News Express

Distribution Software Hardware Headline Event

IA-64でLinuxは飛躍するか？

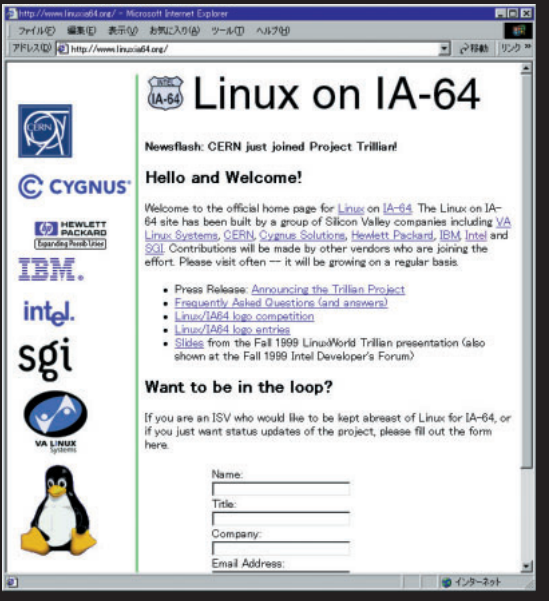
1991年にLinus B. Torvalds氏が最初のLinuxを作ってから9年が経った。Linuxは当時の主流だったi386をターゲットに開発がスタートした。

その頃、PCのOSは、MS-DOSが主流であり、シングルタスク、メモリ空間は640Kバイト、コンソール画面という非常に制限された環境だった。i386は、マルチタスクのサポート、4Gバイトのメモリ空間を持った完全なる32ビットCPUなのに、その真価を発揮していなかったのである。

その後、i486、Pentium、Pentium II、Pentium IIIと新しいCPUが出てきた。アーキテクチャがめざましく改良され、その処理スピードは何百倍にもなった。また、Windowsの登場は32ビットCPUの能力を必要としていた。しかし、CPUのクロックが1GHzに届こうとしている今も、基本的には高速のi386として使っているのが現状だ。

ところが、ハードウェアの進歩はすごいもので、メモリやハードディスクの実装容量が、i386アーキテクチャの限界を超える日が近くなっている。そして、もうすぐ出荷される64ビットCPUのItaniumは、i386で確立されたIA-32アーキテクチャを捨てて、IA-64アーキテクチャを採用する。

IA-64用のLinuxはすでに稼働し始めていて、この土俵ではLinuxが先頭を走っている。今までサーバ用途では、ある程度の地位を占めてきたLinuxだが、これを機会に一層の飛躍が望めないか、今後に期待がかかっている。



Hardware

Linux搭載の小型サーバ「PLASMA2000」

発売日 1999年11月25日

発売 プラネックスコミュニケーションズ株式会社
TEL 0120-415976
価格 オープン価格（参考価格12万8000円）

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

プラネックスコミュニケーションズから、OSにLinuxを採用したコンパクトなインターネット／イントラネットサーバ「PLASMA2000」が発売された。オープン価格だが、同社による参考価格は12万8000円。

OSにRed Hat Linux 6.0を採用し、連続稼動が必要な環境でも、安定した性能を発揮できるとしている。またインターネットと接続する際に問題となるセキュリティについては、ファイアウォール機能により対応している。

WWW、メールなどインターネット接続環境に必要なサーバ、Samba、Netatalkなどイントラネット環境向けファイル／プリントサーバの両方を備える。またシリアルポートに接続したモデムやTAの共有も可能だ。

コネクタを備えてはいるが、本機にキーボードやマウス、モニタを接続する必要はない。「PLASMA」というユーティリティソフトが動作しており、初期設定や運用時の設定変更は、Webブラウザを用いてリモートから行うようになっている。

CPUにCyrixのMediaGX 133MHzを採用し、メモリは64Mバイト、ハードディスクは標準で4Gバイトを搭載している。サイズは290mm（W）×225mm（H）×53mm（D）と、ほぼ電話帳サイズだ。また縦置き、横置きどちらも可能なため、設置場所に困ることはない。

Hardware

ウィルスチェッカをバンドルしたLinuxサーバ「Xuni-LS」

発売日 1999年11月4日

発売 株式会社テンアートニ
TEL 03-5298-2924
価格 オープン価格

URL http://www.10art-ni.co.jp/

テンアートニから、ウィルスチェッカをバンドルしたLinuxサーバ「Xuni-L S」（テンユニエル エス）が発売された。

コンパックのサーバ専用機ProLiant800に、英語版のRed Hat Linux 5.2をインストールし、アンチウィルス専門メーカーのトレンドマイクロが開発したサーバタイプのウィルスチェッカ「InterScan VirusWall」をバンドルしたもの。サーバ専用ハードウェアとLinuxの組み合わせにより、安定した運用を実現できるとしている。セキュリティを重視して、インストールされているソフトウェアは、最小限に限定されている。

InterScan VirusWallは、サーバ上で動作し、メールやダウンロードしたファイルに混入しているウィルスを発見、除去できるウィルスチェッカ。アンチウィルス機能以外にも、スパムメールの防止など、LAN全体のセキュリティを高める機能を持つ。今までにSoaris版、WindowsNT版などがリリースされている。Linux版の開発には、テンアートニが協力している。

基本モデルでは、CPUにPentiumIII 500MHzを1個（最大2個）、メモリは256Mバイト、ハードディスクは9.1Gバイト×2を搭載している。またUPSも装備しており、ハードディスクの二重化とあわせて、フォールトトレラントに配慮している。

Software

Webサイトのアクセスログを解析「SiteTracker 4.1」

発売日 1999年11月8日

発売 株式会社アスキー
TEL 03-5351-8590
価格 9万8000円〜

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

アスキーは、ホームページの閲覧状況などを解析する「SiteTracker 4.1」を発売した。1つのサイトを扱うプロフェッショナル版が9万8000円、複数のサイトを扱えるエンタープライズ版が19万8000円から124万7500円。

Linux（x86、glibc2）、Windows 95／98／NT（x86）、Solaris（SPARC、x86）、FreeBSD 2.x、Cobalt Qubeの各プラットフォームに幅広く対応している。Apache、Microsoft IIS、Netscapeなど主要なWebサーバプログラム上で動作可能だ。またエンタープライズ版は、プロキシ（Squid、Microsoft Proxyなど）、FTPサーバ（Wu-ftp、Microsoft IISなど）にも対応している。

SiteTrackerはサーバマシン上で動作し、操作はWebブラウザで行う。ログファイルをローカルにコピーせずに操作できるため、帯域を無駄に使うことがない。

ユーザーがどのようなキーワードで検索したか、アクセス頻度の高いサイトはどこか、初めて訪れたユーザーがどこから来たか、などの分析が可能。多種のサマリレポート機能を用いて、さまざまな視点の統計が行える。分析結果は、グラフ化して見ることができる。また、Excel、Word、Accessなどへのデータエクスポートが可能で、データの再利用が簡単に行える。

Hardware　　発売日　1999年11月16日

デュアルCPU構成が可能なAlphaプロセッサ搭載サーバ「AlphaServer DS20E」

URL　http://www.compaq.co.jp/

発売	コンパックコンピュータ株式会社
TEL	0120-018589
価格	225万3000円〜

Alpha 21264プロセッサを最大2個搭載可能なサーバ「AlphaServer DS20E」がコンパックコンピュータから発売された。価格は225万3000円から。DS20Eは、同社のサーバ中のエントリーラインに属する製品。OSは、コンパックTru64 Unix、OpenVMS、Linuxに対応。

標準モデルは、500MHzのAlpha 21264（EV6）プロセッサを1個（最大2個）、メモリを256Mバイト搭載している。ハードディスクはオプション。

2000年初頭には、667MHz（EV67）のプロセッサ搭載モデルを追加予定。600MHzのモデルからは、プロセッサボードの交換でアップグレードが可能。また、3Dグラフィックス機能を付加したワークステーションモデルも追加される予定。

ラックマウントでの使用を前提にした設計で、特にラック実装時は従来機のほぼ半分の省スペースを実現している。Alphaプロセッサの浮動小数点演算能力を活かした科学技術計算用途、または大規模な商用サーバに最適としている。

Hardware　　発売日　1999年11月11日

小型、低価格のスイッチングハブ／デュアルスピードハブ「FX-05SMC／FX-08SMC／DNS-502／DNS-802」

URL　http://www.planex.co.jp/

発売	プラネックスコミュニケーションズ株式会社
TEL	0120-415976
価格	オープン価格

プラネックスコミュニケーションズは、小型で低価格のスイッチングハブ2機種（FX-05SMC、FX-08SMC）、スイッチングポート付きデュアルスピードハブ2機種（DNS-502、DNS-802）を発売した。オープン価格だが、同社による参考価格は、6900〜9900円。

100BASE-TX／10BASE-T、全二重／半二重（DNSシリーズは1ポートのみ）を自動認識するため、既存の10BASE-T環境を無駄にせず、段階的な100BASE-TX化が行える。スイッチング方式には、ストア＆フォワードを採用。FXシリーズは最大1000、DNSシリーズは最大8000のMACアドレスを自動学習可能。

サイズは、5ポートモデルが114mm（W）×82mm（D）×28mm（H）、8ポートモデルが160mm（W）×82mm（D）×28mm（H）と非常にコンパクト。底面の磁石で、机の脇などにも設置できる。

FX-08SMC

「日刊アスキーLinux Weekly News」

12/6スタート！▶▶

http://www.linux24.com/

Linuxのあらゆる情報をデイリーで発信している日刊アスキーLinux（http://www.linux24.com/）から、最新ニュースをメールで配信するサービス「日刊アスキーLinux Weekly News」が始まった。

毎週月曜日の朝に、日刊アスキーLinuxに掲載された記事から1週間分をまとめてダイジェストでお届けする。

そのほか、米国の有力Linuxサイト「Linux Today」のコラム、「Linux.com」のTips、日刊アスキーLinuxで好評の「今日のコラム」、今週のアンケート、今週のイベント、アスキーの雑誌に掲載されたLinux関連記事などが掲載される。

週刊メールニュース「日刊アスキーLinux Weekly News」の申し込みは、日刊アスキーLinux（http://www.linux24.com/）で受け付けている。

News Headline from 日刊アスキーLinux

www.Linux24.com

米Red Hatと米Cygnus Solutionsが合併

1999年11月16日

米国の大手Linuxディストリビュータ Red Hatと、オープンソースの開発環境や組み込み用ソフトウェア開発で有名なCygnus Solutionsが合併に合意した。法的手続きを経て正式に合併完了するのは、2000年1月27日となる。合併後は、Red Hatの社長Matthew Szulik氏が社長兼CEOに就任する。

Cygnus Solutionsは、日本法人として、「日本シグナスソリューションズ」を構え、組み込み用リアルタイムOS「embedded Cygnus operating system」や、UNIX、Windowsなどのマルチプラットホームに対応したコンパイラ／開発ツール「GNUProツールキット」、ソースコード解析ツール「ソースナビゲータ」などを手がけている（いずれも国内における製品名）。

今回の合併で気になるのが国内における展開だが、レッドハット、日本シグナスソリューション両社とも、合併発表後間もなく、現段階では発表できるものがないことに加え、2000年の合併完了までは法的にも具体的なコメントができないことを告げた。よって、正式な発表は2000年1月27日以降になる。

Red Hat （http://www.redhat.com/）

Cygnus Solutions

（http://www.cygnus.com/）

「LASER5 Linux 6.0」配付版がマザーボードにバンドル配付される

1999年11月15日

レーザーファイブは、国内大手ベンダー数社のマザーボードへの「LASER5 Linux 6.0」配付版のバンドル配付を開始したと発表した。バンドルされる「LASER5 Linux 6.0」配付版は、CD-ROMと簡単なインストール案内が付いている。なおバンドル先のベンダーは、契約により公開できないとのこと。

なお同社では、今後マザーボード以外のハードウェアへのバンドルも検討している。

レーザーファイブ

（http://www.laser5.co.jp/）

オライリー・ジャパンが、「オープンソース・ソフトウエア」の日本語Web版を公開

1999年11月12日

オライリー・ジャパンが、書籍「オープンソースソフトウェア ― 彼らはいかにしてビジネススタンダードになったのか」の邦訳を同社Webサイト上で公開している。同社によれば、原著者の意向（原書のテキストファイルが、今年5月末に米オライリー社のWebサイトで公開）を翻訳書にも反映させ、本の内容、オープンソースの考え方を広く認知してもらい、この運動の本質を理解してもらうために公開に至ったとのこと。

オライリー・ジャパン

（http://www.oreilly.co.jp/）

System Commander 4の無償バージョンアップキャンペーン

1999年11月8日

ソフトボートは「System Commander 4」から「System Commander 2000」への無償バージョンアップキャンペーンを、12月1日より開始する。キャンペーン期間は、System Commander 2000の発売前日まで。

System Commander 4は、Windows 98／95／3.1、WindowsNTや、Linuxを1台のパソコンに共存させ、切り替えて使うことができるマルチOSブートユーティリティ。同製品は2000年2月にSystem Commander 2000にバージョンアップされる予定。主なバージョンアップ機能は、「Windows 2000への対応」、「NTFSやLinuxファイルシステムのパーティション操作のサポート」、「Activate Move／Copy」、「Undoウィザード」、「ネットワークインストール」、「OSウィザード」の改善等が予定されている。

また、System Commander 4の姉妹製品Commander Premium Packについても、System Commander 2000への無償バージョンアップキャンペーンが行われる。

ソフトボート （http://www.softboat.co.jp/）

アクセスが組み込み用Webブラウザ「NetFront for Linux」を発表

1999年11月8日

アクセスは組み込み用Webブラウザ「NetFront for Linux」を発表した。12月15日より、アクセスのWebサイトにて評価版の無償配布を開始する。プログラムの実行コードサイズは、約2Mバイトと小さい。

NetFrontは、情報家電向けにコンパクトに実装されたWebブラウザで、国内外の約50製品に採用され、すでに400万台以上が出荷されている。Linuxが組み込み分野でも注目されているため、評価版を無償配付し、市場のニーズの掘り起こしを行うという。今後、市場の要求を見て、2000年中に製品版をリリースする予定。価格は現時点では未定。

NetFront for Linuxは、NetFrontの最新バージョンである2.5をLinuxに移植したもので、cookie、Javascriptなどの機能をサポートしている。

動作環境はカーネル2.0.23以降、glibc 2.0または2.1、GTK+1.2、zlib 1.1が必要。LASER5 Linux 6.0、Vine Linux 1.1での動作を確認済み。

アクセス （http://www.access.co.jp/）

Linux用のジャーナルファイルシステムがリリース

1999年11月8日

米Namesys（The Naming System Venture）は、Linux用のジャーナルファイルシステムを搭載した「ReiserFS」の

正式版をリリースした。ReiserFSはファイル名だけではなく、ファイルそのものをB-treeに収納している。またパフォーマンスが向上しているという。

ReiserFSの開発には、SuSEが協力しており、SuSE 6.3にはReiserFSが搭載され、ジャーナリングがサポートされる予定。

ReiserFSはNamesysのサイトからダウンロードできる。

Namesys
(http://www.devlinux.org/namesys)

「SoundBlaster Live!」のLinux用ドライバをGPLに基づき公開

1999年11月8日

米Creative Technologyは、「SoundBlaster Live!」のLinux用ドライバの正式版を、GPL（GNU General Public License）に基づき公開した。今までも、同製品のドライバのβ版は公開されていたが、カーネルのバージョンによる制限があった。今回のREADMEファイルには制限については記載されていない。

今回、同社は「Creative Open Source page」というページを持つサーバを公開した。今後、オープンソースのソフトウェアに関しては、このサーバで公開されることになると思われる。現在、SoundBlaster Live!用のドライバのほかに、PC-DVD Dxr2用のβ版ドライバが公開されている。

Creative Technology
(http://www.creative.com/)
Creative Open Source page
(http://opensource.creative.com/)

Linux上からMacintoshフォーマットのディスクを読み書きする「Linuxmac」

1999年11月7日

米LinuxOneは、Linux上からMacintoshフォーマットのディスクとCDを読み書きするための製品「Linuxmac」を発表した。

Linuxmacは、マウスによるドラッグ＆ドロップ、ポイント、クリックをサポートしていて、ファイルのコピーであれば単にファイルアイコンをフォルダにドラッグするだけでよい。Linuxmacの全機能は、マウスで操作可能だという。

LinuxOne （http://www.linuxone.net/）

米SGIが「Linux Kernel Crash dumps」プロジェクトの開始を発表

1999年11月7日

米SGIは「Linux Kernel Crash dumps」プロジェクトの開始を発表した。Linux Kernel Crash dumpsは、システムがクラッシュした場合、カーネルメモリイメージを保存し、リブートするときにそのメモリイメージを回復、分析することによって、その原因を探ることができるようにしようというもの。システムがクラッシュした場合の、信頼性が高い調査方法となることを意図している。

メモリイメージは、SCSIディスクのswapパーティションへのポインタとして、/dev/vmdumpの中に保存される。そして再起動の際に、lcrash（Linux Crash）というプログラムにより、swapパーティションがマウントされる前に回復され、レポートが/var/log/vmdumpの中に作成される。

「raw I/Oのためのパッチ」、「カーネルの変更、ライブラリ、コマンド」、「/etc/rc.d/rc.sysinitへのパッチ」の3つから構成されており、バージョン1.0のソースコード、RPMともにSGIのサイトからダウンロードできる。

SGI （http://www.sgi.com/）
SGI - Linux Kernel Crash dumps
(http://oss.sgi.com/projects/lkcd/)

ディアイティがネットワーク設定ソフト「LINSE」を配布開始

1999年11月5日

ディアイティは、Linux上で動作するネットワークセットアップユーティリティ「LINSE」の配布を始めた。

LINSEは、X Window System上で、メールの設定やユーザーの追加／削除、DHCPサーバ、RADIUSサーバ、プロキシサーバ、IPマスカレードの設定などをGUIによって行うことができる。動作環境はTurboLinux4.0で、Red Hat Linux6.0も確認中。順次ほかのディストリビューションでも確認を行っていくという。

LINSEはまた、GPLライセンスのオープンソースとして提供される。Webからのダウンロードは無料。CD-ROMによる配布は入手費用として5000円が予定されている（開始は12月1日が予定されている）。

なお、PCやPC周辺機器を販売しているロジテックが、LINSEを組み込んだ製品の出荷を予定している。

ディアイティのLINSE Webページ
(http://www.dit.co.jp/linse/)

テンアートニらが、開発したシステムをオープンソースとして公開

1999年11月2日

外食産業のニュートーキヨーと、JavaとLinuxでシステムインテグレートを行うテンアートニは、共同で開発したJava ServletとLinuxによる受発注システム「セルベッサ（CERVEZA）（スペイン語でビールの意）」を、オープンソースとして公開した。

提供の方法は、ソースコード、実行プログラム、インストールの説明書がそれぞれダウンロード可能なほか、セルベッサの使用、改変再利用、改変再販売も無償で行うことができる。また、テンアートニでは、改変したソースコードを再登録する仕組みを用意し、システムのバリエーションを拡げていく方針。テンアートニはこれにより、サポートサービスなどの収益をねらう。

セルベッサは、バックエンドのデータベースにOracle8 for Linuxを採用し、Java ServletやXMLによるアプリケーションサーバを構築。ニュートーキヨーのチェーン200店舗を結ぶ。

ニュートーキヨー
(http://www.newtokyo.co.jp/)
テンアートニ
(http://www.10art-ni.co.jp)

Distribution ▶▶▶

Windowsパーティションにインストールできる「Linux MLD 4」12月10日発売

メディアラボは、Windowsとの親和性が高いLinuxディストリビューション「Linux MLD 4」を8800円で発売する。

「Linux MLD 4」は、Windows 95／98、Windows NT 4.0から、インストールできるLinuxパッケージの最新版。今回のバージョンアップで、カーネルは2.2.12、X Window Systemの標準環境はKDE 1.1.2になった。XFree86は、最新版の3.3.5を採用し、より多くのビデオカードでX Window Systemが利用できるようになった。商用ソフトウェアは、「Wnn6 Ver.3」、「DynaFont（5書体）」がバンドルされる。

Windows 95／98からインストールする場合、インストーラがWindowsからハードウェア情報を読み、デバイスを自動設定する。また、WindowsのFAT領域にインストールできるため、パーティションを切り直す必要もない。さらに、Windowsのネットワークで、ほかのマシンに接続されたCD-ROMドライブが利用できれば、ネットワークインストールをすることもできる。

FAT領域にインストールするとはいえ、Linuxの標準ファイルシステムであるext2のディスクイメージを1つのファイルとしてFAT上に作り、そのイメージをloopbackデバイスによってマウントするので、ファイル名なども通常のLinuxと全く同じように利用できる。また、通常のディストリビューションと同じように、ext2パーティションを作って、そこへインストールすることも可能だ。

メディアラボ （http://www.mlb.co.jp/）

Caldera Systems、ネオナジー「OpenLinux 2.3 日本語版」を発売

米Caldera Systemsはネオナジーと提携し、11月27日に「OpenLinux 2.3 日本語版」を発売した。価格は1万2800円で、販売をネオナジーが行う。カーネルは2.2.10。デスクトップ環境はKDE 1.1.2で、ディスプレイでの視認性にすぐれた日本語フォント「さいもん」を採用。商用アプリケーションは、かな漢字変換ソフトウェアATOK12 SE、Wnn6 Ver.3、日本語ワードプロセッサdp/NOTE、パーティショニングツールPartitionMagic Caldera Edition、ブートマネージャBootMagicなどが付属する。付録CD-ROM No.2にFTP版を収録したので、インストールする場合は、42ページからの記事を参考にして欲しい。

ネオナジー （http://www.openlinux.ne.jp/）

「Slackware 7.0」がリリース

10月29日に、Slackware 7.0がリリースされた。Slackware 7.0はコアコンポーネントに、カーネル2.2.13とglibc 2.1.2を採用しており、X Window Systemとデスクトップ環境には、XFree86 3.3.5、KDE 1.1.2、October Gnome（1.0.53）が使われている。

Slackwareの前バージョンは4.0であったが、新バージョンはいきなり7.0となった。Slackware ProjectリーダーのPatrick Volkerdingは、次のように語っている。

「他のディストリビューションが、マーケティング目的のためにバージョンナンバーを上げているため、（バージョンナンバーが若い）Slackwareの内部のソフトウェアは、古いものだという誤解を招いている。Slackwareのライブラリ、コンパイラなどは、一般に6.0といわれているものよりも、良いもののはずだ。今後、ほかのディストリビューションがバージョンナンバーを上げなければ、もうこういう事はやらないつもりだ」

インストーラはテキストベースで、デザインは4.0までのバージョンとまったく同じになっている。

SlackwareのWebサイトのほか、いくつかのミラーサイトから、ダウンロードが可能だ。またCD-ROM4枚組のSlackware 7.0がWalnut Creekから39.95USドルで発売されている。

Slackware (http://www.slackware.com/)

「TurboLinux Server 日本語版 6.0」12月15日発売開始

ターボリナックス ジャパンは、11月29日に、「TurboLinux Server 日本語版 6.0」、「TurboLinux 日本語版 6.0 SOHO Edition」を発売すると発表した。価格は、「TurboLinux Server 日本語版 6.0」が2万9800円、「TurboLinux Server 日本語版 6.0 SOHO Edition」は1万9800円。教育／研究機関を対象にしたアカデミックプライスは、それぞれ1万9800円と1万4800円。

両製品は、セキュリティや安定性を重視した、サーバ構築用Linuxディストリビューションの最新版だ。旧バージョンのカーネルが2.0.38だったのに対し、新バージョンでは2.2.13を採用。これにより、マルチCPUでのSMP環境でパフォーマンス大幅に向上した。「TurboLinux Server 日本語版 6.0」は、セキュリティ上危険性のあるパッケージを排除し、Enhanced Software Technologiesの商用バックアップソフトウェア「BRU」をバンドルする。

また、今回ラインナップに加わったエントリーモデルの「TurboLinux Server 日本語版 6.0 SOHO Edition」には、「BRU」はバンドルされないが、ホライズン・デジタル・エンタープライズの「HDE Linux Controller for TurboLinux 」をバンドルする。「HDE Linux Controller for TurboLinux 」は、Webブラウザからサーバ設定や起動・停止、バックアップなどが行えるツールだ。

両製品とも、対応ハードウェアの認識・設定、インストールまでの範囲で、WebおよびE-Mailによるサポートを90日間、3インシデントまでサポートする。

既存のTurboLinux Server 日本語版ユーザー、同アカデミックユーザーには、同社のWebサイトを通じアップグレード価格での販売が行われる。

なお、同社は11月29日に東京都渋谷区3-3-5 モリモビル5階に移転した。

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

「Corel LINUX 1.0」11月30日より北米で出荷開始

カナダのCorelは、11月15日に「Corel LINUX 1.0」をリリースした。ダウンロード版は同日より、製品版は11月30日より出荷される。

Corel LINUXは、Debian GNU/Linuxをベースにしたディストリビューション。英語版のCorel LINUXは、カーネルに安定版の2.2系列を採用し、デスクトップ環境にKDEのカスタマイズバージョンを用いている。さらに入力する項目がわずか4カ所のインストーラ、Windowsネットワークの参照、FTP機能の統合を行ったオリジナルのファイルマネージャなどCorel独自のソフトウェアが含まれる。

これらの機能を含んだダウンロード版は、同社のFTPサイトから取得できる。ダウンロード版に、Netscape Communicator、Adobe Acrobat Reader、Corel WordPerfect 8 for Linux（ライト版）や、そのほか再配布可能なソフトとソースコードCD-ROMを加えたスタンダード版は59.95USドル、スタンダード版にCorel WordPerfect 8 for Linux（完全版）、BRU Backupソフト（パーソナル版）、Civilization（ゲーム）、4Front Technologiesの拡張OSSサウンドドライバなどを加えたデラックス版は89.95USドル。

なおCorelは11月1日の記者会見で、時期は未定としながらも、Corel LINUXの日本語化を行うと発表した。日本での取り扱いは、メディアビジョンが行う。

Corel（http://www.corel.com/）

「Storm Linux Betaリリース4」発表

カナダのStormix Technologiesは、Storm Linuxベータリリース4を発表した。10月にリリースされた、ベータリリース3の不具合を修正したもの。同社のFTPサイト、またはミラーサイトから、ダウンロード可能。

Storm Linuxは、Debian GNU/Linuxベースのディストリビューションで、Storm Administration System（SAS）と呼ばれるツールが特徴だ。SASにより、ネットワーク上のマシンをリモートで管理、設定することができ、SASのインターフェイスは、キャラクターベース、GUIの両方をサポートしている。Storm Linuxのインストールは、SASを利用したグラフィカルインターフェイスで行う。インストール後は、普通のDebian GNU/Linuxと同等の機能を持つ。

また同社は、製品版のStorm Linux 2000に関するアナウンスも近日中に行う予定。StarOffice 5.1a、VMware（試用版）、Applixware Office 4.4.2（試用版）など多数のソフトをバンドルする。

Stormix Technologies（http://www.stormlinux.com/）

Products

OSを選ばないIDE接続のミラーリング専用RAIDディスク

トラストガード/TGDF

最近のハードディスクは容量増加が著しいが、故障時の損失も増大しているので、ハードウェアRAIDで信頼性を高めたいというニーズは強まっている。Linuxの場合、専用ドライバのいらない本機のようなRAIDディスクが、こうしたニーズに対するひとつの解といえよう。

製品名　トラストガード/TGDF
価格　9万9800円（13Gバイト／1999年12月末までのキャンペーンモデル）
12万8000円（13Gバイト）～18万2000円（34Gバイト）
問い合わせ先　株式会社エヌエスピー研究所
E-mail: trustguard@nspi.co.jp、TEL. 0120-797-112
http://www.nspi.co.jp/

トラストガード/TGDFは、IDEインターフェイスに接続するミラーリング（RAID1）専用のRAIDディスク・ユニットである。ミラーリングとは、2台以上の記憶デバイスの間でまったく同じデータのコピーを保持することで、1台が故障してもデータのアクセスを途絶させない技術である。本機の場合、1インチ厚の3.5インチIDEハードディスクを2台まで内蔵し、一方のハードディスクが故障しても、残ったハードディスクで稼働することによりシステムを運用し続けられる。

今回評価したのは、1999年12月末までの限定キャンペーンモデル（13Gバイトのみ）である。なお、試用機は出荷前の試作品なので、仕様や性能が製品版と異なる可能性がある。

Linuxでも専用ドライバは必要なし

ハードウェアでRAIDを実現する製品には、PCIバスなどのシステムバスにRAIDコントローラを接続するタイプと、SCSIやIDEのケーブルにRAIDコントローラを接続するタイプに大別できる。本機は後者のタイプに属し、PCから見ると単なるIDEハードディスクとして認識される。そのため、PC側のIDEハードディスク用ドライバが存在し、かつハードウェアレベルでの接続に問題がなければ、利用可能だ。つまりOSごとに本機専用のドライバを必要としないというメリットがある。もちろんLinuxでも利用できるし、そのほかのOSについても、**表1**のように動作保証されている。

シリンダ／ヘッド／セクタ数あるいはLBAのセクタ数などのジオメトリパラメータは、内蔵ハードディスクのものがそのままPCに伝えられるようだ。メーカーによれば、一般的なIDEデバイス同様、本機とほかの

写真1
フロントパネルを空けて内蔵ハードディスクを取り出すところ。ハードディスクは専用アダプタで装着されており、鍵を開けて取り外す。装着時には、鍵を閉めると自動的にハードディスクの電源がオンになる。ディスクの冷却にも注意が払われており、2つの空冷ファンを装備したり、熱伝導のよい金属製ケース（クロームメッキ）を採用したりしている。

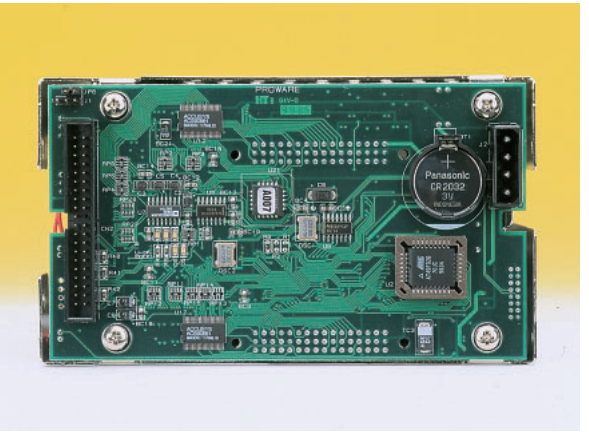


写真2
背面を覆っている金属カバーを取り外したところ。ユーザーが触れるのは、IDEコネクタと電源コネクタ、マスター／スレーブ設定用ジャンパポストだけだ。普通のIDEハードディスクと同じであり、設定やケーブルの接続は容易といえる。基板上にあるほかの部品は、RAIDのコントロール回路の一部で、この裏面に主な回路が実装されている。

IDE／ATAPIデバイスを1本のケーブルに接続して運用できるという。セットアップからソフトウェアのインストール、運用まで、単体のIDEハードディスクと同じ扱いができるわけだ。

性能は単体のハードディスクとほぼ同じ

本機のフロントパネルには、各ハードディスクごとに電源やアクセス、故障などを示すインジケータLEDが備わっている。ドライブが故障してアクセス不能になると、故障を示すLED（Fault LED）が赤く点灯してブザーが鳴り出して、ユーザーに知らせる。その際、PCの電源を止めずに故障したハードディスクを取り出して、新品に交換できる。その後、無事だったハードディスクから新品へデータをコピーするリビルドという作業が自動的に行われる。リビルド中でも、PCから本機へは読み書き可能だ（もちろんアクセス速度は遅くなるが）。つまりドライブが1台故障してから回復するまで、PCを止めることなく運用を続けられるわけだ。なお、PCからのアクセスがまったくない状態で13Gバイトモデルの試用機がリビルドを完了させるのに、約1時間20分かかった。

RAIDで気になる性能については、内蔵ハードディスクを単体で使うのとほぼ同じといえそうだ。Red Hat Linux 6.1JにてBonnieというベンチマークでテストしたところ、シーケンシャル書き込みでは13%ほど本機のほうが単体のハードディスクより遅かったが、逆に読み出しはわずかに速いくらいで、実用上問題はないだろう（転送レートは16M～17Mバイト/秒だった）。ただしこれはあくまでもUltraDMA/33での結果で、PIOモードでは1Mバイト/秒を切ることすらあり、実力をまったく発揮できない。本機はなるべくUltraDMA/33モードで使いたいところだ（なおキャンペーン以外の通常モデルはUltraDMA/66に対応している）。

小規模システムへ手軽にRAIDを導入するのに好適

SCSIディスクを用いる上位機種のTGDPシリーズと比べると、本機は容量や速度を高めるRAIDレベルに対応していない、故障発生を遠隔地に知らせられない、といった制限がある。しかし、RAIDに慣れていないユーザーが比較的容易に扱えることもまた事実だ。そのほか、ディスクの予備を用意しておけば、ディスクを交換するだけで、その時点でのバックアップを取れるという副次的な機能もあって便利だ。本機は、クライアントPCや小規模なネットワークでのファイルサーバをIDEハードディスクで運用している環境で、ディスクサブシステムの信頼性を高めたい場合に好適だろう。

記録容量	13Gバイト～34Gバイト
PCとのインターフェイス	UltraDMA/66対応IDE（キャンペーンモデルのみUltraDMA/33）
内蔵ハードディスク	UltraDMA/66対応7200rpm IDEハードディスク
機能	ミラーリング（RAID1）、ホットスワップ可、ブザー音による故障の通知
装着可能なドライブベイ	5.25インチ・フルハイト
動作保証OS	Windows 98／NT／2000、Linux、SGI IRIX、Sun Solaris
最大消費電力	DC5V：7A、DC12V：4A（いずれも瞬間最大値）
外形寸法（mm）	149（W）×230（D）×85（H）
重量	約3kg（内蔵ハードディスク2台を含む）
保証期間	ハードディスク：1年（交換・即日配送）、空冷ファン：1年（修理・交換）、コントローラ部：3年（修理・交換）

表1　トラストガード/TGDFの主な仕様

翻訳魂

インターネット上の資源を活用したり、PC-UNIXを使いこなしたりするためには、英文を読み書きしなければならない場面が、まだまだ多い。英語が苦手で、憂鬱になっている人のためのお助けソフトがついにLinux／FreeBSDで利用できるようになった。

製品名	翻訳魂
価格	1万5000円（標準添付：1ライセンス） 追加ライセンス（別売）7800円（1ライセンス）
問い合わせ先	オムロンソフトウェア株式会社 TEL 044-246-6006 http://www.omronsoft.co.jp/

オムロンソフトウェアから11月5日に、Linux／FreeBSDに対応した日英・英日翻訳ソフト「翻訳魂」が発売された。WindowsやMacintosh向けの翻訳ソフトと決定的に違うところは、クライアント／サーバ型の構造を持つことである。そのため、クライアントとサーバを別のマシンで動かすといった、分散処理も可能である。

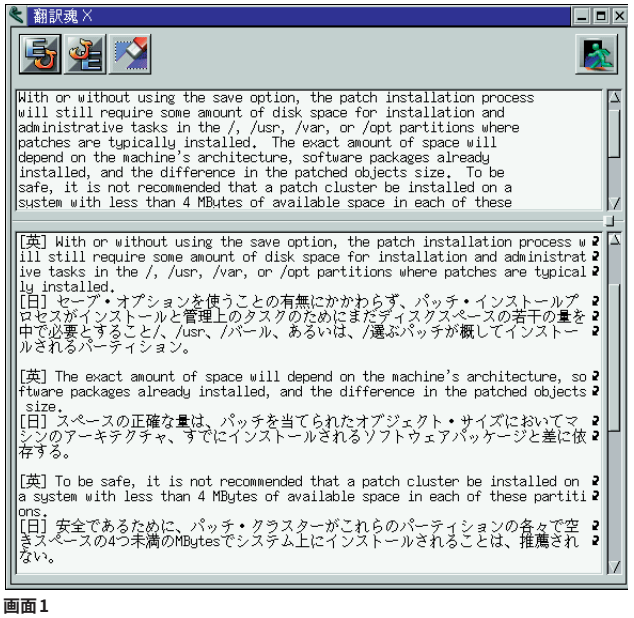
Mule（Emacs）クライアントはLisp言語で記述されていて、そのソースコードはGPLライセンスで公開されている。また、クライアント／サーバ間の翻訳のためのプロトコルも公開されているので、技術があれば、自分好みのクライアントプログラムを作成でき、WindowsやMacintoshからさえも利用可能である（インストール後のファイル/usr/local/honyaku/lisp/PROTOCOLにプロトコルが説明されている）。

インストールは、製品付属のインストールツールInstallをroot権限で実行する。インストール先のディスク（デフォルトでは/usr/localディレクトリ以下）に十分な空きがあれば、メニューに答えていくだけで、インストール、サーバ設定が終了する。今回はLASER5 Linux 6.0で試用した。

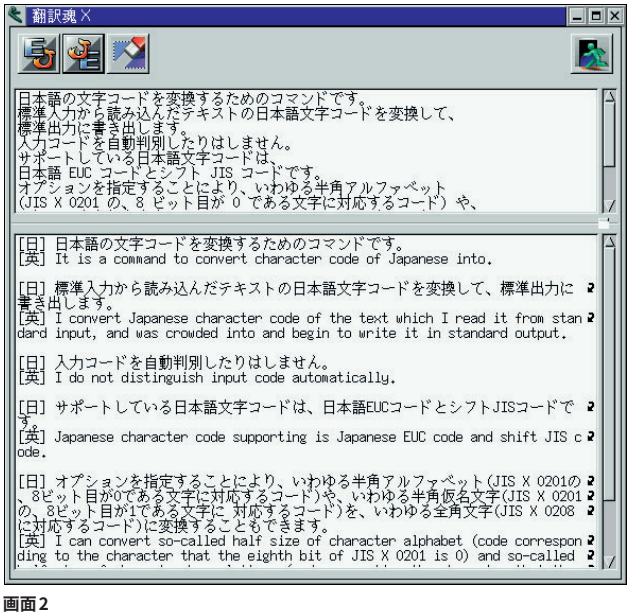
X版翻訳クライアント xhonyaku

X Window System環境が使えるならば、GUIをもつ翻訳クライアントを利用できる。コマンド名はxhonyakuである。使い方はいたって簡単。2分割されたウィンドウの上半分（原文入力領域）に文書を書き込み、文書に合わせて左上の「E／J」（英文和訳）または「J／E」（和文英訳）ボタンを押すだけである。するとウィンドウの下半分（翻訳結果表示領域）に翻訳結果が表示される。

まず英文和訳を試してみる。使っ



画面1
X版クライアントでの翻訳例。上側のウィンドウに元の文章を入れて、左上の「E／J」ボタンを押すと数秒後には結果が得られる。



画面2
和文英訳も同様に、日本語の文章を入力して「J／E」ボタンを押すだけで翻訳される。

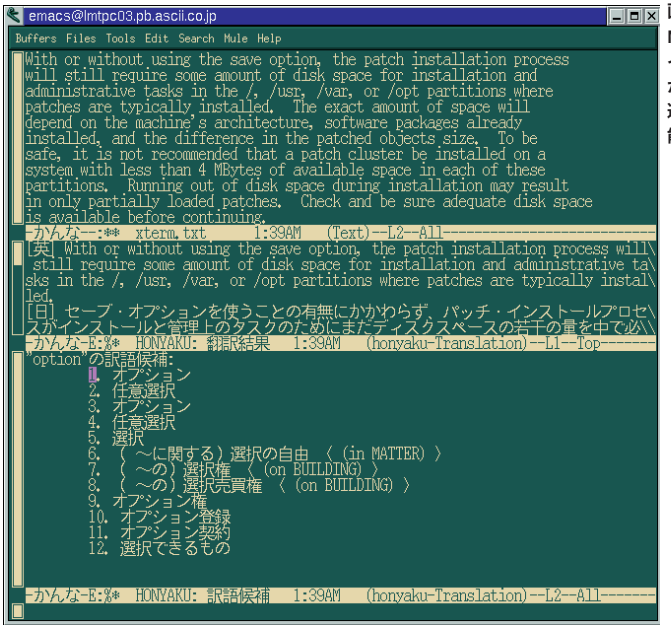
た例文は、Solarisの2000年問題パッチを当てるときに表示されたメッセージである。リモートログインしているターミナルからこのメッセージをコピー、翻訳クライアントのウィンドウの上半分にペーストし、「E／J」ボタンを押すと、翻訳結果が下のウィンドウに表示される（**画面1**）。

これくらいの文書であれば、Pentium 166MHzマシンで、5秒くらいで結果が出る。HDDの音などを聞いていると、辞書参照のためにかなり大きな資源を使っているようである。より速い処理を求めるなら、CPUパワーより、ディスクアクセスの速さに気を使ったほうがよさそうである。

日本語の文章が少しぎこちない感じだが、英日が対になって結果が表示されるので、意味を取りたい場合の手助けには便利である。コンピュータ専門語辞書も搭載されているので、コンピュータメッセージ、ネットワークセキュリティ情報（CERT Advisory）など、読むのに時間をかけられない文書を参照する場合、特に便利だろう。

次に和文英訳を試してみる。ちょっとしたツールを作り、いろんな人に使ってもらいたい場合、英文の説明書を付けておけば、思いがけない人にも使ってもらえるかもしれない。そんな場面を考えて例文を作り、翻訳させてみた結果が**画面2**である。

文章の入力は、コピーペーストで貼り付けてもよいが、ウィンドウの上半分（原文入力領域）ではXIMプロトコルをサポートしているので、xwnmoやkinput2などのIMを利用して直接入力もできる。翻訳結果は、同じくコピーペーストで別のアプリケーション（エディタなど）に貼り付けて最終的な文書に仕上げる。



画面3
Mule（Emacs）版翻訳クライアントを使うと、英単語から訳語候補を一覧表示、選択することで再翻訳が可能だ。

Mule版翻訳クライアント

「翻訳魂」の翻訳サーバの機能は、X版クライアントxhonyakuで使い切っているわけではない。

もうひとつの付属クライアント、エディタMuleから呼び出して使うMule版クライアントでは、訳語や品詞候補を表示しながらの再翻訳、単語辞書引きなどができる。また、このクライアントは、emacs-lispで記述されており、Muleが動作する環境があれば利用でき、カスタマイズすることも可能である。

Muleでは範囲を指定してから、英日なら［F9］を、日英なら［F10］キーを押すと翻訳される。Emacs（Mule）で、先ほどの英文を使用して訳語候補の表示を行ってみた（**画面3**）。翻訳結果が不自然だった場合に、英文の単語にカーソルを移動してスペースキーを押すと、訳語候補の一覧が表示され、その中から適切な訳を選ぶと再翻訳される。

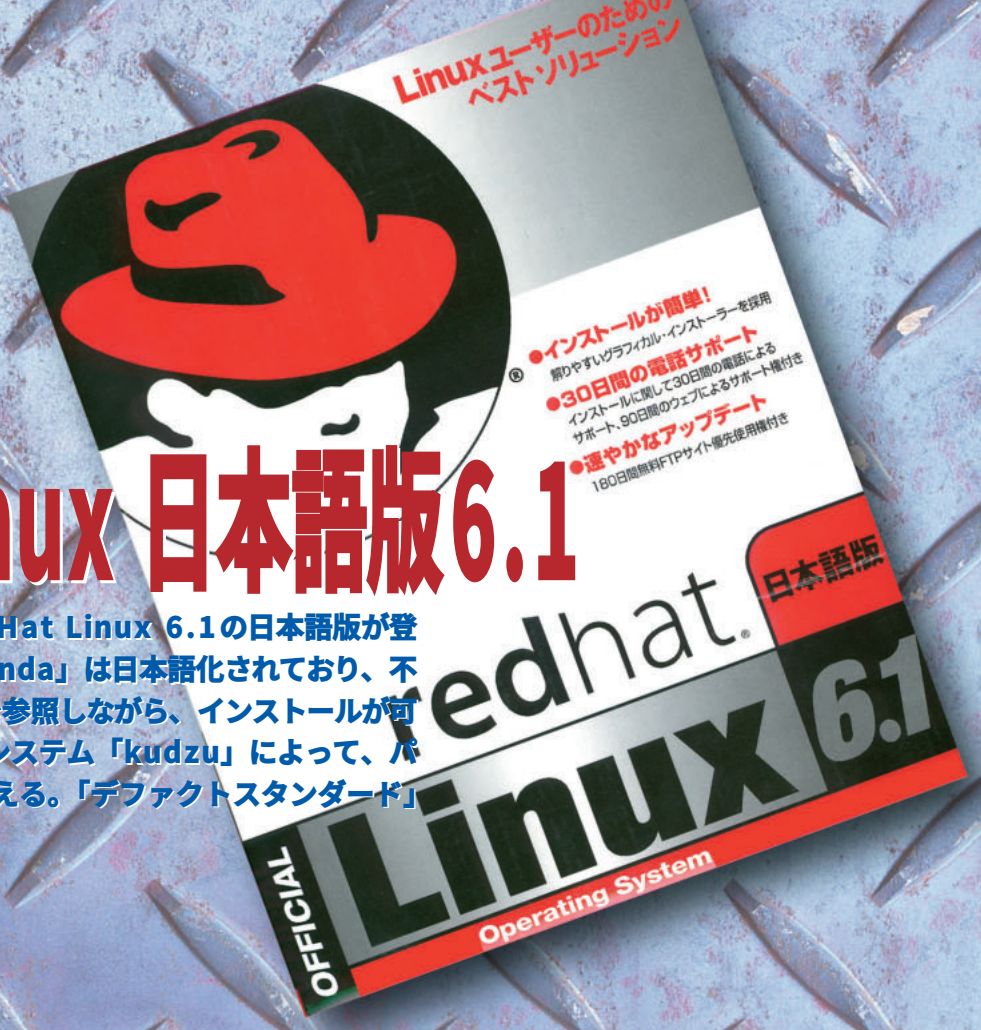
ほかに、青空文庫（http://www.aozora.gr.jp/）にあるような文学作品なども試してみたが、わからない部分は日本語のまま残したり、あまりに時間がかかる場合（複雑な文章）には、翻訳をあきらめるなど、無理のない軽快な翻訳を持ち味としているように感じた。

製品のもうひとつの顔であるマニュアルは、簡潔に要領よくまとめられていて好感が持てる。印刷された物は、インストールに必要な部分だけで、CD-ROM内のマニュアル（doc/ディレクトリ以下のドキュメント）が、すべてのマニュアルである。

ネットワークライセンス方式を採用しているため、1人で複数のウィンドウで実行したり、複数の人が同時に使うためには追加ライセンスを購入する必要がある。それにしても、昔のUNIX上の翻訳ソフトの値段を知っている人には、驚異的な低価格だろう。

Red Hat Linux 日本語版6.1

米国でのリリースから1カ月、Red Hat Linux 6.1の日本語版が登場した。GUIインストーラ「Anaconda」は日本語化されており、不慣れなユーザーでも画面上でヘルプを参照しながら、インストールが可能だ。またハードウェアの自動認識システム「kudzu」によって、パーツの交換、追加も手間いらずで行える。「デファクトスタンダード」Linuxの最新バージョンを紹介する。



OpenLinux 2.3 日本語版

1999年10月号で紹介したCaldera OpenLinuxの最新バージョンが日本語化されてデビューした。早い時期からKDEを採用し、使いやすいデスクトップ環境を提供してきたOpenLinuxだが、ディスプレイで読みやすい日本語フォントを採用するなど、そのこだわりは日本語版にも受け継がれている。グラフィカルなインストーラ「Lizard」も日本語化され、よりインストールがしやすくなった。日本に上陸したばかりの新ディストリビューションを紹介する。

Red Hat Linux 6.1日本語版

Red Hat Linux 6.1（英語版）が10月4日に発売されたとお知らせしたのは、先月号のことだ。その1カ月後の11月12日に、日本語に対応したRed Hat Linux 6.1日本語版が、1万2800円で発売された。

システムのコアコンポーネントであるカーネルは2.2.12、Cライブラリはglibc 2.1.2と米国版と同じバージョンを用いている。デフォルトのデスクトップ環境であるGNOMEは1.0.39に、またKDEは1.1.2と前のバージョンよりも新しくなっており、安定さを増している。さらにXFree86も3.3.5になっており、対応するビデオカードが増えている。新しめのビデオカードを使っていても大丈夫だ。

注目の新機能

ついにRed Hatも「Anaconda」と呼ばれるグラフィカルインストーラを採用した。テキストベースでもGUIでも、最終的に設定する項目は変わらないのだが、GUIのほうが簡単なように思えるから不思議だ。またインストール中、画面左側に常に説明文が表示されているため、初心者への負担はずいぶんと軽減されている。

バージョン6.1の目玉とも言える新機能が、ハードウェアの自動検出／設定プログラム「kudzu」だ。「くず」と読んで、たいしたものじゃないと思うのは、間違いだ。

システム起動時のメッセージに「Checking for new hardware」という行があるはずだ。これが「kudzu」の出力だ。ビデオカードやネットワークカードを変更すると、この出力の後で、ハードウェア環境の変更を検出したというメッセージと、それにともなう設定の変更内容が表示される。ユーザーは内容を確認したうえで、変更を指示できるというわけだ。もちろん、マシンを再起動する必要などまったくない。もし手元に予備のハードウェアがあるなら、ぜひ一度「kudzu」の機能を試してみたい。

付属の商用ソフト

製品版には、日本語入力ソフトウェアのATOK12 SE、Wnn6 Ver.3、日本語ワープロの一太郎Ark for Java Technology Preview、dp/NOTE、そしてTrueTypeフォントのDynaFontが5書体含まれている。

ただ、収録のしかたには、もう一工夫が望まれる。たとえば、ATOK12 SEはtarボールの形式で収録されている。できればRPMパッケージ化しておいて、コマンド一発で済むようにして欲しいものだ。

サポート

登録したユーザーは、インストール関する30日間の電話サポート、90日間の電子メールによるサポートが受けられる。また登録ユーザー専用のFTPサイトが、180日間優先的に利用可能だ。

付録CD-ROMにFTP版を収録

付録CD-ROM No.1に商用ソフトウェア、サポート権を含まないFTP版Red Hat Linux 6.1日本語版を収録した。

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

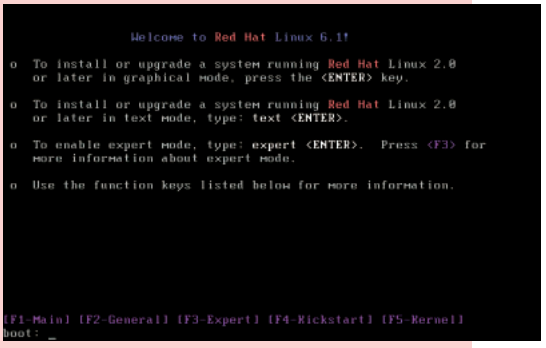
画面1
デスクトップ画面。デフォルトでは、GNOMEを採用している。

画面2
gEdit上でATOK12を使用中。

Red Hat Linux 6.1のインストール グラフィカルインストーラ Anaconda

CD-ROMブート可能なマシンでは、Red Hat Linux 6.1日本語版（以下Red Hat 6.1J）のCD-ROMをドライブに入れて再起動します。CD-ROMブートできない場合は、Windows上で起動用フロッピーを作ります。手順は以下の通り。

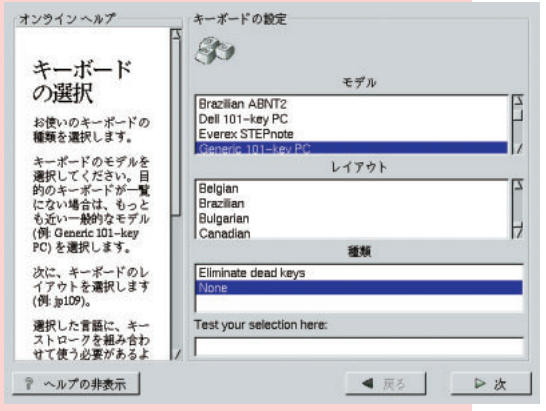
(1)Red Hat 6.1JのCD-ROMをドライブにセット。(2)DOS窓を開き、D: [Enter]と入力。CD-ROMドライブが「D:」以外の場合は、それを入力。(3)cd \dosutils [Enter]と入力。(4) rawrite -f \images\boot.img -d a [Enter]と入力。(5)「Please insert a formatted diskette into drive A: and press --ENTER--:」と表示されたら、フォーマット済みのフロッピーディスクをAドライブに入れ、Enterキーを押し、しばし待つ。



インストーラの起動

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

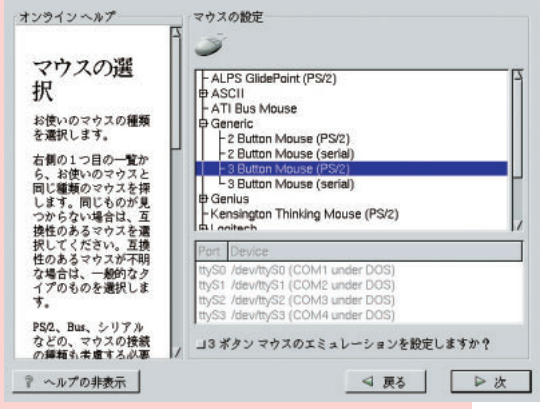
旧バージョンと同じテキストベースでインストールしたい場合、またはグラフィカルインストーラが動作しなかった場合は、text [enter]と入力します。



キーボードの選択

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

しかし、インストーラに不具合があるため、この設定は反映されません。そこでLinuxを起動後、次のように設定する必要があります。まず、/etc/sysconfig/keyboardというファイルにエディタで「KEYTABLE=jp106」と記述します。続いて/etc/X11/XF86ConfigというファイルのSection keyboardでXkbModelに"jp106"」を、XkbLayoutに"jp"」をそれぞれ記述します。これでコンソール、X Window Systemどちらの環境でも、キー配列が正しくなります。



マウスの選択

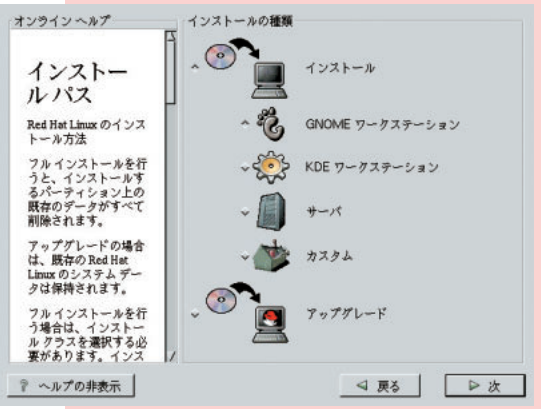
現在使われているマウスは、ほとんどがPS/2マウス、またはシリアルマウスです。これらを使っているなら、「Generic」という項目の下から選びます。2ボタンマウスを選ぶと、自動的に「3ボタンマウスのエミュレーションを設定しますか？」という項目が有効になります。これは左右のボタンを同時に押すことで、中ボタンの代用をさせる機能です。

インストールパス

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

以下、「カスタム」を選択したと想定して、説明を続けます。

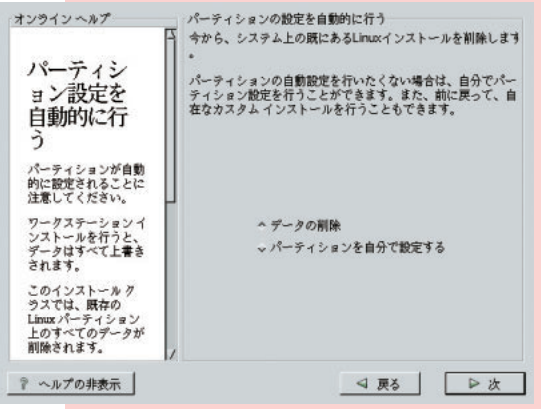
「アップグレード」は、Red Hat Linux 6.0などに上書きアップグレードする際に選択します。



パーティション設定を自動的に行う（前画面で「カスタム」以外を選択時）

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

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

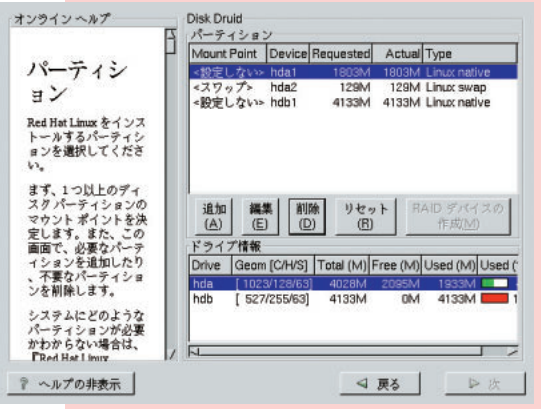


パーティション

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

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

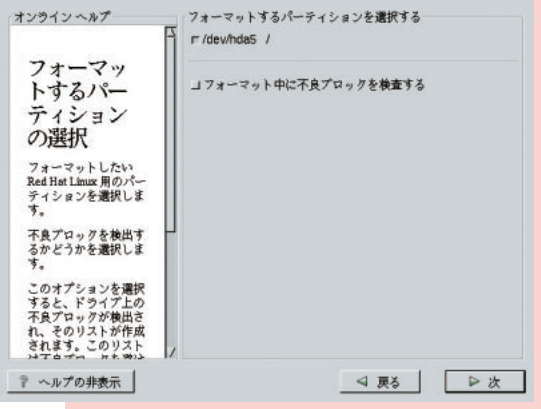
デュアルブート環境を作る際には、パーティション構成をメモしておくことをお勧めします。

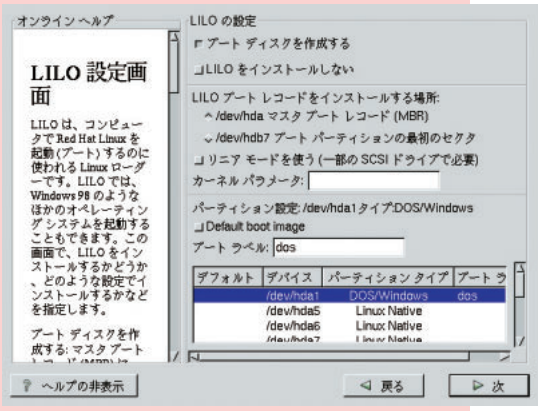


フォーマットするパーティションの選択

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

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



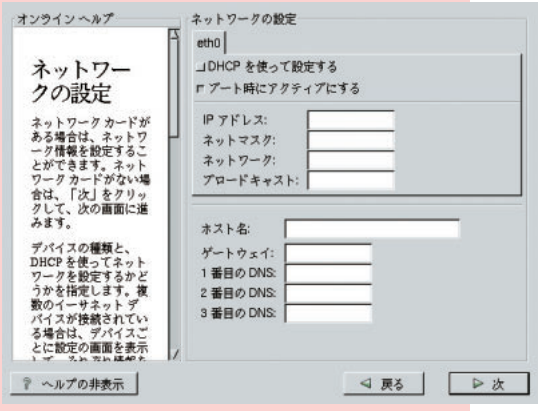


LILO設定画面

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

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

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

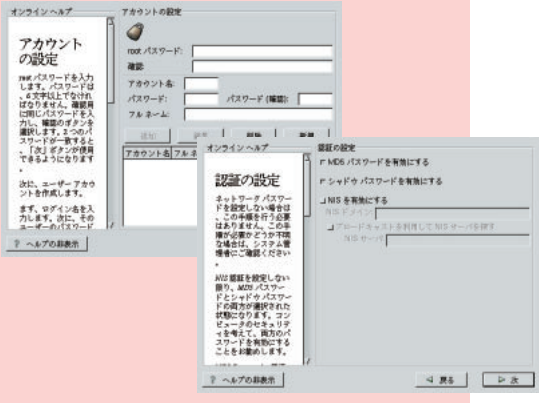


ネットワークの設定

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

タイムゾーンの選択

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



アカウントの設定、認証の設定

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

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

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

パッケージグループの選択

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

12

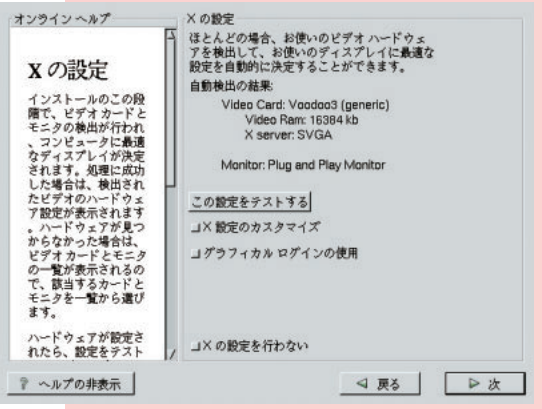
Xの設定

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

XF86Setup起動時に文字化けが発生する場合には、画面左下のボタンを押してXF86Setupを終了し、コマンドラインから「export LANG=C」と入力後、再度XF86Setupを起動してください。

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

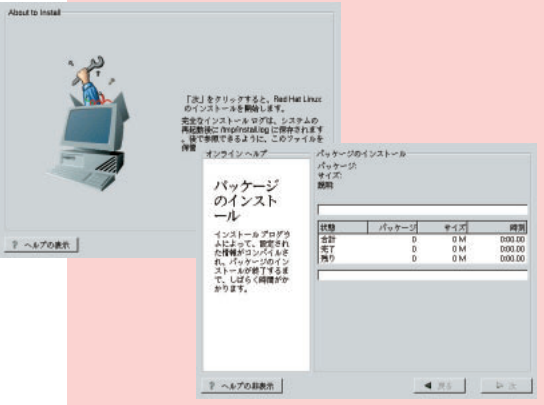
13



インストール開始

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

14



インストール終了

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

15

OpenLinux 2.3 日本語版

10月号で紹介したCaldera OpenLinuxが正式に日本語対応し、OpenLinux 2.3 日本語版として、11月27日に発売される。価格は1万2800円。

Caldera OpenLinuxは、米Caldera Systems社が開発・販売・サポートを手がける、米国で人気のディストリビューションだ。今回発売されるOpenLinux 2.3 日本語版（以下OpenLinux）は、東京にあるインターネットソリューションプロバイダー、ネオナジーが国内販売を行う。

Linuxとしての基本スペック

システムは、カーネル2.2.10、日本語Localeに対応したglibc 2.1.1を採用。デスクトップ環境は、日本語対応版のKDE 1.1.2で構築され、同梱の各種ソフトウェアも日本語化されている。

KDE（The K Desktop Environment）は、UNIX系OSで利用可能な統合デスクトップ環境で、美しいデザインと高い機能を持つ。OpenLinuxでは、日本語に対応するだけでなく、ディスプレイ表示を前提にデザインされたフォント「さいもん」を標準装備し、メニューなどの文字表示に利用している（**画面1**）。

Xサーバは、XFree86 3.3.5+X-TTパッチで、日本語TrueTypeフォントをきれいに表示できる。

OpenLinuxは、グラフィック表示によるわかりやすいインストーラ「Lizard」により、Linuxの初心者でも比較的簡単にインストールできる。実際のインストール手順は、次ページからの解説を参考にしてほしい。

ソフトウェアパッケージ管理システムは、標準になりつつあるRPM（Red Hat Package Manager）だ。

商用ソフトウェア

製品版には、日本語入力ソフトウェアATOK12 SE、Wnn6 Ver.3、サウンドシステムOpen Sound System（OSS）がバンドルされる。さらに、日本語ワードプロセッサdp/NOTE、英語ワードプロセッサWordPerfect 8も含まれる（**画面2**）。

また、製品版にはWindowsなど、ほかのOSとのマルチブート環境を構築するのに便利なパーティショニングツールPartitionMagic Caldera Edition、ブートセレクタBootMagicも同梱されているので、マルチブートを考えている初心者のかたにはパッケージ購入をお勧めする。

サポート

30日間までの電話サポート、あるいは90日間までの電子メールによるサポートが、あわせて5件まで無料で受けられる。Linuxのインストールのほか、インターネット接続、メールクライアント環境など、ソフトウェアの基本的な設定についても標準サポートされる。

付録CD-ROMにFTP版を収録

付録CD-ROM No.2に商用ソフトウェア、サポート権を含まないFTP版OpenLinuxを収録した。今までの日本語ディストリビューションとは一風変わったOpenLinuxを体験してほしい。

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

画面1
GNOMEと並んで人気の高いデスクトップ環境のKDEを日本語化。メニューにディスプレイフォント「さいもん」を採用。KDE付属の予定管理ソフトKOrganizerも日本語化されている。

画面2
製品版に付属の英語版WordPerfect 8で GPL Ver.2を表示。自動的に文法チェックをしてくれるのが嬉しい。日本語を使う場合は、dp/NOTEが利用できる。

OpenLinux 2.3日本語版のインストール
グラフィカルインストーラ　Lizard

CD-ROMブートが可能であれば、OpenLinux 2.3 日本語版（以下OpenLinux）のCD-ROMから起動します。

CD-ROMブートができない場合は、Windowsなどでブートフロッピーを作成します。Windowsが起動したら、フォーマット済みのフロッピーディスクをドライブに入れ、OpenLinuxのCD-ROMをドライブにセットします。AutoRunで起動するインストーラで「製品のインストール」―「インストールディスクの作成」を選択してフロッピーを作ります。また、CD-ROMの\cpl\launch\floppyディレクトリにあるバッチファイルでも同様の手順で作成できます。

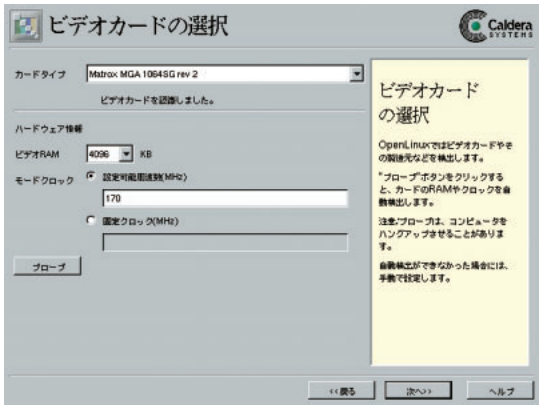
インストーラを起動すると、黒地の画面に「boot:」のプロンプトが表示されますので、通常はEnterキーを押してください。

使用言語、マウス、キーボードの選択

アニメーションのあと、言語の選択画面になります。デフォルトの「Japanese」を選び、「次へ>>」ボタンを押して次に進みます。

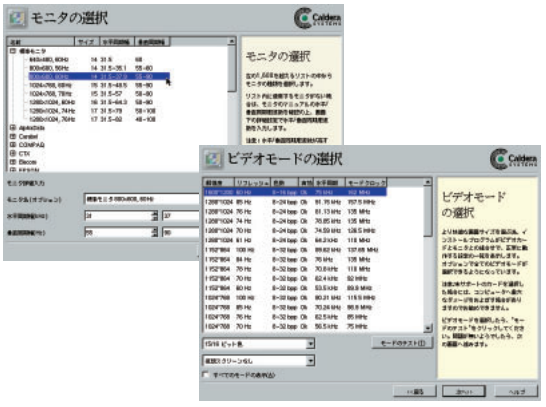
続いてマウスの設定をします。ここでマウスを動かすと自動的に種類を判別しますが、うまくいかないときはキーボードで設定します。2ボタンマウスを使っているなら、「3ボタンのエミュレーション」にチェックするとよいでしょう。これは、左右のボタンを同時に押すことで、真中のボタンと同等の働きをさせる機能です。

次はキーボートの選択です。利用するキーボードに合わせて種類を選択します。日本語106、または109キーボードを利用するなら「日本語106キーボード」を選びます。



ビデオカードの選択

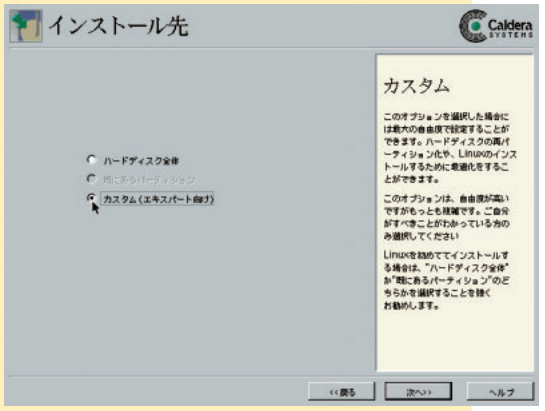
自動的にビデオカードを検出し、「カードタイプ」に認識結果が表示されます。「プローブ」ボタンを押すと、搭載しているビデオRAMの容量やモードクロックを自動設定できます。ただし、プローブに失敗して、コンピュータが動作しなくなることもありますので、そのときは手動で設定します。



モニタ、ビデオモードの選択

利用するモニタを選択します。リストに利用しているモニタがあればそれを選びます。日本で販売されている機種を含め、数多くのモニタデータが用意されていますが、リストの中にない場合は、モニタの取扱説明書で水平同期幅、垂直同期幅を調べて、直接数値を入力しましょう。モニタが追従できないほど高い数値を設定すると、モニタに物理的なダメージを与えてしまうので注意が必要です。

次に、ビデオモードを選択します。利用できるモードの一覧が表示されますので、解像度、リフレッシュレートと、表示色数を選びます。選んだら、「モードのテスト」ボタンを押して、実際にそのモードで表示できるかチェックします。

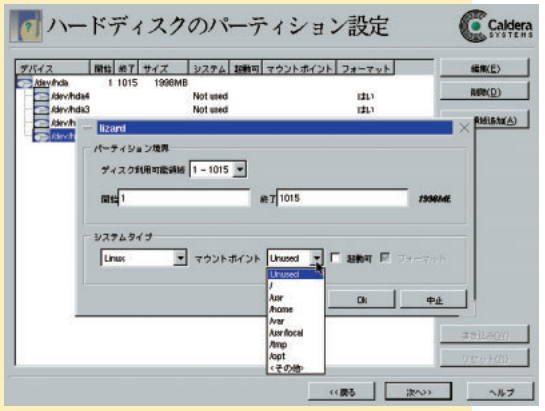


インストール先の選択

OpenLinuxをインストールするハードディスクの使い方を選びます。1台のハードディスクをOpenLinux専用にするなら「ハードディスク全体」を選びます。この場合、選んだハードディスク内のパーティションが切り直され、すべてのデータが消去されますので注意してください。

「既にあるパーティション」は、ほかのLinuxや、PartitionMagicのようなパーティション作成ツールで、あらかじめLinuxパーティションを作ってある場合に選びます。

「カスタム」は、パーティション設定をすべて手動で行う場合に選びます。ここでは、カスタムを選択したものとして解説します。



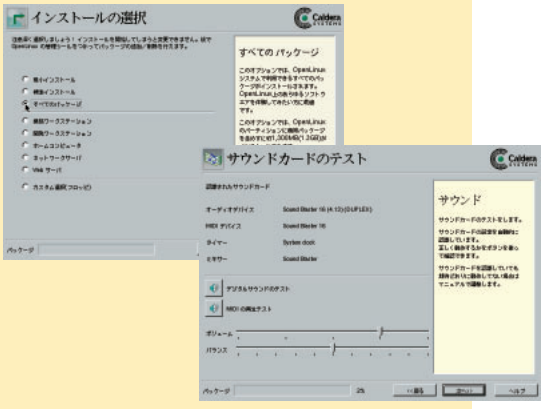
ハードディスクのパーティション設定

ディスク1台につき、4つの領域が表示されます。拡張パーティションがあると、その中の論理領域も表示されます。

設定する領域を選び、「編集」ボタンを押すとダイアログが開くので、各種条件を設定します。OpenLinux用のパーティションは、「システムタイプ」を「Linux」、「マウントポイント」を「/」にして、「起動可」にチェックします。スワップパーティションは、「システムタイプ」を「Swap」にして、搭載メモリ量の1～2倍ほどの容量を割り当てましょう。

設定がすんだら、「書き込み」ボタンを押してパーティション情報を書き込みます。必要な領域を消さないよう十分に注意してください。

次に、「パーティション情報」画面で領域をフォーマットします。

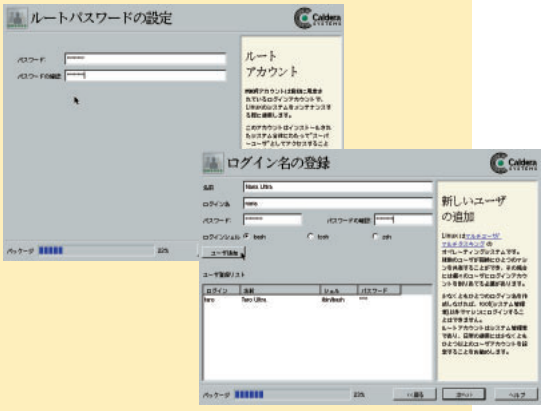


インストールの選択、サウンドカードのテスト

用途に応じてインストールするパッケージを選びます。画面の右側に各インストールオプションの説明が表示されますので、これを参考にするとよいでしょう。この説明画面は、ハイパーリンクになっていますので、下線付きのリンクがある用語をクリックすると、その説明を読むことができます。

ここでは「すべてのパッケージ」を選んだものとして解説します。「次へ>>」ボタンを押すとファイルのコピーが始まります。これ以降、ファイルのコピーはバックグラウンドで行われます。マルチタスクの特性をうまく活かしていますね。

サウンドカードがある場合は、サウンドカードのテスト画面になります。ここでボリュームとバランスを調節できます。



ルートパスワードの設定、ログイン名の登録

システム管理者であるルートユーザーのパスワードを設定します。「パスワード」欄に入力後、「パスワードの確認」欄に同じパスワードを入力します。

次は一般ユーザーの登録です。最低でも1つのユーザーアカウントを作る必要があります。名前、ログイン名、パスワードのほか、ログインしたときにデフォルトで起動されるシェルの種類も設定できます。ユーザー情報を記入したら「ユーザ追加」ボタンを押して、アカウントを登録します。

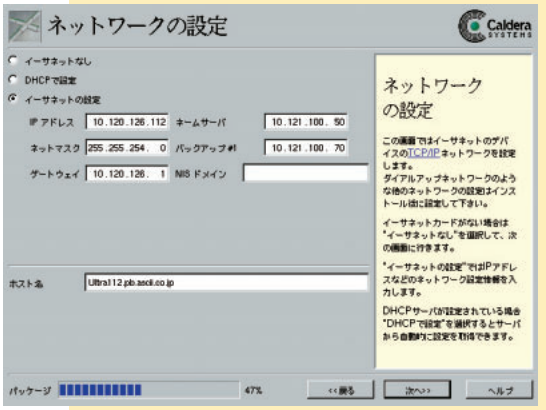
「ユーザ追加」ボタンを押したあと、名前の入力から同じ手順を繰り返すことで、いくつでもアカウントを登録することができます。

ネットワークの設定

イーサネットで利用するTCP/IPの設定を行います。ダイヤルアップなど、イーサネットカードを使わないネットワークの設定はここでは行えないので、インストールしてから個別に設定します。

DHCPサーバが利用できる場合は、「DHCPで設定」を選んで「ホスト名」を設定するだけです。

DHCPサーバを利用しないなら、「イーサネットの設定」を選び、各項目に必要な情報を入力します。予備のネームサーバがある場合は、「バックアップ#1」欄で指定します。



Linuxの起動

OpenLinuxを起動するためのプログラムLILO（LInux LOader）を書き込む場所と、LILOから起動するほかのOSを登録します。Windows 9Xとデュアルブートする場合は「マスターブートレコード（MBR）」を選び、Windows 9Xがインストールされているパーティションにチェックを入れます。

PartitionMagicなどのブートセレクタを利用している場合は、「インストール先(T)」を選びます。

複数OSでのマルチブートには、状況によってさまざまな問題が発生しがちです。114ページからの「特集2　マルチブート完全制覇！」もあわせてご覧ください。

タイムゾーンの選択

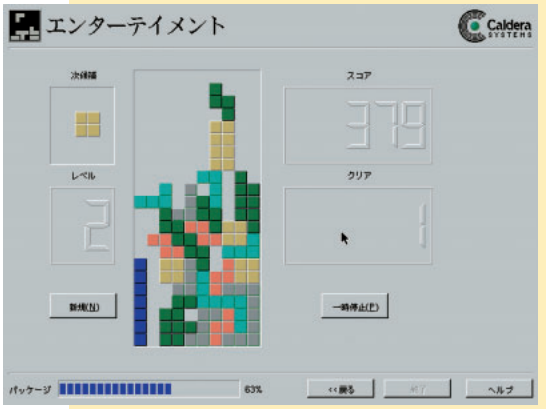
そのままで「Asia/Tokyo」が選択されていますので、日本国内で利用する限り変更する必要はありません。変更する場合は、世界地図上をクリックするか、リストから地域を選びます。

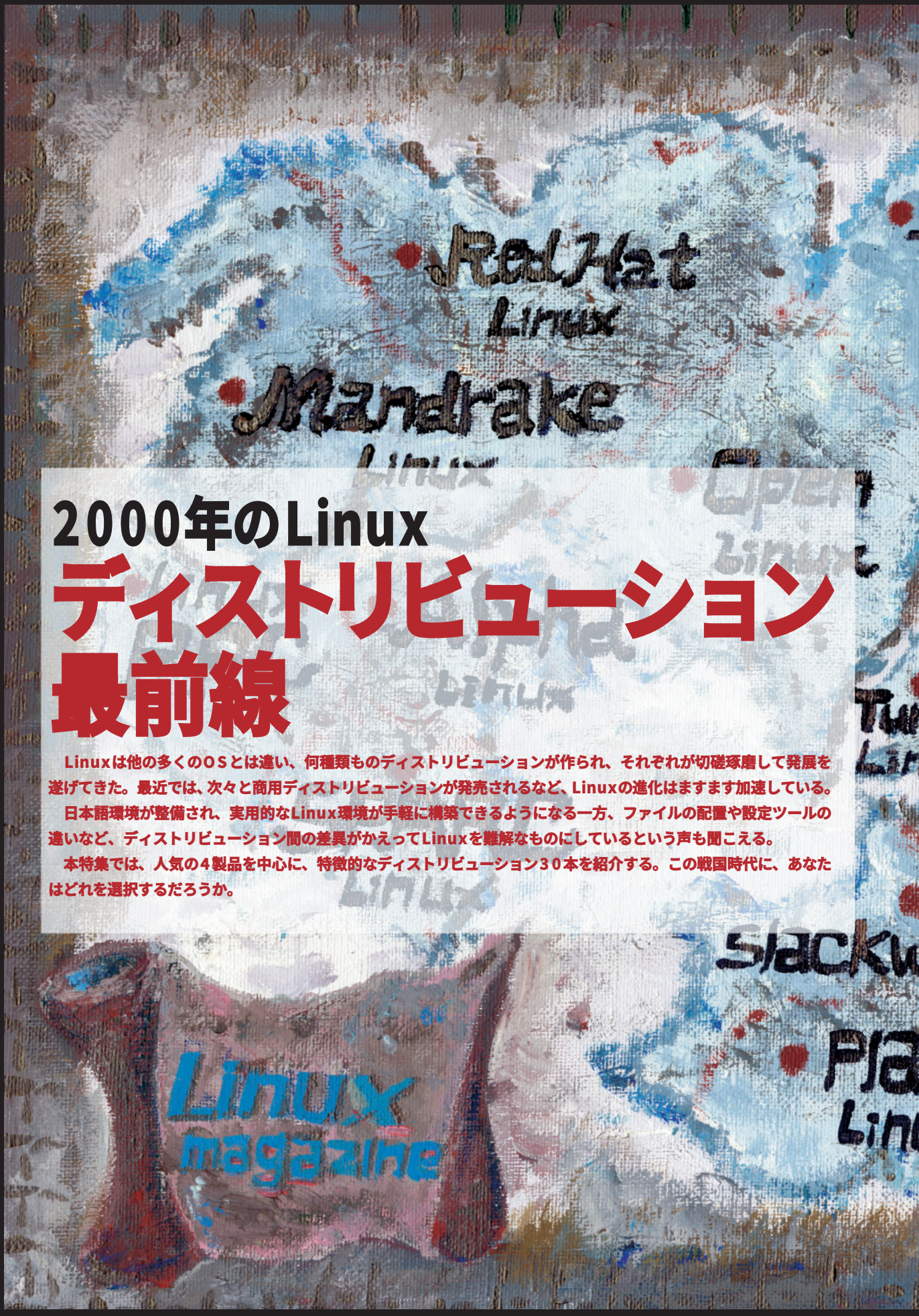
ラジオボタンは「ハードウエアクロックをローカルタイム（I）に設定」にチェックしたままにしておきます。

インストール終了までゲームで遊ぶ

これで設定は終わりです。ファイルのコピーと設定が終わるまで待つだけです。ただ待つのは退屈でしょうということなのか、みなさんよくご存じの落下型ゲームで遊ぶことができます。カーソルキーとスペースバーで操作します。

インストールがすべて終了すると、「終了」ボタンの文字がグレーから黒にかわり、ボタンを押せるようになります。ボタンを押すと、OpenLinuxが起動します。





2000年のLinux ディストリビューション最前線

Linuxは他の多くのOSとは違い、何種類ものディストリビューションが作られ、それぞれが切磋琢磨して発展を遂げてきた。最近では、次々と商用ディストリビューションが発売されるなど、Linuxの進化はますます加速している。

日本語環境が整備され、実用的なLinux環境が手軽に構築できるようになる一方、ファイルの配置や設定ツールの違いなど、ディストリビューション間の差異がかえってLinuxを難解なものにしているという声も聞こえる。

本特集では、人気の4製品を中心に、特徴的なディストリビューション30本を紹介する。この戦国時代に、あなたはどれを選択するだろうか。

illustration : Aki

Liunxのディストリビューションとは？

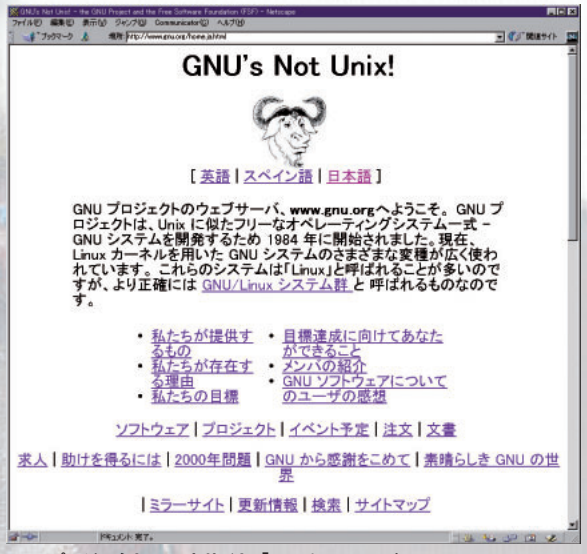
Linuxは、マイクロソフトのように、すべてを1社で開発しているわけではなく、インターネットを利用して多くのエンジニアたちが意見を交換しながら開発している。そして、Linuxの名前を冠した複数のディストリビューション（＝配布パッケージ）が存在し、新しいディストリビューションが今も増え続けている。さまざまなパッケージが販売されているため、Linuxに最初に触れたときには「どれが本当のLinux？　なにが違うの？」と思うことだろう。実はどれも本物なのだが、そうしたLinuxの「ディストリビューション」とは何なのか、まずは少し整理しておこう。

Linuxは、91年にフィンランド、ヘルシンキ大学の学生だったLinus B. Torvalds氏が開発を始めたOSである。当時、学生にOSの勉強をさせるための教材としてソースが公開されていた16ビットCPUの8086用MINIXの機能に不満があったため、32ビットCPUである80386を生かしてマルチタスク、仮想記憶などを実装するために新たに開発した。本来、「Linux」とはカーネル（OSの基本部分）のみを指している。その「Linux」は現在もTorvalds氏が中心になって開発、改良が進められている。

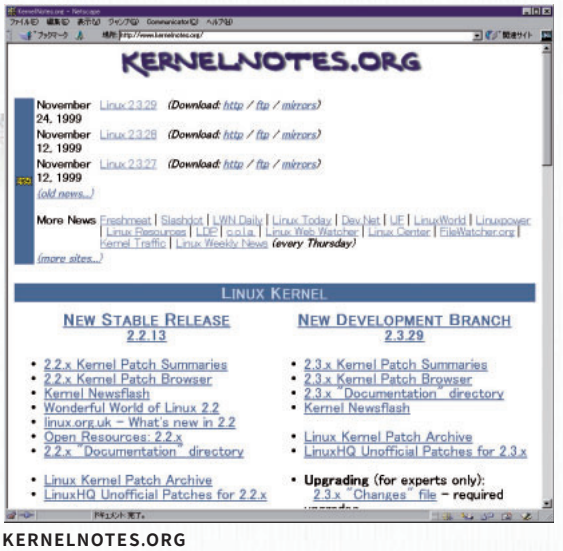
カーネルだけでは使えないので、いろいろなコマンドやアプリケーションプログラムが、Linux用に開発されたり、移植されたりしている。Linuxで使われているコマンドの多くは、UNIXのコマンドと同等の機能を持つプログラムであり、GNUプロジェクトによって開発されたものを使用している。AT&Tが開発したUNIXとそれに含まれるコマンドは商用OSとしてライセンスされているため、GNUプロジェクトはライセンスフリーなUNIXシステムを目指して、UNIXのソースコードを使わずに、新たにプログラムを開発している。

LinuxカーネルとGNUプロジェクトのプログラムは、GPL（The GNU General Public License）というライセンスによってソースコードを公開している。GPLではソフトウェアの使用条件として、自由（ソースコードを含めた再配布など）を妨げないことを使用者に要求している。これがLinuxはフリーなソフトウェアと言われる所以である

しかし、必要なプログラムのソースコードを個別に入手して、コンパイルすることでシステムを構築するには、膨大な時間と知識が必要になる。そもそも、最初にソースコードをコンパイルする環境がないといけない。この問題を解決すべく、誕生したのがディストリビューションだ。コンパイル済みのバイナリパッケージとインストーラなどをセットにして、ユーザーが手軽にLinux環境を手に入れられるように



GNUプロジェクトのWebサイト「GNU's Not Unix!」
http://www.gnu.org/home.ja.html



KERNELNOTES.ORG
http://www.kernelnotes.org/
Linuxカーネル最新版の情報は、このWebサイトで調べることができる。

するためにディストリビューションは作られた。

当初はフリーソフトウェアだけで構成されていたディストリビューションだが、現在では、商用ソフトウェアやサポートを含むパッケージ製品として販売されているものも多い。

そして、ディストリビューションとしてまとめられたもののうち商用ソフトウェアを除いた部分は、GPLなどのライセンスに準じて、それぞれのFTPサイトからダウンロードできるように公開されている。雑誌や書籍の付録CD-ROMにディストリビューションが収められていることも多いが、フリーソフトウェアとして公開されているFTP版をCD-ROMに収録しているわけで、製品版とは違い商用ソフトウェアやサポートが含まれていないのはそのためである。

Linuxカーネルには安定版と開発版の2種類ある

Linuxの開発は日々続けられていて、いろいろな機能を取り込んでいる。しかし、開発中のカーネルにはバグも多くあるだろうし、取り入れた新機能によって動作が不安定になることもあり得る。そこで、開発がある程度一段落した時点で新機能の追加を中止して、動作確認とバグフィクスを行った安定版カーネルを提供している。

開発版カーネルと安定版カーネルは、バージョン番号の2番目の数字で区別していて、奇数が開発版、偶数が安定版となっている。現在の安定版カーネルは2.2系で執筆時点での最新バージョンは2.2.13である。

開発版のほうは2.3系で最新バージョンは2.3.29である。2.3系の開発はいったんフリーズされ、安定版の2.4カーネルがもうすぐリリースされる予定だ。

ディストリビューションが採用しているのは、もちろん安定版のほうである。安定バージョンとはいえ、セキュリティホールや新たなバグが見つかると改訂されるので、最新バージョンのリリースには注意しておきたい。

Linuxを使いやすくしたパッケージ管理システム

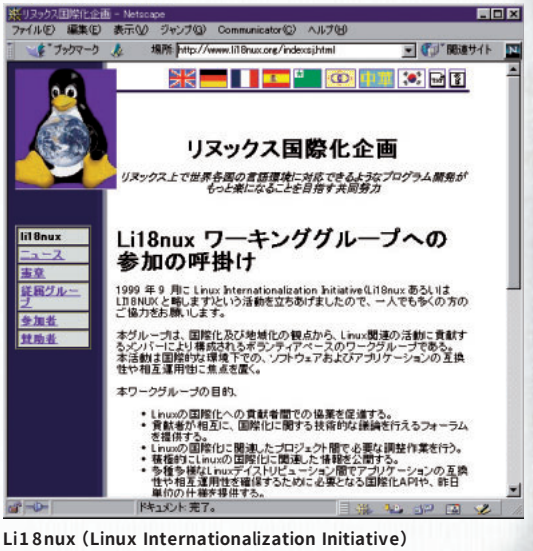
ソフトウェアの管理をしやすくするために、多くのディストリビューションではパッケージ管理システムを採用している。

1つは、Red Hat Softwareが開発したRPM（Red Hat Package Manager）形式というパッケージ管理システムだ。RPMを採用しているディストリビューションはRed Hat系と呼ぶこともある。Red Hat Linuxはもちろんのこと、TurboLinuxやLASER5 Linux、Vine Linuxなど、日本でメジャーなディストリビューションが採用している。

もうひとつは、Debianプロジェクトが開発したdeb形式のパッケージ管理システムだ。Debian GNU/LinuxやCorel LINUXなどが採用している。

本来、Linuxで新しいソフトウェアをインストールしたり、バージョンアップしたりするには、ソースプログラムを入手してコンパイルすればよいのだが、それにはディスクスペースと時間が必要だ。CPUとOSが共通ならばバイナリで提供した方が効率的である。そういう理由からパッケージ管理システムが開発された。そして、格納ディレクトリやパッケージファイルに必要なライブラリ、ほかのプログラムとの依存関係を記述しておくことで、ユーザーの便宜が図られるという利点もあ

XFree86 Project, Inc
http://www.xfree86.org/



Li18nux（Linux Internationalization Initiative）
http:/www.li18nux.org/indexsj.html
Linuxの国際化を進めるワーキンググループが1999年9月に発足した。

る。また、ソースプログラムもパッケージ形式で配布されている。

商用ソフトウェアでソースプログラムを公開していない場合、バイナリをパッケージ形式で配布することによって特別にインストーラを用意しなくても済むことも、Linuxを普及させる点で有利に働いているだろう。

グラフィックス環境

Linuxのウィンドウ環境は、X Window Systemを使用していて、どのディストリビューションでも標準的に使われているのは、XFree86というフリーソフトウェアである。X Window Systemはクライアント／サーバモデルとなっていて、そのXサーバを使用するハードウェアに合わせてカスタマイズすることによって、キーボードやマウスの入力、画面の表示を行うといったハードウェアの違いを吸収している。Linux用に新しいグラフィックスカードを購入するときなど、使用しているディストリビューションのXFree86が未対応でも、最新バージョンのXFree86では動作する可能性があるので調べてみるといい。

1年ほど前までは、ウィンドウマネージャとしてWindow MakerやAfterStep、fvwmなどを使用していた。これらのウィンドウマネージャは、見た目こそWindowsやMac OSのようだったが、ウィンドウ上で動作するそれぞれのアプリケーションはまったく独立して動作していた。

しかし、最近のディストリビューションでは統合デスクトップ環境であるGNOME、KDEが標準になってきた。ファイルマネージャを使ってドラッグ＆ドロップでアプリケーションの連携が行えるようになり、それぞれが統一されたインターフェイスでルック＆フィールを実現している。また、システム設定の変更は、エディタで直接ファイルを編集するのがふつうだったが、デスクトップ環境ではGUIのツールによって各種の設定を変更できるようになってきた。

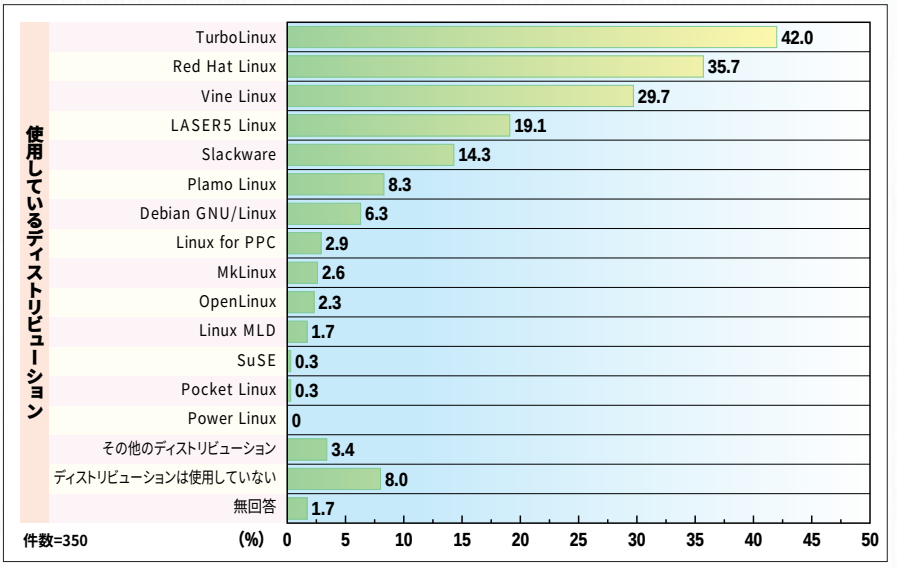


図1　使用しているディストリビューション　（有効回答350件）
Linux magazine 11月号の読者アンケートの集計結果より

Cライブラリの違いと日本語環境

Linuxで使われている基本Cライブラリは、libc5（＝glibc1）、glibc2.0、glibc2.1とバージョンアップしてきた。どのバージョンも国際化が不十分であり、日本語に対応するため、X_LOCALEを使ってX Window System上で対応したり、ライブラリにパッチを当てたりしたため混乱を招いている。そのため、Linux International Initiative（略称: Li18nux）というワーキンググループで国際化を進めている。

現在リリースされている日本語対応のディストリビューションでは、glibc2.0＋wcsmbsライブラリまたは、glibc2.1を使用している。

本誌読者が使用しているディストリビューションは？

本誌では、毎号読者アンケートで「使用しているディストリビューション」を調査している。11月号（10月8日発売）のアンケートの集計結果は**図1**のグラフのようになった（有効回答350件）。複数回答なので合計は100%を超えている。

TurboLinuxが42%とトップで、それをRed Hat Linuxが35.7%、Vine Linuxが29.7%と追いかけている形だ。LASER5 Linuxは発売から約1カ月と、ほとんど時間が経っていないが19.1%とかなりがんばっているように見受けられる。もっとも、その号の付録CD-ROMに、LASER5 Linux 6.0（FREE版）とTurboLinux 日本語版 4.2（ノンサポート版）を収録しているため、早速使用してみたという人も多かったのだろうと推測できる。

最近あまり取り上げられない老舗のSlackwareが、14.3%となかなかの数字なのは、サーバ用途などで安定して利用されている証拠なのかもしれない。

Section 1

人気の日本語版ディストリビューション4製品

TurboLinux 日本語版 4.0／Red Hat Linux 6.1 日本語版／LASER5 Linux 6.0／Vine Linux 1.1 CR

初期のディストリビューションは、海外で作られたものしかなく、そのまま日本語を使えるものはなかった。しかし、Linuxで日本語を使いたいという要望は強く、日本の有志が、UNIXの日本語環境ソフトウェアをLinuxに移植していった。これらの日本語化プログラムを集めたのがJE（Japanese Extensions）というパッケージだった。

JEは、Slackwareというディストリビューションに追加して使うようになっていたこともあり、数年前までは日本でのデファクトスタンダードは、このSlackwareだったのだ。

その後、これらの日本語環境ソフトウェアや、新たに日本語対応させたソフトウェアを集め、はじめから日本語が使えるディストリビューションが登場した。

現在では、このような日本語版ディストリビューションが主流となり、フリーソフトウェアだけでなく、商用ソフトウェアやサポート権を含むパッケージ製品として販売されているものも多い。

その中でも人気の高いのが、次にあげる4つの日本語版ディストリビューションだ。

- **TurboLinux 日本語版 4.0**
- **Red Hat Linux 6.1 日本語版**
- **LASER5 Linux 6.0**
- **Vine Linux 1.1 CR**

Section 1では、この4製品それぞれの概要、機能と特徴を紹介する。

この特集では、製品版を紹介するが、商用ソフトウェア、サポート権を含まないフリー版が、それぞれFTPや雑誌付録CD-ROMなどで配布されている。

興味を持ったディストリビューションがあったらトライしてほしい。

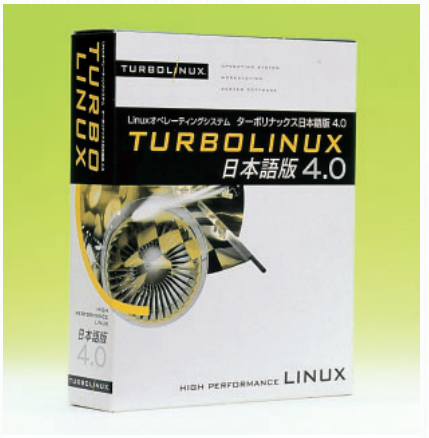
TurboLinux PRO 日本語版 4.2

Red Hat Linux 6.1 日本語版

LASER5 Linux 6.0

Vine Linux 1.1 CR

TurboLinux 日本語版 4.0
TurboLinux PRO 日本語版 4.2



TurboLinux 日本語版 4.0
価格　9800円

TurboLinux 日本語版 4.0（以下TurboLinux）は、ターボリナックスジャパンが1999年の7月9日に発売したディストリビューションだ。日本語ディストリビューションとしては初めてカーネル 2.2とGNOME、KDEを採用した。発売からやや時間がたつが、後発のディストリビューションに比べても遜色ない機能を持つ。

現代的なLinux環境を提供

TurboLinuxが採用しているLinuxカーネルのバージョンは2.2.9。Pentium以上のCPUに最適化し、セキュリティフィックスを行うなど、チューニングが施されている。インストール時にSMPカーネルを選ぶことも可能だ。

カーネルと並び、システムの根幹をなす標準Cライブラリはglibc 2.0＋wcsmbsだ。 Red Hat Linux 5.2や、Vine Linuxもglibc 2.0を使っているため、フリーソフトRPMパッケージなどは、これらのディストリビューションと同じものがほとんど使える。

XサーバのXFree86は3.3.3.1で、X-TTパッチをあててあるため、日本語TrueTypeフォントの表示もできる。

デスクトップ環境には、最近話題のGNOMEとKDEの日本語対応版を選ぶことができる（**画面1**）。これらの環境は、見栄えもよく、さまざまな設定ツールを装備するなど、Windowsにも匹敵する操作性を提供する。

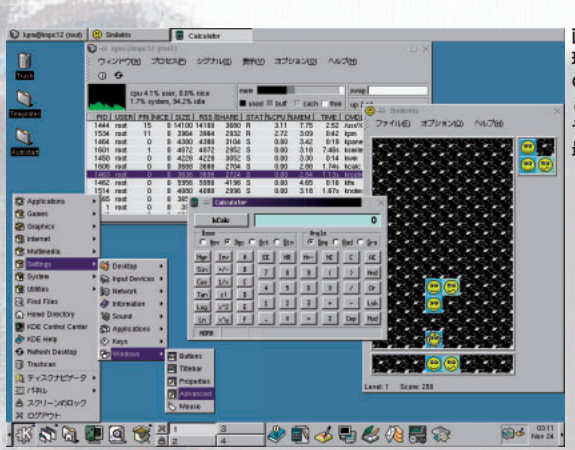
これらの統合デスクトップ環境は、CPUへの負荷も大きく、Pentium 200MHz以下のマシンではマウスのレスポンスが悪くなり、かえって使いにくくなる。しかし、心配は無用だ。After StepやWindow Makerなど、非力なマシンでも軽快に動作するウィンドウマネージャも健在だ（**画面2**）。日本ではまだなじみのないICE Window Managerを試すのもいいだろう（**画面3**）。

充実した独自の設定ツール

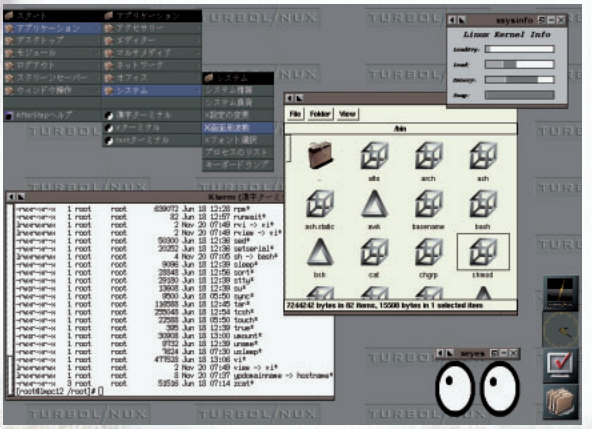
Linux初級者にとって、システムの設定作業は大きな負担といえる。Red Hat Linux系のディストリビューションのLinuxconfや、GNOMEやKDEのControl centerなど、各種設定を一括管理するためのツールも増えつつある。TurboLinuxにはLinuxconfが付属しないかわりに、TurboToolsという設定ツール群が付属する。たとえば、Xサーバの設定をするturboxcfgのように、これらのツールは“turbo”で始まる名前が付けられており、テキスト画面でのGUIで操作する。RPMパッケージ管理、ネットワークの設定、ウィンドウマネージャの変更、ディスク管理など、日常的な管理作業の手間をかなり軽くしてくれる。

ATOK12 SEなど、充実した商用ソフトウェアをバンドル

TurboLinuxには、表に示す商用ソフトウェアがバンドルされている。

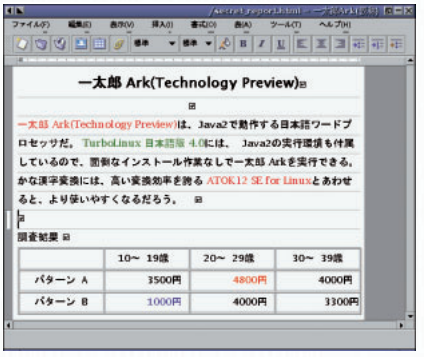


画面1　デスクトップ環境には、GNOMEのほかKDEを選ぶこともできる。アップデートRPMパックで最新版にしておこう。



画面2　TurboLinux日本語版 3.0で標準だったAfterStep。動作が軽いので、ノートPCなどで使うとよいだろう。

画面3　TurboLinux PROでは、ウィンドウマネージャ切り替えツールのturbowmcfgでICE Window Managerも選択できる（TurboLinuxにもプログラムは入っている）。日本語に未対応なので今後に期待したい。



画面4　一太郎Arkは、HTML技術をベースにしたワードプロセッサだ。Javaのオーバーヘッドも、最近の高速なマシンでは気にならない。

TurboLinux PRO 日本語版 4.2
価格　2万9800円

UNIX系OSでは、CannaやWnn4などのかな漢字変換ソフトウェアがよく使われているが、これらに不慣れなWindowsユーザーにとっては、ATOK12 SE for Linuxが強力な味方となるだろう。

また、商用日本語入力ソフトウェアとして、Wnn6 Ver.3も用意されているので、好みに合わせて使ってほしい。

日本語フォントは、リョービのTrueTypeフォントが付属する。書体は、明朝、ゴシック、丸ゴシック、極太ゴシック、毛筆の5種類で、必要にして十分なセレクションだ。

このほか、サウンドカード用ドライバのOpen Sound System（OSS）も付属しているので、サウンドカードを利用するなら、忘れずにインストールしよう。

製品版ではないが、Javaで作られたワードプロセッサ、一太郎Arkの Technology Preview版も同梱されている（**画面4**）。通常のワードプロセッサとは違い、HTMLベースのものなので、できることはある程度限られるが、十分実用になる。

TurboLinux PROはオフィススイートで強化

1999年9月10日に発売された、TurboLinuxの姉妹製品がTurboLinux PRO 日本語版 4.2（以下TurboLinux PRO）だ。これは、TurboLinuxに商用オフィススイートApplixware 日本語版 4.4.2（以下Applixware）をバンドルしたものだ。

Applixwareは、ワードプロセッサ、表計算、グラフィックス、プレゼンテーション、データベースなどを統合したアプリケーションソフトウェアだ。

機能の詳細は、1999年11月号、12月号で紹介しているのでお持ちの方はそちらをごらんいただきたい。

TurboLinux PROのシステムまわりは、基本的にTurboLinuxと同じだが、XFree86、GNOMEをはじめ、たくさんのモジュールがアップデートされている。

日本語かな漢字変換	ATOK12 SE for Linux Wnn6 Ver.3
日本語ワードプロセッサ	一太郎 Ark Technology Preview
商用TrueTypeフォント	リョービ　日本語TrueTypeフォント5書体
マルチブート	System Commander Lite 英語版
ドライバ	Open Sound System
オフィススイート	Applixware 日本語版 4.4.2（TurboLinux PROのみ）

TurboLinux、TurboLinux PROの主なバンドルソフト

サポート体制

TurboLinux、TurboLinux PROとも、対応ハードウェアの認識、設定からインストールまで、Web、E-Mailによるサポートが受けられる。期間はユーザー登録後90日間、件数は3件までとなる。

ターボリナックス ジャパンは、TurboLinux、TurboLinux PROのシステムを最新版にするアップデートRPMパックをFTPサイトで公開している。また、登録ユーザーには、実費にてCD-ROMを配布している。まずは、Webサイトをチェックしてほしい。

なお、サーバ向け「TurboLinux Server 日本語版 6.0」（2万9800円）、「TurboLinux Server 日本語版 6.0 SOHO Edition」（1万9800円）が、12月15日に発売される予定だ。

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

Red Hat Linux 6.1 日本語版

1999年9月に米Red Hat Softwareの100%出資により、レッドハット日本法人が設立された。Red Hat Linux 6.1日本語版（以下Red Hat 6.1J）は、そのレッドハットが最初に出した日本語版のディストリビューションであり、11月12日に発売された。

Red Hat 6.1Jは、Red Hat Softwareから10月4日にリリースされたRed Hat Linux 6.1（英語版）を元に、インターナショナル・ローカライゼーションを行っている。インストーラなどのメッセージを日本語化しているが、基本的な部分はオリジナルの英語版とまったく変わらない環境を提供している。

Red Hat 6.1J製品版は、バイナリCD、ソースCD、コマーシャルCDの3枚のCD-ROMとブートフロッピーディスク1枚、マニュアル1冊で構成されていて、価格は1万2800円。製品版には、インストールとX Window Systemが利用できるまでの30日間の電話サポート、90日間のWeb（メール）サポート、登録ユーザーに対して高速な回線に接続したFTPサーバを提供する180日間の無料FTPサイト優先使用権といったサポートが含まれる。

公認 Red Hat Linux 6.1日本語版
1万2800円

グラフィカル・インストーラを採用

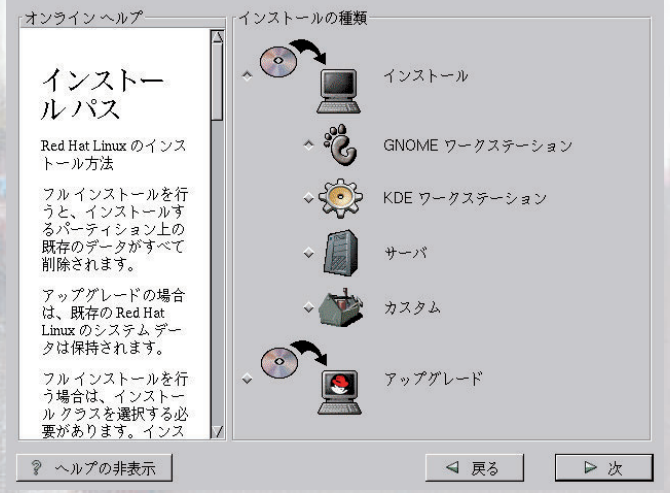
インストーラは、このバージョン6.1からグラフィックス表示となった。画面左側に常時オンラインヘルプが表示され、マウスで操作できるところなどWindows 95／98のインストーラのようにわかりやすくなっている。このあたり、テキストベースのインストーラでは説明がわかりにくかった点が改善された（**画面1**）。

システム周りは英語版と同じバージョンであり、カーネルに2.2.12、GNU Cライブラリはglibc2.1.2、フリーのX Window Systemである XFree86は最新のバージョン3.3.5を採用している。そして、デスクトップ環境には、日本語化されたGNOMEとKDEを選ぶことができる。

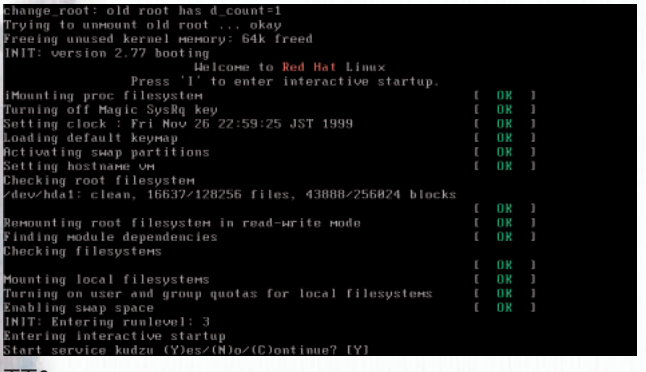
ブートプロセスで自動的に組み込まれる各種ハードウェアドライバは、ときには間違って認識されることがあるが、そういう場合にそのサービスを起動するかしないかを、インタラクティブに設定できるモードを選べるようになったのは便利な点だ（**画面2**）。

商用ソフトウェアは日本語対応ソフトだけ

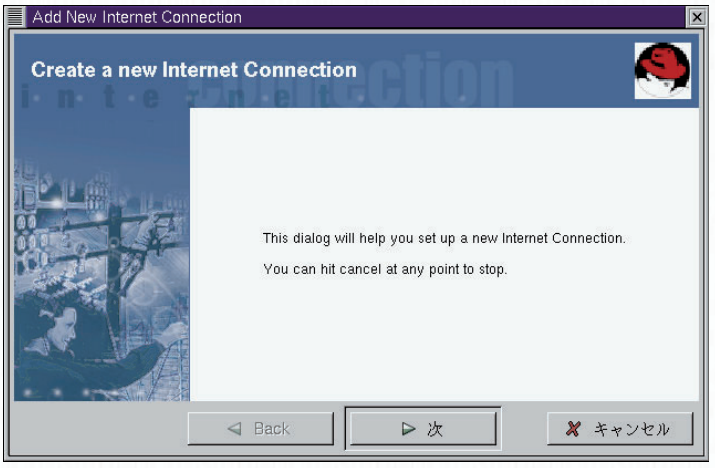
製品にバンドルされている商用ソフトウェアは、あまり多くない。日本語入力ソフトとして「ATOK12 SE」と



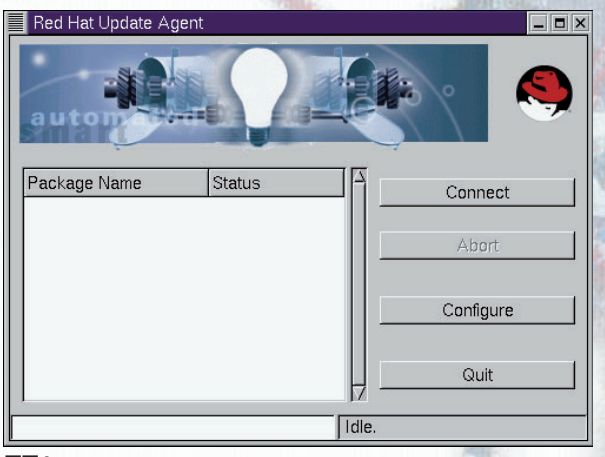
画面1
日本語のメッセージ表示でわかりやすくなったグラフィカルインストーラを採用。この画面ではインストールするパッケージを、マウスでクリックして選ぶことができる。



画面2
起動時のコンソール画面で、「Press 'I' to enter interactive startup」と出力されたところでIを入力すると、以降のサービスを組み込むか否か対話形式で決めることができる。



画面3
インターネットへの接続もウィザードによって設定が簡単に行える。メインメニューから［インターネット］―［Dialup Configuration Tool］と選択する。



画面4
新機能のアップデートエージェントは、Red HatのFTPサイトへ接続して更新されたソフトウェアのリストを表示、ダウンロード、インストールすることが可能。最新パッケージへのアップデートが便利になった。

「Wnn6 Ver.3」、日本語ワードプロセッサの「dp/NOTE」と「一太郎Ark Technology Preview」、そして、DYNALABの商用TrueTypeフォント5書体である。

これらはすべてコマーシャルCDに入っていて、ユーザーが別途インストールする必要がある。DYNALABのフォントはインストール方法や使いかたがマニュアルに記述されていないためセットアップが大変である。この点に関しては早急な改善を期待したい。

PPP接続もGUIのウィザードで簡単に

デスクトップ環境も整ってきていて、GNOMEやKDEのメニューからLinuxconfや各種GUIの設定ツールを使うことで、ほとんどコンソール画面を使う必要がないくらいだ。

ウィザード形式でインターネットに接続するための設定ツールも用意されている（**画面3**）。画面に従ってプロバイダ名、電話番号などを入力していくだけで、セットアップが完了する。

アップデートエージェントで最新パッケージに

Red Hat 6.1から加わった新機能としてアップデートエージェントがある（**画面4**）。インターネット経由でRed Hat社のサイトと接続し、更新されたソフトウェアを調べて表示してくれる。そして、必要なパッケージを選ぶと、ダウンロード、インストールまで自動的に行われる。いままでFTPサイトを見にいって、ユーザーがバージョンの違いを調べていたことを考えると便利なツールである。製品を購入、登録したユーザーだけしか利用できない点は非常に残念だ。

英語版リリースから約1カ月遅れと、短期間で発売になったRed Hat 6.1Jであるが、日本語ローカライズの検証が甘く、いろいろ不具合が残っているようだ。たとえば、インストーラで日本語106キーを選んだにもかかわらず、インストール後のキーボードの設定がUS 101キー配列になってしまうことや、日本語（＝マルチバイト）が使えないコマンドが一部入っていることなどである。

デスクトップ用途で使用するユーザーのために、日本語関係のそういった点を改善したアップデートパッケージを早く整備することが望まれる。しかし、サーバ用途や英語版での動作保証しかないソフトウェアを使用する場合などには、今回追加された機能やサポートを日本法人から受けられることなど、日本語版ならではのメリットは大きい。

なお、サーバ用途向けに「日本語redhat Linux6.0 Server Edition」（4万9800円）もあるが、こちらは五橋研究所（LASER5）から発売されている。

日本語かな漢字変換	ATOK12 SE for Linux Wnn6 Ver.3
日本語ワードプロセッサ	一太郎Ark Technology Preview dp/NOTE　2.01
商用TrueTypeフォント	DynaFont 5書体 DF華康明朝体W3、DF華康ゴシック体W5、DF特太ゴシック体、DF行書体、DFまるもじ体W5

Red Hat Linux 6.1日本語版にバンドルされている商用ソフトウェア

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

LASER5 Linux 6.0
LASER5 Linux 6.0 日本語入力キット

LASER5 Linux 6.0は、Red Hat Linux 6.0（英語版）をベースに日本語化されたディストリビューションである。発売元のレーザーファイブは、Red Hat Linuxの日本語版を開発してきた五橋研究所（Laser5）の子会社で、'99年8月の米Red Hat Software社とのパートナー契約終了に伴い、このLASER5 Linux 6.0を「Red Hat Linux 6.0（英語版）と99%互換のオリジナルディストリビューション」として、'99年9月より発売されている。

2種類の製品形態

LASER5 Linux 6.0には、次の2種類の製品が用意されている。

■LASER5 Linux 6.0

「LASER5 Linux 6.0」（以下、製品版）は、Red Hat Linux 6.0（英語版）をベースに日本語対応とした製品である。90日間3件までのインストールサポート（電話、FAX、電子メール、Web）があり、動作不能の場合は購入日から30日以内なら返品も可能である。さらに、多くのソフトウェアがバンドルされている。

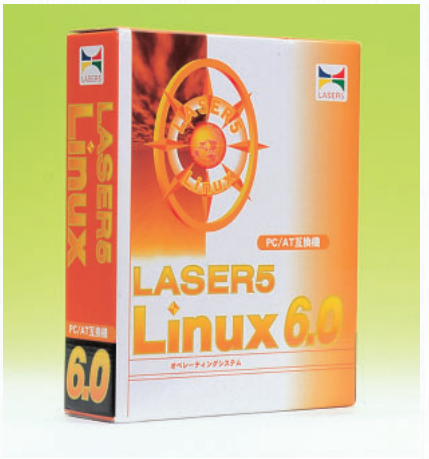
■LASER5 Linux 6.0日本語入力キット

もう1つの製品、「LASER5 Linux 6.0日本語入力キット」（以下、廉価版）は、製品版からサポートと付属ソフトウェアを省いた製品である。

システム構成

システム構成で目を引くのが、Cライブラリとしてglibc2.1.1を採用している点だろう。glibc2.1は、ロケール情報を定義するlocaledefコマンドがマルチバイトに未対応であるため、採用を見送っている日本語ディストリビューションは多い。この問題には独自の対応を行っているようで、試用した限りでは特に問題を感じなかった。

カーネルのバージョンは2.2.5を採用しており、2.2系列の特徴である、SMP（対称型マルチプロセッシング）やソフトウェアRAIDなどに対応している。また、XFree86はバージョン3.3.3.1が搭載されている。



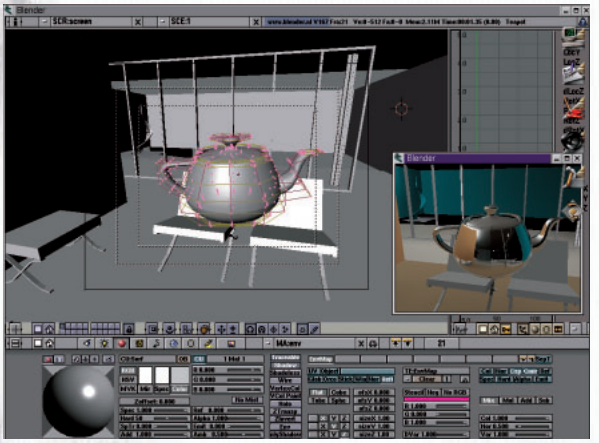
LASER5 Linux 6.0
1万2800円

豊富なバンドルソフトウェア

前述したとおり、製品版には数多くのソフトウェアがバンドルされている。中でも、後述するかな漢字変換システムや2種類のOSを切り替えて利用可能にする、ブートマネージャ「System Commander Lite」が便利に使えるだ

日本語かな漢字変換	ATOK12 SE for Linux、Wnn6 Ver.3
商用TrueTypeフォント	DynaFont 5書体
英日辞書引きソフト（研究社「新英和・和英中辞典」）	eWnn Version 1.01
日本語ワードプロセッサ	一太郎Ark Technology Preview dp/NOTE（体験版）
マルチブート	System Commander Lite英語版
セキュリティ	F-SECURE　SSH1、SSH2（評価版） Roxen Challenger Web Server（168bit暗号）Phoenix Adaptive Firewall
ウィルス対策	F-SECURE Anti-Virus Sophos Anti-virus
3Dレンダリング	BLENDER
オフィスサーバソフトウェア（試用版）	RDB各種、バックアップツール、オフィスツール
ドライバ	商用周辺機器ドライバ
開発ツール	コンパイラ

LASER5 Linux 6.0バンドルソフトウェア一覧



画面1　3Dレンダリングソフトウェア「BLENDER」

ろう。また、3Dレンダリングソフトウェア「BLENDER」は、他のディストリビューションにはバンドルされていないソフトウェアである（**画面1**）。

廉価版には、これだけ多くのソフトウェアはバンドルされていないが、かな漢字変換システムのように使用頻度の高いものがバンドルされているので、実用には十分だろう。

万全の日本語対応

かな漢字変換システムとして、変換効率のよい「ATOK12 SE for Linux」（ジャストシステム）、「Wnn6 Ver.3」（オムロンソフトウェア）がバンドルされている。また、「XIM Switcher」と呼ばれる、かな漢字変換システムをGUIで切り替えるツールがバンドルされているのが、他のディストリビューションにない特徴といえよう（**画面2**）。

さらに、デスクトップ環境のGNOMEやKDE、GUIシステム設定ツールのLinuxconf、Gimpなどもメニューが日本語化され、表示にも十分な配慮がなされている。

充実のマニュアル

製品版には9冊ものマニュアルが用意されており（廉価版は6冊）、インストール方法はもとより、Linuxconfを使ったシステム管理やGNOMEの利用方法にいたるまで、このマニュアルからさまざまな情報を得ることができる。このマニュアルの充実ぶりは、今回紹介するディストリビューションの中でも特筆できる特長といえよう。

さらなる改善が望まれるインストーラ

インストーラは、Red Hatユーザーにはおなじみのテキストベースのものである（**画面3**）。実用上は、特に問題はないのだが、多くのディストリビューション（本家のRed Hat Linux 6.1も）が、グラフィカルインストーラに移行し始めているため、多少古くさいような印象を与えるのは否めない。

また、前バージョンとなる「日本語redhat Linux 5.2」からのアップグレードを行う際は、libwcsmbs（glibc2.0の日本語対応の不備を補うライブラリ）のアンインストールをユーザーが行わなければならないという問題がある。これは、日本語化の際に対応してほしかったところだ。

非互換の1%は？

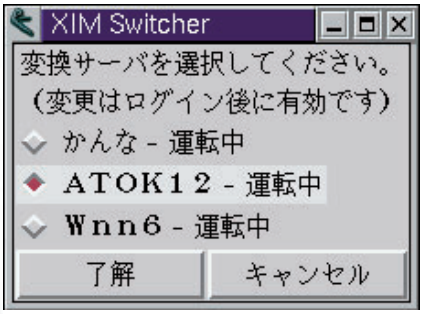
さて冒頭にも書いたとおり、LASER5 Linuxは「Red Hat Linux 6.0（英語版）と99%互換」を謳っているディストリビューションである。では、非互換の1%はどこなのだろうか？

非互換の部分として挙げられるのは、terminfoが置かれている場所である。このファイルは、Red Hat Linux 6.0（英語版）では/usr/shareディレクトリ以下に置かれているのに対し、LASER5 Linuxでは/usr/libディレクトリ以下に置かれている。これは、現在のFHS 2.0で規定されているディレクトリ構造と異なるため、terminfoを使用するアプリケーションで不具合が発生する可能性がある。

LASER5 Linux 6.0 日本語入力キット
6800円

高い総合力

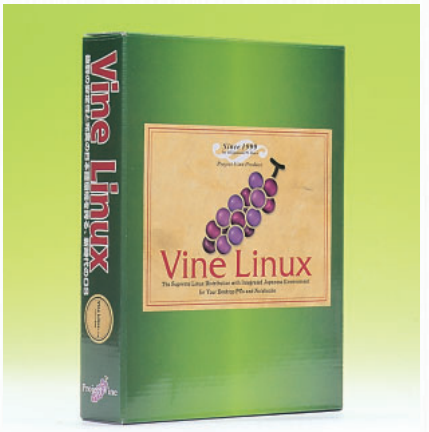
Red Hat Linux 6.0（英語版）の発売から、約5カ月経ってから発売された製品だけあって、全体的にLASER5 Linuxはよくまとまった仕上がりとなっている。製品版は、初心者から上級者まで幅広いユーザーに、廉価版もLinuxそのものを有効利用したい中級以上のユーザーにお勧めの製品である。



画面2　XIM　Switcher

画面3　テキストベースのインストーラ

Vine Linux 1.1 CR



Vine Linux 1.1 CR
価格　7800円

Vine LinuxはProject Vineが開発し、技術評論社が製品版であるVine Linux 1.1 CRを7800円で販売している。Vine Linux 1.1はRed Hat Linux 5.2をベースとしたディストリビューションで、CD-ROM2枚、インストールフロッピー2枚、マニュアルで構成され、商用TrueTypeフォントも付属する。

CD-ROMはVine Linux 1.1本体とVine Plusの1枚ずつで、Vine PlusにはVine Linuxに対応した追加パッケージなどが含まれている。また、パッケージ数がそれほど多くないため、1枚のCD-ROMにソースも収録されている。Vine Linuxのかな漢字変換ソフトウェアは標準でCannaだが、Wnn6 Ver.3をバンドルしたVine Linux 1.1 CR with Wnn6も9800円（限定2000本）で販売されている。

付属マニュアルは他のディストリビューションと比べ分量は少ない。また、説明が文章中心で、図とあまり関連性がないので、初心者に向けもうもう少し工夫が欲しい。

前述のとおりVine Linuxは約1年前にリリースされたRed Hat Linux 5.2をベースにしているため、カーネルのバージョンが2.0.36、glibcは2.0.7、XFree86は3.3.3.1と、ほかの日本語Linuxディストリビューションと比べやや古さが目につく。特にXFree86などのバージョンが上がると、新しいビデオカードなど、サポートされるハードウェアが増えるので、新規にマシンを購入する際、現状ではVine Linux以外のディストリビューションを選択するユーザーも多いのではないだろうか。

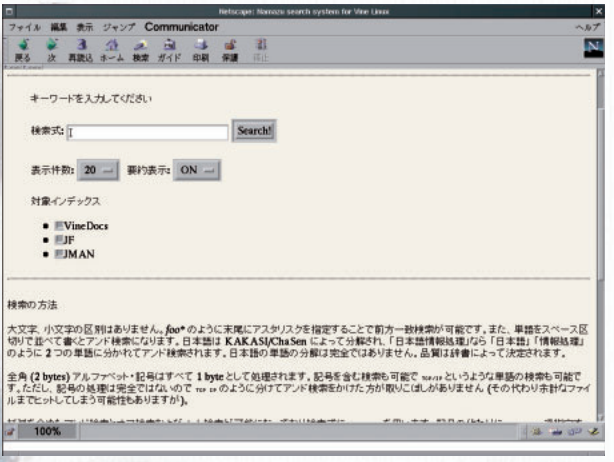
細部までこだわった日本語環境

Vine Linuxではインストール直後から快適な日本語環境が提供され、Linuxが初めてのユーザーでも日本語環境の設定で悩むことがないよう配慮されている。

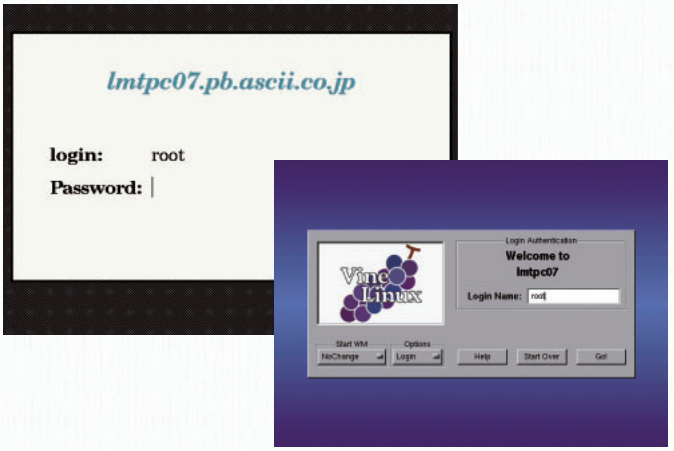
まず、他のディストリビューションに見られない充実したなEmacs（Mule）の設定ファイル.emacsの標準装備があげられる。Emacs（Mule）を初めて使うユーザーにとって、.emacsを慣れないELispで記述することは困難だ。そこでVine Linuxの用意する.emacsは使用頻度が高いメールクライアント（Mew）、ニュースリーダ（Gnus）、IRCチャットクライアント（irchat）をデフォルトで呼び出せるようになっているほか、nを2回入力し「ん」を表示させるなど細かいところまで設定が済まされている。

ただ、Vine Linuxのデフォルトではメールの送受信などをMHというコマンドで行うMew1.70がインストールされるので、.emacsもMH用となっている。MHではなくIMを採用したMewの最新版（1.94）を使用する時は設定変

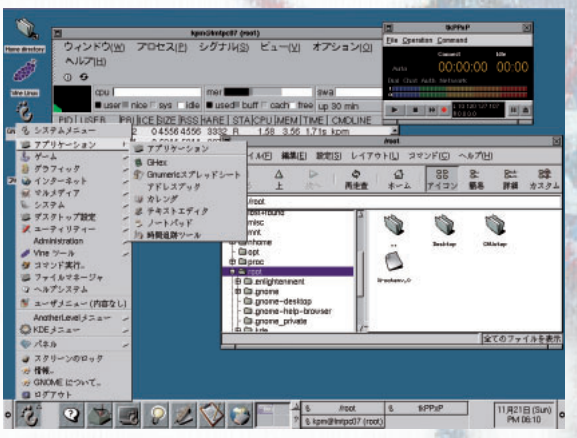
画面1　Vine Linux 1.1のデスクトップ。デフォルトのWindow Maker



画面2　namazuとApacheで日本語ドキュメントを検索



画面3　Vine Linux1.1（左上）とVine Seed（右下）のグラフィカルログイン画面



画面4　Vine Seedのデスクトップのウィンドウマネージャを GNOMEにしたところ

更が必要となる。

また、JFやJMANといったプロジェクトが、日本語化したLinux関連ドキュメントやマニュアルを標準で用意し、これらを全文検索エンジンnamazuとTcl/Tkベースのフロントエンド（TkNamazu）やNetscapeなどのWebブラウザ（Apacheサーバを起動して使用する）でキーワード検索が可能だ（**画面2**）。

印刷環境についてもProject Vineで独自に追加、作成した国産プリンタのドライバ（プリントフィルタ）を豊富に用意しているので、比較的簡単に日本語印刷が可能だ。

Vine Linuxならでは

Vine LinuxではオリジナルのVine Toolsがあり、グラフィカルなメールクライアント（Vmail）とエディタ（Vedit）が用意されている。これらは必要最小限の機能にまとめられ、Windowsユーザーなどに慣れている人にはうれしいツールだ。

また、ユーザーの情報交換の場所でもあるVine-users MLは初心者でも入りやすい雰囲気なので、これからLinuxを始めるユーザーにとっても心強い存在になるのではないだろうか。メールの流量も多くVine Linux自体の勢いも感じられる。

注目のVine Seed

Vine Linuxの開発は、はねひでや氏を中心としたProject Vineが行っている。Project Vineは日本のLinux黎明期から、Linux上での日本語環境構築のため尽力してきたPJE（Project Japanese Extensions）のメンバーを中心としている。PJEではLinuxにアドオンという形で日本語環境を提供していたが、それでもLinuxに慣れないユーザーにとって、日本語の環境構築は容易でないため、より根本的なところ（たとえばベースとしているRed Hat Linux 5.2のインストーラ）から日本語化を行っている。

現在Vine Linux次期バージョンのアルファ版（Vine Seedと呼ばれている）の開発が進められている。次期バージョンではglibcの最新版、glibc2.1や2.2系カーネルを採用し、デスクトップ環境もKDE、GNOME、XEmacsを収録するなど、GUI環境を好むユーザーの選択肢が増えそうだ。

しかし、リリース時期が未定なことや、現時点でほとんどのディストリビューションが、2.2系カーネルを採用していることを考えると、一歩遅れている感も否めない。

さて、Vine SeedをFTPサイトからダウンロードし、試験的にインストールしてみたところ、インストーラは現行のVine Linux 1.1がテキストベースであるのに対し、現段階でのVine SeedはRed Hat Linux 6.1のインストーラ（Anacondaと呼ばれる）を採用している。AnacondaはGUIベースのインストーラなので、Linuxの経験が浅い初心者も、より直感的にインストール作業が進められるのではないだろうか。

またグラフィカルなログイン画面もさま変わりし（**画面3前面**）、ログイン時に各種ウィンドウマネージャが選択できるようになっている（**画面4**はGNOME）。

次期バージョンのVine Linuxにも、現行のVine Linuxと同様、設定面で初心者にやさしく、導入後は完成度の高さで安定した環境の提供を期待する。

Section 2

日本語版ディストリビューション機能比較

TurboLinux 日本語版 4.0／Red Hat Linux 6.1 日本語版／LASER5 Linux 6.0／Vine Linux 1.1 CR

ここでは、Section 1で紹介した4つのディストリビューションを機能別に比較する。それぞれの特色がよく現れている次の5項目について、横断的に解説する。

- **インストーラ**
- **X Window Systemの環境**
- **設定ツール類**
- **日本語環境とフォント**
- **バンドルソフト、マニュアルとサポート**

インストーラ

ついこの間までLinuxのインストーラはテキストベースだった。ところが9月にリリースされたCaldera Open Linux2.2をきっかけに、先月リリースされたRed Hat Linux 6.1もAnacondaというGUIインストーラを取り入れるなど、インストーラのGUI化は今後ますます加速していきそうだ。

また、Red Hat Linux 6.1日本語版ではパーティション作成ツールのfdiskがなくなっているなど、以前のインストーラより細かい設定がやりにくくなっている。インストーラがブラックボックス化されてきているのも最近の流行といえるだろう。

●TurboLinux 日本語版 4.0

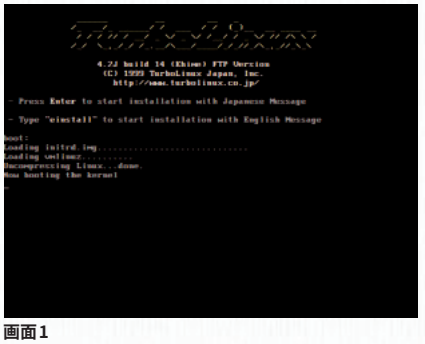
ウィンドウマネージャはたくさん用意されているけれどインストーラが決めてしまい、インストール後もどうやってウィンドウマネージャを変更すればよいのかわからないというユーザーも多いだろう。

そこでTurboLinuxではインストール中に使いたいウィンドウマネージャを選択できる**（画面5）**。このあたりはユーザーの遊び心をよくとらえたチューニングといえるだろう。

ISAカードの自動認識やSMP、APMのカーネルタイプが選択できるなど、設定が多彩にできるTurboLinuxだが、ハードディスクからのインストールがサポートされていないのが少し気になった。CD-ROMドライブを装備してないノートPCも多くあり、フロッピーからインストーラを起動して各種ファイルをハードディスクから読み込むというケースも考えられるので、ぜひサポートして欲しいところだ。

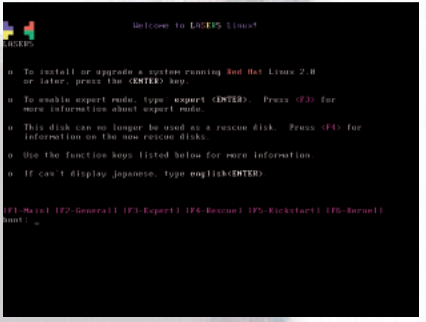
●Red Hat Linux 6.1日本語版

今回取り上げる4つのディストリビューションの中で、唯一GUIインストーラを採用しているのがRed Hat Linux 6.1日本語版だ。Red Hat Linux 6.1日本語版に採用されているAnacondaは

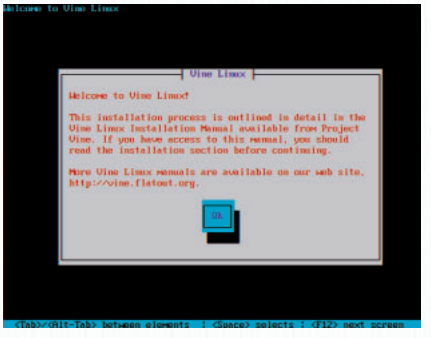


画面1
Turbo Linuxのインストーラ起動画面。ロゴがこっている。

画面2
SVGAサーバで表示されるRed Hat Linux 6.1日本語版のインストーラ。他のディストリビューションもGUIになってゆくのだろうか。



画面3
LASER5 Linuxのインストーラ。Red Hat Linux 6.0とほぼ同じだ。



画面4
Vine Linuxのインストーラ。

インストーラ起動後にSVGAサーバを立ち上げ、X上のマウス操作でインストールを行う。

たとえばタイムゾーンの設定で、これまでは全てのタイムゾーン一覧から、キーボードでカーソルを上下させて選択していた。目的のタイムゾーンを選択するるのにひたすらカーソルを移動させていくのは、結構面倒なものだ。

そこでRed Hat Linux 6.1日本語版では画面に表示された世界地図上の日本にマウスを合わせて時間が設定できる**（画面6）**。好みもあるだろうが、このような設定方法はGUIならではだ。

また、これまでどおりテキストベースのインストールも選択できるので、このあたりは、環境に合わせ適宜選択すれば良いだろう。

●Vine Linux 1.1 CR

Vine Linuxではキーボード選択の際、JP106とUS101の選択だけでなく、それぞれでCtrlキーとCaps Lockキーの入れ換えもできる**（画面8）**。Ctrlキーはaキーの左側がよいという人にとって、Linuxをインスールするたびにキーレイアウトを設定しなおすのは面倒なものだ。このあたりは開発者が昔からのLinuxユーザーであるVine Linuxの特徴で、玄人好みな設定項目だ。

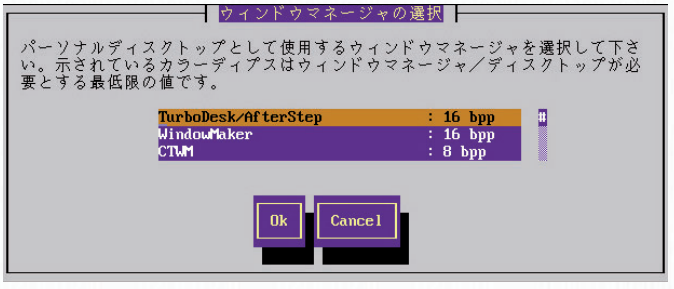
また、初心者ユーザーを想定しているのかデフォルトでsendmailが起動しないようになっている。このあたりの配慮もうれしいところだ。

●LASER5 Linux 6.0

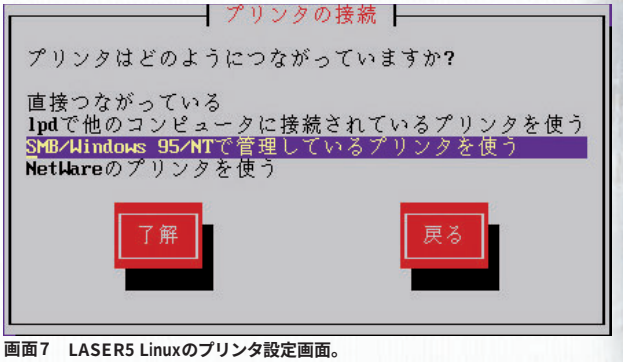
最後にRed Hat Linuxと99%の互換性をうたうLASER5 Linuxだが、メニューを日本語化した英語版Red Hat Linuxという向きで、他のテキストベースのインストーラに比べると特にチューニングされた箇所がない分やや見劣りする。

●どれが良いの？

これからLinuxをインストールしてみようという人にはRed Hat Linux 6.1日本語版がお勧めだ。日頃マウス操作に慣れたユーザーにとっては、同じ設定内容なら、アイコンを見ながらの直感的な作業のほうが、心理的負担が少なくて良いだろう。ただし、少しLinuxに慣れてきたらインストーラだけでなく、インストール後の環境も含め、自分に合ったディストリビューションを探して欲しい。



画面5
TurboLinuxではインストール中にウィンドウマネージャが選択できる。



画面7　LASER5 Linuxのプリンタ設定画面。

画面6
GUIならではの設定。画面はRed Hat Linux 6.1日本語版で、マウスポインタを地図上にあわせるとタイムゾーンが選択できる。

画面8
Vine Linuxではキーボードの選択場面で、Ctrlキーと Caps Lockキーの入れ換えができる。

ウィンドウ環境

Linuxでグラフィックス画面を使って操作するのは、従来X Window System＋ウィンドウマネージャという形であった。Linux（あるいはUNIX）は、コンソールからコマンドを入力することで操作したり、サーバ用のソフトウェアを動作させたりすることは得意だが、そのウィンドウマネージャの使い勝手は、Windows 95／98やMac OSに比べると非常に物足りなかった。

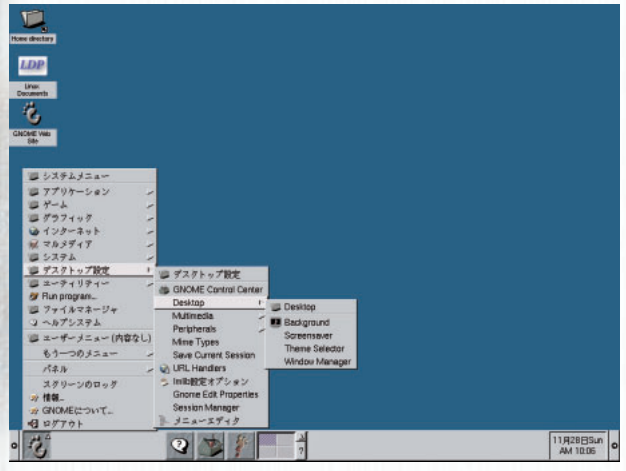
しかし、1999年になって多くのディストリビューションが、GNOMEとKDEという統合デスクトップ環境を標準採用した。統合デスクトップ環境では、アプリケーション開発用のツールキットと、共通アクセサリを用意している。それを使うことにより、メニューの表示形式やダイアログボタンなどのルック＆フィールが統一され、ウィンドウシステムの操作性が向上した。

GNOME（GNU Network Object Model Environment）は、GNUプロジェクトが開発しているデスクトップ環境で、ツールキットには「GTK＋」を採用している。GNOMEには専用のウィンドウマネージャは存在せず、Enlightenmentや Window Makerなどのウィンドウマネージャの中でGNOMEに対応するものを使用することになっている。

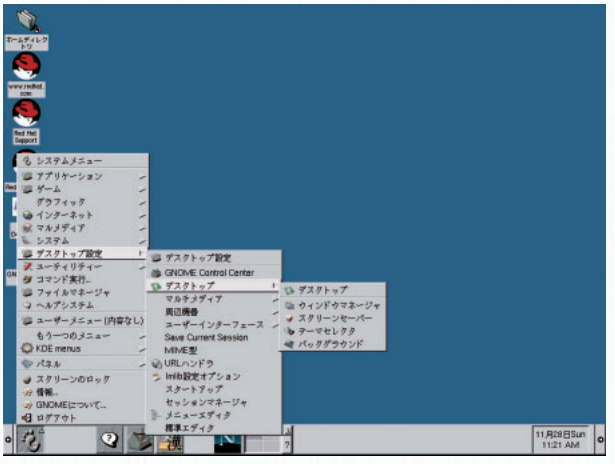
KDE（The K Desktop Environment）は、「The KDE Team」によって開発されていて、ツールキットには、ノルウェーのTroll Tech社が開発した「Qt」が使われている。

統合デスクトップ環境の採用で使用感が似てきた

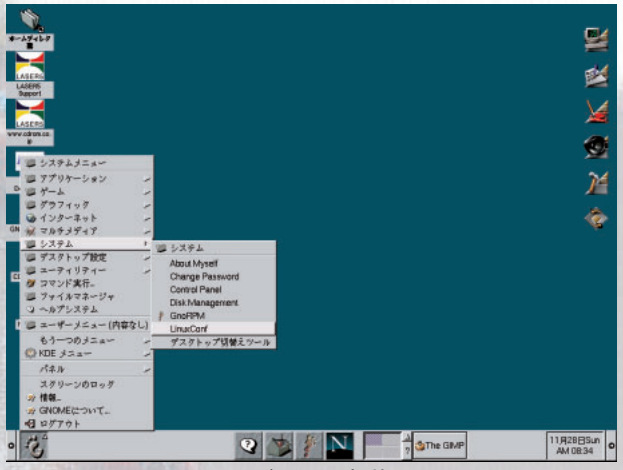
TurboLinux、 Red Hat Linux、LASER5 Linuxの3つとも、GNOMEとKDEの両方を含んでいる。それぞれのディストリビューションで、GNOMEを選んだ場合のメインメニューが**画面1**～**画面3**である。メッセージが日本語化されているか否か微妙に差があるが、基本的に同じように使うことができる。



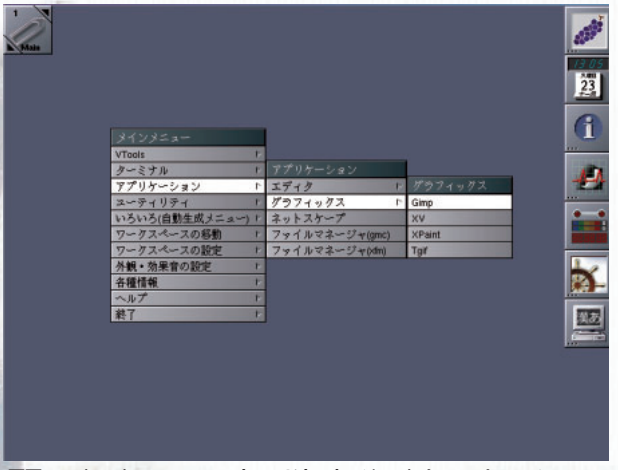
画面1　TurboLinux 日本語版4.0のデスクトップ環境GNOMEのメニュー。



画面2　Red Hat Linux 6.1 日本語版のデスクトップ環境GNOMEのメニュー。



画面3　LASER5 Linux 6.0のデスクトップ環境GNOMEのメニュー。



画面4　Vine Linux 1.1 CRのウィンドウマネージャWindow Makerのメニュー。

KDEとGNOMEのどちらかを選ぶかは、ユーザーの好みだが、両方ともインストールした場合には、Red Hat 6.1やLASER5では、メニューから相互のプログラムが呼び出せるようになっている（下から6番目に「KDEメニュー」がある）。

Vine Linux 1.1には、デスクトップ環境が含まれていない。ウィンドウマネージャのWindow Makerを採用している**（画面4）**。次期バージョンではデスクトップ環境も含まれるようだ。

なお、KDEのデスクトップをOpenLinux 2.3日本語版の例だが挙げておこう**（画面5）**。OpenLinux 2.3では、KDEが標準のデスクトップ環境となっていてGNOMEは入っていない。

GNOMEやKDEには、ファイルマネージャが含まれていて、Windowsのエクスプローラと同じ使い勝手を実現している。**（画面6）**。

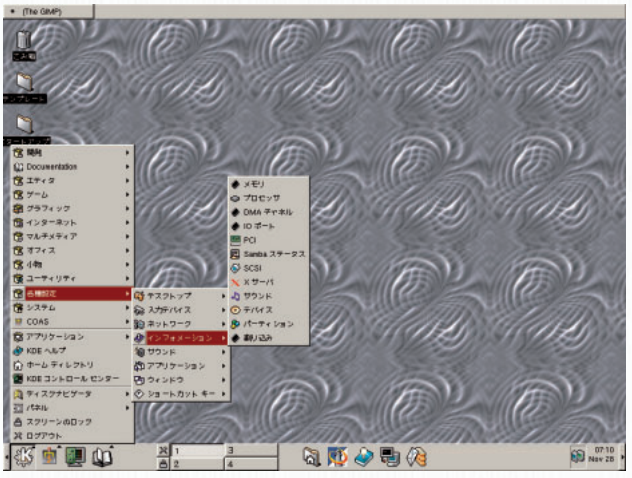
ウィンドウマネージャの切り替え

TurboLinuxでは、コンソールで動作するユーティリティ「TurboWMCfg」（プログラム名: turbowmcfg）によって、**画面7**にあるようなウィンドウマネージャを選ぶことができる。

Red Hat LinuxとLASER5 Linuxは、メニューの「システム」からデスクトップの切り替えプログラム「Desktop Switcher」を起動することで、GNOMEとKDE、それ以外の3種類を選ぶことができる**（画面8）**。

ところで、CPUがPentium 200MHz以下といったちょっと古めのマシンで、統合デスクトップ環境を使ってみるとわかるが、動作が遅く時々マウスが止まることもあるぐらいだ。実は非常に大きく重いシステムになってきている。そういう場合には、デスクトップ環境ではなく、ウィンドウマネージャだけで利用するVine Linuxのほうが使い勝手がよいだろう。

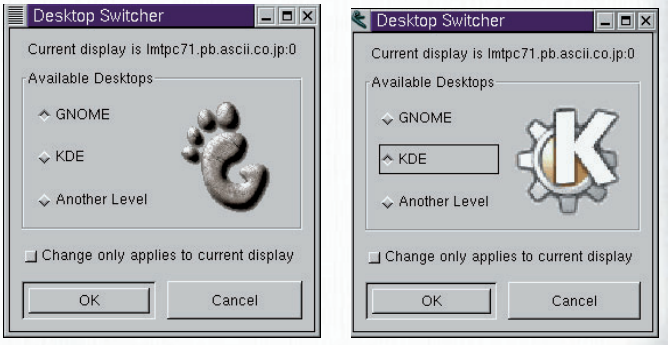
XFree86は、TurboLinux 4.0、Vine Linux 1.1、LASER5 Linux 6.0では3.3.3.1を、TurboLinux PRO 4.2では3.3.4を、そしてRed Hat Linux 6.1では3.3.5を使用している。新しいバージョンではサポートカードが増えているため、インストール時にグラフィックスカードを認識しないという問題が解決されていることもある。



画面5　OpenLinux 2.3日本語版のデスクトップ環境KDEのメニュー。

画面6　GNOMEのファイルマネージャ「GMC」（GNU Midnight Commander）

画面7　TurboLinuxのウィンドウマネージャ切替ツール「TurboWMCfg」。



画面8　Red Hat Linux 6.1 日本語版、LASER5 Linux 6.0でウィンドウ環境を切り替えるプログラムDesktop Switcher。

システム管理ツール

Linuxを含め、UNIX系OSでシステムを設定するには、シェルでコマンドを打ち込むか、設定ファイルをエディタで書き換える方法が一般的だ。しかし、この方法は、初心者にとって非常に敷居が高いうえに、システムを一括管理しにくいという欠点がある。

これを解決すべく、各ディストリビューションには、GUIで操作する設定ツールが付属しており、旧来の方法より簡単にシステム管理ができるようになっている。ところが、ディストリビューションによって付属するツールが違うので注意が必要だ。

今回取り上げた4つのディストリビューションに付属するツールをみると、TurboLinuxが自社開発の独自ツールを積極的に取り入れているのに対し、ほかの3つは比較的共通点が多い。

●TurboTools

TurboLinuxは、テキストベースのGUIツール群「TurboTools」を用意している。これは、パッケージ管理、ファイルシステム管理、ユーザー管理、ネットワーク設定などを行う個別のツールの総称で、各ツールは独立したコマンドとなっている。

独立しているとはいえ、統一されたルック&フィールを持ち、シェルのコマンド名補完によって簡単に見つけられるように各コマンドの名前を“turbo”で始めるなど、使い勝手を考慮している（**画面1**）。中でも、ウィンドウマネージャを切り替えるturbowmcfgは簡単に使えるので重宝するだろう。

しかし、次に紹介するLinuxconfなど、他のディストリビューションで採用しているツールもあればより便利だ。

●Linuxconf

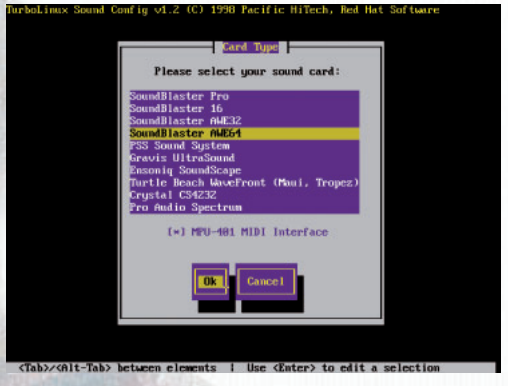
TurboLinuxを除いた3つのディストリビューションにはLinuxconfという統合システム管理ツールが用意されている。Linuxconfは、ユーザー管理、ネットワーク設定はもちろん、sendmailやSAMBAサーバの設定まで行うことのできる超多機能システム管理ツールだ（**画面2**）。

ユーザーインターフェイスも豊富で、シェルで使えるキャラクタベースGUI、X用のGUIのほか、Webブラウザからも利用できる。

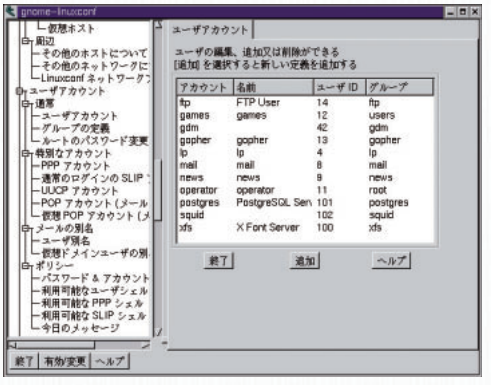
TurboLinux 4.xを使っている方も、開発元であるLinuxconf ProjectのWebページ（http://www.solucorp.qc.ca/linuxconf/）から、Red Hat Linux 5.1／5.2用RPMパッケージを入手すれば、英語版が利用できる。

●RPMパッケージ管理

ここで取り上げたディストリビュー

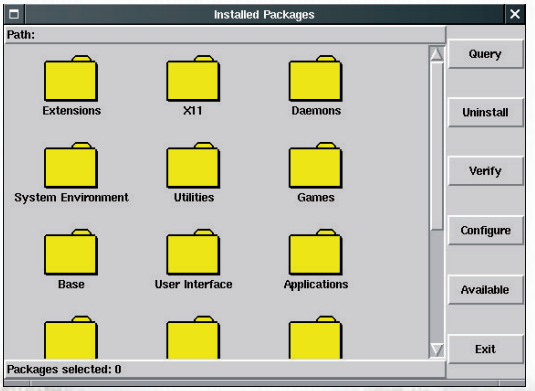


画面1
TurboToolsの1つ、turbosoundcfg。コンソールから使えるのは、意外と便利だ。



画面2
LASER5 LinuxのLinuxconfだけが日本語化されている。多機能なぶん、読むべき情報量も多いので日本語表示はありがたい。

画面3
Gnome RPMは、Windowsのエクスプローラーライクなユーザーインターフェイスをもつ。



画面4
以前はよく使われたRPMパッケージ管理ツールglint。動作が単純なので、ちょっと使いたいときによいだろう。

ションは4種類ともRPM（Red Hat Package Manager）という仕組みを使いソフトウェアのパッケージ管理を行っている。これによってインストールやアンインストールが正しく行えるようになっているのだ。

RPMパッケージは、コマンドラインからrpmコマンドにを使うことよって管理することもできるが、Gnome RPM（gnorpm）により、GUIで管理することができる（**画面3**）。

LASER5 Linuxと、Vine Linuxでは、Gnome RPMより単純なglintというツールを利用することもできる（**画面4**）。

●ハードウェアの自動認識

Red Hat Linux 6.1には、Linuxの起動時にハードウェアを自動認識する「Kudzu」というソフトウェアが新たに追加された。これは、起動する際にハードウェア情報を調べ、前回の起動時に保存しておいた情報と比べることで、ハードウェア環境の変更を検出し、自動設定する仕組みだ。

前回の起動時と構成が変わっていると、**画面5**のようにKudzuが起動する。ユーザーは、Kudzuが検出した変更内容を確認して、設定変更するかどうかを決めることができる。

デスクトップ設定

X Window Systemのデスクトップをカスタマイズするのはもはや常識なのかもしれない。GNOME、KDE、Window Makerとも、柔軟なカスタマイズが可能だ。

●GNOME、KDE

Vine Linux以外に標準で付属するGNOMEとKDEは、デスクトップを「テーマ」でカスタマイズできる。それぞれ、GNOMEコントロールセンター（**画面6**）、KDEコントロールセンター（**画面7**）からテーマを変更できる。

●Window Maker

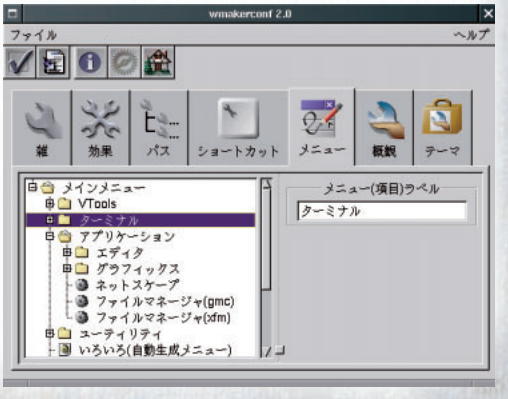
Vine Linuxは、Window Makerが標準のウィンドウマネージャとなる。Window Makerは、GNOMEやKDEのような統合環境は提供しないが、その分、動作が軽快だ。カスタマイズの自由度も高く、テーマによる一括カスタマイズも可能だ。Vine Linuxには、Window Makerの設定変更ツール、wmakerconfを日本語化したものが用意されている（**画面8**）。

各ディストリビューションで採用している主な設定ツールを70、71ページの表にまとめたので、あわせて参考にしてほしい。

画面5
Red Hat Linux 6.1では、起動時に新しいハードウェアが見つかると自動設定をするかどうか、ユーザー確認する画面になる。

画面6
GNOMEのデスクトップテーマ選択を行う。この例は日本人にはなじめない色使い？

画面7
KDEコントロールセンターは、テーマごとに実際の表示イメージを見せてくれる親切設計。



画面8
シンプルで上品な外観を持つWindow Makerのカスタマイズツールwmakerconf。

日本語環境とフォント

Linuxで日本語入力を行うための、かな漢字変換ソフトウェアとしてよく使われているのは、Canna、Wnn4、Wnn6 Ver.3（以下Wnn6）、ATOK12 SE for Linux（以下ATOK12 SE）の4種類である。CannaとWnn4はフリーソフトウェアで、そのうちのCannaは今回の4ディストリビューションすべてに含まれている。

本格的に使うなら商用のかな漢字変換ソフトが欲しい

Wnn6はオムロンソフトウェアが販売している製品で、非常に多くの語彙を収録した辞書によって変換効率を高めている。WnnはUNIX上で長い間使われてきた実績もあり、Mule（Emacs）エディタとの相性が良いという特徴がある。

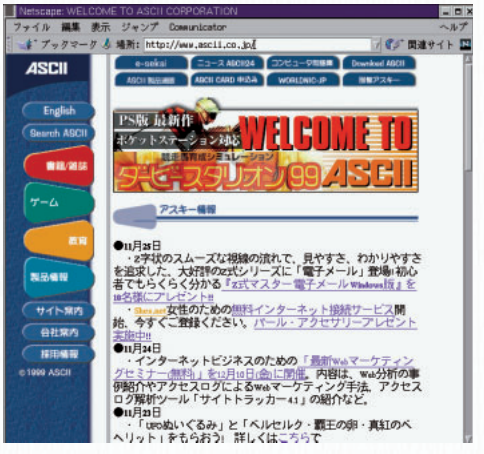
ATOK12 SEは、Windows環境で根強いファンがいるATOKをLinuxに移植したもので、強力な構文解析によってかしこい変換を実現している。なお、ATOK12 SEは単体では発売されていないため、これを使いたい人はバンドルされているディストリビューションを購入することになる。

Vine Linux 1.1は、標準ではCannaを使用しているが、Wnn6をバンドルした製品「Vine Linux 1.1CRW」（以下Vine Linux CRW）も限定発売している。そのほかのTruboLinux、Red Hat Linux、LASER5 Linuxの各製品版には、Wnn6とATOK12 SEの両方をバンドルしている。

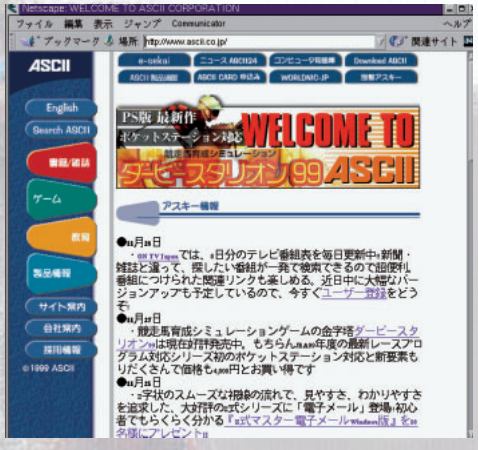
Wnn6やATOK12 SEのインストール方法は、ディストリビューションごとにそれぞれ違っている。TurboLinuxでは、バイナリCDからLinuxをインストールすると同時に、ATOK12がインストールされる。Vine Linux CRWも、LinuxをインストールするとWnn6が標準の日本語入力になるようにインストールされる。

LASER5 Linuxでは、Linuxインストール後にプロダクトCDからインストールする必要がある。Red Hat Linuxも、同様にコマーシャルCDからインストールする。LinuxではCDをセットするだけでインストーラが起動されるというような便利な機能はないので、どうしてもマニュアルを見ながらインストールすることになる。もっと簡単に

画面1　TurboLinux日本語版4.0をインストールした直後のNetscape Navigator
数字のほうが、日本語より大きくなっている。



画面2　Red Hat Linux 6.1日本語版をインストールした直後のNetscape Navigator
数字のほうが、日本語より少し小さくなっている。



画面3　LASER5 Linux 6.0をインストールした直後のNetscape Navigator
数字のほうが、日本語よりかなり小さくなっている。

画面4　Vine Linux 1.1CRをインストールした直後のNetscape Navigator

使えるようにして欲しいものだ。

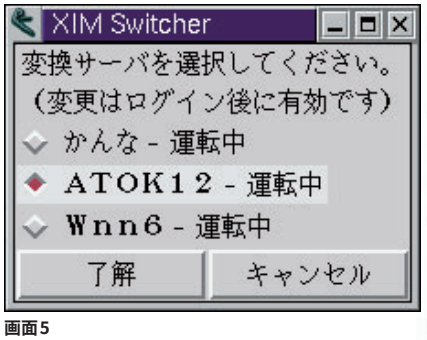
なお、LASER5 Linuxでは複数のかな漢字変換を簡単に切り替えて使えるように、XIM Switchというプログラムを用意している（**画面5**）。ほかのディストリビューションには、このような機能がほしいところだ。

TrueTypeフォント

WindowsやMacで日本語を表示する場合、TrueTypeフォントを使用して美しい表示を実現している。X Window SystemはTrueTypeフォントを扱えないため、日本語版のディストリビューションでは、X-TT（X-TrueType Server）と呼ばれるパッチを当てて、TrueTypeフォントを使用できるようにしている。

また、製品版のディストリビューションには、商用TrueTypeフォントを数種類組み込んできれいな日本語表示が行えるようになっている。

なお、TurboLinuxは、リョービ日本語TrueTypeフォント5書体をバンドルしている。LASER5 LinuxとRed Hat Linuxは、どちらもDYNALABのDynaFont5書体をバンドルしている。



画面5
LASER5 Linuxでは、メニューからこのXIM切り替えプログラムを起動し、GUIでかな漢字変換ソフトを選ぶことが可能だ。

Vine Linuxでは、DynaFont4書体をバンドルしている。

ところで、最近X-TTでも使われているフリーなTrueTypeラスタライザ「FreeType」が、Apple社の特許を侵害している可能性があると報道された。そのためか、Red Hat Linux 6.1日本語版は、XFree86にX-TTを組み込んでいない。そしてDynaFontも単にファイルがCD-ROMに入っているだけでインストーラがないため、ユーザーが自分で環境を作る必要がある。

Netscapeの表示で見るインストール直後の設定

そこで、4つのディストリビューションで、インストール直後にNetscape Navigatorを使ってみた（**画面1〜4**）。日本語と英数字のバランスを見てもらうとわかるように、Navigatorのフォントの設定を何もしないうちは、こんなにも表示が読みにくいのだ。TrueTypeフォントを使ったフォントの設定をすれば**画面6**のようにきれいに表示することができる。

Vine Linuxでは、インストール後のデフォルトの設定でも十分なように使い勝手のよい日本語環境を構成している。ほかのディストリビューションでは、Wnn6やATOK12 SEといった変換精度の高い日本語入力ソフトと、商用TrueTypeフォントに魅力を感じる人もいるだろう。

ディストリビューションはユーザーの便宜を図るために、各種のソフトウェアをまとめてパッケージングしているわけで、ユーザーはLinuxをインストールしたらすぐに使える環境が構築されていることを期待するだろう。そういった点では日本語環境の整備はまだ不十分と感じる。

画面6
フォントを画面のような設定にするまでに何回も試行した。これはTurboLinux日本語版4.0のNetscape Navigatorの画面。

バンドルソフトウェア

一般的なデスクトップ環境、およびサーバソフトウェアは、どのディストリビューションもひととおりそろっている。そのため、ここでは、特徴的なものに話題を絞ることにしよう。

●オフィス製品

Red Hat Linuxには日本語ワードプロセッサ「dp/NOTE」が、LASER5 Linuxには「dp/NOTE（体験版）」がバンドルされる。また、TurboLinux、Red Hat LinuxとLASER5 Linuxには「一太郎 Ark for Java Technology Preview」が付属する。

表計算ソフトウェアは、Vine Linux以外の製品に、GNOME系フリーソフトウェア「gnumeric」が含まれる。このソフトは、グラフ作成の機能こそないものの、表計算ソフトウェア本来の機能は市販のものに劣らない（**画面1**）。

TurboLinux PROにバンドルされる商用オフィススイート「Applixware日本語版 4.4.2」は、ワードプロセッサ、表計算、グラフィックス、プレゼンテーション、データベースなどのソフトウェアを統合した製品だ（**画面2**）。

ワードプロセッサの「Words」は、文書中にクリップアートを挿入したり、Microsoft Wordの文書を読み込むことができ、表現力もかなりのものだ。

また、プレゼンテーション作成ツールも、簡単な操作で資料を作成できるが、ATOK12 SEで日本語を入力すると、ときおりおかしな動作をした。

Linuxで動作する日本語対応オフィス製品はまだほとんどないが、Applixwareが現時点で唯一実用になる製品だろう。なお、本製品は、TurboLinux PROのバンドル販売のみとなっている。

●3Dグラフィックソフトウェア

LASER5 Linuxには、3Dレンダリングソフトウェア「BLENDER」のフリー版が付いている（**画面3**）。そのままでも基本的な機能は利用できるが、ライセンスを購入するとラジオシティ計算をはじめ、より高度なCG計算が可能になる。

●英日辞書引きソフトウェア

LASER5 Linuxには研究社「新英和・和英中辞典」のデータをもつ辞書引きソフトウェア「eWnn」がバンドルされている。動作には、Mule 2.3を必要とするが、LASER5 LinuxにはMuleが含まれていない。ぜひとも改善するようお願いしたい。

今回は、試用のためTurboLinux上で動作させてみた（**画面4**）。動いてしまえばとても便利に使えるソフトウェアだ。

●ブートマネージャ

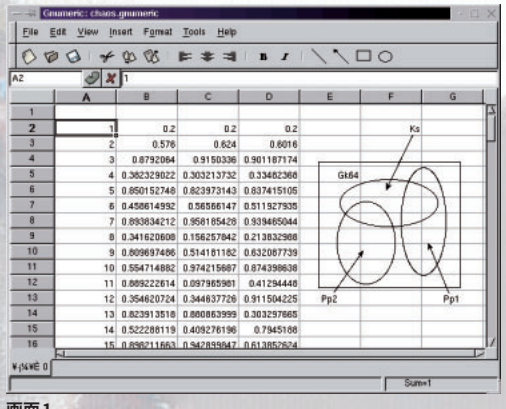
複数のOSを共存させ、起動時に利用するOSを選べるようにするのがブートマネージャだ。1台のハードディスクに複数のLinuxをインストールしたい場合や、LinuxとWindowsを共存させたいときに便利なソフトウェアだ。

TurboLinux、TurboLinux PRO、LASER5 Linuxには、ブートセレクタソフトウェア「System Commander Lite」が添付される。LASER5 Linux日本語入力キットには付属しないので注意してほしい。

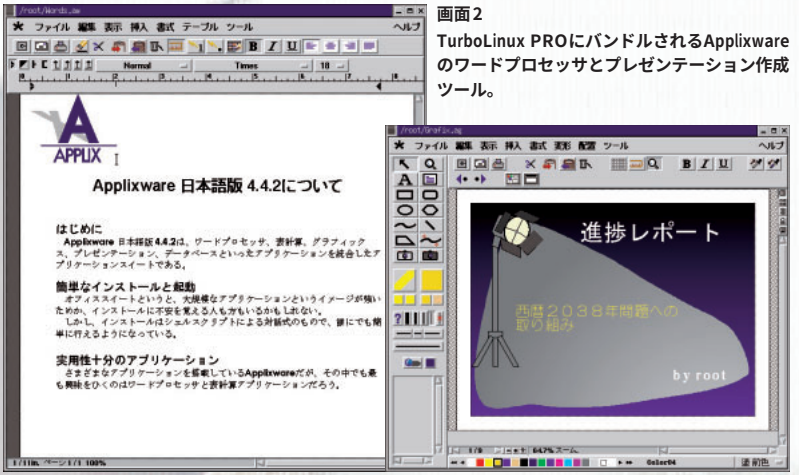
このユーティリティは、「System Commander」の機能を限定したものだが、TurboLinuxとLASER5 Linuxに付属のものでは若干制限内容が異なるようだ。詳細については各販売元に確認してほしい。

マニュアル

マニュアルの量はディストリビューションによって大きく違う。これは、



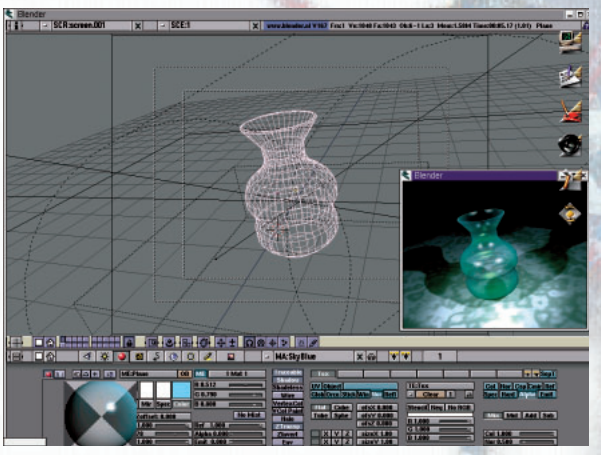
画面1
虚飾を廃し、表計算ソフトウェア本来の機能を優先させるGnumeric。



画面2
TurboLinux PROにバンドルされるApplixwareのワードプロセッサとプレゼンテーション作成ツール。

意外と見落とされがちな点だが、Linuxに不慣れなユーザーにとっては重要なポイントの1つだ。

では、順に見ていこう。



画面3
3Dレンダリングソフトウェア BLENDERで描画した謎のガラス器。ひと昔前のSFを彷彿とさせる。

●LASER5 Linux

一番充実しているのがLASER5 Linuxで9冊組だ。中心になるのは「インストールガイド」、「ユーザー活用ガイド」、「システム管理ガイド」の3冊だろう。それぞれとても丁寧に解説されており、Linuxコマンドの説明も充実している。さらに、JF（Japanese FAQ）、JMAN（Japanese man）、LDP（Linux Documentation Project）など、1万ページを超える文書を収録したドキュメントCDも利用できる。このCD-ROMは、Linux、およびWindowsから全文検索エンジンNamazuを使って自由に検索可能だ。

また、LASER5 Linux 日本語入力キットには、サポート権、System Commander Lite、ドキュメントCDが含まれないので、これらに関する3冊が省かれ6冊組となる。

●TurboLinux

TurboLinuxは300ページ弱のものが1冊付属し、TurboLinux PROは3冊に増えているが、Applixware関連の説明が増えている以外はTurboLinuxとほぼ同じ量になる。TurboLinux独自のコマンドだけでなく、基本的なLinuxコマンドも説明されているので、初級者の心強い味方になるだろう。

●Red Hat Linux

Red Hat Linuxには、150ページほどの「インストレーションガイド」が1冊付属する。インストール説明と、GNOME環境の利用解説、バンドルする商用ソフトのインストールがまとめられている。初心者がこのマニュアルだけでLinuxに入門するのは難しい。

●Vine Linux

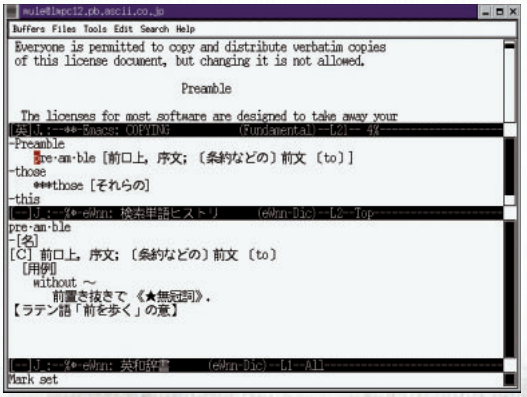
Vine Linuxのマニュアルは、130ページほどのものが1冊だ。インストール、クイックスタートガイド、Vine Toolsの使い方などが説明されている。マニュアルの情報量は少ないが、全文検索システム「TkNamazu」でVine関連文書、JF、JMANといった有用な情報を検索できる（**画面5**）。

サポート

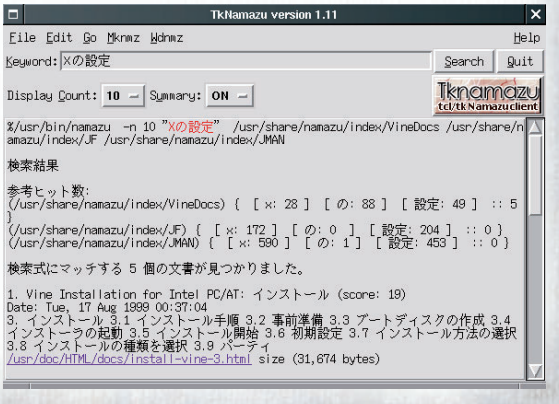
製品に付属するサポートの範囲は、各社ともインストールまでに限定しているが、TurboLinuxと、LASER5 Linuxは、インストール後に受けられる有償のオプションサポートを用意している。

各社の基本的なサポート内容は、70～71ページの表にまとめているのでそちらをご覧いただきたい。

正式なサポートのほかにも、ユーザー同士の情報交換の場として、メーリングリストやWeb掲示板も活用しよう。LASER5 Linux、Vine Linuxはメーリングリストが、TurboLinuxは、メーリングリストとWeb掲示板がそれぞれ開設されている。それぞれの案内は各社のWebページを見てほしい。



画面4
日英辞書引きソフトウェアのeWnnをMule上で使う。LASER5 Linuxでも使いたい。



画面5
TkNamazuでLinux情報の宝庫の全文検索。これなしで迷い込んだら宝を見つけるのは至難の業だ。

日本語版4ディストリビューションの主なスペック

製品名	TurboLinux 日本語版 4.0	TurboLinux PRO 日本語版 4.2	公認 Red Hat Linux 6.1日本語版
販売元	ターボリナックス ジャパン株式会社		レッドハット株式会社
URL	http://www.turbolinux.co.jp/		http://www.jp.redhat.com/
CD-ROM／フロッピーディスク	CD 3枚／FD 2枚	CD 5枚／FD 3枚	CD 3枚／FD 1枚
マニュアル	1冊	3冊	1冊
価格	9800円	2万9800円	1万2800円
動作条件			
CPU／メモリ	Pentium互換／32Mバイト以上		Pentium互換／16Mバイト以上
ハードディスク	600Mバイト（最低200Mバイト程度）		500Mバイト以上
インストール			
インストーラ 日本語対応	日本語／テキスト		日本語／グラフィカル
ブートメディア	CD-ROM、フロッピーディスク、DOS		CD-ROM、フロッピーディスク、DOS
インストール元メディア	CD-ROM、FTP、NFS		CD-ROM、ハードディスク、FTP、NFS、HTTP
システム構成			
パッケージ管理システム	RPM		RPM
デスクトップ環境	GNOME、KDE、AfterStep、Window Maker、Enlightenment、CTWM、OpenLook、TWN	GNOME、KDE、AfterStep、Window Maker、Enlightenment、CTWM、ICE、OpenLook、TWN	GNOME、KDE、fvwm、fvwm2、Window Maker、AfterStep
日本語入力ソフト	ATOK12 SE、Wnn6 Ver.3、Canna	ATOK12 SE、Wnn6 Ver.3、Canna、SKK	ATOK12 SE、Wnn6 Ver.3、Canna
バージョン			
（注） カーネル	2.2.9		2.2.12
標準Cライブラリ	glibc 2.0.7		glibc 2.1.2
XFree86	3.3.3.1	3.3.4（3.3.5）	3.3.5
GNOME	1.0.4	1.0.7	1.0.39
KDE	1.1.1		1.1.2
設定ツール			
ユーザー管理	turbousercfg		linuxconf、kuser
ネットワーク	turbonetcfg		netcfg、linuxconf
Xサーバ設定	turboxcfg、XF86Setup		XF86Setup
ウィンドウマネージャ変更	turbowmcfg		switchdesk-gnome
サウンド	turbosoundcfg		sndconfig
起動デーモン	turboservice、tksysv、ksysv		linuxconf、Control Panel
パッケージ	turbopkg、gnorpm		gnorpm
主なバンドルソフト			
日本語入力	ATOK12 SE for Linux、Wnn6 Ver.3		ATOK12 SE for Linux、Wnn6 Ver.3
日本語TrueTypeフォント	リョービ 日本語TrueType 5書体		DynaFont和文5書体
ブートマネージャ	System Commander Lite		—
そのほか	OSS、一太郎Ark Technology Previewなど	Applixware 日本語版 4.4.2、OSS、Nmail、一太郎Ark Technology Previewなど	dp/NOTE、一太郎Ark Technology Preview
サポート			
サポート内容	対応ハードウェアの認識、設定からインストールまで ※1		インストール、無料優先FTPアクセス、Red Hatアップデートエージェンシー
サポート方法	Web、E-Mail		電話、Web（E-Mail）
サポート期間	90日間		30日間の電話サポート、90日間のWeb（E-Mail）サポート、180日間の無料FTPサイト優先使用権
サポート件数	3件		特に制限なし
備考	※1 有償オプションとして、インターネット接続、プリンタ設定、サウンド設定などのサポートあり		

（注） バージョン内の（ ）かっこは、付属のCD-ROMからインストールが可能なバージョン

製品名	LASER5 Linux 6.0	LASER5 Linux 6.0 日本語入力キット	Vine Linux 1.1 CR
販売元	レーザーファイブ株式会社		株式会社技術評論社
URL	http://www.laser5.co.jp/		http://www.gihyo.co.jp/
CD-ROM／フロッピーディスク	CD 6枚／FD 2枚	CD 3枚／—	CD 2枚／FD 2枚
マニュアル	9冊	6冊	1冊
価格	1万2800円	6800円	7800円（9800円 ※1）
動作条件			
CPU／メモリ	i486互換／16Mバイト以上		Pentium以上推奨／32Mバイト以上
ハードディスク	600Mバイト以上		600Mバイト以上
インストール			
インストーラ 日本語対応	日本語／テキスト		日本語／テキスト
ブートメディア	CD-ROM、フロッピーディスク、DOS		CD-ROM、フロッピーディスク、DOS
インストール元メディア	CD-ROM、ハードディスク、FTP、NFS、HTTP		CD-ROM、ハードディスク、FTP、NFS、SMB
システム構成			
パッケージ管理システム	RPM		RPM
デスクトップ環境	GNOME、KDE、twm、fvwm、fvwm2、Window Maker、AfterStep Lesstif WM		Window Maker、Fvwm2
日本語入力ソフト	ATOK12 SE、Wnn6 Ver.3、Canna		Canna（Wnn6 Ver.3 ※1）
バージョン			
（注） カーネル	2.2.5（2.2.11）		2.0.36
標準Cライブラリ	glibc 2.1.1		glibc 2.0.7
XFree86	3.3.3.1（3.3.4）		3.3.3.1
GNOME	1.0.11		—
KDE	1.1.1（1.1.2-pre3）		—
設定ツール			
ユーザー管理	linuxconf、kuser		linuxconf、Control panel
ネットワーク	netcfg、linuxconf、netconf		netcfg、linuxconf、netconf
Xサーバ設定	XF86Setup、Xconfigurator		XF86Setup、Xconfigurator
ウィンドウマネージャ変更	switchdesk-gnome		—
サウンド	sndconfig		sndconfig
起動デーモン	linuxconf、Control Panel、ksysv		linuxconf、Control panel
パッケージ	gnorpm、glint		gnorpm、glint
主なバンドルソフト			
日本語入力	ATOK12 SE for Linux、Wnn6 Ver.3		（Wnn6 Ver.3 ※1）
日本語TrueTypeフォント	DynaFont和文5書体		DynaFont和文4書体
ブートマネージャ	System Commander Lite	—	—
そのほか	eWnn Ver.1.0 for Linux、一太郎Ark Technology Preview、BLENDER、SSH1、SSH2、Adaptive Firewall、Atnti-Virus など		—
サポート			
サポート内容	基本サポート（インストールまで） ※1	—	ユーザー登録後インストールまで
サポート方法	電話、FAX、Web（E-Mail）	—	書面、FAX、E-Mail
サポート期間	ユーザー登録後から90日間	—	特に制限なし
サポート件数	3回	—	特に制限なし
備考	※1 代表的なパッケージの設定方法もガイドする「オプションサポート」あり		※1 Wnn6をバンドルした「Vine Linux 1.1 CR with Wnn6」（9800円）を2000本限定発売

Section 3

ディストリビューションライブラリ

前セクションまでは、4つの商用ディストリビューションの紹介や比較を中心に解説してきた。ここで取り上げたディストリビューションは、いずれもユーザー数が多く、完成度も高い。

しかし、それだけがすべてではないはずだ。たとえ、使っているユーザーが少なくても、日本語が利用できなくても、荒削りであっても、前述の4つのディストリビューションに負けないくらいの魅力を持ったディストリビューションはたくさんある。そこで、このセクションでは、そういったディストリビューションも数多く紹介していきたいと思う。

ただ、すべてランダムに紹介してしまうとまとまりがなくなってしまうので、多少強引ながら、次のカテゴリーに分類して紹介していくことにする。

■ポストRed Hat系

ここで紹介するのは、パッケージ管理システムにRPM（Red Hat Package Manager）を採用しており、次世代のRed Hat Linuxの後継の座をねらう、強力なディストリビューションたちである。中には、自称「ベターRed Hat」と名乗っているものもあり、鼻息は荒い。具体的には、Caldera OpenLinux、SuSE Linux、Linux Mandrake、Kondara MNU/Linuxなどがある。

■Debian系

非常に強固なポリシーと強力なパッケージ管理機能を持つDebian GNU/Linuxの流れを組むディストリビューションたちで、パッケージ管理システムにdeb方式を採用している。Debian GNU/Linuxが採用しているバイナリパッケージである.debは、Red Hat Linuxが採用している.rpmに対抗しうるだけの豊富なリソースを持ち、最近サポートするディストリビューションが増え始めている。具体的には、本家のDebian GNU/Linux、Storm Linux、Corel LINUXなどがある。

■Slackware系

世界的なLinuxブームの牽引車の役割を果たしたディストリビューション、Slackwareの流れを組むディストリビューションである。すでにRed Hat Linuxにその役割を譲った感があるのは否めないが、国内にもまだまだ多くのユーザーがいる。けっしてわかりやすいディストリビューションではないが、システム内部を十分理解したいユーザーに勧められることの多い、硬派なディストリビューションである。具体的には、SlackwareやPlamo Linuxなどがある。

■非x86系

Linuxは、開発経緯（Linuxは、元々Linus Torvalds氏がx86プロセッサのアーキテクチャを勉強する目的で始められたものである）や普及台数などの関係から、PC/ATマシンをターゲットにしたディストリビューションが多く、ユーザー数もPC/ATマシンの利用者が圧倒的に多い。しかし、Macintoshに使われているPower PC、AlphaやSPARCなど、他のプラットフォームでもLinuxが動作するように移植され、ディストリビューションが用意されている。具体的には、LinuxPPC、Alpha Linux、Sparc Linuxなどがある。

■個性派ディストリビューション

Linuxは用途や目的に応じて、誰でも自由にディストリビューションを作ることができる。中にはある限定された用途だけでの利用を前提に作られたディストリビューションもある。ここでは、そんな個性的なディストリビューションを紹介する。

具体的には、次のようなものがある。

- **サーバ専用のディストリビューション**
- **Windowsと共存を図るディストリビューション**
- **フロッピーディスクだけで起動するディストリビューション**

ここで紹介しているディストリビューションは、知名度が低いものや、まだまだ実用できないものもあるが、けっして負け組でないことは理解しておいてほしい。どのディストリビューションもいきなりトップに立つ可能性は十分あるのだ。それが、Linuxの面白さであり、Linuxの魅力なのだ。何よりも、Linux自身が数年前まではそういう存在だったのだから…。

ポストRed Hat Linux系のディストリビューション

ここで紹介するのは、RPMを採用していながら、独自の工夫を加えることで、Red Hat Linuxに追いつき、追い越そうという意気込みを持ったディストリビューションである。トップシェアを狙うのに十分な実力と知名度を持っており、今後が期待されるディストリビューションである。

Kondara MNU/Linux
http://www.kondara.org/

Kondara MNU/Linux（以下、Kondara）は、Kondara Projectによって開発されており、12月1日にデジタルファクトリジャパンから発売されたばかりの最新ディストリビューションだ（**画面1**）。

「コンダラ」、「MNU」とはどちらも聞き慣れない言葉だが、「MNU」は、「Mount is Not Umount」の略語、「コンダラ」は、アニメ「巨人の星」の主題歌の一節「思い込んだら」を「重いコンダラ」と聞き違えたことが元になっているそうだ。

開発版の「Rawhide」ベース

Kondaraは、Red Hat社が公開している「Rawhide」をベースに、日本語対応などの拡張を加えたものである。そのため、Red Hat Linuxと互換性があり、Red Hat Linuxに対応したアプリケーションが動作するという。

Kondaraには、Red Hat Linux 6.1で新たに採用された、GUIインストーラ「Anaconda」が日本語化されている（**画面2**）。また、ハードウェアの自動検出機能である「kudzu」が含まれている。これにより、ハードウェアの交換後に起動すると、設定変更を自動的に行ってくれる。

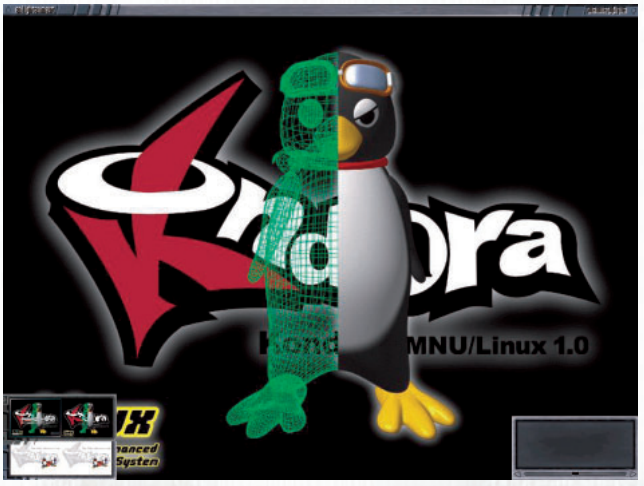
最新コンポーネントを収録

Kondara Projectの方針として、各種のコンポーネントは、極端に不安定なものを除いて、積極的に最新版を取り込んでいる。たとえば、Netscape Communicator／Navigatorは日本語対応を行ったバージョン4.7、Gimpは開発バージョンである1.1.10が採用されている。

腕に覚えのあるユーザー向け

初心者への敷居の高さを下げるためか、最近リリースされたディストリビューションの多くが、GUIのインストーラを装備している。もちろんKondaraもそうだ。

しかし、Kondaraは初心者よりは、ある程度経験を積んだユーザー向けといえるだろう。それは、ソースパッケージからバイナリパッケージを構築する方法の説明が、マニュアルに詳細に載っていることからも明らかだ。海外で新しいソフトウェアが発表されたと聞くと、すぐにソースを取ってきて、日本語化して、RPMパッケージを構築したくなるようなユーザーに最適のディストリビューションである。



画面1　Kondara MNU/Linuxのデスクトップ

画面2　日本語化されたGUIインストーラ「Anaconda」

SuSE Linux

http://www.suse.com/

SuSE Linuxは、ドイツのSuSE社によって開発されているディストリビューションで、最新バージョンは6.2である（ただし、この本が発売されるころにはバージョン6.3がリリースされる予定となっている）。

圧倒的なパッケージ数

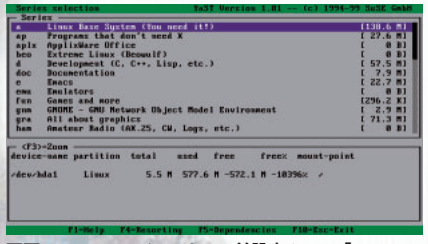
SuSEには、1300ものアプリケーションが収録されたバイナリCD-ROMが6枚も付属している。この中にはReal Player 5.0やVMwareの試用版など、思いつく限りのアプリケーションがバンドルされている。すべてインストールすると、6Gバイト弱ものディスク容量を必要とするなど、重厚長大なタイプのディストリビューションである。

独自色の強いディストリビューション

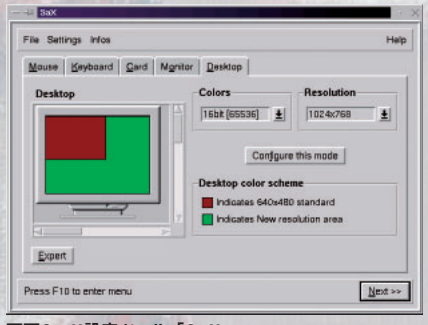
SuSE Linuxは、パッケージ管理システムにRPMを用いているが、Red Hat Linuxと似ている部分はほとんどない（元々は、Slackwareベースで開発されていたようだ）。

たとえばインストールは、まず「Linuxrc」というプログラムによっていくつかのモジュールがロードされ、そのあとで「YaST」と呼ばれるパッケージセットアッププログラムがロードされる（**画面1**）。このYaSTは、インストール後もシステム管理ツールとして利用する。Xの設定ツールも「SaX」（SuSE advanced X-configuration）という独自のツールがバンドルされている（**画面2**）。

また、標準のデスクトップ環境はKDEだが、メニューを見る限り、かなり手が加えられており、標準のKDEに比べてアプリケーションの数がかなり多くなっている（**画面3**）



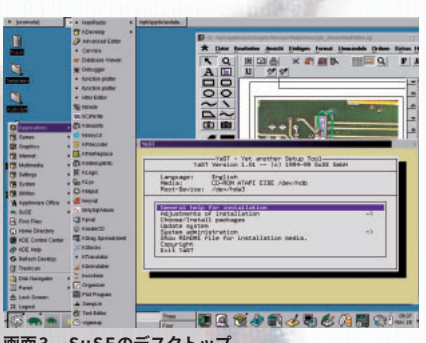
画面1　SuSEのインストーラ兼設定ツール「YaST」



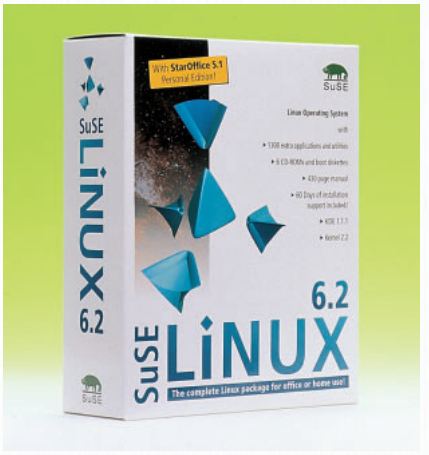
画面2　X設定ツール「SaX」

わかりづらいインストーラ

SuSEの最大の問題点は、インストーラだろう。前述のテキストベースのインストーラ「YaST」は、とてもクセのあるインターフェイスを持っているうえ、一覧表示されている設定項目の中からユーザーが自分で判断してインストールを進めないといけないため、次に何をすればいいのかわからなくなってしまうことが何度もあった。



画面3　SuSEのデスクトップ
標準デスクトップはKDEだが、かなり手を加えているようで、メニューが入りきれないほどたくさんのアプリケーションが並んでいる。



SuSE Linux 6.2
SuSE、Inc（49.95USドル）

なお、次バージョンとなるSuSE Linux 6.3では、このYaSTがバージョンアップして「YaST2」となり、GUIツールとなる。マルチリンガルに対応し、モジュール構造を持たせて、拡張可能となるようだ。

望まれる日本進出

SuSE Linuxは、とてもこの紙幅では伝えきれないほどの高機能ぶりなのだが、残念なことに日本語対応についてはまったく考慮されていない。もちろん、Linuxには変わりないので、自分でコンパイルすれば利用できないことはないだろうが、使いやすいものとはいえない。同様に、430ページにも及ぶ詳細なマニュアルや60日間のインストールサポートも多くのユーザーにはあまりメリットがない。

Caldera OpenLinuxが日本進出へ積極的な動きを見せているなか、SuSE Linuxのも一刻も早く日本市場へ参入し、そのユニークな機能と高い実力を披露してもらいたいものだ。

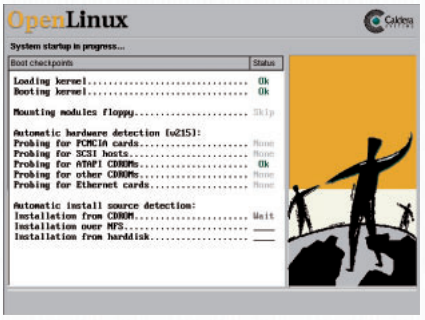
OpenLinux
http://www.openlinux.ne.jp/

OpenLinuxは、米国のCaldera Systemsによって開発・販売・サポートがなされているディストリビューションで、最新バージョンは2.3である。

美しいインストーラ

OpenLinuxで一番の特徴といえるのが、「LIZARD」（**画面1**）と呼ばれる美しいインストーラである。美しいムービーとともに起動し、ほとんどの項目がマウスだけで設定できるようになっている。

また、Windowsとのデュアルブートも考慮し、ディスク管理ツール「Partition Magic」の機能限定版がバンドルされており、GUIでLinux用のパーティションを作ることができる。また、ブートセレクタ「BootMagic」もバンドルされており、初心者でもあまり戸惑わずに、デュアルブート環境を構築できるだろう。



画面1　OpenLinuxのインストーラ「LIZARD」

しっかりと対応された日本語環境

これまで、国内ではあまりメジャーな存在ではなかったが、ネオナジーと業務提携を結び、「OpenLinux 2.3 日本語版」を'99年12月に発売した。

同製品では、インストーラの日本語表示が可能になったほか、「ATOK12 SE for Linux」「Wnn6 Ver.3」といった高精度のかな漢字変換システムがバンドルされている。さらに、デスクトップ環境のKDEも日本語化されており、日本語利用についてはまったく問題はないはずだ。

幅広いユーザー層にアピール

GUIベースのインストーラから初心者向けと思う読者もいるかもしれないが、開発元のCalderaではOpenLinuxをビジネス対応のディストリビューションと位置づけている。精選されたパッケージ構成とサポート・トレーニングなどの環境の充実ぶりは、上級者までの幅広いユーザー層にアピールするだろう。

Linux Mandrake
http://www.linux-mandrake.com/en/

Linux Mandrakeは、フランスのMandrakeSoft社によって開発されているディストリビューションである。見た目の華やかさと、積極的な国際化という2つの特徴を持つ。Red Hat Linuxをベースにしており、「99%コンパチブル」であるとしている。

おしゃれなデスクトップ

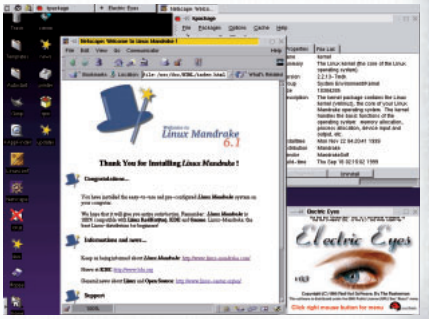
フランスというお国柄のせいか、Linux Mandrakeは、とても「おしゃれ」なディストリビューションである。Red Hat Linuxのインストールの容易さや安定性という特徴を維持しつつ、華やかで初心者にも親しみやすいデスクトップを実現している。同じウィンドウマネージャでもアイコンのデザインや配色が違うせいか、ベースになったRed Hat Linuxなどよりもあか抜けた雰囲気を持つ（**画面2**）。

また、最新バージョンの6.1では、デフォルトのKDEを始めとして、GNOME、Enlightenment、Window Maker、AfterStepなど多くのウィンドウマネージャやデスクトップ環境がバンドルされている。

ユーザーによる日本語版

メーカーの手による日本語版は発売されていないが、木村順一氏が、Kondara Project、Vine Project、日本KDEユーザ会の成果物を用いて日本語化を行ったMandrake-JPを発表している。

Linux Mandrakeは、アメリカでもMacmillan社のブランドで発売されており、シェアを伸ばしているという。デスクトップのデザインは、いくらでもカスタマイズできるものだが、初心者には簡単なことではない。デフォルトで美しいデスクトップを用意したことが、Linux Mandrakeの人気の原因のひとつであろう。



画面2　Linux Mandrakeのデスクトップ

Debian系のディストリビューション

RPM方式をパッケージ管理システムとして採用しているディストリビューションは多い。しかし、このところdeb方式をパッケージ管理システムとして採用することを表明するディストリビューションが増えてきている。ここでは、それらのディストリビューションをDebian系として紹介していく。

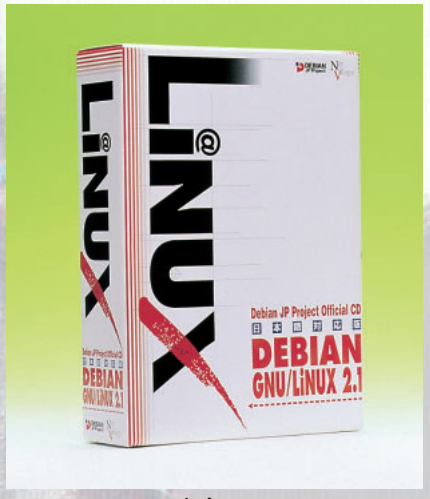
Debian GNU/Linux

http://www.debian.org/

Debian GNU/Linux（以下、Debian）は、ボランティアグループであるDebian Projectによって開発されているディストリビューションで、最新バージョンは2.1（コードネーム「slink」）である。Debianは、一度インストールさえしてしまえば二度と再インストールする必要がないといわれるくらい、非常に強力なパッケージ管理機構を持つ。

ユニークな開発形態

Debianの開発母体であるDebian Projectは、Ian Murdock氏によって1994年に設立されたボランティア組織である。インターネットを活動の拠点として、「社会契約」（Debian Social Contact）と呼ばれる開発方針に従い、メーリングリストやIRCを利用して開発がなされている。また、重要な開発方針は、開発者の同意や選挙などによって決定されるなど、その民主的な開発スタイルは特徴的である。



DEBIAN GNU/LiNUX 2.1
ネットビレッジ（5800円）

■100%フリーソフトウェア

Debianは、DFSG（Debian Free Software Guidelines）にしたがって使用、修正、配布がなされている。DFSGに完全準拠している、100%フリーのパッケージは「main」、DFSGにしたがってはいるが、動作の際DFSGにしたがっていないものを「contrib」、DFSGに準拠していないものを「non-free」と呼んで、区別した形で配布されている。

■Debianの開発スタイル

Debianは、Linuxカーネル同様に「安定版」と「開発版」の2つの系列を持ちながら、開発されている。OSとして正式に公開されたものを安定版（stable）と呼び、開発途中で公開されているものを開発版（unstable）と呼ぶ。unstableは、新規パッケージやバージョンアップしたパッケージを受け付け、リリースを行う時期が近くなると、コードフリーズ（frozen）となり、バグフィックス以外の変更は受け付けなくなる。その後、問題ないことがわかった時点でリリースとなる。一方、コードフリーズされたバージョンのコピーが次のunstableとなり、次バージョンに向けて開発が行われることになる。

■Debian JP Project

日本においても、1996年8月よりDebian JP Projectが発足しており、日本語パッケージや日本語インストーラの開発が進められている。ただ、他のディストリビューションと大きく異なるのは、「Debianの日本語対応版」を作ることが目的ではなく、「Debian本体に日本語パッケージを組み込むこと」を目的としている点である。Debian JP Projectでは、この方針を明確にするため、それまで作成してきたJPパッケージを今後作成しないことを1999年5月に表明している。

強力なパッケージ管理

Debianにおいて最も特徴的なのがパッケージ管理である。Debianのパッケージ管理システムは、インテリジェントなパッケージの更新、依存関係の処理、設定ファイルの保護など強力な管理機構を特徴としている。パッケージ選択のユーザーインターフェイスをdselectが担い、実際のパッケージ管理はdpkgが行っている。

dpkgは、パッケージ間の依存関係を依存（depends）、衝突（conflict）だけでなく、提案（suggests）、置換（replace）、推奨（recommends）といったきめ細かいレベルで管理してい

る。さらに、「仮想パッケージ」という仕組みにより、複数のパッケージに機能ごとに仮想的な名前をつけておき、それらに依存するパッケージは、仮想パッケージ名のほうに依存するよう設定しておくことで、個々のパッケージ管理の手間を軽減している。

また、CD-ROMやハードディスクからだけなく、Anonymous FTPサイトからもパッケージのインストールが行えるので、常に最新状態に保つことが可能である。

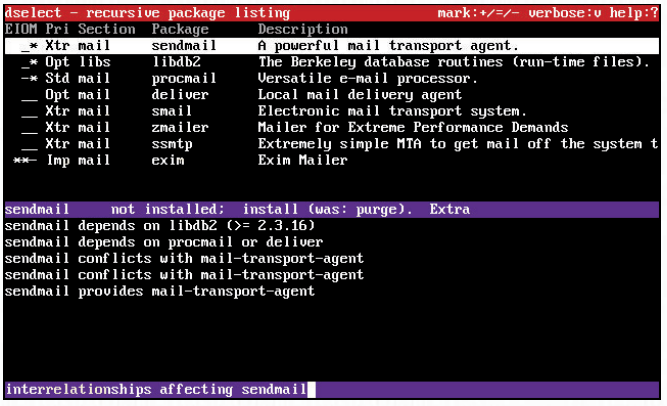
これらの多くの機能を備えているDebianのパッケージ管理は、数あるディストリビューションの中でも群を抜いている。Corel LINUXやStorm Linuxなどの新しいディストリビューションがDebianをベースにしているのも、このパッケージ管理の強力さゆえである。

初心者には敷居が高い

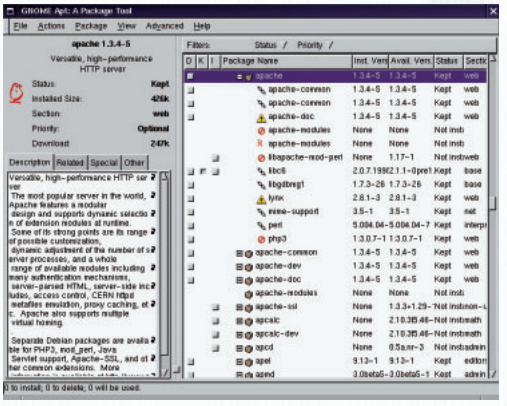
一貫したポリシーと強力なパッケージ管理機構を持つDebianだが、初心者には敷居の高いところも多い。たとえば、インストーラはハードウェアの自動検出を行ってはくれないし、インストール時に、システム設定などをこと細かに質問してくるため、インストールするのには、マシンやネットワークなどの知識が多く求められる。

また、パッケージ管理のフロントエンドであるdselectも、強力なパッケージ管理機構が災いして、一度依存関係で問題が起こると、解消するにはシステムに関する知識が必要になるし、メッセージも英語でわかりづらい（**画面1**）。

もちろん、ある程度経験のあるユーザーには大きな問題ではないのだが、やはり初心者には敷居が高いといわざるをえない。ただし、これらの問題を解決すべく、次期バージョンとなるコードネーム「potato」では、インストーラへの変更など、改善が図られるようである。



画面1　パッケージ管理フロントエンド「dselect」



画面2　次世代のパッケージ管理システム「GNOME-APT」

Column

2つのDebian「推進」プロジェクト

Debian JP Projectが、今後はJPパッケージを作成しないと発表したが、それとは別に、2つのDebian関連のプロジェクトが国内で活動している。もうひとつは「Dice Linux」で、ひとつは「でびまる」である。

「Dice Linux」は、Debian GNU/Linuxと100%の互換性を持つサブセットとして、Project Dice（http://dice.debian.gr.jp/）によって開発されている。開発コンセプトとしては、膨大になったDebianのパッケージを厳選し、コンパクトな構成での配布が挙げられる。また、dselectを用いないでインストール可能な簡単インストーラを搭載し、初心者にもとっつきやすくするようである。さらに、エディタ「dedit」、メーラ「demail」、PPPツール「dppxp」など「Dice Tools」（通称dtools」と呼ばれるツールも収録の予定である。

Project Diceによると、Dice Linuxは新しいディストリビューションではなく、Debian GNU/Linuxの配布および導入への新たなスタイルであるとしている。ただし、現在は開発段階で、まだ具体的な成果物は公開されていない。

一方の「でびまるGNU/Linux」は、Debian GNU/Linux 2.1.r2をベースに、でびまる製作委員会（http://llc.linet.gr.jp/~yochi/debimaru/）によって開発されており、100%の互換性を持つ。開発コンセプトとしては、CD-ROM1枚に収まるようパッケージを厳選し、開発中の独自インストーラによって、インストール手順の簡素化と自動化を図ることが挙げられる。また、dselectを使用しないパッケージ管理ツールの開発や、インストール直後からの日本語利用を可能にするなど、初心者にも配慮がなされている。現在、評価版として「でびまるGNU/Linux 0.1」（コードネーム、「大化」）が、ftp://ftp.linet.gr.jp/pub/Debimaru/で公開されている。

両プロジェクトとも開発コンセプトなどは非常に似ているが、具体的な成果物が提供されているわけではないので、まだはっきりしたことはわからない。今後の活動に期待したいところだ。

Corel LINUX
http://linux.corel.com/

Corel LINUXは、CorelDrawやWordPerfectといったWindowsアプリケーションベンダーとして知られている、カナダのCorel社が開発・販売を行っているディストリビューションである。Debian GNU/Linuxをベースにしており、主にデスクトップ分野への進出をターゲットとしたディストリビューションである。国内では、メディアビジョンが販売を行う。

初心者にもやさしいインストーラ

Corel Linuxは、新しいディストリビューションだけあって、最近の流行ともいえるグラフィカルなインストーラを装備している（**画面1**）。さらに、入力項目を極力減らすことで初心者の負担を軽減している。

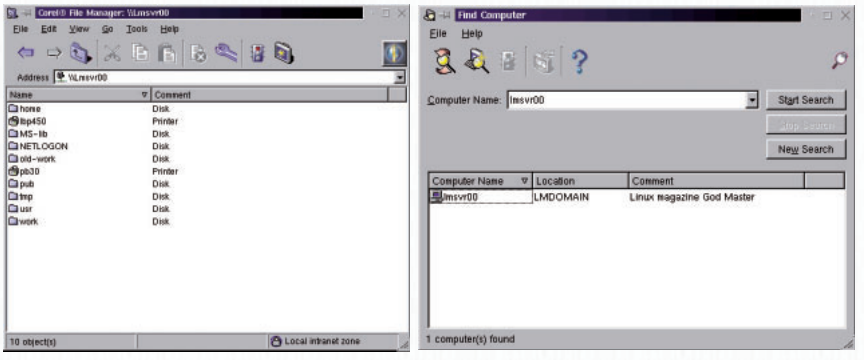
KDEベースの美しいデスクトップ

デスクトップは、テーマセレクタが搭載されたKDE 1.1.2がベースになっており、さまざまなカスタマイズが可能となっている。デスクトップ画面は、KDE標準のものではなく、Corelがいくつか手を加えている（**画面2**）。

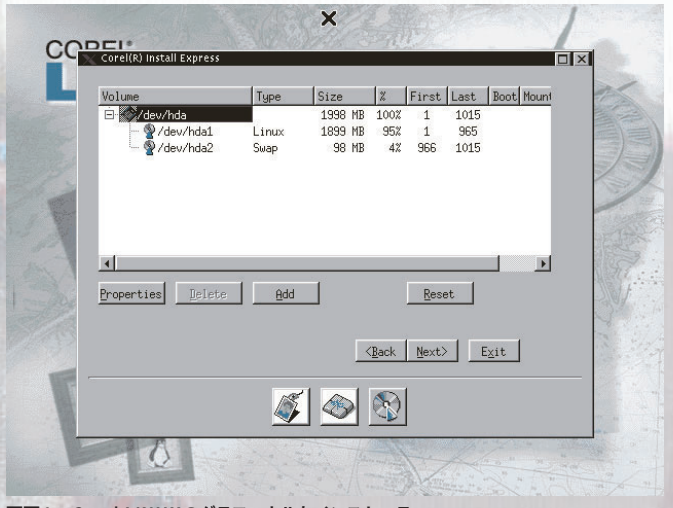
たとえば、ファイルマネージャは、KDE標準のkfmではなく、「Corel File Manager」と呼ばれる、独自のものが搭載されている。このCorel File ManagerにもKfmの特徴である、ローカルリソースとリモートリソースへのシームレスなファイルアクセスが可能となっている。

Windowsとの親和性

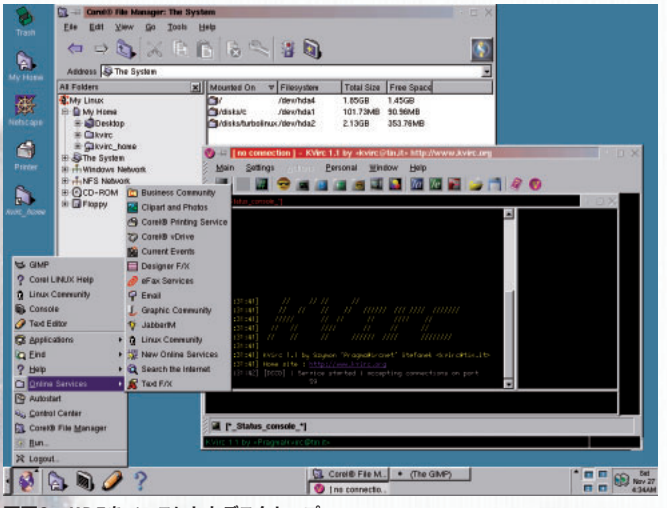
デスクトップ市場への展開を考えているCorel LINUXは、現在のデスクトップ市場を握っているWindowsとの親和性も考慮されている。たとえば、KDEのメニューから［Find］－［Computer］を実行し、ダイアログボックスにコンピュータ名を入力すると、ネットワーク上にあるWindowsマシンを見つけることができる。さらに、その検索結果に表示されているマシン名をクリックすると、そのマシンが公開しているリソースにアクセスすることができる（**画面3**）。これは、smbclientの機能を利用して実現しているのだろう。Windows NTをサーバとして利用している会社などでも、デスクトップマシンとしてCorel LINUXを導入する際の障壁を取り除いてくれるはずだ。



画面3　Windowsマシンの検索とアクセス



画面1　Corel LINUXのグラフィカルなインストーラ



画面2　KDEをベースにしたデスクトップ

Storm Linux
http://www.stormlinux.com/

Storm Linuxは、カナダのStormix Technologies社が開発を行っているディストリビューションで、Debian GNU/Linuxの2.0（slink）をベースにしている。このStorm Linuxは、原稿執筆時点ではまだベータテストの段階で、ベータ4が最新のリリースとなっている。開発コンセプトは、「easy install」「easy to use」で、KDEをデスクトップ環境に採用し、Windowsを使っているLinux初心者を主なターゲットとしているようである。

GUIとCUIを両サポートするインストーラ

Storm Linuxはグラフィカルインストーラを搭載している（**画面1**）。ただ、Storm Linuxのインストーラが他のディストリビューションのインストーラと異なるのは、ひとつのプログラムでテキストベースのものとグラフィカルなものの両方のインターフェイスを選択できることである。

また、Debian GNU/Linuxをベースにしているにもかかわらず、インストーラはハードウェアの自動検出も装備しており、「easy install」を実現している。インストールが終了すれば、100%Debian GNU/Linuxとして動作する。

モジュール化された管理ツール「SAS」

Storm Linuxの最大の目玉といえるのが、このSAS（Storm Administration System）である。SASは、ユーザーから見ると単なるGUIベースの管理ツールのように見えるが、その仕組みはもっと奥が深い。

具体的には、クライアントサイドで動作して、ユーザーからの入力や表示を行うインターフェイス部分である「SAT」（Storm Administration Tool）と、サーバサイドで動作して、プログラムのインストールやコンフィギュレーションを行う「Module」、SATからの送られてくる命令を解析してModuleの管理を行う「MID」（Module Interface Daemon）という3つのモジュール群からなっている。

この仕組みの具体的なメリットとしては、インターフェイス部分をカスタマイズすることで、さまざまな特徴を持ったStorm Linuxを構築できるなど、システムに高い柔軟性を持たせることができる点が挙げられる。

今後の展開

Storm Linuxはまだ開発途上であるため、今後どのような形で発展していくかはっきりとはわからない。当初は、デスクトップ市場への進出をターゲットとしていたようだが、同様にデスクトップ市場への進出を狙うディストリビューションにCorel LINUX（こちらもDebian GNU/Linuxをベースにしている）があるため、方針を変更して「ネットワークに強い」ディストリビューションへ方向を変更する案もあるようだ。こういった戦略を立てられるのも、SASという柔軟なアーキテクチャを持っているからといえよう。日本語対応も含めて、今後に期待したいところだ。

画面1　GUIインストーラ

画面2　SASのインターフェイス「SAT」

Slackware系のディストリビューション

現在では、パッケージ管理システムを備えたディストリビューションが普及しているが、ほんの少し前までは、tarボールを自分で展開し、インストールまで手作業で行うのが一般的だった。その時代に最も多く使われていたのが、Slackwareである。ソフトのインストールは今よりたいへんだったが、自分で手を動かすことで身につけた知識も多かったように思う。「生Linux」を体験したいユーザーに最適なSlackware系のディストリビューションを紹介しよう。

Plamo Linux

http://www.linet.gr.jp/~kojima/Plamo/

Plamo Linuxは、こじまみつひろ氏らがSlackwareをベースに日本語化を行ったディストリビューションである。元になったSlackwareと同様に、パッケージ管理システムを持たないシンプルな構成が特徴で、ユーザー自身によるカスタマイズが容易になっている。「Plamo」という名称も、プラモデルのように手を加えて自分用の環境を作っていくことを表している。

簡単なインストール

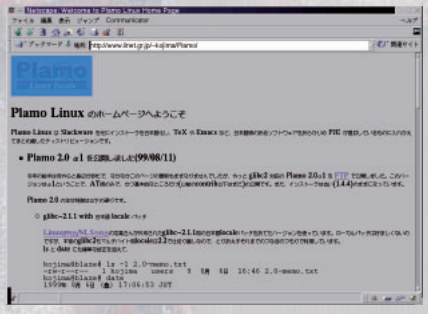
インストールに関連するプログラムのメッセージが日本語化されているため、英語で苦労することなくインストールできる。

また国内でよく利用されているSCSIカード、ネットワークカードのドライバを組み込んだカーネルが用意されているため、対応したハードウェアを使っているマシンなら、特に設定をすることなくそのまま利用可能だ。SCSIカードやネットワークカードは比較的種類が多く、適切なドライバを選ぶのは、初心者には荷が重い作業なので、このような配慮がされているのはありがたい。

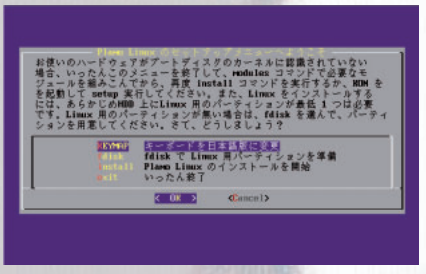
さらに、作者のこじま氏がノートPCをメインに使っているため、PCMCIAやAPMといったノートPC用のコンポーネントにも力を入れている。

NEC PC-9801にも対応

いわゆるDOS/Vマシンが普及する以前は、事実上NECのPC-9801シリーズが国内の標準マシンだった。このPC-9801シリーズへのLinuxの移植は、京大マイコンクラブによって行われ、「Linux/98」という名称で公開されている。Plamo Linuxには、このLinux/98を元にした環境も用意されている。



画面1　Plamo Linux公式Webサイト



画面2　テキストベースのインストーラ

冒頭でも述べたように、Plamo Linuxはパッケージ管理システムを持たないが、代わりに「お勧めパッケージセット」というものが用意されている。パッケージ選択時にこれを選ぶと、作者であるこじま氏が使っているのと同じソフトウェア環境がインストールされるもので、パッケージ選択で悩むことなく、定番ソフトがひととおりそろった日本語対応のLinux環境が得られる。

以上の特徴からわかるように、Plamo Linuxは日本国内での使用に特化したディストリビューションだ。同時にLinuxに不慣れな初心者への配慮がなされており、初めてインストールする場合でも、それほど戸惑うことはないだろう。

待たれるメジャーバージョンアップ

Plamo Linuxの安定版は、1.4.4が最新である。「最新」と書いたが、カーネルが2.0.36、Cライブラリがlibc5という組み合わせは、今となっては少々古いものといわざるをえない。

だが、'99年8月に、Plamo 2.0のアルファ版（ベータ版よりさらに初期の開発版）が発表されている。こちらはカーネルが2.2.10、Cライブラリが-glibc 2.1.1と新しくなっており、デスクトップ環境も日本語化したGNOMEとKDEが用意されているということだ。

シンプルで扱いやすいといった特徴を保ったまま、最新のコンポーネントをそろえたバージョン2.0の正式リリースが待ち望まれる。

Slackware

http://www.slackware.com/

Slackwareは、Patrik Volkerding氏らによって作られているディストリビューションである。'92年4月に最初のバージョンがリリースされており、今回の特集で紹介する中では最も歴史の長いディストリビューションである。また、日本国内でのLinux普及にも貢献したディストリビューションだといえる。当時、Linuxで日本語環境を使用するためには、まず英語版をインストールしたあと、JE（Japanese Extension）という追加パッケージをインストールする必要があった。JEの作者である真鍋氏がSlackwareを使っていたため、「JEを使うにはSlackwareが安心」と言われていた。このころからLinuxを使っていたユーザーには、今でもSlackwareを使い続けている人が多いようだ。

最新版の特徴

Slackwareは、バージョン4.0で初めてカーネル2.2系列を採用したが、そのリリースは他のディストリビューションよりも遅めだった。また、RPM系のディストリビューションが普及が進んだこともあり、このままフェードアウトしていくのかとも思われていたが、本特集に合わせたかのようにバージョン7.0が発表され、ファンを安心させた。

なおバージョンナンバーが4.0から一気に7.0になっているが、これは数字が小さいと古くさいものだと勘違いされるので、それを防ぐためとのことだ。他のディストリビューションを追い越したバージョンナンバーにふさわしく、各コンポーネントに最新のものを採用している。例を挙げれば、カーネルは安定版の2.2.13を、また2大デスクトップ環境のKDEは1.1.2を、GNOMEはOctober GNOMEと呼ばれる1.0.53をそれぞれ用いている。

多くのディストリビューションが、グラフィカルなインストーラを採用するなどして、初めて使うユーザーの敷居を下げるような方向に変化している。だがSlackwareは、最新の7.0でもテキストベースのインストーラを採用している。初心者への配慮よりは、Slackwareを使い続けているユーザーに、以前と同じ使い勝手で、最新バージョンを提供することを目指しているのであろう。

ウィザードを目指す人にお勧め

率直なところ、まったくのコンピュータ初心者にSlackwareは勧めにくいが、以下の3項目のどれかに当てはまる方なら、きっと気に入るだろう。

- **旧バージョンのユーザー**
- **UNIX経験の豊富な人**
- **Linuxを学びたい人**

■旧バージョンのユーザー

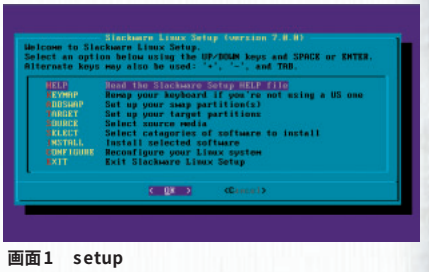
バージョン7.0は、カーネルをはじめとする各種コンポーネントに最新のものをそろえているが、各種設定ファイルの構成や置き場所は、以前と大きくは変わらない。そのため、今までの経験や知識をそのまま活かしつつ、最新の環境を体験できる。他のディストリビューションがバージョンアップするのを横目に、旧バージョンを使い続けていたユーザーは、これを機にバージョンアップしてみてはどうだろう。

■UNIX経験の豊富な人

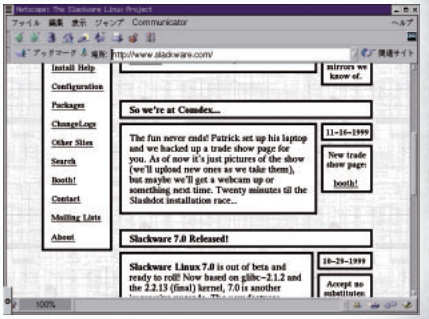
以前から仕事でUNIXマシンを使っていて慣れている人が、Linuxを使い始めるときにもSlackwareは適している。それはSlackwareが、UNIXの流儀を引き継いだディストリビューションだからだ。それはたとえば、何か設定を変更する際には、GUIツールでなくエディタでテキストの設定ファイルを書き換えるといったことだ。Slackwareなら、UNIXでの経験が違和感なく活かせるはずだ。

■Linuxを学びたい人

Linux上で何かをするというよりは、いろいろいじり回してLinux、またはUNIXそのものに関する知識を増やしたい人には、Slackwareが適している。それはSlackwareがシンプルで手を加えやすいからだ。Slackwareをベースに、自分なりのシステムを構築して、ウィザードを目指そう。



画面1　setup



画面2　Slackware公式サイト

非x86系のディストリビューション

Linuxは、対応するCPUの種類がきわめて多い。現存するOSの中では、トップクラスだ。その数多いCPUの中から、利用者の多い（x86とは比べられないが）Alpha、PowerPC、そしてSPARC用Linuxの現状を紹介する。

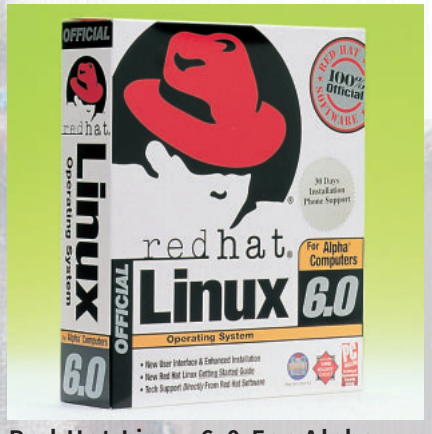
Alpha Linux
http://www.alphalinux.org/

Alpha AXPプロセッサ（以下Alpha）は、DEC（現コンパック）が開発した世界最速のCPUである。初めから64ビットのRISC（コラム参照）CPUとして開発されている。

一般的にはそれほどなじみのないCPUだが、Debian GNU/Linux、Red Hat Linuxなど多くのディストリビューションが、Alpha版Linuxを用意している。編集部で確認したディストリビューション（Red Hat Linux 6.0とKondara MNU/Linux 1.0）では、カーネル、Cライブラリなどの主要コンポーネントにx86版とほぼ同等のバージョンが採用されていた。

64ビット対応済み

特筆すべきは、カーネルも含めたLinuxシステム全体が、すでに64ビット化されていることだ。コンパイラも64ビット用のコードを生成するため、CPUの性能をフルに活かすことが可能だ。



Red Hat Linux 6.0 For Alpha
Red Had Software（79.95USドル）

一般的なx86マシンと比較してAlphaマシンが優れている点は、64ビットの圧倒的に広いアドレス空間が利用できることと、浮動小数点演算が非常に高速なことの2点だ。Alpha版Linuxを用いることで、これらの利点を活かした安定したシステムを、低コストで実現できる。

大規模なデータベースや、企業の基幹サーバといった用途では、32ビットのアドレス空間（2の32乗＝約40億）は、すでに十分なものとはいえない。その点、Alphaの64ビットアドレス空間（2の64乗＝40億の40億倍）では、事実上無制限ともいえるデータ量を扱うことができるため、有用である。またLinuxを用いることで、優れたコストパフォーマンスと高い安定性を両立させることが可能だ。さらにコンパックのTru64 Unix用バイナリの動作も可能（ライセンスが必要）なため、Tru64 Unix用の商用アプリケーションも扱える。

タイタニックを沈めた計算力

また、科学技術計算やシミュレーション、特殊映像効果などの分野では、大量の浮動小数点演算を実行する。

Alpha版Linuxは、これらの用途向けに、高い演算性能を低コストで提供する。またLinuxは、ソースが公開されているため、必要に応じてチューニングも行えるのも、有利な点だ。

さらに、複数のマシンによる並列計算が必要な場合にも、Linuxを搭載したAlphaマシンが有効だ。x86 PCと比べるとAlphaマシンは高価だが、その分性能も高いため、同じ計算能力を得るために必要な台数が少なくてすむ。結局、全体のコストはAlphaマシンを用いたほうが安くなる。たとえば、'97年に公開された映画「タイタニック」の映像処理には、100台以上の、Linuxを搭載したAlphaマシンが使われているそうだ。豪華客船を沈めたのは、LinuxとAlphaだったのだ。

数値計算でRISCの性能を発揮させるためには、アーキテクチャに合わせて高度な最適化が行えるコンパイラが必要になる。Alpha版Linux用のFortranとCコンパイラ（ベータ版）が、コンパックより提供されており、Linux環境からでもAlphaの性能をフルに活かすことが可能だ。

Linux PPC
http://www.linuxppc.org/

PowerPCを搭載したMacintosh（PowerMac）用Linuxには、MkLinux（Micro kernel）とLinuxPPC（Mk Linuxと対比する意味で、Mono-Linuxともいう。Monoはモノリシック、一枚岩の意味）に大別される。

MkLinuxは、マイクロカーネル構造のOSであるMach上で、Linuxサーバを動作させるものだ。AppleとOpen Software Foundation（現The Open

Group）が開発支援を行っていた。'98年にAppleが手を引いた後も、コミュニティ主導で開発が続けられている。

LinuxPPCは、x86版と同様のモノリシックカーネルを採用している。MkLinuxよりもパフォーマンスが高く、また対応するデバイスや機種も多いため、PowerMac用Linuxの主流になっている。以下の解説でも、主にLinuxPPCを取り上げている。

PowerPC版Linuxを扱っているディストリビューターにはLinuxPPC社、TurboLinux社などがある。またDebianプロジェクトでも、PowerPC版が作られているようだ。

ほぼ最新のソフト環境

いくつかのディストリビューションの主要コンポーネントのバージョンを確認してみたところ、LinuxPPCの最新版、1999 Q3は、カーネルが2.2.6、Cライブラリが2.1.1、Debian GNU/Linuxのpotato（開発版）は、カーネルが2.2.12、Cライブラリが2.1.2だった。もちろんこれらは、後から最新版に入れ替えることができるが、始めから最新に近いものが使われているのは、何かと便利だ。

MacOSとLinuxPPCのデュアルブートは、BootXというプログラムを用いることで、簡単に行える。BootXの実体は、MacOS用のアプリケーションであり、MacOS／LinuxPPCの切り替え以外にも、x86版Linuxではlilo.confに記述するような起動パラメータを指定できるもので、インストール時にも用いる。

USBデバイスにも対応

ベストセラーとなったiMac以降の機種は、レガシーデバイスを大胆に廃止しており、マウス、キーボードの入力デバイスもUSBで接続するようになっている。PowerMac用Linuxカーネルは、すでにUSB機器（入力デバイス）に対応しており、iMac、青白ボディのPower Mac G3、PowerBook G3でLinuxPPCを使うことができる。今のところ、最新モデルであるPowerMac G4、iBook、iMacDVには未対応だ。しかし、LinuxPPCコミュニティは非常に活発で、本稿執筆時には、上記の最新モデルすべてでカーネルの起動に成功したという情報が得られている。正式対応が発表されるのも、そう遠いことではなかろう。

UltraLinux
http://www.ultralinux.org/

SPARCは、Sunが設計したRISC CPUであり、主に同社のワークステーションやサーバに用いられている。UltraLinuxは、SPARC版Linuxのことであり、32ビット版と64ビット版の2種類が存在する。Sun4c／m／d／uの各シリーズのマシンが動作する。

SPARC対応のディストリビューションとしては、Red Hat Linux 6.0、Caldera OpenLinux 2.2がある。いずれもx86版の最新版よりも1世代だけ前の製品であり、極端に古いわけではない。またDebian Projectからも安定版（slink）／開発版（potato）の両方がリリースされており、どちらもx86版とほぼ同じバージョンのコンポーネントを用いている。同じソースコードからビルドされているので、別に不思議なことではない。

純正OSに匹敵する安定性

一般に、Sunのワークステーションやサーバには、Solaris／SunOSが用いられている。純正のOSがあるにもかかわらずLinuxの需要があるのは、LinuxのSolarisに匹敵するほどの安定性に加えて、オープンソースの自由な開発環境が得られるからであろう。また、Linuxは動作に必要なハードウェアリソースがそれほど多くないため、少々時代遅れのマシンの再利用という用途もあるだろう。

Column

RISC

RISCは、Reduced Instruction Set Computerのことで、単純な命令だけを用いて、設計されたCPUのことである。それ以前のCPUは、複雑で長い命令を含んでいた。だが、それらの命令は使える状況が限定されていて、実際にはほとんど利用されていないことがわかったため、RISCが生まれた。複雑な命令を持たないので、CPUの内部構造が簡潔になる。その結果、クロックを上げやすくなり、高性能が得られる。今回x86以外のアーキテクチャとして紹介したAlpha、PowerPC、SPARCは、いずれもRISCだ。

RISCの対語はCISC、Complex Instruction Set Computerだ。インテルのx86シリーズプロセッサ、特にPentiumまではCISCの代表的な例だ。同じx86でもPentiumPro以降のCPUや、AMDのK6やAthlonシリーズは、x86命令を内部でRISC的な命令に変換して処理しているため、CISCといいきっていいのか微妙な存在だ。

逆にRISCにも、用途に合わせた複雑な命令を加えることが多くなっている。たとえば、PowerPC G4には、Velocity Engine（AltiVec）という新しい命令セットが追加されている。現在では、RISC／CISCの境界は曖昧になりつつある。

個性派ディストリビューション

ディストリビューションの中には、ある機能だけに特化させたものや、特別な仕組みを持たせたものなど、個性的なものも多い。ここでは、そんなディストリビューションをいくつか紹介しよう。

サーバ用ディストリビューション

Linuxは、その堅牢性などからサーバ用途で利用されることが多い。しかし、サーバ用途での利用となると、デスクトップで利用していたときよりもセキュリティに気を配らなければならないし、設定も面倒になる。そのような状況を踏まえて、サーバ用途に限定したディストリビューションがいくつかリリースされている。

■TurboLinux Server 日本語版 6.0

まず、12月15日より販売開始される、ターボリナックスジャパンの「TurboLinux Server 日本語版 6.0」(2万9800円)、「TurboLinux Server日本語版 6.0 SOHO Edition」(1万9800円)がある。

両製品は、SMPのパフォーマンスが向上するようにカーネルがチューニングされており、サーバメモリを最大で4Gバイトまで拡張可能にしているうえ、プロセス数やセッション数も拡張されている。また、用途別のインストールが可能なうえ、SOHO Editionでは「HDE LinuxController for TurboLinux」というWebブラウザからサーバ設定を行えるツールをバンドルして、利便性を図っている。

サポートは、対応ハードウェアの認識および設定、インストールまでの範囲に限定した、WebやE-mailによる90日間3件までのサポートが付属している。

TurboLinux Server 日本語版6.0
ターボリナックスジャパン(2万9800円)

■日本語 redhat Linux 6.0 Server Edition

TurboLinux Serverと同様のコンセプトを持つ製品として、レーザーファイブの「日本語 redhat Linux 6.0 Server Edition」(4万9800円)がある。こちらは、Red Hat Linux 6.0(英語版)をベースに、独自の日本語インストーラから、2枚差しのネットワークカード設定、IPマスカレード設定、Samba設定、メールサーバ設定、DNS設定、DHCP設定、UPS設定などが行えるようになっている。

画面1　プロサーバ for Linxuのシステム設定メニュー

サポートは、インストールはもとより、各サーバ機能の設定や運用方法までを対象とした、電話、FAX、Web(E-mail)による180日間に3回までの質問が可能な基本サポートが付属している。

■プロサーバ for Linux Ver.2.0

「プロサーバ for Linux Ver.2.0」(1万4800円)は、Debian GNU/Linuxをベースにギデオンが開発し、富士マグネディスク(http://www.fujifilm.co.jp/fmd/linux/lintop.html)が販売している、サーバ用ディストリビューションである。前述の2製品よりもさらに設定項目を減らし、GUIベースでサーバ設定が行える製品である(**画面1**)。

サポートは、販売元の富士マグネディスクがE-mailによる質問を受け付けており、2回までが無料となっている(3回目以降は有料)。

なお、E-mailに感染したウィルスをサーバ上で検出するデータフェロー社製「FSAV Linux」を搭載した「プロサーバ with アンチウイルス」(初年度のみ12万円〜)も、販売されている。

サーバは、Linuxでは最も重要な使用用途であり、またLinuxは元々サーバに利用するのに十分なパフォーマンスを持ったOSである。今後、このようにサーバ用途に特化したディストリビューションは増えてくるだろう。

Windows共存ディストリビューション

これは、Windows上にLinux環境を構築し、1台のマシンでWindowsとの共存を図るディストリビューションである。設定がより簡単であることや危険なパーティション操作が不要であることなどがメリットとして挙げられる。

■Linux MLD4

Linux MLD4（以下、MLD4）は、メディアラボ（http://www.mlb.co.jp/）によって開発され、12月10日から販売されるディストリビューションである。パッケージ管理システムにはRPMを採用しているので、rpmコマンドで新しいパッケージを追加することも可能である。

MLD4の特徴は、Windows 95/98のファイルシステムであるFAT16/32上に、MLD4用の大きなファイルを1つ作成し、そのなかにext2ファイルシステムを構築することである。つまり、Windowsのなかに Linux環境を構築するのである。インストールは、Auto Runによるインストーラから、コンパクト版（必要なディスク容量は250Mバイト以上）と標準版（必要なディスク容量は650Mバイト以上）のいずれかを選択して、インストールを開始するだけである。ハードウェア情報やネットワーク情報などは、インストーラがWindows上のレジストリから自動的に収集してLinuxに設定してくれるため、MLD4のインストール中は、設定項目は一切ない。また、Windowsのファイルシステム上にインストールするため、パーティション操作も不要である。

インストール後は、インストール中作成したフロッピーディスクか、WindowsのMS-DOSモードなどから起動する。その仕組みは、LOADLIN（DOS上で動作するLinuxブートローダ）を実行し、FAT上にあるext2領域をループデバイスを使ってマウントしている。

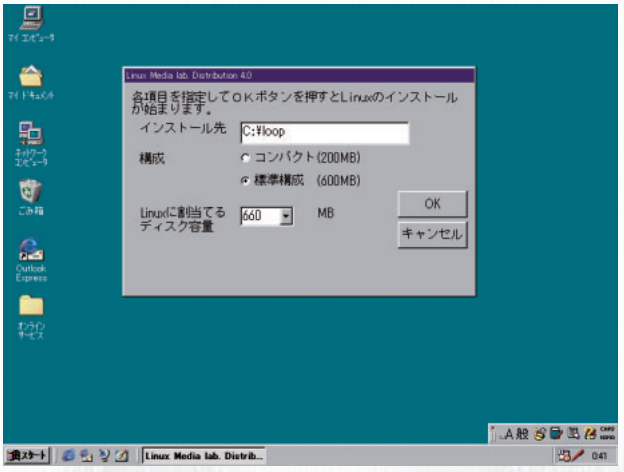
Windows上のLinuxというと、どうしても機能限定版といったイメージを連想すると思うが、標準版をインストールすればWnn6、XEmacs、Netscapeといった定番のアプリケーションはもとより、GNOMEやKDEといったデスクトップ環境も利用することができる。

Linux MLD4
メディアラボ（8800円）

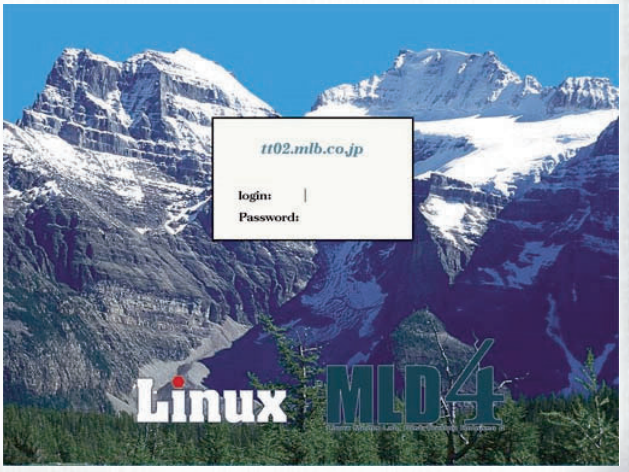
■Phat LinuxとWinLinux 2000

海外にもMLD4と同様の仕組みを持つディストリビューションがいくつかある。ひとつは、高校生のCameron Cooper氏らが開発を行っているPhat Linux（http://www.phatlinux.com/）。Red Hat Linux 6.0をベースにしており、パッケージ管理システムもRPMを用いている。最新バージョンの3.2にはKDEがプレインストールされている。

もうひとつは、米JRCP社が開発しているWinLinux 2000（http://www.winlinux.net/）。こちらもRPMを用いているようだが詳細は不明である。現在は最終ベータ版が公開されている。



画面1　Windows上で動作するMLD4インストーラ



画面2　MLD4のグラフィカルログイン

フロッピーLinux&etc.

これはフロッピーディスクだけで動作するディストリビューションで、RAMディスクを利用するものが多い。具体的なメリットとしては、環境を持ち運べるので、どこでも同じように作業ができるということや、ファイアウォールなどを簡単に構築できるということなどが挙げられる。こういったアプローチは需要があるのか、かなり数が多いので、ここではいくつか特徴のあるものだけを紹介する。

■hal91 Floppy Linux

hal91 Floppy Linux（http://home.sol.no/~okolaas/hal91.html）は、Oyvind Kolas氏によって開発されているディストリビューションで、Red Hat 4.2をベースとしている。最新バージョンは0.2.0で、ポータブルな利用やレスキューディスクを主な用途としている。基本システムは、フロッピーディスク1枚で動作するが、追加ディスクを利用すれば、CDプレーヤ、Lynxやncftpなどが利用できるようになる。

■LRP

LRP（Linux Router Project：http://www.linuxrouter.org/）は、Dave Cinega氏が中心になって開発されているディストリビューションで、Debian GNU/Linux 2.0をベースとしている。最新バージョンは2.9.4で、そのプロジェクト名からもわかるとおり、フロッピーディスク1枚で動作するルータを主な目的としており、ファイアウォール、IPマスカレードなどの機能が標準で装備されている。

■Tomsrtbt

Tomsrtbt（Tom's Root/Boot：http://www.toms.net/rb/）は、Thomas A. Oehser氏によって開発されているディストリビューションで、1.722Mバイトフォーマットのフロッピーディスク1枚を利用する。最新バージョンは1.7.185で、レスキューディスクを主な用途としている。PCカードやSMB、VFATを含む12種類にも及ぶファイルシステムのサポートなどが、レスキューディスクとしての性格をよく表しているといえよう。また、このTomsrtbtのサイトには、他のフロッピーLinuxへのリンクが多くはられているので、興味のある方はのぞいてみるといいだろう。

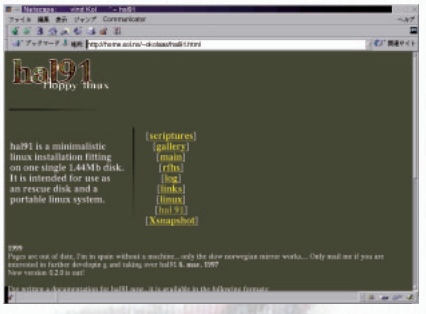
●

さて、このSection 3では、さまざまなLinuxディストリビューションを紹介してきたが、まだまだ紹介しきれないほどたくさんのディストリビューションが世界中で作られている。

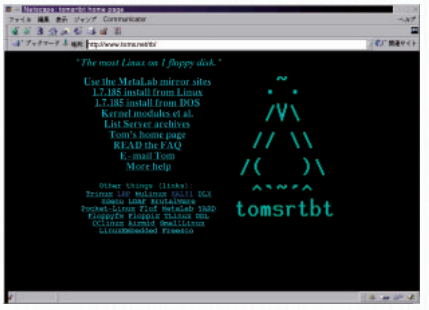
たとえば、Stampede GNU/Linux（http://www.stampede.org/）は、Pentium用に最適化されたgccである「PGCC」でコンパイルすることで、Pentiumで最適なパフォーマンスを発揮するバイナリを用いているディストリビューションである。

ほかにも、ディストリビューションという具体的な形にはいたっていないものの、クラスタリングに特化した試みである Extreme Linux（http://www.extremelinux.org）やMobileGear上でLinuxを動かそうとするPocketLinuxプロジェクト（http://www.asahi-net.or.jp/~bg3k-ysd/pocketlinux/）などがある。

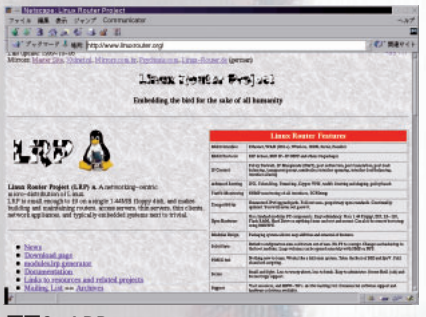
これらのディストリビューションも今後少しずつ紹介していきたいと思う。

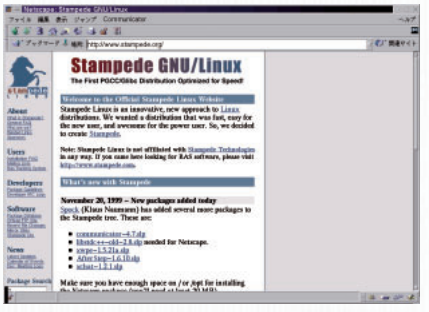


画面1　hal91 Floppy Linux



画面3　Tomsrtbt


画面2　LRP



画面4　Stampede GNU/Linux

ディストリビューションの向かう先は？

1999年は「Linuxの年」と呼ぶのにふさわしいくらい、Linuxが注目され、いろいろな動きがあった1年だった。そして、そんな活況を背景に、Linuxディストリビューションは爆発的に増加した。個人、ベンダ、ボランティアグループが各々の趣向を凝らし、ディストリビューションを競うように公開している。開発者のLinus Torvaldsはもちろん、もはや世界の誰もすべてを把握している人はいないに違いない。

ディストリビューションの共通項

すっかり混沌とした感のある昨今のディストリビューション事情だが、今年リリースされたディストリビューションを見ると、その中に共通の傾向をいくつか見いだすことができる。それは、次のようなものだ。

■初心者にやさしいインストーラ

ひと頃と異なり、昨今のディストリビューションのインストーラは、初心者の取り込みへの努力がくみ取れる。グラフィカルインストーラの採用やハードウェアの自動検出、入力項目の削減など、いずれも初心者ユーザーに向けられた機能が多くのディストリビューションに搭載され始めている。初めてインストールする初心者でも、ハードウェア的なトラブルにあわない限り、もはやインストールで行き詰まってしまうことはないように思う。そういう意味では、Linuxのインストーラは、Windowsと遜色のないレベルへと近づきつつある。

■ユーザーインターフェイスの統一化

次に、ユーザーインターフェイス（ ウィンドウマネージャ）の標準化が挙げられるだろう。ほとんどのディストリビューションが、デスクトップ環境としてGNOMEやKDEを採用している。そして、特に海外ではその標準のデスクトップ環境としてKDEを採用するディストリビューションが増えている。

■日本語入力環境の充実

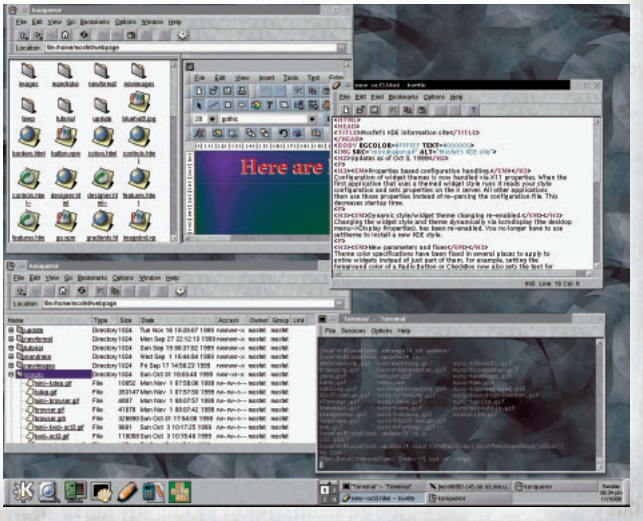
これは国内だけの傾向だが、ATOK12 SE for LinuxやWnn6 Ver.3といった商用かな漢字変換システムのバンドルも国内の商用ディストリビューションで共通することだ。これは流行と見る向きもあるだろうが、むしろ、Windowsを経由してきたユーザーには、旧来のUNIX環境で利用されていたCannaやWnn4といった、かな漢字変換システムの変換精度に満足できなかったと見るべきだろう。

2000年のLinuxは？

これらの傾向が意味するものは何だろうか？　誤解を恐れずに言うなら、こういった一連の傾向は、「フリー（無料の）Windows」を目指した動きだったといえるのではないだろうか。一度浸透したWindowsパラダイムを「否定」するのではなく、「取り込もう」としているのだ。

しかし、Linus Torvaldsが「今後、Linuxカーネルに大きな変更はない」と語っているとおり、Linuxは開発期を過ぎて、すでに成熟期を迎えつつある。もう環境が提供されるのを待つ必要はない。いよいよ、Linuxならではという新しいパラダイムを切り開くときがやって来ているのだ。

2000年は、しばらく動きがなかったWindows 2000もようやくリリースされる。市場にもさまざまな変化があるだろう。さて、Linuxにとって2000年はどのような年になるのだろうか？



画面　KDE 2.0のスナップショット
マルチバイト対応のQt2.0をベースに作られた次世代のKDE 2.0。CORBAベースのオフィススイート「KOffice」のバンドルを始めとして、数多くの機能を持つ。次世代のLinuxの標準インターフェイスとなるのか？

Linuxマシン バリバリチューニング

もっと速く！　もっと強力に！
それぞれ3万円以下で実現できるLinuxパワーアップ！

photo : Junko Kitade(Dee)

高速化と多機能化のどちらを目指す？

Windowsに比べると、LinuxはOS自体が「軽い」構造なので、CPUをはじめとするハードウェアの性能が低めでも快適に使える、とよくいわれる。しかしユーザーが直接触れるクライアントPCに関しては、GNOMEやKDEなど「重い」ソフトウェアが標準で添付されるようになってきた。たとえLinuxが「軽く」ても、その上で動くソフトウェアが「重け」れば、全体としてハードウェアへの負担が重いことに変わりはない。

この記事は、最近ちょっと遅く感じられるクライアントのLinuxマシンを、PCパーツの交換・追加で高速化しよう、というのが目的の1つだ。ただ、Linuxに対する不満という点では、ハードウェアサポートの弱さから実現できる機能がWindowsより少ないことのほうが、実行速度なんかよりずっと大きいのではないだろうか？　そこで、Linuxマシンの高速化だけではなく多機能化にも挑戦してみた。パフォーマンスチューニングには興味がないという向きは、Linuxマシンでテレビを見たり、3Dゲームを楽しんだりしてみるのはいかがだろうか？

損をしないPCパーツの買い方

PCをチューニングするなら、PCパーツの購入は欠かせない作業であり、またお楽しみの一部でもある。しかしPCパーツでは、製品自体は同じでも外観や付属品の構成、保証規定などが異なる複数のパッケージが店頭に並んでいることが珍しくない。どれを買えばよいのか、迷った経験を持つ人も多いだろう。そこでパッケージごとの大雑把な傾向を以下で説明しよう。

■化粧箱入りの日本語版パッケージ

メーカーや輸入代理店が1年程度の無償保証とサポートをつけて販売しているパッケージだ。万一の故障にはショップではなくメーカー（か代理店）が対応する（初期不良だけショップが対応することもある）。海外メーカー製品の場合、マニュアルぐらいは代理店が日本語化する。しかしドライバを含むソフトウェアは、日本語化にコストがかかるため、英語版でほぼ動作するならそのまま出荷するところと、あくまで日本語化にこだわるところがあって面白い。

ただし、もともと保証やサポートの対象でないLinuxの場合、メリットはマニュアルが日本語で読みやすいのと、ハードウェア単体の保証がしっかりしているという2点に限られる。価格も高めなので、どちらかといえばLinuxよりWindowsでの使用頻度が高い用途に向いている。

■バルク品

PCメーカーなどOEM向けに納入されるべき簡素なパッケージの製品が、さまざまな（あやしい）流通ルートを経て、そのまま販売店の店頭に並んでしまったもの。付属品は薄いマニュアルとドライバCDだけという最低限のパッケージであることが多い。たいていは化粧箱入りより安価なので、Linuxでの使用頻度が高いなら、割り切って買うのも悪くない。ただし基本的にメーカーではなくショップが品質を保証することは憶えておきたい。つまり故障したらメーカーサポートは受けられず、ショップが対応する場合がほとんどだ。そのせいか、バルク品の故障は初期不良交換期間が過ぎたら受け付けないショップもある。またメーカー保証がないということは、保証付きの化粧箱入り製品とバルク品では仕様が異なる可能性もあるということだ。

こうしたリスクがあることから、今回掲載したPCパーツはなるべく化粧箱入りの日本語版を選んでいる。コスト最優先でバルク品を選べばもっと安く済んだはずだ。。

■化粧箱入りの英語版パッケージ

中身の品質はバルク品より信頼できるが、ショップが保証する点はバルク品と同じ。アメリカでは売れていても、日本では売れそうにない製品が、英語版のまま販売されることがある（Voodoo3 2000/3000 PCIはその好例だ）。Linux向けならWindows用ドライバが英語版でも関係ないので、これもターゲットに入る。ただ価格が高い場合も多いので注意したい。

■ジャンク品

古いPCから取り出した製品あるいは古いPCそのものを指す。保証やサポートはまったくないが、ときたま格安の新品に出会うこともあり、買う楽しみがある。Linuxで動作する古いパーツが必要なら、これもターゲットに入るだろう。

本記事に関してのご注意

本記事の中でテストした製品に関しては、特記しないかぎり、どのメーカーもLinuxを正式にサポートしておらず、動作保証もしておりません。本記事に関して各メーカーに問い合わせることはご遠慮ください。また本記事で動作を検証した結果は、メーカーおよび編集部で保証するものではありません。ご了承ください。

テスト環境について

各ハードウェアの動作を確認したのは、Intel製440BXチップセットを搭載したマザーボードに、Celeron-300MHzとPC100 SDRAM 128Mバイトのメモリを組み合わせた自作PCである。拡張カードやIDEデバイスについては、Intel以外のメーカーによる互換チップセットの環境で正しく動作しない可能性もあるので注意していただきたい（最近は、こうした互換性問題も少なくなってきたが）。

本記事のように、最新ハードウェアをLinuxでテストする場合、単にドライバを組み込むだけで動くのは幸運なほうで、カーネルのアップグレードを強いられることが珍しくない。つまり、なるべく新しいカーネルを採用したディストリビューションのほうが、何かと面倒が少ない。もちろん新しいディストリビューションのほうがハードウェアのサポート範囲も広いので好ましい。

そこで今回は、最新のRed Hat Linux 6.1をベースに日本語化された公認Red Hat Linux 6.1日本語版（以下Red Hat Linux 6.1J）にてテストを行った。本誌付録CD-ROMにも収録されているので、実際に試してみるとよいだろう。カーネルのバージョンは2.2.12と、原稿執筆時点で最新の2.2.13よりわずかに古いだけだ。このように、新しすぎたせいか、日本語が正しく表示されないなどの不具合が散見されたのが難点ではある。

今回は、カーネルのリコンフィグ／リコンパイルを要求するハードウェアが非常に多かった。カーネルのリコンフィグ／リコンパイルに慣れていない場合は、実際に試す前に、本誌第2号174～180ページの記事（http://www.ascii.co.jp/linux/allascii/linuxmag/articles/no_02/p174-180.pdf)などの資料を参照しながら、リコンパイルに慣れておいたほうが無難だろう。

Column

Linux関連製品を購入できるショップ

インターネット通販でLinux関連製品を取り扱っているショップは、日本に限らず世界中にあるので見つけるだけなら簡単だ。しかし、実際に店頭でLinux関連製品を手に取って確かめたり、店員さんに直接質問したりできるショップとなると、まだまだ少ないのが現状だ。ここでは秋葉原でLinux対応製品を店頭に並べている2つのショップを紹介しよう。どちらも特色あふれる面白いショップだ。通販も受け付けている。

■ぷらっとホーム

秋葉原でも有数の老舗である同社は、UNIX製品を幅広く取り扱うことで知られている。Linuxをはじめとする PC UNIXだけではなく、SunやSGI製品、Windows、果てはBeOSまで扱っている。店舗はビルの3F～5Fにあるが、そのうちLinux対応ソフトウェアは4Fに、またハードウェアは主に5Fで販売されている（ここには同社独自ブランドの製品がある）。3FではPCパーツも販売しているので、ここに来ればLinuxマシン構築に必要なものはすべて揃えられるだろう。Linuxプレインストールサーバも販売されている。

住所：東京都千代田区外神田1-11-4
ミツワビル4F～5F
TEL：03-3251-7611
URL：http://www.plathome.co.jp/

■otto（おっと）UNIX本舗

おっとUNIX本舗は、中古のサーバやワークステーションの販売を得意とするショップだ（新品も扱っている）。数年前には数十万円もしたPCサーバが、中古とはいえ数万円で購入できるため、企業ユーザーが部課の経費で購入していくそうだ。Linuxをプレインストールした PCサーバも販売している。またPCだけではなくSUNやHPのUNIXサーバ／ワークステーションもあるが、これにLinuxをインストールして運用するユーザーもいるとのこと。Linuxのインストールや環境構築は自分でできるから、自作より信頼性の高いハードウェアを安価に入手したい、というヘビーユーザーが同店のリピーター客になっているようだ。

住所：東京都千代田区外神田3-12-15
チチブ電機ビル5F
TEL：03-3257-2228
URL：http://www.ottonet.co.jp/

写真1-1　ぷらっとホームの店内
Linux対応ソフトウェアのある4Fには、このように多くのディストリビューションが並んでいたほか、商用XサーバのAccelerated Xなど、ほかでは入手しにくい製品も数多く見かける。

写真1-2　おっとUNIX本舗の店内
奥には大手メーカー製の中古PCサーバが格安の価格でずらりと並ぶ。手前のガラスケースには、内蔵プリンタサーバや専用メモリ、テープドライブなど入手の難しいパーツが収まっていた。

ATAPI CD-RWドライブでCD-Rを焼こう！

最近のメーカー製PCには、CD-RWドライブ内蔵モデルが少なくない。いずれも搭載しているのは、ATAPI接続のCD-RWドライブだ。SCSI接続タイプだとSCSIホストアダプタの分だけコストが上がるので不利であることは、容易に想像がつく。また登場当初はトラブルの多かったATAPIタイプだが、現在では安定動作するようになったことも、採用増加に貢献しているようだ。

LinuxでもATAPI CD-RWドライブを利用する方法はある。もしSCSIタイプと遜色なく使えるなら、外付けSCSI CD-RWドライブと比べて5000円〜1万円ほど節約できる。この差は大きい。

というわけで、今回購入したのがリコー製MP7060A-EDという製品だ（**下の写真**）。同社製のCD-RWドライブの中では最速の6倍速でCD-Rを焼ける。この価格で6倍速は結構安いほうだが、書き込みソフトなしのバルク品なら2万5000円を切る場合もある。

ATAPI CD-RWドライブのセットアップ

Linuxカーネルには、ATAPIデバイスをSCSI機器に見せかけてコントロールする「SCSIホストアダプタ・エミュレーション」という機能が実装されている。もともとATAPI（ATA Packet Interface）とは、SCSIで使われているコマンドパケット体系を、ATAすなわちIDEに導入した規格なのだ。そこでIDEインターフェイスをSCSIホストアダプタに見せかける特殊なドライバを設けると、CD-R書き込みツールのようなSCSIのコマンドパケットを直接生成するようなアプリケーションで、ATAPIデバイスが使えるようになるわけだ（もちろんIDEとSCSIで異なる部分は多く、制限も存在するはずだ）。ATAPI CD-RWドライブも、SCSIホストアダプタ・エミュレーションによりSCSI CD-RWドライブとして認識させて使うのだ。

ここでは、2ポートのIDEインターフェイスのうち、セカンダリのほうにMP7060A-EDをマスタ、手持ちのパイオニア製ATAPI DVD-ROMドライブをスレーブにして接続して試した（DVDに意味はない。ただのCD-ROMドライブとして使った）。この場合、前者はhdc、後者はhddというデバイス名が割り当てられる。

購入したATAPI CD-RWドライブ

株式会社リコー

MP7060A-ED

オープンプライス

http://www.ricoh.co.jp/

購入価格：2万7800円

読み出しは最大24倍速（CAV）、書き込みはCD-Rで最大6倍速、CD-RWなら最大4倍速が可能なCD-RWドライブ。購入したのはATAPI（IDE）接続で内蔵タイプのみ。姉妹製品としてSCSI-2を採用した外付け／内蔵ドライブもある。CD-R書き込みツールはEasy CD Creatorデラックス版が、パケットライトはDirctCD2.5が添付されている（Windows用）。

■カーネルのリコンパイル

SCSIホストアダプタ・エミュレーションを有効にするには、カーネルコンフィグレーションを変更してリコンパイルしなければならない。それにはまずroot権限で、

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

と実行して設定画面を表示させる。コンソールモードならxconfigの代わりにmenuconfigを使うとよい。設定画面では**表2-1**のように各項目を設定し直す。「y」だとカーネルに組み込み、「m」はモジュール、「n」は組み込まない、という意味だ。**表2-1**で特に重要なのは「Include IDE/ATAPI CDROM support」をモジュールにすることだ。カーネルに組み込むとLinux起動時にこれがドライブを確保してしまい、SCSIホストアダプタ・エミュレーション・ドライバが利用できなくなってしまうのだ。

設定が済んだら、「Save and Exit」でコンフィグレーションファイルを保存し、リコンパイルする。

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

■liloやconf.modulesの修正

次にliloの設定を変更して、これまで動いていた環境を維持しつつ、新しいカーネルで起動できるようにしよう。それには**リスト2-1**のように/etc/lilo.confを書き替える。①はデフォルト

で新カーネルを起動する指定で、②は新カーネルのために追加したセクションである。この最下行（本来は折り返していないことに注意）が、ATAPI CD-RWドライブとDVD-ROMドライブをSCSIエミュレーションドライバ（ide-scsi）で動かすためのおまじないだ。

あとは、以下のように/bootを書き替える。

```
# cd /usr/src/linux
# cp System.map /boot/System.map-cdrw
# cp arch/i386/boot/bzImage /boot/vmlinuz-cdrw
# cd /boot
# rm System.map
# ln -s System.map-cdrw System.map
# /sbin/lilo
```

最後のliloの実行を忘れずに。

また、モジュールの依存関係などを記したconf.modulesにも追加の記述が必要だ（**リスト2-2**）。①はSCSI CD-ROMドライバ、②はide-scsiドライバの自動ロードを指定している。③はide-cdドライバがATAPIドライブ2台を制御しないよう無視させる指定だ。

リスト2-1　lilo.confの修正箇所

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux-cdrw ―――――①

image=/boot/vmlinuz-2.2.12
  label=linux
  initrd=/boot/initrd.img
  read-only
  root=/dev/hda1

image=/boot/vmlinuz-cdrw
  label=linux-cdrw
  initrd=/boot/initrd.img ――②
  read-only
  root=/dev/hda1
  append="hdc=ide-scsi
hdd=ide-scsi"
```

■cdromデバイスの再構築

ここでいったんLinuxを再起動し、新カーネルが立ち上がったら/dev/cdromへのシンボリックリンクを修正する。

```
# rm /dev/cdrom
# ln -s /dev/scd1 /dev/cdrom
```

これでDVD-ROMドライブを/dev/cdromでアクセスできる（CD-RWドライブは/dev/scd0）。DVD-ROMドライブに適当なCD-ROMを入れてマウントしてみよう。

```
# mount /mnt/cdrom
```

マウントが成功すると、dmesgは**画面2-1**のように、各ドライブごとに①や②のような諸元を記録する。一見しただけではATAPIデバイスとは分からない。

CD-R関連ツールのインストール

次にCD-Rを焼くためのツールをインストールしよう。CDのイメージを作るmkisofsや実際の書き込みを行うcdrecordは、**表2-2**のようにRed Hat Linux 6.1JのCD-ROMに収録されている。そこで、

```
# cd /mnt/cdrom/RedHat/RPMS
# rpm -ivh cdrecord-1.8a29-2.i386.rpm
# chmod +s `which cdrecord`
# rpm -ivh mkisofos-1.12b5-5.i386.rpm
```

セクション	設定項目	設定内容
Block Device	Enhanced IDE/MFM/RLL … support	y
	Include IDE/ATAPI CDROM support	m
	SCSI emulation support	m
	Loopback device support	m
SCSI	SCSI support	yまたはm
	SCSI CD-ROM support	yまたはm
	Enable vendor-specific extensions	y
	SCSI generic support	yまたはm
Filesystems	ISO 9660 CDROM filesystem support	y
	Microsoft Joliet CDROM extensions	yまたはm

表2-1　ATAPI CD-RWドライブのためのカーネルコンフィグレーション

cdrecord のWebページ	http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/index.html
cdrecord（Red HatのCD-ROM）	/RedHat/RPMS/cdrecord-1.8a29-2.i386.rpm
mkisofs（Red HatのCD-ROM）	/RedHat/RPMS/mkisofs-1.12b5-5.i386.rpm
X-CD-RoastのWebページ	http://www.fh-muenchen.de/rz/xcdroast/
xcdroast 0.96e-1	ftp://ftp.infomagic.com/pub/mirrors/linux/RedHatContrib/libc6/i386/xcdroast-0.96e-1.i386.rpm
Red Hat Linuxミラーサイト一覧	http://www.redhat.com/download/mirror.html

表2-2　使用したファイルと情報

リスト2-2　conf.modules

```
              :
alias scd1 sr_mod ―――――――――――――①
alias scsi_hostadapter ide-scsi ―――――②
options ide-cd ignore=hdc,hdd ―――――③
```

とインストールできる。chmodコマンドを忘れるとroot権限を持たないユーザーがcdrecordを実行できないので注意したい。

次はX-CD-Roast 0.96eというGUIベースのCD-R書き込みツールをインストールする。これは、cdrecordなどのコンソールコマンドを呼び出して書き込みを行なっている。ただし、すでにインストールしたcdrecord 1.8は使わず、内部に抱えた古いバージョンのコマンドを使ってしまうのがデメリットだろうか。もっとも、今回のように比較的新しいデバイスを揃えたにも関わらず、X-CD-Roast 0.96eの動作に問題は特になかった。

X-CD-Roastは、**表2-2**を参考にrpmファイルをダウンロードしたら、

```
# rpm -ivh xcdroast-0.96e-1.i386.rpm
```

でインストールできる。

CD-R関連ツールのインストール

ファイルやCDをバックアップするような用途なら、コンソールコマンドのcdrecordよりGUIのX-CD-Roastのほうが扱いやすいのは自明だ。ブータブルCD-ROMの作成など、X-CD-Roastにはない特殊な機能が必要なときにcdrecordなどのコンソールコマンドを使えばよいだろう。

さてX-CD-Roastを起動したら、まずは「Setup」ボタンを押して各種設定をしなければならない。重要なのは**画面2-2**の「CD Setup」である。最初「Audio-Read-Mode」をAutodetectにしていたところ、音楽CDのトラック読み出しに失敗してしまったのだ（ノイズしか記録されなかったのにエラーは出なかった！）。

```
                    :
scsi0 : SCSI host adapter emulation for IDE ATAPI devices
scsi : 1 host.
  Vendor: RICOH     Model: CD-R/RW MP7060A   Rev: 1.30   ①
  Type:   CD-ROM               ANSI SCSI revision: 02
  Vendor: PIONEER   Model: DVD-ROM DVD-104   Rev: 1.14   ②
  Type:   CD-ROM               ANSI SCSI revision: 02
Detected scsi CD-ROM sr0 at scsi0, channel 0, id 0, lun 0
Detected scsi CD-ROM sr1 at scsi0, channel 0, id 1, lun 0
sr0: scsi3-mmc drive: 24x/24x writer cd/rw xa/form2 cdda tray
Uniform CDROM driver Revision: 2.56
sr1: scsi3-mmc drive: 0x/0x cd/rw xa/form2 cdda tray
                    :
```

画面2-1　SCSIエミュレーションが成功したときのdmesg出力

画面2-2　X-CD-Roastのセットアップ画面

また、音楽CDを読み出すATAPIデバイスは、SCSIエミュレーションを有効にしないと、「Audio-Read-Device」に選択肢として表れないので注意したい。わざわざリスト2-1／2-2で、CD-RWドライブだけではなくDVD-ROMドライブまでSCSIエミュレーションを有効にするよう設定したのは、DVD-ROMドライブで音楽CDのトラックを読み出すテストをしたかったからだ（最大10倍速で音楽データを取り出せることを確認できた）。

X-CD-Roastの実際の使い方については、ヘルプなどを参照されたい。GUIなのでコンソールコマンドよりはるかに簡単に習熟できるだろう。しかし、最近のWindows用書き込みツールに比べると、X-CD-Roast 0.96eの機能はやはり貧弱に感じられる（ブータブルCD-ROM未対応など）。すでに新機能が追加された0.98のアルファ版が登場しているので、今後に期待したいところだ。

Linuxでテレビを見よう！

PCの画面でテレビを見たり動画を録画したりするには、TVチューナーやビデオキャプチャのカードをPCに組み込むのが一般的だ。Windowsならカードに付属のドライバで容易に実現できるが、Linuxだってフリーソフトウェアなどでこうした機能を利用できるのだ（ちょっと面倒だけど）。

今回Linuxで試したのは下の写真にある2製品で、どちらもTVチューナーとビデオキャプチャ両方の機能を備えたPCIカードである。実売価格はリモコンの有無やメーカーブランドなどによりピンキリだが、キャプチャだけのカードなら5000円程度から販売されているので、気軽に試せるのではないだろうか。

製品選びで重要なのはコントローラチップだ。Linuxの場合、Conexant（旧Brooktree）製のBt848（**写真3-1**）やBt878（**写真3-2**）を採用した製品が一番動かしやすいようだ。ZORANなど他のコントローラチップもサポートされているが、製品数やネットでの成功例はBtシリーズのほうが多い。それがこの2製品を選んだ理由だ。もっともBtシリーズなら確実に動作するというわけではないが。

Linuxでテレビを見るには

Justy JTT-02でもAMI-999でもテレビ画像を表示する場合、TVチューナーで選局して得られる動画像をコントローラチップでキャプチャし、グラフィックスカード上のフレームバッファに転送する、という手順が踏まれている。これをLinuxで実現するのがVideo4Linuxというソフトウェアで、ビデオを扱うのに必要なAPIを提供する。普通Video4Linux本体はLinuxカーネルと共に配布されているし、コントローラのドライバなども含まれていることが多い。

余談だが、Video4Linuxよりさらに扱いやすいビデオ環境をLinuxで構築するために、Video4Linux2というプロジェクトがすでに登場している。Video4Linux2対応のドライバ（後述のbttvなど）もリリースされており、今後が楽しみだが、ここでは従来のVideo4Linuxを用いることにする。

購入したTVチューナー＆ビデオキャプチャカード

株式会社トライコーポレーション

Justy JTT-02

2万4800円

http://www.justy.co.jp/

購入価格：1万6800円

別名TV WATCH MATE Proという製品で、TVチューナーとビデオキャプチャをサポートするPCIカードと赤外線リモコンがセットになっている。コントローラはBt848Aで、TVチューナーのモジュールはPhilips製だった。Windows 98用ドライバが付属していないのが難点か。

Amigo Technology

AMI-999

オープンプライス（輸入品）

http://www.amigo.com.tw/

購入価格：7680円

いちおう化粧箱には入っているがノーブランド品扱いだった輸入品で、TVチューナーとビデオキャプチャ対応のPCIカードが入っていた（リモコンはなし）。コントローラにBt878を採用し、Lucky Gold製TVチューナーモジュールを搭載していた。Windows 98用はもちろんNT用ドライバも存在する。

■Justy JTT-02のセットアップ

写真3-1、**2**のBt848／878をVideo4Linux上で動作させるには、bttvというモジュールを用いる。幸いなことにJusty JTT-02は、Red Hat Linux 6.1Jが採用するカーネル2.2.12付属のbttvで問題なく動作した。またVideo4Linux自体もRed Hat Linux 6.1Jではモジュールとして組み込まれるので、カーネルのリコンフィグは不要だ。

次は/etc/conf.modulesを編集して**リスト3-1**を追加し、各モジュールが適切にロードされるようにする。リスト3-1で、①はbttvと同時にtunerモジュールもロードするための指定だ。②はbttvにキャプチャカードの自動検出とFMラジオ機能のオフを指定している。④ではTVチューナーの種類にPhilips NTSCを選んでいる。なおリスト3-1でJusty JTT-02とAMI-999はbttvのオプ

ション指定が異なり、前者は②を、後者は③を有効にしなければならない（行頭の#を入れ替える）。

conf.modulesを変更したら、

```
# modprobe bttv
# lsmod
```

と実行して各モジュールのロードを確かめてみよう。正常ならbttvのほかtunerとi2c、videodevがロードされたと表示されるはずだ。

■AMI-999のセットアップ

Justy JTT-02同様、AMI-999のコントローラチップBt878もbttvで動作するはずなので、細かいオプション設定（**リスト3-1**参照）を除けば、組み込み手順は前述のJusty JTT-02とほぼ同じはずだ。しかし、実際にはTVチューナーから音声が正しく出ないというトラブルが生じた。どうやらRed Hat Linux 6.1Jに付属していたbttvの問題だったようで、bttvのバージョンを上げたら解決できた。以下にbttvのアップグレード方法を記そう。原稿執筆時点でbttvの最新バージョンは0.7.6だったが、ここではより安定していると思われる0.6.4hを、後述するxawtvのホームページからダウンロードして使っている（**表3-1**参照）。

bttv-0.6.4h.tar.gzファイルを適当な

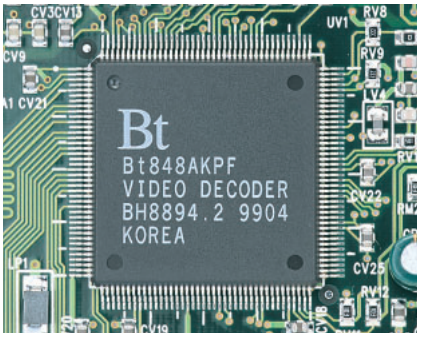


写真3-1　Conexant Bt848A
2〜3年ほど前に多くのビデオキャプチャカードが採用したコントローラ。アナログのビデオ信号をキャプチャしてデジタル化し、PCIにバスマスタで転送できる。TVチューナーの機能は含まれない。

ディレクトリにおいたら、**リスト3-1**の変更をした後で（③を忘れずに！）、

```
# tar zxvf bttv-0.6.4h.tar.gz
# cd bttv-0.6.4h
# make
# make ins
```

と実行し、モジュールの作成・組み込みを行う。なおRedHat Linux 6.1Jではここでエラーが生じることがあったが、一度カーネルをリコンパイルすれば解消される。またmake insで新しいモジュールがロードされるので、エラーが生じなければ、

```
# cd driver
# cp -i *.o /lib/modules/`uname -r`/misc
```

写真3-2　Conexant Bt878
Bt84xシリーズの改良版コントローラで、Bt848Aの機能はすべて継承しつつ、音声もモノラルでキャプチャする機能が追加された。ステレオ音声をキャプチャできるBt879という姉妹製品もある。

と実行してモジュールを所定のディレクトリにコピーしよう。あとはLinuxを再起動して新しいモジュールをロードすれば、AMI-999は動くはずだ。

■xawtvのインストールと設定

残るはテレビ画像のビューアを用意することだ。Linuxでは、Gerd Knorr氏によるxawtvというビューアがよく知られており、機能も豊富である。ここでは最新バージョンである3.01を使用した（Webページやダウンロードファイル名は**表3-1**を参照）。

xawtv 3.01はtar.gzファイルで配布されているが、rpm形式でのインストールにも対応している。そのためRed Hat Linuxへのインストールはとても簡単で、

```
# rpm -ta xwatv-3.01.tar.gz
```

Video4LinuxのWebページ	http://roadrunner.swansea.uk.linux.org/v4l.shtml
Video4Linux2のWebページ	http://millennium.diads.com/bdirks/v4l2.htm
xawtvのWebページ	http://www.in-berlin.de/User/kraxel/xawtv.html
bttv 0.6.4h	http://www.in-berlin.de/User/kraxel/v4l/bttv-0.6.4h.tar.gz
xawtv 3.0.1	http://www.in-berlin.de/User/kraxel/v4l/xawtv-3.01.tar.gz
XAnimのWebページ	http://xanim.resnet.gatech.edu/home.html
XAnim（Red HatのCD-ROM）	/RedHat/RPMS/xanim2.80.1-3.i386.rpm
RealProducer G2	http://www.realnetworks.com/products/producer/
RealPlayer G2アルファ版	http://proforma.real.com/real/player/linuxplayer.html
LIRCのWebページ	http://fsinfo.cs.uni-sb.de/~columbus/lirc/
Red Hat Linuxハードウェアサポート情報	http://www.redhat.com/corp/support/hardware/index.html

表3-1　使用したファイルや情報
※バージョンが上がってファイル名が変わることがあるので注意されたい

▼リスト3-1　/etc/conf.modulesの追加部分

```
# capture driver
alias char-major-81 videodev
alias char-major-81-0 bttv
pre-install bttv modprobe -k tuner ——①
options bttv card=0 radio=0 ——②
#options bttv card=19 radio=0 ——③
options tuner type=2 ——④
```

```
# cd /usr/src/redhat/RPMS/i386
# rpm -Uvh xawtv-3.01-1.i386.rpm
```

と実行するだけだ。ただ最後にxsetが実行されるので、X環境でインストール作業をしたほうがよさそうだ。

次はxawtvの設定だ。xawtvはその起動時にユーザーのホームディレクトリから.xawtvという設定ファイルを読み込むので、ユーザーごとに.xawtvを用意する必要がある。**リスト3-2**が今回作成した.xawtvだ。ここで①には地上波ならjapan-bcastを、CATVならjapan-cableを指定する。②は音声出力をつなぐサウンドカードのミキサーデバイスの指定で、lineまたはvolにすればxwatvから音量を調節できる。③では起動時の入力映像（ここではテレビ）を選べる。④はオーバーレイ方式の指定で、問題があればgrabdisplayに変更してみよう（ただしCPUの負荷が増える）。⑤はプリセットチャンネルの設定で、リストでは編集部のある東京に合わせている。ほかの地域では、[]内の名称とチャンネル番号を差し替えてほしい。

ここまで設定したら、Xの起動後にxawtvを実行してみよう。トラブルがなければ**画面3-1**のように各ウインドウが開いてテレビ画像が表示されるはずだ。一方、テレビの音声はTVチューナー＆キャプチャカードの背面にある音声出力端子から得られる。これをサウンドカードの音声入力端子に接続すれば、ほかの音とミキシングされて出力されるようになる。またリスト3-2の②を設定しておけば、音量も＋／－キーで調節できる。

▼リスト3-2　.xawtvの例

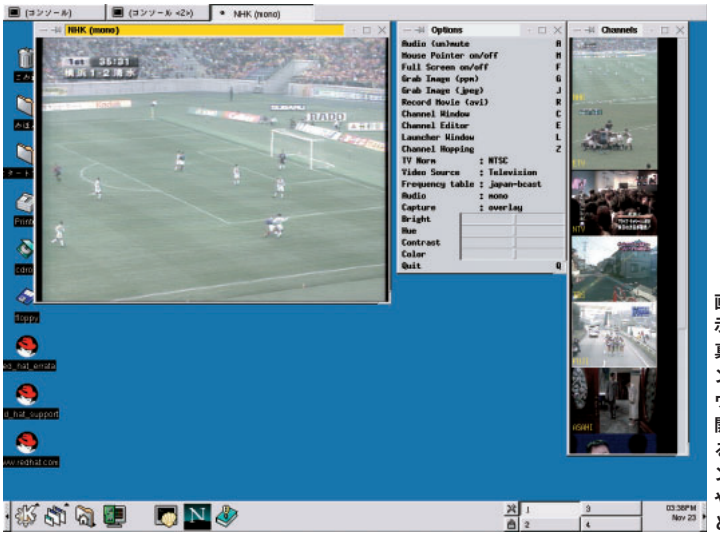
```
[global]
freqtab = japan-bcast ―――― ①
mixer = line ―――― ②
fullscreen = 1024x768

[defaults]
source = television ―――― ③
norm = ntsc
capture = overlay ―――― ④

[NHK]
channel = 1
[ETV]
channel = 3
[NTV]
channel = 4
[TBS]
channel = 6
[FUJI]
channel = 8 ―――― ⑤
[ASAHI]
channel = 10
[TOKYO]
channel = 12
[MXTV]
channel = 14
[VIDEO]
source = Composite1
```

Linuxでビデオキャプチャ

テレビが表示できたら、録画もしてみたいところだ。bttvはテレビ画像の録画はもちろん、外部ビデオ入力からのキャプチャもサポートしている。Video4Linux対応のキャプチャツールはいくつかあるが、もっとも手軽なのは、すでにインストールしたxawtvを使うことだ。xawtvのウインドウ上でRキーを押すと、record moviesウィンドウが現れる。これでキャプチャすると、320×240ドット／15ビットカラー／モノラル音声／15フレーム/秒ならCeleron-300MHz搭載PCで実用的な画質が得られた。もっと画質を上げたりフレームレートを高めたりするには、より高速なCPUが必要だろう。

録画したファイルはAVIビューアのxanimで表示できる。xanimはRed Hat Linux 6.1J標準添付なので、プログラムCDからrpm -Uvhでインストールできる（**表3-1**を参照）。

xawtvなどでビデオをキャプチャしても、そのままでは動画を自分だけで楽しむしかない。しかし、Real Networksが提供するVideo4Linux対応のRealProducer G2（**画面3-2**）を使えば、事実上インターネットの標準であるRealMedia形式の動画を作成できるのだ。RealServerと組み合わせて個人でRealMedia放送局を作ってしまえば、Linuxでキャプチャ環境を作る醍醐味を味わうことができるだろう。Linux用RealProducer G2はフリー版を同社のWebサイトからダウンロード



画面3-1　xawtvの表示例
真ん中のOptionsウィンドウや右のChannelsウィンドウは、自由に開いたり閉じたりできる。キーボードからワンタッチでチャンネルや音量を変更できるなど、使い勝手はよい。

できる（**表3-1**参照）。Red Hat Linux 6.1ではパッチをあてる必要があるので注意したい。

作成したRealMedia形式のファイルをLinuxで表示するにはRealPlayer G2（**画面3-3**）が必要だ。しかしRed Hat Linux 6.1で安定動作するプレーヤはまだリリースされていないようだ。RealPlayer G2のフリー版は、表3-1のWebサイトからダウンロード可能だが、まだアルファ版なので動作はかなり不安定だ。RealMedia形式を表示できる環境がLAN上にない場合、個人が映像を撮り溜めるのにRealMedia形式を使うのは、あまり現実的とはいえないだろう。

TVチューナーのリモコンも使える？

試用したJusty JTT-02には赤外線リモコンとその受光機が同梱されており、Windowsでは利用できた。またAMI-999にもリモコン受光機の接続用とおぼしき端子が装備されていた。こうしたちょっとマイナーなデバイスに対してもドライバが開発されているのがLinuxの面白いところだ。現在、LIRC（Linux Infrared Remote Control）と呼ばれるプロジェクトで、さまざまな赤外線デバイスをLinuxで利用できるようにする試みが進められている（Webサイトは**表3-1**参照）。その成果として、すでに一部のTVチューナーカードでリモコン機能がサポートされるようになった。残念ながら今回はテストできなかったが、Justy JTT-02のリモコンもLinuxで利用するのは不可能ではなさそうだ。

購入したサウンドカード

クリエイティブメディア株式会社
Creative VIBRA 128 for DOS/V
4500円
http://www.creaf.co.jp/

購入価格：3600円

Sound Blasterシリーズで知られるCreative Labs.の廉価版PCIサウンドカード。小さな箱にカードとペラ1枚の説明書とCD-ROMしか入っていないチープな構成だが、日本語版である。サウンドチップはCreative（Ensoniq）ES1371と認識された。PCM（Wave）を鳴らすならこれで十分だろう。

サウンドカードの設定

TVチューナーカードやビデオキャプチャカード単体では、音は聞けても、音声をキャプチャできない。やはりサウンドカードと併用すべきだろう。お勧めなのは、今回試したクリエイティブメディアのVIBRA 128のように、安価なPCIサウンドカードである。1万円以上の高価なサウンドカードはその特徴をLinuxで生かせないことが多いし、また音質も最近の安価なカードはそれほど悪くないからだ（だいたいテレビ音声に必要な音質などタカが知れている）。ISAカードはセットアップが面倒なわりにスペックが見劣りするので避けよう。

Red Hat Linux 6.1Jの場合、VIBRA 128はsndconfigというツールで自動認識されてドライバも組み込まれるので、インストールは非常に楽だ（**画面3-4**）。同じメーカーのSound Blaster PCI128も自動認識された。新規にサウンドカードを購入する際には、各ディストリビューションのハードウェアサポート状況を確認してから製品を選ぼう（Red Hat Linuxのハードウェアサポート情報は、**表3-1**のWebサイトにある）。

画面2　RealProducer G2
Linux用のフリー版だ。

画面3-3　RealPlayer G2
Linux用はまだアルファ版だ。

画面3-4：sndconfigの認識画面
テストに使ったCreative VIBRA 128を自動で認識したところ。サウンドチップが「Ensoniq ES1371」と識別されているのがわかる。このあと、特にユーザーが設定することはほとんどない。

高速・大容量ハードディスクを増設しよう！

ハードディスクの大容量化と低価格化はとどまるところを知らず、ついに30GバイトのIDEハードディスクが3万円台で購入できるまでになった。

ところで現在入手できるIDEハードディスクは、ほとんどがUltraDMA/66対応のものだ。しかし、これまで多くのPCはUltraDMA/33モードまでのサポートだったので、その性能を活かすことができなかった。ただ、少なくとも新製品のマザーボードはほとんどがUltraDMA/66をサポートするようになってきたし、Promise Ultra66などの拡張カードを使えば既存のPCもUltraDMA/66対応にすることができるなど、状況は変わりつつある。今回はPromise Ultra66をLinuxで使う方法を探った。

UltraDMA/66対応はまだリスキー？

Promise Ultra66をはじめ、UltraDMA/66に対応したドライバは開発版カーネル2.3.xから収録されている。しかし後述するように、パッチをあてることで安定版カーネル2.2.xのままでUltraDMA/66に対応可能だ。まだ標準リリースに取り込まれていない機能である以上、ある程度のリスクは覚悟しなければならない。少なくともパッチに付属しているドキュメント類はかならず目を通すようにしよう。

今年末にリリースされる予定のカーネル2.4にはUltraDMA/66に対応したドライバも取り込まれるはずで、それまで待つというのもひとつの手だ。

UltraDMA/66ドライバのインストール

Promiseは**表4-1**に示した自社ホームページで、Ultra66のLinuxドライバのβ版を配布しているが、今回はUnified IDEパッチ付属のドライバを用いた。Unified IDEパッチはカーネルと同様に、kernel.orgから入手可能だ。日本ではRingServerにミラーされているので、たとえば表4-1のso-netなどからダウンロードできる。

カーネル2.2.12対応のUnified IDEパッチも存在するが、今回は安定版カーネルの最新版2.2.13をインストールして、Unified IDEパッチも最新版を使用した。カーネル2.2.13も**表4-1**のRing Serverなどからダウンロードできる。

ダウンロードが終わったら、カーネルを展開して、Unified IDEパッチを適用する。実際のコマンドは**画面4-1**の通りだ。

パッチ適用後にmake xconfigを実行すると、カーネルの設定画面が表示される。**画面4-2**は、カーネル設定画面の「Block devices」セクションだ。Unified IDEパッチを適用する前の同セクションの画面と比較すれば一目瞭然だが、たとえばABIT BP6マザーボードなどが採用しているHPT366や、Promise Ultra66が採用しているPDC20246など、UltraDMA/66に対応したドライバが追加されている。

さて、Promise Ultra66のサポートを有効にするには、「PROMISE PDC20246/PDC20262 support」を「y」にする。また、マザーボード上のUltraDMA/33サポートを有効にする「Intel PIIXn chipsets supoort」も「y」を選択する。

「Block Devices」セクションで注意しなければならないのは、「Use DMA by default when available」オプションが有効になっているかどうかだ。もし「n」となっている場合、「y」に変更しなければならない。

今回用いたUnified IDEパッチは「Block Devices」セクション以外は変更しないので「Block Devices」セクション以外の設定については別途ドキュメントなどで確認してほしい。make xconfig以降のコンパイルや新しいカー

購入したIDEコントローラ／ハードディスク

Promise Technology

Ultra66

オープンプライス

http://www.promise.com/

IBM

DPTA-371360

オープンプライス

http://www.ibm.com/harddrive/

購入価格：2万1240円

Ultra66は、2ポートのIDEを持つUltraDMA/66対応のIDEコントローラだ。ディスクBIOSがあるので、接続したハードディスクからブートできる。購入したのはシネックス扱いの日本語版パッケージで、簡易な日本語マニュアルが付属していた。ハードディスクはIBM製Deskter 34GXPシリーズのうち、13.6GバイトのDPTA-371360を選んでいる。なお購入価格は両方の合計だ。

ネルのインストール手順については、91〜93ページの記事を参照されたい。

hdparmをアップデートする

LinuxでIDEハードディスクのパラメータを変更したり、ステータスを確認するにはhdparmコマンドを用いる。hdparmコマンドは、Red Hat Linuxでは/sbin/hdparmにインストールされる。ところでRed Hat Linux 6.1Jのhdparmはバージョン3.4だが、Ultra DMA/66のステータス表示ができないという問題がある。現在の状況がわからないのはやはり気持ち悪いので、この際hdparmも最新版に入れ替えてしまおう。

hdparmの最新版はバージョン3.6で、配布元は**表4-1**のとおりだ。ダウンロードしたhdparm-3.6.tar.gzを、次のコマンドで展開する。

```
# tar zxvf hdparm-3.6.tar.gz
# cd hdparm-3.6
```

Ultra-DMA Mini-Howto	http://pobox.com/~brion/linux/Ultra-DMA.html
Unified IDEパッチ	ftp://ring.so-net.ne.jp/pub/linux/kernel.org/kernel/people/hedrick/
カーネル2.2.13	ftp://ring.so-net.ne.jp/pub/linux/kernel.org/kernel/v2.2/linux-2.2.13.tar.bz2
hdparm 3.6	ftp://metalab.unc.edu/pub/Linux/system/hardware/hdparm-3.6.tar.gz
Bonnie	http://www.textuality.com/bonnie/
Promise提供のUltra66ベータ版ドライバ	http://www.promise.com/Latest/latedrivers.htm

表4-1　使用したファイルと情報

hdparm自体のコンパイルは簡単だが、標準のMakefileにはインストール先に/sbinがないので、このままRed Hat Linux 6.1Jにインストールするとhdparmが更新されない。そこで、Makefileの「install」セクションを一部修正する。**リスト4-1**は、hdparm-3.6のMakefileの「install」セクションに変更を加えたものだ。まず古いhdparmをちゃんと削除するよう①の行を追加する。次に以前と同様/sbinディレクトリにhdparmをインストールするように、②の行を「#」でコメントアウトして、③を新たに追加する。

変更の必要なのはこれだけだ。あとは、

```
# make ; make install
```

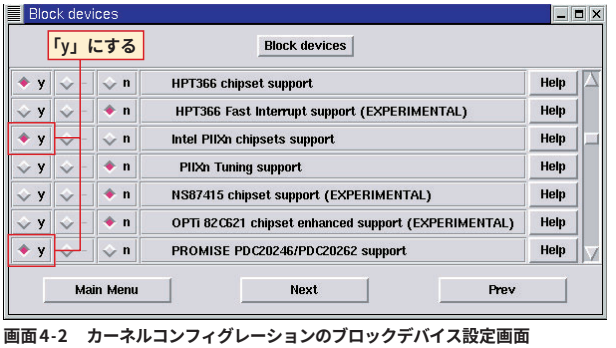
と実行すると、hdparmバージョン3.6が/sbinにインストールされる。

lilo.confを修正する

IDEハードディスクをUltraDMA/66モードで使うには、これまでの40芯ケーブルではなく、80芯のフラットケーブルを使わなければならない、とATA/ATAPI4規格で定められている。UltraDMA/66に対応したコントローラはどのケーブルが使われているか自動認識するのだが、Unified IDEパッチのPromise PDC202xxドライバはま

```
# cd /usr/src
# rm linux
# bzip2 -cd /tmp/linux-2.2.13.tar.bz2 | tar xvf -
# bzip2 -cd /tmp/ide.2.2.13.19991111.patch.bz2 | patch -p0
# mv linux linux-2.2.13
# ln -s linux-2.2.13 linux
# cd linux
# make distclean ; make xconfig
```

画面4-1　パッチをあてるためのコマンドライン
ここではパッチとカーネルのアーカイブが/tmpにあると想定している。



画面4-2　カーネルコンフィグレーションのブロックデバイス設定画面

リスト4-1　hdparm-3.6のMakefile修正箇所

```
                                        :
install: all hdparm.8
  if [ -f $(destdir)/sbin/hdparm ]; then rm -f $(destdir)/sbin/hdparm ; fi ―――― ①
  if [ -f $(destdir)/usr/sbin/hdparm ]; then rm -f $(destdir)/usr/sbin/hdparm ; fi
                                        :
#  install -m 755 -o root -g root hdparm $(destdir)/usr/local/sbin ―――― ②
  install -m 755 -o root -g root hdparm $(destdir)/sbin ―――― ③
  install -m 644 -o root -g root hdparm.8 $(destdir)/usr/local/man/man8
                                        :
```

だこの機能に問題があり、自動認識できない。この問題を回避するには、Linux起動時にパラメータを指定しなければならない。そのパラメータとは「ideX=ata66」（Xは0～3の整数）で、今回はこれまで使ってきたオンボードIDEインターフェイス（ide0とide1）と併用する設定にしたため、Ultra66に割り当てられたide2とide3に対してこのオプションを指定する。

Linux起動時にパラメータを与えるには、/etc/lilo.confに「append」コマンドで指定するのがもっとも簡単な方法だ。**リスト4-2**はPromise PDC202xxドライバをインストールした場合の、/etc/lilo.confの例を示している。①の「default」という指定は、複数のカーネルイメージのうちどちらを標準とするかを、「label」行の名前で変更している。新たにインストールしたイメージ（ここでは②）の動作を確認するまでは、古いイメージを標準としたほうが安全だ。③がここで追加した「append」行だ。

なお、②のイメージを起動するには、Linux起動時に表示される「boot:」プロンプトで、ラベル名（ここでは「linux-idepatch」）を入力すればよい。

リスト4-2　lilo.confの追加例

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux ―――――――――――――①
image=/boot/vmlinuz-2.2.12
  label=linux
  initrd=/boot/initrd.img
  read-only
  root=/dev/hda1
image=/boot/bzImage-ide ―――――――②
  label=linux-idepatch
  read-only
  root=/dev/hda1
  append="ide2=ata66,ide3=ata66" ―③
```

Ultra66の効果のほどはいかに？

せっかくUltraDMA/66対応にしたのだから、ベンチマークテストでハードディスク性能がどれくらい向上しているか調べてみよう。ただ、その前にhdparmを用いてIDEの設定をチューニングしておきたい。**表4-2**は、hdparmのDMA転送に関係するオプション一覧だ。このほかにもパフォーマンスに影響するオプションは多数あるので、詳細はmanを参照されたい。

テストにはBonnieというベンチマークプログラムを用いた（配布元は**表4-1**参照）。テスト用PCはメモリを128Mバイト搭載しているので、ベンチマークのファイルサイズは256Mバイトとして、Linuxのディスクキャッシュが影響しないよう注意した。

ベンチマークの結果は**表4-3**のようになった。意外なことに、読み出しの性能はまったく変わらない。これは今回テストに用いたハードディスクの性能の上限が、UltraDMA/33の転送レートで十分間に合ってしまうためだと予想される。もっと高速・大容量のドライブでは差が出るはずだ。

一方、書き込みについては、Promise Ultra66がPIIX4E（オンボードIDEコントローラ）より速いことは明らかだ。

写真4-1　Winux
最大3台のIDEハードディスクを切り替えてブートできるハードディスクセレクタ。マルチOS環境を常用する場合、安全にOSを切り替えられる。IDEの転送モードなどには一切関係なく使える。

複数のハードディスクでOSを切り替えるには

今回のようにディスクを増設した場合、そこにOSをインストールするとLILOによりOSの切り替えも複雑になる。それにLILOなどのブートセレクタソフトウェアが壊れてしまうと再構築はかなり面倒だ。

Winux（**写真4-1**）はブートセレクタ機能をハードウェアで実現するユニットだ。4つのハードディスクを管理できるが、同時に利用できるのは2台までだ。仕組みとしてはハードディスクのジャンパを切り替えているだけなので、ソフトウェアを必要としないのは強みだ。複数のOSを1台のPCで本格的に運用するなら、導入を検討してみてはどうだろうか。

表4-2　hdparmのオプション
これはhdparmのもつ非常に多くのオプションのなかの一部である。

（なし）	設定済みのオプションを表示する
-i	ドライブからステータスを得る
-t	メディアからの読み出し速度を測定する
-T	キャッシュからの読み出し速度を測定する
-d1	DMA転送を有効にする
-d0	DMA転送を無効にする
-d1 -X66	UltraDMA/33を有効にする
-d1 -X68	UltraDMA/66を有効にする

表4-3　ベンチマークテストの結果
書き込みでUltra66の効果が出ていることがわかる。

IDEコントローラ	転送モード	書き込み［Mバイト/秒］	読み出し［Mバイト/秒］
Ultra66	UltraDMA/66	24.3	19.9
Ultra66	UltraDMA/33	24.3	19.9
PIIX4E	UltraDMA/33	21.9	19.9

Column

Linuxマシンを LAN接続するための製品選び

ここ数年でLAN製品の価格は著しく低下し、家庭やSOHOでも小規模なLANなら格段に導入しやすくなった。写真5-1／5-2のように、デスクトップPCとノートPCそれぞれ1台をLAN接続することを想定して、Linuxで使えるネットワーク機器を選んでみると、特に10BASE-Tは安く済むことがわかる。ただ、製品の選び方には若干の注意が必要だ。

■デスクトップPC用ネットワークカード

VIA TechnologiesのVT86C100Aというコントローラ（あるいはその互換製品）を搭載したPCIカードは、100BASE-TX対応でLinux用ドライバがあり、2000円～3000円で販売されている。写真5-1／5-2のコレガ製PCIカードもこの互換チップを採用している。一方、10BASE-T専用PCIカードは1000円でもお釣りが来る価格だったりして、確かに安い。しかしその多くは保証なしのバルク品だったり、あやしいNE2000互換を謳っていたりと、正直1000円程度の差額を惜しんで手を出すほど魅力的ではない。将来100BASE-TXのネットワークに移行するのも容易になるので、ここは100BASE-TX対応PCIカードを購入しておくのが無難ではなかかろうか。

VT86C100A搭載カードは、Red Hat Linux 6.0以降ならインストール時に自動認識されるはずだ。モジュール名はvia-rhineである。5.2のころは、このvia-rhineモジュールは存在するのに自動で認識されないというトラブルもあったが、手動で設定するだけで使えた。ただディストリビューションとカードの組み合わせによっては、なぜか動作しないことがあるとも聞くので、安心を求めるなら、やや高価だが歴史のあるIntel（旧DEC）21140-AF採用カードを選ぶという手もある。

■ノートPC用ネットワークカード

ノートPCの場合、LAN接続にはPCカードを使う。ところがPCカードには、16bit PCカードとCardBusカードの2種類があり、選択肢が複雑だ。

10BASE-TXの場合、PCカードには16bit PCカードしか存在しないので迷わずに済む。価格も2000円～5000円程度なので買いやすい。写真5-1のプラネックス製PCカードはpcmcia-cs 3.0.13以降なら自動認識される。

一方、100BASE-TXの場合、性能面ではCardBusを選びたい。16bit PCカードではPCとの転送レートがせいぜい1M～2Mバイト/秒程度で、10Mバイト/秒の100BASE-TXの性能をまるで生かせないからだ（それでもよいなら10BASE-T専用PCカードを選べばよい）。しかし、PCカードを制御するソフトウェアpcmcia-csの現行バージョン（3.1.x）では、CardBusサポートはExperimental、つまり実験レベルに止まっている点が不安だ。写真5-2のスリーコム製CardBusカードをpcmcia-cs 3.1.2で使ったことがあるが、問題は感じられなかった。しかしこのカードは実売で1万7000円前後と高価なのが難点だ。ならば6000円～1万円程度のIntel 21143コントローラ採用カードを使いたいところだが、コントローラのリビジョンによって動作しない場合があり厄介だ。

■ハブ

2台のPCをLAN接続するなら、クロスケーブルで接続するとハブが必要ないので、より安く済む。しかし、クロスケーブル直結では通信できないネットワークカードも存在するし、ハブは導入しておくべきだろう（10BASE-Tならダイヤルアップルータ内蔵ハブで代用する手もあるが）。

10BASE-TX LANの場合、安価なのは10BASE-T専用シェアードハブしかないので、あとはポート数で選べばよい。5ポートなら2000円台から購入できる。10BASE-T／100BASE-TX LANの場合も、選択肢はスイッチングハブしかないだろう。というのも、シェアードハブやデュアルスピードハブは安価なスイッチングハブに駆逐されて市場にほとんど残っていないからだ。なお100BASE-TXシェアードハブは10BASE-T機器を接続できず不便なので、安くても買うのは避けよう。

家庭やSOHO向けの廉価なスイッチングハブは、5ポートなら1万円以下から購入できる。ただ、全ポートのうち全二重で通信できるのは2ポートだけ、といった変則的な製品もあるので、スペックをよく確認して購入しよう。

写真5-1　10BASE-T LANのセット例
左から、PCカードはプラネックスコミュニケーションズのENW-3503-T、PCIカードはコレガのFastEtherII PCI-TX（これだけは100BASE-TXも可）、5ポートのシェアードハブはエレコムのLD-PHB5NA。購入価格は合計で7260円だった。100BASE-TXが普及し始めていることもあって、最近10BASE-T製品はとても安い。ファイル共有を多用しなければ、10Mbpsという速度でも十分だろう。

写真5-2　10BASE-T／100BASE-TX両用LANのセット例
左から、PCカードはスリーコムジャパンの3CXFE575BT-JP、PCIカードはコレガのFastEtherII PCI-TX、5ポートのスイッチングハブはコレガのFast SW-5P。購入価格は合計で2万7000円と、10BASE-Tの4倍弱になってしまったが、これは3CXFE575BT-JPが高価なせいだ。逆に100BASE-TXスイッチングハブの価格はここ1～2年でかなり下がり、家庭やSOHOでも導入しやすくなった。

SMPマザーボードに換装しよう！

実はCeleronがDual構成でも動作することが世間にバレてから、パワーユーザの間でDualonブームが起きたのは記憶に新しい。なぜこれほど流行したのだろうか。まず、導入コストが安いことがあげられる。Dual対応マザーボードはそれほど高くはないし、Celeronがひどく安いことはご存じのとおりだ。また、Celeron-300AMHzが容易に450MHzにオーバークロックできたことも理由の1つだろう。

そのほか、LinuxやWindows NTがSMP（対称型マルチプロセッサ）に対応していたこと、マルチタスク環境を高速化することなどさまざまな利点があげられるが、最大の理由はやはり楽しいことだろう。SMPマシンを個人で所有できるなんて、考えるだけでも楽しいではないか。

今回選んだマザーボード

SMPマシンを組むために選んだのは、ABIT ComputerのBP6だ。最初からSocket370を2つ搭載するという、ずいぶん思い切った仕様が特徴的だ。同社はSocket370仕様のPentium III（現在発売されているFC-PGA版Pentium III）でSMPシステムを組むためのマザーボードだと言ってきた。しかし実際に登場したFC-PGA版Pentium IIIは、Celronと物理形状はほぼ同じなのに、信号レベルで互換性がない。また現在のFC-PGA版Pentium IIIはSMP構成をサポートしないとIntelは言っている。結局BP6は、Dual Celeronにしか使いようのないマザーボードなのだ。それでも、価格は最安値で1万3000円台とこなれてきているし、UltraDMA/66対応のHPT366チップ（**写真6-1**）をマザーボードに搭載するなど、マニア心をくすぐるお買い得商品といえる。

SMPを導入するには

いままで使っていたPCのマザーボードをBP6など別のものに付け替える場合、Linuxしか使っていない場合は意外に簡単で、拡張カードやドライブをごっそり移し替えてしまえばよい。もっともISAカードを使ってる場合は、システムリソースの設定を必要とするが、今回はPCI／AGPカードのみということで話を進める。

LinuxをSMP構成に対応させるにはリコンパイルが必要だ。場合によっては最新版のカーネルをインストールする必要があるだろう。リコンパイルの方法は91～93ページの記事を、また最新版カーネルの入手先は99ページの表4-1を参照してほしい。

カーネルをSMP対応にするには、make xconfigを実行した後に表示されるカーネル設定画面の「Processor type and features」セクション（**画面6-1**）において、「Symmetric multi-processing support」を有効にする。なお参考までに記すと、Celeronの場合「Processor family」は「PPro/6x86MX」であり、最大の性能を得るには「MTRR support」を「y」にしなければならない。

なお、一からLinuxをインストールする場合は、たとえばRed Hat Linux 6.1Jのカーネルは最初からSMPサポート付きでコンパイルされているので、インストールするだけでSMPサポートを有効にできる。こちらのほうがより手軽なので、一度手元にあるディストリビューションがSMPに対応しているかどうか確認してみるとよいだろう。

SMP動作の確認方法

SMPサポートを有効にしても、果たして両方のCPUがちゃんと動いているか見た目ではわからない（昔のBeBoxみたいにLEDのCPUメーターでも付いていればいいのだが）。もっとも手軽に

購入したCPUとマザーボード

ABIT Computer

BP6

オープンプライス

http://www.abit.com.tw/

Intel

Boxed Celeron（433MHz）

オープンプライス

http://www.intel.com/

購入価格：3万900円

BP6は登場当初、Celeron専用SMPマザーボードとして話題を呼んだマザーボードだ。ドーターカードを必要とせず安価なCeleronを2個用意するだけでよいので、SMPシステムとしては非常にコストパフォーマンスが高い。またUltraDMA/66対応を含む計4ポートのIDEを標準装備している点も特徴の1つだ。BP6と組み合わせたCelronはファン付属の化粧箱入りを選んでみた。購入価格はBP6×1枚とCeleron-433MHz×2個の合計だ。

確認する方法は、Xを起動してxosviewを実行することだ（**画面6-2**）。CPU0とCPU1の2つのCPUが同時に動作していることが見て取れる。/proc/cpuinfoにはCPUの情報が記されているので、こちらもcatなどで見てみるといいだろう。CPU0とCPU1、それぞれの情報が記されているはずだ。

SMPの注意点とは

SMP構成で注意しなければならないのは、SMPでは動作しないドライバが存在することだ。たとえばAPM機能はSMP構成では利用できないし、98ページで紹介したPromise Ultra66ドライバもまだ開発版のせいか、SMPでは不具合が出るという。また、ベンチマークプログラムのなかにはCPU占有率が100%を越えてしまうなど、各CPUごとの情報を表示できないものがある。

BP6固有の問題としては、HPT366（**写真6-1**）のドライバがカーネルに取り込まれていないことがある。PIIX4Eに内蔵のIDEインターフェイスを用いればすむ話だが、どうしてもという場合、Promise Ultra66の場合と同様、開発版のドライバが用意されている。手順はUltra66の場合と同じだが、起動後にhdparmでステータスを変更してはならない、など制限があるので、十分注意して用いる必要がある。

SMPでどれだけ速くなるのか

せっかくSMPを導入したのだから、できれば2倍速くなってほしいのが人情だ。しかし、ハードディスクやメモリなどの入出力デバイスは1セットしかないので、実際には2倍になるわけではない。それではどれくらい高速になるのだろうか。

よく知られている通り、マルチスレッドに最適化したソフトウェアでなければ単独でSMPの恩恵を受けることは難しい。複数のソフトウェアを同時に使っている場合は別だが、その快感を数字で表わすのはちょっと難しい。もっとも、SMP化で一番恩恵を得るのはこの部分だろう。日常の作業にこそ効果が現われるといえる（負荷がかかっても遅くなりにくいのだ）。

しかし、Linuxカーネルのコンパイルならば、容易にSMPの効果を調べられる。makeにはジョブの数をコントロールする-jオプションが用意されている。LinuxのMakefileは、デフォルトでジョブ数を1としているが、これを増やして同時にジョブを動かすことで、SMPの威力を知ることができるのだ。今回は3に増やして実験してみた。CPUはCeleron-433MHz×2個である。/usr/src/linux/.configは、今回テストしたハードウェア全てのドライバを含んだコンフィグレーションにしたうえで、次のコマンドを実行して消費した時間を計測している。

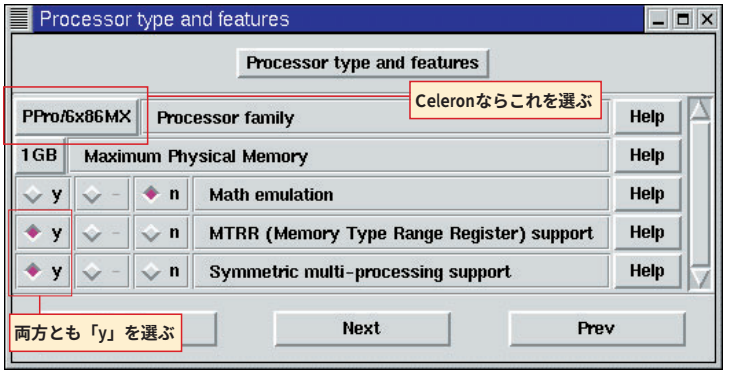
写真6-1　BP6搭載のHPT366
HighPoint Technologies製のUltraDMA/66対応IDEコントローラ。ドライバはまだカーネルに含まれておらず、使いこなしが難しいので、Intelチップセット内蔵（PIIX4E）のコントローラを使うのも手だ。

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

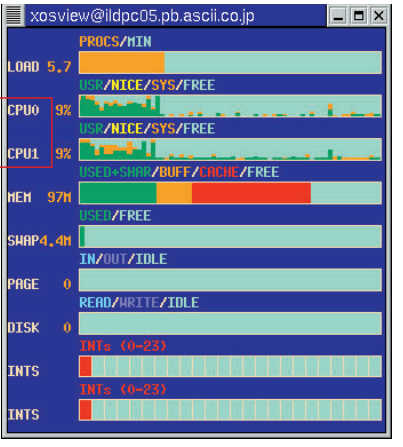
その結果は、

SMP構成：3分52秒
非SMP構成：6分24秒

となった。約1.7倍高速になっている。入出力デバイスが独立していないことを考えるとよい成績だといえるだろう。SMP化の価値は十分にあることがわかる。



画面6-1　カーネルコンフィグレーションのCPUタイプ設定画面



画面6-2　xosviewの表示

グラフィックスカードを替えて3Dを楽しもう

PCパーツの中でもグラフィックスカードは最も早く世代が交代していく部類に属するだろう。おかげで購入するタイミングが難しい。特にLinuxの場合、新しすぎるとソフトウェアサポートが間に合わず、かといって古いとアップグレードにならないということになりやすい。

そんな状況のもとで今回選んだのが、Millennium G400とVoodoo3 2000だ（下の写真参照）。どちらも販売済み製品のなかでは最新だが、登場してから数カ月ほど経っているため、意外に安価に入手できる（特にバルク品だと1万円〜1万5000円と購入しやすい）。それにソフトウェアサポートも充実している。

Milllenium G400はAGP版だが、Voodoo3にはAGP／PCIの両方ともある。AGPの場合、どちらを選ぶかは用途次第だ。後述するようにLinuxで3Dグラフィックスを楽しむなら、現時点ではVoodoo3のほうが面白い。

ところで全般的にPCIグラフィックスカードをラインアップしているメーカーは少なくなってきている。最新製品は特に少ない。このVoodoo3 2000 PCIも化粧箱入りは見つけるのに苦労した（バルク品のほうが探しやすいだろう）あきらめてAGP対応システムに切り替える時期に来ているのかもしれない。

購入したAGP／PCIグラフィックスカード

Matrox Graphics

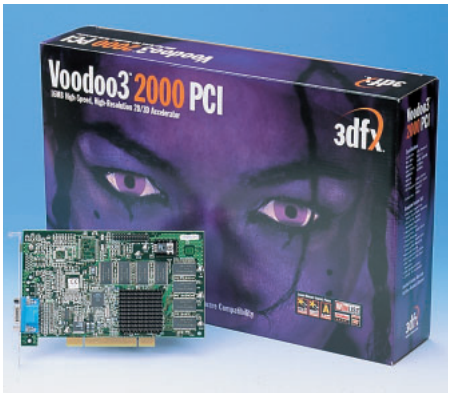
Miillennium G400／DUAL 16MB

オープンプライス

http://www.matrox.com/mga/

購入価格：1万7800円

購入したのはG400シリーズのうち、デュアルディスプレイ出力とメモリ16Mバイトを装備したモデル。シングル出力にすれば1万5000円を切るのでよりお買い得だ。なおAGPのみでPCIには対応していない。写真はinfoMagic扱いのものでWindows用日本語版ドライバが添付されている。代理店によって添付品が異なるので注意しよう。



3Dfx Interactive

Voodoo3 2000 PCI

オープンプライス（輸入品）

http://www.3dfx.com/

購入価格：1万7800円

Voodoo3 2000は最も安価でビデオ出力もないベーシックなモデルだ。搭載メモリは16Mバイト（どのモデルも同じ容量）。AGP版なら日商エレクトロニクス扱いの日本語パッケージがあるが、PCI版は英語版を輸入したものに限られる。そのせいか、写真のような化粧箱入りより、安価なバルク品のほうが比較的見つけやすい。

XFree86をセットアップする

何はともあれグラフィックスカードを交換したら、X Window Systemが表示できなくては話にならない。Millennium G400とVoodoo3の場合、1999年11月時点で最新のXFree86 3.3.5なら問題なく動作するので、これにアップグレードしておきたい。表7-1のXFree86のサイトからダウンロードするか、各ディストリビューションのサイトからRPMなどのパッケージをダウンロードしてアップグレードしよう。

Red Hat Linux 6.1Jの場合、どちらのグラフィックスカードでも、差し替えて起動すれば自動的に検出されて、RedHatのXFree86設定ツールが起動する（**画面7-1**）。自動で設定されるので便利だけど、グラフィックスメモリの容量やディスプレイはユーザーが指定しなければならない。起動前に調べておこう。

今回テストしたMillenium G400、Voodoo3に対応したXサーバはともに完成度は高く、DGAを用いるxawtvも何の問題もなくオーバーレイ表示できた。ただしMillennium G400のデュアルディスプレイ出力やテレビ出力はサポートされておらず、使えない。またチューンナップという点では、グラフィックスカードが十分すぎるほどの2D性能を備えて久しく、XFree86の2D表示では、性能差などまったく体感できない（よほど古いカードからのアップグレードなら、多色高解像度の表示で体感できるかもしれない）。やはり面白いのは3Dグラフィックスだ。

Voodoo3で3Dグラフィックスを楽しむ

Voodoo3はLinuxにおいて、現在も

っとも3Dグラフィックス機能に優れるグラフィックスコントローラといえる。それは3dfxの独自APIであるGlideや後述のOpenGL互換ライブラリで、3Dのハードウェアアクセラレーションが有効に使えるからだ。

■Glideを組み込む

まずはGlideをインストールしてみよう。それにはXサーバーの2D環境が正しく動作しているか事前に確認しておきたい。特に640×480ドットなどゲームで用いられる低解像度が表示できるか注意したい。3Dアプリケーションが解像度を変更しようとしても、もとのXサーバ自体にその解像度の設定がなければ、起動に失敗してしまうのだ。

Voodoo3のGlide用3Dモジュールは3dfxのオフィシャルサイト（**表7-1**参照）からダウンロードできる。Red Hat Linux 6.1Jの場合、以下のファイルが必要だ。

```
Device3Dfx-2.3-1.src.rpm
Glide_V3-2.60-8.i386.rpm
```

これらをダウンロードするには、Webサイトで「Device3Dfx Voodoo3」を検索し、詳細を記したWebページを見つければよい。なお、どちらのモジュールもバージョンが上がってファイル名が変わることがあるので注意が必要だ。また、Glide_V3＊.rpmにはglibc 2.0と2.1それぞれのためのモジュールが提供されているので、あらかじめ/lib/libc＊のファイル名を確認するなどして、glibcのバージョンを調べておこう（上記はglibc 2.1用）。

画面7-1　Xconfiguratorでの自動検出例
これはRed Hat Linux 6.1J付属のXFree86設定ツールで、Voodoo3 2000を自動認識したところ。後で手動設定することも可能だ。

さてインストールだが、**画面7-2**のようにコマンドを実行してrpmファイルを展開後、Linuxを再起動すればよい。インストールが成功したかどうかは、Xの起動後に次のコマンドを実行すると確認できる。

```
# /usr/local/glide/bin/test3Dfx
```

正しくインストールされていれば、3dfxロゴが一瞬だけ表示されるはずだ。これでLinux＋Glide対応の3Dグラフィックスアプリケーションを利用できる。

■OpenGL対応3Dゲームを動かす

Glideの次はOpenGLだ。Linuxには、Mesaと呼ばれるOpenGL互換の3Dグラフィックスライブラリがあり、Linux環境でOpenGL対応ソフトウェアを利用するのによく使われる。特にVoodoo3をはじめとする一部のグラフィックスカードとXFree86を組み合わせると、ハードウェアアクセラレーションでMesaのファンクションを高速に実行できるという特長がある。ここでは、OpenGL対応の3DゲームであるQuakeIII Arenaのテスト版でMesaを試してみよう。

QuakeIII ArenaはWin32やLinuxなど複数のプラットフォームをカバーするマルチユーザー／ネットワーク対応3Dゲームである。試用したのはLinux版QuakeIII Arena Test 1.08だが、評価後に最終テスト版であるQuakeIII Arena Demo Test 1.10がリリースされたので、そちらを使うとよいだろう。Quake III Arena Testには、実行に必要なMesaライブラリも一緒に配布されるので、別途Mesaをインストールする必要はない。とはいっても環境によっては最新版のMesaライブラリが必要になることもある。その場合、表7-1のMesa配布元から最新版のライブラリをダウンロードしよう。

一方、QuakeIII自体は50Mバイト前後もあるアーカイブで配布されるので、一次配布元よりはもっとネットワークに余裕のあるサイトからダウンロードするほうがよいだろう（**表7-1**を参照）。アーカイブはtar.gzとrpmの2種類があるが、rpmを選んでおけば以下のように簡単にインストールできる。

```
# rpm -Uvh <rpmファイル名>
```

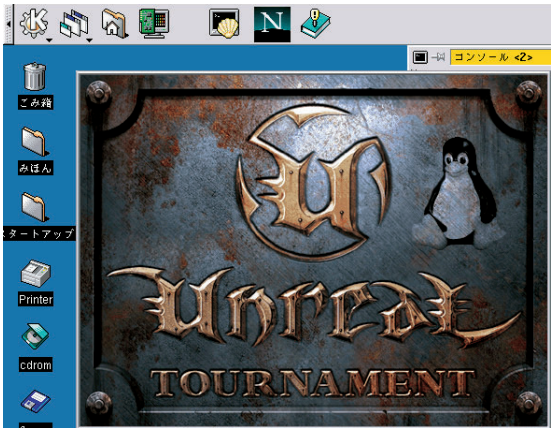
XFree86	http://www.xfree86.org/
3dfxオフィシャルサイト	http://www.3dgamers.com/
Mesa	http://www.mesa3d.org/
Mesaライブラリ（一次配布元）	ftp://ftp.mesa3d.org/mesa/MesaLib-3.0.*
Quake3 Arena（一次配布元）	ftp://ftp.quake3arena.com/
Quake3 Arena Test 1.08	ftp://www.cdrom.com/pub/idgames/idstuff/quake3/linux/old/q3test-1.08-glibc-2.i386.rpm
Quake3 Arena Demo Test 1.10	ftp://www.cdrom.com/pub/idgames/idstuff/quake3/linux/q3demoTEST-1.10-5.i386.rpm
Unreal Tournament	http://unreal.epicgames.com/TournamentVersions.htm

表7-1　使用したファイルや情報

```
# rpm --rebuild Device3Dfx-2.3-1.src.rpm
# rpm -Uvh /usr/src/redhat/RPMS/i386/Device3Dfx-2.3-1.i386.rpm
# rpm -Uvh Glide_V3-2.60-8.i386.rpm
```

画面7-2　Glide用モジュールをインストールするコマンドライン

画面7-4　QuakeIII Arena
Linux版QuakeIII Arena Test 1.08をサーバーモードで起動し、Windows版QuakeIIIからLAN経由で接続し、マルチユーザーでテストした。実際の動作は安定している。



画面7-4　Unreal Tournament
UnrealTournament 3.48 demoの起動画面。Windows版との大きな違いはUTロゴの隣で微笑むTux君か？　こちらも動作は安定していた。

インストール自体はこれで終わりだが、描画パフォーマンスを向上させるおまじないをしておこう。

```
# export FX_GLIDE_SWAPINTERVAL=0
```

これはリフレッシュレートと同期を取らずに画面を描画する指定だ。毎回入力するのは手間なので、ログインスクリプト~/.bashrcなどに上記のコマンドを書き込んでおくといいだろう（最下行に書き加えるだけだ）。

QuakeIII Arena Test Linux版の実行は次のように行う。

```
# cd /usr/local/games/q3test
# ./linuxquake3
```

これでQuakeIIIが起動するはずだ（**画面7-3**）。Xサーバのバージョンによっては、linuxquake3実行時に「+set in_dgamouse 0」というオプションを指定して、DGAでのマウスサポートを無効にしないと正しく動作しない場合がある。問題があったら試してほしい。

QuakeIII Arenaをテストした PCは、CPUがCeleron-300MHzと（3Dゲーム用としては）決して速くはないが、ハードウェアアクセラレーションにより快適にゲームができる速度で動作した。ただ1.08では解像度が640×480ドットより上げられないという問題があった。1.10では解決しているかもしれない。

また、今冬QuakeIIIと並んで注目を集めているUnreal Tournamentにも Linux版のDemoがある（**画面7-4**）。コラムのように、今後3D環境がさらに整備されて、こうした3Dグラフィックス対応ソフトウェアがさらに増えることを望みたい。

Column

3Dグラフィックスは XFree86 4.0に期待？

3dfxのVoodooシリーズはMesaそのものにハードウェアサポートが組み込まれているので、バイナリ形式で提供されているRed Hat Linux 6.xならば、rpm形式のGlideおよびMesaライブラリを組み込むことで、最新のゲームも実行可能な環境を容易に作ることができる。

他のビデオカードの対応状況はどうだろうか。nVidiaはRIVAシリーズに対応したMesa/GLX開発版をリリースしており、http://www.nvidia.com/より必要な情報を入手可能だ。GLX（OpenGL eXtention）とはもともとOpenGLの開発元であるSGIによって開発された技術で、X Window SystemでOpenGLを利用するためのしくみである。SGIがGLXをオープンソースとして提供したおかげで、GLXを利用してハードウェアアクセラレーションを行うドライバが開発されるようになった。

Millenium G400の3DアクセラレーションドライバもやはりGLXベースで、OpenGLライブラリにMesaを用いている（http://glx.on.openprojects.net/）。

XFree86の次期メジャーリリースとなるXFree86 4.0では DRI（Direct Rendering Infrastructure）が導入され、3Dグラフィックスのさらなる高速化が図られることになる。DRIは Precision Insight（ http://www.precisioninsight.com/）を中心に開発されており、GLXおよびMesaもDRIを利用するようになる。またドライバのモジュール化も図られるので、カードベンダによるXFree86対応の強化も期待できる。たとえばATI TechnologiesはPrecision Insightと協力してXFree86 4.0対応ドライバを開発中と表明している。

なお現在、XFree86 4.0のプレリリース版としてXFree86 3.9xを入手可能だ。3dfxのサイトではXFree86 3.9.16のLinux版をrpm形式で配布しているので、Voodoo3を持っていれば比較的手軽に試すことができる。

Column

Linuxノートで USBマウスは使えるか？

タッチパッドなどノートPCの内蔵ポインティングデバイスが使いにくいと感じているのは筆者だけではないだろう。そこでモバイル向けの小型マウスを一緒に持ち歩いて使うわけだが、こうしたモバイル用マウスはUSB接続がほとんどで、Linuxで一般的なPS/2接続は少ない。それに最近のノートPCだとPS/2コネクタがないことも珍しくない。なんとかLinuxノートでUSBマウスを使う方法はないだろうか？

■USBマウスを認識させるには

実は、最新の安定版Linuxカーネルである2.2系列には、すでにUSBをサポートするソースコードが含まれている。とはいってもデフォルトではなく、またExperimental（実験的）サポートでもなく、カーネルのコンフィグレーションファイルを直接編集しなければ、画面8-1のUSB設定メニューすら表示されないのだ。

そこでまず、/usr/src/linux/arch/i386/config.inというファイルをエディタで編集する。テストしたカーネル2.2.12の場合、その175行目に、

```
# source drivers/usb/Config.in
```

という文字列があるので、先頭の「#」を取り除くと、画面8-1のUSBドライバの設定画面が表示されるようになる。なおテスト環境ではIntel製チップセット内蔵のUSBコントローラを使うので、画面8-1ではUHCIを選択している。この項目はPCに合わせて変更する必要があるので注意したい。あとは通常の手順でカーネルをリコンパイルしよう。新しいカーネルで再起動したら、

```
# modprobe usb-uhci
```

とコマンドを入力・実行する。そこでUSBマウスを接続すると、コンソールにさまざまなメッセージが表示され、USBマウスが認識されたことを告げてくれる。

■XでUSBマウスを利用するには

USBマウスをX環境で使うには、そのためにデバイスファイルが必要だ。そこで、

```
# mknod /dev/usb-mouse c 10 32
```

と実行して、キャラクタデバイスのメジャー番号10番（miscデバイス）、マイナー番号32番（USB mouse）というデバイスを作成する。

次はXFree86の設定だ。/etc/X11/XF86Configなどの設定ファイルでは、PS/2マウスとして指定すればよい。PS/2マウスを使うXFree86の設定がすでにあるなら、デバイス名を/dev/mouse（あるいは/dev/psaux）から/dev/usb-mouseに変更すればすぐに使えるようになる。ただし、Xの起動前にUSBマウスが認識されていないとエラーが発生してしまう。また内蔵ポインティングデバイスとUSBマウスを併用したいなら、リスト8-1のようにXinputセクションを設定ファイルに追加しよう。

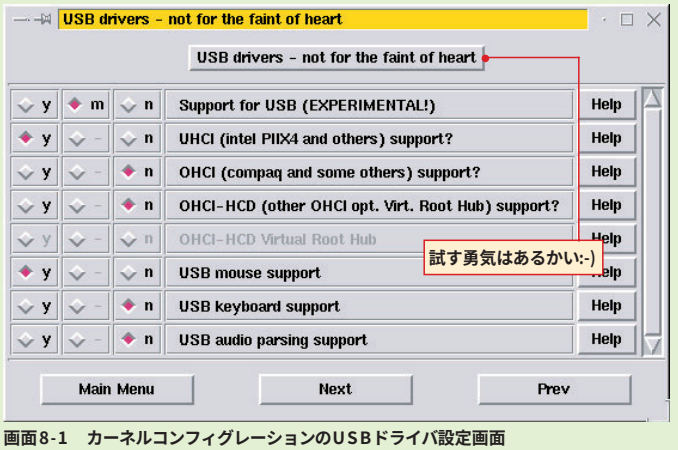
■ホイール機能はカーネル2.4待ち!?

あとはマウスのホイール機能が使えれば文句なしだが、残念なことに写真8-1のマウスやマイクロソフト製IntelliMouseで実験した限りでは、カーネル2.2系列では動作しなかった（第3ボタンとしては使える）。どうやらカーネル2.2.xのUSBドライバにバグがあるのが原因らしく、カーネルを最新の開発版である2.3.28にアップグレードしたらホイールを利用できた。とはいえ、開発版カーネルを常用するのはお勧めできない。カーネル2.3系列ではマウス以外のUSBデバイスもサポートされているので、今は安定版の2.4を期待して待とう。

▼リスト8-1　/etc/X11/XF86Configファイルの変更箇所

```
        :
# ************************
# Pointer section
# ************************

Section "Pointer"
  Protocol  "PS/2"
  Device    "/dev/psaux"
        :
EndSection

Section "Xinput"
  SubSection "Mouse"
  DeviceName "Second Mouse"
  Protocol "PS/2"
  Port "/dev/usb-mouse"
  AlwaysCore
EndSubSection
        :
```

併用するならこのブロックをここに挿入する



画面8-1　カーネルコンフィグレーションのUSBドライバ設定画面

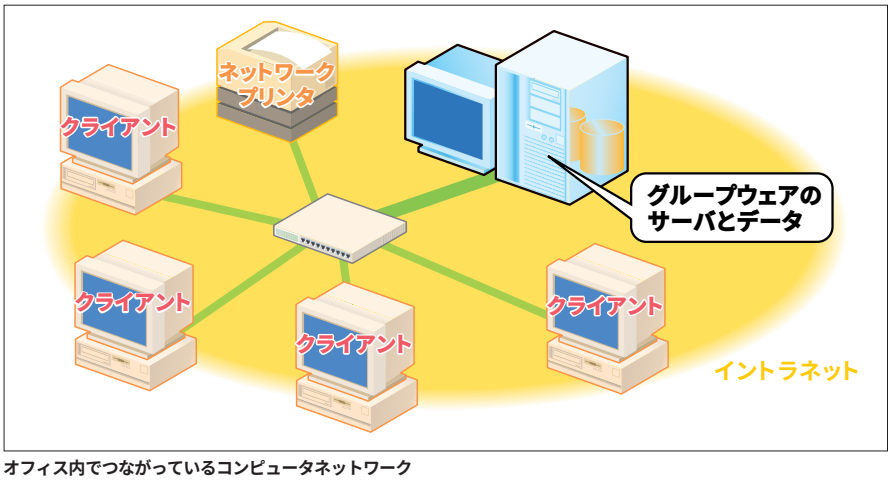
写真8-1　テストしたUSBマウス
サンワサプライのモバイルマウス（シルバー、品番MA-MBUSBSV）。実売価格は3200円前後だが、ホイールのない製品ならもっと安い。小型でケーブルも短く、持ち歩くには便利だ。

Webでグループ管理
サイボウズ(Linux版) Office3

文：上間 俊雄
Text：Toshio uema

オフィスで仕事をするときに社内の情報共有が必要だ。これを怠ると、各自がてんでばらばらに作業をしてしまい、だれがどこで仕事をしているのかわからないなど、仕事の効率が落ちてしまう。たとえば、上司に休暇の許可を取っていたとしても、他の社員たちには知らされてないといったようなことだ。これでは、仕事のスケジュールが立てられなくなってしまう。オフィス内で縦のつながりがあっても、横や斜めのつながりが希薄になっていては、効率が良いとはいえない。

このように書くとオフィスの情報共有があまりなされていないようだが、昔からオフィス内で情報は共有されている。どこのオフィスにも社員の予定が書かれたホワイトボードがあり、これを見れば、だれがどこで何をしているかということがひと目でわかるようになっている。しかし、部署の全員が出払っていたりして、他の部署の社員が電話を取ってしまったりすると、わざわざこのホワイトボードを見にいったり、だれも出ない内線電話をしたりとこれも効率が悪い。最悪、お客さんからの電話をたらいまわししてしまうかもしれない。これでは、せっかくビジネスチャンスをも失いかねない。



オフィス内でつながっているコンピュータネットワーク
すべてのデータはサーバにあり、クライアント（ユーザー側）はサーバからデータを参照する。

そこで、コンピュータネットワークを利用して、オフィス内で情報共有をすることにした。コンピュータネットワークは情報共有用のサーバがあれば、オフィス内のどこからでもアクセスして、情報を参照できるのだ。情報やデータはすべてサーバ上にあり、情報が歪められたりすることがない。

これがグループウェアの始まりである。グループウェアとは、オフィス内のコンピュータネットワークと、そこにつなげられたコンピュータを利用して情報共有するための専用のソフトウェアを指す。具体的には、情報共有のためのサーバソフトウェアと、そのサーバにアクセスするためのクライアント（ユーザー側）のコンピュータにインストールされたソフトウェアをまとめていう。

Webグループウェアの登場

最近ではオフィスへのコンピュータの導入が進み、ひとり1台の時代になってきている。さらに、インターネットの普及とともにオフィス内もネットワーク化されている。そのため、これまで一部の企業にとどまっていたグループウェアの利用が一般化してきた。これで、仕事の効率が上がるはずだが、新たな問題が2つ起こってきた。どちらも突き詰めればコストの問題と言える。

まず、せっかくグループウェアを導入しても利用しない、あるいは利用できない人が存在する。グループウェアで情報共有ができるといっても、1人でも利用できなければ意味がない。この原因は、アクセスのわずらわしさや、複雑な設定操作についていけないということにある。グループウェアを利用するために、分厚いマニュアルを見ながら仕事をしたり、使い方を何度も教わっていたりすれば仕事の効率も上がらない。それに、サポートするほうも大変だ。

また、グループウェアのソフトをユーザーのマシンにインストールするのにかかる手間やコストの問題もある。ユーザーが利用しているマシンのOSが新しくなると、グループウェアの総入れ替えという作業も出てくる。これらにかかるサポートの手間とコストは馬鹿にならない。オフィス内の全員で情報共有をできるようにするためには、ユーザーが簡単に利用できて、なおかつサポートにかかるコスト面をできるだけ抑えなくてはならないのだ。

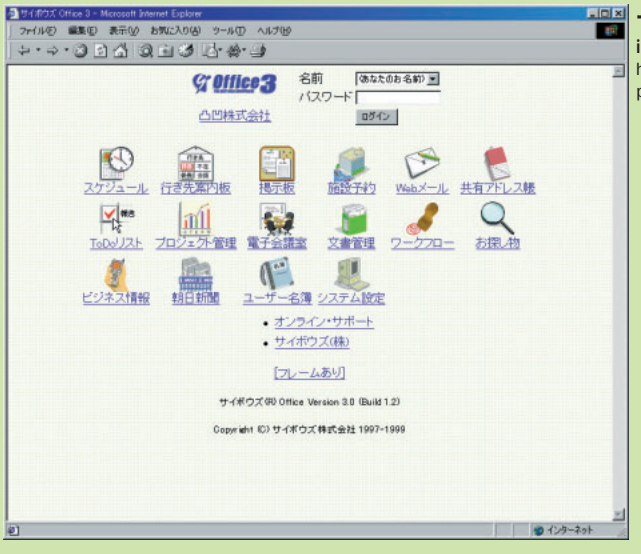
そこで、登場してきたのが「Webグループウェア」である。現在のオフィスのコンピュータネットワークは、インターネットやイントラネットの普及でTCP/IPベースのネットワークに移行されつつあり、Webサーバを立てる環境がすでにできている。そこで、オフィス内にWebサーバを構築して、ユーザーはWebサーバにあるグループウェアをWebブラウザ上から利用することが考えられた。このWebサーバにあるグループウェアを「Webグループウェア」という。

ユーザーはWebブラウザを利用しているため、使い慣れたWebブラウザのインターフェイスですんなり使うことができるのだ。ユーザー側にグループウェアの専用ソフトウェアを導入しなくて済むし、ユーザー側のマシンやOSなどにも左右されにくい。どのOSにもWebブラウザがインストールされているからだ。

グループウェアのサーバといえば、これまでWindows NTの独壇場であった。それは、オフィス内で利用されているネットワークがWindowsのネットワークであったし、ユーザー側で利用されているOSが、ワープロソフトや表計算ソフトが揃っているWindows 95／98だったからだ。もちろん、そこではサーバ側、クライアント側それぞれにグループウェアを導入する。しかし、グループウェアがWebサーバで動くということになれば、インターネットの申し子とも言われてきたLinuxに一日の長があるかもしれない。もちろん、Linuxサーバを導入するに当たってのコストはWindows NTよりも安い。また、GUI化されたLinuxをクライアントにしても良いだろう。

サイボウズOffice3 Linux版

サイボウズOffice3 Linux版（以下サイボウズ3）は、Webグループウェアのけん引役として早くからLinuxをサポートするなど、現在まで多くの企業で導入、運用されてきた実績がある。また、Linux以外にもWindows 95／98、Windows NTをはじめ、FreeBSDやSolarisなどのUNIX系サーバ、そしてオールインワンサーバとして注目を集めたCobalt Cubeにも対応している点も特徴だ。現在のバージョンはサイボウズOffice3となっていて、旧バージョンのサイボウズOffice2よりも、インストールのしやすさや、ユーザーごとのカスタマイズ性がかなり進



サイボウズ
info@cybozu.co.jp
http://cybozu.co.jp/cb3/support/question.html

歩している。

サイボウズ3は、10種類のグループウェア機能が用意されている。特に今回注目を浴びているのが、「パーソナライズ」という機能だ。ユーザーはサイボウズ3にログインしたら、ユーザー専用のWebページが提供される。この専用Webページはユーザーが自分だけのオリジナルページとして作成することができる。これは、Yahoo! Japanの「My Yahoo!（http://my.yahoo.co.jp/）」や、Exciteの「マイエキサイト（http://www.excite.co.jp/）」のような感覚だ。たとえば、見落としがちな掲示板の書き込みなどを考慮して、メッセージを新着順に表示したり、更新されたものだけを表示するなど、自分の注目したいものを集めたページにカスタマイズできる。昨今、インターネットのポータルサイトでよく見かけるこのような機能をいち早く取り入れた点は非常に新鮮だ。ほかにも、サイボウズ3ではWebメールの「サイボウズ Webメール 3」や、稟議などの申請ができる「ワークフロー」などが利用できるようになった。

サイボウズ3はサイボウズ社のサイトからダウンロードして利用する（βバージョンを付録CD-ROMに収録）。その前にWebサーバのApacheがインストールされているLinux環境が必須だ。なお、固定されたIPアドレスが割り当てられているLinuxサーバを利用しよう。今回は、LASER5 Linux 6.0と、Apacheを用いてインストールすることにする。Apache以外でも動作する可能性はあるのだが、どのLinuxのディストリビューションでもデフォルトのWebサーバとして採用されているApacheを利用したほうが問題は起きにくい。

サイボウズ3はApacheと連動して動作するので、Apacheで決められたディレクトリにインストールしなければならない。そこで、事前にドキュメントルートのディレクトリ、CGIを格納するディレクトリを調べておこう。サイボウズ3では、ドキュメントルートのディレクトリが「/usr/local/www/data/」、「/home/httpd/html/」のどちらか、CGIを格納するディレクトリが、「/usr/local/www/cgi-bin/」、「/home/httpd/cgi-bin/」のどちらかでない場合には、手動で入力する必要があるので、事前に調べておくことが必要だ。なお、LASER5 Linux、Turbo Linux、Vine LinuxなどのRed Hat系のLinuxディストリビューションでは、何もせずそのままインストールできる。

しかし、Apacheをソースからコンパイルして、デフォルトの設定のままインストールしたときや、Debian GNU/Linux、Plamo Linuxのディストリビューションで使用するときには、それぞれのディレクトリが違うので注意しよう。各ディレクトリは、Apacheの設定ファイル「srm.conf」の「DocumentRoot」「ScriptAlias」の行で確認できるはずだ。

サイボウズ3は、「http://cybozu.co.jp/」より、ファイルをダウンロードできる。また、今月号の付録CD-ROMにβ版を収録している。ダウンロードしたファイルは、「/tmp」などの適当なディレクトリにコピーしよう。このあとのインストール作業はすべてコンソールで行う。インストーラは日本語でも表示されるので、日本語が表示できるようにkonや、GUI環境ではktermを使って作業を進めよう。ダウンロードしたファイルはtar＋gzip形式で圧縮されているので、GNU tarで解凍する。

```
$ tar zxvf cbof30jal.tar.gz
```

解凍が終われば、root権限でインストーラを起動しよう。「cb3setup」というシェルスクリプトだ。

```
$ su
Password:

# ./cb3setup
```

はじめに日本語が表示されているか聞いてくるので、表示されているなら［Y］キーを押して、［Enter］キーを押す。次にソフトウェアのライセンスをよく読んで、同意したら、インストールを続行しよう。CGIを格納するディレクトリ、ドキュメントルートのディレクトリを指定する。Red Hat系ならそのままで［Enter］キーを押して続ける。「製品の選択」では、3つのパッケージか、単体でオプションを指定できる。「サイボウズ ワークフロー 3」のみ単体での購入となるので、インストールするときは忘れずに指定しておこう。あとは対話式で答えるだけでファイルがコピーされ、インストールが終了する。

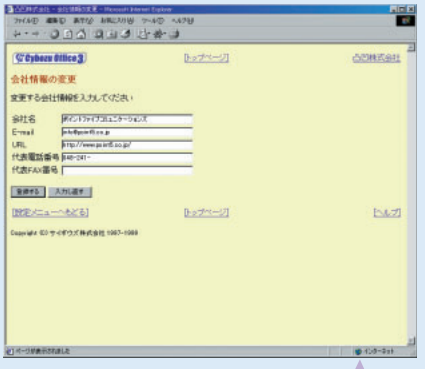
インストールが終了したら、ブラウザを起動して実際に実行してみよう。サイボウズ3のCGIを直接指定する。LASER5 Linux6.0の場合には、

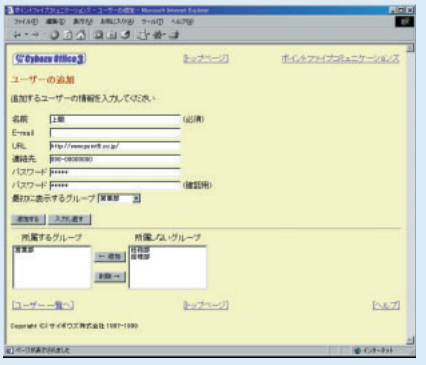
```
http://ホスト名/cgi-bin/cb3/office.cgi
```

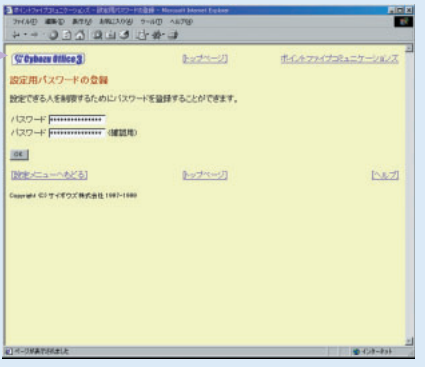
となる。ホスト名には実際のサーバ名が入る。

起動と初期設定

Webブラウザを起動して、URLを入力すると、サイボウズ3が起動される。初めて起動したら、まず、トップページの「システム設定」から「サイボウズ Office 共通設定メニュー」の内容を変更してみる。ここではパスワード設定とユーザー、グループの追加、会社情報を設定できる。さらに「ページの設定」では、ユーザーがログインするときに表示されるメニューの設定ができる。最後に「管理・運用」の「設定用パスワードの登録」で、設定変更時のパスワードを設定する。これでひと通りの設定は完了だ。あとは、ユーザーがおのおのアクセスしてログインすればよい。





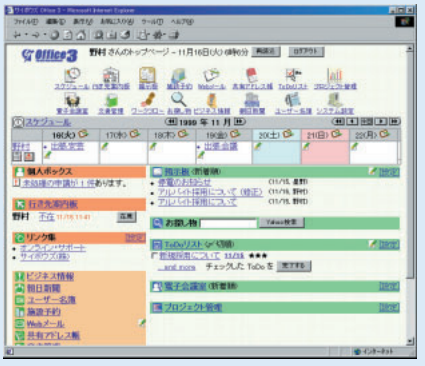


インストールが終了したら、まず管理者が初期設定を行う。会社の情報やグループとユーザーをそれぞれ追加していこう。管理用のパスワードを設定することも忘れないように。

ユーザーログイン後のトップページ

ユーザーがログインするとユーザー専用のトップページ画面となる。この画面は、今回のバージョンから新たに取り入れられたパーソナライズ機能で、ユーザーに関係しているものだけが揃えて表示されている。もちろん、ユーザーごとにトップページのレイアウトをカスタマイズすることができる。

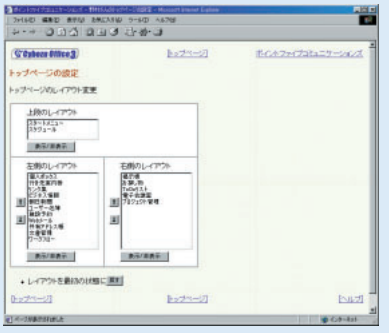
Webグループウェアの特徴として、インターネットへの接続と同じTCP/IPと、ホームページを見るためのWebブラウザを利用しているため、特にインターネットとの連携がはかりやすいようになっている。まず、「リンク集」では、よく利用するWebサイトを登録でき、リンクをクリックするだけで、そのホームページに移動できる。



それほど利用しないが、あると便利なWebサイトが「ビジネス情報」にまとめられていて、企業情報を調べたり、時刻表や列車の経路検索、ニュース・天気予報のWebサイトにリンクされている。また、「お探し物」の検索フォームに調べたい事柄を入力すると、Yahoo! Japanに移動して、インターネット上の検索結果が表示されるようになっている。

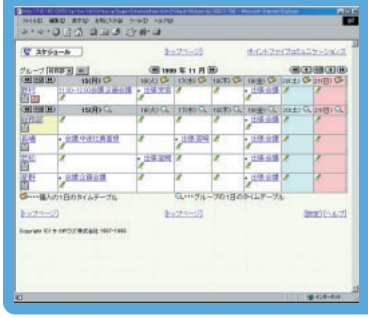
トップページのカスタマイズ

ユーザーごとにトップページのレイアウトをカスタマイズすることができる。カスタマイズはユーザーのトップページの下にある「このページの設定」から行う。必要な項目を並べ替えたり、項目の「表示／非表示」の設定などができる。ユーザーパスワードの変更はユーザーのトップページの「パスワードの変更」から行う。



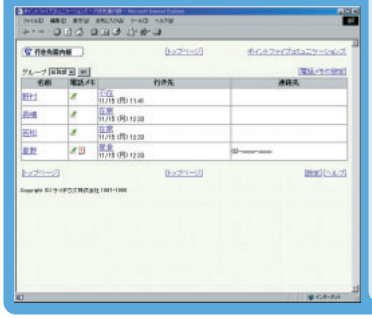
ここまでできるサイボウズ3

スケジュール



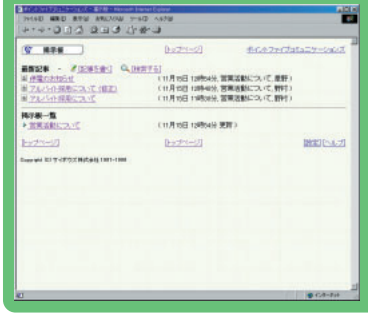
個人のスケジュールは、ほかの人たちにはわからないもの。しかし、グループウェアを利用することで、だれがいつ、どこで、何をしているのかがひと目でわかり、スケジュールを組みやすくなる。

行き先案内板



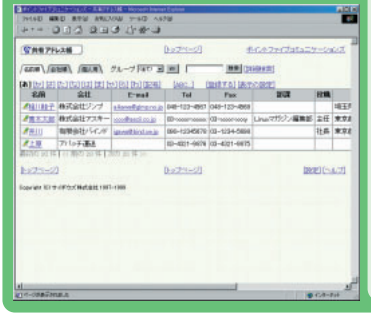
行き先案内板は、部署ごとにあるホワイトボードのような感覚で利用できる。それに加えて、他のグループ（部署）も参照できるため、違う部署に電話がかかってきても、連絡が取りやすい。

掲示板



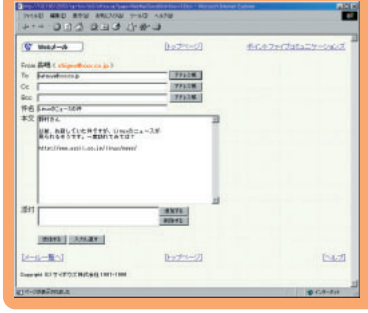
掲示板は告知などの一方的な通知に利用する。本来は紙に印刷してからそれぞれの部署に配布されるということが多いが、掲示板の機能を利用すれば瞬時に告知することができ、紙のコストも省ける。

共有アドレス帳



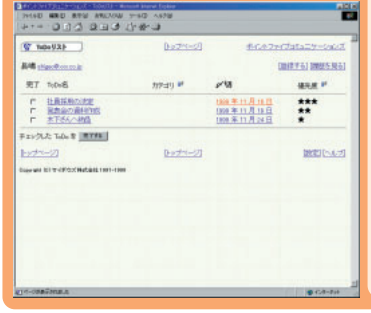
オフィスではアドレス帳を共有したほうが便利な場合が多い。たとえば、外出中の社員宛てに緊急の用件が入った場合に、別の社員がたとえ面識がない取引先に対しても電子メールやFAX、宅配などの対応ができるのだ。

Webメール



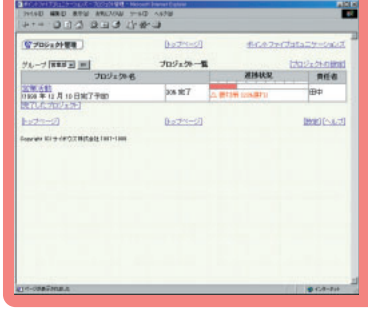
WebブラウザからEメールを利用できる。専用メールクライアントのような高度な機能はないが、Webベースであるためどの端末からでもメールの読み書きができる。「共有アドレス帳」の登録アドレスへ簡単にメールできる。

ToDoリスト



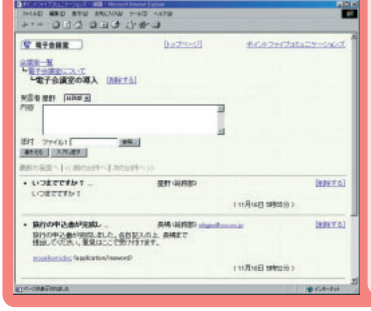
ToDoリストは現在進行中の仕事の優先度をひと目でチェックできる機能だ。Webグループウェアとしての特徴として、データの共有があるが、ToDoリストでもグループでの共有ができ、部署内の状況が把握しやすい。

プロジェクト管理



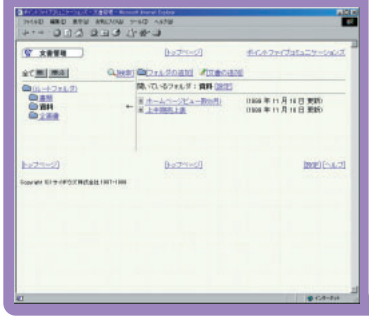
プロジェクトを設定しておくことで、プロジェクトの進捗状況を把握できる。ユーザーは個々に活動テーマを設定して、さらに細かな期限や達成度を設定できる。また、完了したプロジェクトも調べることができる。

電子会議室



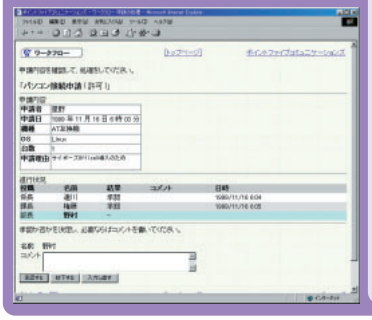
「掲示板」機能が一方的な通知として使うのに対して、「電子会議室」は議論する場所として利用する。ユーザーは議題を設定して、各ユーザーが投稿できるようになっている。議題はスレッド構造になっている。

文書管理



企画書や資料、ドキュメントなど、共有して使えば便利な文書をディレクトリ構造で管理する機能。さらに文書ことにキーワードを設定しておくことで、文書を検索することができる。

ワークフロー



組織が大きいと、許諾や稟議に時間がかかるが、「ワークフロー」の機能を利用して早急に返事をもらうことができる。ユーザーはこの内容を詳しく設定でき、許可を取るべきユーザーに申請する。

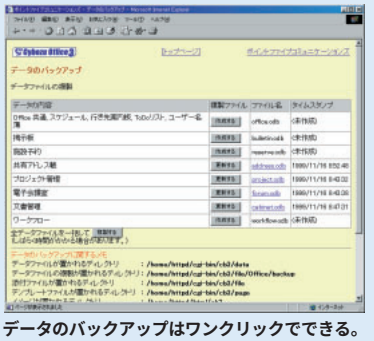
専門的な知識がなくても管理・運用できる

グループウェアはユーザーがデータを共有して利用するため、必ず管理者を置いて、管理しなければならない。この管理者が行う作業は、ユーザーやグループのアカウント管理と、バックアップだ。サイボウズ3では、この管理機能も同じWebブラウザから利用する。

ユーザーやグループの設定は初期起動したときと同じく「システム設定」の「サイボウズ Office 共通設定メニュー」で利用でき、フォームに入力するだけでユーザーとグループの追加や削除、変更作業が行える。

データのバックアップは管理者として一番気を付けなければならない作業だ。これを怠るといつデータがなくなるかわからないので、だれも利用しなくなる。「システム設定」の「データのバックアップ」で行うことができ、1クリックで全データが一括してバックアップされる。管理者は退社前や出社時に、このボタンを押すだけだ。また、バックアップしたデータを別のマシンやメディアにコピーするときには、Webブラウザでファイルをダウンロードする手順で、リンクされたファイルをクリックするだけでダウンロードできる。

Webメールの機能を利用しているときには、ユーザーのメールのデータがすべてサーバ上にあるので、こまめにバックアップを行ったほうがよい。これは、「データのバックアップ」からはできないので、cronを設定してバックアップしておこう。または、FTPやSambaを利用して、ローカルのMOやCD-RWにコピーするのもいいだろう。



データのバックアップはワンクリックでできる。

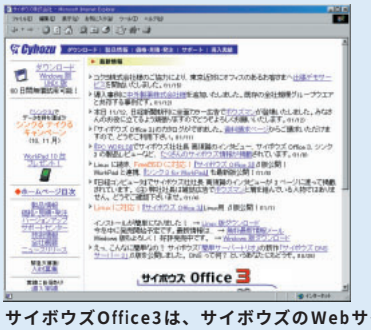
サイボウズ3の購入方法

サイボウズ3は、Webサイトを通じて販売されているため、パソコンショップで購入することはできない。購入してもパッケージやマニュアルが郵送されるのではなく、インターネットからのダウンロードか雑誌のCD-ROMからインストールして使う。60日間の期間内は全機能を無償で試すことができ、ライセンスキーを購入することで、利用期間の制限がなくなる。

サイボウズ3は機能を単体で販売しているのが特徴で、それぞれライセンス数50ユーザーからとなっている。複数の機能をまとめたパッケージは3種類用意されていて、「サイボウズ Office パック EX 3」は、「ワークフロー」以外の機能が利用でき、50ユーザーで使用する場合は19万8000円。

50ユーザーも必要ない場合のために「サイボウズ Office SOHO 3」というSOHOパッケージが用意されており、こちらは10ユーザー7万9800円と非常にリーズナブルだ。

また、スケジュール、行き先案内板、掲示板、施設予約、Webメールといったよく利用する機能だけを集めた「サイボウズ Office パック 3」は、50ユーザーを9万9800円で購入できる。なお、「ワークフロー」の機能のみ単体で購入するようになっている（9万9800円）。前バージョンからのバージョンアップの場合には割引販売も実施されている。



サイボウズOffice3は、サイボウズのWebサイト（http://cybozu.co.jp/）からダウンロードして利用する。

複数OSを1台にマルチブート完全制覇!

文：竹内充彦、竹田善太郎
Text : Michihiko Takeuchi , Zentaro Takeda
Photo : Shuichi Mito

サーバーマシンとしての利用を除いて、Linuxインストール時に必ずといっていいほど直面する問題が、Windowsなど、それまで使っていたOSとの棲み分けである。潔く「Linuxさえブートすればそれで OKさ」といってのけられる達観したユーザーなら話は別だが、一般的にはなかなかそうもいかない。仕事のデータの受け渡しがExcel形式だったり、お気に入りのゲームがWindows版だったりと、理由はさまざまだが、とにかく現在稼動しているWindows環境はどうしてもそのまま残しておきたい。とはいえ、PCをもう1台調達するというのは、予算的にも設置面積的にもそうそう余裕がない。そんな都合のいい要求を満たしてくれるのが「マルチブート環境」だ。

失敗しないマルチブート環境構築から、さらに快適なマルチブート環境の構築、マルチブート環境トラブルシューティングなど、マルチブート完全制覇を目指そう！

マルチブート環境とは？

1台のパソコンで複数のOSを使うと言っても、同時にブートするわけではない（当たり前だ）。ブート時にどのOSを使うかを選択可能にするのだ。

MS-DOSの時代であれば、フロッピーディスクからブートするのが当たり前だった。したがって、システムを起動するたびにフロッピーディスクを差し替えれば、異なるOSをブートできた。しかし、昨今のOSは大型になっており、フロッピーだけでブートするものは少なくなっている（Linuxもうまく収めればフロッピーディスク1枚からブートするのだが…）。

ハードディスクもフロッピーディスクのようにシステム起動前に差し替えるという方法がある。リムーバブルハードディスクを装備すればいい。しかし、これは専用のソケットやら何やらを備えなければいけないし、デスクトップ型PCでベイに余裕がない場合は、外付けユニットを調達しなければならない。

そこで、登場するのがブートマネージャを使ったマルチブート環境である。あらかじめ接続されているハードディスクに、複数のOSをインストールしておき、起動時にソフトウェアでOSを選べるようにするというものだ。

通常、PCを起動するとBIOSが自動的にハードディスク等のブートデバイスの先頭を読みに行く。この先頭部分はマスターブートレコーダ（MBR）と呼ばれている。ここにInitial Program Loader（IPL）と呼ばれる小さなプログラムが書き込まれているのだ。いったんこれをメモリ上に読み込み実行する。IPLには、OSのブートプログラムがディスクのどこにあるかがわかっていて、それをメモリに読み込み実行することで、OSがブートがする仕組みになっているのである。

このとき、ブートするOSを複数の中から選べるように作られたプログラム（ブートマネージャまたはブートセレクタなどと呼ばれる）を、このMBRに書き込んでおくことで、起動するたびに、任意のOSを選んでブートすることが可能になるのだ。

ハードディスクドライブが1台しか接続されていなくても心配することはない。パーティション分割という技を使い、ドライブの中身を分割して使用することで、擬似的に複数台のハードディスクとして扱うこともできるのだ。

本特集では、このタイプ、つまりMBRにブートマネージャを書き込むタイプのマルチブート環境構築について解説していく。

マルチブート環境を構築しよう！

「マルチブート環境を構築する」と言うと、なにやら難しいことのように聞こえるかもしれないが、実はそんなことはない。Windowsの環境にLinuxを追加インストールする場合なら、なおさらである。

なぜなら、多くのLinuxディストリビューションパッケージでは、初めから、そうした環境が想定されており、最低限必要な構築ツールが付属しているのだ。したがって、ハードディスクにLinuxをインストールするだけの空き容量さえあれば、すぐにでも構築できるのである。

本特集でも、まずLinuxディストリビューションパッケージのインストールCDのみで構築する方法を解説する。

これからLinuxをインストールするユーザーはもちろん、すでにインストールしたが、ブートセレクトをもっと快適にしたいユーザーにも役立つはずだ。とくにハードディスクドライブとパーティションについての基礎知識については、詳しく解説しているので、参考にしてほしい。

インストールCDだけでの構築では、準備に手間がかかったり、ブートセレクタが使いにくいと感じたりするかもしれない。そういった問題を解決するために、最近では、マルチブート環境支援ユーティリティなるものが市販されている。これを使えば、より複雑なマルチブート環境を、より簡単に構築することができる。本特集ではこうしたユーティリティも紹介していく。もし、作業がわずらわしいと感じる、あるいはより快適な環境を望むなら、こうした製品を使ってみるといい。

もし、あなたのハードディスクドライブに余分な空きスペースがあるなら、迷うことはない！ 今すぐ、マルチブート環境を構築しよう！

Linuxで構築するマルチブート環境

マルチブート環境の代表的な例として、Windowsが稼動しているPCに、Linuxを追加インストールする方法を紹介する。これは、標準的なLinuxディストリビューションパッケージだけで構築できる、最も手軽なマルチブート環境だ。

それには、現在使っているPCのハードディスクに、Linuxをインストールする領域を用意しなければならない。そこで、まずはじめに、PCのハードディスクについての基本的な知識を押さえておこう。

ハードディスクの接続

パーツを集めてPCを組み立てた人ならば、自分のPCに接続されているディスク装置については知っているはずだ。問題なのは、出来合いのPCをショップで購入してきて、未だにケースの蓋を開けたことがないという人の場合である。まず、PCとディスク装置の関係の基礎を知っておこう。

PC互換機は、ディスク装置の接続用として、IDE規格に準じたコントローラを2系統備えている。プライマリIDEコントローラとセカンダリIDEコントローラである。IDEコントローラ1系統につきディスク装置を2台まで接続できる。この2台はマスタードライブとスレーブドライブと呼ばれる。したがって、プライマリ／セカンダリの2つのコントローラに、それぞれマスター／スレーブのディスク装置を接続すれば、合計4台のディスク装置を接続できることになる。プライマリIDEコントローラのマスター～セカンダリIDEコントローラのスレーブまでを、接続されている順に、ドライブ0～3と表現する（**図1**）。

また、これとは別にSCSI規格に準じたコントローラを搭載していれば、さらに7台～15台のディスク装置を接続することができる（**図2**）。

現在、ショップで売られているPCのほとんどは、IDE接続されたハードディスク1台と、同じくIDE接続されたCD-ROMドライブ1台から構成されている。したがって、筐体に余裕のあるモデルならば、先ほど述べた理由から、IDEで接続するディスク装置をあと2台増設できることになる。

SCSI接続のディスク装置は、IDE接続のそれに比べてCPUに負荷をかけずに高速に読み書きできることもあり、サーバなど、特定の用途のハイエンドマシンに採用されていることが多い。また、SCSI接続用のディスク装置はIDE接続用のものに比べて高価であり、IDEとは別にSCSIコントローラが必要なことから、一般的な用途のPCには、あまり採用されていない。

さて、それでは個々のディスク装置であるハードディスクドライブの中身はどのよう使われているのだろうか？

パーティションとは？

ハードディスクドライブは、その中身を複数の領域に分割して利用できるようになっている。このように、複数に分けられたディスクの個々の領域を「パーティション」と呼び、1台のドライブに複数のパーティションを設定する作業のことを「パーティション分割」と呼んでいる。1台のドライブを複数の「パーティション」に分割すると、OSやアプリケーションからは、あたかも複数のドライブが存在しているかのように見えるようになる（**図3**）。

せっかくの大容量のハードディスク

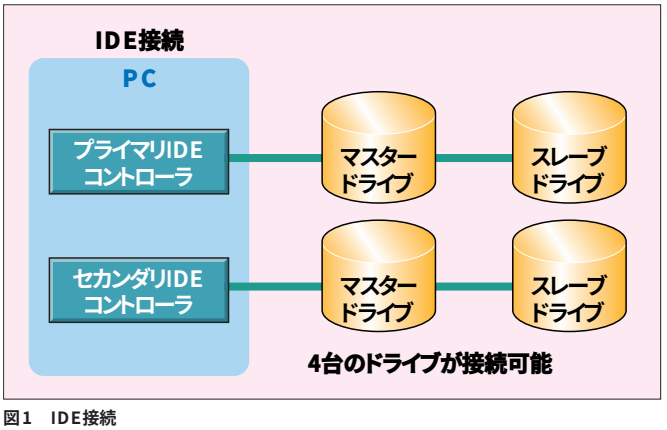


図1　IDE接続

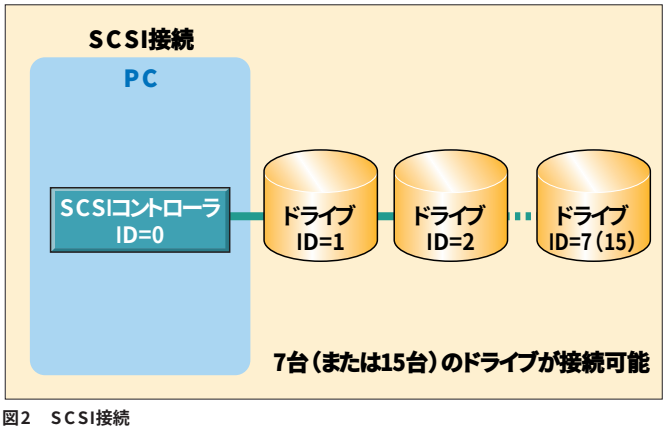


図2　SCSI接続

を、なぜわざわざ小さなドライブに分割しなければならないのか、という疑問を抱くかもしれない。これはもっともな疑問で、1台のハードディスクをまるごと1台のドライブとして使ったほうが、混乱も少なく、使いやすい場合もある。

しかし、LinuxやWindows 98を含む、現在使われているOSの多くは、取り扱えるドライブの最大容量に制限があったり、極端に大きなドライブを使用したりする場合、ドライブの記憶領域の利用効率が悪くなるなどの事情がある。このため、市販のWindows PCの多くは、出荷時にOSの起動用のドライブ（Cドライブ）と、もう1つの自由に使用できるドライブ（Dドライブ）の2つのパーティションに分割されていることが多い。

このほか、本特集で扱うマルチブート環境、すなわち1台のPCにWindowsとLinuxなどの複数のOSをインストールして、場面に応じて使い分けたいような場合、それぞれのOSが使用するドライブは専用の形式でフォーマットしなければならないため、必然的に1台のハードディスクを複数のパーティションに分割することになる。

パーティションの情報は、ハードディスクの先頭にあるパーティションテーブルと呼ばれる領域に記録され管理されている。

パーティションの種類

パーティションはハードディスクを区切った「枠」のようなものであり、その中身をどのように使うかについては、そこにインストールされるOSが決めることである。しかし、あるOSが起動したあとで、ディスク上のパーティションをマウントしようとする場合、そのパーティションの中身（フォーマット形式）が、そのOSでサポートされているかどうかが、あらかじめ分かるようになっていなければならない。このため、パーティションテーブル中の各パーティションのエントリには、そのパーティションがどのようなOSでどのような形式に使われているものなのかを示す「タイプ」情報（パーティションタイプID）が含まれている（**表1**）。

通常、OSに付属するパーティション設定プログラム（DOSのFDISK.EXE、Linuxのfdiskなど）が、パーティションの領域確保とともに、パーティションの「タイプ」を設定するが、自分のOSがサポートしていない「タイプ」のパーティションの設定を変更したり、追加・削除したりすることができないことがある。たとえば、DOSやWindows 95／98に付属しているFDISK.EXEでは、FATパーティション以外のパーティション情報を操作することはできない。しかし、Linuxに付属するfdiskコマンドでは、Linuxのext2形式のパーティションだけでなく、DOS／WindowsのFAT／FAT32、OS2のHPFSなどのパーティション情報を操作できる（ただし、パーティションの中身のフォーマット処理については、たとえばDOSやWindows 95／98に付属しているFORMAT.COMのような、別のユーティリティが必要になる）。

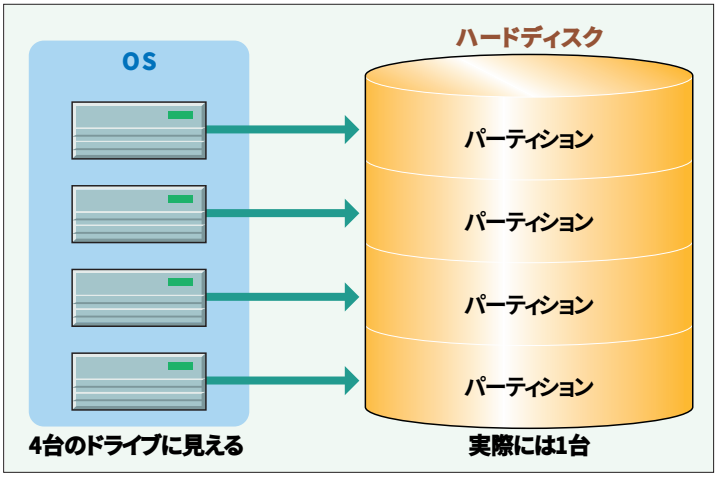


図3　パーティションの概念

ID	パーティションタイプ
01	FAT12（DOS）
04	FAT16 32Mバイト以下（DOS）
06	FAT16（DOS）
05	拡張領域（DOS）
07	HPFS（OS/2）/NTFS（WindowsNT）
0A	OS/2のブートマネージャ
0B	FAT32（Windows9x）
0C	FAT32 LBA（Windows9x）
0E	FAT16 LBA（Windows9x）
0F	Windows95/98の拡張領域
3C	PartitionMagic
82	Linuxのswapパーティション
83	Linuxのファイルシステム（ext2）
85	Linuxの拡張領域
86	NTFS（Windows NT）
87	NTFS（Windows NT）
A0	ハイバネーション領域
A5	BSD/386のファイルシステム
A6	Open BSDのファイルシステム
A7	NeXT STEPのファイルシステム
B7	BSDIのファイルシステム
B8	BSDIのswap領域
EB	BeOSのファイルシステム

表1 代表的なパーティションタイプID

■DOS／Windows用のパーティション

DOSやWindows 3.1／95／98では、「FAT」と呼ばれるディスクフォーマット形式が用いられている。FATには、Windows 95 OSR2以前に使われていた「FAT12」「FAT16」と、Windows 95 OSR2以降で使えるようになった「FAT32」といった3種類のフォーマットが存在する。このうち、FAT16については、扱えるドライブの大きさが、最大2Gバイトに限られているため、現在ではほとんど使われることはない。

DOSやWindows 9xでは、1台のディスク装置上に、1つの「基本領域」（プライマリパーティション）と1つの「拡張領域」（拡張パーティション）を確保できる。基本領域は、それ全体を1つの「ドライブ」としてフォーマットするが、拡張領域については、その中に複数の「論理ドライブ」を割り当てて、それぞれを単一のドライブとしてフォーマットすることになる（**図4**）。たとえば、Windowsがプリインストールされたpcで、出荷時に内蔵されているディスク装置が「Cドライブ」と「Dドライブ」の2つのドライブにフォーマットされている場合、そのディスク装置は、1つの基本領域（C）と1つの拡張領域が確保されており、拡張領域の中にさらに1つの論理ドライブ（D）が確保されている。

■Linux用パーティション

Linuxの場合、Windowsでいう「基本領域」に相当する「プライマリパーティション」を、1台のディスクに最大4つまで確保し、それぞれを1台のドライブとみなせるようになっている。Windowsによって、すでに基本領域と拡張領域が確保されてしまっているディスクでも、あと2つまでのプライマリパーティションを追加して、Linux用に使うことができる（もちろん、ディスクに空き容量がある場合に限るが…）。

基本領域とは別に、Windowsで拡張領域が確保されており、かつ拡張領域内に空き領域（論理ドライブが割り当てられていない領域）がある場合は、その中にLinux用の論理ドライブを作成することも可能だ（**図5**）。拡張領域の中にドライブを作成する場合、あらかじめ確保されている拡張領域の大きさを超えるドライブの作成ができないなど、領域確保に制限があることが問題になるが、後述するようなPartitionMagicやシステムコマンダーなどのマルチブート支援ユーティリティを使用すれば、領域を拡大するなどの操作も簡単に行える。

現在のLinuxでは、「ext2」という形式のドライブをOSの起動ファイルや各種データファイルの保存用に使用するほか、スワップ専用のパーティションを使うこともできる。スワップパーティションは、仮装記憶に用いるディスク領域だが、パーティションを分けておくことで、Diskfullなどからくるアクシデントを避けられる。主要なディレクトリごとにパーティションを分けるやり方もあるが、これは各パーティションの容量の見積もりが難しい。したがって、一般的にはext2形式のパーティションを1つと、スワップパーティションを1つの、合計2つのパーティションを用意すればいいだろう。

■Linuxから見たパーティションの名前

Linuxをはじめ、UNIX系のOSでは、ディスク装置は「デバイスファイル」という特殊な形式のファイルとして仮想化されている。ここでは「デバイスファイルとは何か」についての説明は省略するが、1台のディスク装置全体、あるいはそのディスク装置の中に存在する個々のパーティションは、（Linux

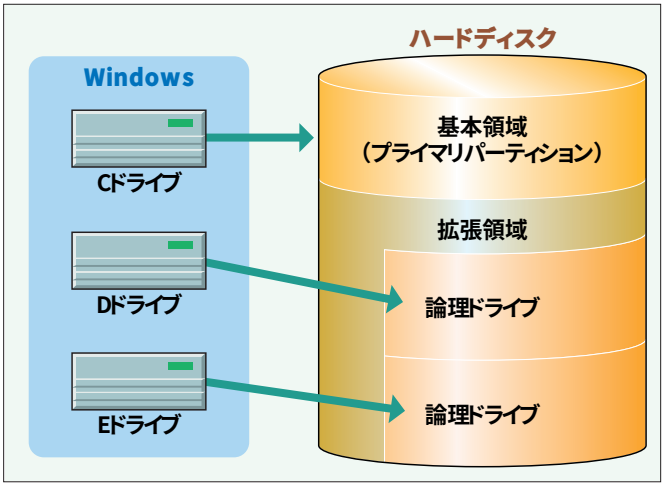


図4　DOS／Windows用のパーティション

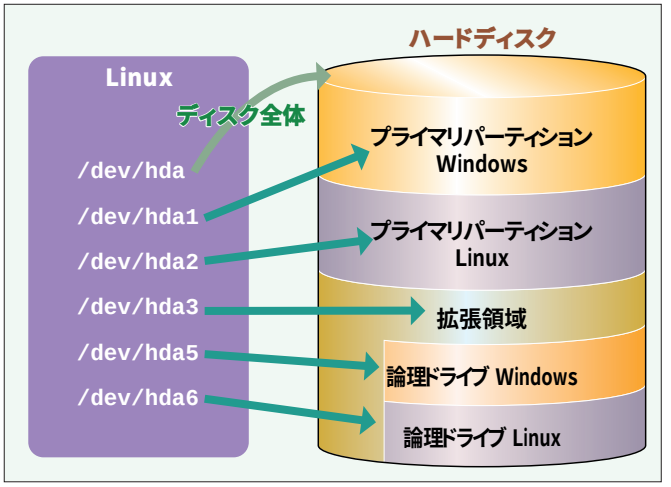


図5　Linux用のパーティション

以外のOSが使用しているものも含めて）すべて個々にデバイスファイルの名前が割り当てられている。

たとえば、ドライブ0（通常、プライマリIDEコントローラのマスタードライブとして接続されているディスク装置）は、「hda」というデバイス名が割り当てられる。Linuxでは、デバイスファイルは/devディレクトリに作成されるので、通常「/dev/hda」というフルパス名で表される。

/dev/hdaはディスク装置全体を表す名前で、ディスク装置の中にパーティションを作成するためにfdiskコマンドを使うような場合に、

```
# fdisk /dev/hda
```

というように使う。

ディスク装置の中にパーティションが作成されている場合、**表2**のような規則にしたがって、hda1、hda2、hda3…、といったようなパーティションごとのデバイスファイル名が割り当てられる。これらのデバイスファイル名は、（Linuxから見る限り）DOS／Windowsでフォーマットされているパーティションに対しても同じように扱われる。したがって、たとえば1台目のハードディスクの1番目のパーティションがFAT形式でフォーマットされている場合、Linuxからそのパーティションにアクセスする場合（たとえば/mnt/dosディレクトリにマウントして、ファイルの読み書きを行えるようにしたい場合）は、次のようなコマンドをroot権限で実行することになる。

```
# mount -t msdos /dev/hda1 /mnt/dos
```

通常、一般ユーザーとしてアプリケーションを使う場合に、このようなデバイスファイルを直接指定することはないだろう。しかし、fdiskなどのパーティション設定ツールや、ファイルシステムの作成（平たくいえばフォーマットを行う）コマンド「ext2fs」、前述のmountコマンド、liloの設定を行う場合などには、このデバイスファイル名が必要になる。

なお、マシンにIDE以外のハードディスク、たとえばSCSIインターフェイスに接続するハードディスクがある場合は、**表3**のようなデバイス名が割り当てられる。

Linuxインストール領域の確保

さて、実際にLinuxをインストールする領域を確保することにしよう。ハードディスクドライブの空き容量に十分な余裕があるものとする。もちろん現在Windowsで利用しているドライブでかまわない。たとえばCドライブの残り容量が2Gバイトあるでもいいし、Dドライブの3.2Gバイトはまるっきり使っていないでもいい。ここからの作業は、ドライブ構成によって異なる手順を踏まえるため、以下のように場合分けてみた。なお、ここではCD-ROMドライブは勘定に入れていない。Dドライブといっても、Cドライブの次のハードディスクドライブという意味なので、CD-ROMドライブのドライブレターがDに固定されている場合などは、Eドライブなどに読み替えて考えてほしい。

1.Cドライブしか存在しない
2.Dドライブ以降がある

1はCドライブのみ存在していて、容量に余裕がある場合だ。ノートPCや最近流行りの省スペース型デスクトップPCなどでは、構造上ディスクドライブの増設ができないので、この手段を選ぶことが多い。この場合はCドライブのパーティションを縮小して、残りをLinuxに割り当てる。2は大容量ハードディスクを搭載したPCで、出荷時にハードディスクがいくつかのパーティションに分割されている場合に有効な手段である。拡張領域の論理ドライブのサイズを変更して、残りをLinuxに割

ドライブ	デバイス名
ドライブ0	hda
プライマリ（拡張）パーティション	hda1
プライマリ（拡張）パーティション	hda2
プライマリ（拡張）パーティション	hda3
プライマリ（拡張）パーティション	hda4
論理ドライブ	hda5
論理ドライブ	hda6
：	
ドライブ1	hdb
ドライブ2	hdc
ドライブ3	hdd

表2　IDE接続ハードディスクドライブの命名規則

ドライブ	デバイス名
ドライブ0	sda
プライマリ（拡張）パーティション	sda1
プライマリ（拡張）パーティション	sda2
プライマリ（拡張）パーティション	sda3
プライマリ（拡張）パーティション	sda4
論理ドライブ	sda5
論理ドライブ	sda6
：	
ドライブ1	sdb
ドライブ2	sdc
：	
ドライブ7	sdh

表3　SCSI接続ハードディスクドライブの命名規則

り当てる。

ところで、実際にLinuxのパーティションを切り出す前に注意が必要である。実は、Linuxの起動パーティションはディスクの先頭から8Gバイト以内の領域から始まっていなければならない。これはBIOSがディスクの1024シリンダ（つまり8Gバイト）を越えたパーティションからブートできないことに起因する問題だ。20Gバイトを越えるハードディスクドライブ当たり前になりつつある現在では、実際にパーティションを分割するにあたって、この点に注意してほしい。

ここであげた以外にも、ハードディスクドライブそのものを増設するという解決策もあるが、それについては130ページの応用編を参照してほしい。

Cドライブを縮小する

Cドライブしかなく、空き容量がふんだんにある場合は、Cドライブのパーティションを縮小して、余った領域をLinuxインストール用のパーティションに割り当てることができる。これには、FIPS.EXEというツールを利用する。FIPS.EXEは、多くのLinuxディストリビューションCDに付属するツールで、データを保持したまま、基本領域を縮小できるという優れモノだ（**図6**）。ただし、拡張領域の縮小はできない。

■FIPS.EXEの実行

まず、FIPS.EXEを用意する。TurboLinuxでは「インストールCD」のディレクトリ\dosutilsの下に「Fips20.zip」というアーカイブファイルがあるので、これをWinZipなどのツールで解凍しておく（ちなみにすでに同じディレクトリにFIPS.EXEがあるが、これはバージョン1.5でVFATに対応していないので使ってはいけない）。FIPS.EXEはマルチタスク環境下で実行してはいけないので、フロッピーディスクからDOS（モード）を起動して実行する必要がある。したがって、次にフロッピーディスクを1枚用意し、［起動専用］でフォーマットする。そこに先ほど解凍したファイル群の中から「FIPS.EXE」と「ERRORS.TXT」をコピーする。

念のために、FIPSの作業に入る前にCドライブにWindowsの［デフラグ］をかけておこう。

さて、いよいよ、作成したフロッピーディスクから起動する。DOSプロンプトが表示されたら「FIPS」と入力しEnterキーを押す。　まず、FIPS.EXEについてのメッセージが表示される。そこで、何かキーを押すと、現在のパーティションテーブルが表示される。

分割したいパーティション番号「1」を入力すると、ブートセクタの情報が表示され、これをフロッピーディスクにバックアップするかどうかを問い合わせてくる。バックアップが終了すると、パーティションの大きさを問い合わせてくる。

ここで、「Old partiton」というのはCドライブとして残すパーティションの大きさである。「New partition」というのが、Cドライブを縮小した結果、新しく作成されるパーティションのことである。矢印キーを使って値を変更する。ここでは、「Old partition」を4200.0MBにしている。残り1721MBがLinuxに割り当てられる。

Enterキーを押すと値が確定し、指定した値でパーティションを縮小した結果を一覧してくれる。再編集したければ「r」、決定なら「c」を入力する。

最終確認に「y」を押すと実際に情報が書き込まれ、FIPS.EXEが終了する。

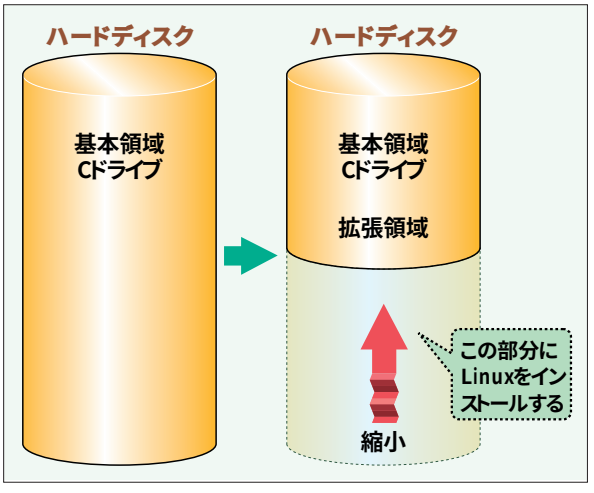


図6　Cドライブを縮小する

```
Partition table:

     |        |     Start      |      |      End       | Start  |Number of|
Part.|bootable|Head Cyl. Sector|System|Head Cyl. Sector| Sector |Sectors  |  MB
-----+--------+----------------+------+----------------+--------+---------+----
1    |    yes |   1    0      1|   06h| 239  837     63|      63| 12670559|6186
2    |     no |   0    0      0|   00h|   0    0      0|       0|        0|   0
3    |     no |   0    0      0|   00h|   0    0      0|       0|        0|   0
4    |     no |   0    0      1|   00h|   0    0      0|       0|        0|   0

↓パーティション縮小後

New partition table:

     |        |     Start      |      |      End       | Start  |Number of|
Part.|bootable|Head Cyl. Sector|System|Head Cyl. Sector| Sector |Sectors  |  MB
-----+--------+----------------+------+----------------+--------+---------+----
1    |    yes |   0   37      1|   06h| 239  535     63|      63|  8602708|4200
2    |     no |   0  536      1|   06h| 239  837     63| 8602709| 12670559|1721
3    |     no |   0    0      0|   00h|   0    0      0|       0|        0|   0
4    |     no |   0    0      0|   00h|   0    0      0|       0|        0|   0
```

画面1 FIPS.EXEの実行画面

Dドライブ以降を使う

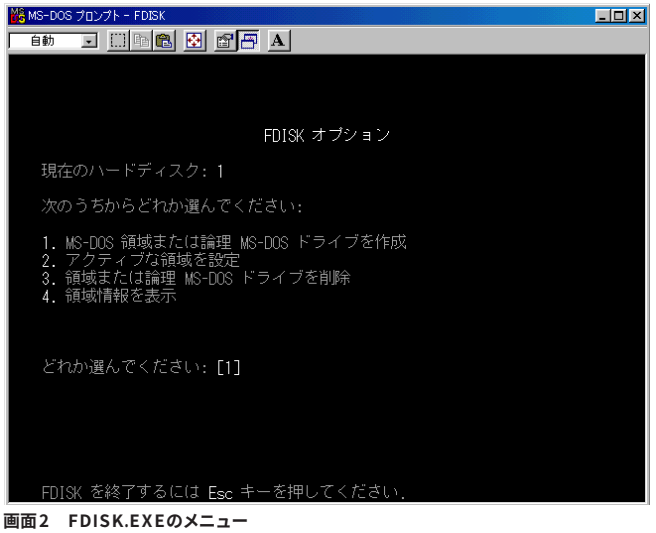
Dドライブ以降を使うというのは、DOS／Windowsで言う拡張領域を使うという意味だ。ここでは、拡張領域内の論理ドライブを縮小して、残りをLinuxインストール用の論理ドライブに割り当てる方法について説明するが、もしDドライブは丸ごと不要という場合でも、論理ドライブの削除だけはしておこう（**図7**）。

論理ドライブのサイズを変更するには、WindowsのFDISK.EXEを使い、いったん論理ドライブを削除して、あらたにサイズを縮小した論理ドライブを作成しなければならない。当然だが、このときDドライブ以降のディスクの内容はすべて失われる。事前に内容をバックアップしておき、論理ドライブを作成し直した後、リストアしなければならない。

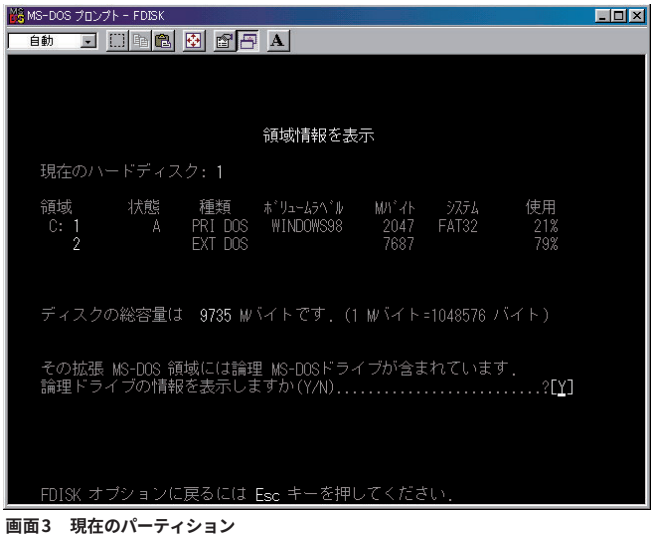
こうした作業がわずらわしいと感じる場合は、マルチブート環境支援ユーティリティ「PartitionMagic」や「システムコマンダー」を利用するといい。こうしたユーティリティでは、ディスク内容を保持したまま、パーティションのリサイズが行えるので、作業行程は少なくて済む。

まずは、Dドライブ以降のデータをバックアップしてほしい。それからWindowsに付属のFDISK.EXEを起動する。大容量ディスクをサポートするかどうかの問い合わせには「Y」と答える。するとメニュー画面が表示される（**画面2**）。「領域情報を表示」を選択すると、現在のパーティション構成が表示される。種類に「EXT DOS」と表示されているのが拡張領域である（**画面3**）。ここでさらに「Y」を入力すると、拡張領域内の論理ドライブの状態が表示される。

Escキーでメニューに戻り、論理ドライブを削除する。メニュー画面から



画面2　FDISK.EXEのメニュー



画面3　現在のパーティション

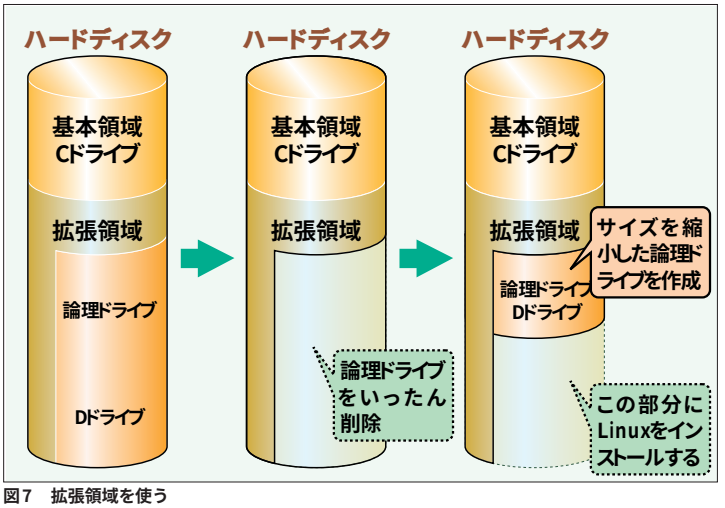
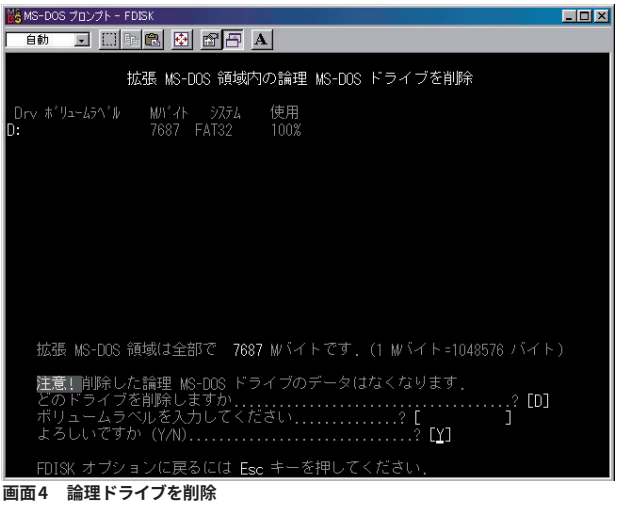


図7　拡張領域を使う



画面4　論理ドライブを削除

「領域または論理MS-DOSドライブを削除」を選択する。さらに「拡張MS-DOS領域の論理MS-DOSドライブを削除」を選択して不要な論理ドライブを削除する（**画面4**）。

次にメニューから「MS-DOS領域または論理MS-DOSドライブを作成」を選択し、「拡張MS-DOS領域内の論理MS-DOSドライブを作成」を選択する。ここでWindowsで使用する容量だけを指定する。ドライブが作成されると、再び論理ドライブ作成画面に戻る（**画面5**）。Linux用の空き容量を残したまま、ESCキーでFDISK.EXEを終了する。論理ドライブに割り当てなかった残り容量が、Linuxのパーティションとして使えるようになった。

Linuxのインストール

インストールのための領域が確保できたら、Linuxをインストールする。ここでは、Linuxパーティションの作成とliloの書き込みをメインに紹介する。ディストリビューションによってインストール手順が多少異なったとしても、この2点については共通なので参考にしてほしい。

コマンド	意味
a	ブートドライブの設定（DOSで言う「アクティブ」にする）
b	BSD系UNIXのディスクラベルを編集
c	MS-DOSの互換フラグを設定
d	パーティションの削除
l	パーティションのタイプIDを一覧表示
m	メニューを表示
n	新しいパーティションの作成
p	パーティションテーブルを表示
q	結果を書き込まずに終了
t	パーティションタイプIDを変更
u	セクタ／シリンダ表示の切り替え
v	パーティションテーブルのベリファイ
w	結果を書き込んで終了
x	拡張設定メニューに移行

表4 fdiskコマンド一覧

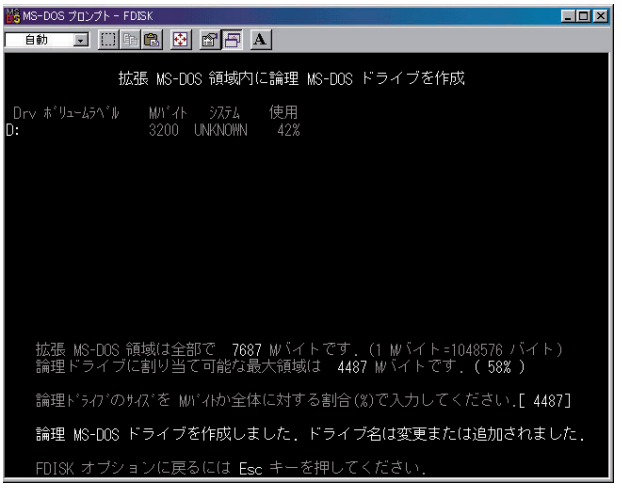
■パーティション作成とマウントポイントの指定

LinuxのインストールCDでシステムを起動し、インストールを開始する。インストールの手順の詳細についてはここでは触れないが、TurboLinuxの場合、インストーラの「ディスクの分割方法」の画面では「手動」を選択し、次の「パーティション設定」では「FDISK（英語）」を選択する。「FDISK（日本語）」でも機能や使い方に違いはないが、マシンの構成によっては画面が乱れるなどの問題が起こる場合もあるため、英語版fdiskの使用を勧める。操作がわかりやすいように、各コマンドの意味を**表4**にまとめておくので参照してほしい。

fdiskを起動したら、まず「p」コマンドを入力して、現在のディスクのパーティション構成をチェックする（**画面6**）。Cドライブを縮小した場合は、残りの部分がWindowsのタイプIDになったパーティションとして見えているはずだ。まずはこれを拡張領域（05）に変更しよう。「t」コマンドで/dev/hda2のタイプIDに「5」を指定する。

新しいパーティション、または論理ドライブを作成するには「n」コマンド（新規パーティションの追加）を入力する。すると、追加するパーティションの種類を問い合わせてくるので、プライマリパーティションを作成する場合は1から4の範囲の数字を、拡張領域内に論理ドライブを作成する場合は、5以上の数字を入力する。

前述したように、Linuxでは複数のプライマリパーティションを利用できるが、ここでは、Linuxをすべて拡張領域内の論理ドライブにインストールする。この方法ならば、Linuxのファイルシステムとスワップ領域を別々に



画面5 論理ドライブが作成された

```
Disk /dev/hda: 240 heads, 63 sectors, 839 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       569   4301608+   c  Win95 FAT32 (LBA)
/dev/hda2           570       838   2033640    c  Win95 FAT32 (LBA)Disk
/dev/hda: 240 heads, 63 sectors, 839 cylinders
Units = cylinders of 15120 * 512 bytes
```

画面6 Linux用パーティション作成前

作成しても、パーティションを1つしか消費しなくて済む。

Linuxシステムを論理ドライブ/dev/hda5に、Linuxのスワップパーティションを論理ドライブ/dev/hda6として作成する。Windowsの論理ドライブが存在する場合は、すでに/dev/hda5（あるいはそれ以上）のドライブ番号は使われているので、それに応じて番号を変える必要がある。

まず、さきほどのnコマンドのパーティション番号入力時に「6」と入力し、次にスワップパーティションのサイズを（メモリ容量に応じて）入力する。このパーティションはスワップ用なので、次に「t」コマンドを実行して、Linuxスワップのタイプとして「82」を指定する。同様に、論理ドライブ/dev/hda5のLinux ext2パーティション（タイプ番号「83」）を作成し、「p」コマンドで作成したパーティションを確認する（**画面7**）。正しく作成されていれば、「w」コマンドでパーティションを更新してインストーラに戻る。以上の作業でパーティションは**図8**のようになった。

インストーラでは、次に、作成したパーティションをどこにマウントするかを指定する画面になるが、ここではLinux用パーティションを1つしか作成していないので、デフォルト（「/」にマウント）の設定でよい。

■インストーラによるブートローダの設定

インストール作業を進め、パッケージの選択やファイルのコピーが行われた後、「LILO設定」の画面になる。「どこにLILOをインストールしますか？」という質問項目があるが、ここでは「ブートパーティションの最初のセクタ」を選択する。「マスターブートレコード」を選択してしまうと、ディスクにすでにインストールされている各種のブートローダ（Windows標準、あるいはPartitionMagicやシステムコマンダーなどの専用ブートローダ）が上書きされてしまう。liloのインストールや設定については、126ページから詳しく解説するので、この時点ではliloをインストールする場所だけ注意すること。

「起動可能パーティション」の設定画面で、Windowsのシステムドライブ（**画面7**では/dev/hda1）を選択して「編集」をクリックし、「起動名」として適当な文字列（たとえば「win98」）を入力すれば、liloからWindows 98も起動できるようになる。

■ブートするパーティションの設定

以上の手順でLinuxをインストールしただけでは、マシンを再起動してもWindowsしか起動できない。新たにインストールしたLinuxを起動するには、Linuxのパーティションを「アクティブ」にする必要がある。パーティションをアクティブにするには、WindowsのFDISK.EXEか、Linuxのfdiskを使う。たとえばWindowsのFDISK.EXEの場合、スタートメニューの「ファイル名を指定して実行」で「FDISK.EXE」を実行すると、メニュー画面が表示される。「2」の「アクティブな領域を設定」を選択し、次の画面でLinuxのブートパーティション（この場合3番目のパーティション）を選択すればよい。liloのインストールが正しく行われていれば、マシンをリブートすると「LILO boot:」というプロンプトが表示され、ここで先ほど指定した起動名（ここでは「win98」）と入力すればWindows 98が、「linux」と入力するか、何も入力しない状態でリターンキーを押せば、Linuxが起動す

```
Disk /dev/hda: 240 heads, 63 sectors, 839 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       569   4301608+   c  Win95 FAT32 (LBA)
/dev/hda2           570       838   2033640    5  Extended
/dev/hda5           570       830   1973192+  83  Linux
/dev/hda6           831       838     60448+  82  Linux swap
```

画面7　Linux用パーティション作成後

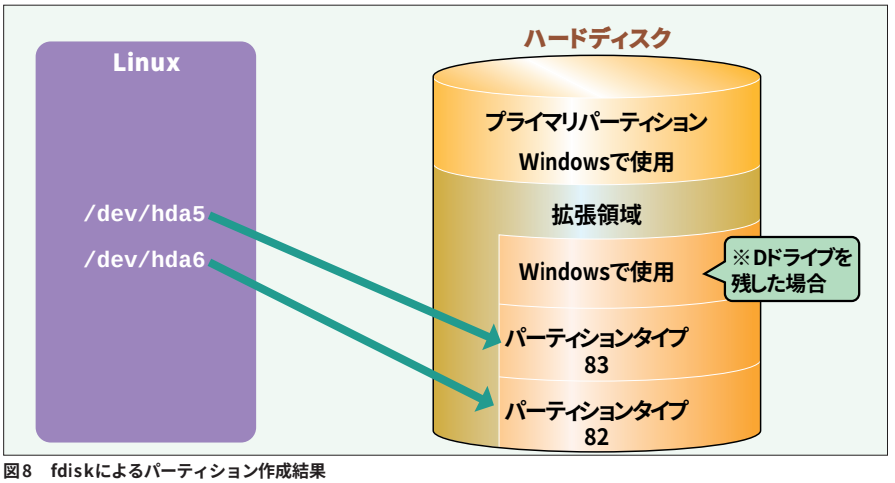


図8　fdiskによるパーティション作成結果

ることになる。

なお、システムコマンダーやBootMagicなどのマルチブート環境支援ツールをインストールしてある場合は、ツールのマニュアルに従って、Linuxの起動パーティションを登録することで、アクティブパーティションの設定を行わなくても、LinuxとWindowsの両者を切り替えてブートできるようになる。詳細については各製品のマニュアルを参照してもらいたい。

liloによるマルチブートの設定

Linuxには「lilo」と呼ばれるブートローダが付属する。liloとは「Linux Loader」の略で、その名の通りLinuxをロード（起動）するためのプログラムであるが、Linux以外のOSを起動することもできる。市販のマルチブート支援ユーティリティに付属するブートローダ（ブートマネージャ）とほぼ同じ働きをするプログラムだが、テキスト形式の設定ファイルを編集する必要があり、起動時にもOSを選択するメニューが表示されないなど、取っつきにくい印象がある。しかし、マルチブート環境を構築するに当たって必要となる機能はひととおり備えている。簡単な環境の構築には十分である。

ここでは、liloのインストールやマルチブート環境での設定方法について解説しよう。

■最初のインストールはインストーラ任せ

現在のほとんどのLinuxディストリビューションでは、OSをインストールすると自動的にliloもインストールされるようになっている。インストーラの手順のどこかに、「ブートローダのインストール」あるいは「LILOの設定」といった画面が現れるはずである。

TurboLinux 4.xの場合、「LILOの設定」の画面の中で、liloをインストールする場所と、liloからブートできるOSの追加を行えるようになっている。しかし、デフォルトで起動されるOSはLinuxに限定され、また、起動までの待ち時間（デフォルトでは5秒）の変更などはできない。これらの設定を変更したい場合は、Linuxのインストール後に、手動でliloの再インストールを行う必要がある。その方法については後ほど述べるが、まずliloをインストールする場所による、動作の違いについて説明する。

■ハードディスクにインストール

liloはLinuxのロード、正確にはLinuxのカーネルファイルをメモリに読み込んで、マシンの制御を読み込んだカーネルに渡すまでの処理を行うプログラムだ。ハードディスクからのブート処理の流れについては116ページで解説されているが、liloの正体は512バイトのセクタに収まるサイズの、ファーストステージローダと、/boot/boot.bのセカンドステージローダ、そして/boot/map（ロードするイメージのジオメトリが記述される）で構成される。Linuxの「liloコマンド」（/sbin/lilo）は、liloのブートローダプログラムを指定された場所に書き込むためのコマンドである。liloのブートローダは、次のいずれかの場所に書き込み可能だ。

1.ハードディスクのマスターブートレコード（MBR）
2.各パーティションの先頭セクタ
3.フロッピーディスクの先頭セクタ

1のMBRにインストールすると、アクティブパーティションの設定に関わらず、MBR上のliloが読み込まれ、そこからいずれかのパーティションのOSを呼び出すことになる。2の場合は、liloがインストールされたパーティションがアクティブになっている場合にはliloが呼び出されるが、それ以外のパーティションがアクティブになっている場合は、liloが呼び出されることはない。3のフロッピーにインストールすると、ハードディスクのアクティブパーティションの設定やMBRの内容に関わらず、liloを起動することができる。この場合、フロッピーディスク自体にOSのブートコードがインストールされている必要はなく、ブートセクタ以外は空のフロッピーでもかまわない。

マシンにLinuxのみインストールする場合は、MBRへのインストールを行うのが一般的だが、マルチブート環境にする場合は、Linux以外のOSが起動できなくなってしまう可能性がある。このため、最初のうちはliloはMBRへはインストールせず、Linuxをインストールしたパーティションのブートセクタにインストールすべきだ。その上で、fdiskコマンドなどを使ってアクティブパーティションをLinuxのパーティションに設定する。後述する手順でliloの設定ファイルを編集し、liloのブートローダを再インストールすれば、任意のパーティション上のOSを起動できるようになる。

なお、各種のマルチブート支援ユーティリティに付属するブートマネージャをインストールしている場合、これらのプログラムはMBRを使用している。このため、Linuxパーティションのブートセクタにliloをインストールして、そのパーティションをアクティブに設定した場合でも、まずMBRにインストールされているブートマネージャが起動し、そこから改めてLinuxパー

ティションのliloを呼び出す形になる。

3のフロッピーディスクへのインストールを行うと、ブートセクタやMBRが破損した場合や、liloの設定をミスした場合に備える緊急用起動ディスクとして使うことができる。また、liloの設定に慣れていない間は、設定の確認用や練習用に起動ディスクを作ってみるのもよいだろう。

liloの設定変更と再インストール

liloのインストールは/sbin/liloコマンドで行うが、設定は/etc/lilo.confファイルに各種のパラメータの値を記述することで行う。lilo.confでは数多くのパラメータを設定することができるが、ここでは基本的なものについてのみ説明する。詳細については、lilo.confおよびliloのオンラインマニュアル（man lilo.conf、man lilo）を参照すること。また、/usr/doc/lilo-0.21ディレクトリに置かれている各種ドキュメントにも目を通すことを勧める。

■lilo.confの編集

デフォルトのlilo.confの内容は、Linuxをインストールしたときの条件によって変わってくるが、おおむね**リスト1**のような内容になっている。

この中で特に注意すべきなのは、boot、timeout、default、other、label、table、image、rootの各パラメータである。各パラメータの意味を順に説明しよう。

・boot=/dev/hda3

liloのブートローダをインストールする場所を指定するパラメータ。/dev/hda1、/dev/hda2、/dev/hda3…は、それぞれ1台目のIDEドライブのプライマリパーティション1番、2番、3番のブートセクタにインストールする場合に指定する。1台目のEIDEドライブのMBRへインストールする場合は、boot=/dev/hda（パーティション番号を指定しない）と指定する。3.5インチフロッピーディスク（1台目、Aドライブ）のブートセクタにインストールする場合は、boot=/dev/fd0とする。

・timeout=50

リブート後にliloのブートローダが「LILO boot:」というプロンプトを表示してから、デフォルトのOSのブートを開始するまでの待ち時間の指定。10分の1秒単位で指定する。デフォルトだと5秒の設定なので、これでは短すぎると感じる場合には、timeout=300（30秒）などと適切な値を設定すればよい。timeoutの値を指定しない（timeout=??という行を削除）すると、無限大（LILO boot:プロンプトでリターンキーを押すかブートするOSのラベルを入力しないかぎり、ブートは行わない）に設定される。

・default=linux

LILO:プロンプトでなにも入力せずにリターンキーを押した場合や、timeout=で設定した時間が経過した後で、自動的に起動されるOSのラベル（後述）を記述する。この例では、「default=win98」と変更すれば、デフォルトで起動されるOSをLinuxからWindows 98へ変更することができる。

・other=/dev/hda1

Linux以外のOSをliloから起動したい場合の指定。other=の後には、起動したいOSがインストールされているパーティションのデバイス名を指定する。この例では、/dev/hda1（1番目のIDEドライブの1番目のプライマリパーティション）を指定している。

なお、linuxの起動イメージについてはこの「other」ではなく、後で説明する「image」で指定する。あくまでもWindowsやOS/2といったLinux以外のOSの起動指定のパラメータであることに注意すること。

・label=win98

直前の「other=…」で指定されたOSをブートする際に、liloのプロンプトで入力する識別文字列（ラベル）。この例でいえば、起動時の「LILO boot:」というプロンプトが表示された際に、

```
LILO boot: win98
```

と入力すると、/dev/hda1のWindows 98が起動されるようになる。

・table=/dev/hda

ドライブのパーティションテーブルが記録されている場所を指定する。通常パーティションテーブルはMBRと同じセクタに記録されているので、上の設定のままでかまわない。この記述を省略すると、パーティションテーブルの内容がOSのブートコードに伝えられなくなるので、OSの種類によっては正しく起動できなくなることがある。

・image=/boot/vmlinuz

起動するLinuxのカーネルコードのファイルの場所を指定する。直後のlabel=行で任意のラベルをつけることができるので、複数のバージョンのカーネルを（同じパーティションから）適宜切り替えて起動したい場合には、ブートしたいカーネルファイルの場所を指定した複数の「image=」、「label=」の組み合わせを記述しておけばよい。

・root=/dev/hda5

Linuxをブートする際に、ルートファイルシステムとしてマウントするパーティションを指定する。ブート用のコード（liloとカーネルコードのみ）を納めたパーティションと、それ以外のファイルを納めたルートファイルシステムが異なる場合は、このパラメータを適切に設定する必要がある。

これらを踏まえて、カスタマイズしたlilo.confの例を**リスト2**に示す。

例では、linuxは/dev/hda5の論理ドライブからブートするようにしている。また、プロンプトに何も入力しないでEnterキーを押した場合は、/dev/hda1のWindows 98がブートする。timeout=を削除し、入力があるまでプロンプトが待ちつづけるようにしてある。また、linuxをブートさせるための起動名は「lin」にした。

また、**リスト3**は、緊急時のためにliloをフロッピーディスクにインストールするためのものだ。「boot=」行が/dev/fd0となっていることに注目してほしい。さらに、/boot/myの下にカスタマイズしたカーネルを置いてあり、それらを起動し分けるようになっている。

■liloコマンドの実行

/etc/lilo.confの記述を終えたら、liloコマンドを実行してブートローダを実際にディスクに書き込む必要がある。フロッピーディスクに書き込む場合は、Windows上、あるいはLinuxのsuperformatコマンドによってフォーマットした、空のフロッピーディスクをドライブに挿入しておく。

あとは、rootユーザーの状態で、コマンドラインから、

```
# lilo
```

と入力すれば、/etc/lilo.confの内容に従って適切に設定されたブートローダプログラムが、指定されたデバイスに書き込まれる。繰り返しになるが、lilo.confの記述に誤りがあった場合などはシステムを起動できなくなるおそれもあるので、最初は、「boot=/dev/fd0」の設定を使ってフロッピーディスクにliloをインストールしてそのフロッピーからの起動を試み、きちんと動作することを確認してからハードディスクにインストールすることを勧める。

以上で基本的なマルチブート環境が構築できた。マルチと言っても2つのOSだけだが、基本的な要素は押えられたものと思う。liloが正しく動作しない場合のトラブルシューティングなどについては、130ページを参照してほしい。

▼リスト1 標準的なlilo.conf

```
boot=/dev/hda3
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux
other=/dev/hda1
        label=win98
        table=/dev/hda
image=/boot/vmlinuz
        label=linux
        root=/dev/hda5
        read-only
```

▼リスト2 lilo.confのカスタマイズ例

```
boot=/dev/hda     ←MBRにインストールする
map=/boot/map
install=/boot/boot.b
prompt    timeout=を削除したので入力があるまでプロンプトが出続ける
default=win98   ←デフォルトでWindows98がブートする
other=/dev/hda1
        label=win98
        table=/dev/hda
image=/boot/vmlinuz
        label=lin         ←Linuxの起動名は「lin」
        root=/dev/hda5    ←/dev/hda5を/にマウント
        read-only
```

▼リスト3　フロッピーにインストールするためのlilo.conf

```
boot=/dev/fd0   ←フロッピーディスクにインストールする
map=/boot/map
install=/boot/boot.b
prompt
timeout=60
default=linux   ←デフォルトでは通常のlinuxがブートする
other=/dev/hda1
        label=win98
        table=/dev/hda
image=/boot/vmlinuz
        label=linux       ←通常のカーネルが起動
image=/boot/my/vmlinuz
        label=my          ←カスタマイズしたカーネルが起動
        root=/dev/hda5
        read-only
```

マルチブート環境支援ユーティリティ

市販ユーティリティを使うメリット

Windowsやディストリビューションパッケージに付属のユーティリティだけでも、マルチブート環境はなんとか構築できる。しかし、複数のツールを使い分けたり、ユーティリティのインターフェイスが文字ベースであったり、ドキュメントが英語だったり、バックアップ等、事前に面倒な作業を強いられたりと、初心者にはなかなか取っつきにくい面が多い。また、WindowsとLinuxを各1バージョンずつインストールする場合は、それでもなんとか構築できるが、WindowsNTやFree BSD、BeOSなど、第三、第四のOSを環境に加えようとすると、付属ユーティリティだけではなかなか難しい。

そこで登場するのが、市販のマルチブート環境支援ユーティリティだ。たとえば、122ページで解説したように、通常ならば拡張領域の空き容量から新たなパーティションを分割する場合、論理ドライブの内容をいったん削除しなければならない。しかし、ここで紹介する製品を使えば、パーティションの容量変更や、パーティションの移動はすべて内容を残したままできるのだ。また、ブートマネージャもliloと違い、GUIを使った視覚的にもわかりやすいものになっており、扱えるOSの数も多く、機能的にもより高度なブートセレクトが可能になっている。

ここで紹介する2つの製品は、どちらも機能的には大変充実しており、甲乙つけがたい製品となっている。少なくとも、本特集で扱うレベルのマルチブート環境の構築においては、どちらの製品も十分な威力を発揮してくれるだろう。それでは、選択の決め手となるのはどこだろうか？

コマンダープレミアムパック

（株式会社ソフトボート）
1万4800円
http://www.softboat.co.jp/

「コマンダープレミアムパック」は、パーティショニング&PC快適化ツール「パーティションコマンダー」とマルチOSブートアップマネージャ「システムコマンダー4」をパックにした製品である。

システムコマンダーは、今や定番ともいえるマルチOSブートマネージャだ。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も豊富で、Windows 9x／NTをはじめ、各種DOS、Linux、＊BSD、NetWare、BeOS、Bright/Vなど、IntelのCPUを対象とした100種類以上のOSに対応したパーティションを作成可能だ。OSウィザードと呼ばれる機能を搭載していて、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。ブートセレクタ機能についても、これらのOSを最大32種類まで管理できる。

いっぽう、パーティションコマンダーは、より高機能なパーティション操作を可能にしたツールで、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回収したり、ディスクアクセスの高速化や、パーティションの複製を実現している。システムコマンダーのパーティション操作機能を補うものだ。Windows環境にインストールした場合、いちおうGUI環境で起動するが、マウスオペレーションなどはライバルの「PartitionMagic」ほど洗練されているとは言い難い。また、些細な問題ではあるが、「パーティションコマンダー」を一度起動すると、たとえ何も処理を実行しなかったとしても、終了時にWindowsの再起動を余儀なくされる。

2つのツールはそれぞれ単品でも販売されているが、「コマンダープレミアムパック」では、この2本のツールをうまく組み合わせて、ハードディスク資源を無駄なく使えるようする、お得なパッケージとなっている。

マニュアル類は充実しているが、特に「見てわかるCommander導入ガイドブック」は、よくできている。この

マニュアルのできが、市販のアプリケーション解説書に迫るていねいさで、1ステップずつ画面を掲載して解説する念の入れようなのだ。初心者が、一人でマルチブート環境を構築する場合には、頼もしい味方になってくれそうだ。

ユーザーズマニュアルも各1冊付属するが、「システムコマンダー」のマニュアルにより多くのページ数が割かれている。

なお、「システムコマンダー」と「パーティションコマンダー」自体をインストールしたり起動したりするのに必要な環境は、DOSおよびWindows 9xとなっており、Windows NTへのインストールには対応していないので注意が必要だ。提供メディアは各ツールともフロッピーディスク×2枚の計4枚。

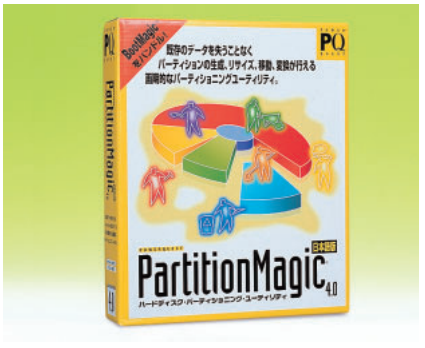
PartitionMagic 4.0

（株式会社ネットジャパン）
1万5800円
http://wwww.netjapan.co.jp/

「PartitionMagic 4.0」は高機能なパーティショニングツールで、基本的なパーティション操作はもちろん、FATタイプの変更、クラスタサイズの調節、FATタイプの変更といった浪費領域の回収、パーティションの複製など、複雑なパーティション操作を簡単な操作で行える。各機能はウィザード形式を取り入れ、簡単な質問に答えていくだけで、自動的に最適な結果を提供してくれる。OSのインストールも、対象となるOS名を答えるだけで、最適なパーティションを設定してくれる。各OSに専用のバージョンが用意されていることもあって、Windows版の操作感も馴染める。とくにパーティションのサイズ変更と移動は、棒グラフで示されたパーティションイメージをマウスでドラッグするといったアバウトな操作も可能になっている。

また、不良セクタの再診断機能というのが面白い。通常のフォーマッタで不良クラスタと診断され、はねられた領域には、実は使えるセクタも混じっている。それを再度診断し直して、使える領域は元に戻してくれる機能だ。クラッシュを繰り返し、不良クラスタが多数存在するディスクは、この機能で若干使用可能な容量が増えるかもしれないし、ひょっとするとアクセスも多少高速になるかもしれない。

操作対象となるOSも、Windows



9x／NTをはじめ、OS/2 3.x、各種DOS、Linux、BeOS等、メジャーなものはほぼサポートされているので、困ることはないだろう。

本パッケージには、ブートマネージャとして「BootMagic」が付属してくる。「コマンダープレミアムパック」同様、本パッケージのみで、パーティション操作とマルチブート環境の構築ができるように配慮されている。

マニュアルの記述はまとまっていてわかりやすい。「コマンダープレミアムパック」のものは、どちらかといえば手順を追った記述になっているが、「PartitionMagic」のマニュアルは、全体像を理解することに力が注がれている。ツールごとに各1冊ユーザーマニュアルが付属する。当然だが「PartitionMagic」のマニュアルにより多くのページが割かれている。パワーユーザーで、パーティション操作を頻繁に行うユーザーにとっては、記述のまとめ方がよく、役立つ。

「PartitionMagic」と「BootMagic」自体をインストールしたり起動したりするのに必要な環境は、DOS、Windows 9x／NTとなっており、Windows NTユーザーは必然的にこちらの「PartitionMagic 4.0」を選択することになる。提供メディアはCD-ROM×1枚。

なお、今回は詳細な評価が間に合わなかったが、同製品は11月26日に「PartitionMagic 5.0」へとバージョンアップしている。実際に読者諸氏が入手するのはこの新バージョンになるはずだ。

バージョンアップした同製品の新機能として興味深いのは次の2点だ。

1.データを保持したまま隣接するパーティションの結合、2.NTFSからFAT／FAT32への変換である。

当然だが、隣接するパーティションの結合も、パーティション内のデータを保持したまま行える。この機能は、マルチブート環境作成時においても、ディスク装置当たり4パーティションの制限にひっかかる場合などに威力を発揮するだろう。

NTFSからFAT／FAT32への変換はWindows NTユーザーには嬉しい機能である。

そのほかにも、各種ユーザーインターフェイスの向上など、強力なものに仕上がっている。

マルチブート環境の構築 －応用編－

ここまでは、基本的なマルチブート環境を構築した。ここからは、トラブルシューティングや、応用的なインストールについて扱う。

liloでうまくブートできない

liloは、OSのブートシーケンスの中でも最もデリケートな部分を処理するプログラムである。このため、/etc/lilo.confの設定を間違えたままハードディスクのブートセクタにliloを書き込んでしまうと、最悪の場合、そのディスク内のOSをいっさい起動できなくなる可能性もある。しかし、liloはパーティションの中身そのものを破壊することはないので、事前の準備を怠りさえしなければ、ちゃんとブートできる状態に復旧できる。

■Linuxを起動できなくなった場合

/etc/lilo.confのimageパラメータやrootパラメータを間違えると、ブート時にカーネルイメージやルートファイルシステムを見つけることができなくなり、Linuxが起動できなくなってしまう。このような場合はliloのブートローダを再インストールするしかないのだが、liloの再インストールにはLinuxを起動する必要がある。

このような場合は、あらかじめ用意しておいた緊急用のLinuxブートフロッピーからLinuxをブートすればよい。Linuxがブートしたら、必要に応じてルートパーティションをマウントし、/etc/lilo.confの編集をやり直して、liloコマンドを実行する。このような場合に備えて、インストール時にブートフロッピーを用意しておくことが肝心だ。また、前述の方法で、きちんとブートできるように設定されたliloのフロッピーを用意しておいてもよい。緊急起動用フロッピーディスクを用意していなかった場合は、OSインストール用の起動ディスクを使ってブートする方法もあるが、手順が複雑になるのでここでは紹介しない。

■Windowsを起動できなくなった場合

Windowsが起動できなくなるのは、MBRにliloをインストールしてしまった場合がほとんどだ。このような場合、まず、Linuxを起動してから、Linuxのパーティションのブートセクタ（あるいは緊急ブート用のフロッピー）にliloをインストールし直す（これをやっておかないと、今度はLinuxがブートできなくなってしまう）。lilo以外のブートマネージャを使う予定がないのであれば、この時点で正しいliloをMBRにインストールし直してもよいが、lilo以外のブートマネージャを使いたい場合には、MBR以外の場所にliloをインストールしなければならない。

liloの再インストールが終わったら、修正したlilo（あるいはWindowsの起動ディスク）からWindowsを起動し、Windowsのコマンドプロンプトから次のコマンドを実行する。

```
fdisk /MBR
```

これで、マスターブートレコードは標準の状態に戻るため、次回にリブートを行うと、その時点でアクティブに設定されているパーティションからブートできるようになる。

lilo以外のブートマネージャを使用したい場合は、fdisk /MBRではなく、各ブートマネージャのマニュアルにしたがって、ブートマネージャの再インストールを行う。

繰り返しになるが、この作業の順序を間違えて、liloの再インストールを行う前にMBRの書き換えを行ってしまうと、今度はLinuxの起動ができなくなるので注意すること。Linuxが起動できなくなったら前項の手順でliloの再インストールを行う。

Linux環境にWindowsを追加する

Linux環境に後からWindows 9xを追加する場合には、以下の点に注意する。

Windowsは、インストール時にMBRを書き換える。つまりliloなどのブートマネージャが上書き消去され、Linuxがブートしなくなってしまうのだ。

そこで、ブートマネージャをMBRに再インストールする必要がある。アクティブパーティションを切り替えるか、liloをインストールした起動フロッピーディスクで、Linuxを起動し、本特集のliloの再インストールの項を参照してMBRにインストールすればいい。

ハードディスクドライブの増設

ハードディスクドライブに空き容量がない場合は、増設するしかない。Linuxはディスク1以降へのインストールもそこからのブートも可能だ。

たとえば、ディスク1の1番目のパーティション（/dev/hdb1）にインストールした場合は、lilo.confのroot=行に以下のように記述する。

```
root=/dev/hdb1
```

ただし、liloをインストールする場所は、あくまでもディスク0なので、boot=行は、

```
boot=/dev/hda
```

または、

```
boot=/dev/hda?
```

となる（?は起動パーティション）。

先頭から8Gバイトを超える領域へのインストール

Linuxはディスクの先頭から8Gバイトを超える領域にブートに必要なファイル（／boot下ディレクトリ）パーティションがあるとブートできない（これが8Gバイト以内にあればOK）。これを「起動コード領界」と呼ぶ。起動コード領界を超える理由など、そうあるものではないと思うかもしれないが、マルチブートを極めていくと意外なところでこの問題にぶつかることもある。

たとえばDOS 6.2xの起動領界2Gバイトだ。Windows 95／98／NTはLinux同様、8Gバイトだが、これらがディスクの先頭に集まるわけで、必然的に領界に余裕のあるWindows 95／98／NTかLinuxが、より後ろのほうに位置することになる。

どうしても8Gバイトを超える領域にインストールせざるを得ない場合でも、ブートパーティションだけは8Gバイト以内の領域に置くようにしたい。「ブートパーティションだけは」というのはピンとこないかもしれない。本特集では、Linuxのファイルシステムを「/」にマウントする1つのパーティションだけで構築したからだ。しかし、Linuxのファイルシステムでディスクを消費するのは、もっぱら/usr以下か、あるいは/home以下である。

そこで、Linuxインストール時に、/usrと/homeをそれぞれ別パーティションに分けてインストールしてしまう方法がある。

この場合、/usrと/home以外の必要最低限のパーティションを8Gバイト以下に設けて/としてマウントし、/usrと/homeのパーティションをそれぞれ8Gバイトを超えた領域に作成して、マウントして使うことができる（**図9**）。

究極のマルチブート環境！？ソフトウェアエミュレータ

1台のPCで複数のOSを使うという点では、ソフトウェアエミュレータを使うという手段もある。Linux上のPCエミュレータとして、VMware社の「VMware for Linux」が有名だ。

VMwareはPCのハードウェアをエミュレーションするため、その上でPC用のOSをブートすることができる。つまりLinux上でWindows 98や、FreeBSDが動くばかりでなく、Linux自体も動かすことができてしまう。

ただしソフトウェアでPCというハードウェアをエミュレーションするため、システムのパフォーマンスを問われる。VMwareの要求するハードウェアスペックは次のようになっている。

- **Pentium II 266MHz以上**
- **96Mバイト以上のRAM**
- **XFree86でサポートされるビデオアダプタ**

また、ソフトウェア要求のほうは、バージョン2.0.32以上のシングルプロセッサカーネルか、2.2.0以上のSMPカーネルに対応している。

動作を確認しているディストリビューションは、Red Hat Linux 5.x／6.0、Caldera OpenLinux 1.3, 2.2／2.3、SuSE Linux 6.0, 6.1 and 6.2だ。XFree86は、3.3.3.1以上が必要である。

VMware for Linuxは現在、Ver.1.1.2の試用版が同社のWebからダウンロード可能である。

http://www.vmware.com/download/download_linux_pre.html

ここから、VMware-112-364.tar.gzをダウンロードする。VMwareのインストールに必要なキーコードも同ページでユーザー登録することにより入手できる。

インストール方法も比較的簡単で、同ページに詳しく解説されているので、マシンパワーに余裕のある向きには、ぜひ試してほしい。

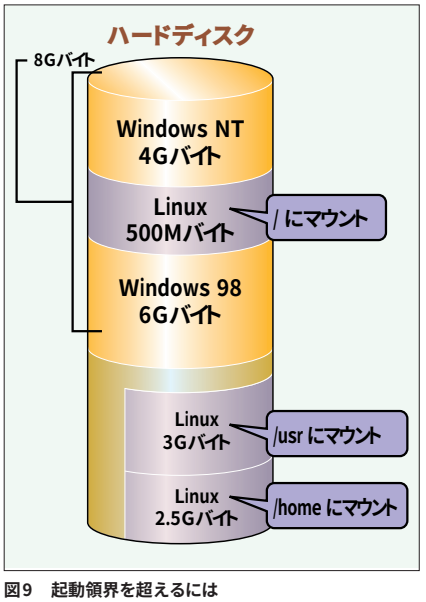


図9　起動領界を超えるには

Windows 2000への期待と憂鬱

Windows 2000+Linuxのトラブル回避

文：政久忠由
Text : Tadayoshi Masahisa

あまりにも前情報が多く、また待たされ過ぎたために食欲が失せた感のあるWindows NTの後継、Windows 2000。そのリリースがやっと実感の湧くものとなってきた。米国では2000年2月17日、日本でも翌日の2月18日に出荷すると発表された（ただし、ProfessionalとServerのみで、Advanced Serverは3月3日、Datacenter Serverは未定）。この1999年12月には、多くの雑誌などでWindows 2000のRC2が収録され、配布される予定であり、多くのユーザーがWindows 2000を実際に体験することになる。

Windows 2000は、リリースが大幅に遅れたうえに、機能もどんどん削られてしまい当初の理想とはほど遠いレベルの実装に落ち着いたものの（まぁ、最初から低い目標では、お金持ちMicrosoft社員のモチベーションもあがらないしね）、それでも従来のバージョンアップと比べると強化、追加された機能は大規模なものとなっている。ユーザーインターフェイスに関しては、Microsoftの常套手段どおり、すでに個別に市場に投入しているコンポーネント（Internet ExplorerやOfficeなど）のユーザー評価の高い部分をシステム標準として組み込んでいる。これはWindows 98とほぼ同等である。オペレーティングシステムの内部的には、ほとんど変わっていないが、プラグアンドプレイやACPIなどのサポートがようやく実装されている。中でも特に期待感もあり、大幅に変更されたのがネットワーク機能である。

ネットワーク機能では、Windows 2000の目玉ともいえるActive Directoryが実装されている。Active Directoryは、ネットワーク上のリソースを一元的に管理し、提供するためのサービスである。このサービスのベースとなっているのは、TCP/IP、そしてそれの上のDNS、LDAP、Kerberos5などのインターネット標準規格だ。Microsoftには珍しく、Windows 2000ではこれらの標準規格にほぼ準拠した実装を

（↙）

（↘）

行っているようだ。このことは、Microsoft以外でActive Directoryと同等の機能を提供する際のハードルが低いことを示唆しているわけだが、実際にそのようなサービスが登場するにはテストを含め1年程度の時間がかかると思われる。

そもそもWindows NTは、過去のしがらみを捨てて新たな地点からスタートしたはずのオペレーティングシステムなのだが、リリースされたものは、Windowsという着物（ユーザーインターフェイス）を着せられ、ネットワーク部には、LanManagerから借りてきたNetBIOSインターフェイスをとってつけただけの、ツギハギだらけのかなりダサいものであった。ユーザーインターフェイスはともかく、ネットワーク部のダサさは、今なお多くのユーザーや管理者を悩ませ続けている。しかし、Windows 2000でやっとこの足かせが解かれることになる。だが、Active Directoryをキチンと理解し、効率のよいネットワークシステムを構築するには、多くの手間と時間が必要とされる。これは、サーバもクライアントもすべてWindows 2000ファミリにしてしまわないとネイティブネットワークの効果がない（にできない）という理由からである。管理者にとっては、とりあえず管理機能は充実したものの、その実、悩みがほかにシフトしただけと感じるだけかもしれない。結局、悩みは尽きないのである。

基本的に今回リリースされるWindows 2000ファミリは、プロフェッショナルとサーバファミリラインで、既存のWindows NTのリプレースということになる。だが、プラグアンドプレイ、パワーマネージメント、USBなどの機能が実装されたことにより、ノートPCを含めた既存のWindows 9xユーザーのリプレースも期待されている（ヘビーゲームユーザーにはちと物足りないだろうが）。このようなことから企業から個人まで、多くのユーザーの移行が予想されるのだ。

さすがに、Linuxだけで構成したネットワークシステムは少ないだろうし、またオープンシステムなのだから適材適所のシステムでネットワークを構成することが望まれている時世であ

る。そういった意味では、LinuxとWindows 2000の混在環境は、あらかじめ問題点をチェックしておく必要があるだろう。ここではLinuxとWindows 2000が混在する環境における予想される問題点を考えてみることにする。

同じシステムに導入する場合はディスク管理システムに注意

まず同じシステムにLinuxとWindows 2000を導入する場合だが、これは従来のLinuxとNTの共存とさほど変わらない。結局のところどちらかを切り替えて使用するわけだから、ほとんど問題が発生することはない。ただし、いくつか注意点があるのでそれらを紹介しておこう。

■新しいディスク管理システムの影響

Windows 2000では、新しいディスク（パーティション）管理を採用している。ダイナミックディスクと呼ばれるこの方式は、従来のディスク管理の基礎であるMBRのパーティションテーブルを利用しないで、パーティション（Windows 2000ではボリュームと呼ぶ）を管理するもので、ディスク構成を変更した際のリブートを軽減している（でもディスク構成を変更したからといって、必ずしもリブートが必要だったわけではなく、ここではミラーセットとかストライピング、マルチボリュームなどの機能の場合）。これは、リモートからディスク管理を効率よく行うための配慮でもある。

現在のLinuxはダイナミックディスク方式のディスク管理には対応していないため、それを適用したディスクを扱うことができない。一応システムID（23）の不明なディスク全体がひとつのパーティションとして見えるだけだ。

ところがである、Windows 2000もこのダイナミックディスク方式のディスクには、インストールできない。従来の方式のディスクにインストールしたあと、そのディスクをダイナミックディスク方式にアップグレードすることはできるが、最初からダイナミックディスク方式にしたディスクには、インストールしようとしても、認識できないと表示されてしまうのである。つまりブートシーケンスが要求される場合にはこのダイナミックディスク方式は利用できないのだ。これは、システムのブートシーケンスが昔のままであるからにほかならない。完全にほかを無視すれば、ダイナミックディスクからでも起動するようにできたはずだが、今のところは多少互換性を気にしているのだろう（単なる手抜きかもしれないが）。

つまるところ、ダイナミックディスク方式は、Linuxのみならず、従来のWindows、そしてWindows 2000でさえ、注意しなければならないものということになる。

■ハードウェア自体の問題も少々

現状のLinuxはプラグアンドプレイやACPIなどを正式にはサポートしていない（現在開発途中）。一方、Windows 2000はこれらをサポートしており、ハードウェアがサポートしていれば、ドライバはこれらの機能を使用してコントロールを行う。ここで、ソフトウェアリセットで各OSを切り替えようとした場合に、まれにハードウェアデバイスが初期化されないことがある。これはACPIなどに対して、各デバイス、そしてシステムBIOSがいまだ手探り状態なためで、OSだけの問題とはいえない。これに関しては、結局のところ、きちんと実績のあるハードウェアを選択する以外に道はない。

問題になる可能性のあるネットワーク機能

Windows 2000では、ネットワークのベースプロトコルをTCP/IPに全面的に移行していて、従来のNetBIOSインターフェイスは互換性のためだけに残されている。だが、従来のWindows環境が一気にWindows 2000に移行できるとは考えにくく、当面は従来と互換性のあるMixedモードが使用されるだろう。このモードでは、一般的なファイル、プリンタ共有において、従来と同じ認証、アクセス方式が保証されるので、LinuxのSamba環境においても、特別問題が生じるものではない。Windows 2000のネイティブモードでは、UNIXと同様、Socketインターフェイス、そしてインターネット標準規格が利用されることになる。Microsoftの独自路線は少なくなっているようなので、実際Sambaが正式に対応するのは、時間の問題だと思われる。ただし、Active Directoryでのドメインコントローラとしては、COM+への対応などもあるので未知数だ。

では、ネットワーク機能におけるWindows 2000とLinuxの互換性について考えてみよう。まずは、問題が生じそうな主なネットワークサービスを列挙してみる。

- ファイル、プリンタ共有
- DNS、DHCP
- タイムサーバ
- telnetサービス

ここほか、IPSec、QoSなど、さまざまなネットワーク機能もあるが、あまり一般用途で利用されるものでもないのでここ

では省略する。

■ファイル、プリンタ共有

まずファイル、プリンタ共有については、Linuxというよりも、これはSambaの対応はどうかということができる。一番大きな問題となるのが認証システムということになるが、Windows 2000のネイティブシステムではKerberos5が採用されている。Kerberos5自体は、そもそもUNIX上で開発が進められてきた認証システムで標準規格化もなされているので、現時点でSamba自体に取り込むことも可能だし、Linuxシステム自体に別途サービスを組み込むことも難しくはない。だからそれほど心配することはないのだが、Kerberos5があの米国の輸出規制に引っかかるようなので、日本で運用してもよいかどうかは不透明だ。

またドメイン環境に関しては、Sambaは開発が進められている最中で、Active Directoryに関しても、現状のLDAPの実装レベルで肩代わりできるかどうかは、もう少し検証が必要だ（理論的にはさほど修正しなくても実現できるはず）。しかし冒頭で述べたとおり、実際問題としてはすぐにネットワーク上のWindowsファミリがすべてWindows 2000にリプレースされるとは考えにくい。つまりここ1、2年は互換モードでネットワークが運用されていくと思われる。互換モードでの運用は、すでに検証が進められていて、Sambaにはすでにいくつか修正が加えられている。実際、互換モードでsamba-2.0.6と接続してみたが、今までどおりに利用できている。

■DNS、DHCP

DNSとDHCPは、Active Directoryドメイン環境では、必須のサービスとなる。ワークグループ環境でもWindows、Linuxを問わず、できることなら運用したいサービスである。

DNSとDHCPは、ともに標準化されているもので、きちんとRFCにそって実装されていれば、この両サービスがどのOS上で行われていても何ら問題はない。ただし、注意点としては、DNSのDynamic Updates (RFC 2136) 機能とActive Directoryの場合はさらにSRVレコードのサポートがなされていることが条件となる。これらの機能は、DNSのリファレンスともいえるBIND Version 8では、当然サポートされている。しかし、BIND Version 4ではサポートしていないので古いシステムの場合は注意が必要だ。また、バグやセキュリティホールの問題もあるので、BIND Version 8もできる限り、最新のものを利用することをすすめる（1999年11月23日現在、8.2.2-P5）。

BIND Version 8では、セキュリティ問題にも関係してくるので、Dynamic Updatesはデフォルトで無効になっている。そこで、少しだけDynamic Updatesのためのポイントを紹介しておこう。以下は、設定ファイル（named.conf）の抜粋である。

```
zone "tmnet.com" {
        type master;
        file "master/tmnet.com";
        allow-update {192.168.1/24;};
};

zone "1.168.192.in-addr.arpa" {
        type master;
        file "master/192.168.1";
        allow-update {192.168.1/24;};
};
```

ポイントは、"allow-update {192.168.1/24;};"だ。これにより、ゾーンにおける指定した範囲（192.168.1/24）からのレコードのアップデート要求を許可している。レコードが追加された場合のBINDのログは次のようになる。

```
;BIND LOG V8
[DYNAMIC_UPDATE] id 26 from [192.168.1.200].1067 at 943552786
(named pid 332):
zone:    origin tmnet.com class IN serial 39
prereq:  {nxrrset} TM04.tmnet.com. IN CNAME
prereq:  {nxrrset} TM04.tmnet.com. IN A
update:  {add} TM04.tmnet.com. 1200 IN A 192.168.1.200

;BIND LOG V8
[DYNAMIC_UPDATE] id 28 from [192.168.1.200].1069 at 943552786
(named pid 332):
zone:    origin 1.168.192.in-addr.arpa class IN serial 19
update:  {add}  200.1.168.192.in-addr.arpa.  1200  IN  PTR
TM04.tmnet.com.
```

DHCPでは、DNSとの連携（DHCPでIPアドレスをリリースした段階でDNSに登録するような機能など）が求められるが必須ではない。

以上のようにDNSとDHCPに関しては、特に問題はないの

だが、そもそもDNS Dynamic Updatesは本当に便利で安全なのかという別問題が浮上してくる。

上記のような場合、各マシンに設定したホスト名が不動の絶対的な意味を持ち、IPアドレスが動的に割り当てられることになる。しかし、管理者から見ると各クライアントのホスト名なんて何でもよくて、EthernetカードのMACアドレスとIPアドレスの組み合わせを管理することで不正なアクセスを防止し、またクライアントのインストールの段階ではマシン名を指定しなくても済むようにしたいと考える人も多いと思う。つまり、管理したMACアドレスに対してのみIPアドレスを配布し、ホスト名は事前にスタティックに登録したDNSから逆引きして設定できたほうが手間もかからず何かと便利なのだ。事実、Red Hat　Linuxをはじめとする多くのLinuxディストリビューションでネットワーク構成にDHCPを利用するとDNSなどから名前を解決して、自ホスト名として登録できるようになっている。しかし、Windows NT、そしてWindows 2000ではコンピュータ名もユーザーアカウント同様の扱いがなされ、ID（SIDと呼ばれる）とパスワードが管理されているために、この管理機構とは別次元のDNSとDHCPのレベルで名前が解決されては困るのだ（SIDなので必ずしもホスト名というわけではないのだが、関連付けされている）。NetBIOS名とホスト名の両方が有効な現状では、どちらか（通常NetBIOS名）が対応づけられていればよいので問題ないが、Windows 2000のネイティブモードであるNetBIOS名を利用しない環境では、コンピュータがドメインのメンバーとして認識されないなどの不具合が生じてしまう可能性がある。

利便性と安全性、そして管理ポリシーに関わるもので、一概にどの方式がよいとはいえないのだが、ホスト名、IPアドレス、MACアドレス、そしてCPUのシリアルナンバーなどなど、ネットワーク上のマシンなどのノードを何で管理するかは今後も難しい問題として残っている。

また中には、UNIX上で動作している古いDNSであるとか、BIND8でもDynamic Updatesは許可したくないという管理者もいることだろう。この場合は、各Windows 2000クライアントの［ネットワークプロパティ］－［インターネットプロトコル（TCP/IP）］－［DNS設定］－［詳細設定］で、「この接続のアドレスをDNSに登録する」をチェックアウトする必要がある。デフォルトで有効になっているので少々面倒だが、すべてのクライアントをこのように設定しなければDNSのエラーログがいっぱいになってしまう。

なおDNSとDHCPについては、管理者とユーザーの双方の負担が軽減されること請け合いなので、小規模な場合でも、ぜひとも導入を検討してもらいたい。もちろんLinuxでの構築をお勧めする。

■時間を合わせる必要性

Windows 2000では、さまざまなリソースを一元管理するActive Directory、そしてユーザーのデータや設定情報などを管理するIntelliMirrorなど、複数のマシン間でのデータの同期が頻繁に発生する。この場合、時間が正確に合っていないと何かと都合が悪い。特にPCの時計の精度はまったく当てにならず、日に数分ずれることもざらだ。そこで必要となるのが、タイムサーバサービスだ。従来のWindows NTでは、NetBIOSインターフェイスを利用した"net time"コマンドが利用できたが、使い勝手も精度も悪い。Windows 2000では、標準規格のネットワークタイムプロトコル（NTP）を実装したサービス（W32Time）が標準で付属している。これでやっとネットワーク上のマシンの時間を効率よく合わせられるようになったわけだ。当然NTPに対応したLinux上のサービスとの相性も一応問題ないようだ。

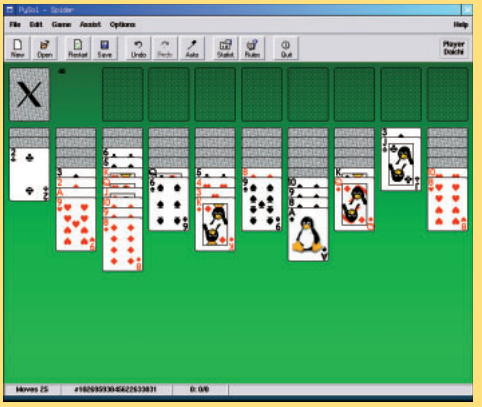
■telnetサービス

Windows 2000 Serverファミリには、telnetサーバサービスが付属している。コンソール環境の充実していないWindows 2000にtelnet接続しても、それほどうれしくないのだけれども、多少は管理に役立つだろう。ただし、認証方式がプレーンテキストではなく、NTLM認証がデフォルト設定になっているようなので注意してほしい。これは設定を変更すればよいだけだ。

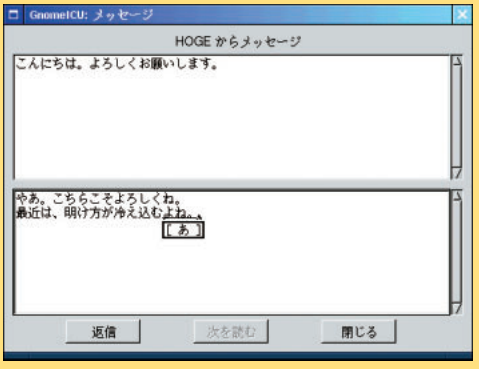
ワークグループ的用途では、従来の環境とそれほど変化がないので、特に目立った不具合は発生しない。しかし、Active Directory、そしてネイティブモードでの運用はいまだ未知数的な要素が多いのでどんな問題が生じるかはわからないといったところが現状である。しかし、ちょっとした不具合は、これまでどおりSamba側などのLinux上のサービス側で迅速に対応していくだろう。また、Windows 2000 Serverファミリなしで、Active Directoryを構築できる日もそう遠くないと思われる。ただし、Active Directoryが市場でどのように評価されていくかという根本的な問題もあり（事実、現状NDS（Novell Directory Services）の評価が高い）、CORBAをベースにした別のディレクトリサービスが登場し、主流になる可能性がないとも限らないのである。

Free Application Showcase

文：出井 一
Text : Hajime Dei



Pysol P.148



GnomeICU P.142

XRACER P.151

※ のついたソフトは、付録のCD-ROMに収録されています。

KDE付属の使いやすいメールクライアント

Kmail

バージョン：1.0.28　　種別：GPL

http://www.kde.org/
http://www.kde.gr.jp/（日本語化）

Kmailは、KDEの基本パッケージに含まれるメールクライアントソフトだ。日本語化されたKDEに付属するKmailでは、文字コードの変換が正しく行われるのはもちろん、メニューやダイアログがすべて日本語で表示される。ウィンドウはWindowsのメールクライアントで一般的な3ペイン方式で、複数のメールアカウントを使い分けたり、受信したメールをフォルダに振り分けたりすることも可能だ。

●インストールと初期設定

日本語に対応したKmailは、日本語化されたKDE（最新版は1.1.2）の基本パッケージのひとつ「kdenetwork」に含まれている。KDEの導入方法などについては、「日本KDEユーザ会ホームページ」（http://www.kde.gr.jp/）を参照されたい。上記ページからリンクされているFTPサイトでは、各種ディストリビューション用のRPMバイナリパッケージやtarボールなどを入手できる。

ktermなどのコマンドラインから「kmail&」として起動するか、Kpanel左端のボタンを押して、[インターネット]－[メールクライアント]を選択すると、Kmailのウィンドウが開く（**画面1**）。ウィンドウは、メールを分類管理するフォルダ、フォルダ内のメール一覧、メールの内容表示と3つのペイン（領域）に分かれている。

初回起動時には自動的にメールボックスが作成され、設定ダイアログが開く。このダイアログは[ファイル]－[設定]でいつでも開けるので、まずは名前とメールアドレス、ネットワーク関係など必須の設定だけ行う。

Kmailでは、sendmailなどを使って自前のメールサーバを運用している環境と、ダイヤルアップ接続によりプロバイダのサーバ（SMTP、POP3）を利用する環境の両方に対応している。また、受信用のアカウントは複数登録できるので、複数のメールアドレスを使い分けている場合でも、一度の操作でまとめてメールを受信できる。

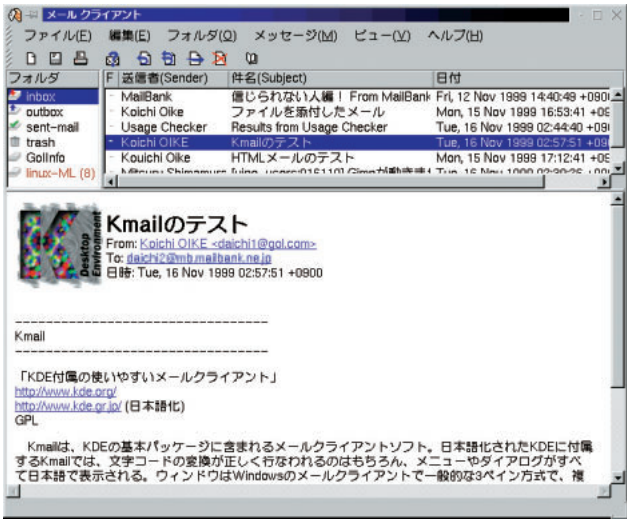
プロバイダにダイヤルアップ接続している環境では、[ネットワーク]ページで[SMTP]をチェックし、プロバイダのSMTPサーバ名を設定する（**画面2**）。受信に関する設定は、[追加]ボタンを押してアカウントの選択で[POP3]をチェックし、POPアカウント用のサーバ名、ログイン名、パスワードなどを設定すればいい（**画面3**）。

●メールの受信と閲覧

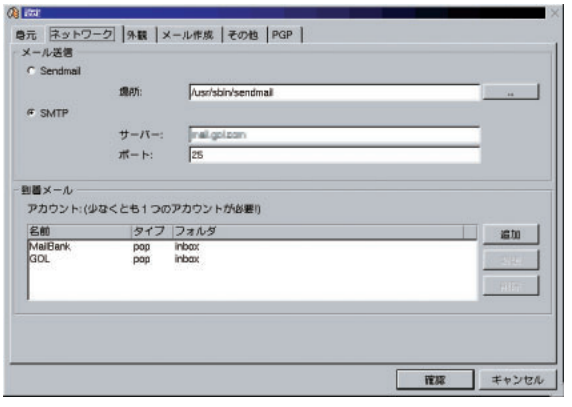
ツールバー左から4番目の[メールをダウンロード]ボタンを押すと、受信用のアカウントすべてに対して新着メールのチェックと受信が行われる。それぞれのアカウントごとにメールチェックを行うことも可能だ。メールの受信中は、小型のウィンドウが開いて、受信済みのメール数などが表示される（**画面4**）。

受信したメールは、初期設定では[inbox]フォルダに置かれ、フォルダ表示の右端に未読メールの数が表示される。また、メール一覧でも未読メールは異なる色で表示される。

メールの閲覧操作には、マウス以外にキーボードも利用できる。たとえば、n／pキーで次／前のメール、＋／－キーで次／前の未読メール、PageUp／PageDownキーで本文のスクロールと



画面1
Kmailのウィンドウ。メニューなどは完全に日本語化されている。



画面2
プロバイダのメールサーバを利用する場合は[SMTP]をチェック。

いった具合だ。残念ながら、スペースキーだけで本文のスクロールと次の未読メールへの移動をインテリジェントに行う機能は用意されていない。

●分類用のフォルダを作成する

初期設定では、[inbox]のほか、送信待ちメール用の[outbox]、送信済みメール用の[sent-mail]、削除メール用の[trash]フォルダが用意されている。ユーザーによる整理用フォルダの追加も可能だが（**画面5**）、フォルダ中にサブフォルダを作成する階層化には対応していない。

メールを別フォルダへ移動するには、マウスでドラッグ&ドロップすればいい。複数のメールをShift-クリックで選択しておき、まとめてドラッグすることも可能だ。なお、ツールバーの[削除]ボタンで削除したメールは[trash]フォルダに移動され、右クリックメニューの[空に]を選択するまでは実際には削除されない。

このほか、複数のメールアドレスを登録している場合は、POPアカウントの設定（**画面3**）により、受信時にメールを格納するフォルダを指定できる。また、サブジェクトや送信者アドレスなどによって、メールを自動的にフォルダに振り分けるには、後述のフィルタ機能を利用すればいい。

●メールの作成と送信

ツールバー左端の[新規メール作成]ボタンを押すと、Kmailのメール作成ウィンドウが開く（**画面6**）。返信や転送の場合は、[発信者に返信]や[転送]ボタンを利用する。

メール本文には日本語をそのまま入力、編集できる。80桁で文字を折り返す設定になっているが、単語間にスペースを含まない日本語の文章では機能しない。このほか、英文のスペルチェックも可能だ。

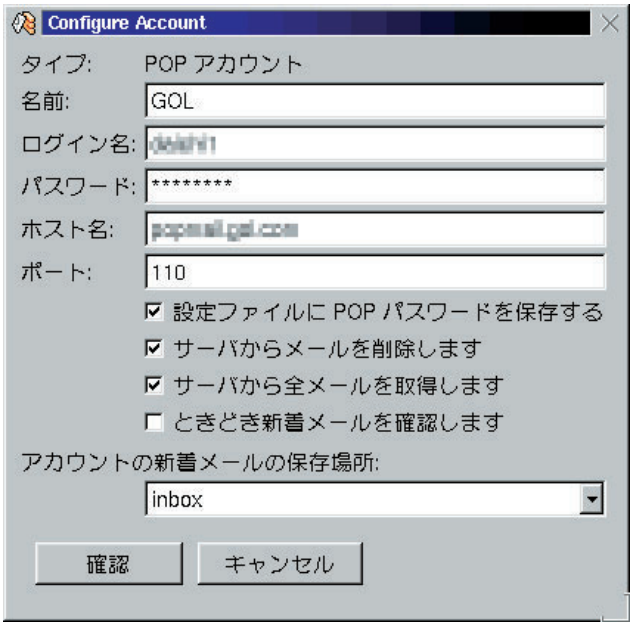
なお、メール作成ウィンドウと一緒にファイル選択ダイアログが開く場合は、設定ダイアログの[身元]ページで[シグネチャファイル]を設定しよう。エディタなどで作成したテキストファイルを指定すればいい。これで、自動的に本文にシグネチャ（署名）が挿入されるようになる（変更可能）。

メールを書き終えたら、ツールバー左端の[送信]ボタンで即座に送信される。このほか、左から3番目の[後で送信]ボタンを押して、いったん[outbox]フォルダに保管しておき、後で[ファイル]－[送信待ちメールを送信]を選択することで複数のメールをまとめて送信することも可能だ。

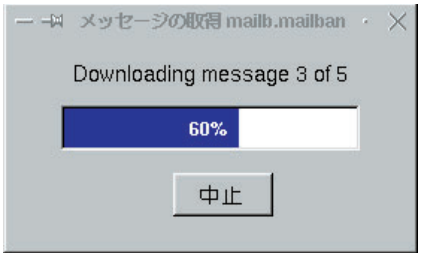
●ファイルの添付

Kmailは、マルチパートMIMEを利用した添付ファイルにも対応している。受信メールにファイルが添付されている場合は、ウィンドウ下部にそのアイコンとファイル名が表示される。これらをクリックすると、画像ならKView、アーカイブならarkといった具合にMIMEタイプに応じて適切なアプリが起動してファイルを開く（**画面7**）。なお、なにかと問題の多いHTMLメールには対応しておらず、マルチパートの内容がそのまま本文に表示される。

メール作成時にファイルを添付するには、kfmなどのファイルマネージャからメール作成ウィンドウにファイルをドラッグ&ドロップする方法と、[添

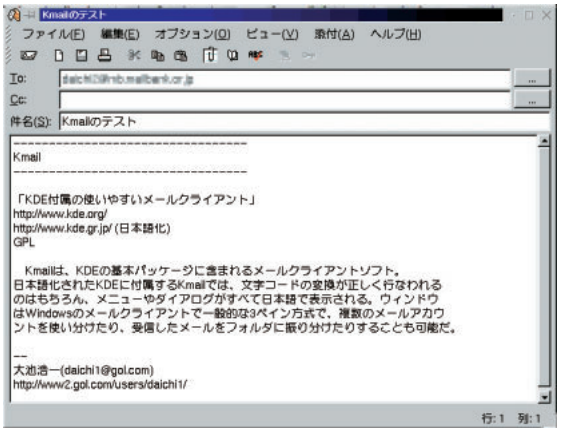


画面3
受信用のPOPアカウントは複数登録可能だ。



画面4
受信中は受信したメール数などがグラフとともに表示される。

画面5
ユーザーがメールを整理するためのフォルダを追加できる。



画面6
新規メールや返信メールの作成はこのウィンドウで行う。

画面7
添付ファイルはアイコン表示され、クリックでアプリが起動する。

付ファイル]ボタンを押してリストから選択する方法が用意されている。どちらの場合も、プロパティダイアログが開いて、MIMEタイプやコーディング方法が表示されるので、[確認]ボタンを押せばいい。添付ファイル名やMIMEタイプがメール作成ウィンドウの下側に一覧表示される。

●本文中のメールアドレスと住所録

本文中のメールアドレスやWeb／FTPのURLは、Kmailによって自動的に判別され、青いアンダーライン付きで表示される。これらをクリックすると、メールアドレスに対するメールを作成したり、WebページやFTPサイトをkfmで表示したりできる。一方、右クリックではメニューがポップアップし、メールアドレスの住所録への追加や、URLのクリップボードへのコピーなどを行える。

住所録に登録したメールアドレスは、メール作成ウィンドウで、[To:]や[Cc:]の右端にある[...]ボタンを押すと参照できる。住所録といっても、各人の名前とメールアドレスが記録されただけの単純なものだ（**画面8**）。ここで直接メールアドレスを入力して登録することもできる。フォルダによるグループ分けは行えないので、大量のメールアドレスを管理するにはちょっと力不足だ。

●フィルタを利用した振り分け

最後に、フィルタを利用したメールの振り分け処理について説明しよう。振り分けのための設定は、[ファイル]－[フィルタ]のフィルタルールダイアログで行う（**画面9**）。

フィルタルールは、条件部と実行部に分かれている。条件部では、ヘッダやメール本文が、指定した文字列を含んでいたり、一致しているかどうかで判断され（正規表現も使用可能）、2つの判断を組み合わせられる。条件に当てはまるメールに対しては、実行部の処理が行われる。指定したフォルダに振り分けるほか、他のメールアドレスに転送したり、アプリを実行したりすることも可能だ。

振り分け処理はメールの受信直後に自動的に行われるため、通常はユーザーが指定する必要はない。ただし、フィルタルールを追加した後など、改めて振り分け処理を行いたい場合は、対象とするファイルを選択後、[メッセージ]－[フィルタ適用]を選択する。たとえば、フォルダ内の全ファイルに対してフィルタを適用したいなら、[メッセージ]－[全てマーク]を選択してから、改めて[メッセージ]－[フィルタ適用]を選択すればいい（キー操作ではK、Ctrl-Jとする）。

画面8
送信先メールアドレスを管理する住所録。

画面9
フォルダへの振り分け処理などのためのルールを設定する。

テキストベースの高速ニュースリーダ

slrn

バージョン：0.9.5.7　　**種別：GPL**

http://space.mit.edu/~/davis/slrn.html
http://kondara.sdri.co.jp/~kikutani/slrn.html(日本語化)

slrnは、テキストベースで動作する高速なニュースリーダだ。画面表示はEmacs上で動作するGnusに似ており、記事のスレッド表示にも対応している。C言語風のマクロ言語「S-Lang」を利用して細かなカスタマイズが可能だ。日本語対応と各種の機能拡張を可能にする日本語パッチが用意されているほか、ニュースサーバから記事をまとめてダウンロードするslrnpullも同梱されている。

●ビルドとインストール

slrnは、ファイル一式をtar＋gzipしたtarボールで配布されている。さらに、菊谷、勝田両氏による日本語化パッチを当てることにより、日本語の記事の表示、ヘルプメッセージの日本語化、マルチパートMIME対応、クリッカブルURLなど、日本語対応と各種の機能拡張が行われる。

日本語パッチを当てるには、slrnのtarボールを展開したディレクトリにパッチファイルをコピーし、「zcat slrn0.9.5.7jp0.pat.gz | patch -p1」とすればいい。その後、「./configure」→「make」という一般的な手順でビルドできる。slrnpullも「make slrnpull」としてビルドしておこう。その後「make install」で両方とも/usr/local/binにインストールされる。

なお、日本語パッチを当てたslrnのビルドと実行には日本語対応のS-Langライブラリが必要だ。幸いなことに、VineやLASER5、Turboには、日本語対応の1.2.2が、Red Hat 6.1の日本語版には最新版の1.3.8が最初から含まれている。

●初期設定を行う

slrnは、ホームディレクトリの設定ファイル「.slrnrc」から、端末ソフト（kterm、rxvtなど）ごとに異なる文字色の設定や、操作に使うキー設定、細かな動作設定などを読み込む。中には日本語の読み書きに必須の設定も含まれているので、まずはサンプル設定ファイル（slrn.rc）をホームディレクトリの設定ファイル（.slrnrc）にコピーし、ユーザーの環境に合わせて何カ所か修正する必要がある。

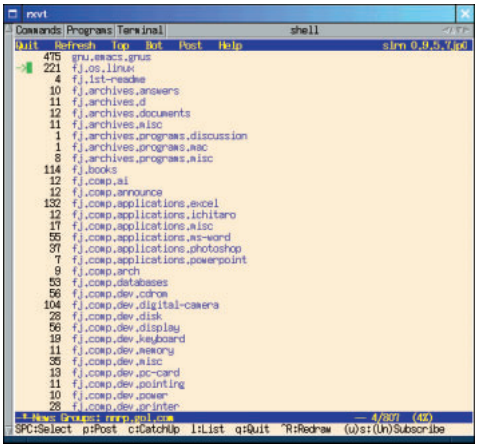
具体的には、tarボールの展開先で「cp doc/slrn.rc ~/.slrnrc」としてコピーする。とりあえず修正が必要なのは、最初のほうにあるホスト名（hostname）、ユーザー名（set username）、名前（set realname）といったところ。先頭の「%」によりコメントになっているので、「%」を削除して正しい内容を設定しよう。

このほか、利用するニュースサーバ名を、「export NNTPSERVER=nntp.hoge.ne.jp」などとして、環境変数NNTPSERVERに設定する必要がある（コマンドラインで直接指定することも可能）。slrnを常用する場合は「.bash_profile」に書いておくとよいだろう。

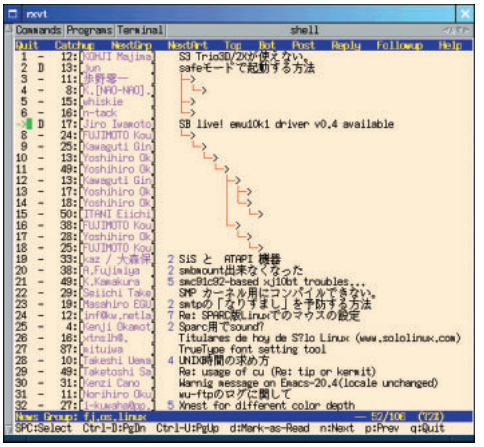
●初めて起動する場合

slrnを起動するには、コンソールや端末ソフトのコマンドラインで「slrn」とすればいい。なお、X上で利用する場合、ktermのデフォルトの配色では見づらいため、別の端末ソフトrxvtを利用するとよいだろう。ktermから「rxvt&」として起動し、rxvtのコマンドラインでslrnを起動する。

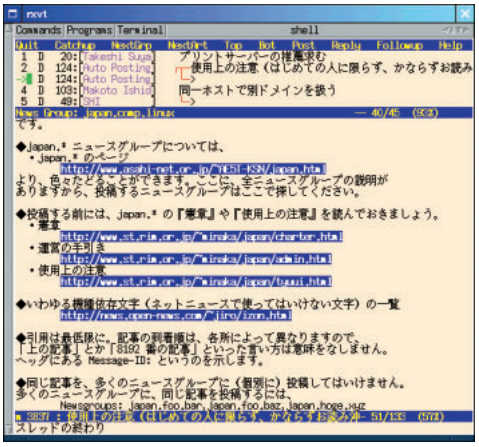
ところで、slrnでは、ニュースグループの一覧や未読管理用の情報を、ホームディレクトリの「.jnewsrc」に保存する。すでにGnusでネットニュースを読んだことがある場合は、Gnusが生成した.newsrcの内容を初回起動時に読み込み、.jnewsrcを自動的に生成し



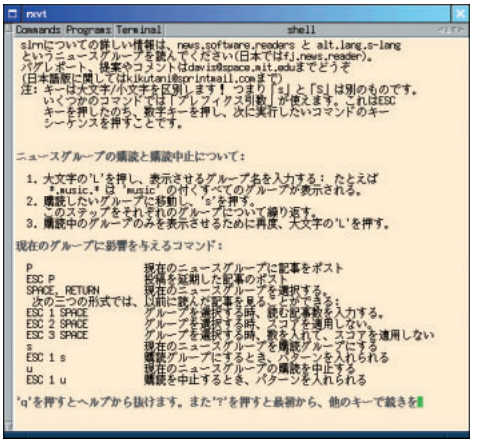
画面1
最初に、ニュースグループの一覧が表示される。



画面2
ニュースグループ内の記事の一覧。スレッド表示に注目。



画面3
記事の内容中のURLや引用部は異なる色で表示される。



画面4
?キーを押すと、キー操作のヘルプが表示される

てくれる。

しかし、これまでにGnusを使ってネットニュースを読んだことがない（ホームディレクトリに.newsrcが存在しない）場合は、ニュースサーバからニュースグループ一覧を取得して.jnewsrcを生成するようにslrnに伝えなければならない。それには、「slrn -f .jnewsrc -create」とする。

たいていのプロバイダのニュースサーバは、国内外の何千ものニュースグループを扱っているため、一覧の取得にはかなりの時間がかかる。.jnewsrcを一度生成してしまえば、その後は新たなニュースグループがときおり追加される程度なので、それほど時間はかからない。

●ニュースグループの記事を読む

ニュースサーバに接続すると、まずニュースグループの一覧画面が表示される（**画面1**）。↑／↓キーやCtrl-P／Nキーで、カーソル「−>」を読みたいニュースグループまで移動させ、スペースキーを押す。もし、読む必要のないグループがあったら、カーソルを移動した後、uキーで購読を中止できる。購読を中止したグループ名の左には「U」が表示され、Esc、1、Lキーで表示から除外できる（購読再開も可能）。

ニュースグループに入ると、記事の一覧が表示され（**画面2**）、読みたい記事にカーソルを合わせてスペースキーを押すと、記事の本文が表示される（**画面3**）。サブジェクト左の数字はスレッド数で、読む際には罫線によってスレッドの構造が示される。

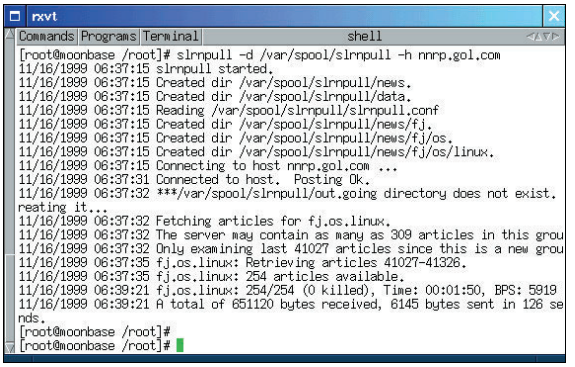
投稿されている記事を端から読み進めていく場合、基本的にはスペースキーだけ押していればいい。記事の終わりまで自動的にスクロールし、次の未読記事に切り替わる。さらに、ニュースグループ内の最後の未読記事だった場合は、次のニュースグループに移動する。なお、初期設定では、記事の終わりや最後の未読記事で、いったん停止する。.slrnrcの150行付近の「set query_next_group 1」と「set query_next_article 1」の1を0に変更すると停止しないようになる。

その他のキー操作については、各画面で？キーを押すと表示される日本語のヘルプ（**画面4**）を参照されたい。基本的にはGnusのキーバインドに準拠している。また、マウスを利用して記事を読むことも可能だ。

●スコアファイル機能とslrnpull

slrnでは、しつこいSPAM記事を排除するため、サブジェクトやユーザー名などに応じて記事を採点し、0点以下なら最初から既読（あるいは削除）状態にするスコアファイル機能が用意されている。採点方法などの詳細は、日本語化slrnページからのリンクでたどれる「score.jis」「KILL_FAQ.jis」を参照されたい。

また、付属のslrnpullを使うと、ニュースサーバの記事をローカルスプールにダウンロードし（**画面5**）、slrnを--spoolオプション付きで起動してオフラインでゆっくり記事を読むことも可能だ。上記ページからリンクされている「README.slrnpull.jis」に、slrnpullの日本語解説がある。



画面5
slrnpullを使って、一気に記事をダウンロードする。

日本語対応のGNOME版ICQクライアント

GnomeICU

バージョン：0.65　　種別：GPL

http://gnomeicu.gdev.net/
http://northeye.org/gnomeicu-ja/（日本語化）

GnomeICUは、ユーザー数が2000万人を超えるコミュニケーションソフト「ICQ」互換のクライアントソフトだ。メッセージ交換やチャット、ファイル転送など、ICQの機能をほぼサポートしている。ふだんはGNOMEのパネルに常駐しており、ダブルクリックするとユーザー一覧を含むウィンドウが開く。メニュー表示やメッセージを日本語対応にするためのパッチも用意されている。動作にはGTK+とGNOMEが必要だ。

●日本語化パッチとビルド

GnomeICUは、ファイル一式をtar＋gzipしたtarボールと、RPM形式のパッケージが用意されている。ただし、オリジナルのGnomeICUは日本語を想定していないため、日本語のメッセージをEUCのまま送信してしまう。WindowsのICQと日本語でメッセージをやりとりするには、北目氏の「GnomeICU日本語化」ページにある日本語化パッチを利用して、送受信時の文字コード変換などの修正を行う必要がある。

パッチ済みのRPM形式のパッケージは「Vine Plus」（http://vine.flatout.org/vineplus.html）などから入手できる。ここでは、tarボールにパッチを当てて日本語対応のGnomeICUを作成する方法を説明しよう。オリジナルのtarボールを展開したディレクトリにパッチをコピーして、「patch -p1 < gnomeicu-0.65-ja3.patch」としてパッチを当てる。あとは、「./configure」→「make」→「make install」と一般的な手順でビルドとインストールを行えばいい。

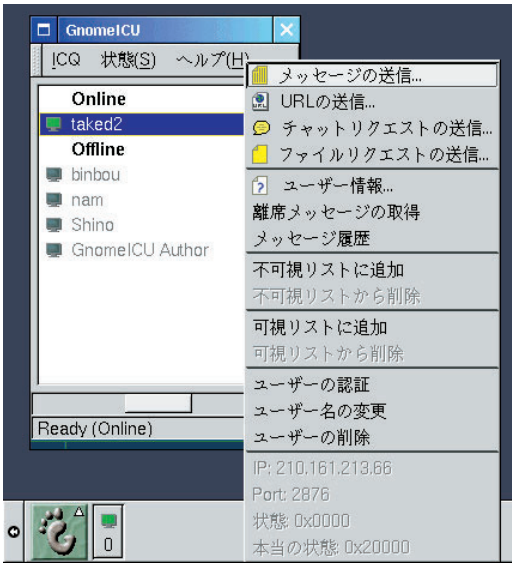
●UINの登録と初期設定

「gnomeicu&」として起動すると、パネルに小さなボタンが表示される。GnomeICUのウィンドウを開くには、このボタンをダブルクリックすればいい（**画面1**）。

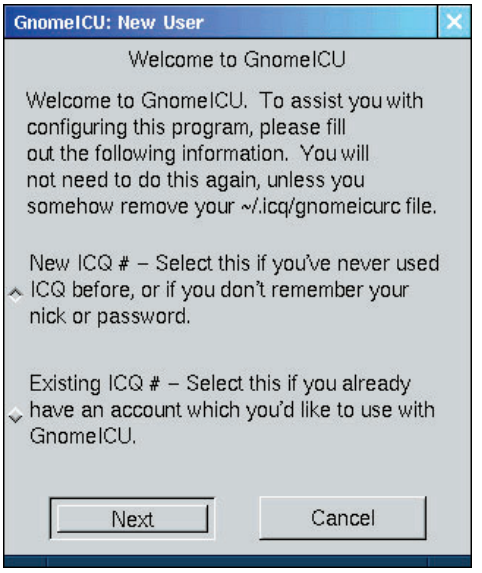
初めて起動した場合は、UIN（ユーザー識別番号）の入力を求められる（**画面2**）。ICQの特徴はこのUINによるユーザー管理にあり、どこから接続しても同じUINなら同一ユーザーだと判断され、同じマシンからでもUINを切り替えれば別のユーザーとして扱われる。

これまでにICQを使ったことのない人は、「New ICQ#」をチェックして、ICQユーザーとしての新規登録を行う。アクセス用のパスワードを入力してしばらく待っていると、ICQサーバから発行されたUINを知らせるメッセージボックスと、個人情報設定用ダイアログが開く（**画面3**）。他のユーザーに公開する情報（ニックネームなど）を差し支えない範囲で設定しよう。

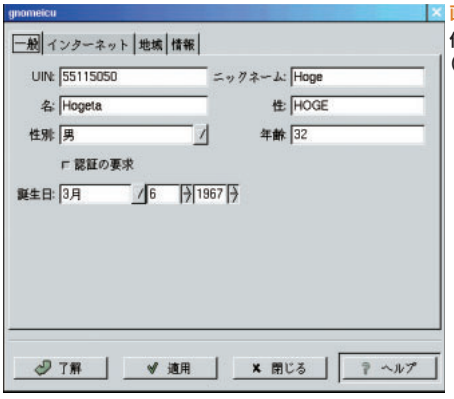
一方、すでにWindowsマシンでICQを使っている人は、あらかじめ自分のUINをICQのウィンドウのタイトルバーや個人情報ダイアログで確認しておき、ここでは「Existing ICQ#」を選択してUINとパスワードを入力する。そのままオンライン状態になるので、パネルのボタンをダブルクリックしてウィンドウを開こう。あとは、[ICQ]－[個人情報の変更]で個人情報設定用ダイアログを開き、ニックネームなどを設定すればいい。



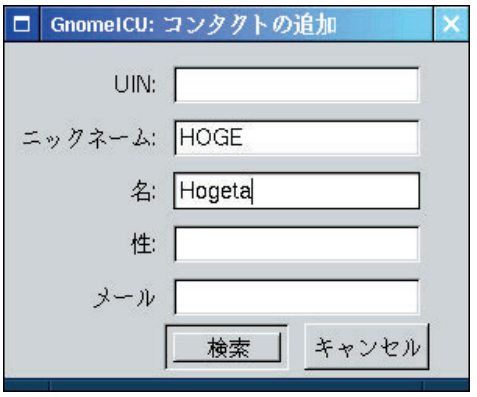
画面1
友達などが現在オンラインかどうか一目でわかる。



画面2
UINを新規発行するか、すでに持っているかを選択する。



画面3
他のユーザーに公開する個人情報（ニックネームなど）を登録。



画面4
コンタクトリストにICQを使っている友達を登録しよう。

なお、こうした設定は、ホームディレクトリの設定ファイル（~/.gnome/GnomeICU）に保存され、次回からは自動的に読み込まれる。

●ユーザーをリストに追加

続いての作業は、ICQを使っている友人などをコンタクトリストに登録することだ。GnomeICUのウィンドウには、コンタクトリストに登録したユーザーだけが表示され、メッセージの交換やチャットなどを行える。

[ICQ]－[コンタクトの追加]を選択すると、登録用ダイアログが開く（**画面4**）。UINを直接入力するほか、（少し時間がかかるが）名前やメールアドレスによる検索も可能だ。UINを入力した場合は直接、他の情報から検索した場合は候補一覧リストを経由して、ユーザー情報ダイアログが開く。正しい相手かどうか確認したら[ユーザの追加]ボタンで登録しよう。なお、相手の設定によっては、登録時に相手の認証が必要なこともある。

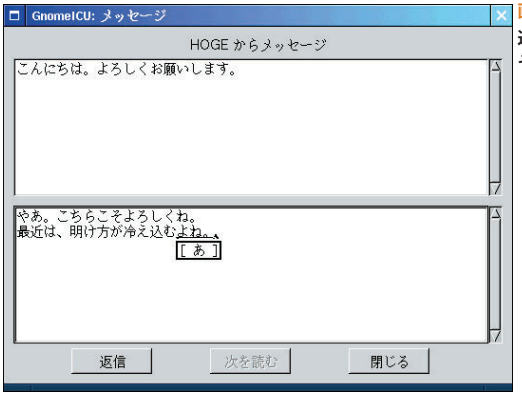
登録が完了すると、そのユーザーのUIN（しばらくするとニックネームに変わる）がウィンドウに表示され、ICQを使用中なら[Online]に、そうでなければ[Offline]に並んで、接続状況を知らせてくれる。

●メッセージ交換やチャットなど

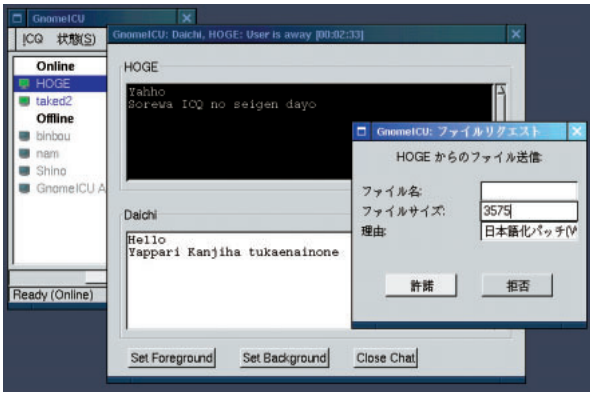
コンタクトリストに登録したユーザーとは、自由にメッセージの交換やチャット、ファイル転送を行える。こうした操作には、マウスの右クリックメニューを利用する。

メッセージの送信は、対象となるユーザーを右クリックし、メニューから[メッセージ送信]を選択する。ダイアログが開いたら、伝えたいメッセージの内容を入力して[送信]ボタンを押せばいい。相手がICQを使用中（オンライン）ならばその場で、ICQを使っていない状態（オフライン）ならば、次にICQを起動した時点でメッセージが届けられる。一方、他のユーザーからのメッセージを受信した場合は、そのユーザーのアイコンが書類の形に変化し、ダブルクリックするとメッセージの内容が表示される。その場で返事を書くことも可能だ（**画面5**）。

チャットをする場合は、相手がオンラインでなければならない。最初に右クリックメニューの[チャットリクエストの送信]で、チャットしたいことを伝える。相手が受諾するとチャット用ウィンドウが開いて、リアルタイムに文字による対話を行える（**画面6**）。ただし、ICQのシステムの制約により、チャットでは日本語を使用できないので注意されたい。ファイル転送もチャットの場合と同様で、最初にリクエストを送り、相手がファイルの受信を受諾した場合にだけ実際の転送処理が行われる。



画面5
送られてきたメッセージにその場で返事を書く。



画面6
リアルタイムなチャットやファイル転送も可能だ。

フレームバッファコンソールで使える日本語ターミナル

JFBTERM/ME

バージョン：0.3.5（開発版）/0.2.3（安定志向版）種別：BSDスタイル

http://www3.justnet.ne.jp/~nmasu/linux/jfbterm/indexn.html

JFBTERM/MEは、Linuxのカーネル2.2以降のフレームバッファコンソールで利用できる日本語ターミナルソフトだ。フレームバッファコンソールでは、コンソール上で日本語を扱うためのKON（漢字コンソール）が動作しないが、JFBTERM/MEを使うことで、1280×1024ドットなどの広いコンソール画面で日本語を表示できるようになる。なお、日本語の表示にはX用のPCFフォントを流用する。

●ビルドとインストール

JFBTERM/MEは、安定志向版の0.2.3と開発版の0.3.5が、それぞれファイル一式をtar＋gzipしたtarボール形式で配布されている。以下の説明では開発版を利用する。

ビルドは「./configre」→「make」→「make install」という一般的な手順だ。なお、設定ファイルを/etcに作成するには、「./configure --sysconfdir=/etc」とする（初期設定では/usr/local/etcに作成される）。

●フレームバッファとは

フレームバッファコンソールでは、文字をVGAテキスト画面に表示するかわりに、フレームバッファデバイス（/dev/fb0など）を経由してグラフィック画面に描画する。このため、1024×768ドットなどの高解像度でコンソールを使用できるだけでなく、フォントの形状を変化させたり、起動時のペンギン（**画面1**）などのようにグラフィックを表示させることも可能だ。

現在、フレームバッファデバイスやコンソールドライバを利用していない場合は、カーネルの再構築やデバイスファイル（/dev/fb0）の作成などの作業が必要だ。詳細な手順については、たとえば「Jun's Homepage」（http://www.nk.rim.or.jp/~jun/）の「Linux kernel2.2の新機能」の記述などを参照されたい。

●フレームバッファの設定

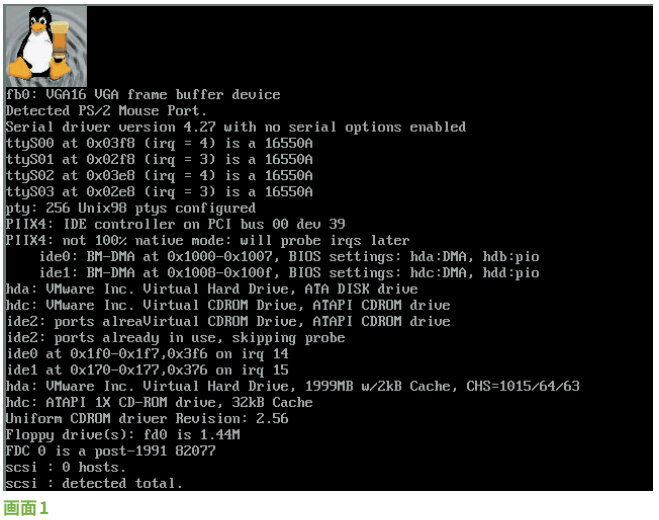
JFBTERM/MEは、現在のフレームバッファコンソールの画面モードをそのまま利用して表示を行う。このため、画面解像度の変更などは事前に行っておく必要がある。

ただし、汎用のVESAコンソールドライバ（vesafb）を利用する場合、画面解像度や色数の設定はLinuxの起動時にしか行えない。具体的には、rootになって/etc/lilo.confの設定に表1のようなエントリを追加し、/sbin/liloを実行して設定を有効にする。リブート後のliloのプロンプトで「linux-fb」と入力すると、1024×768ドットの画面に切り替わるはずだ。

これに対し、Matrox系のビデオカード（Matrox Millenium G200、G400など）で利用できるネイティブなコンソールドライバでは、起動後にfbsetコマンド（http://www.cs.kuleuven.ac.be/~geert/bin/）を使って画面解像度を自由に変更できる。

●JFBTERM/MEの初期設定

JFBTERM/MEは、表示に使用するフォントなどの設定を、/etc（あるいは/usr/local/etc）のjfbterm.confから読み込む。インストール時にはこのフ



画面1
フレームバッファコンソールでは起動時にCPUの数だけペンギンが表示される。

画面2
使用するフォントの設定は、jfbterm.confで行う。

```
image = /boot/vmlinuz
  label = linux-fb
  root = /dev/hdaN※1
  vga = 773※2
  append = "video=vesa:ypan"※3
```

リスト1　/etc/lilo.confに追加するエントリの例

```
if [ "$TERM" = "kon" ]; then
  PS1="\033[32;1m[\u@\h \W]\\$\033[37;0m "
else
  PS1="[\u@\h \W]\\$ "
fi
```

リスト2　.bashrcに追加するプロンプト設定用のブロックの例

※1 NはLinuxのルートパーティションの番号（1〜）。2台目のIDEドライブは「hdbN」、SCSIドライブは「sdN」。
※2 256色表示の場合、640×480は「769」、800×600は「771」、1024×768は「773」、1280×1024は「775」
※3 ypanはハードウェアスクロールに関するオプション設定

ァイルは作成されないため、サンプルファイルの記述を元に各人が記述する必要がある。

まず、cpコマンドなどを使って、サンプルファイル（jfbterm.conf.sample）の内容をそのままjfbterm.confにコピーし、Emacsなどのテキストエディタで内容を編集する（**画面2**）。とりあえず、表示に使用する英字・漢字フォントの設定部分だけは必ず確認しよう。というのも、それぞれのディストリビューションによって、これらのフォントの存在するパスが微妙に異なっているからだ。

たとえば、初期設定の英字フォントは、サンプルでは「/usr/X11R6/lib/fonts/misc/7x14.pcf.Z」となっているが、VineやLASER5、Turboでは「/usr/X11R6/lib/X11/fonts/misc/7x14.pcf.gz」だ。漢字フォントは、サンプルでは「/usr/X11R6/lib/fonts/misc/k14.pcf.Z」だが、LASER5とTurboでは「/usr/X11R6/lib/X11/fonts/misc/k14.pcf.gz」、Vineでは「/usr/X11R6/lib/X11/fonts/japanese/k14.pcf.gz」と修正する。このほかの、先頭が「#」で始まる行は、コメントなので無視して構わない。

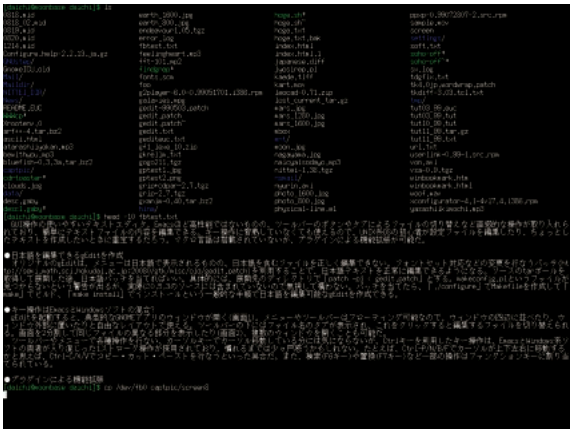
●JFBTERM/MEの起動

フレームバッファコンソール上で、「jfbterm」とすると、/etc/jfbterm.confの記述に基づいてフォントファイルなどを読み込み、JFBTERM/MEが起動する（**画面3**）。英字・漢字の表示はもちろん、カラーlsなどによるカラー表示もちゃんと反映される。

CPUの速度にもよるが、1024×768ドット程度の画面ならば、画面表示やスクロールのもたつきは感じられないはずだ。スクロールが遅い場合は、/etc/lilo.confでハードウェアスクロールを指定している部分（**リスト1の※3**）を確認してみよう。

正常に動くことが確認できたら、次は初期設定のフォントを別のフォントに変えてみるのも楽しい。たとえば、英字を通常のコンソールと同じVGAフォント（vga.pcf）にしてみたり、漢字にまる文字フォント（maru16.pcf.Z）を指定するといった具合だ（**画面4**）。なお、フォントの読み込みはJFBTERM/ME起動時に行われるので、こうした変更を有効にするにはいったん「exit」としてJFBTERM/MEを終了し、再度起動する必要がある。

ところで、VGAフォントを利用すると、画面からはJFBTERM/MEを実行中なのか判別できないことが多い。そこで、JFBTERM/MEでは環境変数TERMの値が「kon」（変更可能）であることを利用して、.bashrc中でプロンプトを変更するといい。たとえば、**リスト2**のブロックを.bashrcに含めると、コンソールやX上のktermではモノクロのプロンプト、JFBTERM/MEでは緑色のプロンプトが表示されるようになる。



画面3
1024×768ドットの画面でJFBTERM/MEを起動。

画面4
まる文字フォントなどを利用することも可能だ。

RPMやZIP／LHAも扱えるアーカイブマネージャ

TkZip

バージョン：1.0.15　　種別：as is

http://www.pcnet.com/~proteus/TkZip/TkZip.html

TkZipはTcl/Tkで書かれたアーカイブマネージャだ。tar＋gzipされたtarボールだけでなく、RPM形式のパッケージや、Windowsで使われるZIP／LHA／RAR形式のアーカイブも扱える。いずれの形式の場合も、アーカイブ内のファイル一覧がウィンドウに表示され、ファイルの展開やビューアによる閲覧といった処理を統一されたGUI操作で行える。

●インストール

TkZipは、ファイル一式をtar＋gzipしたtarボールで配布されている。Tcl/Tkはインタプリタ言語なので、コンパイルの必要はない。インストールはTkZip自身が行う。展開先のディレクトリで「./TkZip&」として起動すると、インストール用のウィンドウ（**画面1**）が開くので、そのまま[Install]ボタンを押せばいい。「Installation Complete」というウィンドウが開けばインストール成功だ。

なお、初期設定のままだと、ファイル名の一覧表示に使われるフォントサイズが小さすぎるので、以下のように修正するとよい。

展開先の（インストール後は/usr/local/binの）TkZipをテキストエディタで編集し、1842、7076、7096、7961行目のフォント設定の8番目のフィールドを「＊」から「140」（14ポイント）に変更する。たとえば、元が、「-*-fixed-bold-*-*-*-*-*-*-*-*-iso8859-1」なら「-*-fixed-bold-*-*-*-*-140-*-*-*-*-iso8859-1」とすればいい。

●ファイルの閲覧と展開

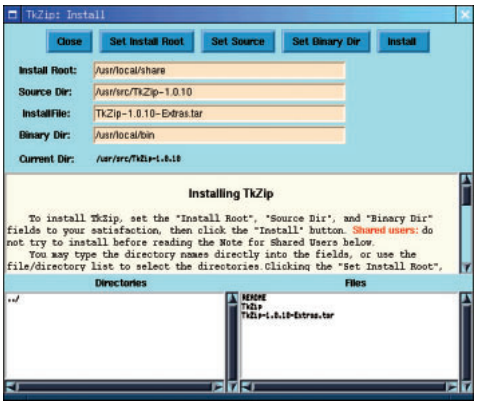
「TkZip&」として起動すると、カレントディレクトリのサブディレクトリとファイル一覧が表示されたウィンドウが開く（**画面2**）。ふだんは、[Options]－[Show Archives Only]をチェックして、アーカイブのみ一覧表示されるようにしておくとよいだろう。

一覧中のアーカイブをダブルクリックすると、アーカイブ内のファイル一覧が別ウィンドウに表示される（**画面3**）。一度に複数のウィンドウを開くことも可能だ。これらのウィンドウの配色（変更可能）は、アーカイブ形式ごとに異なっている。実際には、TkZip内部で各アーカイブ形式に対応するコマンドを実行している。たとえば、tarボールならtar／gzip、ZIPならzip／unzip、LHAならlhaといった具合だ。これらのコマンドの存在はTkZipが自動的に検索し、[File]－[Show System Info]で確認できる。

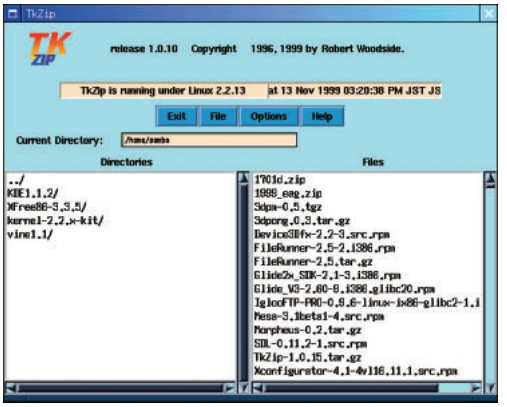
ファイルの閲覧や展開を行うには、まず対象とするファイル名をクリックで選択する。Ctrl-クリックによる複数選択や、Shift-クリック（あるいはドラッグ）による範囲選択も可能。全ファイルを素早く選択するには[Select All]ボタンを使用する。続いて、閲覧なら[View]ボタン、展開なら[Extract]ボタンを押せばいい。展開の際には、展開先のディレクトリを選択するダイアログが開くので（**画面4**）、展開先を選択（あるいは直接入力）して[Extract]ボタンを押す。

閲覧に使われるビューアは、[Select viewer]ボタンで変更できる。Emacsなどいくつかの候補の中から選択できるほか、ユーザーがビューアの実行ファイルを直接指定することも可能だ。[User defined]を選択し、ウィンドウ中の[Executable]に「kterm -e less」と入力する。これで、[View]ボタンでktermが起動し、lessを使ってファイルを閲覧できる（**画面5**）。

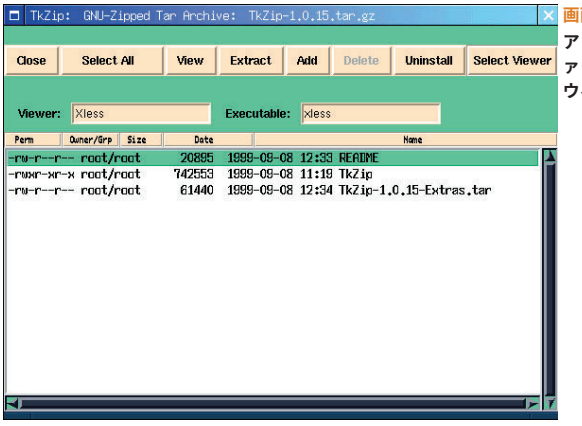
さらに、後述のMIMEタイプの設定によっては、ファイル名をダブルクリックするだけで、ファイル名の拡張子に応じたコマンドを起動できる。たとえば、画像ファイルをダブルクリック



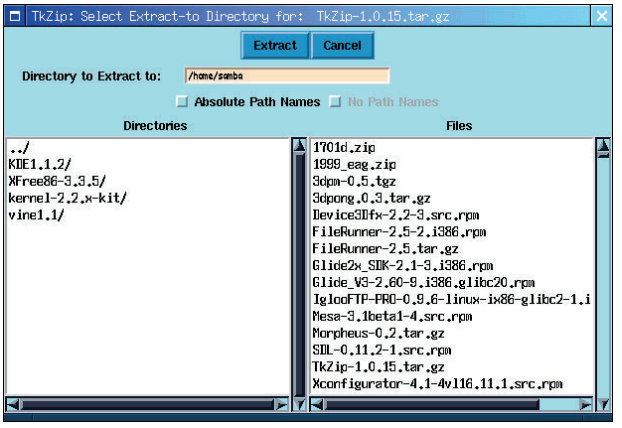
画面1
インストールはTkZip自身によって行われる。



画面2
メインウィンドウはディレクトリとファイル一覧の2パネル構成。



画面3
アーカイブ内のファイル一覧を別のウィンドウに表示。



画面4
ファイルを展開するディレクトリを選択する。

すると画像ビューア（xvなど）が起動するといったことが可能になる。

●アーカイブの作成

TkZipでは、アーカイブにファイルを追加したり、新たなアーカイブを作成することもできる。こうした操作を行う場合は、[Options]－[Show Archives Only]のチェックを外して、全ファイルが一覧表示されるようにしておくとよいだろう。

ファイルを追加するには、アーカイブ内のファイル一覧を表示しているウィンドウで[Add]ボタンを押す。展開時と同じようなウィンドウが開くので、追加するファイルを一覧から選択すればいい。ディレクトリごと追加することも可能だ。

一方、アーカイブを新規作成する場合は、メインウィンドウで[File]－[New Archive]を選択する。新規作成用のダイアログが開いたら、アーカイブ形式とアーカイブファイル名を設定して[Create]ボタンを押す。すると、そのアーカイブのウィンドウが、ファイルをひとつも含まない状態で開くので、あとは[Add]ボタンを使ってファイルを追加していけばいい。

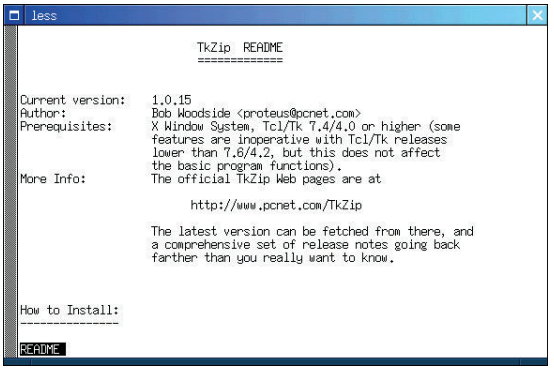
●柔軟なカスタマイズ機能

TkZipでは、[Options]以下のメニューにより、動作を細かくカスタマイズできる。

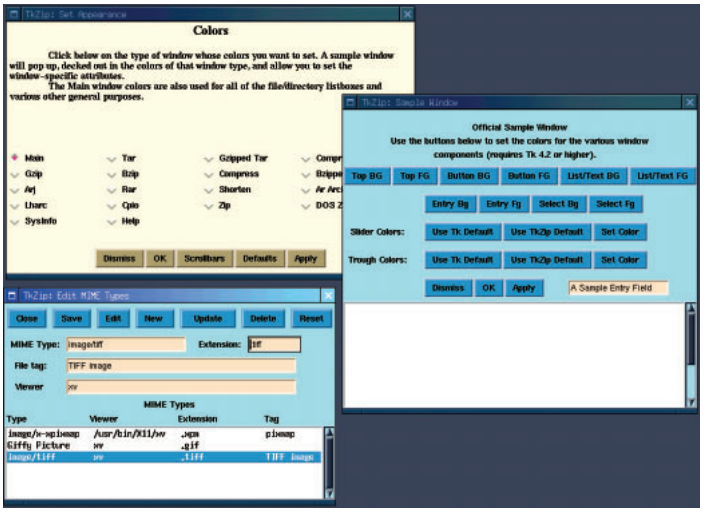
たとえば、初期設定ではファイルの上書き時に警告を発するが、これを無効にするには[Options]－[Warn of Overwrites]のチェックを外す。逆に、決して上書きしないようにするには[Options]－[Never Overwrite Existing Files]をチェックする。

画面表示に関しては、[Options]－[Set Appearance]でダイアログが開き、メインウィンドウやアーカイブ形式ごとに、ウィンドウの背景色などを細かく設定できる（**画面6上**）。こうした設定は、[Options]－[Save Options]で設定ファイルに保存される。

このほか、[Options]－[Edit MIME Types]のMIME設定ダイアログでは、ファイルの拡張子と実行するコマンドを関連付けられる。画像ファイルやHTMLファイルなどの設定を行うとよいだろう（**画面6下**）。



画面5
kterm上のlessを使ってアーカイブ内のファイルを閲覧中。



画面6
ウィンドウの背景色やMIME設定など細かなカスタマイズが可能。

Pythonで書かれたバリエーション豊かなカードゲーム

PySol

バージョン：3.00　　種別：GPL

http://wildsau.idv.uni-linz.ac.at/mfx/pysol.html

PySolは、PythonとTcl/Tkを使って書かれたバリエーション豊かなカードゲームだ。Windowsでおなじみの「クロンダイク」や「フリーセル」、「スパイダー」など、計111種類!?にも及ぶトランプの一人遊び（ソリティア）をプレイ可能だ。プラグイン形式で新たなゲームを追加できるほか、カードのデザインなどを変更するカードセットも提供されている。

●ビルドとインストール

PySolは、ファイル一式をtar＋gzipしたtarボールで配布されている。Pythonで記述されているためコンパイルの必要はない。展開後のディレクトリで「make install」とすると、/usr/local/binに実行ファイルが、/usr/local/share/pysol/3.00にデータファイルがそれぞれコピーされる。

さらに、3.00用のカードセットを導入すると、バリエーションに富んだデザインのカードを利用できる。こちらもtarボールで別途配布されているので、展開後のディレクトリで「cp -a data/cardset-*/usr/local/share/pysol/3.00」として、データファイルのディレクトリにコピーする。

なお、PySolの実行には、最新のPython（1.5.2以降）とTcl/Tk8.0以降が必要だ。Pythonのtarボールは「Python Language Website」（http://www.python.org/）から、RPMは「Python & Linux」（http://andrich.net/python/）からダウンロードできる。

●ゲーム中の操作

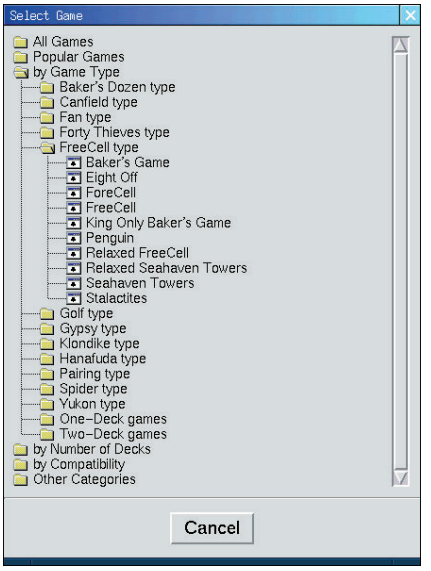
「pysol&」として起動すると、クロンダイク（**画面1**）が始まる（次回からは終了前にプレイしていたゲームになる）。ゲームの種類を変更するには、[File]－[Select game]以下のメニューから選ぶ。計111種類もあるため、「クロンダイクタイプ」のように、タイプ別にサブメニューが用意されている。最初は、[Popular games]以下にあるクロンダイク、フリーセル、スパイダーなど一般的なゲームのうち、すでにルールをよく知っているものをプレイするとよいだろう。このほか、全ゲームをさまざまな視点で分類したウィンドウから、プレイするゲームを選ぶことも可能だ（**画面2**）。

ゲーム中の操作にはマウスを使用する。左ボタンのクリックでタロン（置札）から新しいカードを引き、ドラッグでタブロー（一時的なカード置き場）に移動するというのが基本的な操作だ。このほか、中ボタンのクリックで積まれている途中の表向きのカードを確認できるし、ダブルクリックか右ボタンのクリックでファウンデーション（台札）へカードを移動できる。

操作の手間を減らすには、[Options]－[Automatic play]以下の項目を選択する。伏せてあるカードを表向きにする、カードを配る、ファウンデーションへカードを移動するという3種類の動作を自動化できるが、ゲームの種類によっては自動化しないほうが良いものもあるので気をつけよう。

ツールバーには、ゲームを最初からやり直したり（[Restart]ボタン）、現

画面1
クロンダイクをはじめ、111種類ものゲームをプレイできる。



画面2
タイプやデッキ数などで分類したウィンドウからゲームを選択。

画面3
見事にクリアすると、このCongraturation画面が表示される。

画面4
よく知らないゲームは、まずルールの確認から始めよう。

在の状態をファイルに保存して後で読み込んだり（[Save]、[Open]ボタン）、新しい配置でゲームをやり直す（[New]ボタン）ボタンが用意されている。クリア時のCongratuation画面（**画面3**）を目指してがんばろう。

●よく知らないゲームをプレイする

ルールがよくわからない場合は、ツールバーの[Rules]ボタンを押してルールの解説を表示する（**画面4**）。説明は簡潔な英語で書かれているので、読むのはそれほど難しくない。なお、一般的なゲームのバリエーションの場合は、基本となるルールとの相違点だけが記述されている。

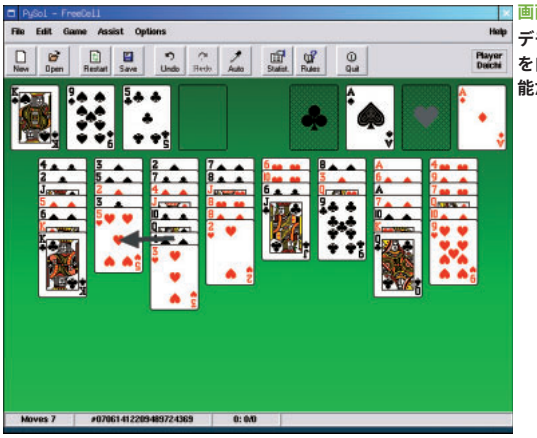
ある程度ルールを把握したら、試しにプレイするのが理解への早道だ。初めのうちは、操作を遡ってやり直せるアンドゥ機能（ツールバーの[Undo]ボタンかZキー）、動かせるカードを教えてくれるヒント機能（[Assist]－[Hint]かHキー）、コンピュータがプレイするデモ機能（[Assist]－[Demo]かCtrl-Dキー）などの機能を使うとよいだろう（**画面5**）。ただし、ヒントやデモで示される手は、単にカードを動かせるというだけで最善手ではない。特に、難易度の高いゲームでは手詰まりになりやすいので注意されたい。

●盤面の表示を変更する

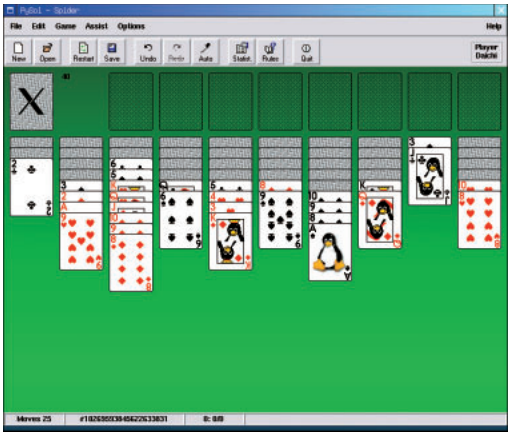
表示に関しては、[Options]以下の項目により、カードの裏面のデザインやテーブルの色使いを変更できる。たとえば、裏面のデザインを変更するには、[Options]－[Card background]で一覧表示される中から選択すればいい。[Options]－[Save options]を選択すると、現在の設定内容が設定ファイルに保存される。

さらに、「カードセット」を切り替えることで、カードの大きさや表面・裏面のデザインをまとめて変更することができる。たとえば、画面の狭いノートパソコンなどでは、[Options]－[Cardset]－[Standard - Small]を選択して、一回り小さなカードを使うと操作性がアップするだろう。なお、選択したカードセットが有効になるのは次のゲームからだ。

PySolに付属するカードセットは、初期設定の「Standard」と「Standard -small」の2種類だけだ。これではさびしいという人は、別途配布されているカードセットを「ビルドとインストール」での説明通りにインストールしよう。さらにカードの小さな「Standard - Tiny」や、絵札がペンギンの「Ace of Penguins」（**画面6**）など、さまざまなデザインの17種類のカードセットが追加される。なお、カードセットの存在はPySolの起動時にチェックされるため、PySol実行中にカードセットを追加した場合は、一度PySolを再起動する必要がある。



画面5
デモ機能を利用してゲームを自動的に進めることも可能だ。



画面6
カードセット「Ace of Penguins」を使ってスパイダーをプレイ。

Sambaの共有ディレクトリをGUIでマウント

LinNeighborhood

バージョン：0.4.1　　種別：GPL

http://www.bnro.de/~schmidjo/

LinNeighborhoodは、Sambaで共有しているWindowsマシンや他のLinuxマシンのディレクトリを階層的に表示し、マウント、アンマウント操作をGUIで行えるようにするソフトだ。マウント後は指定したファイルマネージャが起動して、ローカルなディレクトリと同様にファイル操作を行える。ビルドにはGTK+1.2以降が必要だ。このほか、smbclientなどいくつかのツールを実行時に使用する。

●ビルドから起動まで

LinNeighborhoodは、ファイル一式をtar+gzipしたtarボールでソースとバイナリのパッケージが用意されているほか、RPM形式のパッケージも別サイト（http://milkyway.thn.htu.se/~ds98rito/）で配布されている。RedHat系LinuxではRPM形式のバイナリパッケージを利用したほうが簡単だ。

なお、マウントに利用するsmbmountとsmbumountは、Linuxカーネル2.0ではsmbfsパッケージに付属し、カーネル2.2以降ではSamba 2.0に付属する。ただし、Samba 2.0ではこれらはオプショナルなので、付属しないディストリビューションもある。

●共有ディレクトリをマウント

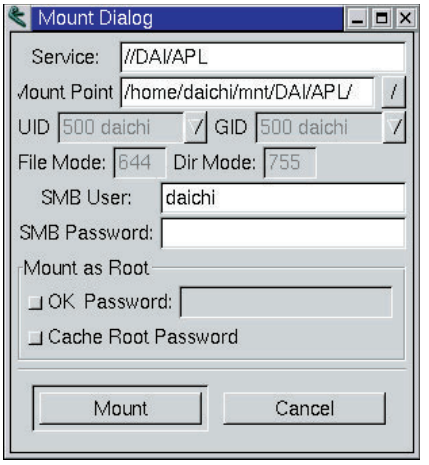
「LinNeighborhood&」として起動すると、ウィンドウが開いて、マシンやディレクトリのツリー表示と、実際にマウントされたディレクトリのリストが表示される（**画面1**）。

初回起動時には、いくつか初期設定を行う必要がある。ツールバーの[Preferences]ボタンを押して設定ダイアログ（**画面2**）を開き、ワークグループ名を設定する。[Save]ボタンで設定ファイル（~/.linneighborhoodrc）に保存しておこう。

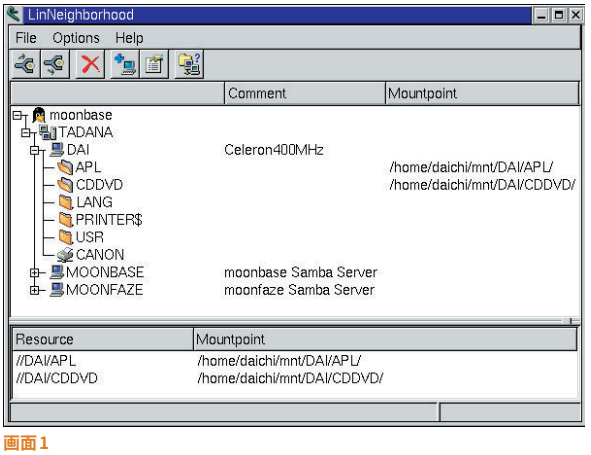
ウィンドウ上のホスト名を右クリックし、[quick browse]を選択すると、ワークグループ名の下にLinuxマシンやWindowsマシンの一覧がツリー表示されるはずだ。これらのマシン名をダブルクリックすると、共有ディレクトリの一覧が表示される。

共有ディレクトリをダブルクリックすると、マウント設定用のダイアログが開く（**画面3**）。ここでは、マウントポイントやユーザー名の変更を行えるが、通常はそのまま[Mount]ボタンを押せばいい。

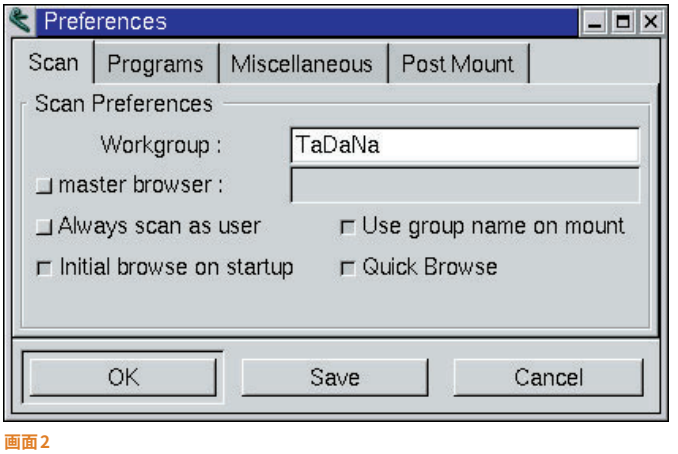
マウントに成功すると、xtermのウィンドウが開いて、ファイルマネージャ（mc）の画面が表示される。ただし、国産ディストリビューションでは、mcが日本語化されているため、xtermでは漢字表示が文字化けしてしまう。設定ダイアログの[Post Mount]ページでファイルマネージャの設定を変更しよう。mcを使うならxtermをrxvtに変えるのがお勧めだ。



画面3
マウント設定用ダイアログでは、ユーザー名の変更も可能だ。



画面1
Sambaで共有しているマシンとディレクトリの一覧が表示される。



画面2
設定ダイアログでワークグループやファイルマネージャを設定。

雪山が舞台の3Dレースゲーム

XRACER

バージョン：0.94　　種別：GPL

http://xracer.annexia.org/

XRACERは、雪山に設置されたコースを浮遊式クラフトで駆け巡る3Dレースゲームだ。コースやクラフトのデザインはシンプルだが、低い視線がもたらすスピード感はなかなかのもの。自動操縦や加速といったパワーアップアイテムも用意されている。なお、OpenGL互換ライブラリのMesa 3.0を利用して3D表示を行うため、実用的な速度でプレイするには、3Dアクセラレータ（Voodoo系など）が必須だ。

●ビルドから起動まで

XRACERは、ファイル一式をtar＋gzipしたtarボールでソースとデータのパッケージが用意されている。ビルド手順は「./configure」→「make」→「make install」という一般的なものだ。なお、GNOMEのサウンドドライバEsounD（esd）を利用していない環境では、「./configure --disable-esd」とする必要がある。

一方、データのほうは、適当なディレクトリ（/usr/local/libなど）に展開し、「export XRACER_HOME=/usr/local/lib/xracer」として環境変数XRACER_HOMEにディレクトリを設定する。常用する場合は.bash_profileなどに書いておくといい。

ライブラリ関係では、Mesa 3.0本体とGLUTが必要だ。これらのライブラリは、手持ちの3Dアクセラレータを利用する状態でビルドされている必要がある。

詳細は「The Mesa 3D Graphics Library」（http://www.mesa3d.org/）などを参照されたい。

画面1
激しい吹雪の中のレース。ブルーシグナルでスタートだ。

●コースの先を読め!

「xracer&」として起動すると、メニュー画面が表示され、Sキーでプレイ開始だ（**画面1**）。初期設定の画面サイズ（640×480ドット）は、-sオプションで変更できる。たとえば、1024×768ドットでプレイするには、「xracer -s 1024x768」とすればいい。

操作にはキーボード・マウス・ジョイステックを利用できる。細かなハンドル操作を行うにはマウスかジョイステックが必須だ。マウスを使うには、プレイ中の画面でDキーを一度押せばいい。左右の動きがハンドル操作、左ボタンがアクセル、右ボタンがブレーキに割り当てられている。

降りしきる雪の中を猛スピードで走り抜けるには、コースを熟知することはもちろん、コース内にいくつか設けられている加速ゾーンやアイテムの使い方も重要だ。所得したアイテムの種類は画面上部に表示され（**画面2**）、スペースキーで使用できる。

レースは同じコースを3周するまで続けられ、ゴール時には順位や合計タイム、ラップタイムなどが表示される（**画面3**）。今後はネットワークへの対応なども行われる予定だ。

画面2
アイテムを取得すると画面上部に表示される（これはシールド）。

画面3
コースを3周するとゴール。ラップタイムなどが表示される。

隠喩としてのコンピュータ

Emacsにストールマンの啓蒙思想を読む

文：豊福 剛
Text：Tsuyoshi Toyofuku

ストールマンが、フリーソフトウェアの思想家であると同時に、何よりもまず偉大なプログラマーである以上、プログラマーとしての思考とソフトウェアに対する思想が、まったく無関係であるはずはない。ストールマンの思想は、GPLやGNU関連のマニフェストで語られているだけでなく、ソフトウェアそのものにも刻印されているはずだ。そういう意味では、GNUプロジェクト以前にストールマンが手掛けた、Emacsというソフトウェアの設計思想の中に、その後のGNUプロジェクトで先鋭化する彼の思想の原型を読み取ることができるのではないだろうか。

GNU Emacsに関するマニュアルや書籍の多くは、GNU Emacsが持つ膨大な機能の解説や使い方の説明について書いてあるが、その設計思想についてまで言及されたものは少ない。しかし幸いにして、ストールマンがEmacsの設計思想をまとめた論文がある（*EMACS: The Extensible, Customizable Display Editor*； http://www.gnu.org/software/emacs/emacs-paper.html）。これはGNU Emacsではなく、'75年前後に作られた最初のEmacsについての論文なのだが、Emacsが作られた歴史的背景やそれが及ぼした影響などにも触れてあり、興味深い内容になっている。

はじめにTECOがあった

GNUプロジェクトが開始されるちょうど10年前の1974年、ストールマンは最初のEmacsを開発する。UNIXが本格的に普及するのは'80年代になってからで、この'74年当時はまだUNIXがいくつかのコンピュータに移植されはじめた時期にすぎない。このころ、MITのAIラボではITS（Incompatible Time-sharing System）という風変わりなOSが使われていた。現在最も代表的なEmacsであるGNU Emacsの開発が始められたのは'84年になってからで、UNIXにEmacsが普及していったのは、さらにこのあとのことになる。

UNIXで使えるEmacsには、さまざまな実装があり、Javaの開発者として有名なジェームズ・ゴスリングが手掛けたGosling Emacs（これはC言語で実装された最初のEmacsである）がGNU Emacsと人気を二分した時代もあった。また、UNIXが普及する以前の'70年代においても、後述するように多種多様なEmacsのバリエーションが開発されたことからわかるように、Emacsは多くのプログラマーに影響を与えた革新的なソフトウェアだったのだ。

Emacsという名前の由来については、*Editing Macros*の略というのが定説のようだが、ストールマンが好きなアイスクリーム屋の名前という説もあるらしい。Editing Macrosと

は、「編集作業のためのマクロ集」くらいの意味なのだろう。最初のEmacsは、ITS上で動く“TECO”という非常に変わった行エディタが提供する独自なプログラミング言語（マクロ）で実装されたのである。

このTECOは、*Text Editor and COrrector*の略なのだが、そもそもは紙テープを編集するための行エディタであったことから*Tape Editor and COrrector*と呼ばれていたらしい。これがどのようなエディタ／プログラミング言語であったのか気になるところだが、Sun SiteにTECOのアーカイブ（http://sunsite.sut.ac.jp/pub/academic/computer-science/history/pdp-11/teco/）があって、TECOのエミュレータやマニュアル、TECOのマクロで書かれたプログラムなどが収集されている。これを見た限りでは、sendmail.cf並みの難解さで、これに比べると悪名高いsedのスクリプトさえ読みやすく感じてしまうくらいだ。

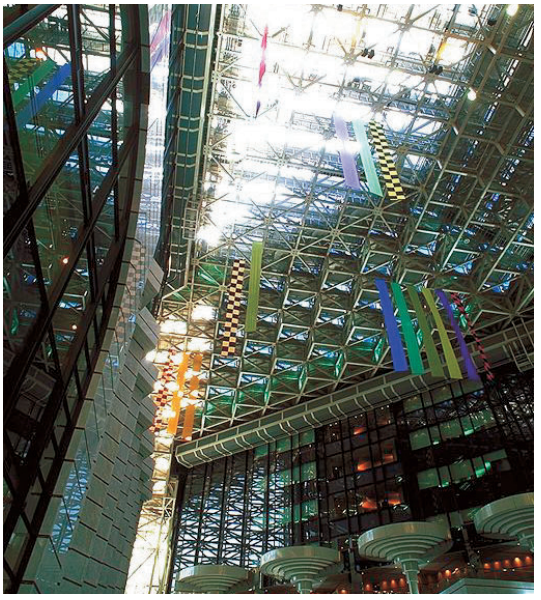
ストールマンがEmacsを書くきっかけになったのは、スタンフォード大学の人工知能研究所が開発した“E”という行エディタに触発されてのことだったらしい。編集に使う言語と同じものを使ってマクロが書けるEは、TSS（Time Sharing System）の一部であったため、Eが提供する基本コマンドそのものを再定義できないという制約はあったものの、非常に強力なプログラミング言語として機能したようだ。

ストールマンの論文には、システムとしてのEmacsの基本構成が解説してあり、具体的には、

・プリミティブ（テキスト操作やI/Oのための）
・インタープリタ
・コマンドディスパッチャ
・ライブラリシステム
・ディスプレイプロセッサ

の5つの構成要素が列挙されている。興味深いことに、これらの構成要素は初めからすべて実現されたわけではない。プリミティブについてはTECOのシステムを前提にしたうえで、TECOのコマンドを実行するコマンドディスパッチャと、画面表示を実行するディスプレイプロセッサが、まず実現された。

このようにTECOを拡張する試みは、Emacs以外にもあり、代表的なものとしては、TECMACやTMACSがあった。TECMACは、TECO上に実装された最初のスクリーンエディタで、そのユーザーインターフェイスにはEmacsも大きな影響を受けている。また、TMACSでは、コマンドによってユーザーのキー入力を共有ライブラリに関連づけるというア



イデアが最初に実装されたエディタだった。これらのエディタのアイデアに影響を受けて、EmacsでもTECOで書かれた関数を呼び出すことができるように、コマンドを再定義できる仕掛けが組み込まれた。

TECOからLispへのシフト

Emacsでは、その後ライブラリシステム（ライブラリとは、関数名、関数定義、ドキュメンテーションで構成されるプログラムファイルのことを指し、Emacsのセッション中であってもEmacsにロードできる）とセルフドキュメンテーション（コマンドの動作内容を問い合わせると、その説明文を表示する仕組み）を実現する機能が組み込まれる。これによって、より読みやすいプログラミングスタイルが実現された。これらの拡張は、TECOで書かれたコードの追加だけでなく、TECOそのものに新しい機能を追加することによって実現された。その一方で、TECOをベースとしたディスプレイプロセッサのメンテナンスは、より困難な状況に陥ってしまった。TECOで書かれたプログラムは、あまりに難解すぎて、新しいユーザーから敬遠されたのだ。

このような状況を解決するために、'79年前後からプログラミング言語としてTECOの代わりにLispが使われるようになった。たとえば、HoneywellのBernard S. Greenbergが書いた「Multics Emacs」はMacLispで書かれており、TECOで書かれたEmacsよりも強力だった。ほかにも、MITのAIラボで開発されたLispマシンのためのエディタ「ZWEI」があった。ZWEIは*ZWEI Was EINE Initially*（ZWEIの最初の名前はEINEだった）の略で、そのEINEは*EINE Is Not Emacs*（EINEはEmacsではない）の略というから、*GNU is Not UNIX*という命名のスタイルのルーツは、当時のAIラボ界隈のハッカー文化にあったのだろう。

プログラミング言語としてのLispは、現在のGNU EmacsにおいてもEmacs Lispという形で実現されていて、これがGNU Emacsの基盤となっている。Emacsの基盤としてLispが採用されたのは、プログラムの読みやすさのためだけでなく、コンパイルすることもできるLispコードを利用することによって、アセンブラのコードを減らすことができるといった効率上の理由も大きかった。ただし、Lisp以前のTECOにおいても、すでにユーザーがプログラミングすることによってシステムを拡張できる構成になっていたことは、注目すべきだろう。とりわけ、TECOもLispもインタープリタ言語であり、ユーザーによる試行錯誤的なプログラミングが対話的に行えたことと密接に関連しているといえるだろう。

Emacsの基本思想は拡張性にある

Emacsのユーザーインターフェイスや機能を模倣したエディタであっても、このような拡張性を持たないものは*Ersatz Emacs*（偽のEmacs）であると、ストールマンは論文に書いている。Emacsの設計思想の根幹は、「拡張性の実現」にあるのだ。それは、キー入力とコマンド実行の対応づけがユーザーによってカスタマイズできるというだけでなく、機能の追加や変更をユーザーが実現できるという点にある。ユーザーは、実装者によるEmacsの設計上の意思決定に制限されない。実装するに値しないと判断された機能であっても、ユーザーが必要とすることはありえるし、またその逆もありえる。システムの実装者は専制君主ないし独裁者であるべきではなく、システムの拡張や改変に対してユーザーは自由であるべきだ、というストールマンの考えがEmacsには反映されているのだ。

さらに、ユーザーによるシステムの拡張は、個人的に実現されるだけではなく、それを公開し、共有できるための仕組みがある点も重要だ。新しい関数や再定義された関数は、ライブラリにまとめることができる。そのため、ユーザーによる拡張が、互いに干渉しあって不整合を生じることが避けられる。新しいアイデアを自由に試してみて、それがポピュラーになれば、システムのコア部分に組み込まれることもある。Emacsの多国語対応版であるMuleがGNU Emacsそのものに組み込まれたことなどは、その代表的な事例といえるだろう。

もちろん、ユーザーがEmacsを拡張するためには、そのプログラミング言語であるEmacs Lispのリテラシーが必要になる。このことに関連して、Emacsのようなプログラマブル・エディタは、初心者がプログラミングを学習するための優れたきっかけになるだろう、とストールマンは主張している。プログラマブル・エディタは、作成したプログラムの効果や結果を即座に見ることができるという利点がある。また、エディタは日常的に使うソフトウェアだけに、ちょっとした編集作業をプログラムとして実現したいというニーズは頻繁に発生する。ユーザーがそれを実現するための方法を学習したいと思ったときに、それがスムーズに実現できるシステムであることの意味は大きい。

Emacsにはストールマンの啓蒙の思想が刻印されているのだ。

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

されどマイクロソフトは永遠に

文：安田幸弘
Text : Yukihiro Yasuda

コンピュータの世界では無敵のマイクロソフトも、アメリカの反トラスト法に対抗するのは難しかったようだ。アメリカの裁判所がマイクロソフト社は独占だという判定を下したというニュースを聞いて、「オッ」と思った人は少なくないと思う。

マイクロソフトが勝とうが負けようが、ぼくはあまり関心がないのだが、コンピュータの世界では決して小さなニュースではなく、この世界で暮らしているぼくとしては、さすがにまったく無関心というわけにもいかない。とはいえ、「オッ」と思ったのは一瞬、よくよく考えてみると「結局、それでどうなるんだろう？」という気がする今日この頃なのである。

唯一超大国のソフトウェア帝国

この判定に対しては、いろいろな立場からいろいろな意見があるようで、アメリカのアンチ・マイクロソフト陣営や消費者団体、そしてマイクソフトと競合しているソフトウェアメーカーの多くが、この決定を歓迎しているという。このへんの話は、ここで繰り返すまでもなくご存じのことと思うが、日本のマスコミがあまり伝えない反応のひとつに、これまでマイクロソフト—というより、マイクロソフトを筆頭とするアメリカの多国籍企業—を批判してきた各国の市民グループが、この判定を歓迎しているわけではない、ということを付け加えておこう。もちろん、今回の判定が、最終的な判決ではなく、単なる事実認定の段階のものでしかないということもある。だが、反応がシラッとしている主な理由は、「マイクロソフトをバッシングしたところで、コンピュータやインターネットの世界の『米国支配』の現実がどうなるものでもないでしょ」ということ。

そもそもアメリカという国は、自国の都合でいろいろとエゲツないことをやる国なのだ。アメリカに限らず、国家なんてものはどこでもそうなのかもしれないが、他の国と違って、アメリカは強大無比な超大国だけに、アメリカのエゲツない行動で迷惑する人々は少なくない。いつぞやの湾岸戦争や先日のコソボのように、自国の利益に反するような事件が起きれば、「正義」を掲げてミサイルをばんばん打ちまくるのに、東チモールでは「アメリカの国益に関係ない」とかなんとか、ムニャムニャ言っているうちに、何千人もの人々が死んでしまった。

だいたい、こんな国のやることなので、ぼくとしては来年の判決には期待していない。正義の味方みたいな反トラスト法だって、アメリカ企業の競争力の話で、外国企業との競争力を維持するためのもの。アメリカの正義がどの程度のものなのかは、来年に予定されているという判決が出た後で議論すべきことなのだろうが、単にマイクロソフトの影響力が小さくなることを期待している人々はともかく、一介の利用者の立場では、どんな判決が出てもたいして影響はないように思う。

たとえば、ニュースではマイクロソフトの解体という可能性もあるという。しかし、これはかえって恐ろしいことかもしれないと思う。つまり、もし、マイクロソフトが解体されたとしても、孫悟空の分身の術みたいに、「ベビー・マイクロソフト」がわらわらと誕生することになるかもしれない。マイクロソフトなんて、ひとつで十分、これ以上、増える必要はないと思う人はぼくだけじゃないだろう。

オープンソース・アクティビズム

フリーでなければ意味がない

さて、それではウワサされているもうひとつの可能性、ソースコードの公開はどうだろう。もしWindowsのソースコードが文字通り「公開」され、いわゆる「オープンソース」になったとしたら、これは画期的であるに違いない。ほとんど革命というべきで、マイクロソフトのダメージは大きい。

とはいえ、いくらオープンソースがブームだといっても、アメリカにとってマイクロソフトは「金の卵」を生む鶏だ。寓話にある通り、金の卵を生む鶏の腹を裂いてソースコードを取り出しても、鶏が死んでしまっては元も子もないというもの。鶏を殺すことなく、金の卵を生む鶏を増やすのが反トラスト法の狙いだとすれば、まずWindowsを文字通りのオープンソースにする可能性はない。

ソースコードの公開という判決になったとして、ありそうなのは、UNIXのソースコードライセンスのように、ソースコードへのアクセスに、いろいろな縛りがかけられる可能性だ。たとえば、ソースコードにアクセスするには、莫大なお金を払ってソースコードライセンスを取得し、ソースコードを外部に漏らさないといった秘密保持契約を結ばなければならないかもしれない。ソースコードライセンスのような契約は、ソースコードにアクセスできる既存のソフトウェアメーカーと、ソースコードライセンスを受けられない弱小メーカーとの開発力の格差を拡大するだけの結果しかもたらさないという可能性さえあるわけだ。

現在のオープンソースのムーブメントが、世界中、誰もが自由に、しかも無料でソースコードにアクセスすることができ、改変したコードを再配布できることにその本質があるわけで、ソースコードライセンスのような形態での「公開」では、われわれのようなシモジモのエンジニアには、何の意味もないのは明白である。

虚しきマイクロソフト・バッシング

最近、ぼくはマイクロソフトはマイクロソフト、巨大なソフトウェアメーカーでいいんじゃないかと思い始めている。Windowsというプラットホームはマイクロソフトのものであって、本来、マイクロソフトが好きな人やWindowsに満足している人がマイクロソフトにお金を払うことに文句を言う筋合いじゃないのだろう。

もし、マイクロソフトの独占が問題だとしても、マイクロソフトを選んだのはユーザー自身なのである。他に選択肢がなかった数年前ならともかく、今はLinuxをはじめ、いくつかの選択肢がある。もし自分たちの問題として、マイクロソフトの独占が問題だと思うのなら、単にマイクロソフトをバッシングするだけでなく、他の選択肢を発展させることを考えるべきだろうし、今ではそれも決して不可能ではないのだから。

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

今月はいろいろと話題があったが、大きかったのは、インテルのcopper mineの発表でしょうか。この前の820チップセットの延期では、あちこちのパソコン雑誌が大変だったけど、今回は、予定通り、最高速を奪取するような733MHz版のPentimu IIIを発表。なんでも、今年最後のCPUの発表で、なおかつ、かつてない15製品の同時発表なのだとか。やはり、インテルにも「焦り」ってのがあるんでしょうかね。Appleユーザーはどう思っているのかね。

Appleは、新製品であるG4をCPUの供給不足を理由に、出荷前にスペックを変更した。ちょっとした変更ならまだしも、CPUクロックを下げるという変更でありながら、なんと価格はクロック変更前のものを維持するという前代未聞のスペック変更であった。はっきり言って、これには非Appleユーザーも驚いた。いままでそんな商売したところを見たことがなかったからである。さすがに、Appleもクレームの多さに辟易したのか、結局、予約ユーザーには、予約時のクロックと価格で販売（ただし、500MHz版はなくなったために450MHz版の旧価格）することにした。

Appleは、iMacの類似製品を訴訟しまくっているようだが、新しいOSを「OS9」と呼んで、同名のOSを古くから販売しているMicrowareから訴えられているし、ColorSyncについても、特許侵害の訴訟を起こされている。価格改定の件では、「優れた企業は失敗を改める」とコメントしたそうだが、こんどはどう「改める」のだろうか？

アイボ再び販売

インターネット販売で、あっというまに予定数を売り尽くしたソニーのロボットAIBOだが、再度販売が行われた。前回の人気に気をよくしたのか、今回は、ヨーロッパ向けにも販売するのだとか。

個人的には、イヌとかの動物が好きではないので、欲しくもない（高くて買えない？）のだが、あの人気ぶりは、なんだか、ブランド物、高級カバンなんかと同じ構造を感じてしまう。知らなかったのだが、意外にイヌとか高いものなのですね。ペットとして見れば、超高級ってわけでもないのでしょうか。

イヌに餌をやらず、病気になると、ペットショップに「不良品だから交換しろ」という客もいる昨今、本物のイヌが死なないだけましかもしれない。

Microsoftがゲームマシン市場へ？

おそらくネタ元は、ウォールストリートジャーナルだと思うが、Microsoftがいわゆるゲームマシン開発に乗り出しているという記事がいくつか出ている。プレイステーション2を脅威と感じたとか、ゲームソフトがどうしてもゲームマシンのソフトを越えられないなどいくつか理由はあろうけれど、やっぱり、例のGatesの負けん気が出たような気がする。AMDのCPUを使ったAT互換機をベースにしたX-BOXというコードネームまで登場しているが、なんとなく本当そうな話。たぶん、今、OEMメーカーを探していたり、交渉中という感じなのでしょうか。やるとすると、Microsoftのことなので、クリスマスまでになんか発表して、対抗メーカーの勢いをそぐ作戦に出そうな気がする（とはいえ、OEMメーカーがなければ話にならないが）。

ただ、ゲーム専用機は子供が大きな主導権を握っているが、パソコンとな

ると、どちらかというと親が主導権を握っており、ちょっと購買層などが違うので、従来通りのやり方ではうまくいかない気がする。

それに、すでにプレステやNintendo64、ドリームキャストなんかで商売しているソフトハウスが乗るかどうかですね。Microsoftにロイヤリティが必要ないPCのソフトと同じライセンスだと、移植物ぐらいはやるかもしれませんが……。

10万円以下のパソコンが多数登場

国内では、次々と10万円以下のパソコンが登場しており、中にはゲートウェイのようにプロバイダに3年契約すると5万円引きってところも出てきた。低価格PCでは、ソーテックのe-ONEが一歩リードした感じだったけど、各社ともに10万円を切って、年末商戦を戦うようだが、台湾地震の影響がここにきて効いているのだとか。メーカーにより影響が違うようで、悲喜こもごもといった感じで、すでに年末をあきらめざるを得ないところもあるとか。

パソコンの価格が安くなると、各部品のコストが問題になるが、その中でも大きな割合を占めるのは、いまやハードディスクかWindowsのライセンスかと言われる状態になっている。ハードディスクは、ディスクメーカーが、低価格マシン対応を整えつつあり、下がる兆しが見えているものの、Windowsのライセンスは、ちっともそういう気配がなく、各社苦慮しているらしい。特に米国では、200〜300ドルといったレンジに入っており、Windowsのライセンスが占める割合はかなり大きいらしい。

Linuxでも、Netscapeなどがもう少しラフに扱っても落ちなければ、いい線いくと思うんですけど（Windows98でもNetscapeは落ちる時は落ちるんだから、もう少しなんだけど）。どうせ、インターネット使いたいってユーザーは、ブラウザとメールぐらいしか使わないでしょ。パソコン安くしたいメーカーがどーんと投資しないかしらね。

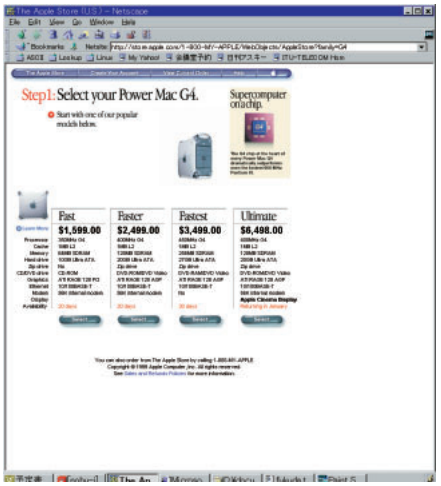
Microsoftの裁判

あんまり長いんで、マスコミがみんなだれちゃったMicrosoft対司法省の裁判だが、裁判の結果に影響する「事実認定」が発表され、トップニュースに返り咲いた。これによると、「Microsoftは、OS市場で独占力を持っている」と裁判所は認定し、これを元に判決を出すことになる。まだ、どういう判決が出るのかは決まってもいないし、Microsoftと司法省の和解というシナリオもありえる状態。さらにいえば、この裁判は、まだ「ワシントン連邦地方裁判所」での裁判であり、判決が出たとしても控訴があり得るわけで、Microsoftに対する処分などが確定したわけでもない。

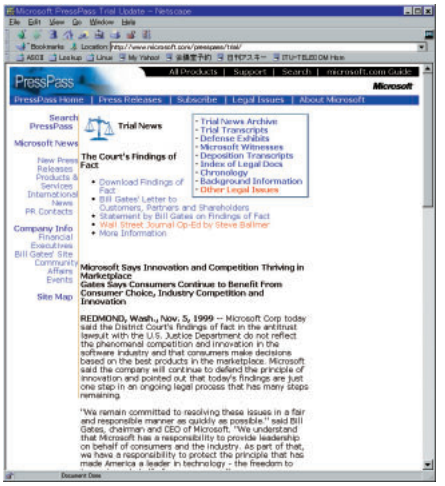
しかし、日本でも報道された通り、「Microsoftに不利な判決が出る」という観測が大勢を占め、雰囲気的には、Microsoftにとっていい方向にはない。このため、Microsoftも、今後は、PCメーカーに対して、影響力を行使しにくい状況になってきた。つまり、何かを強制しようとすれば、それが新たな訴訟や、現在の裁判にいい影響を与えないことになるからだ。

当面、Microsoftもあまり強引なビジネスはしにくい状態だし、他のOSやアプリケーションに対する攻撃的な活動も控えめになるかもしれない。

とはいっても、これで、他のOSがシェアを拡大できるかどうかは、やはりOSの出来にかかっているわけで、タナボタ式に、シェアが上がるわけでもないだろう。やっぱり、シェアを上げるには地道な努力が必要というわけだ。もっとも、千載一遇のチャンスではあると思うが。



AppleのMac G4直販ページ
450MHzが最速
http://store.apple.com/1-800-MY-APPLE/WebObjects/AppleStore?family=G4



Microsoftホームページの裁判ニュース
http://www.microsoft.com/presspass/trial/

正しきタコへの道

初級Linuxer養成講座

第4回　LANのないLinuxなんて……(1)

「LinuxはSOHOサーバに最適」という文句をよく耳にする。事実、Linuxは、ファイルサーバ、Webサーバ、データベースサーバ、プリントサーバなど、相当数のクライアントの面倒をみるのに十分な機能と性能を持っている。しかし、それほど大げさに考えなくても、LinuxをLANに接続することのメリットは大きいし、たとえ2台のパソコンしかない場合でも、それらをLANで接続するかしないかによって、使い道は大きく変わってくるのだ。面倒なことは抜きに、ともかくLANを使ってみよう。

文：竹田善太郎
Text : Zentaro Takeda

前回と同じような「つかみ」になってしまって申し訳ないのだが、Linuxを使いたい動機の多くに「ネットワークのサーバとして利用したい」という場合があるらしい。実際、ファイルサーバやプリントサーバとしてSambaを使ったり、WebサーバとしてApacheを使う話題は、Linuxを主題とした雑誌や書籍では**お約束**として、かならず取り上げられている。かくいう筆者も、この話題についてはいくつかの雑誌で記事を書かせてもらった。

一方で、「パソコン」と「ネットワーク」という2つの言葉を聞くと、現在ではすぐに「インターネット」が連想されると思う。個人でパソコンを使う場合、ほとんどはモデムによるダイヤルアップ接続でインターネットを使っているのだろうが、LANカードを使ってパソコン同士を接続しているユーザーは、思いのほか少ないのではないだろうか？ 会社でならともかく、**家でまでLANを引く意味はない**と考えるユーザーが多いのかもしれない。実際、実用的な意味で家庭内でのプリンタ共有やファイル共有が必要になるようなユーザーはそれほど多くないだろう。

しかし、近い将来、家庭電化製品やAV機器のネットワーク接続が一般的に可能になったり、デジタルTVが普及するようになれば、家庭内になんらかの形のLANを敷設することは当たり前のことになるだろう。もちろん、その際に使われるネットワークが現在の10Mbpsあるいは100MbpsのEthernetであるかどうかはわからないが、その上を流れる情報は、現在のTCP/IPをベースにしたものになることは、ほぼ間違いないだろう。そのような時代になってあわてることのないように、いまから家庭内LANを使ってみるのも悪くないのだ。LANの構築に慣れていれば、デジタル家電の時代になって、高額なLAN敷設料金をヤクザな業者に払う必要もないだろうし、手持ちのパソコンなどを家庭内LANに接続すれば、デジタル家電の使い勝手もきっと良くなることだろう。

いろいろ書いたが、別に**実用的な目的**がなくても、**単なる興味**だけで家庭内にLANを引いてみるのもよい。そもそも、現在のインターネットの基礎になっている「TCP/IP」というプロトコルは、LANのような環境においてもまったく同じものが使われているのだ。TCP/IPのLANを自分で構築すれば、家庭内に自営の**超小型インターネット**を作ったことにもなるだろう。

以前ほどは喧伝されなくなったが、**イントラネット**というのも、自前のTCP/IPネットワークをつくって、限定的にインターネットと接続する、というような意味だと思ってもらって間違いない（ずいぶん乱暴な解釈かもしれないが、大筋では間違っていない）。だから、たとえ2台ほどのマシンをEthernetとTCP/IPで接続して、ISDNルータなどでインターネットにダイヤルアップ接続しているだけでも、**ウチはイントラネットやってますから**と他人に吹聴できる。

LinuxとLAN

さて、家庭内にLANを引くからといって、別にLinuxなどのUNIX系OSが**必須**になるというわけではない。Windows系のOSだけでTCP/IPのLANを構築することは、現在においてもまったく問題はないし、プリンタ共有やファイル共有しかしないのなら、EthernetやTCP/IPについての専門的

な知識はほとんど必要がない。しかし、インターネットへの接続を複数のマシンで共有したり（つまり、複数のマシンから1本の電話回線を共有して、同時にインターネット上のサービスを利用できるようにする）、プロキシサーバなどを使ってWebページの閲覧性能を改善しようとすると、Windows系OSでもできないことはないのだが、Windows NTなどの**高価な**OS、あるいは市販のパッケージソフトなどが必要になるし、設定に当たってはTCP/IPについての専門的な知識と、Windows固有の「コツ」が必要になってくる。もちろん、Linuxを使って同じことをさせようとする場合でも、必要となる知識は同じだし、設定だって一朝一夕にはできるものではない。しかし、「高価な」ソフトウェアの使用料金を誰かに払わなければならないということはない。その一点だけでもLinuxに軍配が上がるのだ。

パソコンを使ってお金を儲けようというのでないのなら、ソフトウェアの代金として支払う金額は、少なければ少ないほうがよい。無論、お金を払えばその分手間は省けることも**多々ある**かもしれないが、少なからず知的興味をもって家庭でパソコンを使っているのなら、安易にお金を費やすことより、少しコンピュータやネットワークのことを学んで、自分の力でパソコンを使いこなすことを考えてもよいだろう。そういう意味からも、**素人**のユーザーにLinuxをおすすめしたいのである。

ネットワークカードとケーブルとハブ

2台以上のパソコン（あるいはパソコンとISDNルータなどのネットワーク機器）を接続するには、パソコン側にネットワークカード（ノート型PCならPCMCIAタイプのネットワークカード）、ネットワークカードに接続するケーブル、そして「ハブ（HUB）」と呼ばれる装置が必要になる。

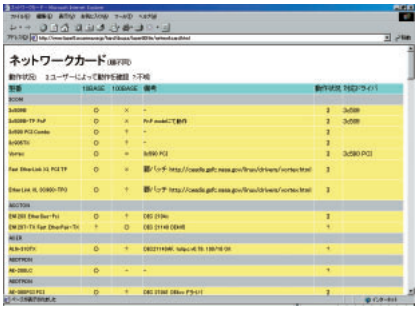
ネットワーク機器で一番やっかいなのが、パソコンに装着するネットワークカードの選択だ。これればかりは、使用しているパソコン本体の機種（あるいはマザーボードの種類、同時に使用している他のカード類の種類）などが問題になることもあるし、Linuxで使えないカードというのも多く存在する。また、広告や商品の箱などに**Linux対応**とうたわれているものでも、インストール時にフロッピーなどからドライバをインストールする必要があったり、面倒な設定ファイルを記述しなければならないものも多い。

ネットワークカードの選択で一番確実なのは、自分が使っているLinuxのディストリビューションの配布元からの情報を元に、確実にサポートされているカードを選択することだ。この際、対応機種一覧を見るときに注意すべきことは、「ベンダーが配布しているドライバが必要」とか、「設定ファイルの書き換えが必要」という注意書きが付記されている製品に注意することだ（**画面1**）。これらの注意書きは**小さな字でわかりづらい形で**書かれていることが多いので、つい見逃しがちだが、注意書きがあるということは、**素直にはインストールできない**ことを示しているのだと思っていたほうがよい。

ドライバの別途のインストールなどが必要のないカードなら、カードをパソコン本体に差してからLinuxを通常の手順で再インストールしたり、TurboLinuxを使っているのなら「turbonetcfg」コマンドを使ってネットワークカードの追加を行うだけで、ハードウェアの準備は完了である（**画面2**）。あとは、ネットワークカードとハブをケーブルで接続して、Linux上でネットワークの各種設定を行うだけでよい。

ホスト名とIPアドレス

ネットワークカードのインストールやケーブルの配線については、これ以上の説明はしない。というより、誌面の都合でできない。本誌のネットワーク関連の記事や、その他の雑誌記事や書籍を参照してもらいたい。といって



画面1　Linuxのネットワークカード対応一覧表の例（LASER5 Linux 6.0の場合）

対応一覧表に「パッチが必要」や「ベンダーが配布するドライバのインストールが必要」となっている場合は、パッチファイルやドライバファイルの入手が必要になる。また、インストールは、それぞれのファイルに付属する（はずの）ドキュメントに従って行う必要があるが、初心者には難しい作業となることも多い。「備考」欄になんの注意書きもかかれていないカードなら、Linuxなら特別な操作なしにインストールできるはずだ。

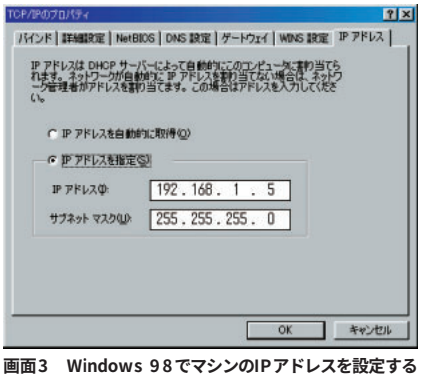
画面2　turbonetcfg（/usr/sbin/turbonetcfg）によるネットワークカードの追加

TurboLinux 4.0では、Linuxのインストール後にネットワークカードを追加したような場合、turbonetcfgユーティリティによって、インターフェイスの追加作業を行う。詳しい使用法については、TurboLinuxのマニュアルを参照してもらいたいが、基本的には画面上の指示に従って、各種パラメータを入力してゆくだけでよい。

も、それほど面倒な作業があるわけではないので、心配することはないだろう。

さて、ネットワークカードやケーブルの配線が済んだら、パソコン同士が通信できるようにいろいろな設定を行ってやる必要がある。市販のTCP/IPネットワークの入門書などを見ると、ずいぶん分厚い本が多く、また書いてある内容もずいぶんと難しいように思われて、躊躇してしまう読者も多いかもしれないが、インターネットに直接接続して、たとえばWebサーバやメールサーバを立ち上げようという**壮大な野望**があるわけでもなく、ただ単に家の中だけで**インターネットごっこ**をしたいのであれば、必要になる設定は驚くほど少ない。もともと、LinuxなどのUNIX系OSには、OS自体にTCP/IPによる通信を行う機能が組み込まれているので、若干の設定ファイルを記述してやるだけで、すぐに通信できるようになっているのだ。

また、LANによる通信の設定は、電話回線を使ってダイヤルアップでインターネットに接続する設定を行う場合に比べても、驚くほど簡単である。Windowsでダイヤルアップ接続の設定を行ったことのある読者ならわかるだろうが、ダイヤルアップ接続の設定には、接続先の電話番号の設定、発信元の電話に関する設定（市外局番や内線からの発信の方法）、モデムの各種設定（通信速度、フロー制御、ダイヤルトーンの有無の設定）が必要になる。Linuxの場合も同じで、これらの設定をきちんとしてやらないと、モデムが電話をかけてくれなかったり、つながってもすぐに切れてしまったり、あるいは、自動的にダイヤルアップ接続しようとしてもうまくいかなかったりと、**さんざんな**結果になる。



画面3　Windows 98でマシンのIPアドレスを設定する
家庭内LANで、すべてのマシンのIPアドレスを手動で設定したい場合には、［TCP/IPのプロパティ］の［IPアドレス］タブで［IPアドレスを指定］を選択し、設定したいIPアドレスの数値をIPアドレスの部分に入力する。［サブネットマスク］については、「192.168～」で始まるアドレスの場合は、なにも考えずに画面のように「255.255.255.0」を入力しておけばよい。

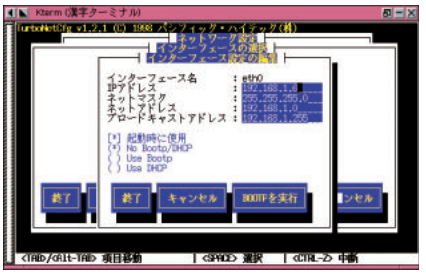
IPアドレスとホスト名

さて、LANのインターフェイスカードやケーブルで結ばれたマシン同士が通信するには、それぞれのマシンに**名札**をつけてやらなければならない。この名札のことを、**専門用語**では気取って**アドレス**と呼んでいる。インターネットやイントラネットの世界では、この名札として**IPアドレス**というものを使っていて、その実体は16ビットの数値である。3桁の数字がピリオドで区切られて4つ並んでいる形式で表記されることが多いので、たとえば「192.168.1.1」とか「10.101.1.1」などの数字の列を、どこかしらで目にしたことがあるだろう。

自分のマシンを、**本物の**インターネットに直接に接続したい場合は、しかるべき機関にお伺いをたてるか、あるいは大金を払ってその手続きを代行してくれる業者に依頼するかして、自分専用のIPアドレスを取得する必要がある。しかし、家庭内でインターネットごっこをしたり、あるいはダイヤルアップルータなどを使ってLANからインターネットを利用できるようにする場合には、そのような手続きは必要ない。その代わりに、「家庭内や会社内などの限られた範囲なら、誰でも自由に使ってよいアドレス」というものが存在する。いわゆる**プライベートアドレス**と呼ばれているもので、前述の「192.168.XXX.XXX」とか「10.XXX.XXX.XXX」とかいうアドレスがそれである。

ダイヤルアップルータなどを使って、一時的にでもインターネットへ接続するのだから、正式なIPアドレスが必要になるのでは、と心配する読者がいるかもしれないが、ダイヤルアップルータのほとんどは、プライベートアドレスからのアクセスを、ダイヤルアップ接続時に一時的にプロバイダから割り当てられる正式なIPアドレスに変換する仕組みがあるので、LANの内側では外のことを気にする必要はまったくない。

LANの設定を始めるに当たって、まず必要になるのは、自分のLANの中で使用するIPアドレスの範囲を決めてやることだ。ここでは、「192.168.1.1」から「192.168.1.254」という範囲のアドレスを使うことにする。LANに接続す



画面4　Linux（TurboLinux）でのIPアドレスの設定
通常は、Linuxをインストールしたときにアドレスの設定も行うが、インストール後に設定を変更したり、インターフェイスカードを追加したような場合は、turbonetcfgでIPアドレスの設定を行う。起動後の［ネットワークの設定］画面で［ネットワークインターフェイス］を選択してリターンを押し、インストールされているEthernetインターフェイス（通常「eth0」という名前になっている）を選択してから［編集］を選択すると、IPアドレスの設定画面になる。［ネットマスク］の設定は、画面3のWindows 98の場合と同様に「255.255.255.0」を指定しておけばよい。その他のパラメータについては、画面のように、デフォルトの値をそのまま使えばよい。

る装置すべてに、「192.168.1.1」、「192.168.1.2」、「192.168.1.3」といった順に番号をつけてゆく（ハブは除く。ハブは単なる電線の接続装置だと思えばよい。ただし、ダイヤルアップルータとハブが一体になっている場合については、ダイヤルアップルータの部分にはアドレスを割り当てる必要がある）。メモ用紙かなにかに、装置の一覧を作って、どの装置にどのアドレスを割り当てたかを書き留めておけばよい。

IPアドレスさえ割り当てておけば、普通の通信は支障なくできるのだが、IPアドレスは単なる数字の列で人間にはわかりづらいので、人間側の**便宜的な**手段として、英数字の文字列による**ホスト名**というものをつけることになっている。これは、IPアドレスとは違って、自分で勝手に好きな名前をつけてやればよい。たとえば、「banana」、「orange」、「grape」など、果物の名前にするか、「pc01」、「pc02」、「pc03」といった、素っ気ない名前にしてもまったくかまわない。

各マシンにIPアドレスと名前を割り当てたら、それぞれのマシンに自分のアドレスと名前を教えなければならない。この手順は、OSの種類やLinuxならディストリビューションによって変わってくるが、Windows 98の場合は、「ネットワークコンピュータ」のプロパティダイアログ（デスクトップ上の「ネットワークコンピュータ」アイコンを右クリックして「プロパティ」メニューを選択する）で設定できる（**画面3**）。Linuxの場合は、インストール時に表示されるアドレスやホスト名の設定画面で行うのがもっとも簡単だが、Turbo Linuxの場合なら、前述のturbonetcfgコマンドでも設定できる（**画面4**）。なお、Windowsの場合はあらかじめTCP/IPプロトコルもインストールしておかなければならないが、その手順については省略する。それほど難しい作業ではないので、オンラインヘルプなどを参照しながらトライしてもらいたい。

必要なのはhostsファイルだけ

ここまでの作業だけで、すでにLAN上のマシン同士でTCP/IPプロトコルによる通信はできるはずである。たとえば、Linuxのコマンドプロンプトから、

```
$ ping 192.168.1.1
```

と入力してやれば、**リスト1**のようなメッセージが表示されるはずだ。Windows 98でも同じようなコマンド（PING.EXE）があるので、MS-DOSプロンプトから試してみるとよい。

ただし、このままでは、たとえば、

```
$ ping orange
```

などと、ホスト名を使って通信しようとしても、できない。マシン同士が、お互いの名前を知る手段がないからである。このため、各マシン上に、LAN上のすべてのマシンの「名簿」のようなものを用意してやらなければならない。これが、いわゆる**hostsファイル**と呼ばれるもので、Linuxの場合は/etcディレクトリの下のhosts、Windowsの場合は起動ドライブの\Windowsディレクトリの下のhostsというテキストファイルになっている。hostsファイルの形式は、LinuxもWindowsも同じである（厳密には違うらしいが、ここでは細かいことは気にしないこと）。

ファイルの書き方も簡単で、IPアドレスとホスト名を並べた行を、LAN上のホストの数だけ追加してゆけばよい（**リスト2**）。ただし、一番上の行にある「127.0.0.1」で始まる行は、特別な意味を持つ行なのだが、ここでは気にしないで**おまじない**のようなものだと思っていればよい。

ここまでで、ちょっと誌面が足らなくなってしまったので、次回はLinuxをLANでつなぐとどんなことができるかについて、話を進めたいと思う。

▼リスト1

```
[zen-t@zen06 zen01]$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=255 time=0.8 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=255 time=0.7 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=6 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=7 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=8 ttl=255 time=0.6 ms
64 bytes from 192.168.1.1: icmp_seq=9 ttl=255 time=0.7 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.8 ms
[zen-t@zen06 zen01]$
```

▼リスト2

```
127.0.0.1       localhost

192.168.1.1     apricot
192.168.1.2     banana
192.168.1.3     citron
192.168.1.4     durian
```

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台
～取材から公開まで～

Linuxの総合情報サイト「日刊アスキー Linux（http://www.linux24.com/）の出張店舗ともいうべき当コーナー。今月は、1999年11月号で紹介した日刊アスキーの記事公開への流れを、さらに詳しくご紹介しよう。

本題に入る前に、少し告知をさせていただきたい。日刊アスキー LinuxはWebメディアとしてスタートしたが、ついにメールサービスも開始することになった。一昔前のプッシュではないが、やはりWebページのみの展開では、サービスとしても不十分であろうという理由からだ。もっとも、Linux関連のニュースはそれほど多くはないので、まずは週1本のペースから始めることになっている。これさえ読めば、その週のLinux関連の動きが分かる仕組みだ。いまのところ、まだまだ検討段階を脱していないので、最終的にどのような中身になるのか分からないが、近いうちに日刊アスキー Linux上でお知らせすることになるだろう。興味があれば、ぜひ登録のほど、お願いいたします。

日刊アスキー Linuxの記事作成方法

1999年11月号の当コーナーでは、技術担当のほうから日刊アスキー Linuxの部内システムについて詳しいレポートをお届けしたが、今月は編集サイドから、実際の取材から記事公開に至るまでの流れをご紹介しよう。

まずは日刊アスキー Linuxのスタッフの素性からご説明したい。現在、日刊アスキーで記事を作成する編集スタッフは3人いる。編集者2人と、アルバイト1人だ。そのほかは、編集長と技術担当という人員構成になっている。といっても、技術担当も実は元（というよりも現役）編集者なので、企画会議には必ず出席するし、原稿の執筆や校正も行う。この全スタッフの中で、雑誌や書籍（紙のメディア）経験がないのはアルバイトの1人だけで、あとの人間は、何らかの形で5年以上は紙メディアに関わってきている。当初、企画立案といった面はさておき、Webメディアも記事作成の実作業では、紙メディアの編集とたいして違いはないだろうと思っていたのだが、実際に作業を始めてみると、その違いにしばしば戸惑うこととなった。

それでは、ところどころ雑誌の作成と比較しながら、取材から記事公開に至るまで、ステップを追いながらご説明しよう。

取材

記事作成作業のもっとも初期に行われる取材に関しては、雑誌とはほとんど違いがない。取材の形態はいくつかある。有名人の来日や、新製品出荷の発表に合わせ、インタビューをしたり、メール、電話で話を聞き、記事の材料を集める。場合によっては写真を撮る。写真は、雑誌を作るときはフィルムカメラを使っていたのが、日刊アスキーになってからはデジタルカメラに変わった。使う機種はメガピクセルを条件に、月刊ASCII DOS/V ISSUEのデジタルカメラ記事担当などに相談して、COOL PIX 950に決定した（**写真右**）。

ちなみに、写真左のカメラは雑誌の取材などで使っていたキヤノンEOS Kiss。雑誌に掲載される写真が、プロ機材ではなく素人にも扱えるEOS Kissで撮られているのを疑問に思われる方もいるかもしれないが、COMDEXなどのショウを取材するときは、軽くて振り回しやすいし、もともとカメラの知識が乏しい人間には、オートですべてを任せられるシロモノのほうが頼りになるのだ。どちらにせよ、クオリティが必要な写真はプロのカメラマンに出動してもらうことになる。

フィルムのカメラからデジタルカメラになって、取材はかなり楽になった。デジタルカメラの導入は、もちろんWebがデジタル媒体だから、という理由からだが、副次的なメリットもあった。フィルムを交換する必要のないことや、持ち運びやすさなどである。特にショウ会場の取材は、デジタルカメラのほうが圧倒的に楽だ。ストロボをつけたフィルムのカメラを持ち運ぶだけでもつらいのに、さらに各社ブースで集めた資料が入ったバッグを肩から提げ、首から入場許可証、背中にはバックパックといったいでたちで、アレやらこれやらの紐が絡まる。さらに、ストロボで重心が上になってしまったカメラは、胸の上で不安定な動きをする。こうした状態でフィルムが切れると、バックパックからフィルムを取り出し、カメラのカバーを開けて取り替

え（このときもカメラ内部を傷つけないように神経を使う）、撮影済みフィルムをしまうといった動作が必要となる。さらに、実際にうまく撮影できたのか、とても不安になるのだ。

余談だが、筆者が海外取材に行ったとき、ふとした拍子でパノラマ撮影のスイッチを入れてしまい、帰社して現像したらすべて横長の写真となってしまっていて青くなったことがあった（いつも被写体を画面一杯には撮影せず、余白をとって撮影しているので、被写体が切れていることがなくて助かったのだが）。

これがデジタルカメラならば、その場で仕上がりが確認できる。さらに、取材中に行うのはせいぜい電池交換程度だ。電池の交換は、フィルムのように神経をとがらせる必要もない。使用済み電池をバックパックにバラバラと放り込み、新しい乾電池を入れて終わりだ。一番大事な撮影データに傷が付く可能性もないのである。

ここで、写真のクオリティについてもう少し述べておくと、雑誌において、

写真
左がEOS Kiss。右がCOOL PIX。ボディは同じ大きさだが、レンズの分EOS Kissのほうがかさばるのが分かるだろう。軽量派のEOS Kissさえ、デジタルカメラの手軽さにはかなわない。

ショウレポートの写真や、ニュースページで使われる小さいモノクロ写真などは、今やデジタルカメラでも十分な場合が多い。カメラの知識が乏しい編集者が撮影する場合など、むしろ液晶モニタで撮影結果が分かるデジタルカメラのほうが、ありがたくさえあるのだ。

取材ではもうひとつ、ノートパソコンも重要だ。編集部とメールのやりとりをするのは言うに及ばず、デジタルカメラで撮影した画像の保存場所にもなるし、講演などの取材の場合は速記しておけば原稿の元にすることができる。もっとも、インタビューでは、まさか相手と話をしながらキータッチすることはできないので、録音機材と紙のノートを使う。

こうして、テキストと画像という、記事の材料がそろうわけだ。

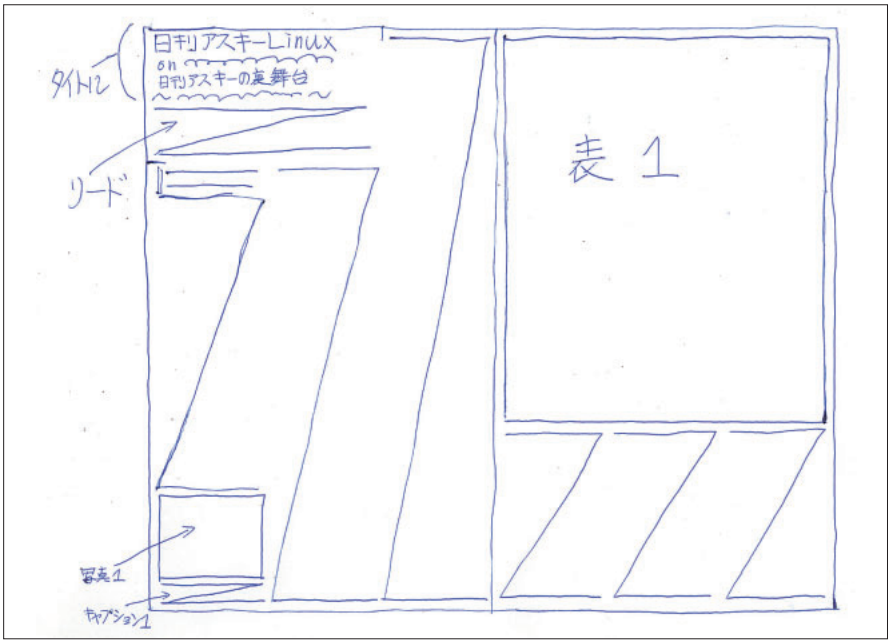


図　当コーナーのラフ

原稿作成

材料がそろったら、次は原稿作成だ。原稿執筆は、たいていの場合編集部に帰ってから行うのだが、速さが求められる場合は、取材現場での記事作成となる。たとえば、11月1日の「Corel LINUXの日本語化が発表」（http://www.ascii.co.jp/linux/news/today/article/article312736-000.html）の記事は、発表会が終わった直後に会場内のロビーで原稿を書き、画像とともに編集部にメールした。それほど急がない場合は、編集部に帰ってきてから、取材テープをテキストに起こしたり、画像を整理して検討して掲載、といった作業を行う。

ここで、雑誌であれば、

1.ラフレイアウトを描く
2.原稿を書く
3.誌面デザインのためデザイナーに文章と写真などを渡す
4.レイアウトが終わった誌面をプリン

トアウトし校正する

という作業を行い、最終的に印刷所にデータを入稿して本になるのを待つ。Webページの作成は1〜4よりも関わる人数が減った分、作業は簡略化される。その代わり、編集者1人でこなさなければならなくなった。

1.ラフレイアウトを描く

「ラフレイアウト（ラフ）」というのは、図のように誌面の割り付けを書いて、タイトルの位置や写真の位置をデザイナーに伝えるためのものだ。誌面の設計図といってもいい。ここで、原稿の文字数なども計算できる。

日刊アスキー Linuxは、短いニュース記事が主なコンテンツなので、あまりラフを描くことはない。長編の記事もたいてい簡単なメモ程度を描くのみだ。これは、文字の位置、画像の位置を直接決めるのが、記事作成者である編集者自身のため、ページレイアウトに関して第三者であるデザイナーの手が介入しないからだ。「自分が分かればそれでいい」のである。これは、作業工程が短縮できるぶん楽なのだが、実は裏をかえせば、デザイナーの仕事も自分たちでやっている、ということになる（もっとも、Webページそのものの雛形は決まっているので、デザインセンスを問われることはない）。

2.原稿を書く

もっとも違うのがこの段階だ。私の場合は、原稿を書きながらタグも付けていってしまうことが多い。本誌1999年11月号の当コーナーでも紹介したとおり、日刊アスキーではオリジナルの記事にXMLを採用している。そのため、タグに関しても日刊アスキーのコンテンツに則ったお手製のタグが使われる。

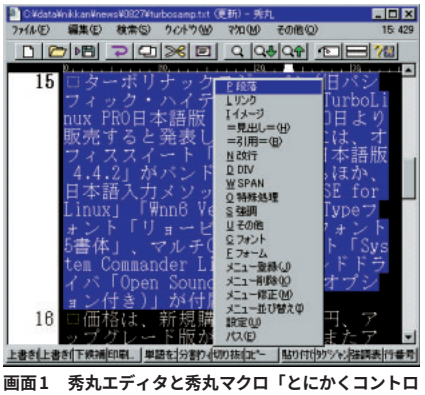
タグ挿入は、ヘッダなどの定型箇所は、テンプレートのファイルからコピーして持ってきてしまう。そのほかの段落（パラグラフ）や箇条書きは、私の場合Windowsの「秀丸エディタ」に“おーなーしぇふ”さん作のタグ挿入のための秀丸マクロ「とにかくコントロールdeエンター」を組み込んで使っている。とはいっても、実は 〜 の挿入程度にしか使っていないのだが。カスタマイズすれば、もう少し秀丸マクロで行う作業が増えると思うのだが、なかなか勉強する時間がとれず、今に至っている。

原稿を書く、という用途に限っては、XMLエディタよりもテキストエディタ＋タグ挿入型マクロが楽だ（**画面1**）。我々の場合、使うツールの第一条件として、原稿作成に適したものでなければならないからだ。一気に本文を書いていき、そのあとでタグを挿入したり、箇条書きの部分を付け足していったりと、まるで決まった手順がないのである。ところでヘッダなどの定型部分を記録しているファイルは、きわめて単純なものだ（**画面2**）。

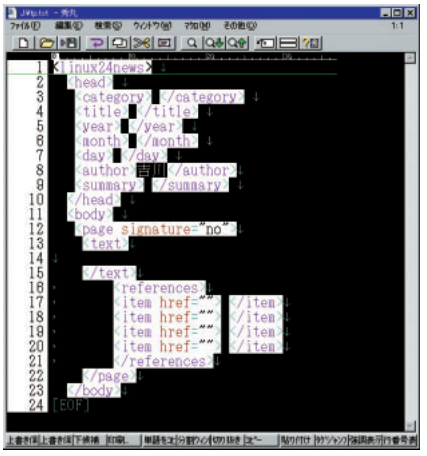
3.デザイナーに文章と写真を渡す

この段階に相当する部分は、我々にとっては部内サーバに原稿と画像ファイルをアップロードする過程にあたる。ここでは、XMLの構造チェックが行われ、日刊アスキーのXML仕様と実際の原稿（XML文書）の構造に違いがあると、エラーが出て登録できない（**画面3**）。構造に違いがある、とは、単純な例でいえばタグの閉じ忘れなどだ。そして、この構造チェックに合格すると、晴れて部内サーバに登録される。この際、原稿（XML文書）に記した通りの、タイトルや作成日、サマリーなどが表となり「データ内容」（**画面4**）として現れ、チェックできる。

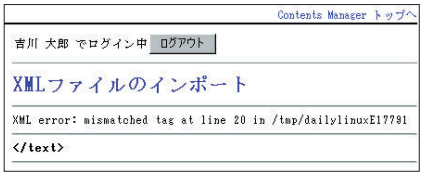
さて、テキストは無事に登録できたが、次は画像の登録がある。当然のことだが、これらはすべて画像データとして登録する。紙の写真だったらスキ



画面1　秀丸エディタと秀丸マクロ「とにかくコントロールdeエンター」。カスタマイズも強力だが、筆者はおもに段落のタグ挿入に使っている。



画面2　ヘッダおよびフッタのテンプレート。日付などはマクロなどを組んで自動的に挿入したいところだが、そうしたことに時間を割くよりは、原稿を書いたほうが効率的なのだ。



画面3　XMLの記述を間違えたときのエラー。このエラーが出たら、編集者は再度構文をチェックする。

ャナであらかじめ画像データにしておくし、図版もドローデータであるIllustratorのデータから、ただの画像ファイルにしておく。

ところで、デジタルカメラで撮った画像は、トリミングしたり解像度、色の明るさを調節している。現在筆者が使用しているデジタルカメラの撮影モードは、1600×1200ドットの大きさである。「大は小を兼ねる」ではないが、もしかしたら、雑誌の編集部から「あのときの写真、ない?」と言われた場合に、解像度が高ければ転用も可能だし、マザーボードの一部分を拡大して掲載する場合も考えられるからだ。だが、この1600×1200ドットという巨大なサイズでは、そのままWebに公開することはできない（物理的にはできても、読者が怒るだろう）。よって、これらを適切な大きさに変更する。こうして加工した画像は、登録した記事の「データ内容」ページで登録する（**画面5**）。

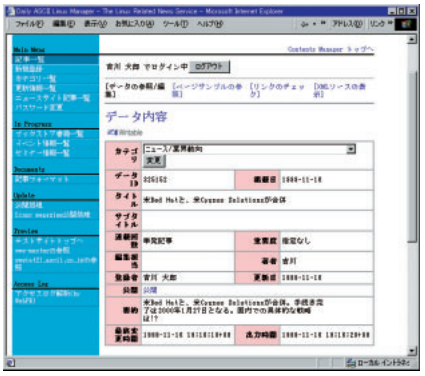
4.校正と査読

そして、最終段階である校正に入る。テキストも画像も登録が終わると、ページサンプルを印刷し、査読（原稿を他者が読んでチェックする）と校正に入る。ここまで来ていちいち紙に印刷するのは一見無駄なように見えるが、モニタ上で原稿をチェックするのと、紙で見るのでは、間違いを見つけられる確率が違うのだ。この違いは圧倒的といっていい。スタッフに紙媒体の経験が長いことが影響しているかもしれない。さらに、原稿に「ここを直したほうがよい」といった“赤字”を入れる場合も、紙のほうがスムーズである。ドキュメントにコメントを挿入する機能をもつソフトもあるが、紙に線を引っ張ったり注意書きをしたほうが、はるかにコトは簡単に済む。

ここまできたら、あとは外部用のサーバへと公開の処理を行えばいい。その仕組みについては、1999年11月号の当コーナーで紹介しているとおりだ。

文章で書くと長くなってしまうが、こうした過程を経ても、短いニュースであれば数10分で済む。日刊アスキーLinuxの運営当初はタグ付けなどに戸惑ったが、いまではまったく問題なくうまく機能しているのである。

（日刊アスキー　吉川）



画面4　データ内容の確認ページ。記事に割り振られるユニークな「データID」は別として、「タイトル」や「登録者」、「要約」といった項目が、画面2と対応していることがわかる。

画面5
「データ内容」ページで、関連画像データを登録する。登録表中にはサムネイルが表示される。サムネイルをクリックすれば、画面左のような、実寸の画像を確認できる。

Column

日刊アスキー Linux スタッフ募集

●募集職種
編集・編集アシスタント
（契約社員、アルバイト）

●資格・待遇
年齢経験不問（未経験者歓迎）
年齢、能力、経験を考慮当社規定により優遇。完全週休2日制(土日祝)

●勤務地
東京都渋谷区（京王新線初台駅より歩5分、小田急線参宮橋駅より歩7分）

●応募要項
①履歴書（写真貼付）
②職務経歴書
コンピュータで得意な分野（なくてもかまいません）についての作文800字以内を、封書に「日刊アスキー応募」と朱書きしたうえで郵送してください。書類選考のうえ、ご連絡いたします（応募書類は返却いたしません）。また、メールでも受け付けています。上記要件を満たしたメールを、linux24@ml.ascii.co.jpまでお送りください。

●応募書類郵送先
〒151-8024
東京都渋谷区代々木4-33-10
株式会社アスキー
日刊アスキー編集部
「日刊アスキー応募」係あて
問い合わせ電話番号　03-5351-8915

viバージンに贈る超入門講座

viはじめました

第5回　起動設定とmap、tagジャンプ機能

この連載も第5回、いよいよご紹介していないviの機能も残すところあとわずかとなりました。今回は後半で、プログラマー以外はあまり使わないかもしれないというコアな機能をご紹介します（じつは面白いんですけれどもね）。それだけでは皆さん面白くないかもしれないので、まずはviの環境設定（前回の続き）から始めましょう。環境設定はツール（vi）の使い始めに一度はまってしまうと、後々なかなかいじることはないかもしれませんが……。

文：佐々木太良
Text : Taroh Sasaki

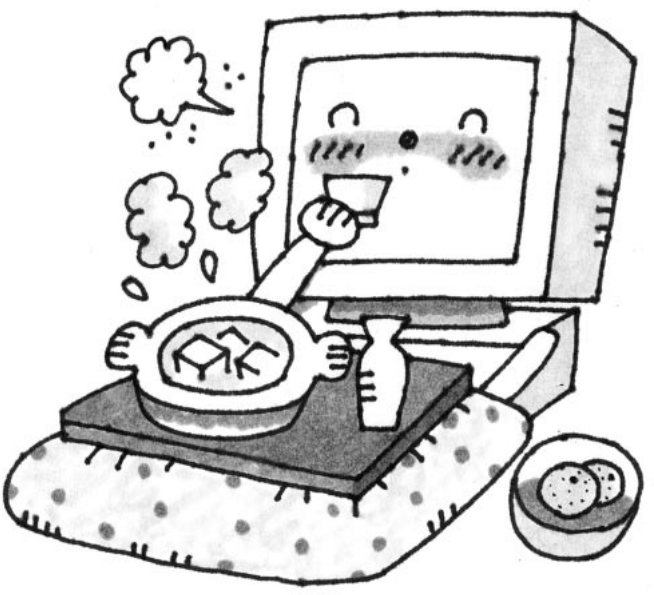


illustration ; Manami Kato

●コマンドの起動設定

UNIX系OSでは、各種コマンド・デーモンの起動条件を設定するファイルとして、『なんとかrc』という名前のファイルがいっぱいあります（『rc』は『run command』の略です）。

私の個人使いのワークステーションでは、ざっとホームディレクトリを見回しても『.cshrc』『.emiclockrc』『.fvwm2rc』『.mailrc』『.mirqrc』『.xinitrc』『.yatexrc』などというものがあります（何のrcファイルか興味のある人は調べてみてください……趣味がばれてしまいそうです）。/etcディレクトリには『csh.cshrc』『locate.rc』『mail.rc』などのファイルの他、マシン全体の動作に関わる初期設定として『/etc/rc.d/』または『/etc/rc』なんていうディレクトリ、ファイルまであります。

ちなみに以前の回で、他のOSではGUI系のセットアップツールがあるのにUNIX系ではエディタでこれらを編集するのが普通、と述べました。実はWindowsでもレジストリというファイルを各GUIツール（コマンド自身が書き込むこともある）が編集しているんですね。エディタで書き換えられる、各ツールの特性に応じた（逆にフォーマットが統一されていない）個別ファイルに環境設定をするのがよいのか、統一フォーマットで1個のファイルにまとめられた環境設定がよいのかは議論の分かれるところだと思います。Xウィンドウシステムの各アプリケーションの起動ファイル（X appricationdefaults）はこの2つの折衷のような路線を走っていますが、それはそれで煩雑な気もします。

いずれにせよ皆さんがviを使いこなせるようになったあかつきには、これらのファイルを編集することになると思うのですが……当のviにもrcファイルはあります。

各個人別のviの設定を保存しておくのは『~/.exrc』というファイルです。viはexの別名である（ことが多い）、ということはすでに触れましたが、どうして『.virc』でないのかは私には聞かないでください。

●.exrc を書こう

ここでいきなり『vi ~/.exrc』としてviの起動設定ファイルをガシガシ書いてもいいのですが、まずは皆さん、前回触れたexコマンド『:set』を使って、手動でviを使いやすい状態までもっていってみてください。

さあ、できましたでしょうか？　じつはviには、この状態で.exrcを保存し直すコマンドがあります。ここでexコマンド、

```
: mk hoge
```

としてみてください。カレントディレクトリに『hoge』という名前の、現在のviの設定一覧が記されたファイルができているはずです。この状態が気に入ったら『mv hoge .exrc』として次回からの起動設定にすることができます。

また、viからいきなり『mk .exrc』と設定をセーブすることもできます（すでに.exrcがある場合には『mk! .exrc』としないといけません）。

ただしnvi／nexバージョンのviでは、いじっていない（デフォルトの状態の）setパラメータもすべて保存されてしまっていますので、多少うざったい感じはするでしょう（vimバージョンではいじったパラメータだけが保存されるようです）。まあ全部のオプションが書かれた状態からスタートして.exrcファイルを編集するのも、見慣れないオプションと出会えたりしてなかなかオツなのではないでしょうか。

●恐ろしくも面白い『map』

exコマンド『map』は、viモードでの各種操作の置き換えを行うことができます。ちょっと[:]map で現在のmapを見てみましょう。**図1**はnvi/nexでの例です。

『^[OA』などは、端末エミュレータのカーソルキーを押したときに『通常出ると考えられる』エスケープシーケンスです（具体的にはkterm・xtermの他、DEC VTシリーズなどの古典的端末もこれらを吐くようです）。これを『hjkl』などのカーソル移動コマンドに割り当てておけば、カーソルキーでカーソルが移動できるという（ごく当たり前の）ことができるようになるわけです。

なおvimではtermcapに書かれているカーソルの上下左右移動のみを使用するため、初期状態でのmapは一切ないようです。

ところでmapできるのはこんなつまらない（?）ものばかりだと思ってはいけません。viコマンドでできるあらゆ

```
+=+=+=+=+=+=+=+=+
^[OA  k     cursor up
^[OB  j     cursor down
^[OC  l     cursor right
^[OD  h     cursor left
^[OH  ^     go to sol
^[[2~ i     insert at cursor
^[[5~ ^B    page up
^[[6~ ^F    page down
^?    x     delete character
Press any key to continue [: to
enter more ex commands]:
```

図1 nvi/nexのデフォルトのmap

Column

システム全体の設定を変えよう

個人のviの起動設定を変更する場合は、本文中で触れているように『~/.exrc』ファイルを編集すれば事が足ります（なお蛇足ですが、他のコマンドの起動設定ファイルも含めてファイル名がドット『.』で始まっている場合が多いのは、ホームディレクトリのリスト『ls ~』を取ったときに、ファイルが大量に表示されるのを防ぐため不可視ファイルにしているのです）。

UNIX系OSではこれとは別に、あるコマンドについて『マシン全体の』起動設定ファイルを置けるようになっているのが普通です（こちらの場合は不可視だとむしろ不都合なため、『.』で始まっていないファイル名が普通です）。

nvi／nex（BSD系）の場合は『/etc/vi.exrc』（これまた謎な名前ですねえ）、vim（Red Hat Linux 6.0）の場合は『/usr/share/vim/vimrc』がマシン全体の起動設定ファイルとなります。使い方は個人の『.exrc』と同一です。

ではどちらのファイルに設定を書けばよいかというと、個人所有のフリーUNIX系マシンではどちらでもあまり変わりはないといえましょう。ただし設定をどんどん変えていきたいのにマシン全体の設定ファイルはroot権限でないといじれないとか、1人で複数アカウントを使い分けるとき（パーソナルマシンでも最低、rootと個人アカウントは使い分けるべきです）、設定がすべてのアカウントに反映されていないとイヤだ、ということはありうるでしょう。よく方針を考えてください。

大学の研究室などで1台のマシンを共用する場合は、初心者ユーザー（新規配属生）などのために管理者が『使いやすい（と信じられている）設定』をしておいてあげることがあります。このような場合にはマシン全体の設定を変えるファイルをいじるべきですが……管理者の場合、あまり個人の趣味を押し付けるのはやめたほうがよいかもしれません（笑）。

なお新規ユーザーアカウントを作成する場合、adduserコマンドを実行するだけではなく、ある程度の環境をセットアップするシェルスクリプトを用意している周到な管理者の方も多いと聞きますが、viの多くのバージョンでは、.exrcファイルは当人の持ち物でないと無視される場合があります。その場合はスケルトンユーザーへのシンボリックリンクで済ませることはできません（だからこそマシン全体の設定を使う方法が用意されているわけですが）。

詳しく触れることは避けますが、起動設定ファイルをクラッキングすることでセキュリティが脅かされるケースがあるからです（それにつけても /etcの各ファイルのセキュリティには気をつけてください）。

さて、これらの優先順位ですが『viのデフォルトの設定』→『マシン全体の設定』→『個人の .exrcの設定』の順に高くなっていくことはちょっと考えれば明らかでしょう（つまりviは起動時にその順序でファイルを読んで内部の設定を変えていくわけですね）。ですから、新規ユーザーが初心者ユーザーのうちは、上級管理者が用意しておいてあげたマシン全体の設定に従い、気になるところが出てくれば（スキルがついてくれば）自分独自の設定をする、ということができるわけです。

このコラムで触れたことは他のコマンドの起動設定ファイルについても当てはまることですが、viの操作法を覚えようという皆さんは他のユーザーをも管理する管理者になることも多いでしょう（そして起動設定ファイルをいじることも多いでしょう）から、ちょっと覚えておいてくださいね。

る動作の短縮形を、通常の入力として割り振ることができるのです。ためしに、

```
[:]map <img I<img src="" alt="">^[?" ^[i<Enter>
```

としてみましょう（ここで『^[』はCtrl[V][ESC]と入力してください）。普通に使っている分には何ら変わりないように見えますが……。viコマンドで『<img』と打ってみると、次の瞬間、

```
<img src="" alt="">
```

が入力され、最初のダブルクォートの間の位置にカーソルが来て入力モードになっていますね（必ずviモードでやってくださいね）。つまり『<img』という文字列をタイプした瞬間、

1. 行頭に「<img src="" alt="">」を挿入
2. インサートモードを抜けた後『"』の前に移動（検索）して
3. 再度インサートモードに入るを実行しているのです。

もちろん、これは.exrcに書いておくこともできます（[:]mkで書き出されます）。

この調子で『<a』『<table』など定義していけば、面倒くさいHTMLタグを書くのも楽勝だぜ〜、などと調子に乗っていると痛い目に会います。たとえば[<][<]に何かの動作を割り当てた場合、viのデフォルトの動作（この例はインデント幅を下げるものでした）が上書きされて機能しなくなります。まあmapに失敗した場合は『unm(ap)』（カッコ内は省略可）コマンドで解除できるのですが。そうでなくとも何の短縮形を何の入力割り当てたか忘れて結局使わなかったり、思わぬ時に短縮形が展開されてひどい目にあったりします……（笑）。よく考えて短縮形を設定しましょう。

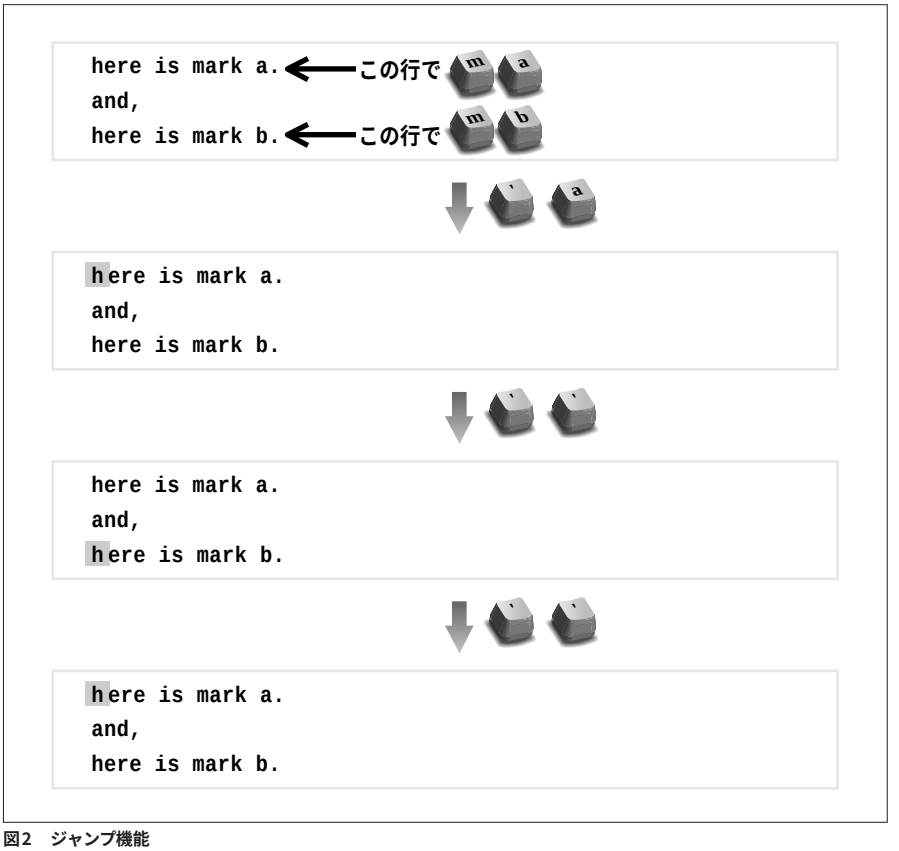


図2　ジャンプ機能

●マークセットとジャンプ

さてviのカスタマイズ関連の話題から離れて、ここからはまた通常のviコマンド・exコマンドの話題に戻ります。ただし一番最後にとっておいたのは、あまり使用頻度が高くないかも、という理由からです（もちろん人によっては覚えればバリバリ活用できるかもしれません）。

まずはviコマンドのマークセットとジャンプです。これは編集中のテキストファイル中に目に見えないマークをセットしておいて、必要なときに一発でその場所（行単位）にジャンプするというものです。マークのセットはviコマンドで[m][マーク文字]です。マーク文字はアルファベット1文字です。マークした場所にジャンプしたい場合は[']][マーク文字]（または[']][マーク文字]）とします。

この機能は実際に実行してみれば一発で理解できるほど簡単で、また有用そうには見えますが……。「あまり使い道はないかも」と書いたのは、『どこに何のマーク文字を振ったか』を人間があまり覚えておけないのではないかと思われるからです。私も数カ所のジャンプなら[m][a]、[m][b]、[m][c]くらいまでは付けてみたりしますが、それ以上になるとどこが何だったか分からなくなるので（笑）、実際にはあまり多数のマークは使いません。

なお[']][マーク文字]よりよく使うのは、現在のカーソル位置と前回ジャンプした位置を交換するジャンプ機能です**（図2）**。ここでいうジャンプは[']][マ

ーク文字]によるジャンプばかりではなく、[:]行番号<Enter> でジャンプした場合のジャンプなども含まれます(viのバージョンにより微妙に違いがあります)。emacsの[C-SPC]と[C-x][C-x]に似ています。

●tagとの連携

ひと昔前までUNIX系ワークステーションのユーザー(=viユーザー)=プログラマー、という図式は正しかったので、viのこの機能は覚えることが必須、のようにいわれていたのですが……。

viには『tag』に簡単にジャンプしたり戻ったり、という機能があります。ここでいうtagとはいわゆる『見出し語』のことで(じつは何でもいいのですが)、たとえばC言語であれば関数の定義や#define でマクロを定義している部分のことです。フリーUNIX系であれば必ず『ctags』というタグ作成コマンドが用意されているので、以後C言語に限って話を進めましょう。

まずctagsでtagファイルを作成します**(図3-a)**。これはカレントディレクトリにある『tags』という名前のファイルがデフォルトで使われます。シェルのコマンドラインから『ctags ソースファイル名』として(『ソースファイル名』の部分にはソースの『.c』ファイルや『.h』ファイルを指定できます)、tagsファイルを作成します。ctagsコマンドには既存のtagsファイルにタグを追加する『-a』オプションや、既存のtagsファイルのタグエントリを置き換える『-u』(GNU ctagsでは『--update』)オプションなどがあり、通常使うときはこれらを指定することが多いので、本気でC言語のプログラマーを目指している人はマニュアルを一読(するばかりでなく、壊してもよいファイルで実験)しておいてくださいね。

この後viを普通に起動します。

```
% vi ping.c
```

カレントディレクトリにtagsファイルがあれば、viはそれを自動的に読み込んでくれます。関数や定数が使われているところにカーソルを合わせ、viコマンド[^]](コントロール+])を実行するとあら不思議、それを定義しているところにジャンプするではありませんか**(図3-b)**。

関数の定義を見終わったら元の場所に戻ってこなくてはいけません。このときはexコマンド[:]pop(vimの場合。nex/nviでは[:]tagp)を使います**(図3-d)**。

さらにtagはある定義の中で別の定義を呼んでいることがあるため、定義を見ている最中にさらに別の定義にジャンプした場合は、次々とtag stackというところに『どこから飛んできたか』が積まれるようになっています。これを見るにはexコマンド[:]tagsを使います(**図3-c**・vimの場合。nex/nviでは[:]di t(display tags)です)。

いってみればWebブラウザでリンクしている部分を選択しジャンプし、読

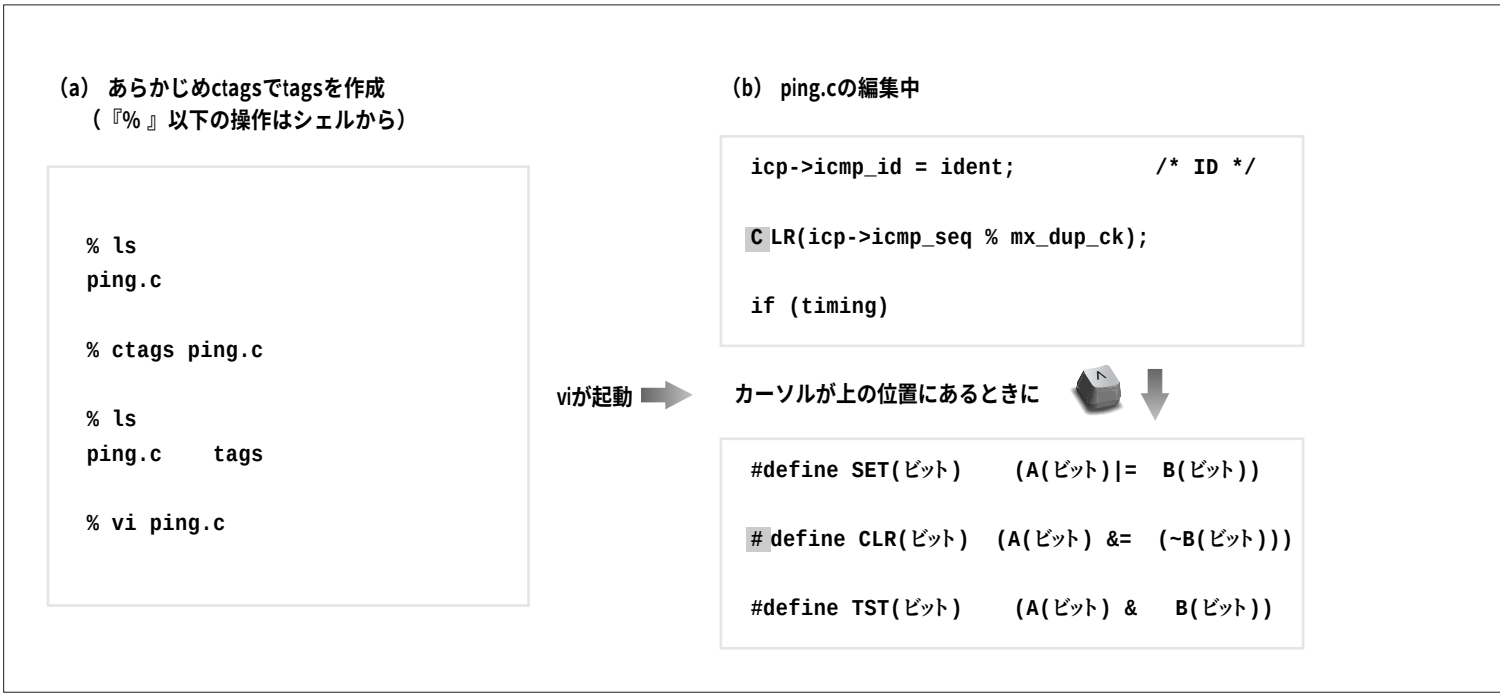


図3 viのtagジャンプ

んだ後『←』ボタンで戻ってきたり、Emacsのinfoでリンク部分を[Enter]で選択して読んだ後[l]で戻ってきたりという、コンピュータならではのクロスリファレンス機能の元祖ということができます。viは単一ウィンドウでファイルを交互に編集するのが原則、という古風なスタイルではありますが、いかに複数ウィンドウが開いたりGUIで操作できたりしても、同様の操作性を得ることは難しいでしょう。…といいつつ有名どころのエディタにはたいていtagジャンプ機能はあるんですけれども……。

ところでtagジャンプ機能は、編集中のファイル（時々刻々変化する）を相手にしているため、この機能の実現にはなかなか工夫がみられます（コラム『tagsファイルの内容』参照）。

●ご意見募集中！

今回ご紹介した機能は、なかなかviのviたる所以ではないか、と思う反面、「ctagsなんて今時あまり使わないかもなー」というみなさまのご意見・ご感想をぜひぜひtaroh@taroh.orgまでお寄せください。この連載も、いよいよ次回で最終回です。次回は大詰めとして、実用的な用法を特集する予定です。ではhappy viing‼

Column

telnetでは

viの貴重な機能であるtagジャンプのキーは[^]]に割り当てられています。telnetでリモートホストのviを使っているとき、このキーはtelnetのデフォルトのキー配置では『通信をやめてtelnetコマンドモードに入る』キーとなっています。読者の中に今、記事通りの操作を試してみて『telnet> 』なるプロンプトが出てきてびっくりした方が52名ほどいらっしゃるかと思いますが、慌てず騒がずリターンキーを押せばまた通信モードに戻ります。

でもこれではtelnetからviのtagジャンプ機能が使えませんね。telnetプロンプトが出たところで、エスケープ文字をviで使わない文字（たとえば[^@]）に割り当ててしまえばOKです。『set escape ^@』（この場合の『^@』は『^』『@』という2文字です。sttyコマンドの『stty erase ^H』などという場合と同様です）。あ、これでは（[^@]は[C-SPC]と同一）今度はemacsが使えませんか……（笑）。

Column

バッファの内容を覗く

本文中ではnex／nviのtag stackを見るためにexコマンド [:]diを紹介しましたが、第2回目で触れた『カット・コピーしたバッファの内容』を見るために、このコマンドを使うことができます。[:]di b （display buffers。vimの場合は[:]di）コマンドを実行してみてください。バッファの内容と、nex／nviではそれが行単位で切り取られたものか文字単位で切り取られたものか、なども一緒に見ることができます。

（c） tag stackの内容表示

```
[:]tags (または[:] di t)
```

```
 # TO tag         FROM line  in file
 1  1 CLR              598  -current-

Press RETURN or enter command to continue
```

（d） 元の場所に戻ろう

```
[:]pop (または[:]tagp)
```

```
CLR(icp->icmp_seq % mx_dup_ck);


if (timing)
```

```
A   ping.c  /^#define   A(BIT)      rcvd_tbl[(BIT)>>3]  ¥/* identify byt/
B   ping.c  /^#define   B(BIT)      (1 << ((BIT) & 0x07))   ¥/* identify /
CLR ping.c  /^#define   CLR(BIT)    (A(BIT) &= (~B(BIT)))$/
```

図4　tagファイルの内容

Column

tagsファイルの内容

本文中で作ってみたtagsファイルの内容を見てみましょう（図4）。このファイルは、検索するtag名、それが定義されているファイル名、そして正規表現による検索パターン（1行目や2行目では、通常の文字『/』が正規表現文字とみなされないようエスケープされて『\/』となっているのがわかります）によってできています。

タグの定義されている位置をファイル中の行番号で指定すれば簡単そうですが、何しろ相手は編集中のファイルであり、いつ行番号がずれるかわからないため、このような工夫がされているのです。

ということは、タグの定義の行（たとえば『CLR』タグであれば『#define〜(~B(BIT))』までの内容が変更されてしまうと、以後タグの定義が引けなくなってしまう、ということがいえます。その場合は再度『ctags -u』によってタグを更新しなければなりません。

なお余談になりますが、ctagsはvi/exが使うためのtagsファイルばかりではなく、Cのプログラムのソースコードを出版するとき目次に使うことができるvgrind型式のファイルというのも生成することができます。

●今月取り上げたコマンド

重要度
★★★ = 覚えるまではviを起動しないほうがよい　☆☆☆ = 知らなきゃ死ぬでしょ　☆☆= 知っていると編集が速くなる　☆= 知っていると自慢できる
(なし) = 知っていてもあまり自慢できない

viコマンド

コマンド	重要度	動作
m文字	☆	マーク『文字』をセット
''	☆☆	前回ジャンプ位置に飛ぶ
^]	☆☆	カーソル位置のtag定義に飛ぶ
'文字	☆	マーク『文字』に飛ぶ
`文字	☆	['][文字]と同じ

exコマンド

コマンド	重要度	動作
mk(exrc)<!> ファイル名	☆	exrcファイル型式で現在の設定す
map 操作1 操作2 ……	☆	viコマンド『操作1』が行われたとき『操作2』が行われたかのように置き換える
unm(ap) 操作		『操作』のmapを解除する

(vimバージョンのvi)

コマンド	重要度	動作
pop	☆☆	tagスタックを1個戻す(元の場所に飛ぶ)
tags	☆☆	tagスタックを見る
di(splay)	☆	カットバッファの内容を見る

(nex/nviバージョンのvi)

コマンド	重要度	動作
tagp(op)	☆☆	tagスタックを1個戻す（元の場所に飛ぶ）
di(splay) t(ag)	☆☆	tagスタックを見る
di(splay) b(uffer)	☆	カットバッファの内容を見る

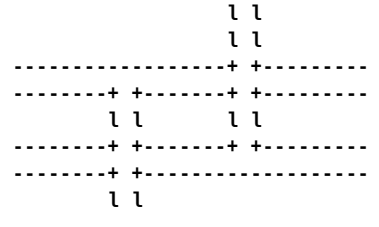
exコマンドモードに入るときの[:]および最後の<Enter>(または<Esc>)は省略。(…)内は省略可能なことを示す。<…>内は必要なら追加するオプションを示す。

＜詰めvi＞

今回は手数の少なさを競うのではなく、『こんなことは実現できないだろうか』というのをエレガントに実現してみましょう。

●地図エディタ

メーリングリストの宴会などで流布される（？）こんな文字地図をviで簡単に作れるように、viをカスタマイズしてみましょう。「[O]で地図エリアを広げ、[H][J][K][L][+]で移動しながら線を引く」のに便利なmapを考えてください。なお縦線は『l』(小文字エル)を使っています。



詰めvi 答え　こんなmapはいかがでしょう(『^[』は[Ctrl + V][ESC]の2文字を打って入力)。

```
map     O       o^[72a ^[
map     H       r-h
map     J       rlj
map     K       rlk
map     L       r-l
map     +       r+
```

「.exrc」にこれを書いておくと、[O]で空白の行を作成し、[H][J][K][L]で動き回りながら線を引けます（もちろんこのマップを使っている間、通常の[H][J][K][L][O][+]は機能しません）。[h][j][k][l]では通常通りカーソル移動ができます。地図の作製が終わったら、「.exrc」は「.chizu」とでもリネームしておくといいでしょう。

賢く使うUNIX

これであなたもスマートなUNIX使い！

―RPM使いになろう（前編）―

Linuxの標準シェルであるbashのコマンドラインを中心として、Linuxの便利な使い方について紹介していく本連載。今回は、Red Hat系ディストリビューションのパッケージ管理システムであるRPM（Red Hatパッケージマネージャ）について取り上げ、特にコマンドライン版rpmの問い合わせ機能を中心に紹介する。

Illustration : Manami Kato

今月のお題

パッケージ名とファイルの対応リストを作る

文：**大池 浩一**
Text : Kouichi Ooike

Red Hat Linuxをはじめ、Vine、Turbo、LASER5などいわゆるRed Hat系のディストリビューションで広く採用されている「RPM」（Red Hatパッケージマネージャ）。ディストリビューションのインストールの際はもちろん、ソフトのインストールやバージョンアップ、アンインストールの際にお世話になっている人は多いはずだ。

RPMの特徴は、専用のデータベースを利用してファイルの管理を行うことにある。各パッケージがどんな名前のファイルをどのディレクトリにインストールしたか、そのパッケージを利用するにはどのファイルが必要か、といった情報が一元管理されている。たとえば、他のパッケージが利用しているライブラリを含むパッケージをアンインストールしようとすると、依存関係のチェックではねられるため、通常はアンインストールできない。

ユーザーがRPMを利用するインターフェイスとしては、コマンドライン版のrpmと、GUI版のglint／gnorpm／Kpackageなどがある。GUI版は初心者にも使いやすく、インストール済みのパッケージがグループ別にツリー表示されたり、パッケージに含まれるファイル名の一覧などをウィンドウに表示できる。しかし、他のコマンドと組み合わせて使う場合は、標準入出力を利用するコマンドライン版のrpmのほうが都合がいい。

RPM使いになろう

このように便利な特徴を持つRPMだが、「rpm -Uvh hoge-1.2.3-4.i386.rpm」などとして、バイナリパッケージをインストールするのに使うだけという人も多いだろう。また、異なるディストリビューション用のバイナリパッケージをそのままインストールしようとして各種チェックではねられたり、「--force」や「--nodeps」といったチェックを無視するオプションを指定してインストールしてシステムをおかしくしてしまった人もいるはずだ。

RPMを便利に使うには、単にインストールの手順を知るだけでなく、現在のシステムの状況をデータベースに問い合わせたり、使っている環境に合ったバイナリパッケージを自分で作成する方法を学ぶ必要がある。

そこで今回は、コマンドライン版のrpmを利用して、データベースへの問い合わせ機能（-qオプション）の使用方法を説明する。「今回のお題」では、rpmの問い合わせ機能を利用して得られた出力を、さらにAWKなど他のコマンドに渡したり、forやif、[]といったbashの制御構文を使って処理することで、rpm単独では実現できないテキスト処理を行う。

RPMパッケージの基礎知識

RPMのパッケージファイルには、ファイル名が「.src.rpm」で終わるソースパッケージと、「.i386.rpm」や「.noarch.rpm」などで終わるバイナリパッケージの2種類に大別できる（例外的に、そのどちらでもない「.nosrc.rpm」もあるがここでは触れない）。

ソースパッケージには、プログラムのソースファイルなどが含まれたパッケージで、バイナリパッケージを作成するために使用する。一方、バイナリパッケージはコンパイル・リンク済みの実行ファイルが含まれており、これをインストールすることにより、いちいちソースファイルをコンパイルすることなくソフトを実行できる。

いずれの種類のパッケージとも、ファイル名の前半部は「パッケージ名－バージョン番号－リリース番号」という書式で統一されている（**図1**）。パッケージ名は、インストール後の問い合わせやアンインストールの際などに使われる名前で、バージョン番号はインストールされるソフトのバージョン、リリース番号はパッケージ自体を何回リリースしたかを示している。たとえば、「hoge-1.2.3-4.i386.rpm」というファイル名を見れば、これはhogeというパッケージで、そこに含まれているソフトのバージョンは1.2.3、パッケージのリリースは4回目、インテルのi386以降のCPUで動作するようにられたバイナリパッケージであることがわかるわけだ。

バージョン番号とリリース番号が独立しているので、新しいパッケージが出たときにも、ソフト自体がバージョンアップしたのか、パッケージの構成が変わっただけなのかをファイル名から判断できる。なお、バージョン番号やリリース番号には、数字以外の文字を含んでいてもよい。ベータ版であることを示すために「1.0beta1」としたり、Red Hatのリリースしたパッケージをさらにvineで修正したので「1vl3」としたりといった具合だ。ただし、区切り文字の「－」を除んでいてはいけない。一方、パッケージ名には「－」が含まれることがある。たとえば、「hoge-devel-1.2.3-4.i386.rpm」というバイナリパッケージの場合、パッケージ名は「hoge-devel」だ。

インストールされたバイナリパッケージの情報は、RPM専用のデータベースに蓄えられる。パッケージの情報はpackage.rpm、パッケージに含まれるファイルの情報はfileindex.rpmといった具合に、データベースは複数のファイルで構成され、通常は/var/lib/rpm以下に置かれている。ただし、これらはバイナリファイルなので、内容を直接参照したり、編集したりすることはできない。バイナリファイルをわざわざ採用しているのは、大量のパッケージが登録されている状況でも、検索などの処理をできるだけ高速化するためだ。

コマンドライン版のrpmを利用して、データベースに問い合わせを行えば、テキストで出力が得られるので、実質的にテキストで登録されているのとあまり違いはない。「今回のお題」では、rpmの問い合わせ機能で得られた出力を

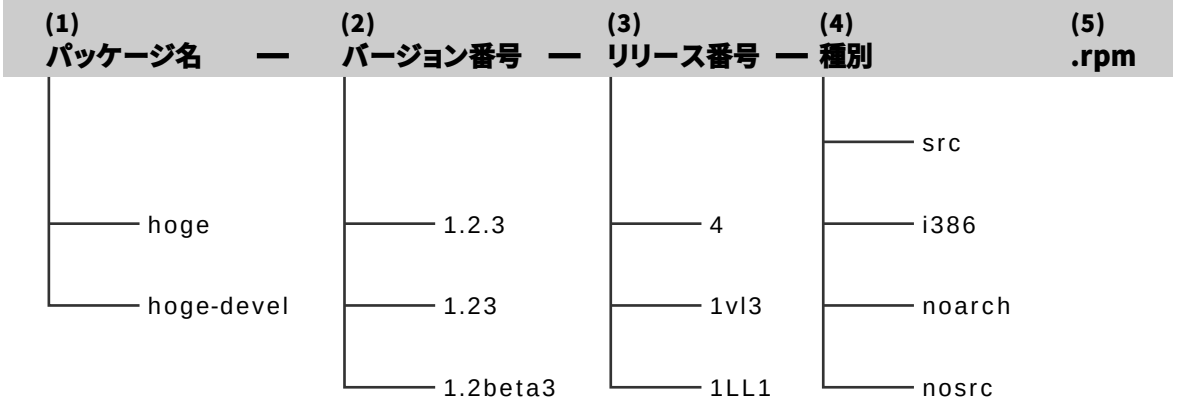


図1 RPMのパッケージファイル名の構造

さらに他のコマンドに渡すことで、rpmだけでは実現できないテキスト処理を行っている。

●問い合わせを行うrpmの-qオプション

rpmに-qオプションを付けて実行すると、すでにインストール済みのパッケージや、これからインストールしようとしているパッケージのさまざまな情報をテキストとして目に見える形で表示してくれる。

インストール済みのパッケージの情報を得るには、「rpm -q パッケージ名」とすればいい。データベースからそのパッケージを検索し、バージョン番号やリリース番号も含めて表示する。たとえば、「hoge-1.2.3-4.i386.rpm」をすでにインストール済みの場合は、

```
$ rpm -q hoge
hoge-1.2.3-4
```

となる。パッケージ名だけでデータベースから検索できるため、コマンドラインではバージョン番号などを指定していないことに注意されたい。

一方、まだインストールしていないパッケージの情報を得るには、-pオプションを追加して「rpm -qp パッケージファイル名」とする。たとえば、

```
$ rpm -qp hoge-1.2.3-4.i386.rpm
hoge-1.2.3-4
```

となる。今度は、データベースにパッケージの情報が登録されていないので、パッケージファイル名をまるごと指定する必要がある。

さらに、-qオプションに以下で説明するオプションを追加すると、パッケージの説明や含まれるファイル名の一覧、パッケージのインストールに必要なファイルといった情報が表示される。なお、以下ではデータベースに対する問い合わせを行うコマンドラインを挙げるが、インストール前でも-pオプションとパッケージファイル名を指定すれば同じ情報を得られる。

まず、そのパッケージがどんなものかわからない場合には、-iオプションを追加する。たとえば、

```
$ rpm -qi hoge
```

とすると、hogeに関する詳しい情報（ディストリビューション、ベンダー、作成日時、ライセンス、パッケージ作成者、入手先URL、説明文など）が表示される（**画面1**）。

続いて、-lオプションでパッケージに含まれるファイルの一覧を表示してみよう。たとえば、

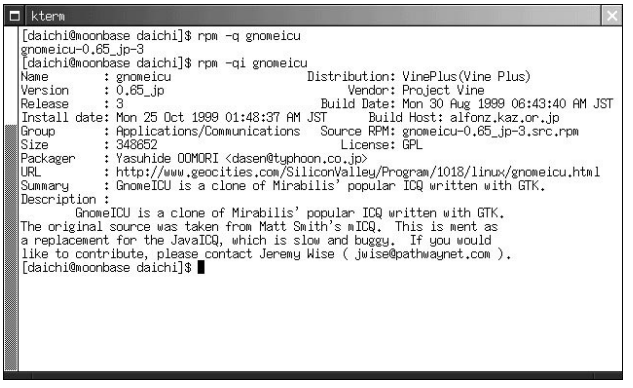
```
$ rpm -ql hoge
/usr/bin/hoge
/usr/doc/hoge-1.2.3
/usr/doc/hoge-1.2.3/AUTHOR
      :
```

のようになる。/usr/binに実行ファイルが、/usr/doc/hoge-1.2.3に各種ドキュメント類がインストールされていることがわかる。

RPMの特徴である依存関係のチェックには-Rオプションを追加する。すると、そのパッケージをインストールするのに必要なファイルが一覧表示される。たとえば、

```
$ rpm -qR hoge
tk >= 8.0.3
/bin/sh
libc.so.6
  :
```

といった出力が得られ、ライブラリのlibc.so.6やコマンドの/bin/shが必要であることがわかる（tkのように動作するバージョンも指定可能）。なお、データベース内のファイル情報に対してチェックが行われるため、こうしたライブラリやコマンドも、あらかじめRPMのパッケージとしてインストールしておく必要がある。逆に、指定したパッケージを必要とする他のパッケージを知る--whatrequiresオプションも用意されている。



画面1 rpmの-qiオプションで表示される情報

他のコマンドやbashの機能と組み合わせる

それでは、rpmの-qオプションの出力を他のコマンドやbashの機能と組み合わせると、何が実現できるのかを見てみよう。なお、以下の例の出力はVine Linux 1.1における結果なので、他のディストリビューションではパッケージ名などが異なるものになるだろう。

まずは、インストールされているかどうか知りたいパッケージ名の一部分しかわからない場合を考えてみよう。パッケージ名を完全な形でrpmに与えられないので、データベースに含まれる全パッケージの情報を出力する-aオプションを追加する。

```
$ rpm -qa
setup-1.9.2-1vl2
filesystem-1.3.2-3
basesystem-4.9-3
    :
```

のように、インストールされているパッケージ名（バージョン番号・リリース番号含む）がすべて出力される。この中から目的のパッケージを探すには、本連載でおなじみの検索ツールegrepにパイプで接続し、パッケージ名の一部を検索パターンとして指定すればいい。たとえば、X Window Systemの「XFree86」という文字列を含むパッケージの一覧を得るには、

```
$ rpm -qa | egrep 'XFree86'
XFree86-SVGA-3.3.5_jp-0.1.1
XFree86-VGA16-3.3.5_jp-0.1.1
    :
```

とする。英字の大文字・小文字を無視するにはegrepに-iオプションをつければいい。また、検索パターンに正規表現を使えば、「^XFree86」（XFree86で始まる）とか、「vl[0-9]+$」（vlと1桁以上の数字で終わる）といった条件に合うパッケージ名を検索できる。正規表現については本連載の第2回めの記事を参照されたい（http://www.ascii.co.jp/linux/allascii/linuxmag/no_03.html）。

続いて、上の例で得られたパッケージ名を使って、各パッケージに含まれるファイル一覧を表示してみよう。とはいっても、検索されたパッケージ名をいちいちキー入力する必要はない。これも本連載ではおなじみ、標準入力の内容をコマンドラインとして、指定したコマンドを実行するxargsを利用する。上の例の出力をさらにパイプでxargsに接続し、rpmを-qlオプション付きで実行させればいい。具体的には、

```
$ rpm -qa | egrep 'XFree86' | xargs rpm -ql
/usr/X11R6/bin/XF86_SVGA
/usr/X11R6/man/man1/XF86_SVGA.1x
    :
```

とする（**図2**）。

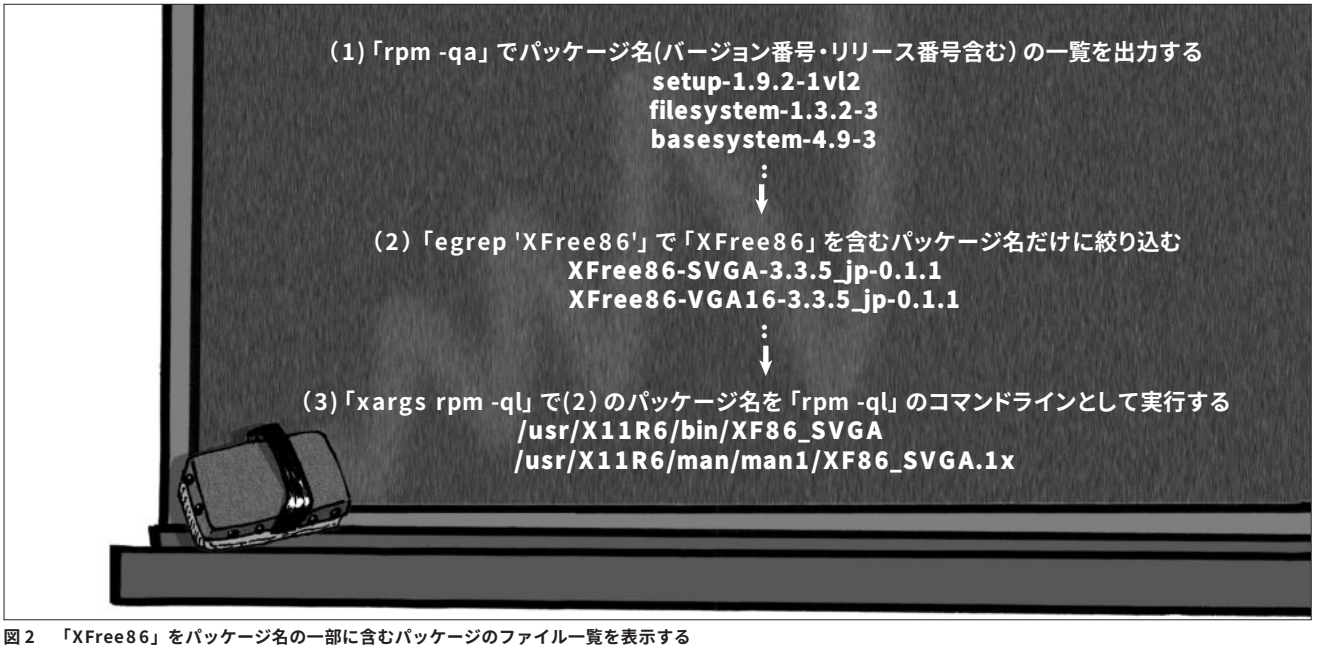


図2　「XFree86」をパッケージ名の一部に含むパッケージのファイル一覧を表示する

xargsは通常、標準入力から読み込んだすべての内容をスペースで区切ってコマンドラインに並べる（コマンドラインの最大長を超える場合は自動的に分割する）。rpmの場合、一度に複数のパッケージ名を受け入れるので、このままで大丈夫だ。具体的には、「rpm -ql XFree86-SVGA-3.3.5_jp-0.1.1 XFree86-VGA16-3.3.5_jp-0.1.1 …」といった具合に、パッケージ名をスペースで区切って指定した状態でrpmが実行される。

なお、この出力と、rpmを-qalオプション付きで実行して「XFree86」を検索した、

```
$ rpm -qal | egrep 'XFree86'
/usr/X11R6/lib/X11/XF86Setup/pics/XFree86.msk
/usr/X11R6/lib/X11/XF86Setup/pics/XFree86.xbm
      :
```

の出力は一致しないことに注意されたい。前者は「XFree86」をパッケージ名に含むパッケージの全ファイルを表示するのに対し、後者は全パッケージのファイルの中から、ファイル名の一部に「XFree86」を含むファイルだけを表示している。

一方、ファイル名を指定して、それがどのパッケージに含まれるものなのか調べることもできる。rpmに-fオプションを追加して、パッケージ名の代わりにファイル名を指定すればいい。たとえば、ユーザー名やパスワード（シャドウ化していない場合）、使用するシェルなどが列挙された「/etc/passwd」は、

```
$ rpm -qf /etc/passwd
setup-1.9.2-1vl2
```

と、setupパッケージに含まれることがわかる。

それでは、指定したコマンドが含まれるパッケージを知るにはどうすればよいだろうか。単独パッケージで配布されているコマンドなら、コマンド名を含むパッケージを探せばいい。しかし、lsやxargsなど基本的なコマンドの場合は、複数のコマンドが1つのパッケージ（fileutilsなど）にまとめられているため、パッケージ名とコマンド名は一致しないのだ。

ヒントは、指定したコマンドをフルパスで表示するwhichにある。たとえば、lsの場合は、

```
$ which ls
/bin/ls
```

となる。この出力を、-qfオプション付きのrpmのコマンドラインで指定すればいい。xargsを使うことも可能だが、whichの出力はたかだか1行なので、ここではコマンド置換を利用する。コマンド置換は、コマンドラインの一部を「`」で囲むと、その部分がサブシェルで実行された実行結果で置き換えられるというもの。詳しくは本連載の第1回めの記事を参照されたい（http://www.ascii.co.jp/linux/allascii/linuxmag/no_02.html）。たとえば、lsの含まれているパッケージを調べるなら、

```
$ rpm -qf `which ls`
fileutils-3.16_jp-9
```

とする。なお、bashでは、コマンド置換の記法として「$()」が追加されているので（コラム参照）、

```
$ rpm -qf $(which ls)
```

としても同じ結果が得られる。こちらのほうが始まりと終わりがはっきりわかって見やすい。

Column

コマンド置換に$()が用意されたわけ

コマンド置換に「`」（バッククォート）を使用する記法は、UNIXで最初に使われたシェル（Bourneシェル）からの伝統だ。bashでは、なぜわざわざ新しい記法「$()」を追加したのだろうか。理由は2つほど考えられる。1つめの理由は、本文中でも述べた「読みやすさ」だ。従来の記法では始まりと終わりの文字が同じ「`」なので、特に複数のコマンド置換がコマンドラインで使われているような場合、コマンド置換の内外が判別しにくい。この点、新しい記法は始まりと終わりをはっきり判別できる。もう1つの理由は、コマンド置換中でさらにコマンド置換を利用する「ネスト」を容易に記述できること。たとえば、lsのあるディレクトリに移動するには「cd $(dirname $(which ls))」とすればいい。従来の記法では「cd `dirname \`which ls\`` と、「\」（バックスラッシュ）を使う必要があり、しかもネストが深くなるにつれてバックスラッシュの数が1、3、5、7、15、…個とみるみる増えて面倒なことになる。

今月のお題

パッケージ名とファイルの対応リストを作る

今回のお題は、**インストール済みのパッケージ名（インストール番号・リリース番号含む）とそこに含まれているファイルを1対1に対応させたリストを作る**というもの。「パッケージ名::ファイル名」という書式で1行に1つずつ記述することにしよう。このリストが手元にあれば、指定した文字列をファイル名の一部に含むパッケージをegrepで探したり、データベースには登録されているのに実際には存在しないファイルを探すといった処理を簡単に行える。

●対応リストを作る方法（その1）

筆者が最初に考えた方法は、

(1) 全パッケージ名（インストール番号・リリース番号含む）のリストを1つずつシェル変数に取得
(2) 取得したパッケージの全ファイル名のリストを1つずつシェル変数に取得
(3) 2つのシェル変数の内容を「パッケージ名::ファイル名」という書式で表示する

というもの。それぞれの処理を個別に考えてみよう。

（1）や（2）のように、リストを1つずつシェル変数に取得するには、bashのfor制御構造を使用する。forは、繰り返し処理を行うために用意された制御構造で、コマンドラインでは「for ループ変数名 in リスト; do 繰り返す内容; done」という形で使われる。ループ変数と呼ばれる特殊なシェル変数に、リストに列挙された文字列が1つずつ設定され、「do」と「done」で囲まれた内容（複数のコマンドがあってもかまわない）が繰り返し実行される（シェル変数やforについての詳細は、本誌10月号の本記事を参照）。

たとえば、(1) の処理は、

```
$ for f in $(rpm -qa); do FOO; done
```

と書ける。-qaオプション付きのrpmの出力がコマンド置換でリストとなり、全パッケージ名（バージョン番号・リリース番号含む）の内容が1つずつシェル変数fに渡される。繰り返す内容は仮に「FOO」としている。

同様に、(2) の処理も、

```
$ for g in $(rpm -ql $f); do BAR; done
```

と書ける。今度は-qlオプション付きのrpmの出力がコマンド置換でリストとなり、(1) で得られたパッケージ名（$f）に含まれるファイル名が1つずつシェル変数gに渡される。繰り返す内容は仮に「BAR」としている。

最後の（3）の処理は、echoを利用して、

```
$ echo "$f::$g"
```

とすればいい。「"」で囲まれた内部でもシェル変数の置換は行われるので、シェル変数fの内容（パッケージ名）と、シェル変数gの内容（ファイル名）が「::」で区切られて表示される。

この3つを組み合わせて、(1) のコマンドラインの「FOO」に（2）のコマンドラインを、(2) のコマンドラインの「BAR」に（3）のコマンドラインをはめ込めば、方法（その1）の完成形が得られる。

```
$ for f in $(rpm -qa); do for g in $(rpm -ql $f);
do echo "$f::$g"; done; done
```

●方法その1の問題点を洗い出す

実際に上記のコマンドラインで対応リストを作ってみたところ、以下のような問題があった。

(a) ファイルを含まないパッケージ

システムに最初にインストールされる「basesystem」というパッケージは、ファイルがひとつも含まれていない特殊なもの（「rpm -ql basesystem」で確認可能）。このため、(2) の処理では、「ファイルを含んでいません」（日本語環境の場合）というメッセージが標準出力に出力され、ファイル名の代わりにシェル変数gに格納されてしまう。その結果、(3) の処理での出力も「basesystem-4.9-3::ファイルを含んでいません」となる。

(b) 規格外のバージョン番号・リリース番号

「RPMパッケージの基礎知識」で述べたように、バージ

ョン番号やリリース番号に数字以外の文字が含まれていてもよいが、区切り記号である「ー」だけは例外だ。しかし、ディストリビューション外で作成されたパッケージには、これらに「ー」を含むものが存在する。たとえば、「kinput2-v3-beta2-1」は、バージョン番号「v3-beta2」に「ー」を含む。本来なら「v3_beta2」か「v3beta2」とすべきところだ。

こうした規格外のバージョン番号・リリース番号を含むパッケージ名を-qlオプション付きのrpmに与えると、どの部分までがパッケージ名に相当するのか判断できず、標準エラー出力に「パッケージ kinput2-v3-beta2-1 はインストールされていません」（日本語環境の場合）というメッセージが出力され、標準出力には何も出力されない。このため、for内部の（3）の処理は実行されず、何も出力が得られないことになる。

（c）rpmが繰り返し起動される

システムにインストールされているパッケージ数は、「rpm -qa | wc -l」とすれば調べられる（VineやLASER5では450個程度、Turboでは600個程度だった）。上記のコマンドラインでは、-qlオプション付きのrpmが、これらの数だけ繰り返し起動・終了されることになる。CPUの速いマシンではそれほど問題にならないかもしれないが、rpmを一度だけ起動してすべての処理を行えるのなら、そのほうが処理が速くなるのは当然だろう。

●対応リストを作る方法（その2）

上記の3つの問題点を回避する方法を考えてみよう。一番重大な問題は、必要な情報（ファイル名のリスト）が得られない問題（b）だ。問題（a）は得られた情報を取り除けばいいし、問題（c）はなんなら無視してもかまわない。問題（b）を回避するには、正しいパッケージ名を知っているはずのデータベースに、すべてのパッケージ名とそこに含まれる全ファイル名を一緒に出力させればいい。

rpmには、パッケージ名などの表示形式を規定する、--queryformatオプションが用意されている。このオプションに続けて書式文字列を指定すると、書式に基づいて情報が表示される。書式文字列の初期設定は「"%{NAME}-%{VERSION}-%{RELEASE}\n"」だ。「%{NAME}」（パッケージ名）、「%{VERSION}」（バージョン番号）、「%{RELEASE}」（リリース番号）を「ー」区切りで表示し、最後に改行する。たとえば、バージョン番号やリリース番号を含まないパッケージ名を表示するには、

```
$ rpm -qa --queryformat "%{NAME}\n"
```

とすればいい。

ところで、--queryformatには副作用がある。パッケージ名以外の情報を表示させる-lや-Rオプションを追加しても、依然として指定した書式でパッケージ名を表示するのだ。つまり、-qalオプションと--queryformatオプションを同時に指定し、書式文字列として初期設定の内容をそのまま指定すれば、

```
$ rpm -qal --queryformat "%{NAME}-%{VERSION}-%{RELEASE}\n"
setup-1.9.2-1vl2
/etc/csh.cshrc
/etc/exports
    :
```

のように、すべてのパッケージについて、パッケージ名とそれに含まれる全ファイル名が順番に出力される。ということは、rpmを一度だけ起動してすべての処理を行えるわけだから、問題（c）もついでに解決だ。

あとは、パッケージ名（バージョン・リリース番号含む）とファイル名が混在した出力を読み込んで、どちらであるか判別し、「パッケージ名::ファイル名」という書式で表示する処理を行えばいい。AWKを使うとこの部分は以下のように書ける（Perlだといくぶん長くなる）。

```
$ gawk '{if (/^[^/]/) name=$0; else print name "::" $0}'
```

ここでは、入力行が「/」で始まらなければパッケージ名とみなして変数nameにその内容（$0）を格納する。そうでなければファイル名とみなして、それ以前にnameに格納したパッケージ名と、現在の入力行の内容（$0）を「::」で接続して表示する。なお、この判別方法では、問題（a）のメッセージ「ファイルを含んでいません」は「/」で始まらないので出力されない。厳密に言えばパッケージ名と見なされて変数nameに格納されてしまうが、すぐに次のパッケージ名が入力されて変数nameを上書きするので、出力には何の影響も与えないのだ。これで問題（a）も解決できたことになる。

よって、上記2つのコマンドラインを組み合わせれば、問題（a）～（c）をすべて解決したコマンドラインが完成す

る。出力を適当なファイル（たとえばrpmfiles）にリダイレクトして保存すれば、egrepなどで素早く検索できる。これを今回の解答としよう。

```
$ rpm -qal --queryformat "%{NAME}-%{VERSION}-%{RELEASE}\n" | gawk '{if (/^[^/]/) name=$0; else print name "::" $0}' > rpmfiles
```

さらに、本連載の先月号の記事で紹介したcronシステムを利用すれば、毎日4時など決められた時間に上記のコマンドラインを起動して、rpmfilesの内容を常に最新のものに保つことも可能だ。

●存在しないファイルを探す

応用例として、「RPMのデータベースにはあるのに、実際には存在しないファイル」を探してみよう。この処理を行うには、ファイルの存在チェックの方法と、チェック結果によって異なる処理をする条件判断の方法を理解する必要がある。どちらも、bashに内蔵されている制御構造（forと同類）を利用することで実現可能だ。

ファイルの存在チェックには、[ ]制御構造を利用する。ファイル（ディレクトリ、デバイスファイル、シンボリックリンクでもよい）が存在するかどうか調べるには、-eオプションとファイル名を指定して「[-e ファイル名]」とする（「[」の後と「]」の前にはスペースが必要）。逆に、ファイルが存在しないことを調べたい場合には、-eオプションの前に「!」を挿入して、「[! -e ファイル名]」とすればいい。

条件判断にはif制御構造を利用する。コマンドラインでの書式は「if 条件; then 処理; fi」だ。条件部には、通常のコマンドや[ ]制御構造を指定でき、コマンドが正常に終了したり、[ ]制御構造でのチェックが成功した場合に処理部の内容が実行される。たとえば、

```
$ if [ ! -e hoge ]; then echo "Not exist"; fi
```

とすると、hogeが存在しない場合に限って「Not exist」という文字列が出力されるわけだ。

さらに、[]制御構造で複雑なチェックを行ったり、if制御構文で複数の条件で場合分けして異なる処理を行うことも可能だ。それらについては次回以降で説明することにして、RPMのデータベースにあるファイルの存在チェックの問題に戻ろう。

データベースに含まれる全ファイルを出力するには、rpmを-qalオプション付きで実行すればいい。出力をひとつひとつ取り出せれば、上記のコマンドラインを使って存在しないファイルをチェックできる。これは、「対応リストを作る方法（その1）」にすでに登場したfor制御構造とコマンド置換を組み合わせる方法で実現できる。

```
$ for f in $(rpm -qal); do if [ ! -e $f ]; then echo "Not exist $f"; fi; done
Not exist ファイルを含んでいません
Not exist /etc/rc.d/rc3.d/S85httpd
      :
```

とすると、存在しないファイルのリストが表示される（先頭の「ファイルを含んでいません」は、問題（a）のメッセージ）。なお、一般ユーザーではアクセスできないファイルもあるので、上記や以下のコマンドラインはスーパーユーザ（root）になって実行するほうがいいだろう。

さらに、検出されたファイルがどのパッケージに含まれるのか表示するには、echoの表示内容を「"Not exist $(rpm -qf $f)::$f"」に変えればいい。しかし、これでは対応リストのときと同じ問題（a）や（c）が解決されないまま残ってしまう。

先ほど作成した対応リストrpmfilesを利用すると、すっきりとした解決方法が見つかる。基本構造は上のコマンドラインと同じだ。

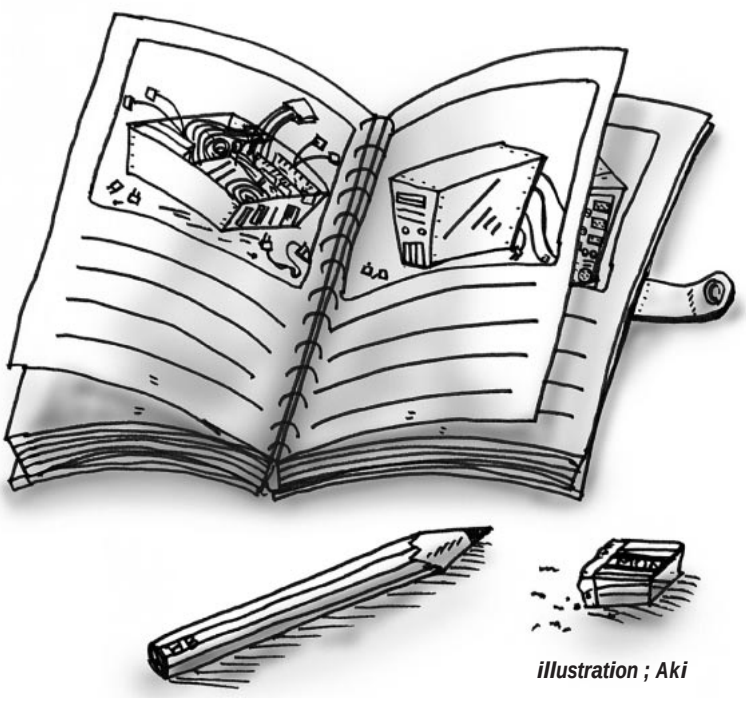
```
$ for f in $(cat rpmfiles); do if [ ! -e ${f#*::} ]; then echo "Not exist $f"; fi; done
Not exist apache-1.3.3-1::/etc/rc.d/rc3.d/S85httpd
      :
```

まず、コマンド置換を使ってrpmfilesの内容をリストとして、forでひとつひとつシェル変数fに取り出している。rpmfilesの各行は「パッケージ名::ファイル名」という書式なので、このまま存在チェックを行うわけにはいかない。そこで、fの内容を参照する際に、bash Ver.2以降で導入されたパターン照合演算子「#」を使って「${f#*::}」とし、「パッケージ名::」の部分を取り除いている（パターン照合演算子については本連載の10月号の記事を参照されたい）。なお、上記2つのコマンドラインでは、ファイル名にスペースが含まれると問題が起きる。腕に自身のある人は対処方法を考えてみよう（ヒント:組み込みシェル変数「IFS」）。

Linux日記

第4回　プロセスの生成から終了まで

Linuxで動作するプログラムやデーモンはプロセスという単位で実行されます。では、そのプロセスはどのようにして生まれ、そして消えてゆくのでしょうか。今回は、それを説明します。



文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

前回psの話を取り上げた。せっかくなので関連する話を続けようと思う。というわけで、今回はUNIXのプロセスの話だ。

■プロセスの状態

前回も少し触れたが、psコマンドの出力のSTAT表示、つまりプロセスの状態について考えてみよう。UNIXに限らず、マルチタスクオペレーティングシステムでは、並行して動作する複数のプログラムは、プロセスという単位で識別される。各プロセスはさまざまな状態を遷移しながら、自身の処理を実行していく。プロセスの状態としては、おおよそ以下のようなものが考えられる。

- **プロセスの誕生**
- **プロセスの実行**
- **何らかの停止状態**
- **プロセスの終了**

これらの状態について見ていこう。また、もっとも日常的なプロセスの生成、すなわちシェルからのコマンドの実行について見てみよう。

■プロセスの生成

まずはプロセスの誕生である。オペレーティングシステムは何らかの方法で、独立したインスタンスである新しいプロセスを作成することができる。たとえばUNIXでシェルからコマンドを起動するのであれば、そのコマンドを実行するプロセスを作る必要がある。もちろん、シェルのコンテキストでコマンドを実行するという選択肢もあるが、普通は、別のプロセスとしたほうが何かと便利であるし、UNIXの各コマンドも、そのような構造で作られている。たとえば、コマンドのバックグラウンド実行は、別プロセスであれば簡単だ。また、異なるプロセスであることにより、それを起動したシェルのことを何も考えずに、各コマンドが自由に自分の環境をいじくり回すことができる。たとえば、コマンドが標準入力をクローズしてしまったらどうなるだろう？　別プロセスであれば、標準入出力といったファイル環境はプロセス固有のものである。コマンドが標準入力をクローズしたところで、シェルは痛くもかゆくもない。これが同一プロセスだと、シェルは以後標準入力を読み込めなくなってしまう。

UNIXでは、まっさらなプロセスを新規に作成するための手段は提供されていない。意外に思うかもしれないが、これは事実である。もちろん、新しいプログラムを実行するための手段は提供されている。execというシステムコールを使えば、指定したプログラムファイルをロードし、それを実行することができる。しかし、execを使った場合、それを呼び出した側のプログラムは消滅してしまう。プロセスは継続するが、プロセスの実体が新しいプログラムに置き換えられてしまうのである。たとえばシェルからlsコマンドをexecを使って起動すると、execを呼び出した時点でシェルのプログラムイメージ

は消滅し、代わりにlsのプログラムイメージがロードされ、実行される。lsが終了したとき、次のユーザー入力を待つシェルはもはや存在しない。これでは対話シェルとしては使えない。どうにかして、シェルを残したままlsを実行する必要がある。つまり、シェルのコンテキストではなく、新しいプロセスとしてlsを実行したいのである。

LinuxやUNIXでは、forkというシステムコールを使って新しいプロセスを生成する。forkは、プロセスのコピーを作成する。つまり、forkを呼び出した側とまったく同じ内容のプロセスが作成されるのである。異なっているのはプロセスのPIDと、forkが返す値だけである**（図1）**。

あるプロセスがforkを呼び出すと、システムは新しいプロセスを作成し、forkを呼び出したプロセスの内容を新しいプロセスにコピーする。コード部は共用され、PIDは異なる値になるが、データ、スタック、環境変数やファイルハンドルなどはすべて新しいプロセスにコピーされる。これによってどうなるか？

forkを呼び出したプロセス（これを親プロセスと呼ぶ）は、forkの処理が完了した時点で、forkの後に続くコードの実行を再開する。これは何の不思議もないだろう。新しく作成されたプロセス（これを子プロセスと呼ぶ）はどのように処理を開始するのか？　実は親と同じである。forkがリターンしたところから実行を開始するのである。呼び出してもいないシステムコールからリターンするというのは不思議に思えるかもしれないが、すべてのデータ（特にスタック）、環境がコピーされているということは、新しいプロセスはforkの呼び出し時の状態を維持しているということである。この状態からリターンすることは不可能ではない。というか、この状態で可能なのは、forkからリターンすることだけなのである。

つまり、forkは入り口が1つで出口が2つあるのだ。あるプロセスがforkを呼び出すと、そのforkは2つのプロセスにリターンするのである。1つは呼び出したプロセス、もう1つは新しいプロセスである。2つのプロセスは、forkの返り値を調べることで、自身が呼び出し側なのか、新規作成側なのかを知ることができる。これにより、親プロセスと子プロセスがfork後に違う処理を行うことができる。UNIXは一番おおもとになるinitプロセスからforkを繰り返し、各ユーザーのシェルやシェルから起動されるコマンド、あるいはバックグラウンドで動作するデーモンなどのプロセスを生成しているのである。

では、一番初めのinitプロセスは何から誕生するのか？　これだけは例外的なもので、ブート処理によってカーネルがロードされた後、カーネルがまったく新規に作成するのである。これればかりは親がない。カーネルがゼロから作るしかないのだ。

■シェルからコマンドを起動する

さて、シェルからコマンドを実行する話に戻ろう。コマンドラインで、空白やタブで区切っていくつかのワードをタイプすると、シェルは最初のワードをコマンド名、以後のワードをそのコマンドに与えるオプションやアーギュメントとみなす。そしてそのコマンド名と同じ実行ファイルを探す。コマンドの実行に先立って、シェルはまず自身をforkする。以後、親プロセスと子プロセスは異なる処理を行う。

親プロセスは、コマンド終了後に再びシェルとして機能するために、子プロセスが終了するのを待つ。&が指定されていれば、終了を待たずに、次のユーザー入力を待つ。子プロセスは、指定された外部コマンドを実行するために、execシステムコールを呼び出す。これにより、子プロセスのシェルイメージは消滅し、代わりに実行コマンド

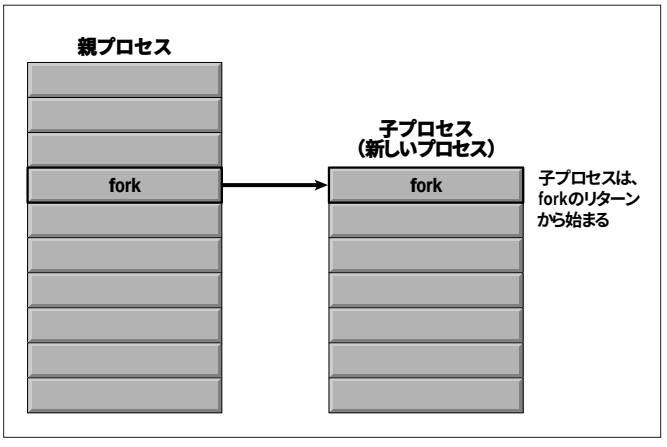


図1　forkの実行

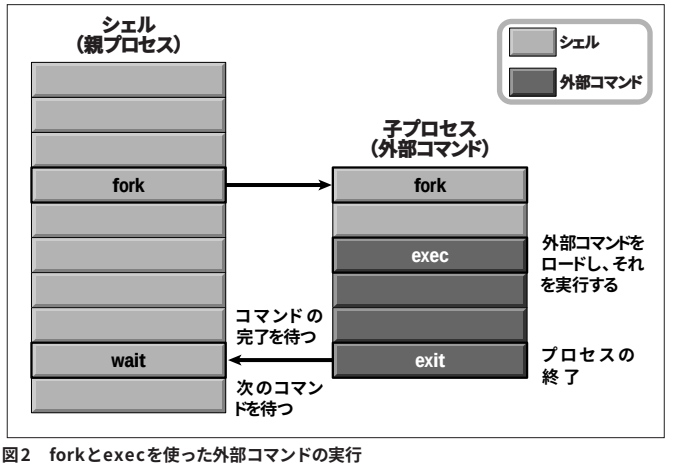


図2　forkとexecを使った外部コマンドの実行

のイメージがロードされ、そのコマンドを実行する（**図2**）。コマンドが終了すると、子プロセスは消滅する。子プロセスの終了は、親プロセスによって検出され、（バックグラウンド実行でない場合は）親プロセスは再び実行を開始する。つまり、プロンプトを表示し、次のコマンドを待つのである。

fork後のexecにより、シェルのコードやデータは失われるが、環境変数やファイルハンドルなどは保存されるという点が重要だ。コマンドが標準入出力を処理できるのは、シェルから標準入出力を継承しているからである。

標準入出力のリダイレクトや複数のコマンドの入出力をパイプで結合するというのは、コマンド側ではなく、シェル側の機能である。

■プロセスの終了

処理を終えたプログラムは、exitというシステムコールを使って終了する。このシステムコールにより、プロセスは消滅する。そして、終了したプロセスが使っていたメモリやファイルなどのリソースは解放される。

forkを呼び出し、新しいプロセスを生成した親プロセスは、子プロセスの終了を検出することができる。waitというシステムコールを実行すると、子プロセスの終了を待つことができる。シェルなどは、この機能を使って、実行したコマンド（子プロセス）が完了するのを待っている。親プロセスは、子プロセスの終了を待たなければならないわけではない。ほかにやることがあればそれを行ってもいいし、子より先に終了してしまっても構わない。親が先に終了した場合、ゾンビプロセスの後始末を行うのは、親の祖先にあたるプロセスである。最終的には、一般プロセスすべての先祖であるinitが処理することになる。

プロセスはexitの実行により、その処理を終了するが、終了コードなど、システム中にはまだそのプロセスの情報が残っている。親プロセスがこの後始末を行うことで、プロセスは完全に消滅する。親が先に終了してしまった場合、あるいは親が忙しくてすぐにこの後始末を行えない場合など、この終了処理待ちのプロセスがpsなどで表示されることがある。プロセスは終了したものの、プロセスの情報がまだシステムに残っている状態のプロセスをゾンビプロセスと呼ぶ。このようなプロセスは、psのSTATでZで表示される。

■シェルのexecとfork

シェルの組み込みコマンド、つまりコードがシェル中にあり、外部から実行ファイルをロードすることなく実行できるものに、execというコマンドがある。ここまでの解説を読めば、どのようなことをするコマンドかは見当が付くだろう。そう、forkせずに実行ファイルをexecするのである。たとえばexec lsとタイプすると、シェルが直接execを呼び出してlsの実行を開始する。もはやシェルは残っていない。lsが終了した時点で、そのセッションは終了し、ログアウトしてしまう（シェルの終了でログアウトするということを思い出そう）（**図3**）。

通常の対話セッションでexecを使うことはほとんどないだろうが、複雑なシェルスクリプトを記述する時などは、便利に使えることもあるだろう。

通常のコマンド実行では、シェルはforkして、実行ファイルをexecする。また、シェルのexecコマンドを使えば、forkせずに実行ファイルをexecする。

Column

ゾンビ

ゾンビ（Zombie）というのは、ブードゥー教の魔術により蘇った死体のことである。ゾンビが有名になったのは、映画「ゾンビ」（原題はNight of the living dead、ジョージ, A, ロメロ監督）だろう。ブードゥー教のゾンビの生態について筆者は知らないが、ロメロ監督のゾンビは、生きている人間を襲ってその肉を食っていた（ロメロ監督のゾンビは、宇宙からの怪光線で蘇った死体が第一世代である）。そしてゾンビに食い殺された人間はゾンビになってしまうのである。日本では、ゲゲゲの鬼太郎に亡霊（もうりょうと読む。違う漢字だったかもしれない）という死体が蘇る妖怪が紹介されている。お通夜の席でほうきを逆さに立てておくと死体が蘇るのを防ぐ効果があるらしいが、UNIXマシンの横にほうきを逆さに立てててもゾンビの発生を防ぐ役には立たないだろう。

UNIXのゾンビは、幸いなことに生きているプロセスを食い殺すこともなく、単に消滅を待つだけである。意味から考えると、この状態はゾンビではなく埋葬待ちの死体に相当するものだが、死体を意味するDead bodyだと、略語がDになってしまい、割り込み不可の停止プロセスと区別がつかなくなってしまうから、あえてゾンビにしたのかもしれない。あるいは単にアルファベットの最後の文字を割り当てたのを、誰かが開始、あえてゾンビと呼んだのかもしれない。その辺の経緯はわからないが、まぁ、こういう用語が正式な技術用語として存在しているというのがUNIXらしいところだ。

差し当たって、UNIXのゾンビは、プロセステーブルのエントリを占めるという以外は、さほど迷惑な存在ではない（いつまでも消えないとしたら何らかの異常が発生していることになるが）。ゾンビ状態のプロセスが蘇って、生きているプロセスを食い殺して回るなんていうウィルスソフトでもあれば、ゾンビの恐怖をUNIXのユーザーや管理者も味わうことになるかもしれない。

forkとexecの組み合わせという点でみると、forkだけ行うというものが残る。実は、シェルはこのパターンもサポートしている。

シェルだけforkして何の役に立つんだと思うかもしれないが、これをやりたい状況というのがあるのだ。もちろん、forkするだけでなく、forkしたシェルからさらにforkしてコマンドを実行するのであるが。

あるディレクトリで実行したコマンドの標準出力を、別のディレクトリで動くコマンドの標準入力に接続したいといった場合を考えてみよう。たとえば、あるディレクトリにcdして、そこでtarを使ってファイルをアーカイブし、それを標準出力に送り、パイプでつないだ別のtarで別のディレクトリに展開するといった作業はどうだろうか。カレントディレクトリ（正確にはカレントワーキングディレクトリ）は、プロセスが持つ属性の1つであり、システムコールで変更できる。シェルであれば、cdコマンドでカレントディレクトリを変更できる。さて、tarの問題を考えてみよう。/fooから/barに階層構造を維持したまま、tarを使ってファイルをコピーできるだろうか（もちろん、絶対パスを指定したり、cpなどを使うこともできるが、ここではそれは置いておこう）。カレントディレクトリは/barとする。

「cd /foo; tar -cf - . ¦ tar -xf -」という形式は使えない。「tar -cf - .」は、カレントディレクトリ以下のファイルをtar形式でアーカイブし、それを標準出力に送る。「tar -xf -」は、標準入力をtar形式として解釈して、それを展開するというものだ。最初のcdで、送り元のディレクトリに移動しているので、最初のtarは目的のファイルをアーカイブして標準出力に送ることができる。しかし、2番目のtarは、/barの下にファイルを展開できない。最初のtarはファイルの位置を相対パスで指定する。たとえば/foo/doc/bugs.txtは、./doc/bugs.txtになる。これを正しく展開するためには、展開側のtarのカレントディレクトリは/barでなければならないが、最初のcdで/fooに移ってしまっている。

最初のtarはカレントディレクトリが/fooの状態で実行しなければならないが、2番目のtarは/barで実行しなければならない。しかしシェルのカレントディレクトリは1つだけである。どうすればいいか？

答えは、「(cd /foo; tar -cf - .) ¦ tar -xf -」である。cdと最初のtarが()で囲まれている。()は、シェルをforkしたのち、子プロセスのシェルでカッコ内のコマンドを実行することを指示する。つまり、カッコに囲まれたcdとtarは、コマンドをタイプしたシェルがforkしてから実行されるのである。forkすることによって、新しいプロセスとして動作するシェルが作られ、それがcdとtarを実行する。つまり最初のtarは、/fooをカレントディレクトリとするシェルから実行されるのだ。そしてその結果は標準出力に送られる。2番目のtarは、ユーザーが直接使っているシェルから実行される。つまりカレントディレクトリは/barのままである。これにより、tar形式で指定された相対パスをカレントディレクトリに適用して、/fooの下のファイルを同じディレクトリ構成のままで/barの下に展開できるのである。

このように、本来のシェルからforkしたシェルをサブシェルと呼ぶ。サブシェルにコマンドを実行させることにより、元のシェルの環境（カレントディレクトリや環境変数など）に影響することなく、一時的にシェルの環境を変更してコマンドを実行することができる**（図4）**。

これはまた、もう1つの興味深いことを提示している。cdコマンドなど、プロセス自体の属性を操作するコマンド

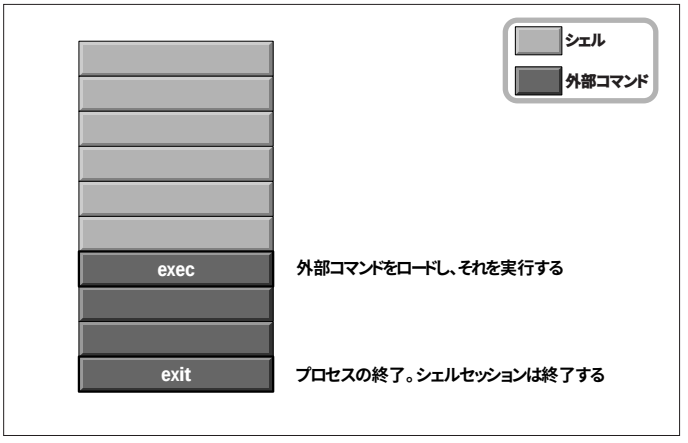


図3　シェルからのexec

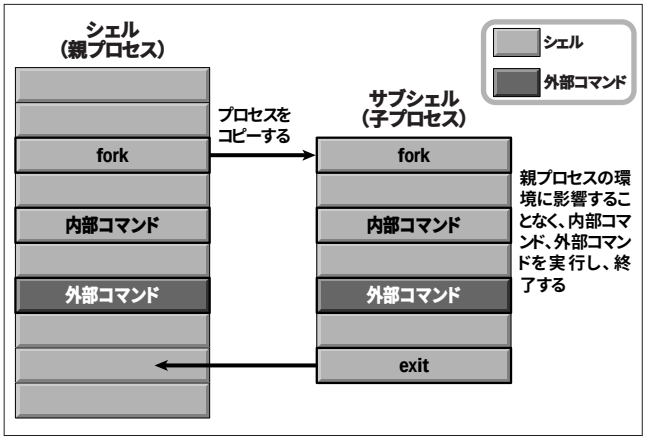


図4　サブシェル

は外部コマンドにできないということである。カレントディレクトリを変更するcdコマンドが外部の実行ファイルだとどうなるか？　シェルがforkしてcdを実行する。新しいプロセスはcdの引数で指定されたディレクトリをカレントディレクトリとするが、親プロセスのシェルのカレントディレクトリは変化しない。cdの終了後、元のシェルのカレントディレクトリは変化しないのである。

このような理由から、シェルそのものの環境を変更するいくつかのコマンドは、シェルプロセスのコンテキストで実行される組み込みコマンドでなければならない。カレントディレクトリの変更以外に、環境変数の設定コマンドなどがこれに相当する。

これに関連する話題はほかにもある。一連のシェルコマンドや外部コマンドのコマンドラインをファイルに記述し、実行属性を設定することで、シェルスクリプトを作成することができる。プログラムファイルと同じように、ファイル名をタイプすることにより実行できるファイルだ。これは外部コマンドの実行と同じように、forkしたシェルから実行される。正確には、シェルがforkした後、外部コマンドとしてシェルを起動し、その引数にファイル名を与えるのである（**図5**）。

シェルスクリプトに似たファイルに、ログイン時の初期化を行う.profile、.login、.cshrcなどのファイルがある。これらのファイルは、シェルの各種初期設定を行うコマンドが列挙されている。内容的には実行可能なシェルスクリプトと同じだが、これらをシェルファイルのように実行しても意味はない。シェルファイル名をコマンドラインで指定して実行する場合、fork／execされるプログラムはシェルである。このシェルはファイルの内容を問題なく実行して終了する。しかし、その結果は親のシェルには反映されない。カレントディレクトリの移動、各種環境変数やシェル変数の設定などは、子プロセス側で行われるだけで、親プロセスには影響しないからだ。シェルの設定のためにこれらのファイルを使う場合は、forkすることなくこれらを実行しなければならない。そのため、sh系では「.」、csh系ではsourceという組み込みコマンドが用意されている。たとえば「.profile」とタイプすれば、ログイン時と同じようにシェルの初期設定を行うことができる。これらのコマンドは、指定されたファイルに記述された内容を自身のコンテキストで、つまりforkせずに実行する。これにより、ファイルに記述された各種環境変更などがそのプロセスに反映される（**図6**）。

■プロセスの実行可能状態と実行状態

何も障害がなければ、プロセスは自身の実行を継続する。何らかの理由により実行を継続できなければ、プロセスは停止する。プロセスがどのような条件の時に実行を継続できるかを考えてみよう。

プロセスがディスクI/Oやキー入力、その他の何らかのイベントを待っていない状態であれば、プロセスはいつでも実行することができる。これを実行可能状態（Runnable）という。そして、実行可能状態のプロセスに実際にCPUが割り当てられると、プロセスの実行が開始される。これが実行状態（Running）である。

実行中のプロセスの数は、システムが備えているCPUの数より多くなることはない。シングルプロセッサシステムなら1つだけだし、4プロセッサシステムなら最大4つである。それに対し

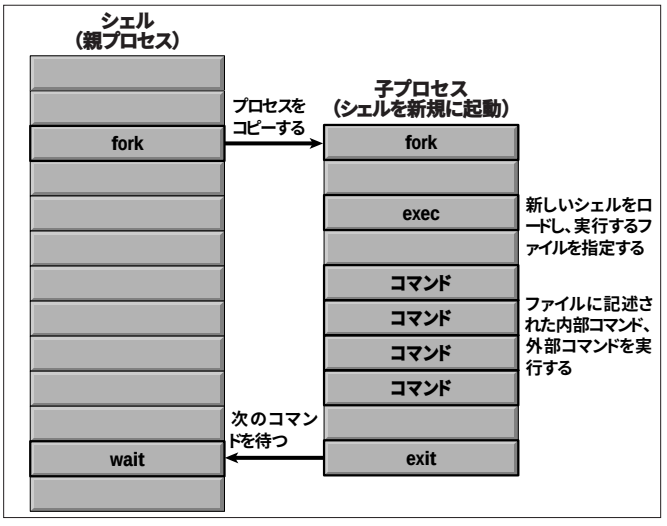


図5　シェルスクリプト

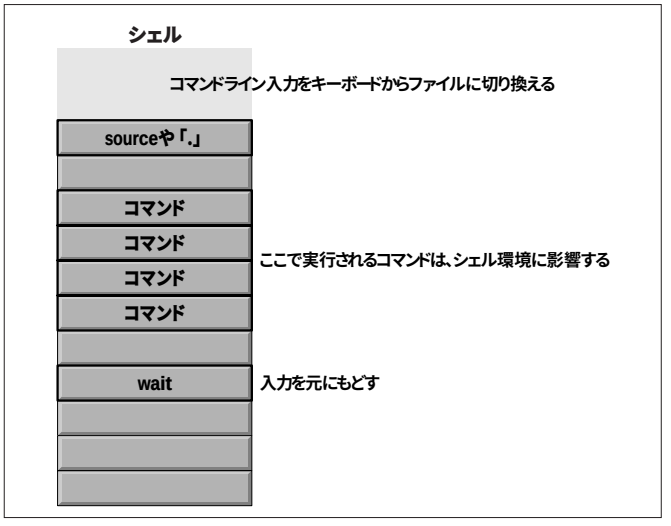


図6　シェル初期化ファイルの実行

て、実行可能状態プロセスの数に制限はない。正確にいえば、実行可能プロセスの中から1つ（CPUが1つの場合）を選んで、それにCPUを割り当てるのである。

実行可能状態のプロセスは、オペレーティングシステムが管理している実行待ちキューの中で待機している。オペレーティングシステムのスケジューラは、各プロセスのプライオリティ（優先順位）や消費したCPUタイムなどを加味した上で、次に実行する、つまり実際にCPUを割り当てるプロセスを決定する。これにより、プロセスの状態が実行可能から実行中になるのである。実行を開始したプロセスは、CPUを一定の時間（タイムスライス）使うか、後で説明する何らかの待ち状態に入ると実行を停止し、スケジューラは別の実行可能プロセスにCPUを割り当てる。

forkにより誕生したプロセスは、停止したり、実行可能になって実際にCPU割り当てを受けて実行するというサイクルを無数に繰り返して、自分の仕事を進めていくのだ。

Column

デーモンの起動

一般的なコマンドは、forkしたシェルからexecするという形で起動するのだが、デーモンの多くは、これとはまた少し異なる形で起動する。デーモンは、一度起動されると、その後ずっと動き続ける。つまり、普通にシェルから起動すると（ブート時に実行される初期化ファイルはシェルスクリプトなので、シェルから起動されることに変わりはない）、いつまでもシェルに戻ってこなくなってしまう。しかし実際は、&を付けずに実行しても、すぐにシェルに返ってくる。もちろん、バックグラウンドでデーモンプログラムは動作している。どうなっているのだろうか？

答えは簡単だ。デーモンは、通常のコマンドと同じように起動された後、forkしているのである。そして、子プロセスがデーモンとして実行を続け、親プロセスはすぐに終了してしまうのである（図7）。親プロセスが終了することで、シェルは処理を続行できるのだ。これにより、子プロセスのデーモンを生成した直接の親プロセスはなくなってしまい、祖父であったシェルが親になる。そのシェルも初期化処理が終わるとなくなってしまうので、デーモンの親は最終的にinitプロセスになる。

また、デーモンの多くは標準入出力を使わないので、この起動処理の段階で標準入出力をクローズしてしまう。これにより、psでTTYが表示されなくなるのだ。

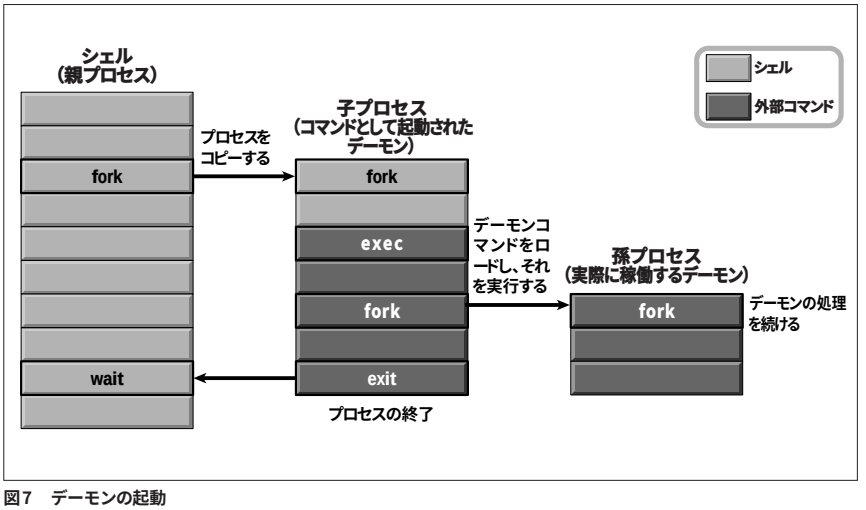


図7　デーモンの起動

■省エネ版のfork

forkは、新しいプロセスを複製するが、実際使われるときには、新しいプロセスはすぐにexecを実行することが多い。環境変数などは継承されるが、元のプログラムのデータは使われることなく失われてしまうのである。

かつてのUNIXは、forkで実際にイメージをまるごとコピーしていたのだが、使われもしないデータをコピーするのは無駄なので、BSDで、イメージを親子で共有するvforkというシステムコールが導入された。Linuxのforkもこの流れを汲むもだ。しかしこの方法では、子プロセスがメモリを書き変えると、親プロセスに影響してしまう。これを防ぐために、Linuxの子プロセスは「コピーオンライト」という処理を行う。書き込みを検出すると、そのページだけ新たに割り当て、コピーしてから実際の書き込みを行うのである（これは、MMUと仮想記憶の仕組みで実現される）。これにより親プロセスには影響を与えずに、イメージをコピーするのと同じ効果がより効率的に得られるのだ。

■なぜforkなのか

最後にちょっとヨタ話をしておこう。なぜ新しいプロセスを作るシステムコールはforkなのか？

このforkは、ナイフとフォークのフォークである。しかし、食器のフォークといえば、先の数はもっと多いのが普通だ。そこで、forkを辞書で調べると、熊手、音叉という訳もある。また、分岐点、分岐するという意味もある。

意味からすると、熊手や音叉よりは、分岐点、分岐するのほうが的確な訳であろうが、それでも筆者は、やはりforkは、先が分かれてるからフォークなんだと思っている。

Webサーバ構築術（第5回）

1人でWebサーバを使っている限り、ややこしいユーザー管理は不要だ。しかし、会社や学校にWebサーバがあり、いろいろな利用者にエリアを開放していくと、ユーザー管理とエリアの確保、パーミッションの変更など、面倒な作業が出てくる。また、万一のことを考えると、ドキュメントルートは直接利用者に触らせたくないものだ。今回は、そうした用途でも使えるエイリアス、リダイレクトの話をしよう。

ドキュメントルートを触らせないファイル管理 ～alias、redirect～

文：中島昌彦
Text : Masahiko Nakajima

ユーザーにドキュメントルートを開放したくない

Webサーバを構築して、部署単位、ユーザー単位にWebコンテンツをアップロードできるような仕組みを作ったとする。このとき、Apacheのドキュメントディレクトリでは、各利用者に対して、ディレクトリパーティションを与えることになる。ユーザーに対するグループ管理もややこしくなり、結果として管理が繁雑になる。

手間をかけずに実現するには、各ユーザーのホームディレクトリ上のpublic_htmlを使うという方法だ。しかし、この方法ではいくつも問題が生じる。まず、URLが~ユーザー名となり、かっこ悪い。管理側としてはもっとも楽な方法だが、利用者側にとってみればこのURLはうれしくない。

ユーザー名が分かるということは、FTPやtelnetでアタックされる可能性もある。Webサーバの管理側としては、外にはできる限りユーザー名をオープンにしたくないという本音がある。また、ユーザー側にしてみれば、~を使わずにスマートなURLでWebコンテンツを公開したいという要望も強い。そこで、こうした両者のコンセプトを生かしたまま実現する手順を3つほど挙げよう。

シンボリックリンクを有効にする

もっとも手軽な方法は、シンボリックリンクを張る方法だ。たとえば、ユーザーアカウントnakaのホームディレクトリ下にpublic_htmlというディレクトリがあるとする。その中にnakaがWebコンテンツを置いていれば、http://www.ho.point5.co.jp/~naka/で個人のWebコンテンツへアクセスできる。

しかし、これではユーザー名が見えてしまうために、セキュリティ面で不安が残る。そこで、Apacheのドキュメントディレクトリ/home/httpd/htmlと~naka/public_html間でシンボリックリンクを張る（**図1**）。

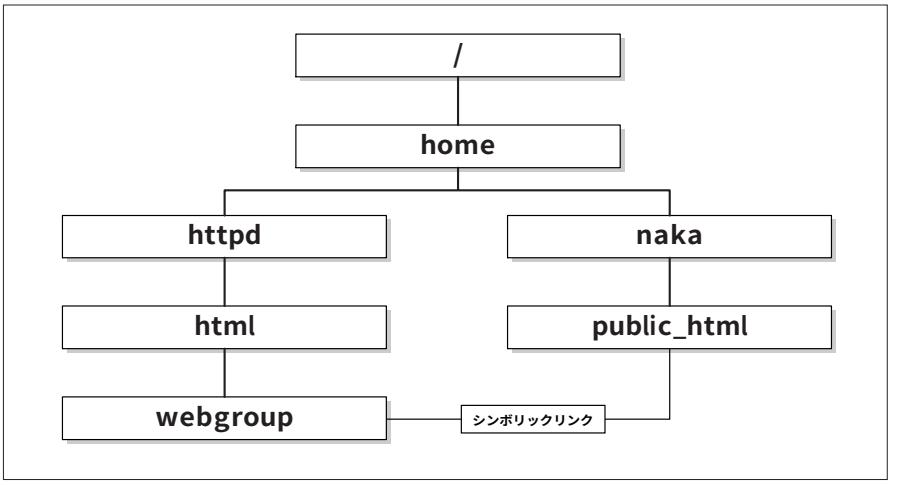


図1　実ディレクトリとシンボリックリンクの関係

```
# ln -s /home/naka/public_html
/home/httpd/html/webgroup
```

ファイルシステム的に、シンボリックリンクを張ってしまえば、ファイル管理も簡単にできるうえに、http://www.ho.point5.co.jp/webgroup/というURLを使って、/home/naka/public_html内へのアクセスが実現する。ただし、シンボリックリンクだけで動くわけではなく、Apacheの設定でシンボリックリンクを有効にしておく必要がある。シンボリックリンクを有効にするには、/etc/httpd/conf/access.conf中でOptionsの行を、

```
Options Indexes Includes
            FollowSymLinks ExecCGI
```

というように、Optionsの中でFollowSymlinksを含めておく（**リスト1**）。ここで重要なことは、Options内でFollowSymLinksを有効にしておくことで、

```
Option All
```

という形であってもかまわない。いずれにしても、FollowSymlinksさえ有効にしておけば、Apacheはシンボリックリンクをちゃんと処理してくれる。よく設定内容が分からないときは、Options Allとして、すべてのオプションを有効にしてしまう方法があるが、Apacheのデフォルト状態では、上のようにFollowSymlinksが有効になっている。標準設定のまま利用している限りなら、シンボリックリンクは正しく動作するはずだ。

/etc/httpd/conf/access.confを書き換える以外にも、シンボリックリンクを有効にしたいディレクトリに、.htaccessを置いておく方法もある。たとえば、

```
Options +FollowSymLinks
```

といった内容をドキュメントディレクトリの .htaccess中に加えておけば、そのディレクトリ以下でFollowSymlinksが有効になる。ドキュメントディレクトリ全体に対して、FollowSymlinksを有効にしたくないときには、この方法を使うほうが効果がある。

「~ ユーザー名」でアクセスさせずに、ユーザー名を隠したままURLを与えつつ、セキュリティ周りも守るとすると、この方法がもっとも手軽な方法だ。しかし、この方法をとらないことも多い。理由は単純で、access.conf中でFollowSymLinksを許すと、Webサーバ全体でFollowSymLinksを許すことになる。もしユーザーが勝手に別のディレクトリにシンボリックリンクを張っていた場合、そのシンボリックリンクも有効になる。万一/etc/にシンボリックリンクを張られた場合、/etc/中の設定内容がすべて見えてしまう。

.htaccessで指定ディレクトリに対してのみ有効にする方法であれば、ドキュメントディレクトリ全体に及ぶ危険性は減る。しかし、今回のケースのように、ドキュメントディレクトリのトップにシンボリックリンクを張る場合、.htaccessはドキュメントディレクトリのトップに置いておかなければいけない。

すなわち、ドキュメントディレクトリ全体にFollowSymLinksを許していることと同じになる。ドキュメントルート以下ではFollowSymlinksを設定しないとするならば、各ディレクトリに.htaccessを置いて回るか、/etc/httpd/conf/access.conf中の<Directory>ですべてのディレクトリを設定して回らないといけない。つまり、ドキュメントディレクトリ以外でFollowSymLinksが利用できる状態を許したくないならば、別の方法をとったほうがスマートだ。

▼リスト1　/etc/httpd/conf/access.conf中のシンボリックリンクを有効にする箇所

```
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.

Options Indexes Includes FollowSymLinks ExecCGI
```

▼リスト2　/etc/httpd/conf/srm.conf中のシンボリックリンクを有効にする箇所

```
Alias /icons/ "/home/httpd/icons/"

<Directory "/home/httpd/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Aliasを使ったディレクトリ管理

FollowSymLinksを使うと、シェルレベルからディレクトリ構成を確認するのは分かりやすい。しかし、シンボリックリンクの危険性と常に向き合わせになる。そこで、たいていはAliasを使ってドキュメントディレクトリを追加する方法が使われる。Aliasはシンボ

リックリンクを張ることなく、URLに対して別のディレクトリを参照させるものだ。今回のファイル管理用途以外にも、ドキュメントディレクトリのパーティションの容量が足りなくなり、別のパーティションに一部のHTMLドキュメントを分割して管理するようなときにも使われる。

この処理は、決して特別なことではなく、Apacheではごく普通に使われている方法だ。たとえば、ファンシーインデックス時のアイコン表示は、ドキュメントルート直下の/icons/というディレクトリを見にいっている。しかし、/home/httpd/html直下には、iconsというディレクトリが存在しない。iconsディレクトリの実態は、/home/httpd/icons/にあり、/etc/httpd/conf/srm.conf中で**リスト2**のように設定されている。

この中で、

```
Alias /icons/ /home/httpd/icons/
```

というグローバルな設定をしているため、ドキュメントディレクトリにiconsディレクトリがなくても、/iconsへのアクセスは/home/httpd/icons/にエイリアスされ、/home/httpd/icons/が参照されるという仕組みだ。

Aliasの使い方はまさにこれで、URLで渡される文字列に対して、実ディレクトリでのAlias先を指定する。たとえばさきほどのwebgroupの例でいくと、

```
Alias /webgroup/ /home/naka
                        /public_html/
```

の1文を/etc/httpd/conf/srm.confに加えておく。/home/naka/public_htmlのシンボリックリンクは不要だ。このAlias設定を有効にするには、一度Apacheを再起動する。

```
# /etc/rc.d/init.d/httpd restart
```

Apacheのリスタート時には、瞬間的にサービスが停止する。しかし、新たなAliasが加わってサービスが再開される。

このAliasによって、http://www.ho.point5.co.jp/webgroup/とhttp://www.ho.point5.co.jp/~naka/でアクセスできる先は同一のものだ。~naka/でのアクセスを抑制するならば、~naka/public_htmlをAliasするのではなく、~naka/webdirといった、まったく違うディレクトリを利用するように設定すればいい。

ユーザーのホームディレクトリ下の、適当なディレクトリに対してAliasを設定することで、FTPでHTMLをアップロードするときに、ドキュメントディレクトリを探すためにハードディスク内をうろつかれる心配がなくなる。いちばんのメリットはまさにこの部分といえそうだ。さらには、利用者に分かりやすいURLを欲しいという要望にも応えられる。

しかも、この方式にはもっと大きなメリットもある。Apache管理者がドキュメントディレクトリを割り当てたファイルシステムの空き容量を気にせずにすむ点だ。万一利用者が巨大なコンテンツをアップロードしたとしても、ドキュメントディレクトリとホームディレクトリのファイルシステムが違えば、ドキュメントディレクトリには一切影響が出ない。ドキュメントディレクトリ内に巨大なコンテンツを置かれる可能性に脅えながら運用しなくてもすむわけだ。

AliasMatchでファイルを正しく引き継ぐ

ただし、Aliasにはいくつかの仕様上の問題点がある。まず、

```
Alias /webgroup/ /home/naka
                        /public_html/
```

と設定した場合、Webブラウザが要求しているURLが/webgroup/という指定であれば、正しく/home/naka/public_html/へエイリアスする。しかし、/webgroupというようにディレクトリの最後に/を付けずに呼び出したときには、Aliasパターンにマッチしない。マッチしなければ、/home/naka/public_html/へエイリアスされない（**画面1**）。

それなら、

画面1　Alias /webgroup/ /home/naka/public_html/の設定で、URLによる結果の違い。左はwebgroupというように、ディレクトリに対して/を入れなかった場合、NotFoundとなる。Alias設定で最後の/までAliasした場合、ディレクトリは必ず/で終わらせないとAliasがかからない

```
Alias /webgroup /home/naka
                        /public_html
```

とするとよさそうに見える。しかし、この設定でも得られる結果はあまり変わらない。今度はAliasの対象元が/webgroupであり、URLが/webgroupという指定であれば正しく動作するが、/webgroup/や/webgroup/1.gifというURL指定では、正しくAliasされない。Apache内部では、あくまでも最後に/が付いたときのみ、ディレクトリとして認識して処理するが、Alias元が/で終わっていなければ、それをディレクトリとして認識しない。

こうした問題点があるため、ディレクトリに対する処理は、AliasではなくAliasMatchを使うケースがほとんどだ。

AliasMatchを使った書き換えをすると、

```
AliasMatch ^/webgroup(.*) /home
                        /naka/public_html$1
```

という、正規表現を使った記述ができる。この場合、URLの頭が/webgroupで始まるものすべては、/home/naka/public_htmlに置き換えてアクセスする。しかも/webgroup以下にある文字列はそっくり/home/naka/public_html以下に付加される。AliasMatchを使えば、Aliasで生じた問題点はすべて解決できるわけだ。

ただし、これですべてが解決できるわけではない。たとえば、ドキュメントルートにwebgroupsというディレクトリを作った場合、/webgroupsへのリクエストは、AliasMatchにひっかかり、/home/naka/public_htmlsと展開される。webgroupとwebgroupsのように片方を含んだ名前が存在するときにトラブルが起こる。結果として、リクエストしたページがないということになる。AliasMatchは必ずしも万能というわけではないのだ。

Aliasの置き換えはログで管理する

Aliasでの置き換えはメジャーな方法であるものの、シェルからはどのようにAliasがかかっているかまったく見えない。何を見にいってエラーが出ているのかは、Apacheのエラーログから判断する。

たとえば、**リスト3**では、error_logから2つのエラーを切り出している。最初のエラーログは**画面1**のエラーログで、純粋にwebgroupというディレクトリもしくはファイルが見つからないということだ。Aliasで設定しているようなときには、このようなエラーログになる。

もうひとつのエラー例は、AliasMatchの場合だ。/webgroupsへのアクセスをしたために、/webgroup部分のみが展開され、存在しないディレクトリが記載される。特にAliasMatchを使っている場合、思わぬパターンでAliasが展開されるときがある。こんなときは、ログファイルを見れば判明する。

ホストをまたがったときにはRedirectを使う

長期に渡ってWebサイトを運用してくると、人気のあるWebページを分離して、別サーバを立てるというケースが出てくる。この場合、ホストをまたがることになる。ところが、AliasMatchでは、同一ホスト内だけでのAliasしかできない。

そこで、ホスト間での転送もできるものとして、Redirectが用意されている。そして、Aliasに対してAliasMatchがあるように、Redirectに対してRedirectMatchがある。正規表現によるリダイレクトパターンを設定できるほうが、RedirectMatchだ。

たとえば、

```
RedirectMatch ^/icons(.*) http:
     //www.in.point5.co.jp/icons$1
```

と指定すれば、iconsディレクトリ以下のみ、www.in.point5.co.jp/iconsにRedirectされる。ヒット数の多いコンテンツを新たにハウジングやホスティングしたサーバへ移動したときには、この方法でRedirectをすることが多い。このほかにも、画像以外のコンテンツは自社内専用線上のWebサーバに置き、画像ファイルなどのイメージプロパティをすべてホスティング先のホストへ置いて、画像ディレクトリをRedirectMatchでホスティング先サーバへ分散させるという方法にも使われる。

いずれにしても、Redirectはあくまでもredirectの処理しかしないため、ブラウザにとっては文字どおり、再度接続し直しの処理となる。このため、ブラウザのURL表示に新たなURLが表示される点には注意しておきたい。なお、RedirectMatchでも、最後にApacheのリスタートが必要だ。リス

▼リスト3　/var/log/httpd/error_logの例

```
[Mon Nov 15 21:18:02 1999] [error] [client 192.168.9.241] File does not exist: /home/httpd/html/webgroup
[Mon Nov 15 21:18:02 1999] [error] [client 192.168.9.241] File does not exist: /home/naka/public_htmls
```

タートしない限り、設定内容は反映されない。

RewriteRuleを使った複雑な管理

RedirectMatchよりもさらに強力なRewriteRuleというURL変更技術をApacheは持っている。しかし、RewriteRuleはモジュール形式で提供されているうえに、標準ではRewriteRuleを組み込まれていない。つまり、RewriteRuleを使うには、ソースレベルからの再コンパイルが必要だ。

たいていのLinux環境では、カーネルのソースはインストールしてあっても、それ以外のソースはインストールしていないだろう。どうせここでApacheソースを入れるならば、最新のApache 1.3.9を導入したい。そこで、RewriteRuleを試す前に、Apache 1.3.9をインストールしてほしい。ソースレベルからのインストールについて、コラムで触れておいた。また、ソースレベルでのインストールになることで、ドキュメントディレクトリ、コンフィグレーションディレクトリなど、すべて本来のApacheオリジナルのものとなる。以後の本記事では、Apache 1.3.9のオリジナルに沿ったディレクトリ構成で解説していく。

RewriteRuleで.htmと.htmlを同一視させる

RewriteRuleに話を戻そう。RewriteRuleのひとつの例として、/usr/local/apache/conf/httpd.confの<Directory />行を、

```
<Directory />
```

Column

Apache 1.3.9のインストール

■Apache_1.3.9.tar.gzの入手と展開

http://www.apache.org/dyn/closer.cgiから、Apacheをダウンロードできるサイト一覧が出てくる。いずれも公式サイトだ。この中から負荷の少なそうなところを選んで最新のApacheソースを入手する。ファイルは、apache_1.3.9.tar.gzというもので、これを/usr/local/srcに保存する。

このあとは、

```
# cd /usr/local/src
# tar xvfz apache_1.3.9.tar.gz
```

という一連の作業で、ソースファイルを伸長、展開する。

さらに、

```
# cd apache_1.3.9
# ./configure
# make
# make install
```

の手順で、標準状態のApacheが/usr/local/apache以下にインストールされる。コンフィグレーションディレクトリは/usr/local/apache/confで、ドキュメントディレクトリは/usr/local/apache/htdocsとなる。

■起動時のdaemon処理

ここまでの作業で、Apache1.3.9が組み込まれるが、このままでは再起動をしたときには既存の1.3.6が動くことになる。そこで、

```
# cd /usr/sbin
# mv httpd httpd.old
# ln -s /usr/local/apache/bin/httpd .
```

として、起動時に動くApacheを新しいものにしておく。これで1回rebootして、Apacheが正しく動くことを確認できれば、1.3.9への切り替えは終了だ。従来どおり、/etc/rc.d/init.d/httpdを使い、restart、stop、startという機能が使える。ただし、既存のApacheコンフィグレーションファイルを引き継いでいないので、新たに/usr/local/apache/confで再作成する。

■rewriteモジュールの組み込み

rewriteを例に、モジュールの組み込みを説明しよう。以前と異なり、Apache1.3からはconfigure時にモジュール指定ができるようになっている。たとえば、rewriteを組み込むには、

```
# cd /usr/local/src/apache_1.3.9
# ./configure --enable-module=rewrite
# make
# make install
```

とすることで、モジュールが組み込める。configureを起動したときにsrc/Configuration.apaci内で必要なモジュールのコメントが削除されるため、直接src/Configuration.apaciを触る必要がなくなったわけだ。

実際にrewriteモジュールが組み込まれたかどうかは、

```
# /sbin/httpd -l
```

で確認できる。この中にmod_rewrite.cが含まれていれば、rewriteモジュールが正しく動作する。

Apache HTTP Server Download
http://www.apache.org/dyn/closer.cgi

```
    Options FollowSymLinks
    AllowOverride None
    RewriteEngine on
    RewriteRule ^(.*)htm$ $1html
</Directory>
```

のようにしておく。これは、＊.htmのアクセスを＊.htmlに置き換えてくれるというものだ。たとえば、これまではHTMLファイルの拡張子を.htmで統一していたものを、＊.htmlに変えたというようなとき、自分で管理しているHTMLファイルに対しては、Perlのスクリプトで一斉に変更できるだろう。

しかし、ほかのユーザーが公開しているコンテンツや、CGI部分まではなかなか修正の手が回らない。こんなときには、htmとhtmlファイルを同一視するような仕組みを組み込んでおきたいものだ。そして、もっとも重要なことは、公開しているWebサイトということは、ブックマークやほかのサイトからリンクが張られている可能性がある。特に会社の場合、必ずしも最新の情報にリンクが張られているわけではない。過去のニュース発表文書のURLにリンクをしてあるようなとき、安易にファイル名すら変更できない。こんなとき、RewriteRuleを使うと既存のURLのままアクセスができる。

これは非常に有効な方法で、拡張子でGIFとgif、jpegとjpgがごちゃまぜになっているときなどでも、RewriteRuleでどちらかに合わせてしまえば、HTMLファイル中で拡張子の大文字、小文字の問題にひっかかって、HTMLを直さなければいけないという問題にも対処できる。

たとえば、アップロードする画像は、jpg、もしくはgifという、小文字の3文字拡張子に限定してしまえば、

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    RewriteEngine on
    RewriteRule ^(.*)[Gg][Ii][Ff]$ $1gif
    RewriteRule ^(.*)[Jj][Pp][Gg]$ $1jpg
    RewriteRule  ^(.*)[Jj][Pp][Ee][Gg]$
$1jpg
</Directory>
```

というリライトパターンの定義さえしておけばいい。こうすれば、HTML中でGIFやJPEGで呼び出すようになっていたとしても、ファイルさえjpg、もしくはgifという統一した形でアップロードしてさえいれば、つまらない部分で画像が読み込めないという心配をしなくてすむ。

Column

URLの変更なしにほかのサーバへ

RedirectMatchやRewriteRuleでホスト間移動をすると、利用者ブラウザに必ずリダイレクト先が表示される。これを抑制する方法の1つとして、ApacheのProxyサーバ機能を使う方法がある。Proxyサーバ機能はApacheのモジュールで構成されるため、画面1のように再度コンパイルが必要だ。--enable-moduleのオプションとしてproxyを指定し、再度makeとmake installをすると組み込まれる。画面1はrewriteモジュールと同時に組み込んだときの例だ。

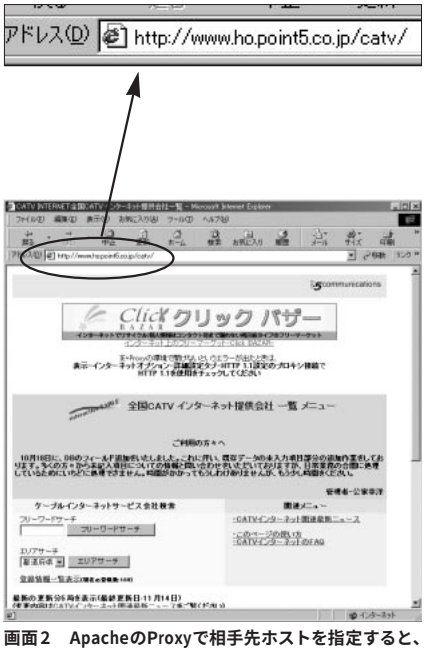
もっとも簡単な設定の使い方は、/usr/local/apache/conf/httpd.confの<Directory>で囲まれていないところに、次のような設定をする。

```
ProxyRequests On
ProxyPass / http://wire.point5.co.jp/
```

この結果が画面2だ。ちょっとしたバーチャルホストが立ち上げられるわけだ。ただし、いうまでもないが、この状態ではApacheが公開Proxyサーバとなっている。このWebサーバをProxyに設定すれば、だれもが全世界へのプロキシサーバとして利用できる。さらに、リダイレクトではなく、Proxyサーバとして動いているので、ネットワークのトラフィック軽減に直結しない点にも十分注意したい。

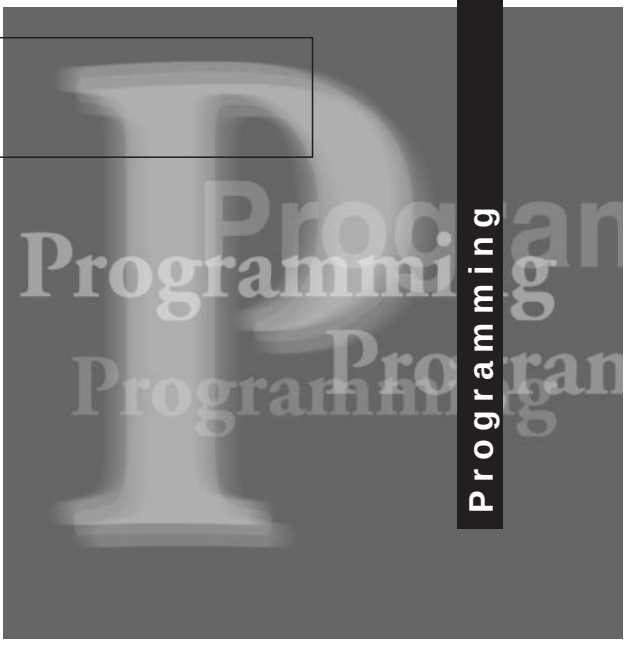
```
# cd /usr/local/src/apache_1.3.9/
# ./configure --enable-module=rewrite --enable-module=proxy
# make
# make install
```

画面1　proxyモジュールを組み込むときのmake例



画面2　ApacheのProxyで相手先ホストを指定すると、ApacheがProxyサーバとなって動作するため、ブラウザのURL表示が変わらない

プログラミング工房



第3回となってこの連載にはタイトルがないことに気づいた! そこで「プログラミング工房」と名づけ、今後もプログラミングにまつわるいろいろな話題を提供していく予定なので、今後ともよろしくお願いします。今回はMingw32を使って、Linux上でWindowsアプリケーションを作ります。

第3回 Windowsプログラミング（1）

文：藤沢敏喜
Text : Toshiki Fujisawa

これまで家庭用電化製品や携帯用機器などに搭載するマイクロプロセッサのソフトウェア開発は、UNIXワークステーション上で行われることが多かった。UNIXは、プログラムの開発環境として、たいへん優れていたからである。

ところが、UNIXワークステーションが高価だったこともあって、低コストの4ビットCPUなどの場合は、開発環境としてDOSしか供給されないこともあった。DOSの非力な開発環境では苦労も多く、泣きながら徹夜でプログラムを書かなくてはならないこともあったのだ。

しかし時代は変わり、5万円のPCでLinuxや＊BSDなどのUNIX互換OSが動作するようになった。このような状況を背景に、ソニーのPlayStation2の開発もLinux上で行われているらしい。これからは、劣悪なプログラミング環境に苦しまずに、UNIX環境での快適なプログラミングができる素晴らしい時代になりそうである。

クローズな世界から オープンな世界へ

Linuxは、サーバ用としてはもちろん、前述のようにPlayStation2のメイン開発環境になるなど、着実にシェアを伸ばしていて、数年後にはデスクトップ環境として利用される可能性さえも秘めている。

しかし、現状ではまだまだWindowsが広く使われているのが事実である。そのため、そのうえで動作するアプリケーションの開発にもWindowsを使わざるを得ない。しかも、Visual C++などMicrosoft製の高価な開発環境を購入する必要がある。

Windows以前のDOS環境では、フリーのCコンパイラもあったし、フリーのソフトウェアも多かったのだが、最近ではVisual C++の購入資金を回収するためか、フリーのソフトウェアはすっかり減り、ほとんどがシェアウェアとなってしまった。OSの独占だけでなく、開発環境の独占による弊害も目立つと感じるのは筆者だけだろうか？

このような、Microsoftの世界で閉じてしまった状態を解き放とうというのが今回のテーマである。つまり、オープンなOSの上のオープンな開発環境を用いて、さまざまなプラットフォーム上で動くオープンなアプリケーションを開発しようというわけである。

Linuxで Windowsプログラミング

今回紹介する「Mingw32」というパッケージを使うと、LinuxやFreeBSD上でWindowsアプリケーションを作成することができる。このMingw32については、以前にもLinux magazine No.2掲載の「**GPG ―自動暗号化システムをつくる―**」の中で、GPGをWin32用にコンパイルする環境としてごく簡単に紹介している。

PC-UNIX環境でWindowsアプリケーションを作成できるということは、UNIX環境に慣れたプログラマーの立場から見ると、とても画期的なことである。筆者は、以前に

このMingw32の実力を測るため、Visual C++で書かれた3万行程度のGUIベースのWindowsアプリケーションをMingw32上でコンパイルしてみたことがある。結論からいうと、実用性は十分だった。ソースファイルにたった数行ほどの変更を加えるだけで、まったく問題なく動作させることができたのだ。

このテストに使用したWindowsアプリケーションを最初に開発したときは、OSに依存した処理を行う部分は切り分けて書いておき、非依存の部分はLinuxやFreeBSD上でもコンパイルできるようにしてあった。そのため、最初の開発やデバッグはすべて慣れたFreeBSD上で行い、そのあとでOSに依存する処理部分などを、Windows上でVisual C++を用いてWin32用のバイナリを作成していた。二度手間のようだが、慣れた環境での開発効率はこの手間に代えがたいものがあるのだ。しかし、これからはこのMingw32があるので、Visual C++が不要になり、二度手間も必要なさそうである。

また、LinuxではVMware（http://www.vmware.com/）という、PCの仮想ハードウェア環境を構築するソフトウェアがあるので、このうえにWindowsをインストールして動かせば、動作テストを含めて、Windowsアプリケーション開発のほとんどの作業をLinux上で行うことが可能ではないかと思う。

さらに最近では、Troll Tech社が提供している「Qt」のように、WindowsとX Window Systemで共通に使えるGUIツールキットもいくつか登場してきている。これらを利用すれば、WindowsとX Window Systemの両方で動作させるプログラムを作成することも可能になりつつある。

読者のみなさんも、ぜひオープンなアプリケーションプログラムを作成してみてほしい。

Mingw32とは？

Mingw32（Minimalist GNU Win32 Package）は、Microsoftが提供するランタイムライブラリをリンクすることができる初期化コードと、ヘッダファイルからなるパッケージである。コンパイラとしては、egcs（gccから派生したCコンパイラ）などを用いることができる。

このパッケージは、標準設定ではWindowsに標準で組み込まれている「CRTDLL」という基本ライブラリしか利用しない。そのため、高速でサイズの小さなWin32アプリケーションプログラムを作成できる。なお、Mingw32

Column

クロスコンパイラ

コンパイラを動作させているOSと異なる環境で実行するバイナリを作成することを「クロスコンパイルする」といい、そのバイナリを作成するために用いるコンパイラを「クロスコンパイラ」と呼ぶ。

クロスコンパイルを行うのにはさまざまなケースがある。たとえば、本文で紹介したMingw32のように、i386互換のCPUで動作するLinux上で、i386互換のCPUで動作するWindows用のバイナリを作成するというのもひとつの例である。さらに、冒頭で述べたように組み込み用4ビットCPUのバイナリを作成するために、SunOS上でコンパイルを行うというケースもある。

また、クロスコンパイルは、新しいマシンや新しいCPUへOSを移植するような場合にもよく用いられる。たとえば、**表c-1**に示すような、既存の開発環境と新しく開発する環境で「マシン／CPU／OS」が異なる場合である。

（1）は、MIPSをCPUとするWindows CEマシンにNetBSDを移植する場合の例であり、（2）はLinux誕生時に用いられた例である。（3）は、1992年に、京大マイコンクラブが386BSDというOSをNEC-PC-9801へ移植したときに用いた方法である。

なお（4）は、筆者が1992年にPC/AT用の386BSDを当時手元にあったNEC-PC-9801NSへ移植したときの例で、DOS上で動作するdjgccを用いて386BSDカーネルをコンパイルした。ちなみに、このときは8文字＋3文字というDOSのファイル名の制限に苦しめられた。さらに、カーネルをコンパイルするマシンと実行するマシンが同一だったため、頻繁なリブートが必要になり、とても苦労した記憶がある。

違うマシンでクロス開発を行えば、エディタなどを利用しているマシンを頻繁にリブートすることが避けられるので作業効率も上がる。Windows 95/98では、ちょっと間違ったプログラムを書くと、すぐにシステムクラッシュが起こり、頻繁にリブートを繰り返さなければならない。ひどい場合にはファイルを失うこともあるのだが、違うマシンでコンパイルをすれば、安全な開発を行うことができる。

ケース	開発(コンパイル)環境の「マシン／OS／CPU」	新しい(実行)環境の「マシン／OS／CPU」
(1)	PCAT/i386/NetBSD	Windows CE/mips/NetBSD
(2)	PCAT/i386/NetBSD	PCAT/i386/Linux
(3)	Sun/sparc/SunOS	PC98/i386/386BSD
(4)	PC98/i386/DOS	PC98/i386/386BSD

表c-1　異なるアーキテクチャ間のクロスコンパイルの例

に関する情報としては、

http://www.geocities.com/Tokyo/Towers/6162/gcc.html

が詳しく、Win32プログラミングのチュートリアルなどの情報や、他のフリーWin32コンパイラの情報もある。

このMingw32は、Windows上で動作させることもできるが、Mumit Khan氏により、egcsによるクロスコンパイル環境が提供されている。これについては、

http://www.xraylith.wisc.edu/~khan/software/gnu-win32/egcs.html

で解説されている。

Windows上で動作するUNIXライクなプログラミング環境としては、Cygnus Solutions（http://www.cygnus.com/）が提供する「Cygwin32」（旧gnuwin32）が有名である。このCygwin32は、cygwin.dllというライブラリによってUNIX互換エミュレーションを行っている。したがって、UNIX用に開発されたプログラムをほとんど変更しないでも、そのままWindow上で使用することが可能となっている。しかし、実行時にはGPLの配布条件に従うcygwin.dllが必要になるというデメリットがある。また、UNIX互換エミュレーション層でのオーバーヘッドがあるため、実行速度の面では不利になる。

一方、前述したようにMingw32はWindowsの基本システムに存在するCRTDLLライブラリだけしか使わないため、自分で作成したバイナリプログラムだけを配布できるという配布条件上のメリットがある。また、cygwin.dllのようなオーバーヘッドもないため、Win32用のプログラムを作成する環境としては優れている。

Mingw32で作成するサンプルプログラム

さっそくこのMingw32を使ったサンプルプログラムを紹介といきたいところだが、ただ単に「Hello World」の表示だけでは、ありきたり過ぎてつまらない。かといって、あまり大規模なものでは、Windowsプログラミングに慣れていない読者には理解するのがたいへんだ。

そこで、あまり難しくなり過ぎない（約200行）程度で記述できるようなもので、多少実用的なプログラムも作成してみよう。具体的には、Mingw32を使って作成した暗号化プログラム「gpg.exe」を呼び出して、GUIで操作できるアプリケーションの作成を題材としてみる。

ただし、紙幅の関係で今月はMingw32のインストールとテストプログラムのコンパイルだけとなる。この暗号化アプリケーションの作成法については、次号で行うことをご了承いただきたい。なお、記事の最後にこの暗号化アプリケーションの概略も紹介しておこう。

ファイル名	サイズ（バイト）
onfig.tar.bz2	13596
intl.tar.bz2	50944
etc.tar.bz2	99954
gprof.tar.bz2	125087
libiberty.tar.bz2	133402
toplevel.tar.bz2	138427
egcs-1.1.1-x86-win32-patches.tar.gz	143121
include.tar.bz2	255036
opcodes.tar.bz2	296810
ld.tar.bz2	333065
binutils.tar.bz2	431676
bin-1999-04-05.tar.gz	588552
texinfo.tar.bz2	963891
bfd.tar.bz2	1247639
gas.tar.bz2	1379313
egcs-1.1.1.tar.bz2	8917442
計	15117955

表1　Mingw32の実行に必要なモジュール群

Mingw32のインストール

では、まずMingw32のインストールから始めよう。LinuxにMingw32のクロスコンパイル環境をインストールするには、下記のWebサイトに詳しい解説がある。

http://www.xraylith.wisc.edu/~khan/software/gnu-win32/mingw-cross-howto.txt

しかし、**表1**に示したように必要なパッケージが非常に多いうえ、インストール手順も複雑である。そこで、手軽に始めたいという読者のために、今回はインストール用のシェルスクリプト（Install-Mingw32.sh）を作成し、付属CD-ROMに収録してあるので、こちらを利用してほしい。

このシェルスクリプトは、インターネットから必要なパッケージをwgetコマンドを利用して自動的に取得し、パッケージのチェックサムを計算によって正しくダウンロードされたかを確認してから、自動的にマシンにインストールを行うようになっている。使用上の注意などは、同梱のREADME.txtを参照してほしい。

■スクリプト実行の注意点

このスクリプトがインターネットから転送するファイルは全部で約15Mバイトで、インストールスクリプトを実行するためには、/var/tmp/に数百Mバイトの空き容量を必要とする。筆者が試したPentium-200MHzのマシンではコンパイルに1時間ほど要した。

なお、このスクリプトは必要なファイルをすべて転送し終わると、キー入力待ちとなり停止するので、ダイヤルアップ接続環境の方は、ここでいったんPPP接続を切るとよいだろう。

このスクリプトは、実行にbzip2、gcc、gmake、automakeといったコマンドが必要となる。最近のLinuxディストリビューションでは、これらのコマンドはあらかじめインストールされていることが多いので問題はないと思うが、もしもインストールされていない場合は、各自の環境に応じてあらかじめインストールしておいてほしい。

このシェルスクリプトで、Mingw32をインストールすると、/usr/local/mingw32ディレクトリ以下に、**表2**のようなサブディレクトリ構成でインストールが行われる。また、/usr/local/i386-mingw32/bin/にある、コンパイラなどの実行バイナリは、/usr/local/bin/からシンボリックリンクが張ってあるので、環境変数PATHに/usr/local/bin/が登録されているかも確認しておこう。

テストプログラムの実行

インストールが終わったら、正常にインストールできたかをチェックするため、次のような小さなプログラムをコンパイルしてみよう。

▼リスト1　contest.c

```
#include <stdio.h>

int main(void)
{
    printf("Hello World\n");
    return 0;
}
```

▼リスト2　Makefile for contest.c

```
CC=i386-mingw32-gcc

contest.exe : contest.c
        $(CC) -o contest.exe contest.c

clean:
        rm -f *.o *.exe
```

1.コンソールアプリケーション

まず、定番の「Hello World」を表示するコンソールアプリケーションから作成してみよう。

contest.c（**リスト1**）と、Makefile（**リスト2**）を作成し、makeコマンドを実行するとcontest.exeができあがる。このcontest.exeファイルを、ftpなどでWindowsマシンへ転送し、MS-DOSプロンプトでこれを実行すれば、「Hello World」という文字列が表示される（**画面1**）。

なお、ftpを使う際はバイナリモードで転送することを忘れないようにしてほしい（ftpのプロンプトから「binary」と入力してから転送する）。また、いちいちftpするが面倒なら、Sambaを利用してLinuxのファイルシステムをWindowsのドライブとして割り当てておくと、より快適な環境が構築できる。

2.ウインドウアプリケーション

さて、今度はメッセージボックスウインドウを開くプログラムに挑戦してみよう。ウインドウを開くとなると、難しそうに思う読者もいるかもしれないが、**リスト3**を見ればわかるとおり、MessageBox関数を呼ぶだけの単純なプログラムだ。

ディレクトリ	説明
i386-mingw32/lib	Win32ライブラリ
i386-mingw32/include	Win32インクルードファイル
i386-mingw32/bin	コンパイラなどの実行バイナリ
bin	コンパイラへのハードリンク
lib	gccライブラリ
include	g++インクルードファイル
man	マニュアル
info	infoファイル

表2　Mingw32のディレクトリ構成

画面1　contest.exeの実行

コンソールアプリケーションのときと同じように、Cプログラムソース（**リスト3**）と、それをコンパイルするためのMakefile（**リスト4**）を用意し、makeを実行するとwintest.exeができあがる。

このwintest.exeをWindowsマシンへ持っていき、マウスでダブルクリックすると、メッセージボックスが表示される（**画面2**）。

ここで、wintest.cを見てみよう（**リスト3**）。まず、WinMain関数が定義されている。Windowsのプログラムでは、まずこのWinMain関数が呼ばれることになっている。

WinMainには引数が4つあり、最初の2つは「インスタンスハンドル」と呼ばれるもので、UNIXでいうプロセスIDのようなものであると考えておけばよいだろう。3番目のlpCmdには、コマンドライン引数が渡される。つまり、プログラムにファイルをドラッグした場合などは、この引数によってファイル名を知ることができる。最後のnShowという引数は、ウインドウの初期表示方法を指示する数値となっている。

3.少し複雑なウインドウアプリケーション

上記の2つはあまりにも簡単なプログラムなので、物足りなく感じる読者は、

http://www.geocities.com/Tokyo/Towers/6162/win32/hello.html

で、「A More Complete Example」として紹介されている「test.c」を試してみるとよい。これをコンパイルして実行すると、**画面3**のようなウインドウが表示される。

このプログラムは、メッセージループとコールバック関数を用いた、標準的な構造をもったWindowsアプリケーションになっているので、勉強には最適である。また、いろいろなプログラムを試したい場合も、このサンプルを改造して実験してみるとよいだろう。

▼リスト3　wintest.c

```
#include <windows.h>

int WINAPI
WinMain( HINSTANCE hInst, HINSTANCE hPrev, LPSTR
lpCmd, int nShow)
{
    MessageBox(NULL, "Helo World", "Title", MB_OK);
    return 0;
}
```

▼リスト4　Makefile for wintest.c

```
CC=i386-mingw32-gcc
LOPT=-Wl,--subsystem,windows
LIBS=-lkernel32 -luser32

wintest.exe : wintest.c
        $(CC) -o $@ wintest.c $(LIBS) $(LOPT)

clean:
        rm -f *.o *.exe
```

来月号の予告

今月号では、Mingw32環境のインストールの説明と、テストプログラムのコンパイルまでを解説した。Mingw32のインストールは時間がかかるが、付属CD-ROMに収録したシェルスクリプトを利用してもらえば難しくはないので、ぜひ挑戦してみてほしい。

さて次号では、Mingw32を用いた実用プログラムとして、暗号化を行うGUIベースのWindowアプリケーションについて解説を行う予定である。このアプリケーションがどのようなものか予告も兼ねて紹介しておこう。

■アプリケーションの仕様

アプリケーションの動作仕様は次のようなものだ。

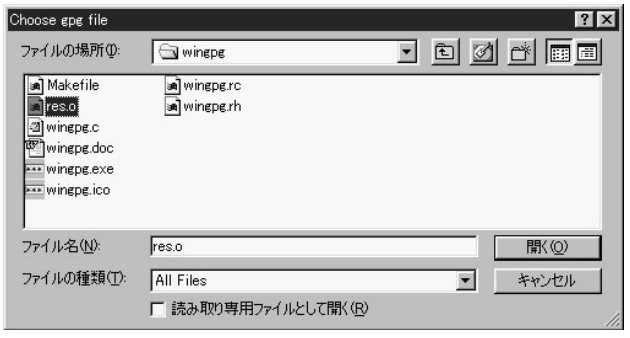
デスクトップに置かれたアプリケーションをダブルクリックすると、Windows標準のファイル選択のためのダイアログボックスが表示される（**画面4**）。

画面2　wintest.exeの実行

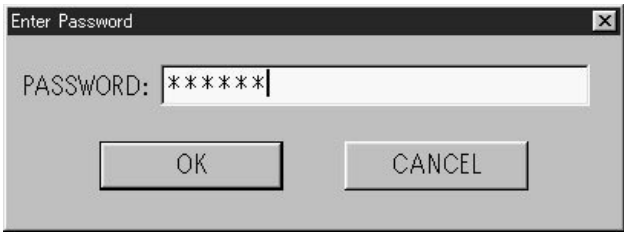
画面3　サンプルプログラム（test.c）の実行

ここでディレクトリを選び、暗号化したいファイルを選択すると、パスワードの入力画面が表示される（**画面5**）。暗号化するためのパスワードを入力すると、自動的にgpg.exeが起動され、暗号化が行われる。暗号化が正常に行われると、その旨を伝えるメッセージが表示される（**画面6**）。なお、プログラムを簡単にするため、GPGの機能のうち共有鍵を使うモード（symmetric）のみをサポートすることにする。

このアプリケーションでは、ファイルの選択とパスワードの入力だけという非常に単純なプログラムだが、簡単に暗号化／復号化を行うことが可能で、実用度は高いと思う。また、このプログラムをサンプルとして選んだのは、GUIで操作ができ、かつ安心して利用できる暗号化プログラムがあまり存在しないというのも理由のひとつである。



画面4　暗号化するファイルを選択するダイアログボックス



画面5　パスワード入力画面

画面6　暗号化終了を通知するウィンドウ

Column

Windowsの開発環境をLinuxから使う

Mingw32のように、コンパイラやアセンブラのソースが公開されていれば、クロスコンパイル環境を構築することは比較的簡単である。しかし、これ以外にもクロスコンパイルする方法はある。PC-UNIX環境の快適さを享受しつつ、Windowsアプリケーションの開発を行うにはどうしたらよいかを考えてみよう。

1.WindowsにUNIXライクな環境を構築する

Windows 3.1の時代に、アスキーからBOW（BSD on Windows）というソフトウェアが発売された。これは、Windows 3.1上で、FreeBSDのバイナリプログラムを動作させてしまうという、魔法のようなプログラムであった。当時、嫌々ながらWindowsプログラミングをせざるをえない状況にいた筆者が飛びついたのはいうまでもない。bash、ls、diff、awk、perlといったツールはもちろん、本物のmuleまでもが使えるというのは本当にありがたかった。その後、BOWはWindows 95や（非公式ながら）Windows NT 4.0などにも対応し、多くのユーザーを獲得している。

また、Cygnus SolutionsがCygwin32環境をインストールすることにより、WindowsでもUNIXライクな環境が実現できる。この環境に加えて、Win32用に移植されたMeadow（旧Mule for Windows）を利用すれば、かなりUNIXに近い開発環境をWindows上に構築できる。

しかし、タスクの切り替えがお粗末なWindowsでは、Cygwin32がUNIXエミュレートに使用しているライブラリ「cygwin.dll」や、OSそのもののオーバーヘッドが大きいため、同じマシンでもLinuxやFreeBSDを使用しているときに得られるような、キビキビと動く快適さは得ることはできず、やはりイライラさせられてしまう。エミュレータなので仕方ないのだが…。

2.Linux上でDOSエミュレーターを使う

これは、1993年ごろ組み込み用の8ビットCPUのソフトウェア開発を行っていた筆者が実際に使っていた方法である。当時、＊BSDではDOSエミュレータがまだ存在しなかったため、Linux上でemudosを使ってDOS上のマクロアセンブラを動作させていた。

emudosでは、DOSからLinuxのファイルシステムをアクセスすることができたので、エディタはLinux上でEmacsを使うことができた。また、Linux上でエミュレータへ転送したり、ROMライタを使えるツールを作成したため、快適に開発を進めることができた。

3.コンパイルのみをWindowsで行い、編集作業をLinuxやFreeBSD上で行う

これを実現するには、2台のコンピュータが必要になる。1台目には、LinuxやFreeBSDをインストールし、編集作業などはすべてこのマシンで行うことにする。

このマシンの適当なディレクトリには、たとえば本文で紹介したwintest.c（**リスト3**）と、右記の**リストc-1**、**リストc-2**、**リストc-3**の4つのファイルを置いておき、このディレクトリをSambaなどでWindowsマシン側に公開しておく。

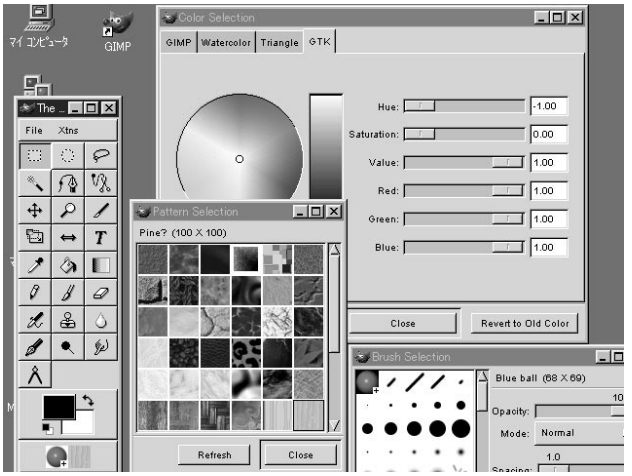
2台目のWindowsマシンでは、（スタート

オープンアプリケーションがもたらす意義

プログラミングには手を出したくないという方も少なくないかもしれない。しかし、今回紹介したMingw32のように、Linux上でWindowsアプリーケーションが開発できるツールさえあれば（しかも、フリー‼）、Linuxだけでなく Windowsもターゲットとしたアプリケーションを開発しようという気になる人もいるだろう。何といってもユーザー数が違う。

また、PC-UNIX上で動作する画像処理ソフトとして有名な「Gimp」は、すでにWindowsへ移植されており、Linux版とまったく同じ操作性を実現している（**画面7**）。このgimpで利用されているGUIツールキット「GTK+」を用いれば、Linuxだけでなく Windows上でも動作するプログラムを簡単に作成できるのだ。

このGimpの例のように、さまざまなOS上で操作性が変わらないアプリケーションが増えてくれば、アプリケーションを理由にひとつのOSに縛られることはなくなる。



画面7　Windowsへ移植されたフォトレタッチソフト「Gimp」

必然的に性能でOSを選ぶユーザーが多くなるだろう。そういう意味では、オープンなアプリケーションが広く使われるようになれば、LinuxやFreeBSDなどのオープンなOSのさらなる普及が望めるのではないかと思う。

アップフォルダなどを利用して）1台目のマシンで公開されているディレクトリを、Nドライブに割り当て、カレントディレクトリとして、MS-DOSプロンプトから公開フォルダにあるリストc-1のバッチファイルを動かしておく。

そして、1台目のLinuxマシンでmuleを起動し、“ M-x compile ”を実行すれば、muleのバッファの中でコンパイルの様子が表示される。エラーがあった場合は、“ M-x next-error ”を実行すれば、エラーのある行へ移動することができる。

さらに、実行ファイルがあるディレクトリを、Sambaなどを利用してデバッグ担当者に公開しておけば、Windowsマシンのキーボードやマウスに一切触れることなく、Windows上で動作するコンパイラを使った開発が行えることになる。

▼リストc-1　00loop.bat（改行コードはCR+LF）

```
net time ¥¥samba-machine /set /yes
@echo off
if exist _make.log del _make.log
set MSDEV=c:¥vc42
set PATH=%MSDEV%¥bin;c:¥windows;c:¥windows¥command
set LIB=%MSDEV%¥lib
set INCLUDE=%MSDEV%¥include

:loop
@echo off
echo WAITING ...
:wait
break = on
if not exist _dosexe.bat goto wait
echo executing...
nmake -nologo make-dos > _make.log
del  _dosexe.bat
goto loop
```

各自の環境に合わせて変更する

▼リストc-2　Makefile（改行コードはLFのみ）

```
make-unix:
        ./00start.sh

make-dos: wintest.exe

wintest.exe : wintest.c
        cl -W3 -owintest.exe wintest.c /MT /link user32.lib
```

▼リストc-3　00start.sh（改行コードはLFのみ）

```
#!/usr/local/bin/bash
 > _make.log
echo w32_make > _dosexe.bat
(
   while [ -e  _dosexe.bat ] ; do  sleep 1 ; done
   kill `ps ax | awk '$5 $6 $7~/^tail-
f_make.log/{print $1}'`
) &
tail -f _make.log | tr -d '[\015]'
```

特別講座

ステップアップC言語

Makefile

作成するプログラムの規模が大きくなると必要になってくるのが、makeコマンドである。makeコマンドは、Makefileという名前のファイルに記述された情報にしたがって、主に複数のファイルに分割されたプログラムを自動的にコンパイルするために用いられる。

複数ファイルのコンパイル

C言語のプログラムを作成する場合は、複数のソースファイル（＊.c）をコンパイルし、それぞれのオブジェクトファイル（＊.o）を作成する。そして、それらのオブジェクトファイルをリンカを用いて結合し、実行バイナリファイルを作成する。また、コンパイルを行う際には、適切なヘッダファイル（＊.h）も読み込む必要がある。

たとえば、

```
io_scsi.c io_usb.c io.c main.c
io_scsi.h io_usb.h io.h
```

という7つのファイルからなるプログラムを考えてみよう。ここで3つのヘッダファイルは次のようにインクルードされるものとする。

- **main.cではio.hのみをインクルードする**
- **io.cでは、io_scsi.hとio_usb.hとio.hをインクルードする**
- **io_usb.cでは、io_usb.hのみをインクルードする**
- **io_scsi.cでは、io_scsi.hのみをインクルードする**

ソースファイルio.cからは、“gcc -c io.c”というコマンドで、io.oというオブジェクトファイルが作成される。また、io_scsi.o、io_usb.o、main.oも同様に同名のソースファイルから作成される。これをリンカで結合したものが最終的に得られるバイナリ実行ファイルである。このバイナリ実行ファイルの名前を「testcmd」とすると、testcmdを作成するには次のようなシェルスクリプトを書けばよい。

```
#!/bin/sh
gcc -c io_scsi.c
gcc -c io_usb.c
gcc -c io.c
gcc -c main.c
gcc -o testcmd io_scsi.o io_usb.o \
                    io.o main.o
```

これで一連の作業が自動化されることになるが、このままでは1つのファイルを変更するたびに全部のファイルを再コンパイルすることになってしまう。プログラムが大きい場合やファイルが多くある場合は、時間がかかってしまい非効率的である。かといって、これを避けるために、変更したファイルを人間が判断して、それだけをコンパイルするようにすると、今度はコンパイルすべきファイルを忘れたりして、これも問題になってしまう。

Makefile

このような問題を解決するために誕生したのが、makeコマンドである。makeコマンドは、各ファイルのタイムスタンプから、変更すべきファイルを見つけ出し、自動的にコンパイルを行うことができる。ここで、どのファイルからどのファイルを作成すべきかといった依存関係を記述するのがMakefileである。前述の例のような場合、Makefileは**リストA**のようになる。

Makefileではマクロが利用できるので、**リストA**では、最初にOBJSというマクロに各オブジェクトファイルを代入していて、あとの行ではそれを$(OBJS)という表記をすることで参照している。

ここで、「:」の左側に描かれているのが生成するファイルで、「:」の右側に書かれている複数のファイルがそれを生成するために必要なファイルである。また、次行では生成するのに必要なコマンドを示している。たとえば次の、

```
io.o : io_scsi.h io_usb.h io.h
        gcc -c io.c
```

の部分なら、「io.oというファイルを作成するためには、io_scsi.h、io_usb.h、io.hという3つのファイルが必要で、“gcc -c io.c”というコマンドを実行する」ということがわかる。ただし、ここで実行コマンドが書かれた行（この例では“gcc -c io.c”の行）の左側は、タブ文字でなくてはならないことに注意してほしい。

makeコマンドは、このMakefileと各ファイルのタイムスタンプを参照し、必要なコマンドを自動的に実行するというわけである。

基本的な使い方をするだけなら、ここまでの説明だけでもmakeは簡単に利用できる。しかし、ここで紹介した以外にも便利な記法が多くあるので、より効果的な活用をするためにもマニュアルには目を通しておくとよいだろう。

また、makeコマンドはタイムスタンプを基にコマンドを実行するかどうかを決定しているため、違うマシン間でNFSやSambaでファイル共有をしている場合は、共有オプションを適切に設定し、各マシンの時刻をxntpdなどで一致させておく必要があることに注意してほしい。

▼リストA

```
OBJS=io_scsi.o io_usb.o io.o main.o

testcmd : $(OBJS)
        gcc -o tstcmd $(OBJS)

io_scsi.o : io_scsi.h
        gcc -c io_scsi.c

io_usb.o : io_usb.h
        gcc -c io_usb.c

io.o : io_scsi.h io_usb.h io.h
        gcc -c io.c

main.o : io.h
        gcc -c main.c
```

PostgreSQLを極める

今回は、前回紹介した「ユーザー定義関数」を利用した便利な機能「トリガ」を紹介します。トリガは、イベントに応じてテーブルに自動処理の機能を持たせる機能で、より実用的な環境の構築が可能になります。

第3回 トリガの利用

文：片岡裕生
Text : Hiroki kataoka

前回は連載2回目にもかかわらず、いきなり「ユーザー定義関数」を紹介しました。ハードルが高いなぁ、と感じられた方もいたかもしれません。しかし、ユーザー定義関数は今回紹介するトリガ以外にも、今後紹介していく予定となっているさまざまな機能の中で顔を出てくる重要な機能なのです。そんなわけで、より快適な機能を利用するためには欠かせないものですので、しっかり理解しておいてください。

今回は、アイデア次第でさまざまな場面に活用できる「トリガ」について解説します。難しい印象を持つ方もいるかもしれませんが、考え方自体は単純なものです。そして、習得する面倒をはるかに上回るだけのメリットを与えてくれる、非常に便利な機能なのです。

さあ、トリガの紹介を始めましょう。

もしも、トリガがなかったら

たとえば、あるデータベースを閲覧／操作するアプリケーションがあり、その内容の信頼性が非常に重要だったとします。そこで、データの信頼性を確保するために、いつ、誰が、どのようにテーブル内容を更新したのかという情報を記録するシステムを考えたとします。もちろん、更新があるたびにその内容をどこかのテーブルに保管するようにアプリケーションを作成すればすむことなのですが、悪意のあるユーザーにデータベース内のテーブルを直接書き換えられた場合には、更新記録はどこにも残らないことになってしまいます。これでは困ります。

しかし、もしもテーブル自体に更新記録を保管する機能が追加できたらどうでしょうか？　そうすれば、テーブルを直接書き換えられたとしても更新記録が残るわけですから、データの信頼性は多少高くなるはずです。

また、すでに稼働している在庫管理アプリケーションがあったとします。このシステムには入荷待ちをしている営業所へのリアルタイムな入荷通知機能がないため、これをあとから追加したいと考えたとします。通常なら、すでに稼働中のシステムの入荷処理プログラムを変更して、入荷情報と入荷待ち営業所の情報とを照らし合わせる処理を追加することになるでしょう。そして、システム全体の総合試験を行い、稼働中のシステムを停止したうえでシステムを入れ替えることになると思います。

もしも、在庫情報を保管しているテーブル自体に、データが追加されたときに何らかの処理を行わせることができたらどうでしょうか？　これなら、既存のシステムをいっさい変更することなく、入荷待ち営業所への入荷通知機能を追加して実現することが可能になります。もちろん、それでもデータ上は有機的に結合しているのです。既存のシステムに手を入れない分、実現へのプロセスが簡素化できる可能性もあります。その気になれば、稼働中のシステムをまったく止めないで追加機能の運用を開始できるかもしれません。

これら2つの架空の想定に出てきた、「もしも…」の部分を実現するのが、今回紹介する「トリガ」なのです。実際には要求仕様もさまざまですから、これらの例では必ずトリガを使うべきだというわけではありませんが、トリガの特徴は大まかにわかってもらえたと思います。

トリガとは？

トリガとは、「あらかじめ決めておいた操作が行われたときに自動的に実行する処理のこと」を指します（**図1**）。なお、あらかじめ決めておいた操作のことを「イベント」と呼びます。PostgreSQLでは、テーブルを更新したときにトリガを起動させることができます。ここでいうテーブルの更新とは、次に挙げる操作のことを指します。

- **レコードの挿入（INSERT文）**
- **レコードの更新（UPDATE文）**
- **レコードの削除（DELETE文）**

たとえば、テーブルAにレコードを挿入した際に、そのレコードの内容に基づいてテーブルBにもデータを挿入したいといったときにトリガが利用できます。つまり、「テーブルAにレコードを挿入」をイベントとして、「テーブルBにもデータを挿入」というトリガを定義すればいいことになります。こうすることで、テーブルAへのレコードの挿入だけで、自動的にテーブルBにもデータが挿入されることになります。

また、PostgreSQLではトリガを起動するタイミングも次の2種類の中から指定できます。

- **イベントの直前（BEFORE）**
- **イベントの直後（AFTER）**

イベントの直前に起動するトリガを利用すれば、トリガの処理中にテーブルの更新をキャンセルさせることも可能です。たとえば、テーブルに挿入するレコードの内容をチェックして、条件によってはレコードの挿入をキャンセルしたい場合などは、イベントの直前に起動するトリガを利用することになります。というのも、イベントの直後に起動するトリガではすでにイベントが完了してしまっていますので、それをキャンセルすることはできないからです。この場合には、「テーブルにレコードを挿入」というイベントの直前に起動する、「レコードの内容をチェック」というトリガを定義すればいいことになります。

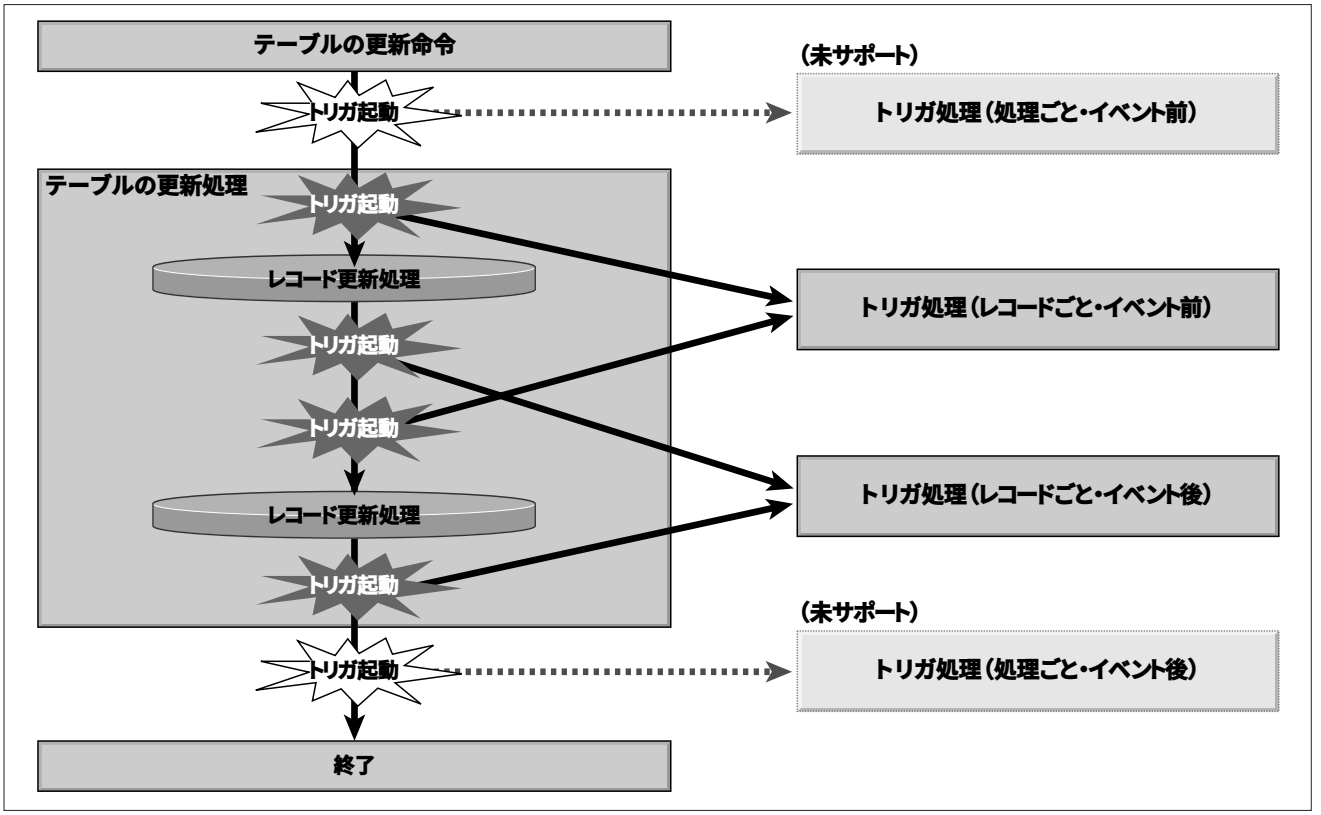


図1　トリガの概念

なお、PostgreSQLではトリガの起動単位に、次の2種類が規定されています。

・レコードごと（ROW）
・処理ごと（STATEMENT）

たとえば、レコードの更新を行うUPDATE文を考えてみます。UPDATE文は1回の実行で複数のレコードを更新する場合がありますが、目的によっては1回だけトリガを起動したい場合もあれば、更新するレコードごとに起動したい場合もあります。そのため、トリガの起動単位にはこのどちらかを指定することができる仕様になっています。しかし残念ながら、現在のバージョンのPostgreSQLでは、処理ごとに起動するトリガはまだ利用できません。

トリガは、非常に便利な機能なのですが、テーブルの更新内容を基に任意の処理が実行できるわけですから、誰でもトリガを作成できるようになっていた場合、テーブルのオーナーが気づかないうちにデータが盗まれるといったセキュリティ上の危険が発生してしまいます。このような理由から、PostgreSQLではテーブルの更新に対するトリガをそのテーブルのオーナーだけに作成権限を与えるように制限しています。

トリガの使い方

PostgreSQLのトリガでは、処理内容として関数名を指定します。つまり、トリガを登録するためには、あらかじめトリガの処理内容を記述した関数（トリガプロシージャ）を作成しておく必要があります。

■PL/pgSQL関数によるトリガプロシージャ

トリガプロシージャは、一般にユーザー定義関数として作成しますが、利用できる言語に制限があります。利用可能なユーザー定義関数は、具体的には以下の2種類のみとなっています。

・PL/pgSQL関数
・C関数

このうち、C関数によるトリガプロシージャの記述は複雑でわかりづらいと思いますので、今回はPL/pgSQL関数によるトリガプロシージャの記述方法のみを解説します。通常の利用目的なら、これだけで十分でしょう。

PL/pgSQL関数によるトリガプロシージャの作成は、以下のSQL文で行います。

```
CREATE FUNCTION 《トリガプロシージャ名》 ()
RETURNS OPAQUE
AS '《PL/pgSQL文》'
LANGUAGE 'plpgsql';
```

トリガプロシージャは、基本的には通常のユーザー定義関数と大差ないのですが、あえて異なる点を挙げると、

・返り値のデータ型が“OPAQUE”（不明なデータ型）
・関数の引数が1つも指定できない

ことがあります。ただし、トリガの引数という別形式でデータを渡すことは可能です。

また、トリガプロシージャの返り値は、トリガの起動元のテーブルと同じ構造のRECORD型（“テーブル名%ROWTYPE”指定も可）か、あるいはNULL値のいずれかになります。イベントの直前に起動されたトリガプロシージャがRECORD型の値を返した場合には、該当レコードがその値に更新されます（または、単に削除されます）が、NULL値を返した場合には該当レコードの更新は行われません。イベントの直後に起動されたトリガプロシージャの返り値は無視されます。

■利用可能なシステム変数

トリガプロシージャ内ではトリガ特有の情報を得ることができるようになっています。たとえば、レコードの更新が原因となって起動されるトリガの場合には、更新前のレコードの内容と更新後のレコードの内容にアクセスすることができます。このため、トリガプロシージャ内では普通のユーザー定義関数に比べ、いくつかのシステム変数が追加されます。次にそれぞれについて説明します。

・NEW（RECORD型）

更新後のレコード内容が格納されています。イベントが“DELETE”の場合にはこの変数は無効となります。イベントの直前に起動されるトリガの場合には、この変数はまだテーブルに格納されていない、新しいレコードの内容を意味します。

・OLD（RECORD型）

更新前のレコード内容が格納されています。イベントが“INSERT”の場合にはこの変数は無効となります。

- **TG_NAME（NAME型）**
  トリガの名称が格納されています。複数のトリガに対して同一のトリガプロシージャを流用する場合の、動作の切り替えなどに利用できます。

- **TG_WHEN（TEXT型）**
  トリガの起動タイミングが格納されています。“BEFORE”か“AFTER”のいずれかです。

- **TG_LEVEL（TEXT型）**
  トリガの起動単位が格納されています。“ROW”か“STATEMENT”のいずれかです。

- **TG_OP（TEXT型）**
  トリガの起動イベントが格納されています。“INSERT”、“UPDATE”、“DELETE”のいずれかです。

- **TG_RELID（OID型）**
  トリガの起動元テーブルのオブジェクトIDが格納されています。

- **TG_RELNAME（NAME型）**
  トリガの起動元テーブルの名称が格納されています。

- **TG_NARGS（INTEGER型）**
  トリガ登録時に指定した、トリガプロシージャへの引数の数が格納されています。

- **TG_ARGV（TEXT配列型）**
  トリガ登録時に指定した、トリガプロシージャへの引数が格納されています。なお、1番目の引数は、TG_ARGV[0]に格納されています。

これらのシステム変数が追加されているという点を除けば、トリガプロシージャも通常のユーザー定義関数と変わるところはありません。通常のユーザー定義関数としてトリガプロシージャを作成すればいいのです。**図2**にNEW、OLDシステム変数、トリガプロシージャの返り値の関係を示します。

■トリガの登録

トリガプロシージャが用意できたら、次にトリガを登録します。トリガの登録は次のSQL文で行います。

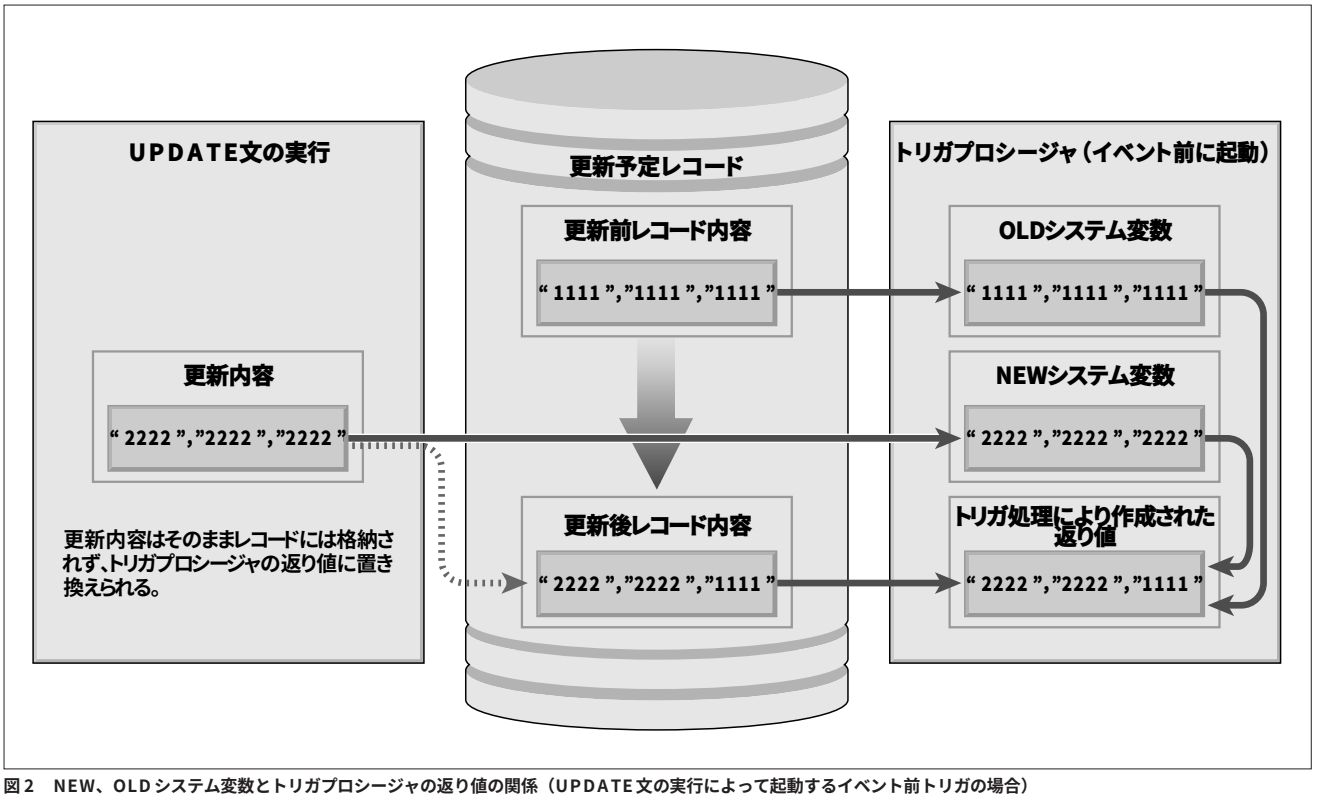


図2　NEW、OLDシステム変数とトリガプロシージャの返り値の関係（UPDATE文の実行によって起動するイベント前トリガの場合）

```
CREATE TRIGGER 《トリガ名》
《タイミング》 《イベント》 ON 《テーブル名》
FOR EACH ROW
EXECUTE PROCEDURE 《トリガプロシージャ名》(《トリガの引数》);
```

《トリガ名》には作成するトリガの名称を指定します。

《タイミング》にはトリガの起動タイミングを指定します。“BEFORE”か“AFTER”が指定でき、“BEFORE”、“AFTER”はそれぞれイベントの直前、イベントの直後にトリガを起動することを意味します。

《イベント》にはトリガを起動するイベントを指定します。“INSERT”、“UPDATE”、“DELETE”が指定でき、“OR”を挟むことにより複数のイベントを同時に指定することができます。“INSERT”、“UPDATE”、“DELETE”はそれぞれ、レコードの挿入時、レコードの更新時、レコードの削除時にトリガを起動することを意味します。

《テーブル名》にはイベントの対象とするテーブル名を指定します。このテーブルで発生したイベントが、トリガを起動します。

《トリガプロシージャ名》には、トリガの起動時に実行する関数名（トリガプロシージャ名）を指定します。

《トリガの引数》にはトリガプロシージャに渡したい引数を指定しますが、この値の受け取りにはTG_NARGSとTG_ARGVというシステム変数を利用しなければなりません。

なお、トリガの削除は次のSQL文で行います。

```
DROP TRIGGER 《トリガ名》 ON 《テーブル名》;
```

■トリガプロシージャの具体例

ここで、トリガプロシージャの例をいくつか挙げてみます。

リスト1はもっとも単純なトリガプロシージャの例です。このプロシージャはどのような条件のトリガにも利用でき、起動理由の表示だけを行います。なお、トリガの起動原因となったテーブル操作にはいっさい影響を与えません。

この例の中で、レコードの削除時（TG_OPが“DELETE”の時）にトリガプロシージャの返り値としてOLDシステム変数を返していますが、この返り値の内容自体には意味はありません。イベントの直前のトリガプロシージャでNULLを返してしまうとイベントがキャンセルされてしまいますので、そのようなことがないように手持ちのREDORD型変数を返しているにすぎません。

リスト2は、レコードの挿入や更新時に、percentという名称のカラムの内容が100を越えないように制限するトリガプロシージャです。なお、レコードの削除時に起動されても誤動作しないように、イベントの種類をチェックしています。

この例ではNEWシステム変数の内容を直接変更して、トリガプロシージャの新しい返り値としています。これにより、レコードの挿入や更新の内容の一部を差し替えることができます。このようにNEWシステム変数を直接変更

▼リスト1　起動理由を表示するだけのトリガプロシージャ

```
CREATE FUNCTION test1_tgr_fnc()
RETURNS opaque
AS '
BEGIN
    -- トリガの起動理由を表示
    RAISE NOTICE
        ''Trigger % was invoked % % on %.'',
        TG_NAME, TG_WHEN, TG_OP, TG_RELNAME;
    -- 終了
    IF TG_OP = ''DELETE'' THEN
        RETURN OLD;
    END IF;
    RETURN NEW;
END;
'
LANGUAGE 'plpgsql';
```

▼リスト2　percentカラムを100以下に制限するトリガプロシージャ

```
CREATE FUNCTION test2_tgr_fnc()
RETURNS opaque
AS '
BEGIN
    -- 削除イベントは無視する
    IF TG_OP = ''DELETE'' THEN
        RETURN OLD;
    END IF;
    -- 新しいレコードのpercent値が100を越えていたら
    -- 100に書き換える
    IF NEW.percent > 100 THEN
        NEW.percent = 100;
    END IF;
    -- 終了
    RETURN NEW;
END;
'
LANGUAGE 'plpgsql';
```

して利用しても問題はありません。というよりも、むしろそのほうが便利な場合がほとんどでしょう。もちろん、OLDシステム変数が有効なイベント（UPDATEイベントなど）の場合なら、同様にOLDシステム変数を利用しても問題はありません。

最後に、2つのテーブル間の矛盾を回避するトリガの例を示します。**リスト3**は、親テーブル（parent）のレコードが削除されたら、自動的に対応する子テーブル（child）のレコードを削除するトリガプロシージャです。このトリガプロシージャは、親テーブルのDELETEイベントに登録して利用します。

この例をうまく組み合わせれば、PostgreSQLにまだ実装されていない外部キーの機能をエミュレートすることも可能です（2000年にリリース予定のPostgreSQLバージョン7.0では、外部キーがサポートされる予定です）。

ここで例としてあげたトリガプロシージャを、実際にトリガとして登録してテストするサンプルが付属CD-ROMに収録されていますので、そちらも参考にしてください。

トリガの利用例

トリガの使い方についてはすでに解説しました。そこで今度は、この連載ではおなじみの「顧客名簿」データベースを例に取り上げて、これに更新記録の自動取得機能をトリガで実現する手順を解説しようと思います。**表1**に、顧客テーブル（customer）の構成を示します。

まず、更新記録を格納するテーブルが必要になりますので、更新記録テーブルを**表2**にしました構成で作成します。更新記録テーブルには、更新者や更新種別（“INSERT”、“UPDATE”、“DELETE”の区別）、更新日時、そのほかに更新前後の顧客テーブルの内容を格納します。

さて、更新記録の格納先が決まったところで、次は本題のトリガを作成しましょう。まず、トリガプロシージャの作成から始めます。

基本的には更新者、更新種別、更新日時と、更新前後の顧客テーブルの内容を記録するだけですから、やることは比較的単純です。**リスト4**がそのトリガプロシージャです。

このリストでは、トリガプロシージャの最初の部分ではイベントの種類を判定して分岐していますが、基本的な動作はどのイベントの場合でも変わりありません。更新記録テーブルに格納する内容が多少変わる程度です。特に難しいところはありませんので、すぐに理解できることと思います。

トリガプロシージャが作成できたところで、今度はデータベースにトリガを登録します。今回の場合は、顧客テーブルのすべてのイベントに対してトリガを登録すればいいのですから、SQL文は次のようになります。

```
CREATE TRIGGER customer_update_tgr
AFTER INSERT OR UPDATE OR DELETE ON customer
FOR EACH ROW
EXECUTE PROCEDURE customer_update();
```

▼リスト3　親が削除されたら子も削除するトリガプロシージャ

```
CREATE FUNCTION test3_tgr_fnc()
RETURNS opaque
AS '
BEGIN
    -- 削除以外のイベントは無視する
    IF TG_OP <> ''DELETE'' THEN
        RETURN NEW;
    END IF;
    -- 削除された親を指している子を削除
    DELETE FROM child WHERE child.parent_id = OLD.id;

    -- 終了
    RETURN OLD;
END;

'
LANGUAGE 'plpgsql';
```

カラム名	データ型	説明
ccode	datetime	顧客コード
name1	text	姓
name2	text	名
addr1	text	住所1（都道府県）
addr2	text	住所2
addr3	text	住所3
zip	text	郵便番号
email	text	電子メールアドレス

表1 顧客テーブル（customer）の構成

カラム名	データ型	説明
uuser	name	更新者
umethod	text	更新種別
udatetime	datetime	更新日時
old_ccode	datetime	更新前の内容
old_name1	text	更新前の内容
old_name2	text	更新前の内容
old_addr1	text	更新前の内容
old_addr2	text	更新前の内容
old_addr3	text	更新前の内容
old_zip	text	更新前の内容
old_email	text	更新前の内容
new_*		更新後の内容

表2 更新記録テーブル（updatelog）の構成

なお、このSQL文ではトリガの起動タイミングをイベントの直後（AFTER）としていますが、今回のような目的の場合は、イベントの直前であってもまったく問題ないでしょう。

これで更新記録の自動取得ができるようになったはずですので、それでは、さっそく試してみましょう。まず、初期状態の更新記録テーブルの内容を表示して、レコードがないことを確認します**（画面1）**。

次に顧客テーブルを更新してみます。レコードの挿入、レコードの更新、レコードの削除という、3つの操作を行います**（画面2）**。以上の操作で、3つの更新記録が作成されているはずですので、さっそく更新記録テーブルの内容を表示してみましょう**（画面3）**。ご覧のとおり、確かに記録されています。以上で更新記録が自動的に格納されるようになりました。

```
ascii3=> select uuser, umethod, udatetime
from updatelog;
uuser  |umethod|udatetime
-------+-------+----------------------------
(0 rows)
```

画面1 初期状態の更新記録テーブルの内容確認

```
ascii3=> -- 登録
ascii3=> insert into customer values (
ascii3->         30, '片岡', '裕生',
ascii3->         '東京都', '小金井市', '中町4',
ascii3->         '184-0012',
'kataoka@interwiz'
ascii3-> );
INSERT 147854 1
ascii3=> -- 更新
ascii3=> update customer
ascii3->         set name1 = 'カタオカ',
name2 = 'ヒロキ'
ascii3->         where ccode = 30;
UPDATE 1
ascii3=> -- 削除
ascii3=> delete from customer
ascii3->         where ccode = 30;
DELETE 1
```

画面2 顧客テーブルの操作

```
ascii3=> select uuser, umethod, udatetime
from updatelog;
uuser  |umethod|udatetime
-------+-------+----------------------------
kataoka|INSERT |Thu Nov 17 09:14:19 1999 JST
kataoka|UPDATE |Thu Nov 17 09:14:19 1999 JST
kataoka|DELETE |Thu Nov 17 09:14:19 1999 JST
(3 rows)
```

画面3 更新後の更新記録テーブルの内容確認

▼リスト4 顧客テーブルの更新記録を取るトリガプロシージャ

```
CREATE FUNCTION customer_update() RETURNS OPAQUE
AS '
BEGIN
    IF TG_OP = ''INSERT'' THEN
        INSERT INTO updatelog (
            uuser, umethod, udatetime,
            new_ccode, new_name1, new_name2,
            new_addr1, new_addr2, new_addr3, new_zip,
            new_email
        ) VALUES (
            USER, ''INSERT'', ''now'',
            new.ccode, new.name1, new.name2,
            new.addr1, new.addr2, new.addr3, new.zip,
            new.email
        );
        RETURN new;
    ELSE
        IF TG_OP = ''UPDATE'' THEN
            INSERT INTO updatelog (
                uuser, umethod, udatetime,
                old_ccode, old_name1, old_name2,
                old_addr1, old_addr2, old_addr3, old_zip,
                old_email,
                new_ccode, new_name1, new_name2,
                new_addr1, new_addr2, new_addr3, new_zip,
                new_email
            ) VALUES (
                USER, ''UPDATE'', ''now'',
                old.ccode, old.name1, old.name2,
                old.addr1, old.addr2, old.addr3, old.zip,
                old.email,
                new.ccode, new.name1, new.name2,
                new.addr1, new.addr2, new.addr3, new.zip,
                new.email
            );
            RETURN new;
        ELSE
            INSERT INTO updatelog (
                uuser, umethod, udatetime,
                old_ccode, old_name1, old_name2,
                old_addr1, old_addr2, old_addr3, old_zip,
                old_email
            ) VALUES (
                USER, ''DELETE'', ''now'',
                old.ccode, old.name1, old.name2,
                old.addr1, old.addr2, old.addr3, old.zip,
                old.email
            );
            RETURN old;
        END IF;
    END IF;
END;
'
LANGUAGE 'plpgsql';
```

最後に、権限の設定について説明しておきましょう。PostgreSQLのトリガでは、イベント操作を行ったユーザーの権限で実行されます。つまり、トリガを誰が登録しようとも、現在のデータベース利用者の権限で動作するということです。ですから、今回のような更新記録を自動的に格納するようなトリガを作成した場合には、格納先となるテーブルにも、一般利用者に対して、少なくともINSERT権限は与えておく必要があります。更新記録テーブルは、通常はレコードの追加しか行われないので、INSERT権限だけあれば十分でしょう。

さあ、使ってみましょう

例によって、今月の付属CD-ROMにも、今回の解説で出てきたデータとソースコードすべてが収録されています。また、それらを簡単にセットアップして実行するためのSQLファイルも用意してあります。詳しくは、付属CD-ROM内のREADME.txtを参照してください。

次回は、クライアント間通信が実現できる通知機能（NOTIFY／LISTEN）について紹介します。

Column

ルール

ルールといっても、ここで何か決めごとを話そうとしているわけではありません。PostgreSQLには、トリガに似た機能を持つ「ルール」という機能があります。このルールについても簡単に紹介しておきます。

ルールを利用すると、トリガと同様にあらかじめ決めておいたイベントによって処理を起動させることが可能です。トリガとの相違点は細かく述べるときりがない（というかまったく異なる？）のですが、注意すべき点としてあえて挙げるなら、トリガならCOPY文によっても起動されますが、ルールは実行されないということがあります。

■ルールの作成

このルールを作成するには、次のSQL文を利用します。

```
CREATE RULE 《ルール名》
AS ON 《イベント》 TO 《テーブル名》 [WHERE 《条件式》]
DO [INSTEAD] 《SQL文》;
```

《ルール名》には作成するルールの名称を指定します。

《イベント》にはルールを実行するイベントを指定します。“SELECT”、“INSERT”、“UPDATE”、“DELETE”のいずれかを指定しますが、“SELECT”は現在のPostgreSQLでは利用できません。“INSERT”、“UPDATE”、“DELETE”はそれぞれ、レコードの挿入時、レコードの更新時、レコードの削除時にルールを実行することを意味します。

《テーブル名》にはイベントの対象とするテーブル名を指定します。このテーブルで発生したイベントが、ルールを実行します。

“WHERE”キーワードとともに《条件式》を指定した場合には、その条件か満たされた場合にのみルールが実行されるようになります。

“INSTEAD”キーワードを指定すると、イベントを実行の代わりにルールを実行します。

《SQL文》にはルールで実行するSQL文を指定します。SQL文の代わりに“NOTHING”を指定すると、何も行わないルールになります。また、複数のSQL文を“;”（セミコロン）でつないで全体を“(”～“)”（かっこ）で囲むことにより、複数のSQL文をルールに実行させることも可能です。

なお、ルールでもトリガと同様に、NEWとOLDという変数が《条件式》と《SQL文》の中で利用できます。

■ルールの使用例

リストc-1は、顧客テーブルが更新されたとき、更新記録にその内容を自動的に記録するルールです。このルールは、トリガとほぼ同様な機能を提供してくれることをわかってもらえるでしょう。なお、紙数の都合でリストc-1にはUPDATEに対するルールのみを紹介していますが、付属CD-ROMにはINSERTやDELETEの分も含まれています。

ルールにはもっと面白い使い方があるのですが、紙数も尽きましたので、続きはまたいずれということにします。

▼リストc-1　ルールによる更新記録の自動作成

```
CREATE RULE customer_update_rule
AS ON UPDATE TO customer
DO INSERT INTO updatelog (
    uuser, umethod, udatetime,
    old_ccode, old_name1, old_name2,
    old_addr1, old_addr2, old_addr3, old_zip,
    old_email,
    new_ccode, new_name1, new_name2,
    new_addr1, new_addr2, new_addr3, new_zip,
    new_email
) VALUES (
    USER, 'UPDATE(RULE)', 'now',
    old.ccode, old.name1, old.name2,
    old.addr1, old.addr2, old.addr3, old.zip,
    old.email,
    new.ccode, new.name1, new.name2,
    new.addr1, new.addr2, new.addr3, new.zip,
    new.email
);
```

Rubyで行こう

今回は「テキスト処理」をテーマに、Rubyプログラミングの実際を紹介していきます。いきなりgrepコマンドを作ってしまいますが、そこはRubyですから簡単かつ美しいです。さあ、Rubyを始めましょう。

第2回　ユー・ガット・Ruby

文：赤松智也
Text : Tomoya Akamatsu

プログラミング言語を紹介する連載記事の場合、どのように説明していくかかなり難しいところです。機能を順番に説明していくという方法もあるでしょうが、それではずいぶん退屈な記事になってしまいます。そこで、この連載では、言語仕様についてはあまり詳細に説明せず、具体的なプログラムを紹介することで、とにかく雰囲気をつかんでもらおうと思います。幸い、『オブジェクト指向スクリプト言語Ruby』（通称、Ruby本）が出版され、売れ行きも好調のようですから、言語機能の詳細を知りたい方は、とりあえずそちらを参照してください。

Rubyの魅力

この連載は、オブジェクト指向スクリプト言語「Ruby」を紹介するものですが、そもそもRubyの魅力っていうのは何でしょうか？　ま、いろいろあるとは思うのですが、主なものは、次のようなものが考えられます。

・便利

Rubyは本格的な機能を持つスクリプト言語です。ですから、Perlの持つ「気軽さ」「手軽さ」「手早さ」などの利点を（資産の蓄積を除いて）すべてを備えており、日常的な処理を簡単にプログラムできます。プログラムをサッと書いて、サッと実行できる開発サイクルの短さもうれしいですね。

・使う身になった言語仕様

Rubyは使って楽しい言語です。いろいろ理由はあるのでしょうが、おそらく一番は「オブジェクト指向」というひとつのパラダイムで統一された言語仕様だからでしょう。それと、Perlの経験を踏まえた便利なクラスライブラリの存在も大きいです。まつもとさんによると、Rubyをコンピュータとの優れたインターフェイスとなる言語として設計したということです。

・新しい

Rubyは新しい言語です。といっても、実際には開発が始まってからもう6年にもなるそうですから、本当はそれほど新しくはないのかもしれませんが、話題になるようになったのはつい最近ですよね。新しもの好きとしては見逃せない魅力があります。

・国産

国産を強調するのは、作者のまつもとさんはあまり喜ばないようですが、やはり我々日本人にとって国産というのは魅力です。作者と日本語で対話できるというのもうれしいですし、日本語処理に困らないというのも長所です。何といっても世界に通じるプログラム言語が国産というのが「日本人の誇り」をくすぐるじゃありませんか。

・友好的なコミュニティ

Rubyの開発とサポートは主にruby-listというメーリングリストで行われています。ここでは、まつもとさんを始めとする多くの人たちによって活発なやりとりが行われています。参加者は750名を越えているそうです。このメーリングリストは、妙に難しいことが語られたりするんですが（突然パッチが流れたりする）、初心者の簡単な質問にも親切なフォローが行われます。Rubyコミュニティ全体にこのようなフレンドリーな雰囲気があります。それと、とにかくまつもとさんが出すメールの数は尋常ではないです。

正直なところ、Rubyはまだまだ若いので、先輩にあたるPerlやPythonに比べて資産の蓄積も少なく、ドキュメントも豊富とはいえないのですが、言語仕様、コミュニティそれぞれに何ともいえない「楽しさ」が感じられるので、今後にとても期待できると思います。

テキスト処理とRuby

「Rubyは何に使えるか」と聞かれれば、「何にでも使えます」という答えになります。そういうプログラムを書けば、ということですが。しかし、それではあまりにも漠然としていますので、紹介の手始めとして、基本となるテキスト処理から紹介しましょう。

元来、UNIXはキャラクタベースのインターフェイスが基本といわれていたものです。テキストデータに対して選択、加工などの処理を行い、新しいテキストデータを生成することが中心でした。また、テキストデータ加工プログラム（フィルタと呼ぶことが多い）を複数組み合わせて、結果を得るのが「カッコイイ」やり方といわれていました。

最近は、UNIX系でもGNOMEやKDEといったグラフィカルなインターフェイスが台頭しているようで、昔ほどテキスト処理が中心とはいえなくなっているかもしれませんが、便利さは今でも変わっていないといえるでしょう。Rubyは、そういったUNIXが持つテキスト処理の精神を（Perl経由で）受け継いでいます。

画面1 「本物の」grepコマンドの実行例
本物のgrepコマンド（正確には、日本語対応のjgrepコマンド）を実行したところ。カレントディレクトリにあるテキストファイルの中で、検索文字列に指定した「Ruby」を検索し、その該当行がファイル名つきで表示されている。-iオプションも付加しているので、「大文字／小文字は区別しない」で検索され、「ruby」にもマッチする。

■いきなりRubyでgrepを作る

では、実際にテキスト処理の実際を見てみましょう。みなさんは、grepコマンドをご存知ですか？ grepは、フィルタコマンド「sed」の“g/re/p”という命令から名付けられたといわれているコマンドで（同様の働きをします）、正規表現を使ってファイルの内容を検索するものです。正規表現というのは、一種のワイルドカードのようなもので、単なる文字列の一致以上のパターンを表現するものです。

このgrepコマンドをRubyで実装した骨格部分となるのが**リスト1**です。実際のgrepコマンドには、もっとたくさんのオプションが用意されていますが、仕組みを理解しやすくするため、ここではすべて省略しています。なお、リスト中の「#」より後ろの部分はコメントです。このRubyで作ったgrepコマンドの仕様説明を省略した代わりに、少々ていねいにコメントをつけてあります。

このgrepコマンドは、実質6行のプログラムです。これをCなどで実装しようとすると、これくらいで収まるはずもなく、なかなかの分量になることでしょう。これだけでも、Rubyの記述力を感じてもらえるでしょう。

読者が、何らかの言語でのプログラミング経験があれば、“gets()”という部分がファイルを1行ずつ読み込むことと、“=~”が正規表現マッチの演算子であることさえわかれば、このプログラムの働きを理解するのは簡単でしょう。ただし、

```
pat = /#{ARGV.shift}/
```

の部分については説明が必要でしょう。これは、ARGVという名前の配列に入っているコマンドライン引数から、先頭の一要素を取り除き、その取り除いた値を埋め込んだ正規表現オブジェクトを新しく作り出して、変数patに代入する、という意味です。

ここで、“ ARGV.shift ”というのは「ARGVというオブジェクトのshiftというメソッドを呼び出す」ことを意味します。Perlでいう“ shift(@ARGV) ”と同じ意味ですが、Rubyではオブジェクト指向風の記法で統一されています。また、“ // ”は正規表現オブジェクトを表す式です。さらに、Rubyでは文字列や正規表現の中に“ #{} ”で囲んだ式を埋め込むことができます。

変数patが参照しているのは正規表現オブジェクトで、変数lineに代入されるのは、各行の内容に対応する文字列オブジェクトです。このようにRubyではあらゆるデータがオブジェクトになっています。この例では、あまりうれしさが感じられないと思いますが、読み進んでいけば少しずつ明らかになるでしょう。

変数lineが参照していた文字列オブジェクトは、次の行の内容が読み込まれ、どこからも参照されなくなると、インタプリタが自動的に後片付けしてくれます。これは「ガベージコレクション」と呼ばれ、Javaなどにも採用されている機能です。ガベージコレクションの詳細はここでは述べませんが、プログラマーの負担を減らすたいへん便利な機能です。

■grepの起動

プログラムを作ったら動かしてみないと。Rubyのプログラムを実行する方法はいくつかあります。一番簡単なのは、Rubyインタプリタのコマンドライン引数として指定することです。

```
% ruby grep.rb pattern file1 file2
```

これで、Ruby版のgrepが実行されます。また、UNIXでは次のようにchmodコマンドを使ってファイルに実行フラグをセットすると、スクリプトファイル名だけで直接実行できます。

```
% chmod a+x grep.rb
% grep.rb pattern file1 file2
```

ただし、この方法の場合スクリプトの先頭に“ #!/usr/bin/ruby ”のように、インタプリタへのパスが指定されていることが必須条件となります。

このほかにも、スクリプトファイルを用意しないで、コマンドライン引数に直接スクリプトを指定する方法もあります。この方法は、1行に収まるような「ワンライナー」と呼ばれるスクリプトによく使われます。ワンライナーの詳細については218ページからのコラム**「ワンライナー」**を参照してください。

■プログラマブルな喜び

▼リスト1　grep.rb

```
#!/usr/bin/ruby
# ファイル先頭の #! は実行するインタプリタ名を指定する方法

# grep.rb pattern file...

pat = /#{ARGV.shift}/          # 先頭の引数の内容を埋め込んだ正規表現を作る
while line = gets() # 引数からなる仮想ファイルから1行ずつ読み込む
  if pat =~ line    # 読み込んだ行が正規表現にマッチしたら
                              # 「=~」はマッチ演算子
    print line                # その行を出力する
  end
end
```

Column

ガベージコレクション

Rubyのプログラムでは、文字列や正規表現などたくさんのオブジェクトが作られます。これらのオブジェクトは、変数や別のオブジェクトなどいろいろなところから参照されます。これらのオブジェクトを作ったまま放置しておくと、だんだんメモリが足りなくなってしまいます。しかし、だからといってまだ参照されている（「生きている」ともいいます）オブジェクトのメモリを解放してしまうと、致命的な問題が発生してしまいます。

このため、CやC++ではfree()とかdeleteのような手段によってプログラムが明示的にオブジェクトの解放を指示してやる必要があります。しかし、これではすべてのオブジェクトがいつまで「生きている」のかということを、プログラマーがすべて把握しておかなくてはなりません。しかし、オブジェクト指向プログラミングでは同時に数百のオブジェクトが活動することも珍しくなく、それらをすべて把握するのは、プログラマーの負担が重すぎます。

快楽プログラミング言語のRubyでは、インタプリタがこの負担をカバーしてくれます。インタプリタは、ときどきメモリ全体をスキャンして、参照されなくなったオブジェクトを発見して、そのメモリを自動的に解放してくれます。ですから、プログラマーはオブジェクトの後始末についてまったく心配する必要がないのです。

この快適なガベージコレクションの機能は、Rubyだけでなく、Java、Lisp、Smalltalkなどの言語でも提供されています。

この6行のプログラムは、とても単純なものです。しかし、これがインタプリタに与えるプログラムであり、簡単に修正、拡張ができるという点は、Rubyの特長を理解するうえでとても重要なことです。この6行のプログラムを基に、自分の好きな機能を持つgrepコマンドを作ることができます。たとえば、マッチした部分を強調したり、マッチした行以外にも前後の数行を表示させるなどということも、比較的簡単に行えます。

これこそが、スクリプト言語の最大の利点だといえるでしょう。grepというコマンドにはたくさんのオプションがありますが、そのオプションだけではカバーできないような機能を実現するのは簡単ではありません。ですが、スクリプトならそんな制限はありません。自分でプログラムすることで、最大限の自由を得ることができるのです（少々おおげさですが）。

■日本語grep

歴史的な事情といえばそれまでですが、不幸なことに日本では複数の文字コードが横行しています。主要なものはJISコード、EUC（Extend Unix Code）、SJIS（シフトJIS、マイクロソフト漢字コード）です。いろいろ不備はあるといわれつつも、Unicodeがこの問題を解決してくれるかとも思われましたが、やはり「もうひとつ別のコード系」を導入するだけで終わってしまいそうな気配です。

しかし、愚痴をいうばかりでは仕方がありません。ここでは、「プログラマブルな喜び」を少々行使して、日本語を表現するのに用いられる3つの文字コード（EUC、SJIS、JIS）を区別なく検索できるgrepコマンド「jgrep.rb」を作ってみましょう。**リスト2**がそれにあたります。

実質3行というほんのわずかの修正で、文字コードをEUCにそろえて比較することができるようになりました。これが「プログラマブルなうれしさ」です。変更になった3行は以下の部分です。

```
require 'kconv'        # 文字コード変換ライブラリ
```

まず、文字コード変換ライブラリをロードします。このライブラリをロードすることによって、Kconvというモジュールが定義されます。

```
pat = /#{Kconv::toeuc(ARGV.shift)}/e
```

変更前のこの部分はこうでした。

オプション	説　明
s	SJISとみなす
e	EUCとみなす
u	UTF-8とみなす
n	多バイトコードを無視する
i	大文字/小文字を区別しない
o	式展開を最初の一度だけ行う
x	正規表現中の空白を無視する。コメントをつけられる
p	POSIX行マッチ、改行も通常文字とみなす

表1　正規表現のオプション

▼リスト2　jgrep.rb

```
#!/usr/bin/env ruby
# この指定方法だとrubyがPATH上のどこにあっても大丈夫

# jgrep.rb pattern file...

require 'kconv'                # 文字コード変換ライブラリ

pat = /#{Kconv::toeuc(ARGV.shift)}/e
# まずパターンをEUCに変換する
# 元の文字コードは勝手に推測する
# 正規表現にもEUCオプションを付けておく
while line = gets()
  line = Kconv::toeuc(line)
  # lineもEUCに変換する
  if pat =~ line
    print line
  end
end
```

▼リスト3　rgrep.rb

```
#!/usr/bin/env ruby

require 'find'                 # ファイルに対する再帰処理のライブラリ

# rgrep.rb pattern

pat = /#{ARGV.shift}/          # grep.rbと同様
Find::find(".") do |path|
  # カレントディレクトリ以下の全ファイル名に対して処理を行う

  next unless File::file?(path)
  # pathのファイルが通常ファイルでなければ検索しない
  open(path)do |f|  # ファイルのオープン
    while line = f.gets()
      if pat =~ line
          print path, ":", line, "\n"
      end
    end
  end                          # 自動的にcloseされる
end
```

```
pat = /#{ARGV.shift}/
```

前にも述べたように“ #{} ”の中には任意の式が埋め込めます。シェルやPerlにも「変数展開」という機能がありますが、単なる変数だけでなく、任意の式を埋め込めるのがRubyの特長のひとつです。

正規表現の末尾にある“ e ”は「文字コードをEUCとみなして正規表現の比較を行う」ことを示すためのオプションです。正規表現のオプションにはこのほかにも、文字コードをシフトJISだとみなす“ s ”、UTF-8とみなす“ u ”、多バイト文字を考慮しない“ n ”、アルファベットの大文字／小文字を無視して比較する“ i ”、正規表現中の式展開を最初の一度しか行わない“ o ”などがあります。オプションについては**表1**を参照してください。

最後の修正行は、次のとおりです。

```
line = Kconv::toeuc(line)
```

“ Kconv::toeuc ”は、元の文字コードは適当に推測して、文字列をEUCに変換してくれます。このkconvライブラリが標準で添付されていることや、正規表現に対する文字コードを指定する機能などは、文字コードで苦労することの多い日本人が作った言語ならではという配慮を感じさせてくれます。

なお、文字コードの変換といえば、Unicodeに対する変換機能を提供するuconvというライブラリも存在しています。こちらは標準添付ではありませんが、

```
http://www.bekkoame.ne.jp/~yoshidam/Ruby_ja.html
```

から入手可能です。

■再帰的grep

リスト3に示すのはgrep.rbの基本部分を流用して作った、コマンドライン引数で指定した正規表現のパターンを、カレントディレクトリ以下の全ファイルに対して検索する「再帰的grep」です。このサンプルでは、ディレクトリの再帰処理のためにRubyに標準添付されているfindライブラリを使っています。

findライブラリは、Findモジュールによってディレクトリ内のファイルに対する再帰的なアクセス機能を提供します。“ Find::find ”は、引数で指定したディレクトリ以下の各ファイル（ディレクトリを含む）に対して、後ろにつけられた“ do … end ”の部分を実行します。このような“ do … end ”の部分を「ブロック」と呼びます。また、“ do … end ”の代わりに“ { … } ”と書くこともできます。この2つの形式の違いは、外見と結合優先度の違いだけで、意味は同じです。

“ Find::find ”のようなブロックを受け取るメソッドは、その内部からブロックを呼び返すことができます。メソッドから与えられた値は、“|| ”の間にはさまれた変数に代入されます。“ Find::find ”は、各ファイルに対してブロックを呼び出すことにより、再帰的なアクセスを提供しています。この“ Find::find ”のようにブロックを使って繰り返しを実現するようなものを「イテレータ」と呼びます。

なお、“ Find::find ”の“ :: ”は、「Findモジュールのfindメソッドを呼び出す」という意味で、“ Find.find ”とまったく同じ意味です。Rubyでは、クラスやモジュールのメソッドを呼び出すことを明示したい場合、“ . ”の代わりに“ :: ”を使うことが慣習となっています。

“ Find::find ”によって各ファイルに対して検索を行うことになりますが、通常ファイルではないディレクトリなどに対して検索を行うのは無駄なことです。そこで、

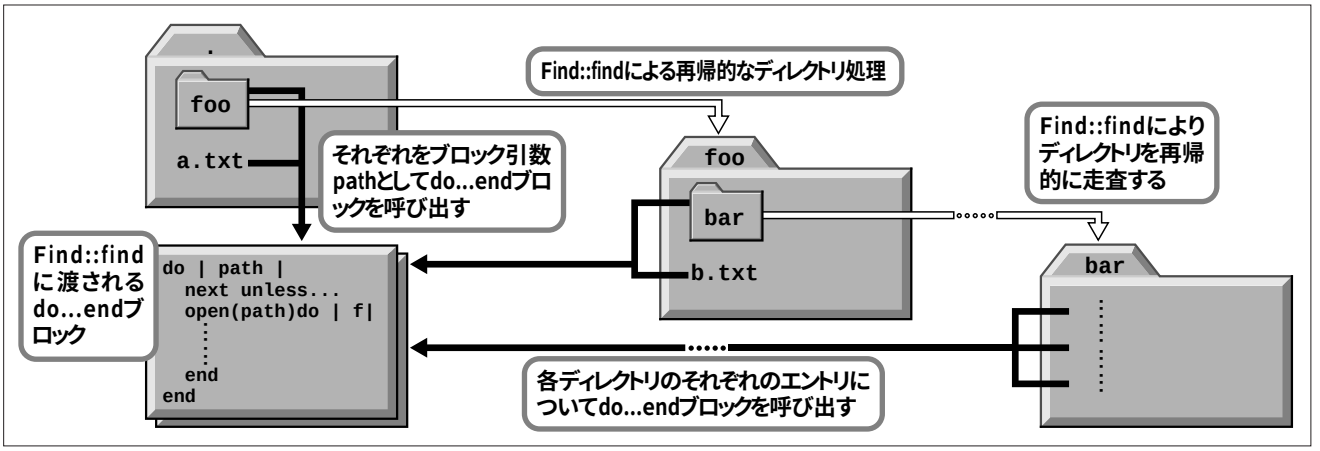


図　イテレータの概念

```
next unless File::file?(path)
```

という行で通常ファイルでないものを検索の対象から外しています。nextは、ループの次の繰り返しにジャンプします（Cにおけるcontinueと同じ働きです）。また、unlessはifの逆です。Perl同様、Rubyでも「if、unless、while、until」は、文の後ろにつけられます。

“File::file?(path)”は、ファイルが通常ファイルかどうか判定しています。このように真偽値を判定する関数名の末尾に“?”をつけるのもRubyの慣習のひとつです。

openにもブロックがくっついていますが、これは繰り返しを行っているのではありません。

```
open(path) do |f| ... end
```

は、

```
f = open(path)
...
f.close()
```

とほぼ同じ働きをします。ブロックを与えることで、そのファイルの有効範囲を明示して、closeを省略することができるわけです。このようにブロックは、繰り返し以外にもいろいろな用途に使われます。

このrgrep.rbは、カレントディレクトリ以下のあらゆるファイルに対して検索を行います。実際にはディレクトリ以外にも、実行ファイルやオブジェクトファイルなど検索しても意味のないバイナリファイルもありますから、それらについても絞り込みを行えるほうが便利でしょう。ま、この絞り込み機能については宿題としましょう。

たとえば、「“.o”の拡張子を持つファイルは検索しない」というルールなら、わずか1行の追加で実現できるでしょう。内容を読み込んでのバイナリファイルの判定はそれよりも少々難しいですね。

▼リスト4　killall.rb

```
#!/usr/bin/env ruby

if ARGV[0] and ARGV[0][0] == ?-
  # ARGVはコマンドライン引数の入る配列
  # コマンドライン引数があり、その最初の文字が - だったら
  # (?- は '-' という文字を表す)、killのオプションとみなす

  killopt = ARGV.shift     # 配列の先頭の要素を取り出す(同時に取り除く)
end

killed = false
cmd = ARGV.shift  # 配列の先頭の要素を取り出す(同時に取り除く)
`ps www`.each do |proc|
  # ps wwwを実行した結果を文字列として取り出し、
  # その各行に対して繰り返す

  proc.chomp!              # 文字列末尾の改行を取り除く
  data = proc.split        # 文字列をスペースで分割する
  proc = data[4]  # (0から数えて)4番目の要素がプロセス名
  if cmd == proc  # 引数で指定したコマンド名とプロセス名が一致したら
    pid = data[0] # pidは0番目の要素
    system format("kill %s %s\n", killopt, pid)
    # ここではkillコマンドでシグナルを送ることにする
    # Ruby自身でもシグナルを送ることができる
    killed = true # 「シグナルを送った」フラグをセット
  end
end

unless killed              # シグナルを送っていなければメッセージを出力
  printf "#{$0}: no process killed\n"
end
```

さらに応用例

リスト4のプログラム「killall.rb」は、プログラム名を指定してプロセスをkillするものです。Linuxでは、同様のプログラムが「killall」という名前で提供されています。このkillall.rbは、プロセスの状態を得るpsコマンドの出力からプログラム名とプロセス番号を抽出して、killコマンドによりプロセスにシグナルを送ります。

このプログラムに関してはもうほとんど説明は不要でしょう。詳細まではわからない部分はあったとしても、大まかな動作の雰囲気は何となくつかめるのではないでしょうか？

ただ“proc.chomp!”については、補足説明しておきましょう。前述したとおり、真偽値を返すメソッド名が“?”で終わる慣習があるのと同様に、Rubyでは、「破壊的な」メソッドとそうでないメソッドの両方があるとき、破壊的なメソッドのほうの名前の末尾に“!”をつける慣習があります。

この「破壊的」という概念は少々難しいのですが、Rubyの理解の基本となります。特に、PerlやBASICなど他の言語に慣れ親しんだ人がよく引っかかるようですから、特別に説明しましょう。

まず、「Rubyの変数はすべて参照である」ことに注目してください。C言語風に表現するなら、「すべてがポインタ変数である」とでもいえばよいでしょうか。この辺りは、Lispなどを知っている方なら自然に感じるかもしれません。

ですから、

```
a = "foo\n"
b = a
```

は、変数a、bともに同じ文字列オブジェクトを指し示します。Perlなどで代入を行うと別の文字列変数に内容がコピーされますが、Rubyの代入は単に変数が指している先を変更するだけです。この原則は、変数の内容がどんな複雑なオブジェクトであっても、整数や文字列のような基本的なデータであっても差別はありません。

さて、上記の2つの変数a、bは同じ文字列オブジェクトを指しているのですから、文字列の内容が変化すれば、両方が同時に変化したように見えます。このように「オブジェクトの内容や状態を変更してしまうようなメソッド」を「破壊的なメソッド」と呼びます。この「破壊」というのは状態が変わってしまうという程度の意味で、コンピュータやプログラムが壊れてしまうわけではありませんのでご心配なく。

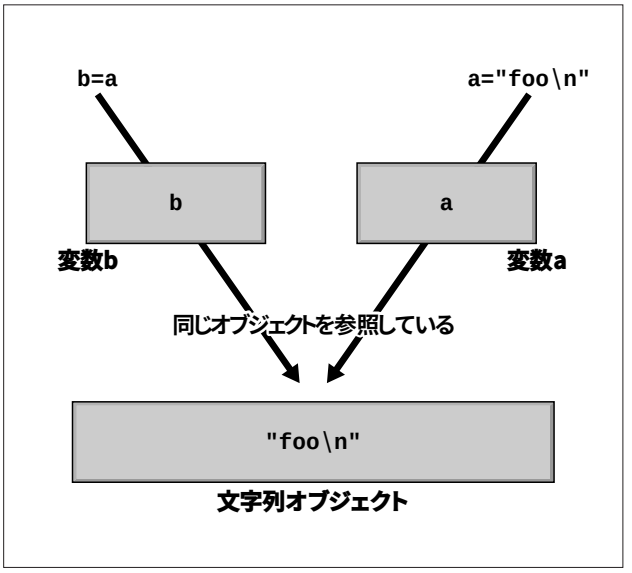


図　Rubyにおける変数の概念

Ruby本について

『Ruby本』読まれましたか？　早々と増刷が決まったそうで、めでたい限りです。この本でRubyの普及がますます進めばよいですよね。しかし、この本は何というか「盛りだくさん」で、うっかりすると消化不良を起こしそうです。とりあえず「用語集」を楽しみつつ、ゆっくり読んでいくことをお勧めします。

残念ながら、第1刷の『Ruby本』には間違いがかなりたくさんありました。第2刷ではその大部分が修正されているそうですが、私を含めて第1刷を入手してしまった人のために正誤表が以下で公開されています。

```
http://www.ruby-lang.org/ja/book/errata/ruby.html
```

さすがRuby、バージョン（刷）が進むたびにバグが見つかって、どんどん減っていくのはプログラムも書籍もいっしょだな、なんて不謹慎なことを考えてしまいました。まつもとさん、ごめんなさい。

Column

ワンライナー

スクリプト言語には「ワンライナー」と呼ばれる分野があります。元々はAWKの世界で始まった呼び名だと思いますが、コマンドラインで直接指定できる程度の小さなプログラムでいろいろ便利なことができるというものです。

Rubyでプログラムをコマンドラインから指定するためには、-eオプションを使います。

```
% ruby -e 'print "hello world\n"'
hello world
```

ワンライナーのために便利なのは-nオプションと-pオプションがあります。-nオプションは、引数で指定した各ファイルから1行ずつ読み込み、各行を“$_”という変数に代入して、プログラムを繰り返し実行するというものです。

```
% ruby -ne 'print $_' file1
```

このスクリプトは、file1の内容を出力します。catコマンドと同じですね。このプログラムは、

```
while $_ = gets()
   print $_
end
```

と同じ意味です。一方、-pオプションは-nオプションと同様に各行に対して繰り返しますが、ループの末尾で変数$_の内容をprintします。ですから、

```
% ruby -pe '' file1
```

は、上記のプログラムと同様にfile1の内容を出力します。

■ワンライナーgrep

これらの機能を使うと小さなプログラムで驚くほど大きな処理が実行できます。たとえば、

```
% ruby -ne 'print $_ if /pattern/ =~ $_'
```

とすると、本文で紹介したgrepプログラムと同じ動作をします。さらに「条件式での正規表現は$_とのマッチをとる」、「引数が省略されたprintは$_の内容を出力する」というルールがありますから、

```
% ruby -ne 'print if /pattern/'
```

だけでgrepできます。もっともこのような省略時の動作は慣れないとわかりにくいプログラムになるばかりですから、できることだけ憶えておいて、あまり多用しない方が幸せかもしれません。

あ、そうそう。このプログラムは

```
% perl -ne 'print if /pattern/'
```

としても、まったく同じように動きます。やはり、注意した方がよさそうですね。(笑)

■単語の置換

Rubyは、Perlの真似をしている部分がたくさんありますから、Perlという単語をRubyに置き換えるという遊びをしてみましょう。以下の文章をRuby用に変換しましょう。

```
%cat good.txt
Perlは素晴らしい。Perlのプログラムは簡潔で高機能だ。これからはAWKなんかよりPerlの時代でしょ。

% ruby -pe '$_.gsub!(/Perl/, "Ruby")' good.txt
Rubyは素晴らしい。Rubyのプログラムは簡潔で高機能だ。これからはAWKなんかよりRubyの時代でしょ。
```

さらに-iオプションを使うと「その場変換」ができます。

```
% ruby -i.bak -pe '$_.gsub!(/Perl/, "Ruby")' good.txt
```

今度はなにも出力せずに終了してしまいました。実は、これでgood.txtの中身が書き換わっています。古いファイルの内容はgood.txt.bakに保存されています。

■文字の置換

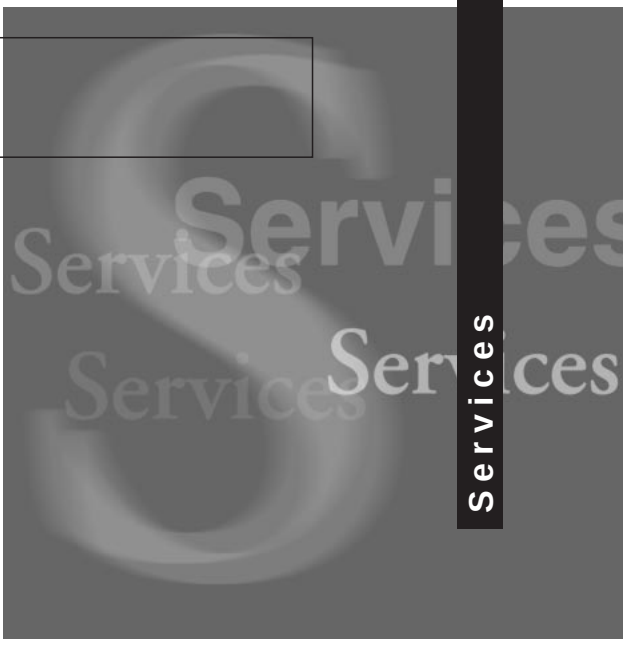
上の文章の英文字を全角に変換するためには、以下のようなワンライナーが使えます。

```
% ruby -r jcode -pe '$_.tr("A-Za-z", "Ａ-Ｚａ-ｚ")' good.txt
Rubyは素晴らしい。Rubyのプログラムは簡潔で高機能だ。これからはＡＷＫなんかよりＲｕｂｙの時代でしょ。
```

ここで利用している、jcodeというのはバイト単位ではなく文字単位で置換を行うためのライブラリです。残念なことにこのスクリプトは最新のRuby 1.4.3でないとバグのため動作しないようです。

Sambaと闘う（第3回）

今回は、SambaのアップグレードとWebベースで設定、管理が行える「SWAT」を紹介する。



SambaのアップグレードとSWAT

文：梅原 系
Text : Kei Umehara

この連載では、一応英語版Red Hat Linux 6.0と、それに添付されているSamba 2.0.3-8を使ってSambaのインストールを紹介してきたが、マイペースでのんびりしていたら、Sambaのインストールだけで2カ月もかかってしまった。この間にSambaの最新バージョンは2.0.6へと、着実に進化している。まあ、連載開始時点でも少し古いバージョンをインストールしたわけだから、最新バージョンとは開きがあるのも当然なのだが、この辺りで、Sambaのアップグレード作業を覚えておくことにしよう。そんなに難しい作業ではないので、怖がる必要はない。

今回のもう一つのテーマはSWATだ。これはSambaをWebベースで管理するためのツールで、これを使うとSambaの管理が非常にわかりやすくなる。もちろん、「エディタで設定ファイルを書き換えるほうがシンプルで間違いがない」というポリシーの読者もいるだろうが、せっかく便利なツールが用意されているので、これもセットアップして、使えるようにしておこう。

Sambaのアップグレード

まずはSambaのアップグレードだ。Sambaは現在も数カ月に1回程度のペースで新しいバージョンがリリースされている。必ずしも最新版を使う必要はないが、バグフィックスや機能強化などによって着実に進化しているので、時々は最新版の機能やバグフィックスの状況を確認して、適当なタイミングでバージョンアップ作業を行いたいものだ。もちろん、Linuxそのものをアップグレードするタイミングでsambaを一緒に…というのも、ぐうたら管理者としてはいいアプローチだ。Linuxのたいていのディストリビューションは半年程度のインターバルでバージョンアップしているし、本誌などの付属CD-ROMなどからも入手できる。なにより、面倒な作業をほとんどしなくてすむのがいい。読者の性格や暇かげん、使っているディストリビューションなどに応じて、最適なやり方を採ってほしい。

どのSambaにアップグレードするか

Sambaのオリジナルは、下記の本家Sambaサイトから入手できる。ただし、エンドユーザーとしては、別の経路から入手したほうが少ない手間で作業がすむことも多いので、アップグレードするためのソフトウェアをどこから入手するかをきちんと検討しておくことが先決だ。

■本家Sambaサイト（http://www.samba.org/）

本家Sambaサイトからは、オリジナルのSambaのソースコードと、主要なOS（ディストリビューション）向けのバイナリが入手できる。ドキュメントなどはすべて英語なので敷居は少々高いが、間違いなく、もっとも早くSambaの最新版を入手できる。

■ボランティア（など）によるSambaディストリビューション

Sambaについては、ボランティアベースで独自のディストリビューションが作成されていることもある。特に有名なのが、小田切氏らによる「Samba日本語版」だ（「日本Sambaユーザー会（http://www.samba.gr.jp/）」のサイトなどから入手できる）。ドキュメントの日本語化、日本語環境での使い勝手の向上などが図られている。各種ディストリビューション向けのバイナリパッケージも用意されているため、エンドユーザーにも容易にインストールできると思われる。また日本Sambaユーザー会のWebサイトには、Sambaの概要やインストール方法、トラブルシューティングなど、多くのノウハウが蓄積されている。これまで、各氏のボランティアによって運営されてきたSamba関連のWebサイトもここに集約されつつあるので、Samba関連の情報を集める上でチェックは欠かせないWebサイトだ。

Sambaの場合、オリジナル版で日本語コード（EUC、シフトJISなど）を含む各国語に対応しているため、使い方さえわかっていれば英語版であっても日本語環境に合わせて使うこともできるのだが、やはり日本語のドキュメントなどが参照できるのはありがたい。日本語環境で使用するには、本家Sambaのものよりもお勧めだ。

■各ディストリビューションのサポートサイト

Linuxディストリビューションでは、それぞれのパッケージ向けにポーティングしたSambaのアップグレードバイナリをWebやFTPなどで公開しており、自由に入手できるようになっていることが多い。ディストリビューションに合わせて細かな設定（各種ファイルの格納位置など）が調整されているため、エンドユーザーとしても非常に簡単に最新版のSambaにアップグレードできる。ディストリビューションによっては、前述のSamba日本語版をベースにしたSambaバイナリを配布していることもある。

筆者の場合は、英語版のRed Hat Linuxを使っているということや、Sambaを実験ベースで使っているといったことから、本家Sambaサイトで配布されているオリジナル（英語版）のSamba 2.0.6を使うことにした。幸い本家Sambaサイトでは、英語版Red Hat Linux 6.0向けのバイナリパッケージも用意されていたので、これをそのまま使うのが（筆者にとっては）一番ラクそうだ。

一般的には、各ディストリビューション対応のパッケージがリリースされるのを待って、それを使ったほうがいい。それは各ディストリビューション向けの細かな設定変更が行われていてエンドユーザーレベルでの作業が非常に単純だからだ。

前準備

自分が使っているディストリビューション対応のバイナリパッケージがある場合、Sambaのアップグレード作業そのものは非常に単純だ。特にRed Hat Linuxなど、RPMでパッケージが配布されている場合には、rpmコマンド一発で作業がすんでしまう。

ただし、その前準備や後始末には、それなりの手間をかけたほうがいい。特に実用環境では、バージョンアップにともなう仕様変更や、アップグレード作業のミスでSambaが動かなくなったりしたら大変だからだ。

■新しいSambaを入手する

前の項で紹介したように、最新版のSambaは、自分の使っているディストリビューションなどに応じて、適切なパッケージを入手しよう。注意してほしいのは、別のディストリビューション向けのパッケージを使わないことだ。ディストリビューションごとに、ファイルの格納先などの設定は微妙に違う（こともある）。ディストリビューションごとの違いを把握したうえで「このパッケージは流用できる」と判断できたならともかく、最新版だからというだけで、やみくもにインストールするのは、自分からトラブルを呼び寄せるだけの行為だ。

■バックアップ

まずは、現在使用しているSamba関連ファイルのバックアップだ。オリジナルのパッケージから再インストールできるドキュメントや実行ファイルなどはバックアップする必要はない。重要なのは、ユーザーが自分で作成・編集した設定ファイルである。前回・前々回で扱った、次のようなファイルをバックアップしよう。

```
/etc/smb.conf
/etc/smbpasswd
/etc/smbusers
```

バックアップといっても、それほど大げさに考える必要はない。これらのファイルを別ディレクトリにコピーしておけば十分だ。筆者の場合は/etc/backupというディレクトリを用意して、/etcの下のファイルを編集する時には、

とりあえず元のファイルをこのディレクトリにコピーしてから作業するようにしている。もう少しかっこよくやりたいのなら、CVSやRCSといったバージョン管理システムを使って、変更履歴を管理してもいい。

ちょっとしたファイルの保存には/tmpにコピーするという方法もあるが、システムによっては、/tmpディレクトリのファイルは、一定期間アクセスがないと削除されるように設定されているので、大事なファイルのバックアップには不向きだ。筆者の友人の中にも、「/tmpでプログラム開発をやっていたもんだから、作っていたコマンドのソースは消えちまった」などと、痛い目にあった人もいる。ご用心、ご用心。

■ドキュメントの確認

繰り返しになるが、Sambaは現在進行形で開発が進められているソフトウェアだ。そのためバージョンアップでは、バグフィックスとともに仕様の拡張や変更、そして以前用意されていた機能が削除されることもある。ドキュメントを読むのは面倒かもしれないが、事前にこれらに目を通しておかないと、あとで思わぬトラブルに巻き込まれることもある。Sambaの場合、/usr/docの下のSambaディレクトリなどに格納されている。

ソフトウェアの仕様変更などについては、Webサイトでも詳しく紹介されていることが多いので、まずはそちらをチェックするのがいいだろう。

■testparmコマンドで設定をチェック

Sambaは、バージョンアップのたびに細かな仕様が変更されている。特に注意が必要なのは、設定ファイル（smb.conf）のパラメータの追加や削除、そしてデフォルト値の変更だ。これらはドキュメントにまとめられているのだが、思わず読み落としてしまうことがよくあるので、作業ミスを前提としたチェック体制があるといい。

Sambaには、smb.confのデバッグ用ツールとしてtestparmコマンドが用意されている。元々はユーザーが作成したsmb.confのエラーチェック用ツールなのだが、その機能を利用すればアップグレード時のチェックにも応用できる。

コマンドラインから以下のようにtestparmを実行することで、ファイル（この場合はtestparm.dump.old）に現在の、つまりアップグレード前のSambaの設定が出力される。

```
# testparm -s > testparm.dump.old
```

出力はテキストファイルなので、エディタやlessコマンドなどで内容を確認してほしい。これが、現在のsmb.confを読み取ったときにSambaがどのように設定されるかのリストだ。なおアップグレード後、このファイルを利用するので、どこかに保管しておこう。

アップグレード作業

設定ファイルのバックアップなどの前準備が終わったら、いよいよアップグレード作業にとりかかろう。

■サービスを止める

アップグレード作業ではSambaの関連ファイルが大幅に差し替えられるので、作業を始める前に、必ずSambaサービスを止めておこう。コマンドラインで“samba stop”と実行すればよい。

■RPMによるアップグレード

筆者のように、RPMベースのバイナリパッケージが配布されているディストリビューションを使っていれば、ファイルのインストール作業そのものはrpmコマンド一発ですむ。読者の使用するパッケージが違う場合には、当然ファイル名も異なるので注意してほしい。

```
# rpm -Uvh samba-2.0.6-19991110.i386.rpm
```

連載を1回目から読んでくださっている読者なら、「あれっ？」と思うかもしれない。そう、rpmコマンドでは、パッケージのインストールもアップグレードも、ほとんど同じように行えるのだ。オプションの-i（install）が-U（upgrade）に変わっただけだ。これでSambaの関連ファイル（バイナリ、ドキュメント、設定ファイルなど）が最新版にアップグレードされる。

■RPMを使わないアップグレード

RPM形式のバイナリパッケージ以外からインストールする場合には、それぞれの配布形態に応じて作業する必要がある。また、ソースパッケージの場合は各種設定をシステムに合うよう書き換えてからコンパイル作業を行わなくてはならないが、今回は省略する。

■設定ファイル、smb.confの更新

Sambaのバイナリパッケージには、サンプルのsmb.

confが含まれていることが多く、Sambaをインストール・アップグレードすると、smb.confもそのサンプルファイルで置き換えられてしまう。すでにSambaを運用している場合、それぞれの環境に応じてsmb.confをカスタマイズしているだろうから、元のsmb.confに復元する必要がある。

rpmなどのパッケージ管理コマンドは、これらのファイルを更新する場合には、元のファイルを別名で保存するようになっている。たとえばrpmの場合、元のsmb.confはsmb.conf.rpmsaveというファイル名で保存される。もちろん、事前に行ったバックアップもあるはずなので、そのファイルを利用してもいい。

ただし、仕様が変更されている可能性もあるので、ドキュメントをしっかりと読んで、追加・削除されたパラメータに対応することや、デフォルト値の変更されたパラメータについては、明示的に値を設定するなどの作業が必要になる。

ドキュメントを参照しながらsmb.confの修正を終えたら、再びtestparmを使ってエラーチェックなどを行おう。まずはオプションを付けないでtestparmコマンドを実行して、単純な文法エラーがないかを確認する。ここで文法エラーがないようなら、先ほどと同じように設定内容をファイルに出力し、古い設定と比較してみよう。

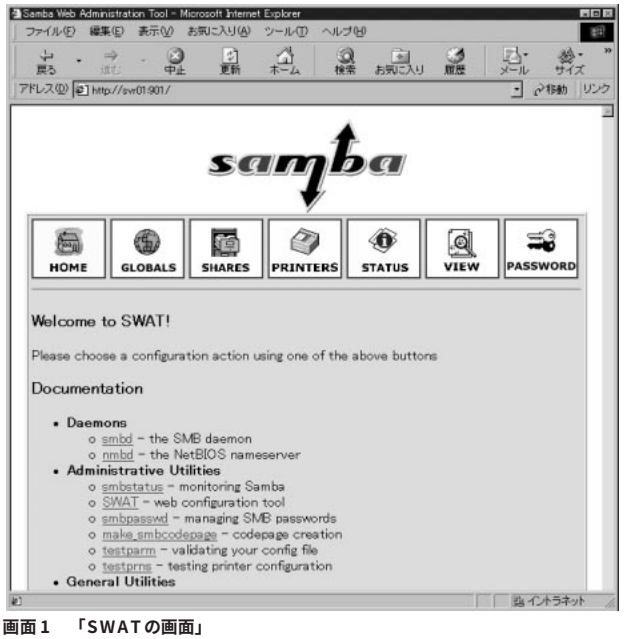
```
# testparm -s > testparm.dump.new
```

この新しいSambaの設定と古い設定とを比較するには、diffコマンドなどを用いる。

```
# diff testparm.dump.old testparm.dump.new
```

パラメータのデフォルト値などが変更されていれば、それをsmb.confに反映させる必要があるかどうかを検討し、必要に応じて対処してほしい。

以上で、Sambaのアップグレードは一段落だ。“ samba start ”を実行してSambaを起動し、クライアントから問題なくアクセスできるかどうか、チェックしてみよう。



画面1 「SWATの画面」

Webベースの設定ツール「SWAT」

ここまではSambaの設定変更、つまりsmb.confの編集には普通のテキストエディタを使用してきた。このやり方にはいろいろとメリットも多いのだが、Windowsに慣れた（慣らされた）身としては、GUIのほうがお気軽に使えるという読者も多いだろう。Sambaの開発チームにも同様のリクエストが届いたのだろうか、現在のSambaには、SWAT（Samba Web Administration Tool）と呼ばれる管理ツールが付属しており、ブラウザからWebベースで設定変更などの作業が行えるようになっている（**画面1**）。

■インストール作業

SWATは、FTPサーバやWebサーバなどと同じように、ネットワークサービスとして実装されている。その点ではSambaサービス（smbd）と同じなのだが、smbdのほうは事前に起動しておく必要があるのに対して、SWATのほうはリクエストに応じてネットワークデーモン（inetd）がSWATを起動するように設定する（smbdをこのように設定することも可能だが）。したがって、Sambaのように事前に起動しておく必要がない。その代わり、inetdの設定ファイルを書き換えて、SWATサービスを登録しておく必要がある。修正する必要があるのは、/etcの下にある、servicesファイルとinetd.confファイルの2つだ。

SWAT自体はSambaに同梱される形で配布されているので、Sambaがインストールされていれば、SWATの実行に必要なファイルもインストールされているはずだ。

■/etc/servicesの編集

/etc/servicesは、サービス名とTCP/IPのポート番号の対応を記述したファイルだ。SWATの場合は、以下の1行を追加すればよい。ディストリビューションによっては、

この指定がコメントの形で記述されていることもあるので、その場合にはコメントマーク（‘＃’）を削除するだけでいい。また以下の指定がすでに記述されている場合は、この作業は必要ない。

```
swat            901/tcp
```

■/etc/inetd.confの編集

SWATを使えるようにするためには、もう1つ、inetd.confの書き換えが必要だ。以下のような行を追加しよう。ディストリビューションによっては、swatプログラムの位置が違う（/usr/local/samba/binなど）こともあるので、読者のシステム構成に合わせてその部分は書き換えてほしい。

```
swat stream tcp nowait.400 root /usr/sbin/swat swat
```

■inetdの再起動

上記の2つの設定ファイルを変更したら、その変更を反映させるためにinetdを再起動する。このためにLinuxを再起動する必要はない。psコマンドを使ってinetdのPID（プロセスID）を調べ、killコマンドを使ってハングアップシグナルを送るだけでいい。

```
kill -HUP PID
```

PIDには、psコマンドで調べたinetdのプロセスIDを指定するわけだが、Red Hat Linuxなど、ディストリビューションによっては/var/runディレクトリに、主要なデーモンのPIDを記述したファイルが作成されるようになっているので、これを使ってもいい。

```
kill -HUP `cat /var/run/inetd.pid`
```

最後の部分を括っているのは、通常のクォートマークではなく、バッククォートだ。bashなど、UNIXやLinuxのシェルは、このバッククォートで括られた部分をコマンドラインとして実行し、その出力で置き換えてから元のコマンドラインを実行するようになっているのだ。

SWATを使ってみよう

さて、2つの設定ファイルを修正し、killコマンドでinetdを再起動したら、SWATが使えるようになっているはずだ。

画面2　パスワード入力のダイアログボックス（Windowsクライアントから Internet Explorerでアクセスした場合）

早速、Webブラウザを起動して接続したいところだが、その前にsmb.confのバックアップを行っておこう。SWATは、smb.confを大幅に書き換えてしまう。ユーザーが書き込んだコメントなどはすべて削除されてしまうし、パラメータの指定順序なども勝手に並べ換えられる。SWATによる管理が気に入れば、それはそれで構わないのだが、「やっぱりテキストエディタを使うほうがいい」と思ったときのために、バックアップを残しておきたい。

X環境でWebブラウザを起動し、“ http://localhost:901/ ”をブラウズすれば、**画面1**のようなSWATの画面（「SWATページ」と呼ぶべきか？）が表示される。Windowsなどのリモートクライアントから使うこともできる。URLの“ localhost ”の部分を、Sambaサーバのマシン名で置き換えるだけだ。たとえばマシン名が“ svr01 ”なら、URLには“ http://svr01:901/ ”を指定する。接続するとパスワードを尋ねるダイアログボックスが表示される（**画面2**）ので、そこでrootのユーザー名とパスワードを指定すれば、Sambaの設定を変更できるようになる。

SWATでできることの大部分は、コマンドラインの各種コマンドや、エディタによる設定ファイルの編集でも実現できる。とはいえ、GUIベースの作業には多くのメリットがある。特に嬉しいのが、ヘルプドキュメントを簡単に参照できることだ。筆者の場合は英語版のSamba 2.0.6をインストールしたので、表示なども英語になっているが、最初に紹介したSamba日本語版の場合は、ヘルプドキュメントまで日本語化されているので、安心して作業が進められる。また、このヘルプを読むだけでも、スキルアップは間違いないだろう。

SWATを使ったSambaの管理については、次回以降少しずつ紹介していくつもりだ。

ニューテクノロジー From SGI

オープンソースに提供されるSGIのテクノロジー

Knowledge

最終回 SGI Linuxで提供されるコアテクノロジー

文：鈴木大輔
Text : Daisuke Suzuki
日本SGI株式会社オペレーティングシステム担当
日本Linux協会理事
ds-suzuki@nsg.sgi.com
daisuke@linux.or.jp

SGIが放つオープンソースソフトウェアには非常に多くの種類がある。Linux自身にかかわる部分からカーネルの少し上のレイヤの部分、さらにはAPIやアプリケーションまで存在する。その中でも、前回まではSGIがこれから提供するコアテクノロジーの一部を紹介した。

しかし、SGIが世に送り出すものは、モジュール単位のものだけではなく非常に多くのものがある。性能向上のためのパッチから、アプリケーションまで実に広いレイヤのものを送り出している。

今回はその概要と中心となるSGI Linux Environmentについて解説しよう。

SGI Linux Environment

1999年9月にSGIが改良を施したLinuxを搭載したIA32サーバマシンが発表された。ハードウェアとしてはIA32アーキテクチャのハイエンドサーバであるが、その特徴はSGIによる機能拡張の入ったLinuxが搭載されていること、SGIによるトータルサポートが提供されていることである。

このシステムに搭載されているSGI Linux Environmentの拡張機能、さらに次期リリースにおける拡張機能は、SGIのオープンソースソフトウェアとしてWebサイト（http://oss.sgi.com/）ですでに公開されている。このSGI LinuxにはSGIの64ビットOSであるIRIXからの機能の一部がポーティングされており、Linuxをエンタープライズサーバとして利用するためのさまざまな機能が盛り込まれている。

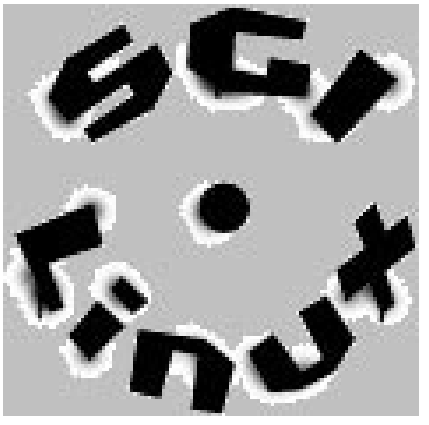
これにより、いままでLinuxが使われにくかった市場にも食い込んでいけるのではないかと思われる。

■SGI Linux 1.0

現在、SGI 1400Lサーバとともに出荷中のSGI Linux Environment 1.0（以降SGI Linux 1.0）では、次のような観点から改良が施されている。このリリースでは、基本的にISPやSOHOなどのインターネット向けサーバ向けのパフォーマンスチューニングと連続運用のための安定性向上を目的としている。具体的な改良点は以下の3点である。

- **Webサーバパフォーマンスの改善**
  —TCP/IPスタックの改良
- **NFSの安定性の強化**
  —knfsdの改良
- **セキュリティの強化**
  —ICMPに対するパッチ

1つ目の改良点がWebサーバのパフォーマンス向上であり、残りが安定性の向上である。Webサーバは標準の



Red Hat Linux 6.0と比較して、約2倍のパフォーマンスが記録されている。独自の調査によるとさらに高いパフォーマンスを発揮することもあった。

SGI Linux 1.0では細かなチューニングが中心であったが、SGIの拡張はほんの始まったばかりなのである。次に紹介するSGI Linux 1.1の機能拡張を見て頂けるとわかるだろう。

■SGI Linux 1.1

11月8日に発表されたばかりのSGI Linux Environment 1.1では非常に多くの改良が盛り込まれている。多くの人々が期待していた機能も取り込まれているのがわかる。その中には、すでにオープンソースで発表されていたものもあるが、SGIはそういったものに対してチューニングや改良を加えて取り込んでいる。

今回の目的の一番の目玉はI/Oの改良によるデータベースパフォーマンスの向上であろう。まだ、具体的な数値は出ていないが、かなりのパフォーマンス向上があると報告されている。SGI Linux 1.1では以下のような機能拡張が施されている。

・厳選したAlan Coxパッチ

Alan Cox氏の作成したパッチキットで、これには、ドライバのアップデートやハードウェアRAIDのサポート、NFSのバグ修正といったものが含まれている。SGIはこれらのパッチのひとつひとつを検証し、SGI Linuxに取り込んでいる。

・DEVFS

デバイスドライバの状態を仮想的にファイルとして参照できるようにした仮想ファイルシステムである。これによって、それぞれのデバイスのためのスペシャルファイルを用意する必要がなくなり、さらに、デバイスへのメジャー、マイナー番号の割り当てを自動化することができるようになる。

・NFSv3 w/NLMv4

このサポートはNFSサーバとクライアント双方に適用されており、最新のNFSv3プロトコルを用いてファイル共有ができるようになる。IRIXなどの商用UNIXの多くは、以前からNFSv3を使っているため、相互接続がより良いものになったといえるだろう。

また、同時にサポートされたNLMv4はネットワーク越しにファイルの排他処理をできるようにした機構である。これによってより信頼性の高いNFSを運用できるようになる。

・I/Oパフォーマンスの改良

I/Oパフォーマンスの向上としては、次の2つの機能を追加した。これらはデータベースを使う上で必要とされてきた機能であり、多くのデータベースベンダーが期待していたが、現在までLinuxでは実装されていなかったものである。

—Raw I/O

ディスクに対してシステムのバッファを通さずに直接アクセスするための機構である。この機能は通常のUNIXでは必ず持っているものであるが、Linuxではデバイスへの直接アクセスもバッファを通してアクセスしていた。これによってデータベースのパフォーマンスや、信頼性が向上する。

—POSIX非同期I/O（KAIO）

POSIX標準で定められた非同期I/OのシステムをLinuxに実装したものである。この実装はカーネルレベルでサポートしているため、KAIOと呼んでいる。KAIOによってデバイスごとにI/O処理を並行して行うことができるようになる。

データベースのパフォーマンステストの結果によると、約35%の性能向上が見られたとのことである。I/Oに限定したパフォーマンス比較では、Raw I/OとKAIOによってデバイスが持つ論理値に非常に近いパフォーマンスを発揮したとのことである。

・3.8Gバイトメモリサポート

現在、IA32上のLinuxでは、仮想メモリ空間が4Gバイトに制限されてしまっている。さらに、物理メモリについては2Gバイトのメモリしかサポートしていない。また、この2Gバイトについてもユーザーに与えられる空間は1Gバイトとなってしまう。

これに対してSGIの改良では、3.8Gバイトの物理メモリをサポートし、ユーザー空間に3Gバイトが割り当てられるようになっている。大量のメモリを使うプログラムも、この改良によって可能になるだろう。

・内蔵カーネルデバッガ

従来のLinuxのカーネル開発ではprintf（printk）を使用してデバッグしていた。Linuxが非常に単純な構造であったころはこれで十分であったが、カーネルが複雑になるにつれ、この方式ではデバッグしきれない状態になっている。

これに対してSGIはカーネル内部にデバッグのための仕組みを実装し、外部から標準のデバッガを使ってデバッグできるようにした。また、リモートマシンからデバッグできるようにそのサポートも実装した。これはすでに以前から公開して、カーネル開発者から

良い反響を得ていた機能を、SGI Linuxとして標準に取り込んだものである。

・クラッシュダンプサポート

上記のカーネルデバッガと同様にカーネルをデバッグするための機構である。商用UNIXでは一般的な機能で、カーネルクラッシュした時のメモリイメージをディスクにダンプし解析するための仕組みである。これによって予期しないカーネルクラッシュの原因を解明することが可能となり、より迅速な問題解決が期待できる。

・Spinlock Metering

これもカーネルのデバッグ用の機能である。LinuxのSMPカーネルではデータアクセスの競合が発生しそうな状況ではデータの保護にspinlock（アクティブなCPU以外を止める）を使っている。

spinlockmeterは、そのspinlockの発生状況をリポートするための機構で、lockstatという新しいコマンドを使うことによって、これを見ることができる。SMPシステムのチューニングやカーネルデバッグには有効であると思われる。なお、この機能はシステム監視によってパフォーマンスに影響を及ぼすことがあるため、標準では有効にしていない。

・Post/Wait Synchronization

この機構はプロセス間同期を取るための非常に高速で軽量な機構（sleep、wakeup、timer）である。これによってプロセス間の同期を少ないコストで実現できるようになる。そして、これを利用するためのライブラリも同時に提供している。

・高速なgettimeofdayシステムコール

Linuxでは標準のgettimeofdayはシステムコールとして実装されている。すなわちカーネルで処理されるため、この関数が実行されるたびにカーネルとユーザープロセスのコンテキストスイッチが発生してしまうのである。これは非常にコストが大きく、大量にこのシステムコールが発生するとパフォーマンスに多大な影響を及ぼしてしまう。データベースではこの関数が実行される回数が非常に多くなってしまうため、パフォーマンスに影響が出ている。

これに対して、SGIの提供する改良した関数ではユーザープロセスですべて処理するようになっている。そのおかげでコンテキストスイッチの起こる回数を劇的に減少させることができ、パフォーマンス向上に役立っている。

・新規ドライバ

以下のドライバが新しくサポートされている。これらはSGI 1400のオプションとして用意されているカードで、それらをサポートするために開発したものである。

—QLogic 1080/1280 SCSI
—QLogic 2100ファイバチャネル
—Alteon、SGIギガビットイーサ

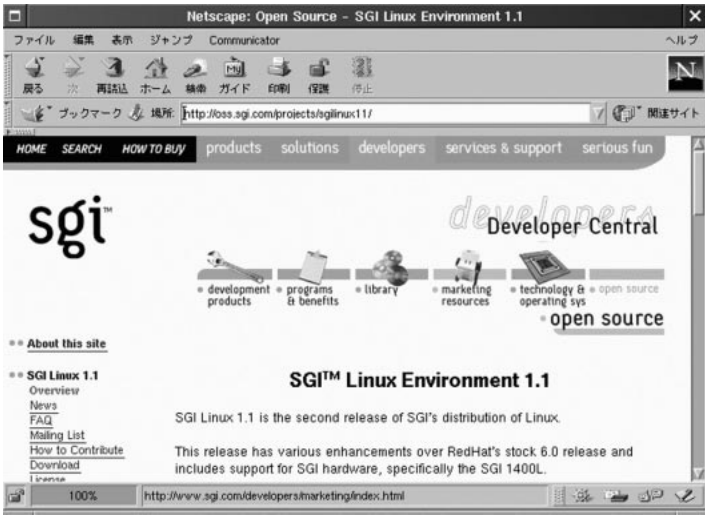
以上のようにSGI Linux 1.1では、多くの機能が追加されている。また、ここにあげた機能以外にも多くのバグフィックスがなされている。

SGIのサーバにインストールされたSGI Linuxについては、これらすべての機能がSGIによって完全なサポートが行われる。もちろんベースとなるRed Hat Linuxの部分も同様にサポートを行う。言いかえれば、エンタープライズコンピューティングに必要で、かつ今のLinuxに不足している機能はすべてSGIがサポートしていくと言っても過言ではないだろう。

もちろん、すでにWebで公開されているように、すべての改良はオープンソースコミュニティへと開放していく予定である。今後登場するであろうものについてもどんどん開放し、SGI Linuxに取り込まれ、Linuxの発展に寄与できるのではないだろうか。

SGI Linuxの今後

SGI Linux 1.1でも多くの改良が発表されたが、SGIの拡張はこれだけでは終わらない。今回のリリースでは、



画面1　SGI(TM) Linux Environment 1.1
http://oss.sgi.com/projects/sgilinux11/

データベースパフォーマンスの向上とカーネル開発環境の向上が目的であり、前回のWebサーバのパフォーマンス向上とあわせてかなりエンタープライズ向けの機能拡張が進んできた。

しかし、まだまだネットワークパフォーマンスやファイルサーバのパフォーマンスには問題もある。第1回で紹介したXFSもそのひとつであろう。また、プロセス管理機構、スケジューラ、SMPの対応についてもさらなる改良が必要であろう。さらに、IRIXの得意とするメディアストリーミングについても、リアルタイム処理などを含めて改良の余地はある。SGIはこういったものに対しても、これからも改良を続けてゆきLinuxをエンタープライズサーバからハイパフォーマンスサーバまで使えるものに育てていく予定である。

さらにサーバ用途だけではなく、今後はワークステーションに関してもLinuxのサポートを行っていく。前回のOpenGLやPerformerがそのひとつであるが、SGIのプロダクトとしても今後出してゆく予定である。これによってSGIはワークステーションからハイパフォーマンスサーバまでLinuxを全面的にバックアップしていくことになる。今後のSGI Linuxはますます目が離せないものになるだろう。

IA64 Linux

ここまではIA32システムの話をしてきた。しかし、IA32のLinuxを開発・改良すると同時に、SGIはIA64へのLinuxのポーティングにも協力している。このポーティングプロジェクトは、Intel、SGI、IBM、HP、VA Linux、Cygnusが参加し、Trillian Projectと呼ばれている。このプロジェクトでは最初のIA64プロセッサであるItanium（aka. Merced）への移植、改良などを行っている。この成果の一部は、Supercomputing '99においてItaniumの16CPUのクラスタマシンが公開されたことで、より現実味を帯びてきているだろう。

これまで述べてきたようにSGIはIRIXで培ってきた64ビットUNIXの技術、スケーラブルシステムの技術をLinuxへ投入していこうとしている。IA64 Linuxがエンタープライズ、スケーラブルな世界で登場するのも近いのかもしれない。

最後に過去3回で紹介してきたオープンソースソフトウェアを含め、SGIが現在提供しているものをまとめておこう。

- **SGI Linux Environment**
  今回紹介したSGIの機能拡張されたLinuxディストリビューション。

- **XFS**
  ハイパフォーマンス・ジャーナルファイルシステム。

- **Linux/MIPS**
  Indy、Indigo2で動作するLinux。

- **Linux on the SGI Visual Workstations**
  VWS320/540で動作するLinuxでSGI Linux 1.1にはパッチが含まれている。

- **Samba for IRIX**
  昨年からサポートを開始したIRIX版のSAMBA。サポートも行っている。

- **Jessie**
  Javaベースのクロスプラットホーム統合開発環境。

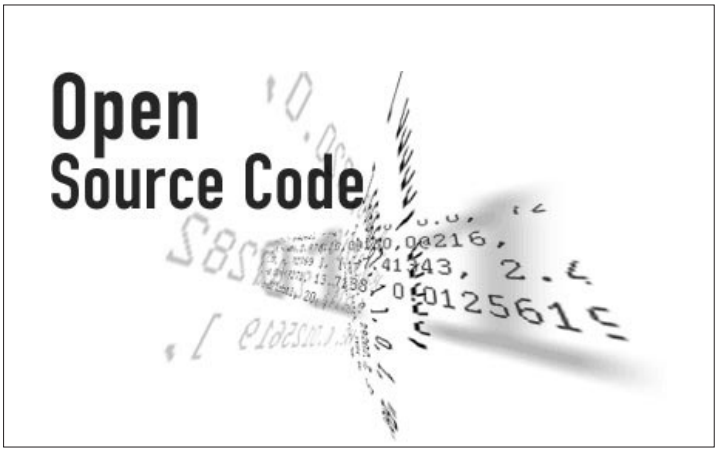
- **CRCalc**
  Constructive Reals Calculator、Javaアプレット。

- **OpenVault**
  ストレージマネージメント/フレームワーク。大量のリムーバブルメディアの管理が統合的に行える。

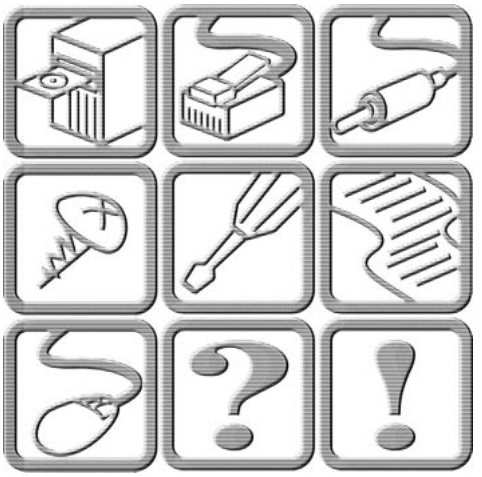
- **STL**
  C++標準テンプレートライブラリ。

- **GLX**
  前回紹介したXのOpenGL拡張。

これら以外にも開発中のもの、オープンソースにする予定のものもある。今後もSGIのLinuxに対する取り組みに期待していただきたい。機会があればまた紹介してみたいと思う。



Try & Try [trái] [trái]



システムライブラリ

文：政久忠由
Text : Tadayoshi Masahisa

僕らの住む世界は、もろくて、はかない。
それゆえ、美しく、尊い。
じゃあ、もろくて、はかないシステムというのはどう。
いらないな。

Linuxと C言語

UNIXとC言語はとても密接な関係にある（詳しくはUNIXやC言語の書籍の冒頭にそのクロニクルとともに必ずといっていいほど書いてあるので見てくだされ。また後ほど出てくるCライブラリでもISOの標準CとかPOSIXといったことにもここでは一切ふれないので、悪しからず）。

Linuxの場合も多聞にもれず、Linuxカーネル自身はC言語で記述されていて、Linux上で動作するプログラムのほとんどもC言語で書かれている。C言語よりもいろいろな面で優れているといわれる、さまざまな言語も利用できるのだが、いまだにC言語が多いのだ。理由はいろいろあると思うけど、今のところプログラミング環境を含め、その経験と実績が根強く残っているのだろう、僕の勝手な意見だけど。

C言語に限ったことではないが、一般にユーザーに評価されているプログラミング言語は、その言語仕様はシンプルかつコンパクトなものとなっている。複雑なものは、シンプルなものを組み合わせればよいという発想だ。これは理にかなっている。ただ、そのコアの部分だけを提供したのでは、ユーザーの手間がばかにならない。そこでライブラリと呼ばれる、あらかじめ共通的によく使われる機能ルーチン（関数）も同時に提供されているのだ。

C言語の入門時に最初にコールするであろうprintf()もこのライブラリルーチンなのである。またシステムのリソースを操作するには、通常、OSに対してシステムコールを発行するんだけど、これをユーザープログラムから直接発行することはまずなくて、そのほとんどはこのCライブラリを経由して利用する（できる）ようになっている。だからこそ、OSのシステムコールの種類やそのパラメータが異なっていたとしても、ユーザーはそのことを気にしないですむのだ。

リンクするということ

プログラムとライブラリを結びつけることを一般にリンクするというが、このリンク方法には大きくわけて2つの方法がある。それは、スタティックに結びつける方法とダイナミックに結びつける方法だ。

スタティックに結びつける方法では、必要となるルーチンすべてをプログラム自体に含める。一方、ダイナミックに結びつける方法では、プログラムには必要となるルーチンの名前を記述しておいて、実行時にルーチンの実体と結びつける。この場合、そのライブラリルーチンは複数のプログラムからのリンク要求を処理できるようにメモリアド

レスに依存しないコードで構成され、通常Shared（共有）ライブラリと呼ばれている。

これらは、それぞれメリットとデメリットがあるのだけれど、最近の傾向としては、スタティックリンクすることは少なくなり、共有ライブラリをダイナミックリンクすることがほとんどだ（いつからかは忘れたけどgccのデフォルトオプションもそうなっている）。

昔は、システム障害時の復旧のことも考えて、/binと/sbinのプログラムは最低限スタティックリンクされていたものではあるが、最近はそれらもダイナミックリンクだったりする。古臭い人々は驚くかもしれないが、最近はそんなものなのだ。

システムライブラリの位置付け

冒頭で述べたようにLinuxとC言語に深い結びつきがあることは、いわゆるCライブラリがシステムの一部に位置付けられていることからわかる。そのため、そのシステムに不可欠なCライブラリをシステムライブラリと呼んでいる。基本的には、特に手を加えないで、そのシステム上でコンパイルして作成した実行ファイルはそのCライブラリに依存したものとなる。Linuxのシステムライブラリは、libc4、libc5、libc6などと呼ばれており、libc4は、GNU Cライブラリのバージョン1に、独自に手を入れたものであったが、現在主流のlibc6では、GNU Cライブラリのバージョン2がそのまま使用されている。各Cライブラリの詳細はここでは省くが、基本的にバージョンアップするたびに機能拡張が行われていると考えればよい。

そういえば、LinuxカーネルもC言語で記述されているわけだが、このCライブラリを使っているのだろうか？答えは、使っていない。一般にCライブラリはユーザーモードで動作するプログラム用に作成されたもので、カーネルモードで実行されることはまったく考慮されていない。だからカーネルのような場合はすべて自前で用意しているのだ。カーネルモードで動作するプログラムでは絶対に使えないというわけでもないのだが、通常カーネルモードではもっと効率に実行できるコードを利用している。

スタティックとシェアード

さて、今回はシステムライブラリを再構築してみようと思うわけだが、その前に少しだけスタティックリンクとシェアードリンクについて見てみることにする。

Linuxの場合、/libと/usr/libにあるlibc.*ファイルがシステムライブラリである。その中心となるのが、/usr/lib/libc.a（スタティックリンク用ライブラリ）と/lib/libc.so.6（共有ライブラリ）だ。通常、拡張子がsoは共有ライブラリで、aはスタティックライブラリと考えればよい。/libにはMathライブラリ（libm.so.*）や暗号用（libcrypt.so.*）、Linuxスレッド用（libpthread.so.*）などのシステム共有ライブラリと呼ばれるものが、/usr/libにはそれ以外のライブラリが置かれる。

ちなみに、各実行ファイルがどのようになっているかは、ファイルの種類を判定するfileコマンドと共有ライブラリへの依存状況を表示するlddコマンドで確認することができる。例として、/bin/bash、/sbin/init、/usr/bin/viを見てみることにしよう。

```
$ file /bin/bash /sbin/init /usr/bin/vi
/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1,
                dynamically linked (uses shared libs), stripped
/sbin/init: ELF 32-bit LSB executable, Intel 80386, version 1,
                dynamically linked (uses shared libs), stripped
/usr/bin/vi: ELF 32-bit LSB executable, Intel 80386, version 1,
                dynamically linked (uses shared libs), stripped
```

fileコマンドの結果については、特に説明の必要はないと思うが、それぞれ共有ライブラリとダイナミックリンクされていることがわかる。最後のstrippedについては後で少し説明する。

```
$ ldd /bin/bash /sbin/init /usr/bin/vi
/bin/bash:
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x4001c000)
        libc.so.6 => /lib/libc.so.6 (0x40020000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/sbin/init:
        libc.so.6 => /lib/libc.so.6 (0x4001c000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
/usr/bin/vi:
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x4001c000)
        libcanna.so.1 => /usr/lib/libcanna.so.1 (0x40020000)
        libc.so.6 => /lib/libc.so.6 (0x40071000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

lddコマンドの結果は、実際に各実行ファイルがどの共

有ライブラリに依存しているのかを表示している。それぞれlibで始まるファイルは共有ライブラリで、HEX表示は、それぞれのプロセス空間内のライブラリがマップされるアドレスだ。

この結果からは、共有ライブラリだけでなく、ld-linux.so.2というファイルがリンクされていることに気づくと思う。このld-linux.so.2は共有ライブラリではなく、ダイナミックリンカローダと呼ばれる各プログラムが実際に実行できるように前準備を行うプログラムだ。

ダイナミックリンクされた実行ファイルは、実行ファイルが生成された段階では、このライブラリのこのルーチンを使用する予定という情報しか含まれていない不完全な状態なのだ。そこで、実行時に実際のリンクを確立しなければならないわけだが、ld-linux.so.2は、必要な共有ライブラリをロードして最終的なリンク状況を確立する役割を担っている。この作業が終わってはじめてプログラムは実行できるのだ。

この処理は完全に自動的に処理されるのでユーザーはまったく気にしなくてもよいのだが、1点だけ注意することがある。それは、共有ライブラリがどのように検索されるかである。共有ライブラリは、環境変数LD_LIBRARY_PATH、そして/etc/ld.so.cache、/usr/lib、/libの順に検索される。そのため、同じファイル名で検索順の高い位置に問題のある共有ライブラリが存在したりするとプログラムが実行できないという問題が発生したりする。なお/etc/ld.so.cacheは、/etc/ld.so.confで指定されたパスと/usr/lib、/libから見つかったライブラリファイルの情報を集めたもので、ldconfigコマンドでメンテナンスできる。共有ライブラリを/usr/libに追加したとしても自動的に/etc/ld.so.cacheが更新される仕組みはまだないようなので、今のところはrootユーザーが手動で実行する必要がある。

コンパイルしてみて確かめる

スタティックリンクとシェアードリンクの違いを実際にプログラムをコンパイルしてみて、確かめることにしよう。

プログラムは何でもよいのだが、とりあえず、C言語の入門書の最初に書いてあるような次のコードで試す（ファイル名はtest.c）。それぞれファイルのサイズとfileコマンドとlddコマンドの結果を見てみよう。

```
#include <stdio.h>
int main(void)
{
        printf("Good?\n");
        return 0;
}
```

まずは、何もオプションを指定しないでコンパイルしてみた。

```
$ gcc test.c
$ ./a.out
Good?
$ ls -l a.out
-rwxrwxr-x    1 tadayo-m tadayo-m     4811 Nov 29 00:53 a.out
$ file a.out
a.out: ELF 32-bit LSB executable, Intel 80386, version 1,
            dynamically linked (uses shared libs), not stripped
$ ldd a.out
       libc.so.6 => /lib/libc.so.6 (0x4001c000)
       /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

次にスタティックリンクを指示する-staticオプションを指定してみた。

```
$ gcc -static test.c
$ ls -l a.out
-rwxrwxr-x    1 tadayo-m tadayo-m   248604 Nov 29 00:55 a.out
$ file ./a.out
./a.out: ELF 32-bit LSB executable, Intel 80386, version 1,
                                  statically linked, not stripped
$ ldd a.out
       not a dynamic executable
```

fileコマンドとlddコマンドの結果は予想どおりだと思うが、ここではファイルサイズの違いに注目しておこう。たかだかprintf()をコールするだけのプログラムなのに、この違いである。これはスタティックリンクを使用しないで、共有ライブラリをダイナミックリンクすることの1つのメリットだ。

もっとも、スタティックリンクとダイナミックリンクのメリットとデメリットを語るには、もっとさまざまな角度から検証する必要があるので、今回はやめておくが、一般には必要に応じて使い分けるようになっている。

そういえば、fileコマンドの結果表示でstrippedは何かということだが、これはシンボル情報の有無を表している。このシンボル情報は、デバッグシンボルとは別ものなのだが、内部のセグメントシンボルを表すもので、objdumpコマンドやnmコマンドで利用することができる。この情報が必要ない場合は、コンパイルするときに-sオプションを指定する。なお、このオプションはスタティックリンクの時にも利用できる。

```
$ gcc -s test.c
$ file a.out
a.out: ELF 32-bit LSB executable, Intel 80386, version 1,
                dynamically linked (uses shared libs), stripped
$ ls -l a.out
-rwxrwxr-x    1 tadayo-m tadayo-m     3012 Nov 29 00:54 a.out
```

通常、このシンボル情報を利用するのは開発者だけなので、ディストリビューションのリリース版などでは、実行ファイルからこれらの情報を取り除いたものが提供されていると思う。これを取り除くと、さらに少しだけファイルサイズを小さくすることができるのだ。最初のオプションなしの場合のファイルサイズと比べると、4811から3012へと小さくなっているのがわかる。

一応、シンボル情報の有無によってどのような違いが生じるかも紹介しておこう。

●nmコマンドの結果

```
$ nm a.out  (シンボル情報なし)
a.out: no symbols
$ nm a.out  (シンボル情報あり)
080494ec ? _DYNAMIC
080494cc ? _GLOBAL_OFFSET_TABLE_
080484a0 R _IO_stdin_used
0804958c A __bss_start
080494ac D __data_start
         w __deregister_frame_info@@GLIBC_2.0
08048410 T main
0804958c b object.11
080494b0 d p.3
         U printf@@GLIBC_2.0
```

●objdumpコマンドの結果を比較した抜粋

```
<  80482ab:     e8 f0 ff ff ff          call   0x80482a0
>  80482ab:     e8 f0 ff ff ff          call   80482a0 <_init-0x4>

<  80482d0:     e8 fb 00 00 00          call   0x80483d0
<  80482d5:     e8 56 01 00 00          call   0x8048430
>  80482d0:     e8 fb 00 00 00          call   80483d0
                                                <frame_dummy>
>  80482d5:     e8 56 01 00 00          call   8048430
                                              <__do_global_ctors_aux>
```

スタティックリンクされたプログラムは残ってる？

システムにスタティックリンクされたプログラムがあるかどうかちょっと調べてみた。結果はいくつか残っているようである。でも特にコメントすることもないんだなぁ…。興味があったら試してちょ。

```
$ file /bin/* /sbin/* /usr/bin/* /usr/sbin/* | grep statically |
cut -f1 " "
/bin/ash.static:
/bin/rpm:
/sbin/ldconfig:
/sbin/sash:
/sbin/sln:
/usr/bin/rpm2cpio:
/usr/bin/statserial:
```

システムライブラリの再構築

さて、いよいよ本題に入ろう。システムライブラリの再構築である。

システムライブラリを再構築したり、新しいバージョンに差し替えたりする人は、採用予定のシステムライブラリの検証やプログラムの動作確認など、通常、開発に携わっている人だろう。

それにも関わらず、ここではとりあえずやってみようというのだからとっても無謀な冒険なのである。

だから、最初に忠告しておく。通常わざわざシステムライブラリをコンパイルし直したりする必要はどこにもない。こんなことをやると、今まで動いていたプログラムが動かなくなったり、システムが不安定になったりして、再インストールしか道はないという状況にもなりかねない。だから、もしやろうと思うのであれば、システムが壊滅してもよい状況など、それなりの覚悟で望んでほしい。また日本

語版と名前のついたディストリビューションを使用している場合、システムライブラリもそれぞれ日本語対応のためにいくつか手が加えられていると思う。しかし、僕の環境は日本語版ではないし、ここで紹介するのも、ftp.gnu.orgから持ってきたglibc-2.1.2を対象としているので、これをそのまま適用すると、それぞれの日本語対応の部分が水の泡となることは間違いない。

そんなわけで、これらをふまえてやってみようという人だけ、試してみてほしい。

また、現システムがlibc5以前の場合は、いくつかの面倒な作業も必要となるのだが、今回は面倒をみない。ここでは、すでにlibc6を採用しているシステムであることが条件だ。でも付属のINSTALLやFAQなどのドキュメントには、いろいろ書いてあるのでそれらを参照にすれば、それほど難しくないのでチャレンジしてみるのもよいだろう。

GNU Cライブラリの最新版を手に入れる

まず、GNU Cライブラリを入手する。これはftp.gnu.orgほかGNUのソフトウェアをミラーしているサイトからダウンロード可能だ。

```
glibc-2.1.2.tar.gz
glibc-crypt-2.1.2.tar.gz
glibc-linuxthreads-2.1.2.tar.gz
```

ファイルは3つだが、cryptは米国の輸出規制に引っかかるために、そこには置かれていないと思う。これの入手先は、glibc-2.1.2.tar.gzを展開したドキュメントの中に書いてあるので、読んで拾ってこよう。また、各ディストリビューションのソースCDに収録されているものでもかまわないが、この場合はライブラリを再構築するメリットがあるかどうかをよく考えてからにしてほしい。僕の場合は、既存のライブラリが2.1.1だったので、バージョンアップしてみたかったということと、Pentium Pro用のオプティマイズをしてみようと思ったからだ。

コンパイル自体は難しくないけど…

これらのファイルを展開してコンパイルするには、ディスク容量は200Mバイト程度必要となる。コンパイルに必要となるツールはINSTALLに書いてあるのでチェックしておいてほしいが、最近のディストリビューションであれば全部揃っていると思う。

ファイルはそれぞれ適当なディレクトリに展開する。cryptとlinuxthreadsは、glibc-2.1.2ディレクトリに展開する。ほかで展開してしまったなら、"mv crypt glibc-2.1.2"として移動すればよい。

まずは、glibc-2.1.2ディレクトリに移動し、compileなどの作業ディレクトリを作成する（必ずしも作業ディレクトリを作る必要はないが、一応）。次にcompileディレクトリに移動し、configureを実行する。

今回の方針は、自分のシステムに最適化したシステムライブラリを作って、それを実運用環境として現システムライブラリと入れ替えてしまうことで、テスト用のシステムライブラリ環境としてではない。だからオプティマイズを最優先としている。指定するオプションは、--prefix=/usr、--enable-omitfp、--enable-add-ons=crypt,linuxthreadsだ。それぞれの意味はドキュメントやconfigure --helpとして確かめてほしい。それが終われば、続いてmakeを実行だけだ。コンパイルにかかる時間は環境によって異なるが、ぼくのPentium 200MHz程度のマシンで4時間ぐらいかかる。きちんと計ったわけじゃないけど、ものすごくかかる。だから僕はいつも寝る前に始めて、結果は翌朝確認している（でもPentium IIIだとあっという間かも。持ってないからわからないけど）。

```
$ cd compile
$ ../configure --prefix=/usr --enable-omitfp --enable-add-ons=crypt,linuxthreads
$ make;make check
```

make checkがきちんと終了すれば基本的には大丈夫だ。また少なくとも安定版としてリリースされているバージョンなので、コンパイルの段階で問題が発生する可能性は低いと思われる。ただし、筆者の環境では、make checkが完全には終了していない。実はmathライブラリのテストのいくつかでエラーが表示されたのだ。特に使うことのないルーチンだったし、pgccを使っていて最適化のしすぎかなと適当な理由も見つかったので、無視してしまったが、本来、無視すべきではない。make checkがきちんと終了しないようなら、インストールはあきらめるべきだ。この場合、configureの--enable-omitfpオプションをはずしてやり直してみてほしい。これで問題ないようであれば、最適化に問題があると思う。あと、僕は以前、環境変数にLDFLAGS=-sを指定しているの忘れたまま

configureを実行してライブラリを作成してしまったことがある。一見コンパイルは成功したように見えるのだが、シンボル情報がないためにld-linux.so.2でリンクが解決できない状態になってしまった。思わぬところに落とし穴があったりするのだ。

make checkがエラーなしに終了したら、いよいよインストールだ。

```
# make install
```

この後、システムをリブートしてみると本当にうまくいったかどうかがわかる。成功を祈る。

システムライブラリを最適化したとしても、カーネルなどと異なり、目に見える成果があるわけではないので、ちょっと寂しいのだが、とりあえず心の達成感を味わうことにしよう。

国際化、ロケールの設定

再起動後きちんと動作しているようなら、次のステップに進もう。

僕の場合、日本語はktermとvi、emacsで利用できれば満足なので、これといって国際化に興味はなかったのだが、一応ロケールデータをインストールしてロケールの設定もしてみよう。

まずロケールデータをインストールする。これは先ほどのcompileディレクトリで行う。実行すると、/usr/share/i18n/ディレクトリのcharmaps、locales、repertoiremapsにロケールを作成するためのデータがインストールされる。

```
# make localedata/install-locales
```

次に実際のロケール情報を作成する。これには、localedefコマンドを使用する。localedefは、定義ファイルとキャラクタマップファイルを指定して必要なデータを生成するようになっている。たとえば、アメリカの英語ロケール（en_US）だと次のように作成する。同様にデフォルトのロケールとなるPOSIXも作成する（下記のコマンドのen_USの部分をPOSIXにすればたぶん大丈夫）。

```
# localedef -i en_US -f ISO-8859-1 -u mnemonic.ds en_US
```

作成したロケールデータは、/usr/share/locale/en_US（POSIX）ディレクトリにLC_COLLATE、LC_CTYPE、LC_MONETARY、LC_NUMERIC、LC_TIMEファイルとして保存される。

さっそく、ja_JP、日本の日本語ロケールを作成しようと思ったのだが、定義ファイルja_JPはあるもののそれを解決できるcharmapファイルがないのだ。作るかインターネットで探せばいいんだけど、当面利用するあてもなく面倒だったので、気分だけを味わうことにした。ja_JPを適当な名前にコピーしてそのファイルの簡単にわかる部分を漢字（EUC）に当てはめた。これだけだと、localedefコマンドはエラーだよというものだから、--forceオプションでどうにか形だけ取りつくろう暴挙にでた（なっちゃないなぁ）。というわけで、気持ちだけ味わってみよう。

```
$ LANG=ja_JP ls -l
drwxr-xr-x    5 tadayo-m tadayo-m     2048  6月 14 08:33 Desktop
drwxrwxr-x    5 tadayo-m tadayo-m     2048  8月  9 17:31 Mail
-rw-------    1 tadayo-m tadayo-m      527 11月 14 04:32 mbox
-rw-rw-r--    1 tadayo-m tadayo-m   117994 11月 24 09:42 test.log
```

タイムスタンプの表示が6月などとなっているわけだが、lsの場合はあまり見栄えがよくない。

続いて日本語のメッセージカタログが用意されているコマンドの表示も見てみよう。これはいいねぇ。

```
$ LANG=ja_JP mount -h
使い方: mount [-hV]
       mount -a [-nfFrsvw] [-t ファイルシステム型]
       mount [-nfrsvw] [-o オプション] special | node
       mount [-nfrsvw] [-t ファイルシステム型] [-o オプション] special
node
          スペシャルデバイスは  -L ラベル  又は  -U uuid で指示できます。
```

日本語版と名前のつくディストリビューションのいくつかは、このロケールの設定や日本語のメッセージをキチンとサポートしている（ようだ）。もし英語版とかを使っていて、ちゃんとしたロケール情報がほしければ、それらを参考にするか、借りてくるのが手っ取り早いだろう。

そんなことをする必要があるんだったら、最初からそれらをサポートしたディストリビューションを使えばいいって？　まぁ、そうなんだけど、日本語版が何となくいやな人もいるでしょ、僕みたいに。

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

Linux 2.4のすばらしき世界

Text : Joe Pranevich

■Linuxの内部仕様

Linuxカーネルとは本質的にどんなものなのだろうか？ カーネルがLinux OSの中心部分であるのと同様に、カーネルそのものも中心とそれ以外の部分とに分けられる。いろいろな点で、このLinux 2.4の「中心部分」はLinux 2.2のものとは異なっている。

Linux 2.2およびそれ以前のLinuxには、基本リソース管理システムが含まれていて、それはI/OポートやIRQラインや、その他の制限付きのコンピュータアーキテクチャの細かい点を、かなり雑に割り当てたり追跡したりしていた。Linux 2.4の上の新しいシステムには、もっと多くの汎用の実装が含まれていて、それはネストしたリソースグループが使えたり、定義済みのリソース型の依存を取り除いたり、もしくはドライバが必要とする大部分の作業を簡単にしたりする。

仮想ファイルシステム層（VFS）もまた、以前のLinuxから大きく変更された。Linux 2.2の処理に大きな制限を与えていたもののひとつは、キャッシュ処理に2つのバッファを使うことだった。1つは読み込み用で、もう1つは出力用だ。Linux 2.4でこの壁は完全に取り除かれる。

Linux 2.2では、一度に実行できるプロセスまたはスレッドは、1024個までとなっている。Linux 2.4はこの古い問題を片づけ、実行時に設定できて、システムのメモリの量だけによって制限されるようなスケーラブルな制限を実装した。

メモリの消費量に関して言うと、Linux 2.4はLinux 2.2が必要とするのとだいたい同じ量を必要とする。Linux 2.4からは、Intelマシンでは最大4GバイトまでのRAMがサポートされている。

■バイナリ形式

Linuxカーネルの中で重要な部分は、プログラムローダであるということをわかっている人は多いだろう。しかし、Linux 2.2において「misc.」バイナリローダが追加されたことに気がついていない人も多い。このバイナリローダは、Windowsや同等のOSとほとんど同じ方法で、ヘルパーアプリケーションをあるバイナリ形式と結びつけることができる柔軟なモジュールだ。

Linux 2.2はカーネルがELFとしてコンパイルされる必要のある最初のLinuxだったが、Linux 2.4はLinux 2.2よりもさらにELF形式に依存している。ELFバイナリ形式をより完全に活用することで、カーネル開発者たちはコードのいくつかの部分をよりモジュール化し、保守しやすくすることができる。

■CPUとバス

Linux 2.4にはLinux 2.2でなされていたのと同様、各種プロセッサへの優れた対応が含まれている。Intelのx86系列の64ビット版の後継チップのリリースはもうすぐだ。Linux 2.4にはこのチップへの直接の対応はないが、そのリリース直後からMercedの上でLinuxが確実に動くようにするための作業をするグループが複数存在することになるだろう。

しかし、プロセッサはコンピュータの中身のほんの小さな一部分でしかない。その演算処理と同様に重要なのは、そのバスのアーキテクチャである。この分野の最大のニュースはISAプラグ&プレイで、ISAバス上でデバイスの設定や検出をしようというものだ。

この分野において、もっと刺激的なニュースがある。USBがLinuxカーネルに組み込まれたのだ。またUSBに加えて、I2Oデバイス、すなわちPCIの拡張のサポートがLinux 2.4に追加されている。理論的には、これによってよりOSに依存しないデバイスやドライバが存在できることになる。

PCMCIAバスは、標準カーネル配布版の中でサポートされている。PCMCIAユーザーはもはや自分のシステムをきちんと動かすのに、別パッケージをダウンロードしてインストールする必要がなくなった。

■ブロックデバイス

ユーザーの立場から見て、Linux上のデバイスには3つの基本タイプがある。ブロックデバイス、キャラクタデバイス、ネットワークデバイスである。それらについて順に説明しよう。

ブロックデバイスとは、データが個々にアクセスできるバイトの配列であるものだ。ブ

ロックデバイスの一般的な例はハードディスク、フロッピードライブ、RAMディスクなどである。あるデバイスが（たとえば自動排出など）特殊な機能をもつ場合、任意のプログラムから使えるI/Oコントロールを使って、それらの特殊機能をサポートする。

IDEは、現在最も一般的にPCで使われている種類のディスクである。Linux 2.4ではLinux 2.2でのIDEのサポートを改良して、ひとつのシステム内で使えるIDEコントローラの数を倍以上の10個に増やした。また、DVDとCD-ROMチェンジャへのサポートを改善する、いくつかの変更もある。

Linux 2.4には、Linux 2.2でサポートされているファイルシステムがすべて含まれている。これらのファイルシステムにはFAT、NTFS、VFATおよびFAT32、HFS、その他非常に数多くのものがある。これらのファイルシステムは、すべて何らかの拡張のために書き直されていて、ときには新しいページキャッシュ機構をサポートするために、たいへん大規模な拡張が行われて、効率がよくなっている。しかし逆に、Linux 2.2用に設計されたバイナリのみのファイルシステムモジュールは、Linux 2.4では動かない。

■フレームバッファ

もうひとつ、より複雑なブロックデバイスのバリエーションはフレームバッファだ。フレームバッファとは単なる一区切りのメモリで、そこへの書き込みが画面上のピクセルの色に影響するような範囲のビデオメモリを表わすものだ。これは他のブロックデバイスよりも複雑だ。なぜならこれはパレット変更するioctlやその他のビデオ関連の関数に対応しているからだ。

Linux 2.4では数多くの新しいドライバの追加と、古いドライバの改良が行なわれている。特にここで重要なのは、数多くの「標準」VGAカードとその設定を、少なくともいくつかのモードでサポートしていることだ。

■キャラクタデバイス

Linuxが認識する、2つめの種類のデバイスがキャラクタデバイスだ。それらは読み出しや書き込みができるが「位置」の解釈ができず、順を追ってアクセスすることしかできない。ターミナル、ポート、キーボード、マウスなどがこれに含まれる。Linux 2.4はこの領域についても改良を加えている。

この方面で最も大きなニュースは、Linux 2.4が初めてUSB接続のキーボードとマウスをサポートすることだ。接続時にこれらの入力デバイスは、自分が「ふつうの」キーボードおよびマウスであるかのようにふるまう。

そのように見えないかもしれないが、すべてのバージョンのLinuxの画面出力はキャラクタモードだ。フレームバッファの場合、Linux 2.2とそれ以降ではターミナルドライバにフレームバッファドライバを重ねて、同一の機能を組み込みのテキストモードとして使う方法に対応している。Linux 2.4ではこのサブシステムに大きな変更はあまりない。しかし今回初めてコンソールから、たとえばプリンタなどのパラレルポートへのリダイレクトをサポートした。

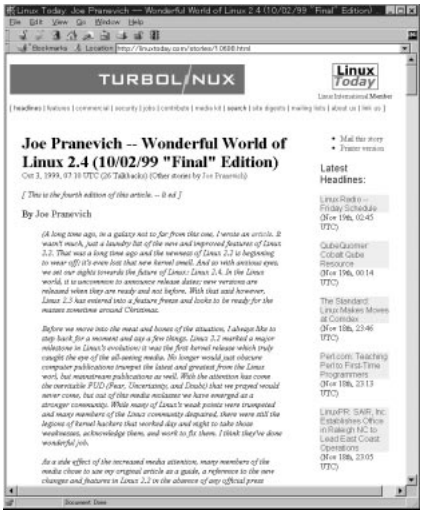
赤外線対応はLinux 2.2から進歩していて、この領域ではネットワークのサポートの改良などの多くの変更がある。

Linuxは一般的には「ユーザーフレンドリー」なOSとは思われていない。したがって、Linux 2.4に、音声合成カードへの対応が含まれていることを知って驚く人もいるだろう。

■ネットワークデバイス

キーボードやマウスの単純さとは大きく異なり、ネットワーク処理とネットワークハードウェアは、Linuxが常に群を抜いている部分である。これらのデバイスは「キャラクタ」でも「ブロック」でもなく、デバイスノードが不要な特別な位置にある。

Linuxのネットワークソケットは、標準的なUNIXのものと同じである。しかし残念ながら、標準にはいくつかの欠陥がある。ただし、それらは標準から外れることなく修正することはできる。Linux 2.2とそれ以前のバージョンでは、ネットワークソケットからの1つのイベントを待っている多くのプロセスが存在する場合、ソケットがアクティブになると、化それらのすべてが動き出す。Linux 2.4にはLinux上で「wake one」を実装する変更点が含まれていて、アクティブ化の際に、1つのプロセスだけを起動する。これによりApacheのようなアプリケーションはずっと効率がよくなる。



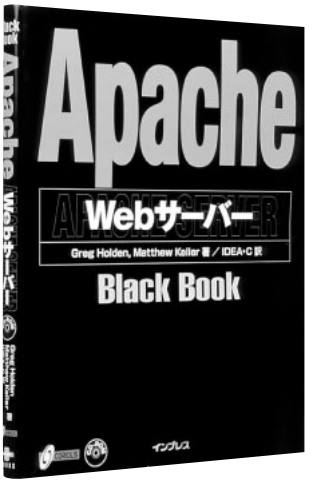
また、Linux 2.4のネットワーク層は完全に書き直されている。さらに、この中には、Windowsを含む多くの一般的なOSの中で使われるネットワーク処理用のスタックの特有の癖に、うまく対応できるようにするための多くの最適化が含まれている。

Linuxを端末として使うユーザーにとって、PPPは重要な部分である。Linux 2.4にはいくつかの大規模な書き換えと、多くのコードのモジュール化が行われ、その中には長いこと待たれていたISDN層からのPPP層と、シリアルデバイス上で使われるモデムなどのようなPPP層との組み合わせが含まれている。

結果として、おそらくLinux 2.4はその新しいデスクトップ機能によって「デスクトップ」Linuxとして知られることになりそうだ。誰かが私の記事を、「Linux 2.4は、サーバや組み込みシステムに非常に役立つ多くの機能を備えている」というように引用しないように願っている。だがこれらには、実際問題、本質的な違いはあるのだろうか？

プロフィール
Joe PranevichはLinuxカーネルの開発に関わっており、数々のパッチも提供している。
jpranevich@linuxtoday.com

Books



Apache WebサーバーBlack Book

Greg Holden, Matthew Keller 著　IDEA・C 訳
インプレス
B5変形判／339ページ／CD-ROM付き　本体価格 2980円

ある日突然、社命でWebサーバを管理することになったら、サーバプログラムには何を選ぶべきか、選択肢は多い。最近ではサーバ用OSを買ったら、ただで付いてくるものまである。だが、現在世の中で最も多く使われているのは、フリーで手に入るApacheである。多くの手によって育まれたApacheは機能が豊富だが反面、設定には慣れが必要だ。本書はそんなApacheの解説書である。初めてApacheに接する人なら、第一章「Apache入門」、第二章「設定ファイルの使用」を読んで、まず動かしてみよう。また、各章が分野別の「○○をするには」を集めた構成になっているため、すでにWebサーバの運用をしている人にとっては、リファレンス的に役立つ。

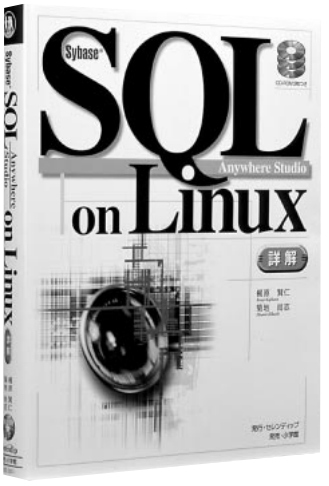
各プラットフォーム向けのApacheを収録したCD-ROMが付属しており、すぐに使い始められる。にわか管理者から中級者まで、お勧めの一冊だ。

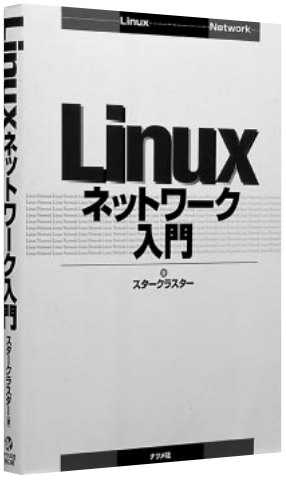
Sybase SQL Anywhere Studio on Linux 詳解

梶原賢仁、菊池尚志 著
セレンディップ
B5変形判／424ページ／CD-ROM付き　本体価格 4500円

Sybase SQL Anywhere Studioは、RDBMSのAdaptive Server Anywhereデータベースサーバと、データベース管理支援ツールをセットにしたものである。Linux版は現在βバージョンを公開中だが、近日中に発売される。クライアント／サーバ型でもスタンドアロンでも使用でき、OSを除いた必要なメモリが数Mバイトと小さいため、Windows用のSybase SQL Anywhere StudioはモバイルDB市場で広く使われている。

本書では、クライアントにWindowsマシン、サーバにLinuxを使用してリレーショナルデータベースを構築していく。付録CD-ROMは3枚組で、日本語redhat Linux 5.2 rel.2と、60日間無償で試用できるSybase SQL Anywhere StudioのLinux版とWindows 95/98/NT版を収録している。ネットワークでつながる2台のPCを用意することで、動作確認をしながらSQLデータベースについて学べるだろう。





Linuxネットワーク入門

スタークラスター 著
ナツメ社
A5判／272ページ　本体価格 2200円

本書は、Linuxでゲートウェイサーバを構築することを目的としたネットワークの入門書だ。導入部では、ネットワークとはどのようなものかを、装置からプロトコル、インターネットの仕組みまで丁寧に解説し、中盤ではLinuxでのイーサネットLAN、PPP接続の設定のしかたを説明する。そして、後半はサーバプログラムの設定へと展開する。ここでは、DNSサーバ、DHCPサーバ／クライアント、IPマスカレードの話が中心となる。

対象とするディストリビューションは、Red Hat Linux、Debian GNU/Linux、Slackwareで、それぞれの設定方法が違う場合は個別にページを割いている。

あくまでゲートウェイサーバ構築を目的としているので、Apache、sendmail、Squidなど、ゲートウェイ以外のサーバサービスには触れていない。これらのサービスを利用する場合には、ほかの解説書などとあわせて読むとよいだろう。

WindowsからLinuxへの近道 Linux超入門

男澤 昌哉、中山 房光夫 著
セレンディップ
B5変形判／352ページ／CD-ROM付き
本体価格 2800円

Linuxクイックリファレンス 第2版

Ellen Siever、O'Reilly & Associates, Inc. 著 山崎 康宏、山崎 邦子、砂畑 浩樹 訳
オライリー・ジャパン
A5判／673ページ
本体価格 4500円

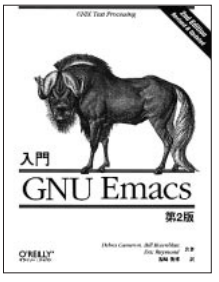
Oracle8 for Linux データベースサーバー導入計画

秋本 芳伸、岡田 泰子、青島 純一 著
翔泳社
B5変形判／548ページ／CD-ROM付き
本体価格 4500円

GTK+／GDKによるLinuxアプリケーション開発

Eric Harlow 著 アンク 訳
翔泳社
B5変形判／512ページ
本体価格 3400円

入門GNU Emacs 第2版



Debra Cameron、Bill Rosenblatt、Eric Raymond 著、福崎 俊博 訳
オライリー・ジャパン
B5変形判／584ページ
本体価格 4800円

図解でわかるLinuxサーバーのすべて

多比羅 悟 著
日本実業出版社
A5判／264ページ
本体価格 1800円

最近売れてるLinux書籍は？

このところのLinux書籍は、「読み物」がロングセラーになっています。たとえば、『オープンソフトウェア』（OPENSOURCES、Chris DiBona／Sam Ockman／Mark Stone著、倉骨 彰訳、オライリージャパン刊、ISBN4-900900-95-8）は、弊店の調査結果では、発売以来18週のうち14週がベスト10入りを果たしています。さらに、『伽藍とバザール』（Eric Raymond著、山形 浩生訳・解説、光芒社刊、ISBN4-89542-168-6）も、8週のうちベスト10入りが6週あります。先月紹介した『特選 星降る夜のパソコン情話〜Linux狂騒曲〜』（中村 正三郎著、ビレッジセンター出版局刊、ISBN4-89436-131-0）も、発売以来売り上げ1位を独占し続け、勢いは衰えていません。これらの3冊は、まだまだ上位を独占するだけの力を持っているようです。

読み物以外では、『PC-UNIXサーバのためのクラッカー撃退計画』（まえだひさこ著、翔泳社刊、ISBN4-88135-800-6）が、8週連続で第2位をキープ（第1週だけ1位）しており、手堅い人気です。さらに、Red Hat LINUX Secrets, 2nd Editioinの訳本『LINUX大全』（Naba Barkakati著、橘 康雄／中川 和夫訳、ソフトバンクパブリッシング刊、ISBN4-7973-0832-X）は、Vine Linux 1.1と英語版のRed Hat Linux6.0が収録された2枚の付属CD-ROMが好評のようで、発売から6週のうち4週がベスト10入りしています。

新刊では、待望の『Qtプログラミング入門』（Programming with Qt、Matthias Kalle Dalheimer著、杵渕 聡訳、 木 淳司監訳、オライリージャパン刊、ISBN4-87311-007-6）が出版されました。Qtは、ノルウェーのTrolltech社によって開発され、KDEなどで利用されているGUIツールキットです。本書には、このQtに関するプログラミングテクニックはもとより、Qtを使ったOpenGLプログラミング、Perlからのアクセスに関する情報も収録されており、至れり尽くせりの内容です。

一方、GNOMEやGIMPなどで利用されているGTK+のほうもプログラミングの解説書が何冊か出版されていますが、一番最初に出版された『GTK+ではじめるXプログラミング』（竹田 英二著、技術評論社刊、ISBN4-7741-0789-1）が依然として人気が高いです。 （書泉ブックドーム 古田島）

読者の声

俺にもいわせろ！

寒い日が続きますが、みなさまお元気でしょうか？　え？　マシンの廃熱で冬もポカポカ？　えーと、それは……う〜ん……。

Linux on ノートPCへのご意見

12月号の特集であるノートへのインストールは大変勉強になりました。

次はぜひアプリケーションの特集をしていただきたいと思います。フリーの表計算ソフトがあれば紹介して欲しいです。

（神奈川県　鈴木章仁さん）

「Linux on ノートPC」は少々期待はずれ。もう少しつっこんだ内容が欲しかった。ノートにインストールはしたものの、Linux初心者の私にとって大部分の記事は難しく感じた。初心者からベテランまで、どの層をターゲットにするか、難しいでしょうね。

（栃木県　まるよしさん）

「Linux on ノートPC」にはたくさんのご意見をいただきました。ノートPCには、ハードウェアが独自ものもあり、それに伴う問題やインストールの難しさもあります。いつかまた特集を組みたいと思いますので、引き続きご意見をお願いいたします。

フリーの表計算ソフトといえば、68ページで紹介したgnumericがお勧めです。商用ソフトでは、TurboLinux PRO 日本語版4.2にバンドルされるApplixwareはいかがでしょう。

Sound Blaster Live!を鳴らす

Sound Blaster Live!が使える“フリー”なLinuxはでないのかな。

（岡山県　桝平雄介さん）

11月号の「Linuxでサウンド三昧」でお伝えしたとおり、Sound Blaster Live!の販売元Creative Technology社が開発中のドライバを公開しています（http://developer.soundblaster.com/linux/）。また、本誌発売のころまでにALSA projectのドライバにも組み込まれる予定です（http://www.alsa-project.org/）。開発版ということもあり、使うにはちょっと手間がかかるようです。早く正式版になると良いですね。

Windowsとの親和性ならMLD Linux？

Linux MLD 4のレポートをお願いします。製品紹介でも可。

（愛知県　山口 猛さん）

今月の特集はいかがでしたでしょうか。Windows 9xをメインに使っている方には便利なディストリビューションですね。最新版のLinux MLD 4は、12月10日発売です。

真のキラーアプリはコレだッ！

近頃、ようやく自分専用のパソコンを持ち、LASER5 Linux 6.0、Turbo Linux 3.0を入れた。安定しているといわれるLinuxだが、なかなか安定起動しない。こうしている間にも…ハッ、GNOMEのランチャがなくなっている!!というわけで、設定ファイル特集やってください。お願いです。

P.S.　キラーアプリは「北斗の拳」（タイピングソフト）だ!!

（東京都　大村 裕さん）

GNOMEは、新しいプログラムなのでまだちょっと不安定ですね。

先月のポストペットに続き、Linuxのキラーアプリ候補第2弾はタイピング練習ソフトの「タイピング奥義　北斗の拳　激打」ですね。新製品の「タイピング泪橋　あしたのジョー　闘打」もLinuxには対応していません（^^;）。Linux版どうでしょう？＞株式会社SSIトリスターさん。

LASER5 LinuxのXFree86

実は、Linux magazineは付録のCD-ROMを目当てに購入しました。Red Hat Linux 6.1を使ってみたかったので……。LASER5 Linux 6.0をパッケージ購入して使ってみたのですが、元々使っていたビデオカードをうまく認識せず、Linuxを使うときだけ古いビデオカードに差し替えていました。でも、

試しに使ってみたRed Hatは、新しいカードの方でX Window Systemを使えてしまったので、大感激です。ということは、LASER5もXFree86のバージョンをUPすればOKということですね。頑張ろう。

（埼玉県　三浦 智さん）

ビデオカードを差し替えてというのは大変ですね。レーザーファイブのftpサイトでXFree86の最新版3.3.5のRPMパッケージが配布されています（ftp://ftp2.laser5.co.jp/pub/contribl5linux/i386/）。

ただし、無保証ですので、それを踏まえた上でご利用ください。幸運を祈ります。

Linux widow？

うちのだんなは、Linuxにはまって妻の私の相手をしてくれません。1月に出産を控える私は不安です。

ですが、プレゼントは当てて欲しいなぁ。

（岡山県　久富隆史さんの奥様）

お子さんが生まれれば、旦那様も変わるでしょう（あ、今度はお子さんばかりで奥様はかまわない……？）。元気なお子さんの誕生をお祈りしています。

立ち上がれ！　ノート君っ！

Intel 468DX 75MHzのノートにPCMCIAのネットワークカードを2枚差してIPマスカレードを動かそうとしています。Red Hat Linux 6.1を入れてみたのですが、Xを入れるとインストーラが途中でコケてしまって悪戦苦闘中です。いつになったらこのノート君はLinuxマシンとして立ち上がってくれるのか？

（千葉県　柳 淳二さん）

その後、うまく動くようになったでしょうか？　今でも動いていないのなら、テキストベースのインストーラを使い、Xなしでインストールしてみてはいかがでしょうか。

Linuxが動くようになったら、rpmコマンドでXFree86をインストールし、XF86SetupなどでXの設定をすればうまくいくかもしれません。

家内安全にLinux

主人の影響でパソコンを始めました。Linuxでは、フリーのゲームを楽しんでいますが、初心者が利用できるソフトの操作性などの紹介を特集して欲しいです。

（愛知県　伊藤純子さん）

お待たせしました。次号は、フリーソフトの特集です。ゲームも含め、100本のソフトを一挙に紹介します。Windowsの世界とはひと味違ったソフトが目白押しです。

いぶし銀のような……

いまだにNEC PC-UX Rev.2ユーザー。

（神奈川　横田和海さん）

シブイですね。

PC-UXというのは、NECのPC-9800シリーズ用のUNIXです。主に、80年代後半〜90年代前半に活躍していたそうです。どのようなマシン構成なんでしょうか。ちょっと気になります。

Challenging spiritは最大の武器

Linuxはわからないことばかり！　ハードディスクのWindows 98はフォーマットしてしまうし、もうやめよ〜かなと思ってまた挑戦している私です。

（福井県　山田 喜代治さん）

Linuxに限った話ではありませんが、何度も失敗した方が上達することってありますよね。失敗は決して無駄になりません。おっと、「エラソーに。お前は失敗から学んでないゾ」と先輩編集者よりツッコミが入りました。

それはさておき、Windowsとの共存のしかたについては、今月の特集2はお役に立ちましたでしょうか？

Linuxビジネスの難しさ

仕事をするふりをしてLinuxをイジっていたが、今度本業にすることにした。友人は納得したが、年老いた父母に「Linux」を説明するのはまだまだ難しい。「タダのモノを配って商売にしている」とか言うと、絶句されてしまった（言い方が悪いともいうが）。

（神奈川県　竹田寛郁さん）

新しいお仕事、頑張ってくださいね。応援しています（私の両親は、私が働いていること自体を信じていないようです）。

えええええっっ (^^;; その2

BeOSも結構おもしろそうですね。特集お願いします（ちょっとちがうかな？）。

（愛知県　山口 猛さん）

はい、その特集はちょっと……。

今月のごめんなさい

1999年12月号の特集1「Linux on ノートPC」の52ページでTurboLinux 日本語版 4.0のインストール元メディアとしてハードディスクとSMBサーバが利用できるという記述がありますが、これらのメディアからのインストールはできません。

お詫びして訂正いたします。ご指摘くださいました岐阜県の勝股 充さんに感謝いたします。