊ ”と、毎週月曜日の午前4時30分にコマンドを起動したい場合には“ 30 4 ＊ ＊ 1 ”と指定します。なお、＜曜日＞には0（日曜日）～6（土曜日）の数字を指定します（OSによっては7が日曜日を意味する場合もあります）。

＜コマンド＞には起動したいコマンドを指定します。このコマンドが起動されるときの環境は、ユーザーがふつうにログインしたときの環境とは異なります（.bashrcなどでの設定は有効にならない）ので、コマンドの実行に必要な環境変数の設定なども必要になります。このため、環境設定とコマンドの実行をひとつにしたシェルスクリプトを別途作成し、このシェルスクリプトを＜コマンド＞に指定することもよくあります。

先ほどの画面2と同じことを毎日午前4時00分にcronで自動実行させるには、**リスト1**のような1行をcronに登録すればよいことになります（リスト1では環境変数の設定も＜コマンド＞に含めてしまっています）。

なお、cronではコマンドの実行がエラーになるか、正常であってもなんらかのメッセージが出力されると、そのコマンドを登録したユーザーにメールが届くようになっています。VACUUM中に起きたエラーメッセージなどもこのメールに含まれることになりますので、cronを登録したユーザーはメールのチェックも怠らないようにします。ふだん利用しているメールアドレスに転送するのも良いでしょう。

なおcronの詳細は、別途オンラインマニュアルなどを参照してください。より柔軟な日時の指定方法などが記述されています。

VACUUMの注意事項

VACUUMを実行するにあたり、いくつか注意事項があります。

VACUUMの実行に要する時間は、そのテーブルを一からINSERT文などで作成した場合とほぼ同等と思ってください。ですから、たとえば数百万レコードもあるようなテーブルのVACUUMには相当な時間がかかります。

また、VACUUM中のテーブルには完全な排他ロックがかかります。たとえ参照のみのトランザクションであっても、VACUUM中のテーブルにはアクセスすることができません。きちんとロックがかかるわけですからデータベースの運用中にVACUUMを行ってもかまわないのですが、もしもテーブルが巨大で、VACUUM完了までに数時間も要するような場合には、事実上運用中のVACUUMはあきらめるしかないでしょう。

PostgreSQLを効率良く活用しようと思うのなら、巨大なテーブルの更新はできるだけ控えるようにして、ふだんのVACUUM対象からは除くようにします。テーブルの更新が頻繁でなければ、そのテーブルに対するVACUUMも頻繁に行う必要はないからです。その代わりに、データベースの運用を停止してもかまわない定期メンテナンス時などに、巨大テーブルのVACUUMを実行するのです。

バックアップ

PostgreSQLを実務に利用しようとするとデータベースのバックアップは欠かせません。ここでは、PostgreSQLデータベースのバックアップ方法を説明します。

PostgreSQLには2通りのバックアップ方法があります。

リスト1　cronによる定期VACUUMの設定例（実際には改行していません）

```
0 4 * * * PATH=${PATH}:/usr/local/pgsql/bin LD_LIBRARY_PATH=/usr/local/pgsql/lib /usr/local/pgsql/bin/vacuumdb -z -t
person ascii7
```

ひとつは、PostgreSQLのデータベースが格納されているディレクトリごと、tarコマンドなどでテープなどにコピーしてしまう方法。これはPostgreSQLが、データベースの保管方法として通常のUNIXファイルシステムを利用しているからできることで、非常にシンプルでわかりやすい方法です。しかしデメリットもあります。バックアップを行う前にPostgreSQLサーバを停止させなければなりません。さらに、PostgreSQLのバージョンが上がるとデータベースの格納形式も変わることがありますので、昔に取っておいたバックアップが今は利用できない、ということもありえます。

もうひとつの方法は、PostgreSQLが用意している“pg_dump”あるいは“pg_dumpall”というコマンドを利用する方法です。pg_dump／pg_dumpallコマンドはその名称から想像できるように、PostgreSQLデータベースの内容をダンプするコマンドです。ダンプした結果は通常のテキストファイルで、その内容はSQL文の羅列です。そしてこのSQL文の羅列を実行すれば、いつでも元のデータベースを復元できます。

バックアップした結果は通常のSQL文ですので、PostgreSQLのバージョン違いによる復元不能はまず考えられませんし、万一そのようなことになったとしても、通常のテキストファイルですからエディタで修正するなどの対処も可能です。そしてこのコマンドを用いた場合の最大の利点は、データベースの運用中でもバックアップが取れるという点です。

以下では、pg_dump／pg_dumpallコマンドによるバックアップの方法を紹介します。

pg_dumpコマンド

pg_dumpコマンドは、指定した1つのデータベースをダンプするコマンドです。すべてのデータベースをダンプしたい場合には、後述するpg_dumpallコマンドを利用します。

pg_dumpコマンドの書式は次のとおりです。

```
pg_dump [<オプション>] <データベース名>
```

＜オプション＞には主に**表2**に挙げたものが指定できます。

＜データベース名＞にはダンプするデータベースを指定します。

pg_dumpコマンドのダンプ結果には、ダンプしたデータベース自体を作成するSQL文（CREATE DATABASE文）は含まれていません。ですからpg_dumpコマンドのダンプ結果からデータベースを復元する際には、あらかじめ、復元先となる空のデータベースを作成しておく必要があります。しかし、この特徴を利用すれば、あるデータベースの内容を他のデータベースにそっくりコピーすることが可能になります。

画面3は“ascii7”データベースの内容を、pg_dumpコマンドを利用して“ascii7a”データベースにコピーしているようすです。最初のpg_dumpコマンドで“ascii7”データベースを“ascii7.dump”ファイルにダンプし、次のpsqlコマンドで新しい“ascii7a”データベースを作成し、

データベース接続に関するオプション	
-h＜ホスト名＞	接続先のPostgreSQLサーバのホスト名を指定する。省略した場合は自ホストのPostgreSQLサーバに接続する
-p＜ポート番号＞	接続先のPostgreSQLサーバのポート番号を指定する。省略した場合は5432
-u	ユーザー名とパスワードの入力を促すプロンプトを表示する。省略した場合は現在のOS上のユーザー名で接続しようとするが、パスワードが必要な場合にはエラーとなる
ダンプに関するオプション	
-t＜テーブル名＞	ダンプするテーブルを指定する。このオプションを省略した場合には、データベース内のすべてのテーブルが対象となる
-s	CREATE TABLE文などのスキーマ定義だけをダンプする
-x	アクセス権をダンプしない
-a	データのみをダンプする（CREATE TABLE文などのスキーマ定義はダンプしない）
-c	スキーマの初期化指示をダンプに含める（CREATE TABLE文に先立つDROP TABLE文など）
-d	データを、INSERT文を利用した書式でダンプする（カラム名の指定は省略される）。このオプションを指定しなかった場合には、COPY文を利用した形式となる
-D	-dとほぼ同じ機能だが、INSERT文を利用した書式にカラム名の指定が含まれるようになる
-o	オブジェクトID（oid）をダンプする
-n	識別子を“"”（ダブルクォーテーション）で囲まない。このオプションを指定しなかった場合にはすべての識別子が“"”（ダブルクォーテーション）で囲まれる
-f＜ファイル名＞	ダンプ結果を出力するファイルを指定する。このオプションを指定しなかった場合には標準出力に出力する

表2　pg_dumpコマンドのオプション

最後のpsqlコマンドで“ascii7.dump”ファイルのダンプ内容を“ascii7a”データベースに復元させています。

pg_dumpallコマンド

pg_dumpallコマンドは、すべてのデータベースをダンプするコマンドです。

pg_dumpallコマンドの書式は次のとおりです。

```
pg_dumpall [<オプション>]
```

<オプション>にはpg_dumpコマンドとほぼ同じオプションが指定できますが、“-t”オプションと“-f”オプションを指定してはいけません。エラーにはなりませんが、特別な目的の場合を除いて、役には立たないでしょう。

pg_dumpコマンドとの大きな違いは、登録されているユーザーの情報もダンプされるという点です。言い換えれば、各ユーザーのパスワードもダンプされるということです。ですから、pg_dumpallコマンドのダンプ結果を保管したファイルの取り扱いには、十分に注意する必要があります。

画面4はデータベースをバックアップしているようすです。この例ではpg_dumpallによるダンプ結果を“all.dump”ファイルに保管しています。なお、バックアップファイル内のユーザー情報の機密を保護するために、事前にumaskコマンドで出力ファイルのパーミッションを調整しています。必要なら、できあがったダンプファイルをテープなどへ保管して、バックアップは完了です。ついでにpg_hba.confなどの設定ファイルもバックアップしておくと、いざというときに便利でしょう。

pg_dumpallコマンドによるバックアップを復元する場合はデータベース全体を初期化した状態で行います。データベース全体を初期化する最も簡単で確実な方法は、PostgreSQLサーバを停止した状態でデータベースディレクトリ（通常は/usr/local/pgsql/data）をサブディレクトリもろとも削除してから、initdbコマンドを実行することです。そしてPostgreSQLサーバを起動してからpsqlコマンドで“template1”データベースに接続し、バックアップ内容を実行させます（**画面5**）。

```
% pg_dump ascii7 > ascii7.dump
% psql -c "create database ascii7a" template1
CREATEDB
% psql -f ascii7.dump ascii7a
        :
   （実行のようす）
        :
EOF
```

画面3　pg_dumpを利用したデータベースのコピー

pg_dumpコマンドの便利な使い方

pg_dumpコマンドでデータベースがコピーできることは説明しました。そのほかにもpg_dumpコマンドのオプションを利用することによって、便利に活用することができます。

データベースでいろいろと開発していると、最新のデータベースの構造（スキーマ）を取っておきたい場合があると思います。たとえば、アプリケーションの改良などにより運用途中でスキーマが変わった場合などです。このような場合には、pg_dumpコマンドに“-s”オプションを付けて実行します。そうすればデータを除いたスキーマだけがダンプできます。

すでに運用中のデータがある程度蓄積されている状態で、どうしてもテーブル構造を変更したい場合には、CREATE TABLE文やALTER TABLE文、SELECT INTO文などを駆使してもできますが、シーケンスやらインデックスなども再度設定し直さなければならなくなってしまいます。いっそのことpg_dumpコマンドで該当テーブルを丸ごとダンプして、できあがったSQL文をエディタで編集してしまうというのはどうでしょうか。“-c”オプションや“-D”オプションと併用すると便利な場合もあります。

次回は

PostgreSQLの運用管理に関する解説は、ひとまずこれで終わりです。次回はWindowsとの連携について紹介したいと思っています。

```
% rm -f all.dump
% ( umask 066 ; pg_dumpall > all.dump )
```

画面4　pg_dumpallを利用した全データベースのバックアップ

```
% （全データベースの初期化）
% psql -f all.dump template1
        :
   （実行のようす）
        :
EOF
```

画面5　全データベースの復元

ACPI（Advanced Configuration and Power Interface）

文：政久忠由
Text : Tadayoshi Masahisa

前回紹介したノートPCのBIOSをACPI対応にアップグレードしたということもあり、今回はLinuxとACPIについて見てみることにしよう。でもその前に前回のNFS設定の補足もあるので、まずはこちらを片付けることにする。

NFS設定の続き

前回、ハードディスク容量が少ないという理由から、ローカルネットワーク内ではNFSを利用してワークスペースを確保し、さらにノートPCというモバイル環境で操作の手間を省くために、そのファイルシステムは一般ユーザーでも操作できるよう/etc/fstabに次のような設定を行った。

```
#/etc/fstab
linux01:/mnt/1  /mnt/1  nfs  noauto,user  0  0
```

これで一般ユーザーでも、次のようにファイルシステムをマウントすることができるようになった。

```
$ mount /mnt/1
```

しかしこの設定では、いくつか問題点というか制限が生じてしまう。それはuserオプションのデフォルト設定にある。

/etc/fstabで指定したuserオプションは、rootユーザー以外の一般ユーザーにもマウントを許可する指定だが、このオプションを指定した場合、noexec、nosuid、nodevという3つのオプションが同時に設定されるようになっている。これらの状態はmountコマンドをオプションなしで実行すると表示されるので確認してみよう。

```
$ mount
linux01:/mnt/1 on /mnt/1 type nfs (rw,noexec,nosuid,
nodev,addr=x.x.x.x,user=tadayo-m)
```

noexecはパーミッションに関わらず実行許可を与えない、nosuidはパーミッションのsuid、sgid設定を無視する、nodevはブロック／キャラクタスペシャルファイル（/devディレクトリのファイル群）解析を行わないというものだ。これらは安全策のためにデフォルト設定されているわけだが、NFSの場合、nodevオプションは問題ないものの、noexecとnosuidは場合によっては期待通りの動作をしないという現象に悩まされることになる。知っていればどうってことないのだけれど、知らないと何が何だかわからない。たとえば、そのファイルシステム上のファイルを実行しようとすると次のようなエラーメッセージが表示される。

```
$ ./sleep
```

```
bash: ./sleep: Permission denied
```

このエラーメッセージが表示されると普通はファイルシステム上のパーミッションをチェックして問題ないことを確認するハズだ。しかし問題ないにも関わらず実行できないことで、途方に暮れてしまうことだろう。僕もこのことをすっかり忘れていて、あるプログラムをmakeしようと./configureを実行したところ上記のエラーに遭遇してしまい、しばし白くなってしまった。

この問題を回避するには、マウントオプションを追加するだけだ。noexecを打ち消すにはexec、nosuidを打ち消すにはsuidを設定する。注意点としては、設定したオプションの評価は左から右へ順に行われるので、最終的に有効にしたいオプションはより右側に記述しなければならないことだ（exec、userの順に設定してしまうと、execはuserの設定で上書きされnoexecが有効になってしまう）。

/usr/binや/usr/local/binなどプログラムディレクトリをマウントするわけではない僕の環境では、suidビットが無効になろうと特に問題はないので、execの指定だけ行うことにした。

```
#/etc/fstab
linux01:/mnt/1  /mnt/1  nfs  noauto,user,exec  0  0

$ mount
linux01:/mnt/1 on /mnt/1 type nfs (rw,nosuid,nodev,
addr=x.x.x.x,user=tadayo-m)
```

NFSと時間設定

NFSでは、リモートプロシージャコールという方法で、実際のファイルの処理はファイルシステムを提供するローカル（ホスト）側のオペレーティングシステム（サーバプロセス）が担当することになる。クライアント側のプログラムからは、ローカルファイルシステム上のファイルであろうと、NFS上のファイルであろうと特に区別しないでアクセスできるようになっているのだけれど、サーバとクライアントの時刻設定がファイルのタイムスタンプに影響し、思わぬ不具合が生じることがある。というのも最初に説明したようにNFS上のファイルはサーバ側で処理され、ファイルのタイムスタンプはサーバ側の時刻で書き込まれるので、クライアント側の時刻を基にするクライアントの実際に処理を行っているプログラムは、予定より過去であったり未来であったりするタイムスタンプに遭遇することがあるのだ。ほとんどは警告メッセージだけで済むが、場合によっては大切なデータを失いかねない問題ともなり得るものだ。

これを解決するにはコンピュータ間で時刻を同期させる必要がある。管理されたネットワークでは、NTPと呼ばれる時間を同期させるための標準プロトコルを用いて、最低ローカルネットワーク内に1つ以上のプライマリサーバを設置し、他のマシンはセカンダリサーバとして設定するか、定期的にNTPクライアントを実行するなどして時刻の同期を行うのが通例だ。

僕の環境では、Linux側でntpd（NTPデーモン）を実行するか、Windows 2000のSNTP（簡易NTP）サービスに同期させるか少し悩んだのだけれど、それほど精度を求めているわけでもなく、ネットワークの遅延を気にしなくてもよいローカルネットワークということもあり、Windows 2000ドメインの適当なサーバをプライマリと位置付け、Linux側で2時間ごとにNTPクライアント（ntpdate）を実行することにした（もし2時間ごとの同期で0.5秒以上ずれるようなら同期間隔を短くする必要がある）。

ntpdateコマンドは、指定したサーバから時刻を取得しローカルマシンにセットするが、その際、settimeofday()とadjtime()という2種類のシステムコールを使い分けることができる。settimeofday()はCMOSクロック（ハードウェアクロック）とカーネルクロックの両方を設定でき、adjtime()はカーネルクロックのみを調整する。

ntpdateコマンドでは、-bオプションを指定することで、settimeofday()をコールするようになっているので、ブート時のスタートアップファイル（rc.localなど）には、“ntpdate -b server01”を指定し、定期的な同期はcronで“ntpdate server01”を実行するようにしておくとよいだろう。なお、ntpdateコマンドでオプションを指定しない場合、ずれが128ms以内であればadjtime()、それを超えるとsettimeofday()がコールされることになる（メッセージにadjustと表示されればadjtime()、stepと表示されればsettimeofday()が利用されている）。

```
# ntpdate -b athena
27 Mar 17:54:07 ntpdate[1149]: step time server
192.168.0.2 offset 1215.206364 sec
# ntpdate athena
27 Mar 17:54:16 ntpdate[1152]: adjust time server
```

```
192.168.0.2 offset -0.005934 sec
```

またntpdateコマンドの場合、時刻を徐々に調整していくといった機能が提供されているわけではないので、データベースサーバを始め時刻の管理が重要なマシンでは、きちんとNTPサーバを導入し、適切な時刻調整を行うよう心がけてほしい。

それにしてもPCの時間は当てにならない。最近のOSは起動時にCMOSクロックから時刻を取得した後は、CPUのカウンタなどを利用してカーネル独自に時間を刻むようになっているので、settimeofday()とadjtime()という2つのシステムコールが用意されているのだ。CPUは動作周波数が高いためCMOSクロックよりも精度は高いといわれているが、今までそれを実感したことはなく、カーネルクロックもよくずれるという印象しかない。結局、基にしょぼいクオーツクロックジェネレータの出力を使用しているからなのか、PC内のノイズが問題なのか、僕のマシンがハズレだからなのかはわからないけれど…。そのうちTVチューナーを搭載しているLinuxマシンでNHKなどの時報同期を策略してみることにしよう。あっ、でもその前に音が出るようにしないと…。

パワーマネージメント

コンピュータのパワーマネージメント機能は、その黎明期から求められていた。コンピュータは電子回路を通過する電子の量が消費電力に直結するため（実際には電気抵抗によって熱などにエネルギー変換されることで消費されるのだけれど）、高速化に正比例して電力を消費してしまうしくみになっている。そのため材料の電気抵抗を小さくしたり、回路を微細加工し必要な電子の量を少なくしたりして、消費エネルギー、つまり発熱を抑えつつ、高速化を図ってきたわけだ。

現在、プロセッサユニット（CPU）を始め、一般コンピュータ用途のほとんどの電子回路はクロック同期により動作している。今やIntelやAMDのCPUは動作クロック1GHzにまで到達し、すでに4GHz程度の見通しが立っているようだが、クロック同期の回路の場合、その高速化の手法は動作クロックを上げることと、そのクロックに乗せて同時にやり取りするデータのバス幅の拡大が中心となる。そのため回路は動作クロックを中心に設計されているわけだが、最近では1cm四方のチップ内でさえ1つのクロックジェネレータで同期させるのは難しく、またパワーマネージメント（発熱を抑える）効果を狙い、整数演算、浮動小数点、アドレス生成などユニットごとにクロックを供給し、必要に応じて各ユニットのクロックを停止させる手法などが採用され、さらに動作クロックを段階的に調整できるような機能も盛り込まれようとしている（昔から一部にはこのような機能を持つノートPC用のCPUもあったような気もするが）。これらの手法は、CPUだけでなく高速化と低消費電力化が求められる各デバイスのチップやチップセットでも同様である。

一般的な利用の場合、topコマンドなどでCPU使用率を見ているとわかると思うが、コンピュータ（CPU）は99％以上、アイドル状態であったりする。連続的なジョブが投入され続けるサーバ用途を除き、ユーザー寄りのコンピュータ用途ではCPUを始めとするハードウェアは短期的な処理能力が求められているに過ぎないのが現状だ。この短期的な処理能力は、局所的なリアルタイム応答とも呼べるが、とにかく何かしらの処理に際して利用しているユーザーがストレスを感じないことが重要となる。だからこそ長期的な処理能力としては、はるかにオーバースペックなハードウェアの需要が発生し、供給されているのである（僕自身はあまり短期的な処理能力に固執していないので、Pentiumの75MHzから200MHzのマシンもとりあえず現役で使っていたりするのだけれど）。

先ほどコンピュータ（当面は各コンポーネントチップレベルかな）に動作クロックを段階的に調整できるような機能が盛り込まれつつあるという話をしたが、今後、短期的な処理能力が重要視されるPCレベルでは、システム全体がこの方向に進むことは間違いなく、たとえばCPUでは連続稼動動作クロック1GHz、最大4GHz（稼働時間はチップの温度次第）といったものになると思う。エンジンが壊れないように注意しながら必要に応じてニトログリセリンを燃料に混入できるというか、ブースター付きというか、まぁチップが発熱で暴走やメルトダウンしないよう監視しながら局所的に処理能力を高められるというわけだ。今のところ発熱に応じて冷却ファンを回したり、その回転数を調整したりできるが、今後は状況に応じて動作クロックを調整することが主流になることは間違いない（でも動作クロック帯を広くするのは素人が考えるほど易しいものではないようなので、一朝一夕にはいかないらしいが）。

そういえば、ハードディスクドライブもその回転数を変動的にしてくる可能性もある。パワーマネージメントの一環でスピンドルのオンオフの中間的な位置付けとして、パワーセーブ時に回転数を抑えるのだ。現在ハードディスク

ドライブは5400rpmや7200rpmが主流だが、パーソナルな用途ではシーク性能はそれほど重要ではないので3600rpmなどより低回転へとシフトしつつある（記録密度の向上に追随しやすいという理由もあるだろう）。さらに常時アクセスすることが希なのだから、アイドル時には回転数さえ落としてしまうというわけだ。パワーセーブモードの1つ、もしくはデフォルトの動作として期待される。

なぜここでこんな話をしたかというと本題のACPIと密接に関連してくるからにほかならない。ちょっと前置きが長くなってしまったが、ACPIに話を移ろう。

ACPIとAPM

ACPIは、Intel、Microsoft、東芝の3社が中心になって進めているコンピュータ（を構成するすべてデバイス）のパワーマネージメント、そしてコンフィギュレーション機能をオペレーティングシステム主導で行うためのインターフェイス規格だ。従来、パワーマネージメント機能はAPMと呼ばれるシステムBIOSレベル（BIOSコード、APM API）、デバイスのコンフィギュレーションはPNPBIOSと、あくまでBIOSレベルであったため、オペレーティングシステムが直接コントロールできるようにはなっていなかったが、ACPIではすべてをデバイスドライバなどのオペレーティングシステムコードによってきめ細かくデバイス状況を把握し、操作できるようになる。たとえばオペレーティングシステムは、実行しているジョブタスクや各デバイスの温度に応じて利用していないデバイスをスリープ状態にしたり、電源の供給を停止したり、動作スピードを制限したりできる。また稼動中にデバイスの使用しているリソースを動的に変更（再配置）することも可能であるため、PCIデバイスなどのプラグイン／アウトといったことにも応用できる。

しかしながら、ACPI、つまりオペレーティングシステム主導のパワーマネージメントとコンフィギュレーションを首尾よく機能させるためには、チップセット、システムBIOS、各デバイス、オペレーティングシステム、デバイスドライバなどコンピュータを構成するハードウェアとソフトウェアのすべてがACPI規格に準拠した仕様になっていなければ意味がない。なお、ここでACPIの詳細を説明するわけにもいかないので、仕様に関してはACPIの公式ホームページ（http://www.teleport.com/~acpi/）を参照してほしい（ACPI 1.0bの仕様書もダウンロード可能）。

システムBIOSレベルのAPM 1.xでは、システムの電源状態を4段階程度に管理することができたが、ACPIでは、システム全体（マザーボードレベル）をG0からG3の4段階、さらにスリープ状態をS0からS5の6段階として管理でき、CPUをC0からC3、各デバイスをD0からD3と4段階にそれぞれ個別に調整できるようになる。Gはグローバル、Sはスリープ、CがCPU、Dがデバイスというのは説明しなくてもわかると思う。

少しだけ説明しておくと、それぞれ0が付いた状態は通常稼動を表している。G0が通常稼動、G1がスリープ、G2がソフトオフ、G3がメカニカルオフだが、ここでスリープを表すS0とS5はG0とG2に該当し、G1にS1からS4が含まれることになる。スリープ状態のS1からS4はそれぞれ監視するイベント、システムコンテキスト（各ハードウェアの状態）の差でレベル分けしたもので、ある程度ユーザー側で設定可能なものもある（通常、数字が大きくなるにつれて監視しておくものを少なくする）。たとえば、S1ではCPUやチップセットを含めすべてのハードウェア状態を把握した状態だが、S2ではCPUやシステムキャッシュの状態が失われることになる。S3ではメインメモリ以外の状態が失われる。S4ではメインメモリの状態も失われる。ここで状態が失われるというのは電源をカットすると考えてよい。当然、それぞれのスリープ状態に入る前に必要に応じてOSはデータ（CPUコンテキストなど）を退避するなどの適切な処理をしておく責任がある（スリープ状態からWake Upした時、CPUなどはリセットされることになる）。CPUに関しては、Sモードとは別にレーテンシを長くして消費電力を抑えるC1からC3のモードが定義されている。どの程度レーテンシを長くするかはファームウェアもしくはユーザー設定が使用されることになる。省電力モードのレベルと考えればよいだろう。また各デバイスでは、通常稼動のD0と電源オフを表すD3が定義されているだけで、D1とD2の電源状態はD0＞D1＞D2＞D3となっているだけで、それぞれの実装依存となっている。ACPIでは、上記以外に電源ボタン（ソフトオフなどに対応）やバッテリー、温度管理などの定義がなされているがここでは省略する。

LinuxとACPI

Linuxでは、ACPIはカーネル2.3系（もう2.4としてリリースされているかな？）で実装されつつある状況で、OS側で最低限実装しなくてはならないコードは、ある程度できているようだ。ただ、ACPIを取り巻く環境、シス

テムBIOS、ハードウェア、OS、デバイスドライバ、設定ユーティリティとすべてのレベルでサポートされないことには機能しないわけで、現状のLinuxではユーザーモードのACPIコントロール、サポートユーティリティを含め、デバイスドライバも開発中ということもあり、現段階ではACPIの片鱗に触れることしかできない。ということで、残念ながらここではACPIの雰囲気しか伝えられない。悪しからず。

なおカーネルは2.3.99pre2を使用した。カーネルは［General setup］の［Power Management support］を選択し、［ACPI support］をカーネルに組み込む形で選択し、［Enter S1 for sleep （EXPERIMENTAL)］も同時に選択して作成した。ACPIとAPMは同時には利用できないので、［Advanced Power Management BIOS support］は選択していない。

また、テスト環境のネットワークデバイスはintel 82557／82558 OEMチップであるが、Linuxの該当するデバイスドライバ［EtherExpressPro/100 support］では開発中ながらパワーマネージメント用のルーチンが組み込まれているので、それを有効にする［Enable Power Management （EXPERIMENTAL)］も同時に設定しておく。あと今回は、ACPIとPCI、ネットワークデバイスドライバのソースファイルでデバックメッセージを表示するためのフラグも設定した。

一応、ACPI4Linuxプロジェクトのホームページを紹介しておく。ここでは、開発中のユーザースペースACPIデーモンやテストツールなどを入手できる。ここではacpid-031700.tar.gz、pmtoos-031700.tar.gz、acpipolicy.tar.gzをダウンロードし使用している。なおソースの状態で提供されているが、コンパイル方法は難しくないので割愛する。

```
http://phobos.fs.tum.de/acpi/
http://phobos.fachschaften.tu-muenchen.de/acpi/
http://www.geocities.com/SiliconValley/Hardware/3165/
```

ACPIに少しだけ触れてみる

まずはACPIを有効にしたカーネルを起動した時のカーネルメッセージから関係するものを抜粋しておく。

```
e820: 0000fc00 @ 07ff0000 (ACPI data)
e820: 00000400 @ 07fffc00 (ACPI NVS)
ACPI: "PTLTD" found at 0x000f6a70
eth0:  OEM  i82557/i82558  10/100  Ethernet  at
0xc8800000, 00:80:45:11:22:9E, IRQ 9.
eth0: speedo_open() irq 9.
eth0: Done speedo_open(), status 00000090.
eth0: Shutting down ethercard, status was 0090.
PCI: 00:0b.0 goes from D0 to D2
eth0: speedo_open() irq 9.
PCI: 00:0b.0 goes from D2 to D0
eth0: Done speedo_open(), status 00000090.
```

ACPI BIOS（ファームウェア）の検出とACPIルーチンが有効に機能し始めたことを表すメッセージとネットワークデバイスを一度シャットダウンし、再度起動していることが分かる。ネットワークデバイスでは、シャットダウンの際にD0からD2に状態を移行し、再度ハードウェアレベルのリンクを確立する際（speedo_open()）に状態をD2からD0に設定している。この状態遷移はインターフェイスのアクティブ／シャットダウンの際に行われるので、次のようにしても確認できる。

```
# ifconfig -i eth0 down
PCI: 00:0b.0 goes from D0 to D2
# ifconfig -i eth0 up
PCI: 00:0b.0 goes from D2 to D0
# pump -k
PCI: 00:0b.0 goes from D0 to D2
# pump
PCI: 00:0b.0 goes from D2 to D0
```

現時点では、ネットワークデバイスの状態操作はこれだけで、アイドル時にモードを変更するような機能は実装されていない。

CPUのモード遷移

次にカーネルに組み込まれたACPIルーチンを見てみることにしよう。ACPIルーチンでもっともわかりやすいのがOSのidleルーチンだ。通常OSはスケジュールするタスクがない状態の時、特殊なプロセス（スレッド）であるidleルーチンを実行して過ごす。idleルーチンは日がな一日、割り込みなどを監視しながら、タスクの隙間で働いている。Linuxの場合、ACPIが有効であれば、acpi_idel()ルーチンが実行されるようになる。このルーチンには、

CPUのモード遷移のためのコードが含まれていて、C0からC3をサポートしている。通常のタスクを実行している状態がC0、acpi_idel()ルーチンに突入した段階がC1、そしてC2、C3へはレーテンシ設定により突入するようになっている。

LinuxのACPIパラメータの調整は、/proc/sys/acpi/で行うことができ、次のような項目が用意されている。

```
-rw-r--r--  1 root  root  dsdt
-rw-r--r--  1 root  root  enter_lvl2_lat
-rw-r--r--  1 root  root  enter_lvl3_lat
-r--------  1 root  root  event
-rw-r--r--  1 root  root  facp
-rw-------  1 root  root  gpe_enable
-rw-------  1 root  root  gpe_level
-rw-------  1 root  root  p_blk
-rw-r--r--  1 root  root  p_lvl2_lat
-rw-r--r--  1 root  root  p_lvl3_lat
-rw-------  1 root  root  pm1_enable
-rw-------  1 root  root  s0_slp_typ
-rw-------  1 root  root  s1_slp_typ
-rw-------  1 root  root  s5_slp_typ
-rw-------  1 root  root  sleep
```

CPUのC2、C3のレーテンシ設定はp_lvl2_lat、enter_lvl2_lat、p_lvl3_lat、enter_lvl3_latが該当する。それぞれ単位はミリ秒だ。この時間とACPIタイマーの時間を比較しながら必要に応じてモード遷移が行われるようになっている。このレーテンシの設定を含め、/proc/sys/acpi/の各項目の値の設定はechoコマンドで行えるようになっているが、0と1、つまり無効／有効を設定すればよいというものではなく、ファイルシステムのパーミッション設定のようにフラグの位置が意味を持つものなので、実際にはLinuxのACPI関連のソースコードを見たり、ACPIの仕様をある程度把握したりしておかないと設定するのは無理である。

一応ここでは、強引にC2とC3に移行させてみるが、動作を保証するものではないので注意してほしい（ユーティリティ群が整備されるのを待ったほうがいい）。まずC2の設定を確認しておく。次にモード遷移を起こさせるためにp_blkに値を設定する。

```
# head p_lvl2_lat enter_lvl2_lat
==> p_lvl2_lat <==
0x00000003
==> enter_lvl2_lat <==
0x00003210
# echo 0x1010 > p_blk
ACPI C2 works
```

そうするとacpi_idle()ルーチンのC2ブロックに突入したときに一度だけ表示されるようになっているメッセージが表示されると思う。でもC2がどの程度効果的であるかはここでは計測していない。

さらにC3に移行するように設定してみる。ここではC3のレーテンシ設定p_lvl3_latとenter_lvl3_latは0xFFFFFFFFであったので適当な値を指定する。試しに100と500を設定する。

```
# echo 100 > p_lvl3_lat
# echo 1000 > enter_lvl3_lat
ACPI C3 works
```

一応、acpi_idle()ルーチンのC3ブロックに突入したときに一度だけ表示されるメッセージが表示された。C2の場合は、これといった変化を感じなかったが、C3の場合は明らかにシステムの応答が遅くなってしまった。ん〜、指定した値がまずかったのかもしれない（短すぎたかも）。実時間で30秒以上経過しているというのにこのシステムのカーネル時間は1秒程度しか進んでいない。でもそれなりのレスポンスはある。しかし、経過時間に依存するようなコマンドは遅い時間流に飲み込まれてしまう。C3からC0に自動的に復帰してくれないと困るんだけどなぁ。やはり指定した値が小さすぎるのかな？

ちなみにhwclockコマンドでCMOSクロックを参照してみると当然ながら正確な時間を刻んでいる。現在調べている途中なので、設定が悪かったのか、C3モードというのはそもそもこういうモードなのかはわからないが、追って報告しようと思う。

今回、acpidやpmtoolsの紹介もしようと思っていたのだけれど、これらも次回にスキップ。

あっそうそう、電源スイッチはレジュームボタンとしてご機嫌に機能してくれている。Xサーバを起動中にレジュームすると復帰時、全画面の再描画がなされない点が困りものだが、まぁいっか。

Rubyで行こう

プログラミングを行うときに、そのプログラミング言語の歴史や特長を知っておくと、いろいろと都合の良いことがあります。そこで、今回はRubyが他のプログラミング言語から受けた影響や特長などを紹介します。

第5回　DNA

文：赤松智也
Text : Tomoya Akamatsu

Rubyとは何もので、どこから来て、どこへ行くのか。今回は初心に帰って、Rubyが他の言語から受けた影響や特長について改めて考えてみましょう。

Ruby誕生

以下はRuby FAQに引用されている、Rubyの誕生についてのまつもとさんの言葉です。

Rubyは1993年2月24日に生まれました。その日同僚とオブジェクト指向言語の可能性について話していました。Perl（Perl4で、Perl5ではありません.）は知っていましたが、おもちゃのにおいがして（今もありますが）好きになれませんでした。オブジェクト指向スクリプト言語は期待が持てました。

Pythonも知っていましたが、本当のオブジェクト指向言語とは思えませんでした。オブジェクト指向がとって付けたもののように感じられたのです。15年来言語マニアでオブジェクト指向のファンでしたので、真にオブジェクト指向のスクリプト言語が心底ほしかったのですが、そのようなものは探しても探してもありませんでした。

そこで自分で作ろうと決心したのです。数カ月たってインタプリタが動き始めました。イテレータ、例外処理、ガベージコレクションなどほしかったものを入れ込みました。

さらにPerlの特徴をクラスライブラリとして取り込みました。Ruby 0.95を日本国内のニューズグループに投稿したのは、1995年12月のことでした。

すぐにメーリングリストを始め、ホームページを作りました。メーリングリストでは活発な意見の交換がなされました。最初からあるruby-listは今では14789通（※2000年3月現在21000通を越えています）のメールを数えています。

Ruby 1.0は1996年12月に、1.1は1997年8月に、安定バージョンとしての1.2と開発バージョンの1.3が1998年12月にリリースされています（※執筆時現在の最新は安定バージョンが1.4.4、開発バージョンが1.5.3です）。

つまり、Rubyは何か特定の目的を満たすためでなく、自分好みの言語がほしいという欲求を満たすために作られたんですね。何だか、一人よがりの使いにくそうなものができてきそうな気がしますが、実際のRubyは全然そんなことはなくて、むしろとても使いやすい、プログラミングしていて楽しい言語に仕上ってます。これはRubyが絶妙なバランス感覚で設計されているおかげでしょう。

Rubyの先祖

世にプログラミング言語はたくさんあります。実に数千以上のプログラミング言語があるのではないかと言われて

います。それらはお互いに影響を与えあっています。Rubyは、「言語おたくが他の言語から良いものを取り込んで」作った言語だと言われています。

ここでは、筆者の独断と偏見で、Rubyに与えた他の言語の影響を勝手に挙げてみます。これは、過去のまつもとさんの発言などを参考に、筆者が推測したものです。

・Perl

良くも悪くも、Rubyに一番影響を与えた言語はPerlでしょう。名前にもそれは現れています（パール（真珠）は6月の誕生石、ルビーは7月の誕生石）。変数名やメソッド名など、Perlからそのままいただいたものがたくさん見つかります。たとえば、$_、$/などの変数名や、split、join、pack、unpackなどはその代表です。Rubyのプログラムはperlのプログラムの単語の順番を並べ替えたような印象を受けるときがあります。

```
Perl: print join(":", split(",", "a,b,c,d")), "\n"
Ruby: print "a,b,c,d".split(",").join(":"), "\n"
```

筆者はもう慣れてしまったので、Rubyのやり方のほうがすっきりと頭に入ります。というか、すでにPerlのほうはsplitの引数の順番はどうだったっけ、とか考え込んでしまうようになってしまいました。これは堕落でしょうか？

・Python

Pythonを知ってますか？ http://www.python.org/を参照してください。Pythonは日本ではそれほど知られていませんが、海外ではRubyよりもはるかに有名なオブジェクト指向スクリプト言語です。先にFAQで引用したまつもとさんの言葉にも登場してましたね。Rubyも最近は海外進出挑戦中なのですが、よく「Pythonがあるのに何でRubyが必要なの？」とか言われているようです。

Pythonはすべての値がなんらかのオブジェクトであることや、比較的単純できれいな文法から、Rubyと対象領域が重なる言語です。しかし、個人的にはインデントでブロック構造を表現する文法や、スクリプト言語にしては組み込みの機能が少ない点などに不満を感じています。PythonのほうがRubyより古いわけですが、PythonがRubyに与えた影響について、以前まつもとさんは「defという予約語はPythonからと言ってもよいかなあ」と言っていました。

・C、AWK

Rubyにはこれらの言語に似た部分がありますが、おそらくPerl経由で影響を受けているのだと思います。

・Lisp

そういえば、まつもとさんはRubyの作者であるだけでなく、Emacs上のメールリーダー「cmail」の作者でもあります。RubyはCで書かれていますが、cmailは全体がEmacsに組み込みのLisp（emacs lisp）で書かれています。ということは、まつもとさんはLispについても詳しいプログラマーということになるわけですよね。

では、RubyにはLispからの影響があるのでしょうか。言語仕様を眺めていて気がつくのは、まず「nil」でしょうか。nilというのはLispの世界で伝統的に偽とか空とかを表現する値の名前です。Rubyでも同じような目的に使われています。

それから「lambda」とか「:symbol」という表記や、Mix-inという考え方とEnumerableなどの「-able」な名前は、Common LispとそのオブジェクトCLOS指向拡張から来たのではないかと考えられます。また、文字を表す「?x」やコントロール記号を表すエスケープ表記「\C-m」などは、emacs lispの記法から来ているのでしょう。emacs lispのプロ（と思われる）であるまつもとさんらしい気もします。

Rubyではメソッド名の末尾に「?」が来たり、「!」が来たりすることがありますが、これもLispの方言のひとつ、「Scheme」から受け継いでいるのだと思います。

・Smalltalk

オブジェクト指向言語であるRubyは、当然オブジェクト指向言語の元祖と言われるSmalltalkから影響を受けていると言えるでしょう。しかし、直接の影響となるとそれほどはっきりとは見えません。あるとすれば、ブロックのパラメータのまわりを囲む「||」や、Enumerableのメソッド名collect、select、detectくらいでしょうか。

・Eiffel

リファレンスマニュアルの用語集には、Eiffelからはrescue、ensureという予約語をもらったと書いてあります。また、retryという単語もEiffelゆずりのようです。EiffelのようなAlgolの影響を受けた言語は、endでブロックが終わるので、何となくRubyに似た印象を受けます。Rubyに出会ったころは、Cみたいな「{}」のほうが良いな

あと思っていたのですが、最近はすっかりなじんでしまいました。

・Sather

Satherというのは Eiffelの影響を受けて開発されたオブジェクト指向言語です（http://www.gnu.org/software/sather/）。以前、まつもとさんはRubyの機能のうち、undefとalias はSatherからと発言していたと思うのですが、今回Satherのドキュメントを調べてみるとこれらはSatherにはない機能のようです。本当はshやcppから来たものなのでしょうか？ あるいは古いバージョンのSatherにはこれらの機能があったのでしょうか？ 調べた範囲内ではわかりませんでした。

そういえば、class 文でのスーパークラスの指定に使われる「<」は、Satherが由来のようです。昔はスーパークラスを指定するために、C++ のように「:」を使っていたと聞いたことがあります。

・CLU

CLUというのは、MITで開発された抽象データ型言語です。オブジェクト指向とまではいかない言語のようです。あまり詳しくは知らないのですが、この言語にはイテレータという概念があって、Rubyのブロックの元になったということです。イテレータとブロックについては、後でもう少し説明します。

ほかにもいろいろありそうです。ずいぶんいろいろな所から影響を受けているようですね。これだけのものを「バランス良く」取り込んで設計したというのがRubyの良さなのでしょう。

BASICの思い出

筆者がまだ若かったころ、BASICでプログラミングを始めました。最近の若者は何で入門するんでしょう。BASICでのプログラミングはそんなに悪いものではありませんでした。ま、ほかを知らなかったこともありますが、プログラムを修正してすぐにrunできる気軽さは、なかなか快適だった覚えがあります。後にPascalなどを使うようになってもコンパイルという手順が面倒に感じたものです。

しかし、不満もありました。第一は、データ構造が数値、文字列、配列くらいしかないので、ちょっと複雑なプログラムを作ろうとすると、データの表現が難しくなってしまうことです。もう1つは、BASICに組み込まれていない機能は「マシン語」や「モニタ」のような「魔法」を使わない限り追加できなかったことです。そのため、すぐに機能的な限界がきてしまったのです。

Rubyは、インタプリタ言語としてBASICと同じような気軽さを持ちながら、データ構造の制限も拡張性の制限もありません。どんなに複雑なデータ構造でも自由に定義できますし、C言語を使った拡張ライブラリで機能を追加することもできます。こんな言語で入門できたらもっと楽しかったでしょうね。

RubyがBASICの影響を受けているとはとても思えませんが、インタプリタであり、手軽なところは似ている気がします。

オブジェクト指向

こうやってRubyの個別の特徴を見ていくと、受ける印象は「どこかで見たような機能が組み合わさってる」というものです。しかし、どこかで見たような機能ばかりですが、結果としてどこにもない（ほどすばらしい）というのも確かなところです。この組み合わせの妙はどこから来るのでしょう。やはり「言語おたく（本人談）」のこだわりからでしょうか。

Rubyは純オブジェクト指向言語を標榜するだけあって、オブジェクト指向機能に関してはこの「どこにもない」度がよりアップしています。

まず、重要な点として「すべてがオブジェクトである」点が挙げられます。同じインタプリタ言語でも、たとえばBASICやPerlではそれぞれのデータ型（文字列、配列など）ごとに文法としても違った扱いをするのに比べて、Rubyでは数でも文字列でも配列でもユーザー定義のデータ型（クラス）でもまったく同じように、差別なく扱います。Perlならリファレンスを使えばすべてを同様に扱えますが、いざ実際のデータを扱おうとするときに、やはり特別扱いの文法が顔を出します。このような差別のない言語としては、RubyのほかにはLispやPythonがあります。

そして、こういう差別なし言語に慣れていない人が、よく引っかかるのが「参照問題」です。「参照問題」とは今私が勝手に命名したのですが、**リスト1**を実行すると以下のように出力されます。

```
b = [9, 2, 3, 4, 5]
a = [9, 2, 3, 4, 5]
```

ここでは変数bに代入されている配列の先頭の要素を変更しただけなのに、変数aに代入されている配列の要素まで変わってしまいます。これは、Rubyの変数がオブジェクトを格納しているというよりも、むしろオブジェクトを参照しているものだから発生する現象です。「b = a」の部分で配列のコピーは行われず、ただaという変数が指し示していた配列を、bという変数も指すようになったというだけのことです。変数aとbは同じ配列（オブジェクト）を参照しているので、一方の内容を変更すると他方の内容も同時に変更されたように見えるわけです。Rubyにおける変数は「名札」と考えてもよいでしょう。

この統一的な扱いは、変数が容器であるような言語に慣れた人にはかえって引っかかる原因となるようです。Cを知っている人ならば「すべてポインタだと思えばよい」というアドバイスが役に立つかもしれません。変数の扱いの違いは、**図1**と**図2**のようになります。

Rubyがオブジェクト指向言語として重要な点は、このすべてがオブジェクトであるということでしょう。オブジェクト指向というくらいですから、オブジェクトを中心に扱えることが最も重要になるわけです。

そして、次に重要なのは優れたクラスライブラリです。これらがあれば、オブジェクトを中心にしたスクリプトプログラミングを手軽に行うという、Rubyの目的を果たすことができるからです。

もちろん、オブジェクト指向プログラミングの重要な要素と呼ばれる継承やクラス定義も大切ではありますが、筆者はこれらのほうがずっと大切だと思います。クラスを作らなくても、オブジェクト指向プログラミングはできます。Rubyが使われる局面の多くは、既存のクラスライブラリのオブジェクトを扱うだけで対応できるはずです。それでも、オブジェクト指向の嬉しさは実感できるのです。

▼リスト1

```
a = [1,2,3,4,5]          # 配列をaに代入する
b = a                    # aの配列をbに代入する
b[0] = 9                 # bの先頭の要素を9に変更
print "b = "
p b                      # [9,2,3,4,5]
print "a = "
p a                      # aも[9,2,3,4,5]になっている！
```

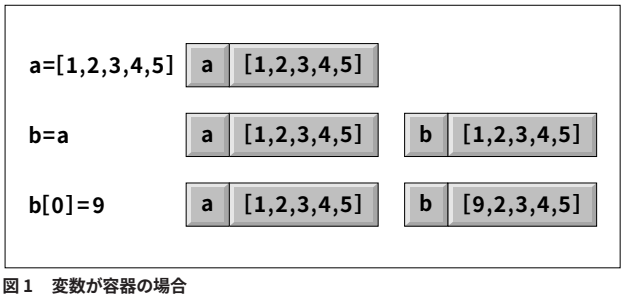


図1　変数が容器の場合

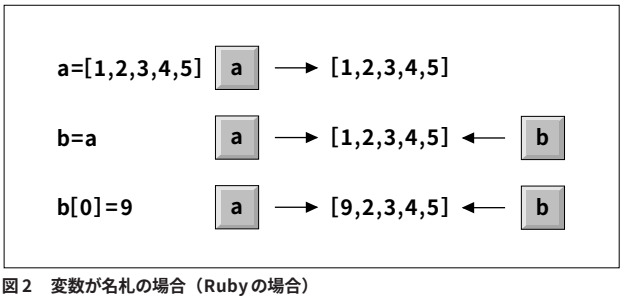


図2　変数が名札の場合（Rubyの場合）

■オブジェクト指向で読み解くcat

catは引数で指定されたファイル（または標準入力）から読み込んだデータを、標準出力に書き出す最も単純なコマンドの1つです。Rubyでcatを実装すると以下のようになります。

```
while line = ARGF.gets()
  STDOUT.print line
end
```

これをオブジェクト指向的に解説してみましょう。

ARGFは「引数から指定されたファイル（複数ある場合は結合したもの）または標準出力」を表すオブジェクトです。このオブジェクトからgetsメソッドを使って1行読み込んできて、変数lineに代入します。読み込まれた行は、文字列オブジェクトです。getsメソッドはファイルの終端でnilつまり偽を返しますので、それまでwhileの本体（endまで）を繰り返します。

STDOUTは標準出力を表すオブジェクトです。このオブジェクトのprintメソッドによって、変数lineに格納された文字列オブジェクトの内容を出力します。

endは単なるwhileの終わりです。ここまでの内容をwhileの条件が成立するまで繰り返します。

■Rubyのオブジェクト指向機能

Rubyでは、オブジェクトを扱えることが最も重要と言いましたが、もちろんクラスを定義するようないわゆるオブジェクト指向言語としての機能も充実しています。

Rubyのオブジェクト指向機能を見るときにすこしびっくりするのは、「多重継承」がないことです。Pythonや

Perlのオブジェクト指向機能でさえ多重継承をサポートするのに、オブジェクト指向言語であることを前面に出しているはずのRubyに、多重継承がないのは少々意外でした。しかし、Ruby本によると、多重継承の問題点を軽減するために、わざとこうなっているということです。多重継承の代わりはモジュールによるMix-inで実現されています。これは他のオブジェクト指向言語にはない、Rubyならではの特徴だと思います。

Mix-inとはモジュールというメソッドなどの属性をまとめたものをクラスに「混ぜ込む」機能です。この機能を使えば、任意のクラスに機能を追加できます。あまり実用的な例ではありませんが、よく多重継承に使われる例と、それをMix-inで表したものを見てみましょう。

この例では「鳥」クラスと「飛行機」クラスはそれぞれ複数のクラスから継承しています。これが多重継承です（**図3**）。

RubyにあるようなMix-inを使うと、これは**図4**のように表現されます。

Mix-inを使った例では「鳥」クラスも「飛行機」クラスもひとつのクラスからだけ継承しています。これを単純継承と言います。そして多重継承の代わりに、共有される性質を持つ「飛べる」モジュールを「混ぜ込んで」います。これをRubyで記述すると、**リスト2**のようになります。

Mix-inを使えば、ほぼ多重継承と同じことができます。つまり、多重継承を使って構成されるクラス構成は、Mix-inを使ってほぼ同等にできるのです。では、多重継承とMix-inの違いはなにかというと、要はMix-inを使ったほうが理解しやすい階層構造になりやすいということのようです。

もうひとつ、Rubyに特徴的な機能としては、「特異メソッド」があります。特異メソッドとは、ある特定のオブジェクトだけが持つメソッドのことで、個別メソッドと呼ばれることもあるようです（**リスト3**）。

オブジェクト指向言語の中にはクラスを持たないで、元になるオブジェクトにメソッドを追加するタイプのものもいくつかあります。あまり有名なものはありませんが、SelfやCecilのような言語がそれで、プロトタイプベース言語とも呼ばれます。一方、通常のクラスによるオブジェクト指向言語は、クラスベース言語と呼ばれます。Rubyはクラスベース言語でありながら、プロトタイプベース言語に必要な機能を揃えているという点でちょっと変わっています。

リスト4のように元になるオブジェクトをcloneし、必要に応じてメソッドを追加することでクラスを一切使わないでプロトタイプベースのオブジェクト指向プログラミングが可能です。とはいえ、正直に言うと、できるのはよいが実際どう活用すればよいのかよくわかりません（笑）。

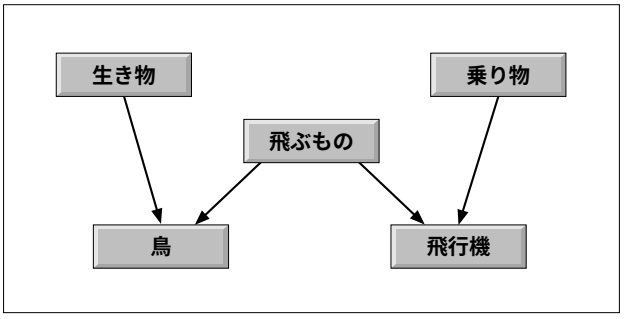


図3　多重継承

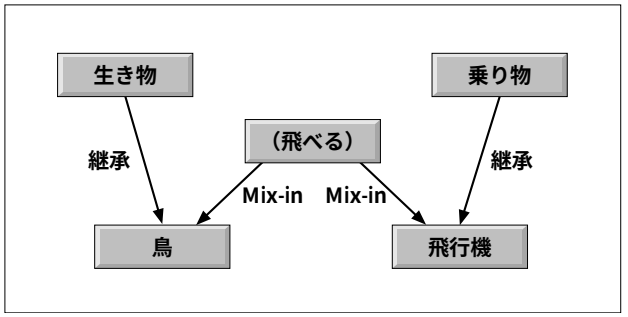


図4　継承とMix-in

リスト2　継承とMix-in

```
class 生き物
  # ここで生き物の性質を定義
end

class 乗物
  # ここで乗物の性質を定義
end

module 飛べる
  # 「飛べる」という属性の性質
  # たとえばメソッド「飛ぶ」
  def 飛ぶ
    ....
  end
end

class 鳥<生き物
  include 飛べる   # モジュールの機能を「混ぜ込む」
  # ここで鳥の性質を定義
end

class 飛行機<乗物
  include 飛べる   # モジュールの機能を「混ぜ込む」
  # ここで飛行機の性質を定義
end
```

■イテレータ

イテレータ（iterator）のiterateとは繰り返すという意味です。イテレータは、もともとCLUという言語で採用された機能です。CLUでのイテレータは以下のように使います。

```
for i:int in int$from_to(1, string$size(s)) do
  c:char := s[i]
  ...
end
```

ここでは、int$from_to(1, string$size(s))の部分がイテレータです。「int$」は「整数の機能を呼び出す」という意味です。イテレータint$from_to(1, string$size(s))は、1からstring$size(s)までの範囲で、doからendまでを繰り返します。Rubyでイテレータを使うならこうなるでしょうか（Rubyのインデックスは0から始まります）。

```
0.upto(s.size-1) do |i|
  c = s[i]
  ...
end
```

ずいぶん表現がシンプルになります。Rubyはオブジェクト指向言語なので、いちいち型の名前を指定する必要がない点と、Rubyのuptoは終端の値を含むので、あらかじめ−1しておく必要がある点が違いますね。あ、そうそう。ブロックは「{ }」を使っても書けます。筆者自身は、ブロックも含めて制御構造のように使うときには「do～end」を、式として使うときには「{ }」を使っています。

RubyはこのCLUのイテレータを一歩進めて、繰り返し以外に使えるようになっています。たとえば、以下のような使い方です。

リスト3　特異メソッドの例

```
a = "peter"
def a.age          # aだけのメソッドを定義
  42
end
b = "james"
def b.age          # bだけのメソッドを定義
  40
end
a.age              # => 42
b.age              # => 40
```

```
[1,2,3,4,5].delete_if {|x| x % 2 == 0}
```

delete_ifは、doからendの間（ブロックと呼びます）を実行した結果が真である要素を取り除きます。ブロックの実行結果とは、ブロック内で最後に実行した式の値です。結果として、これは「偶数の要素を取り除く」という働きをします。このようなブロックの値を使うというのは、CLUのイテレータにはなかった働きです。Rubyではさらに、繰り返し以外にもイテレータを使うようになりました。

```
btn = TkButton::new
btn.command{print "button pushed\n"}
```

TkButtonのcommandメソッドは、ボタンがクリックされたときにブロックを実行するように登録します。ここまで来てしまうと、もうまったく「繰り返し」ではありませんね。ま、0回とか1回でも繰り返しであるという詭弁を使うことができるのかもしれませんが。そういうこともあって、最近はイテレータとは呼ばないで、「ブロック付きメソッド」というように呼ぶことが増えているようです。RubyのブロックとCLUのイテレータのもう1つの違いは、Rubyのブロックはオプショナル、つまり同じメソッドにブロックを与えたり、与えなかったりできるということです。配列のソートメソッドsortを例に考えましょう。

```
[5,2,3,1,4].sort                  # => [1,2,3,4,5]
[5,2,3,1,4].sort{|a,b|a<=>b}      # => [1,2,3,4,5]
(省略時と同じ働き)
[5,2,3,1,4].sort{|a,b|b<=>a}      # => [5,4,3,2,1]
```

リスト4　プロトタイプベースのオブジェクト指向プログラミング例

```
Point = Object.new
def Point.move(x,y)
  @x = x
  @y = y
end
def Point.x
  @x
end
def Point.y
  @y
end
Point.move(0,0)   # 原点に設定
p1 = Point.clone  # 新しい点を作る
p1.move(100,100)  # 新しい点を(100,100)に
```

sortはブロックを与えられた場合にはブロックの値（a、bの大小関係に応じて正、ゼロ、負の値）を基準にしてソートします。同じメソッドがブロックを与えられるかどうかで挙動が変わるわけです。

というわけで、今月は他言語との比較という観点からRubyを眺めてみました。似ているようでひと味違うRubyの特徴が伝わりましたでしょうか？ 今月は具体的な例が少なかったので、来月はもうちょっと実用的なプログラムを紹介したいと思います。

Column

今月のRuby 1.5

開発バージョンの1.5はもうすぐ機能拡張を凍結して、次の安定バージョン1.6として夏ごろにリリースするとアナウンスされました（ruby-list:21332）。

そこで、今月からしばらくの間、この開発版のウォッチを行おうと思います。今月は一番気になる非互換な変更点についてまとめます。

Ruby 1.5は1.4以前に対して非互換な変更をいくつか含んでいます。これらの変更によって、少数ですが書き換えが必要となるプログラムがあるので、注意する必要があります。2000年3月時点での、Ruby 1.5における非互換な変更は以下のとおりです。

・hashのvalueとしてのnil

1.4まではnilはハッシュの値として認められておらず、nilを設定することはその項目を削除することと同等でした。1.5ではnilは他のすべての値と同様、自由にハッシュの値とすることができます。ハッシュの項目の削除をnilを設定することで行っていたプログラムは、deleteメソッドで明示的に削除する必要があります。

```
hash[key] = nil# 1.4
hash.delete(key)# 1.5（1.4でもOK）
```

・シンボルのオブジェクト化

識別子を表現する「:symbol」という書式は1.4以前では数値でしたが、1.5ではSymbolクラスのオブジェクトになりました。Symbolクラスは、Fixnum同様、即値オブジェクトです。シンボルが数値であることを期待している以下のようなプログラムは、書き換えが必要になります。

```
if id.kind_of?(Integer)     # 1.4
  a = id.id2name
end

unless id.kind_of?(String)  # 1.5
                        （1.4でも可）
  a = id.id2name
end
```

・ScriptError例外

SyntaxError、NameError、LoadError、NotImplementErrorの各例外が、StandardErrorのサブクラスからScriptErrorのサブクラスに変更になりました。ScriptErrorは、StandardErrorのサブクラスではなくExceptionの直接のサブクラスですから、クラス名を指定しないrescue節で捕捉されません。明示的にクラス名を指定する必要があります。

・Thread.joinがなくなった

Threadクラスのクラスメソッド Thread.joinがなくなりました。代わりに、Threadクラスのインスタンスメソッドを使ってください。1.4でもこのメソッドを使うと警告が出力されていましたから、ほとんどの人はすでに修正していると思います。

```
Thread.join(th)# 1.4
th.join        # 1.5（1.4でもOK）
```

これらの変更はChangeLogとToDoファイルを調べてリストしましたが、まだ見落としがあるかもしれません。気が付いた人がいれば、編集部経由で教えてください。メールアドレスは<linuxmag@ml.ascii.co.jp>です。

■Ruby 1.5の入手法

安定版と違って、Ruby 1.5はアーカイブファイルで入手できません。ネットワークからアノニマスCVSで入手する必要があります。アノニマスCVSによって、Ruby 1.5のソースを入手する手順は**リスト**のとおりです。cvsコマンドはインストールされているものとします。

この手順により、カレントディレクトリにソースコードを格納したrubyというディレクトリができます。あとは通常の手順に従い、configure、make、make installでコンパイルとインストールしてください。

なお、近いうちにCVSサーバがcvs.ruby-lang.orgに移動する予定があるそうです。もしかしたら、今月号が発売される前に移動してしまっているかもしれません。上の入手手順がうまくいかなかった場合には、

```
http://www.ruby-lang.org/ja/download.html
```

を参照してみてください。

▼リスト　Ruby 1.5のソース入手手順

```
・ログイン
% cvs -d :pserver:anonymous@cvs.netlab.co.jp:/home/cvs login
(Logging in to anonymous@cvs.netlab.co.jp)
CVS password: guest

・チェックアウト
% cvs -d :pserver:anonymous@cvs.netlab.co.jp:/home/cvs checkout ruby
cvs server: updating ruby
U ruby/.cvsignore
...
```

Linux ガベコレ

Linux Garbage Collection

文：しのはらひろあき
Text：Hiroaki Shinohara

第1回
【login】（ろぐ-いん）
【ls】（える-えす）
【distribution】（でぃすとりびゅーしょん）
【kernel】（かーねる）

login
【ろぐ-いん】


［唱和する祈祷師］

（1）“ ものごとの始まり ” の意。いにしえのオペレーティングシステム「Linux」において儀式を始める際、慣例的に唱えられていた呪文が転じた。現在でも残る古文書によれば、当時の祈祷師はこの言葉を唱和したあと、RPMパッケージをインストールする、動かないのでソースからコンパイルする、コンパイルに失敗したのでふて寝するなど、数々のいかがわしい行為にふけったとされる。したがって、loginで始めるこの用語集も、いかがわしいものであることが予想できる。

（2）Linuxで、もっとも使われる頻度が低い基本コマンドのひとつ。多くの人類が、このコマンドをタイプすることなく歳をとり、命を落とし、やがて土に還っていった。

シェルのコマンドラインから、

```
$ login＜Enter＞
```

とすると、

```
login:
```

と出てログイン画面になる。これにより、現在使用しているマシンに何度でも再ログインすることが可能。作業中、イヤなことがあった、生きることに虚しさを感じたといったとき、今までの環境を捨てて心機一転やりなおすために使用する。

編者も、一生にただ一人と思った女性と別れる、会社を辞めるなど、たびたび人生のloginコマンドを使いこなしてきた。その経験から、若い人たちにはっきり言っておきたいのは、一時の感情に身をまかせていたずらに再ログインを繰り返してはならないということである。一度捨てた生活は、あとで “ あのころはよかった ” と思っても、二度と取り戻すことができないのだ。ただしこれはcsh系シェル環境の場合で、Linuxで一般的に使われているbashではlogoutすればもとの環境に戻る。自分の人生のログインシェルがbashでなかったことを悔やむ、今日このごろである。

（3）コンピュータユーザーの使用OSを判別する “ 踏み絵ワード ” のひとつ。**「コンピュータを使うには、まず××する」**の虫喰い部分を埋めさせることで、その人の使っているOSを知ることができる。

▶「ログイン」と答えた人

Linuxユーザー。または他のUNIX互換OS利用者。しかし、チンパンジーにタイプライタを与えたとき偶然シェイクスピア並の名作を書き上げる可能性がゼロではないことを考えると、相手がたまたまこのことばを発したということもありうる。より確実を期すため、回答者が猿ではないか、タイプライタを隠していないか、前もって確認しておくこと

▶「ログオン」と答えた人

Windowsユーザー、またはOS/2ユーザー。ただし、現在ではOS/2ユーザーは稀。さらに、NEC ACOSの保守管理をしている人という可能性も捨てきれない

▶「部下を呼ぶ」と答えた人

IT革命の進展に取り残されぎみな中間管理職。会社では厳しい上司を演じているが、家に帰れば妻と中学生の娘

に頭が上がらない。できれば若い部下とは心を通じていたいと願っている。そういう人はまずは自分から、すすんでコミュニケーションを取ってみてはどうでしょう。今年の夏ごろ、木星があなたの運勢に強い影響を与え始めます。ラッキーカラーは青（結城モスラ先生・談）

▶「わからない」と答えた人

正常な社会生活を営んでいる人

ls

【える-えす】

(1) ディレクトリの内容を一覧するためのコマンド。

(2) Linuxで、もっとも“無難なコマンド”。いつ実行しても、システムを壊す、ハードディスクをフォーマットする、コンピュータが突然“デイジー”を歌い出すといったことがないため、安全に使用できる。安全なので、手持ちぶさたなときやマシンが正常に動いているか確認したいとき、本当はろくに知らないのにLinuxを使えることを誇示したいときなどに、なんとなく実行される。だが、いつも優しく“安全”な男性が、女性にとっては逆に物足りず魅力に欠けるのと同様、最近では優しすぎるlsにも変化が求められつつある。

編者も若かりしころは、バイクでチキンレースをしたり、霊験あらたかな壺を買ってみては和牛商法に手を出したりといろいろあった。女性にとってはこのように危ないところのある男性のほうが、母性本能をくすぐられるものなのである。

したがって、幅広い層に受け入れられることを目指す新時代のLinuxディストリビューションでは、ちょっと目を離しているとファイルを消してしまったりする機能をlsに付けることが時代の要請といえよう。

distribution

【でぃすとりびゅーしょん】

本来Linuxとは、OSの基本部分である“カーネル”を指すが、これだけでは機能を果たすことができないため、コマンド・周辺ソフトウェアまでパッケージにしてインストールしやすく実用に供するものとして配布される。このパッケージのことをディストリビューションと呼ぶ。

現在では多くのディストリビューションが存在するが、なかでも有名なものをいくつか挙げると、酒粕だけ除いた粗悪酒のように質の悪い「カストリビューション」、使えば使うほど太っていく「関取ビューション」、エディタで“HP”と打ち込むと“ヒューレット・パッカードのことですか”などといちいちイヤミを言う「揚げ足取りビューション」、北アメリカから侵入してきて、爆発的に全国で繁殖、スズカケノキなど街路樹の葉を食べる「アメリカシロヒトリビューション」、母さん、新しい車買うんだから金出してくれよ、遊んでばかりいないで旅館のほうもきちんと見てくれないと、ケッ、急に親父が亡くなったからイヤイヤ引き継いだけど、俺は旅館の経営なんて興味ないね、勝手にやってくれよ、あっ、待ちなさい、やっぱり若旦那は頼りにならない、こんなとき先代がいてくれたら……の「跡取りビューション」、長い修行の末、コンピュータを使うことの虚しさを知った「悟りビューション」、たくさん並べて飽きてきたあたりで真打ちとして登場する「オオトリビューション」などがある。

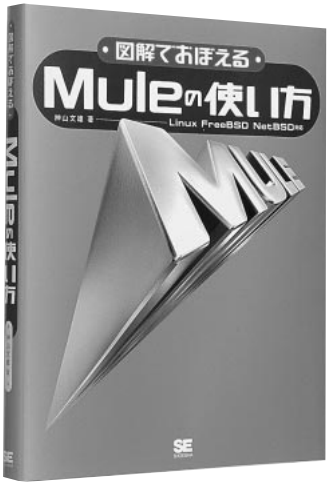
kernel

【かーねる】

今からおよそ110年前の1890年、アメリカ・インディアナ州南部のヘンリービルに生まれる。学校をやめて農場の手伝いをするなど、恵まれているとはいえない幼少期を過ごしたのち、市電の車掌・修理工・ボイラー係や保険外交員などを転々。波乱に満ちた経歴をもつ。その後、1930年・40歳のとき、ケンタッキー州コービンでガソリンスタンドをオープン。ここでも愛息を亡くすという不幸に見舞われるが、併設の食堂で出していたチキンのオリジナルレシピとサービスのよさがやがて評判に。その名声から、1935年に当時のケンタッキー州知事から“カーネル”（大佐）の名誉称号を受ける。カーネル・ハーランド・サンダース、すなわち「カーネル・サンダース」の呼び名はここから来ている。その後、カーネルは65歳でチキンレストラン「ケンタッキーフライドチキン」のチェーン展開を始め、90歳で亡くなるまでに48カ国6000店に育て上げるなど、生涯現役で活躍を続けた。

なお、英語のつづりは“Colonel Sanders”なので、OSの根幹部分のことを指す“kernel”とは日本語の読みが同じでも、まったく関連性はない。

しのはらひろあき
略歴：
1643年　田畑永代売買禁止令発布
1882年　結核菌発見
1919年　コミンテルン設立
1932年　満州国建国
1965年　筆者、千葉県に生まれる
計算機専門家の証である“珠算検定3級”の合格者。都内在住34歳。



図解でおぼえる Muleの使い方

神山 文雄 著
翔泳社
B5変形判／288ページ　　本体価格 2600円

UNIXの世界に足を踏み入れたばかりの人間にとって、「vi使い」とか「Emacs使い」といった言葉は一種の畏れにも似た特別な響きを持っている。「Mule使い」もそうだ。Muleは、Emacs系のテキストエディタのひとつだが、それだけでは語れない不思議な魅力を秘めたツールである。現に「Mule使い」は、確実に他の「〜使い」よりも対象に対する愛着が深い（ように感じる）。

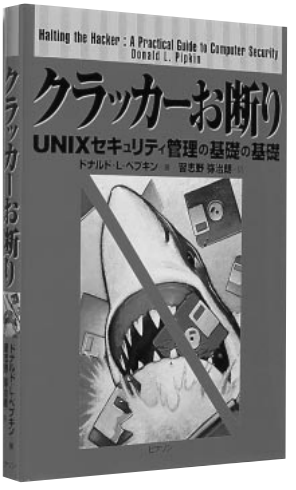
本書は「Mule使い」への第一歩を記すためのガイドとして好適な入門書だ。インストールと設定の方法、テキスト入力の基本、カット＆ペーストなどの編集コマンドの用例、おぼえておくと便利な機能など、Muleを使うために必要な基本操作を順を追って解説している。実際の操作画面も豊富に掲載されていて（オレンジ色が目に痛いのはやや難ありだが）、わかりやすい構成になっているのもマル。

クラッカーお断り　UNIXセキュリティ管理の基礎の基礎

ドナルド・L・ヘプキン 著　習志野 弥治郎 訳
ピアソン・エデュケーション
A5判／275ページ　　本体価格 2000円

本来、「ハッカー」とはコンピュータに対する造詣が深く、システム開発の場面などで特別な力を発揮する優れた人を指す。純粋なるハッカー達は、コンピュータシステムへの悪質な侵入者に対して（自らと区別するために）「クラッカー」なる言葉を用いる。本書は、その「クラッカー」達が用いる手法、行動パターンや心理を解析することで、セキュリティを適用するうえで考慮すべきポイントを明らかにしていく。著者のペプキンは、システム管理者もクラッカーから学ぶべきだと説く。孫子も言っているように「敵を知る」ということは、戦いを有利に進めるために不可欠な要素なのだ。

硬くなりがちな内容を平易な文体でわかりやすく解説しており、セキュリティについて知りたい、既得の知識を整理しておきたい、といった読者にお勧めだ。コラムも面白く、セキュリティ関連のトピックにまつわる薀蓄が語られていて結構ためになる。



Javaによる分散アプリケーション開発

国際大学ーアリス・エフ ソフトウェアR&Dセンター 著
トッパン
B5変形判／408ページ／CD-ROM付き　　本体価格 2800円

先月号のこのコーナーでJavaの入門書を紹介したばかりだが、今月は一足飛びに分散アプリケーション開発の解説書である。インターネットショッピングモールシステムの開発をモデルケースとして、Javaによるシステム開発の手法をトータルに解説する内容だ。もちろん対象読者はシステム開発に携わる上級者である。

この手の解説書は、内容が中途半端で実際の事例に適合しなかったり、逆に細かなプログラミングリファレンス的な方向に偏っていたりして、開発の現場では意外に役に立たないことが多い。本書がとっている、1冊を通してひとつの事例をベースに解説していくという手法はよく見られるものだが、開発環境をIBMのVisualAge for Javaに限定し、モデルケースの仕様をきちんと定義することで、開発に必要な具体的ノウハウを過不足なく紹介している。読み手を選ぶ内容ではあるが、マッチする読者にとってはありがたい1冊だ。

復活! TK-80

榊 正憲 著
アスキー
B5変形判／64ページ／CD-ROM付き　本体価格 2480円

本書は、1976年に開発されベストセラーとなった国産ワンボードマイコン「TK-80」を、CD-ROMに収録のシミュレータによってWindows上によみがえらせるものである（95／98／NT4.0で動作）。TK-80はNEC製のIntel8080互換CPUを使ったマイコンキットで、プログラムを手でアセンブルして16進キーボードから直接メモリに入力し、実行のようすは8桁の7セグメントLEDに表示する。この簡素なマシンが、PC-8001から、PC-8801、PC-9801を経て現在のPCへと発展するのである。

そのほかCD-ROMには、TK-80のマニュアル（PDF）、シミュレータのソースコード、岸田孝一氏らが作成した「もぐらたたき」などのゲームプログラム6本が収録されている。TK-80で育った世代にはたいへんなつかしい内容だ。資料としての価値もあり。

データベースLinux ― Java ServletによるWebサーバー構築 編

インプライズ株式会社
監修　加藤 大受／田原 孝 著

アスキー

B5判／388ページ／CD-ROM付き

本体価格 3800円

TURBOLINUXはじめの一歩

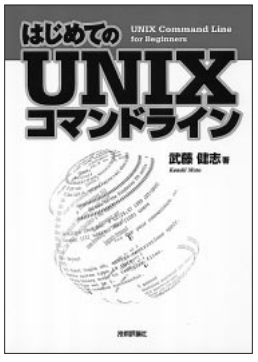
伊藤 雅俊 著

ソフトバンク
パブリッシング

B5変形判／264ページ

本体価格 1400円

はじめてのUNIXコマンドライン



武藤 健志 著

技術評論社

A5判／296ページ

本体価格 1980円

図解標準 最新UNIXハンドブック

伊藤 和人 著

秀和システム

B5判／488ページ

本体価格 2981円

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

自分専用の1Uラックマウントサーバの構築と費用の節約方法

Text : Tom Adelstein
http://linuxtoday.com/stories/15158.html（原文）
http://www.ascii.co.jp/linux/linuxtoday/article/article404384-000.html（邦訳全文）

この記事では1Uラックマウントコンピュータの謎を取りあげ、どのようにすれば読者が自分専用のコンピュータを作れるか検討してみよう。

まず、小さいシャーシの調達先を見つけることが、その計画のもっとも大変な仕事であることに気づくだろう。そこで、これをずっと簡単にする方法をお教えしよう。General Technics（www.gtweb.net）は3ベイ付きIPCケース1U ATX 150W（ベージュ）のシャーシを268ドルで販売している。これは従来型ケースの79ドルと比べてちょっと高価に思えるかもしれない。需要と供給、それに他の販売業者の価格を考えると、General Technics（GT）はライザカードが1つ付いていて、これまでに見つかった中でもっとも低価格で高品質のケースを提供している。GTのケースはCS440という名前だ。

Intelのマザーボードが1Uラック用でもっとも一般的だ。標準のIntel純正CA810 ボードは十分なリソースを提供してくれる。Fry'sはCeleronのみのバージョンを119ドルで販売している。普通のCA810にはネットワークインターフェイスカードが付属していない。IntelはCA810LAのボード上にオプションの10/100BASE-T Ethernetカードを提供している。しかし、その選択肢をとらずにIntelのPRO/100+ Dual Port Server Adapter（PILA8472 DPA）を選びたいと考える人もいるだろう。

この1Uケースには標準のPCIスロットに適合するライザカードを1枚取り付けるための場所がある。IntelのPILA8472 DPAを使えば、1つのスペースに2つのアダプタを持つこともできる。このDPAの価格は、購入方法と購入先によって異なるが、200ドルから300ドル程度だ。

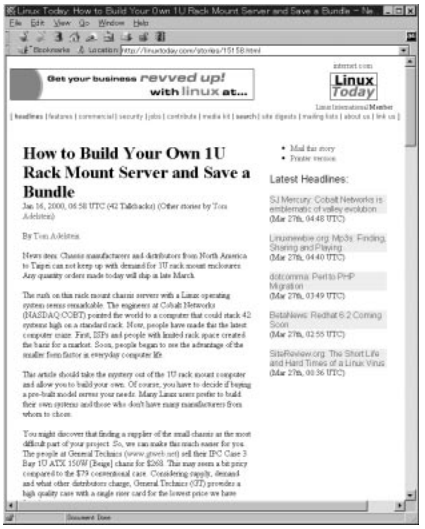
Ethernetカードが決まったら、プロセッサとメモリ、ハードディスクが必要だ。一般的な選択は、たとえばCeleron-500MHz 128Kバイト L2-cache PGA-370で、これには3年間の保証がついている。

メモリの価格は毎日のように変動している。Intel CA810ボードには2つの168ピンのDIMMスロットがあり、最大512Mバイトの RAMを使うことができる。とはいえ、薄型のメモリを買う必要がある。

前述の1Uラックマウントサーバには、標準のATXフォームファクタを使ってハードウェアや電源装置を取り付ける。CS440ラックマウントシャーシは1台の標準ATAPI CD-ROMおよびフロッピーディスクドライブを組み込むことができる。

さて、この1Uラックサーバを、1200～1500ドルの範囲で構成されたいくつかの有名ブランドのシステムと比べてみよう。私たちはCeleronの466MHzプロセッサを使ってシステムを互換性のあるものにする。コストをまとめると以下の表のようになる。

CS440 1Uラックマウントシャーシ：$268
Intel CA810マザーボード：$119
Intel Celeron 466MHzプロセッサ：$85
1ポートアダプタ：$20
64Mバイト DIMM：$80
13Mバイトハードディスク：$125
CD-ROMとフロッピー：$75（オプション）
合計：$697



プロフィール
Tom Adelstein は公認会計士で、Bynari, Inc.のCIO／CFOである。彼は商業や技術に関するいくつかの書籍および記事の著者であり、情報工学の分野での経営、コンサルティング、および実務の経験を持つ。

Linux Todayの対FUD活動で明らかになった進行中の「世界制覇」

Text : John Matthews　（Linux Todayシリコンバレー特派員）

http://linuxtoday.com/stories/13109.html（原文）
http://www.ascii.co.jp/linux/linuxtoday/article/article411936-000.html（邦訳全文）

■Linuxについての対FUD（※注）サイト開設

何カ月間かの作業が済んで、Linux Todayの対FUDサイト（Linux Counter-FUD Site）の「第1部」がついに開設されつつある。この記事の中のわくわくするような話題は、FUDとはあまり関係がなく、このサイトの準備作業で私たちが発見したことに関するものだ。

ニュース記事を精選して、FUDへの反撃にもっともふさわしい素材を分類し、要約するにつれ、私たちは自分たちの作ったものの成果による「衝撃」に対して何度も驚かされた。私たちの見ていたものはLinuxに関するFUDへの強固な反撃であるだけでなく、Linuxの「世界制覇」への進撃が実際に行われつつあることの堅固な証拠でもあったのだ。

■LinuxがWindowsに取って代わるにはどれほどの規模の変革が必要か

Linuxコミュニティの人々はここしばらく「世界制覇」を口にしてきた。LinuxがWindowsに取って代わるのにどれほどの規模の変革が必要になるかは、「驚異的な」、「度肝を抜くような」、「信じがたい」などというのが妥当なところだろう。Microsoftによる市場支配について重要な点をあげてみることで、それがどの程度のものなのかを実際に示してみよう。

・「地球上のコンピュータの90%」の支配
・開発ツール
・アプリケーション
・包囲されたUNIX：いまだに持ちこたえている唯一の分野は、きわめて高性能なサーバとUNIXを搭載したエンジニアリング用ワークステーションだ。そしてMicrosoftはNT5（現在Windows 2000に改名されている）のリリースを約束しているが、それはこの市場のかなりの部分を奪い去ることになるだろう

というわけで、何らかの代替となるテクノロジーが現れ、それがMicrosoftのものよりあらゆる面で明らかに優れていたとしても、この市場支配に対して本気で挑戦してMicrosoft製品に取って代わろうとするテクノロジーが「大変革」を起こすというのはとても信じがたいことなのだ。商用ソフトウェアからオープンソースのソフトウェアへの切り替えは単なる「ベンダーの変更」よりもずっと大きな変更なのだから、Linuxの場合はなおさらである。オープンソースとは、商用ベンダーにとっては値下げをして営業方法を大きく変革することを余儀なくするものであり、また企業のIT部門にとってはそれを採用することによってソフトウェアの配備およびサポート方法の大変革が必要となるようなものだ。

しかしどうだろう。このLinuxによる大変革は実際に起こりつつある。Linux対応を発表する商用ベンダーの数が爆発的に増えているのだ。

■見落としがちな潮流

すべてのニュースに毎日目を通していても、この大変革を見落とすことはあり得る。オンラインの情報源で「Linux」や「オープンソース」についていくらでも検索できるが、それでもこれを捕らえることはできない。

私たちが対FUDの記事―たとえばLinuxで使えるGUIや実務でのLinuxの利用、米Computer Associatesのようなベンダーが Linuxを売り込むために何をしているか―などといった話題を集めてまとめたように、Linuxに特に焦点を絞った話題を注意深く順番にたどり、1999年初頭からの記事を眺めていけば、Linuxに関して起きていることが少なくともその分野においては大事件であり、またその分野で実に大きな「木」が育ちつつあるということがわかるだろう。Linuxの記事に対し、考え得るすべての話題についてこうした処理を繰り返し、そこで見つけたものすべてを総合して考えれば、そこではじめて、起こりつつあることの重大性が認識できるようになり、「森」の姿がはっきり見えるようになる（実際のところ、これは想像よりもずっと大変な作業だった。たとえば「実務でのLinuxの利用」のような一括検索しただけでは見つけられない話題が数多くあるため、毎日流れてくるニュースをすべて調べて各記事に目を通し、見つけたリンクや引用を「採取」しなければならないのだ）。

■FUDへの反撃からLinuxの「世界制覇」の進展の観察への転換

これが読者やLinuxおよびオープンソースのコミュニティやメインストリームの報道機関のために私たちが行っていることであり、また、私たちが見つけたものを読み、それをもっと深く知りたいときには元記事を調べることに時間を使おうとするすべての人のためのものである。私たちは単に、業界紙にいままでに現われ、そして現在も存在している「Linuxに関するFUD」に反撃しようとしているだけだ。

というわけで、皆様にこの森をご覧いただきたい。Linux Todayの Linux Counter-FUD Site を巡る旅をはじめよう。

※【FUD】不安（fear）、不確実性（uncertainty）、疑い（doubt）の頭文字で、競争相手の製品ではなく自社製品を使うように顧客を口説くために、競争相手の機器やソフトウェアの未来には暗雲が立ち込めていると断言することによって暗黙の強制をすること。（「ハッカーズ大辞典」福崎俊博訳・アスキーより）

読者の声

俺にもいわせろ！

みなさんこんにちは。担当は先月の英語キーボードに続き、ケース、マザーボードとAthlonを買ってしまいました。しかし、いじる時間がなく部屋の暖房と化しています。もう十分に暖かいのに。しくしく。

4月号特集1へのおハガキ

「往年の名機復活大作戦」非常に楽しく読ませていただきました。努力すれば昔のマシンでも十分役に立つことはすばらしいことです。私も研究室でCD-ROM、モニタが故障して余っていたマシン（Pentium 100MHz）をプリントサーバとして復活させました。

（埼玉県　島村健一さん）

企画として面白かったが、私の机の上で現役のFMV DESKPOWER 5100DSを「旧型PC」といわれたことにはかなりのショックを受けた。5年くらい前の機種だもんな。

（兵庫県　徳田献一さん）

今回の特集は最高（笑）でした。次回はさらに進んで、「CPU 20MHz、メモリ 4Mバイト以下、ハードディスク300Mバイト」のようなPC用のLinuxを特集してほしいです（笑）。私の家にたくさん転がっております……。

（東京都　田邉純一さん）

最近のCPUは、物理的／電気的なスペックもコロコロと変わってしまうので、アップグレードが面倒ですね。自作機ならマザーボードごと交換ということになるのでしょうが、メーカー製マシンだとそうもいかず…。

実は、編集部にもPentium 90MHz～166MHzクラスのマシンがたくさんあるんです。そのほとんどが実験機かサーバとして稼働しています。用途さえ選べば高スペックは必要ないんですが、やっぱりパワーアップもしたいですよね。

4月号の特集2へのおハガキ

「マルチメディアLinuxで遊ぼう」という記事、大変参考になりました。テレビまで見られるとは知りませんでした。最近のLinuxは次々と新しいバージョンが出て、ユーザーとしてはうれしい限りです。が、「Linuxでは何ができる」というような基本的なことがまだまだ勉強不足でわかっていません。「Linuxでは、あんなことからこんなことまでできる！」みたいな特集をやってもらえると助かります。それでは、これからも期待しています。

（千葉県　阿部滋治さん）

4月号の特集はヒットでした。私もMIDI系とインターネット中継を突っ込んでやってみようと思います。

あと、Kondara-Zooは付録に付きませんか？

（広島県　岡田まさみさん）

テレビが見られない。Windows 2000にもLinuxにも対応しないアヤしいカードなので、ダメだという気はしてたけど……。

（埼玉県　佐藤信博さん）

Linuxはマルチメディアと縁がないと思っている方も多いようですね。周辺機器メーカーの対応など、Windowsにかなわない部分もまだまだありますが、それでも数年前からは考えられないほど環境がよくなりました。これからも、Linuxをコキ使おう！という企画を展開していきたいと思います。もちろん、みなさまよりのご意見もお待ちしております。

岡田さん、Kondara-Zoo収録の件、検討させていただきます。

肩の力を抜いて楽しくいきましょう

LinuxPPCに取り組んでみた。数カ月。いったい何ができるようになったのか、自問してみてもこれといった結果が思いつかない。ちょっとくじけそうです。

（東京都　坂倉典夫さん）

結果を求めるだけでなく、「Linuxをいじる」ということを目的にしてもいいのではないでしょうか。Mac OSとはまったく文化の違うOSに触れる、それだけでも立派な使い方です。自信をもって色々と試してみてください。応援しています。

インストールLinux

最近、「インストール依存症」です。Linuxはそのための道具と化している感があり、いったいどのディストリビューションがいいのか、日々苦悩しております。まあ、Red Hat系が充実していますから、“本家Red Hat”と思うと、今度はVineか…、いや、Turboはどうなんだ、LASER5はRelease 2が出た…と。OpenLinuxが動作しませんので、とりあえず、MLD4を購入して基本的な勉強を始めました。ext2にインストールして快調に動作。しかし、もう一つ欲しいというこの衝動。

（福島県　香西 薫さん）

ここまでたくさんのディストリビューションがあると選ぶのが大変ですね。担当もずいぶんたくさん試してきましたが、それぞれに得手不得手があるようで、すべての方にオススメというものはありません。ただ、どのディストリビューションでも、基本的な設定方法は共通していますので、蓄積した知識は無駄にはならないでしょう。って、同じLinuxなんだからあたりまえですね。

今月もお待ちしておりました

先日、このコーナーに掲載されたことを妻に自慢したら、謝礼はいくらかしら？　と真っ先に問いただされました。ささやかな私の趣味の範囲まで家計と結びつけるのは勘弁してくれ〜、と妻に言ったら、そのコーナーって情報誌の『ぴ○』のはみだし欄みたいなものネ。とあっさり一撃をくらってしまい、その後は何も言えませんでした。学生のころ、『ぴ○』に投稿するやつはなんて暗いやつだと思っていましたが、今は私も同類項なのかもしれません。トホホ…（『ぴ○』のファンの方ごめんなさい）。(^^;)

（山形県　遠藤浩二さん）

遠藤さんは毎月欠かさずおハガキを送ってくださっています。いつも楽しく読ませていただいております。

担当は学生時代「はみぴあ」が大好きでした。さすがに、投稿したことはありませんが…。人混みが苦手なので、考えてみれば「はみぴあ」のためだけに『ぴあ』を買っていたんですね。ところで、『ぴ○』って『ぴあ』ですよね。違ってたらどうしよう。

遠藤さんの奥様、すみません。謝礼はないんです（いまのところ）。不甲斐ない「読者の声」担当のせいなんです。お許し下さい。(__ __)

夢のLinuxファミリー

親子3人でLinuxに取り組んでいます。パソコン歴は13年くらいですが、息子のほうが理解は早いようで、歳の差を感じる今日このごろです。パソコンは全部で7台動いています。

（三重県　早川 勇さん）

いいですね、お子さんと一緒にLinux。今では、小中学校でもPCを取り入れた授業があるそうで、担当も子供とLinuxで遊びたいものです。あ、その前に結婚しなきゃ。くぅ。

Intel i810でXFree86

i810チップセットで、X Window Systemを動かそうとしていますが、いまだに成功しません。どこかでHow toを掲載していただければありがたいです。

（兵庫県　木下光彦さん）

今月の特集1で試用したVAIOもi810チップセットです。i810は、XFree86 3.3.6からサポートされていますので、XFree86のバージョンをご確認のうえ、記事を参考にして設定を行ってみてください。幸運をお祈りします。

ディストリビューション詰め合わせ

最近のディストリビューションは、入手するための価格が上がったように思える。3〜4年前には、5000円も出せばほとんどのものが収録されたCD-ROMのセットが手に入った。最近は見なくなったがどこに行ったのだろうか？

（茨城県　岡田広幸さん）

数年前まで、InfoMagic社のLINUX Developer's Resource CD-ROMなど、SLSやSlackwareなどの有名ディストリビューションを収録した低価格CD-ROMを購入するのが、手軽にLinuxを入手するのによい方法でした。

ここで、ちょっと宣伝。3月25日発売のLinux magazineムック「Linux スターターコレクション」（1886円＋税）には、Red Hat 6.1J改、Turbo 4.5、LASER 6.0 Rel.2、Slack 7の4ディストリビューションが付録として収録されています。今までで特に評判のよかった特集や、人気連載「viはじめました」も全話掲載しています。本誌共々、こちらもよろしくお願いいたします。

BSDユーザーも読んでます

BSDユーザーです。「PostgreSQLを極める」読んでますヨ！

（徳島県　庫元孝文さん）

FreeBSDユーザーの千葉県、佐野 貴さんからもおハガキをいただきました。非Linuxユーザーの方にもお楽しみいただける記事をお届けできるよう頑張ります。