News Express

Distribution Software Hardware Headline Event

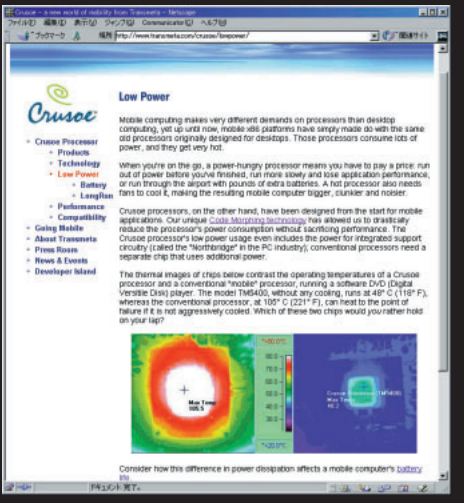
新x86互換CPU「Crusoe」登場

1月19日に、これまでまったく無名（Linus氏が勤めている会社としては有名？）だった米Transmetaから、「Crusoe（クルーソー）」という新CPUファミリーの発表があった。PDAあるいはモバイル向け超低消費電力プロセッサTM3120と、WindowsノートPC向け高性能プロセッサTM5400という2種類で、CPUの内部アーキテクチャはVLIWを採用し、コードモーフィングという方法でx86のインストラクションをソフトウェアで変換して実行するとのことだ。

1999年に、Cyrix（ナショナルセミコンダクター）、IDT（Integrated Device Technology）という2社がx86互換CPU市場から撤退したため、Intel対AMDという構図になっていたところに、Transmetaがどう食い込んでいくのか楽しみである。

TM5400は、ダイナミックに動作周波数を変えたり、同時に動作電圧を落としたりして消費電力を下げる「LongRun」という技術を備えている。量産出荷はまだ先のようだが、TM5400を搭載したノートPCが早く見たいものである。インテルも「SpeedStep」という省電力機能を搭載したモバイルPentium IIIプロセッサを発表しているので、今年発売されるノートPCは、今までよりバッテリー持続時間が長くなっていくだろう。

そして、もうひとつ注目なのは、TM3120を搭載するPDA用のOSとしてモバイルLinuxが開発されていることだ。Flashメモリだけでハードディスクを搭載しないシステムで動作する。Linus氏がこういう携帯型PCのLinux開発に力を入れたら、来年はLinuxは持ち歩くのが普通になるのかも。



Hardware　発売日　2000年1月

モバイル向けCPU「Crusoe」

URL　http://www.transmeta.com/

発売　Transmeta
TEL
価格　65USドル～（TM3120）、119USドル～（TM5400）

Linus Torvaldsが在籍することで知られる米Transmeta社は、1月19日にモバイル向けCPU「Crusoe」ファミリーの「TM3120」と「TM5400」を発表した。

Crusoeは、内部的にはVLIW（Very Long Instruction Word）アーキテクチャのCPUだが、コードモーフィングソフトウェアを用いることで、x86互換CPUとして機能する。コードモーフィングソフトウェアは、アップデート可能であり、x86の命令セットが拡張されたとしても、ハードウェアはそのままで新しい命令に対応できる。

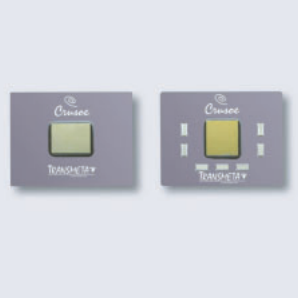
またもうひとつの特徴として「LongRun」と呼ばれる電力制御機能を備えている。LongRunは、CPUへの負荷に合わせてクロックと電圧を毎秒数百回調節することで、低消費電力を実現する機能。

TM3120は低価格のモバイルインターネット接続機器向けで、モバイルLinux OSとの組み合わせが想定されている。すでに量産中で、価格は333 MHz版が65USドル、400 MHz版が89USドル。

TM5400はWindows用の軽量ノートPC向けで、高性能と低消費電力の両立が可能とされている。現在サンプル出荷が行われており、価格は500 MHz版が119USドル、700 MHz版が329USドル。

製造プロセスは、TM3120が0.22μm、TM5400が0.18μmで、どちらも474ピンのBGAパッケージで提供される。

同社では2000年中頃までに、Crusoeを用いた機器が登場するとしている。



Software　発売日　2000年2月9日

Linux学習パッケージ「TurboLinuxラーニングキット」

URL　http://www.turbolinux.co.jp/

発売　ターボリナックス ジャパン株式会社
TEL　03-5766-1660
価格　4980円

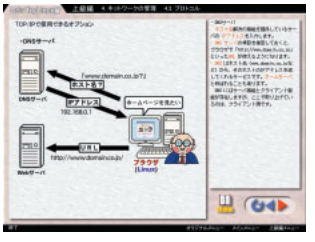
ターボリナックス ジャパンは、初心者向けのLinux学習用パッケージ製品「TurboLinuxラーニングキット」を2月9日から発売する。価格は4980円。

同社がFTPサイトや雑誌で公開・配布予定のLinuxディストリビューション「Turbo Linux 日本語版 4.5」（無償版）に、PFUアクティブラボと東電コンピュータサービスが共同開発した、Windows上で動作するLinux学習ソフト「Let's Try Linux編」と米V Communications社のユーティリティソフト「System Commander Lite for TurboLinux」を組み合わせたもの。

「System Commander Lite」により、1台のマシン上でWindowsとTurboLinuxを切り替えて使用できる。

「Let's Try Linux編」は、Windows 9x／NT 4.0に対応しており、Windows上でLinuxの基礎知識やコマンドなどを学習できる。Linuxのインストールを疑似体験することも可能で、初心者が実際にインストールを行う際の負担を低減できるとしている。なお同ソフトを使用するには、Sound Blasterまたは互換のサウンドカードが必要となる。

「TurboLinux 日本語版 4.5」は、カーネルは2.2.13、glibcは2.0.7、XFree86は3.3.5／3.3.6、GNOMEは1.0.54（Octoberリリース）、KDEは1.1.2を採用している。なお、商用ソフトは含まれていない。



Software　発売日　2000年1月31日

サーバ管理ツール「HDE Linux Controller 1.0」

URL　http://www.hde.co.jp/

発売　株式会社ホライズン・デジタル・エンタープライズ
TEL　03-5456-3260
価格　3万円

ホライズン・デジタル・エンタープライズは、Linux用のサーバ管理ツール「HDE Linux Controller 1.0」（以下HDE-LC）を1月31日に発売した。

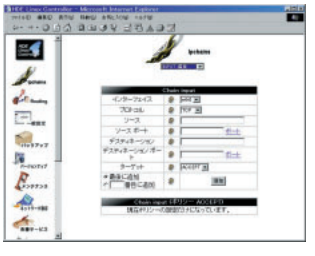
HDE-LCは、無償配布版「HDE-WUI」の商用版で、LinuxサーバをWebブラウザから設定、操作できる管理ツール。

ユーザー管理／IPアドレス／DNS設定、リソースチェック、各種サーバの起動／停止など、HDE-WUIと同等の機能に加え、ipchainsの設定やルーティング情報の設定などインターネットサーバ向けの新機能を持つ。

動作確認済みのディストリビューションは、Red Hat Linux 5.2／6.0、TurboLinux 日本語版 4.0、TurboLinux Server 日本語版 6.0、LASER5 Linux 6.0。

クライアント側の対応OSは、Windows 9x／NT 4.0、漢字Talk7.5.5以降。Internet Exproler 4.0以降またはNetscape Communicator 4.5以降のWebブラウザに対応している。

価格は3万円。90日間のサポート（電話、メール）、インターネット上のアップデートサーバによるバージョンアップ権（90日間）が付属する。



Software

Webアプリケーションサーバ「ロータス ドミノR5」

発売日　2000年1月28日

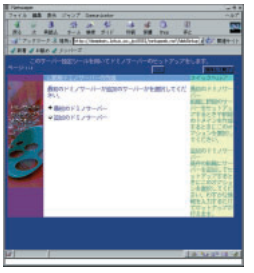
発売　ロータス株式会社
TEL　03-5496-3230
価格　9万9400円～（1サーバあたり）

URL　http://domino.lotus.co.jp/

ロータスは、Webアプリケーションサーバ「ロータス ドミノR5」の、Linux対応版を1月28日に発売した。対応するディストリビューションは、Red Hat Linux 6.0／6.1（英語版＋日本語ロケール）となっている。発売中のWindows NT版、UNIX版、OS/2版と同等の機能を持つ。

電子メール、カレンダー＆スケジュール、ディスカッション、チームルームなどグループウェアとしての機能を持つ。またOracleやDB2などの基幹DBMSとの連携機能を持ち、ドミノのクライアントやWebブラウザからリアルタイムで、DBMSの参照・変更・作成・削除を実行できる。

大塚商会は、Linux版ドミノR5にハードウェアや導入作業、サポートをセットしたソリューションパッケージの提供を予定している。販売開始時期、価格は未定。同様のサービス提供は、日立製作所、日本アイ・ビー・エムでも検討中。



Hardware

FM／TVビデオキャプチャカード「PicPom LINUX」

発売日　1999年11月12日

発売　株式会社インタウェア
TEL　03-5411-8010
価格　オープンプライス（直販価格 1万8800円）

URL　http://www.interware.co.jp/

インタウェアは、PC/AT互換機上のLinuxに対応したFM／TVビデオキャプチャカード「PicPom LINUX」を発売した。オープンプライスだが、同社のWebサイトからの直販価格は1万8800円。

チューナー部分は、TVがVHF1～12、UHF13～62チャンネル、FMが70～108MHzに対応している。最大640×480ドットで表示可能だが、フルモーションキャプチャできるのは、320×240ドットまで。

ビデオキャプチャ、FMラジオ、Webでの画像公開、音声録音の各ユーティリティが付属している。これらのユーティリティとドライバのソースコード、Linux上でビデオを扱うためのAPIである「Video4Linux」の解説の日本語訳も収録されている。

推奨使用環境は、カーネル2.2.5以降、XFree86 3.3.3以降。またサウンドカードが正しくインストールされていることが必要になる。

Hardware

無音IPルータ「COOLGREEN Flex Router」

発売日　2000年1月20日

発売　クールグリーンコンピュータ有限会社
TEL　045-640-3208
価格　3万9800円

URL　http://www.coolgreen.co.jp/

クールグリーンコンピュータは、Linuxカーネル採用のIPルータ「COOLGREEN Flex Router（フレックスルーター）」を1月20日に発売した。

IPマスカレード、ファイヤーウォール、DHCPサーバ／クライアントなどの機能を持っている。

CPUにIDTのWinChip C6 150MHzを採用し、32Mバイトのメモリ、3.5インチFDDを備える。ネットワークは10／100BASE-TXに対応している。消費電力は平均17W（最大35W）。本体サイズは78（W）×279（H）×288（D）mmで、重量は約5kg。

フロッピーディスクから起動可能なようにカスタマイズされたLinuxを用いている。またファイルシステムは、RAMディスク上に作成されているため、システム終了時には電源スイッチを切るだけでよい。

PCのノイズの原因とされるハードディスクや冷却ファンを使用していないため、起動時のフロッピードライブの動作音以外は、無音での運用が可能としている。

Hardware

ラックマウントサーバ「Raxys-CS」シリーズ

発売日　2000年2月1日

発売　ロジカルイフェクト株式会社
TEL　03-5822-3322
価格　14万7000円～

URL　http://www.villagecenter.co.jp/

ロジカルイフェクトは、ラックマウントサーバ「Raxys-CS」シリーズを2000年2月1日から発売する。汎用部品で構成することで、低価格を実現したという。同社によれば、キャッシュサーバ用途で、年間3000台の販売を見込んでいる。プリインストール可能なOSは、Windows 98／NT Workstation／NT Server、Linux、FreeBSDなど。

Raxys-CSは、1U、2U、4Uの3タイプがラインナップされている。1Uサイズの「Raxys-CS 1U Si8」は、CPUがCeleron 466MHz～Pentium III 550MHz、メモリが64～512Mバイト、HDDが13～27Gバイトの範囲で選択可能。価格は19万7000円から。

2Uサイズの「Raxys-CS 2U NX」は、CPUがPentium III 500M～700MHz、メモリが64～512Mバイト、HDDが13～27Gバイトの範囲で選択可能。価格は29万7000円から。

4Uサイズの「Raxys-CS 4U V」は、CPUがPentium III 500～700MHz、メモリが64～512Mバイト、HDDが13～27Gバイトの範囲で選択可能。価格は14万7000円から。

セキュリティ統合ソフト「Check Point 2000 Edition」発売
2000年1月27日

イスラエルのCheck Point Software Technologiesは25日、Linuxをサポートしたインターネットセキュリティ統合ソフト「Check Point 2000 Edition」を2月に発売すると発表した。価格は未定。

すべてのファイアウォールおよびVPNゲートウェイを介したトラフィックパターンを分析し、不正や疑わしいイベントを検出する機能を備える。不正行為を検出すると、管理者に自動的に対応方法を通知する。

またセキュア認証APIがサポートされ、サードパーティ製の声紋や指紋などの認証製品なども使用できる。

管理ツールとして新たに「Visual Policy Editor（VPE）」が追加された。VPEは、ファイアウォール、VPN、サーバ、ネットワーク、ルータなどのオブジェクト、およびそれらの相互関係をグラフィック表示するソフトウェア。

対応プラットフォームは、Red Hat LinuxとWindows 2000。

Check Point Software Technologies
（http://www.checkpoint.co.jp/）

Red Hat、TurboLinuxがIBMのJava技術ライセンスを取得
2000年1月25日

米Red Hat社、米TurboLinux社の両社は24日、米IBMのLinux用Java開発者向けキット「JDK for Linux」のライセンスの取得を発表した。

JDK for Linuxは、SunのJava互換性試験をパスしており、「JVM（Java仮想マシン）」が含まれている。現在のバージョンは1.1.8。JDK for Linuxが含まれることにより、ユーザーはJava環境を別途インストールする手間から解放される。

両社は次期製品からJDK for Linuxを導入する予定。

Red Hat（http://www.redhat.com）
TurboLinux（http://www.turbolinux.com/）

組み込み用途のEmbedix Linux 1.0リリース
2000年1月26日

米Lineo社は、組み込み用途の「Embedix Linux 1.0」をリリースした。

同製品は、組み込み機器用に調整されたディストリビューションで、カーネル2.2.xベース、x86とPowerPCに対応、8Mバイトの RAMと、3Mバイトの ROMもしくはフラッシュROMが必要、組み込み機器向けの、サイズが小さいHTTPサーバや、ネットワーク機能を実装、といった特徴を持つ。

また同製品は、同社の組み込み機器向けのLinux製品の最初のリリースであり、2000年第2四半期には、「Embedix Linux」の統合開発環境である「Embedix SDK」や、コンパクトなWebブラウザ「Embedix Browser」を、また2001年第1四半期には、「Embedix Linux」上にWindows CE互換レイヤーを提供することによって、Windows CEアプリケーションの「Embedix Linux」への移植を可能にする「Embedix PDA」などの製品出荷を予定しているという。

米Lineoは、米Caldera Systemsの子会社で、MS-DOS互換の「DR-DOS」を組み込み機器向けに販売しているソフトウェアベンダー。現在は開発の主力をLinuxにシフトしつつある。

Lineo（http://www.lineo.com/）

ターボリナックスジャパン、伊藤忠テクノサイエンスほか3社と提携
2000年1月24日

ターボリナックスジャパンは24日、伊藤忠テクノサイエンス、東洋情報システム、ダイヤモンドPC、理経とLinux関連分野で提携した。今回の提携は、ハイエンドLinux市場におけるソリューション開発やサポート強化のため、技術、サポート、営業の分野で、協業体制を確立するためのもの。

また同社は「TurboCluster Server」のリリース（2000年第1四半期予定）に合わせ、クラスタ技術に関しても協業体制を強化していくという。

ターボリナックスジャパン
（http://www.turbolinux.co.jp/）
伊藤忠テクノサイエンス
（http://www.ctc-g.co.jp/）
東洋情報システム
（http://www.tis.co.jp/）
ダイヤモンドPC
（http://www.diapc.co.jp/）
理経（http://www.rikei.co.jp/）

LinuxPPCのCEO、MACWORLD Expo/Tokyo2000で講演決定
2000年1月24日

「LinuxPPC」の正規代理店である市川充商店によれば、米LinuxPPCのCEO、Jeff Carr氏が、千葉・幕張メッセで開催される「MACWORLD Expo/Tokyo2000」で講演することが正式に決定した。日時は2月17日の14時45分～16時。

講演のテーマは「Think Linux! LinuxPPCの世界」。内容は、LinuxPPCの紹介、最新製品「LinuxPPC 2000」のデモンストレーション、今後の展望、米IBMとの共同作業が進むPowerPCマシンに関する情報など。さらに、米国New Yorkで2月1日から始まる「LinuxWorld Conference」で発表する最新情報を日本でも聴講できるという。

LinuxPPC（http://www.linuxppc.com/）

Apache 1.3.11リリース
2000年1月24日

Apache Software Foundationは、Apache 1.3.11をリリースした。Win32版、IBMのOS390版の改善やバグフィクスが行われている。

Apacheは、世界で50%以上のシェアを持つオープンソースのHTTPサーバで、Apache Software Foundationは、Apacheの開発をするために設立された組織。

Apache Software Foundation
（http://www.apache.org/）

TurboLinux、中国市場で米Microsoftを破ったと発表
2000年1月17日

米TurboLinuxは、中国語版のLinuxディストリビューション「TurboLinux簡体中文版4.0」が、米MicrosoftのWindows 98のアップグレード版や、Windows NT Server 4.0の販売実績を上回ったと発表した。

これは、256店舗ある中国国営のソフトウェア販売店「連邦軟件公司」の販売実績（1999年8月中旬～12月中旬）によるもの。

TurboLinuxは売上トップになった要因として、中国政府の支持、中国のインターネット経済の急成長、同製品の価格49USドル（約5154円）が、Windows 98の245USドル（約2万5770円）と比較して安いことなどを挙げている。

同社は、現地法人を北京に設立し、中国語（簡体字および繁体字）をサポートしたLinuxディストリビューションを市場に投入した。また、開発チームを北京に設置し、中国市場のニーズに応じたサポート体制を敷いている。最近では中国大手ハードウェアメーカー3社とバンドル提携を結ぶなど、市場拡大を図っている。

TurboLinux（http://www.turbolinux.com/）

フリーのWebグループウェア「TrueOffice」登場

2000年1月15日

QUPA.comは、掲示板やスケジュール管理、施設の予約などの機能を持ったグループウェア「TrueOffice Version-Revison 01-02」をダウンロード可能にした。同ソフトウェアはフリーのため、誰でも無料で使用できる。ただし、TrueOfficeのインストール作業やサポート作業を、サービスとして他者に提供する場合は有料となるなど、若干の制限がある。

「グループ管理」「スケジュール管理」「行き先表示」「施設予約」「掲示板」「ToDo管理」「おしらせ」「WEB LINK」といった、グループウェアとしてはスタンダードな機能を持つ。WEB LINKとは、共有ブックマーク機能で、メンバーが登録したURLを、全員で共有することができるもの。

プログラムはすべてC言語で記述され、PerlやJavaなどで作成された同種のソフトウェアよりもメモリ効率がよく、高速に動作するという。カスタマイズについては、テンプレートのHTMLファイルを編集することで行える。さらに、ユーザー独自の処理をプラグインとして組み込み可能にする仕組みを開発中だという。

対応するプラットフォームは、x86のLinux（カーネル2.x以上）のほか、FreeBSD（x86）、Solaris（x86、Sparc）、Windows 9x／NTとなっている。

また対応Webサーバは、Apache、NCSA httpd、Internet Information Serverなどである。

QUPA.com（http://tr.qupa.com/）

XFree86 3.3.6リリース

2000年1月13日

XFree86 Projectから、XFree86 3.3.6がリリースされた。

XFree86は、LinuxやFreeBSDなどのPC UNIXで採用されているX Window Systemの実装のひとつ。

XFree86 3.3.6はメインテナンスリリースであり、バグフィクスと対応ビデオチップの追加以外に大きな変化はない。新たにサポートされたビデオチップとビデオ機能内蔵チップセットは、

Rage 128／Rage Mobility（ATI）
SiS540／630／300（SiS）
Lynx（Silicon Motion）
Savage2000（S3）
GeForce256（NVIDIA）
i810（Intel）

となっている。

また、XFree86の次期バージョン4.0のプレリリースである3.9.17も1999年の年末からダウンロード可能になっている。

XFree86 Project（http://www.xfree86.org/）

Corel LINUX OS上で、Windowsアプリケーションのリモート実行が可能に

2000年1月8日

カナダのCorelは、「Corel LINUX OS」に、米GraphOnの「Bridges」を統合することを発表した。これにより、Corel LINUX OSはどのようなネットワーク上でもシームレスにWindowsアプリケーションを使用できる、初めてのLinuxディストリビューションとなる。

GraphOnのBridgesは、ローカルマシン上からリモートマシン上のアプリケーションを実行するためのソフトウェア。対応するOSとして、Windows 9x／NT、Linux、Solarisなどが挙げられている。

BridgesのLinux用クライアントと、Windows NT用のサーバの両ライセンスを含んだCorel LINUX OSは、2000年の中頃に出荷が予定されている。

Corel（http://linux.corel.com/）
GraphOn（http://www.graphon.com/）

Intel、インターネット専用端末「Web Appliance」にLinuxを採用

2000年1月6日

米Intelは5日、インターネット専用端末「Web Appliance」に関する事業計画を発表した。同端末は、インターネット接続機能およびメールの送受信機能と電話機機能を統合したものとなる。CPUは同社のCeleronを搭載し、OSにはLinuxが採用される。

同社の事業計画は、Intelブランドの Web Applianceを、通信事業者やISP（インターネット接続業者）を通じて、一般ユーザーに販売するというもの。通信事業者やISPには、同端末のリモート管理ソフトも提供する予定。

日本法人のインテルは、同端末をNECのインターネット統合サービス「BIGLOBE」の事業部と日本市場向けに共同開発しており、最初の製品を2000年7月頃に出荷する予定だという。サイズは未定だが、据置き型になるようだ。価格は未定。

Intel（http://www.intel.com/）
インテル（http://www.intel.co.jp/）
BIGLOBE（http://www.biglobe.ne.jp/）

Windowsエミュレータ「Wine」#24リリース

2000年1月5日

Windowsエミュレータ「Wine」がマイナーバージョンアップされ、#24になった。Wineは、X Window System上でWindowsのエミュレーションを行うソフトウェア。対応OSは、Linux、FreeBSD、Solarisなど。#24では、「Microsoft Word97」の「Webレイアウト」モード時の画面表示の不具合を解消している。なお、コンパイラ「gcc 2.95」を使用することで、Windowがあちこちに動き回るバグを回避できるという。

Wine Headquarters
(http://www.winehq.com/)

Inprise、「InterBase 6」のソースコードを公開

2000年1月4日

米Inpriseは3日、同社が開発中のRDBMS（Relational Database Management System）「InterBase 6」のソースコードを2000年の第1四半期に公開すると発表した。

「InterBase 6」の対応プラットフォームは、Linux、Solaris、Windows NTなど。同社がInterBaseのソースコードを公開するのは初めてとなる。

Inprise（http://www.inprise.com/）

CD-R有償配付サービス「Linux4u.net」正式運用開始

2000年1月4日

Linux関連のパッケージや、LinuxディストリビューションをCD-Rで配付するサービス「Linux4u.net」が、試験運用を終え、正式運用を開始した。

配付されるパッケージは、「ディストリビューション安定版」と「ディストリビューション開発版、Linux関連パッケージ」の2つに分類されている。

例を挙げると、安定版は、Vine Linux、Debian GNU/Linux、Kondara MNU/Linux、Plamo Linux、Red Hat Linux（予定）、Slackware（予定）、LASER5 Linux（予定）、開発版および関連パッケージは、Debian GNU/Linuxの「Potato」、Vine Seedなどとなっており、さらに、DebianのFTPサイトをミラーリングしたハードディスクを、8000～1万円＋実費程度で販売する予定もあるという。

価格は、ディストリビューション安定版が1枚あたり350円、ディストリビューション開発版、Linux関連パッケージが1枚あたり400～600円。たとえば、ディストリビューション安定版にラインナップされている「Debian GNU/Linux 2.1 Slink r2 for i386」は、4枚組のため、1400円になる。送料は別途必要。

注文方法は、Webページによるオンライン申し込みで、午後3時までの発注であれば、その日のうちに発送される。

Linux4u.net
(http://www.linux4u.net/)

レッドハット、SCSIドライバ開発

1999年12月21日

レッドハットは20日、米Adaptec製SCSIコントローラ「AIC-7899」のLinux用SCSIドライバを開発したと発表した。これは、同社とNECとの技術提携によるもので、NECのサーバ「Express5800シリーズ」で動作を確認しているという。同ドライバは、レッドハットのWebサイトで12月25日から提供される予定。

AIC-7899は、Ultra160/m対応のSCSIコントローラ。Ultar160/mは、Ultra2との下位互換性があり、転送速度は毎秒160Mバイト（Ultra2の2倍）。

レッドハット
(http://www.redhat.com/jp/)

Linux用のデバッグツール「Linux Trace Toolkit（LTT）」リリース

1999年12月19日

Linux用のデバッグツール「Linux Trace Toolkit（LTT）」のバージョン0.9.0が発表された。

LTTは、ソースプログラムを1ラインまたは1命令ごとに実行し、プログラムの問題個所を把握できるGUIベースのトレースツール。時系列でカーネルの状況などをグラフィカルに表示できる機能に加え、フォーカスしたい部分を拡大表示するズームイン機能も備えているのが特徴となっている。

LTTのニュースリリース（英文）
(http://www.info.polymtl.ca/home/karym/www/trace/)

「FrameMaker for Linux」ベータ版の無料ダウンロード開始

1999年12月19日

「Acrobat Reader」のLinux版（日本語PDFを表示可能）を無償配布している米Adobe Systemsは、「Acrobat Distiller Server」が開発中であることと、「FrameMaker for Linux」ベータ版がダウンロード可能になったと発表した。

Acrobat Distiller Serverは、大量のPostScriptファイルをPDFファイルに自動変換するためのサーバ用ソフトウェア。

現在開発中ではあるが、最初のターゲットプラットフォームはLinuxで、その後Solaris、Windows NTがサポートされる予定。出荷開始は2000年第1四半期を予定している。

FrameMakerは、同社の看板製品の1つで、PDF、PS、HTMLなどを扱えるDTPソフトウェア。FrameMaker for Linuxベータ版では日本語はサポートされていない。「FrameMaker for Linux」動作環境は、以下の通り。

- Pentium以上のプロセッサ
- メモリ32Mバイト以上（80～150Mバイト以上を推奨）
- HDD空き容量100Mバイト以上
- glibc 2.1以降
- portmap/rpcbind（RFC 1833）サービスがインストールされていること
- XFree86 version 3.3 以降

Adobe Systems（http://www.adobe.com/）

Distribution ▶▶▶

Red Hat Linux 6.1改訂 日本語版、2月10日リリース

レッドハットは、「Red Hat Linux 6.1改訂　日本語版」を、2月10日にリリースすると発表した。昨年の11月12日に発売されたRed Hat Linux 6.1日本語版（以下6.1J）の改訂版である。価格は、6.1Jと同じく1万2800円。

6.1J発売後に指摘されていた、日本語環境への対応不足を改善している。具体的には、システム管理ツール「linuxconf」の日本語化、kedit、kmailの日本語への対応、日本語の共有名をサポートしたSambaの日本語版（2.0.5aJP2）の採用などが行われている。またLinux Japanese Locale Working Groupの指針に基づき、日本語ロケール名としてja_JP.eucJPを追加している。

6.1J発売後に発見されたセキュリティホールへの対応やバグフィックスが行われているほか、「Vine Linux」で開発された印刷機能を取り込んで、数多くの国産プリンタを利用できるようになったとしている。

レッドハットによれば、現在店頭にある6.1Jは、2月10日から全国一斉に改訂版に置き換えられるとしている。また、同日から同社のFTPサイトよりダウンロード可能。

レッドハット（http://www.redhat.com/jp/）

Debianの新リリース「potato」がフリーズ

Debian Projectは1月16日に、Debian GNU/Linuxのリリース2.2に相当する「potato」のフリーズを宣言した。フリーズとは、バグフィックス以外のパッケージ更新を原則禁止して、正式リリースに向けて完成度を高める段階に入ったことを意味する。Debian Projectは、フリーズの期間を約2カ月としている。また「potato」のフリーズにともない、新たな開発版のリリースとして、「woody」が作成された。「potato」のフリーズ期間中は、安定版、フリーズ版、開発版の3種類のディストリビューションが存在することになる。

同時に国内のDebian JP Projcetも「potato-jp」のフリーズと開発版リリースの「woody-jp」の作成を発表した。しかし「potato-jp」は、リリースされないことになっている。これはDebian JP Projectの目的が、ローカライズ版や日本語対応用追加パッケージの作成ではなく、Debian本家の国際化であるためだ。実際、Debian JP Projectで作成されたパッケージのDebian本家へのマージが行われており、potatoではDebian Projectが配布しているパッケージのみで、十分な日本語環境が揃うようになっている。なお、potato-jpのパッケージが必要であれば、Debian JPのFTPサイトからインストールすることが可能だ。

Debian Projcet（http://www.debian.org/）

マイスターLinux Mandrake 6.1発売

五橋研究所は、日本語版のLinuxディストリビューション「マイスター Linux Mandrake 6.1」を1月28日に発売した。これは、仏MandrakeSoft社のLinuxディストリビューションを日本語化した「Linux Mandrake-JP 6.1」をベースにしたものだ。

Linux Mandrake 6.1は、いわゆる「ベターRed Hat」系のディストリビューションで、欧米では人気が高い。カーネルは2.2.13、Cライブラリはglibc 2.1.1、XFree86は3.3.5を採用している。

パッケージにはCD-ROMが3枚（バイナリ、ソース、マイスター）と、日本語のインストールマニュアルが含まれている。マイスターディスクには、MandrakeSoft社のアップデートパッケージ、XFree86の最新版、日本語化パッケージ、商用のDynaFont 5書体などが含まれている。これらをインストールすることで、日本語環境や最新のカーネル（2.2.14）、XFree86（3.3.6）などが利用できる。インストール作業は付属のスクリプトを用いて、簡単に行える。価格は3480円で、サポートは行われない。

五橋研究所（http://www.cdrom.co.jp/）
MandrakeSoft（http://www.linux-mandrake.com/）

Red Hat Linux 6.1 Deluxe for Alpha Systems出荷開始

米Red Hat社は、コンパック（旧DEC）のAlphaプロセッサに対応した「Red Hat Linux 6.1 Deluxe for Alpha Systems」（英語版）（以下Red Hat for Alpha）を1月12日より出荷開始した。国内でもレッドハットのWebページで、日本語版開発開始のアナウンスがされている。

Red Hat for Alphaは、グラフィカルなインストーラ、X Window System上のKDE／GNOMEが利用可能など、PC版のRed Hat Linux 6.1と同様の特徴を持っている。

価格は79.95USドルで、180日間のFTPサイトへの優先アクセス権、90日間のWebベースのインストールサポート、リファレンスガイドやスターティングガイドの提供、30日間の電話によるインストールサポート、Workstation Bonus PackやPowerTools applicationsが含まれる。

Red Hat, Inc.（http://www.redhat.com/）
Compaq Computer（http://www.digital.com/）

Products

ISDNルータと組み合わせて使うWebキャッシュサーバ

COOLGREEN GigaCache Server

ISDNルータを使えば複数のマシンから同時にインターネットを利用できるが、その場合、速度の低下は免れない。本機は、WebやFTPで読み込んだデータをキャッシングし、何度も同じデータを転送しないようにすることで、データへのアクセス速度を向上させる低価格キャッシュサーバだ。

製品名　COOLGREEN GigaCache Server
価格　6万4800円
問い合わせ先　クールグリーンコンピュータ有限会社
TEL 045-640-3208
http://www.coolgreen.co.jp/

ISDNルータの普及とともに、家庭やSOHOでもISDNによるインターネット利用が盛んになっている。ISDNルータのIPマスカレード機能を使えば、LANで接続した複数のマシンで同時にインターネットを利用できるが、回線を共有する以上、マシン1台あたりの転送速度は低下する。

このような問題を緩和する方法として、WebやFTPのデータをディスクなどに記録しておき、過去に取得したデータは再び転送しないようにする“キャッシング”がよく使われている。

本製品は、キャッシングに特化したLinuxベースのオールインワン・サーバパッケージだ。ISDNルータとともに使うことを前提としており、既存の環境に追加して手軽に利用できる。

低価格で実用性重視のキャッシュ専用サーバ

プレインストールされているソフトウェアは、OSにVine Linux 1.1を採用し、キャッシュソフトウェアとしてSquid 2.2 STABLE5が組み込まれている。Squidは、速度を重視したキャッシュプロキシとして知られており、データが蓄積されるにつれて効果を発揮する。

一方、ハードウェアをみると、CPUは、IDTのWinChip C6 200MHz、ハードディスクは4.3Gバイトという構成になっている（**表1**）。最近のデスクトップマシンと比べるといかにも非力なスペックに見えるかもしれないが、16Mバイトのメモリと1.5Gバイトのディスクスペースをキャッシュデータ用に予約しており、たかだか128kbpsのISDNで使うキャッシュサーバとしては十分余裕のある構成だ。不必要な高スペックを追求しないで、

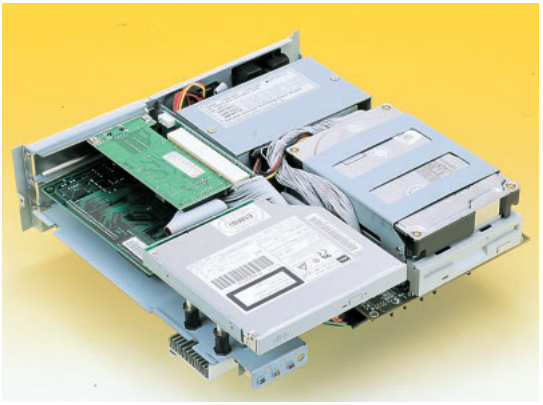


写真1　本体内部の表面
コンパクトな筐体に収めるため、CD-ROMドライブは薄型の24倍速を採用している。ハードディスクは4.3GバイトのEIDEドライブだ。左側に見える緑色の基板はイーサネットカード。

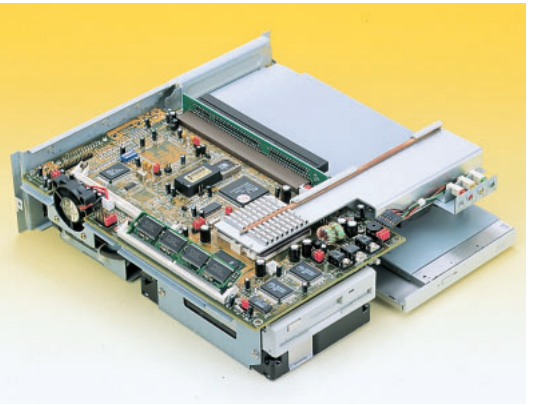


写真2　本体内部の裏面
CPUは、IDT WinChip C6を採用しており、ヒートパイプを利用したユニークな冷却方法をとっている。メモリは72PINのSIMMで、2スロットあるうちの1つは空いている。

価格を抑えている点には好感が持てる。また、低消費電力、低発熱のCPUを選択したことで、ブックシェルフ型のコンパクトな筐体でも放熱不足になることはないだろう。

接続やIPマスカレードの機能はISDNルータにまかせ、キャッシュのみの単機能サーバとすることで信頼性も高くなる。telnetやFTPなどのサーバソフトウェアはインストールされておらず、セキュリティも確保されている。編集部で調べたところ、開いているポートはSquidのものだけだった。このポートに対しては、IPアドレスなどによる接続制限はかけられていないが、ISDNルータのIPマスカレード機能を利用していれば外部から接続されることはないだろう。ただし、初期状態では、rootのパスワードが設定されていないので、誰でもコンソールからrootログインできてしまう。利用する前にパスワードを設定したほうがいいだろう。

写真3　本機の背面パネル
通常のPCと同様のコネクタが並ぶが、通常はシリアルポートやパラレルポートを使うことはないだろう。キーボートは標準DINコネクタで接続する。

IPアドレスを設定すればすぐに利用可能

IPアドレスなどのネットワーク設定は、Yamaha RTA50iなどに合わせてあるが、利用する環境と違っている場合は、コンソールからrootでログインし、Linuxconfコマンドを使用して設定する。設定手順は、付属の「ネットワーク設定ガイド」に簡潔にまとめられている。あとはブラウザでプロキシサーバに本機のIPアドレスを指定するだけで利用できるようになる。ひとたび動くようになれば、CRTを接続しておく必要もない。

キャッシュという仕組みは、初めて転送するデータに対しては効果を持たないので、運用当初は速度の向上は体感できないが、利用しているうちにデータが蓄積され、威力を発揮するようになる。多くの利用者がよく訪れるサイトに対しては特に有効だ。

小規模事業所や家庭での導入をお勧めする

本機は、Linuxやネットワークの知識はほとんどなくても簡単に設定、運用ができるため、専門のネットワーク管理者を置くことのできない小規模事業所や、家族数名でインターネットを利用している家庭にお勧めする。

また、Linuxに精通していれば、カーネルを2.2系に入れ替えるなどして、パフォーマンスアップをねらうのもいいだろう。ただし、この場合はメーカーからのサポートは受けられないので注意が必要だ。

CPU	IDT WinChip C6 200MHz
メインメモリ	32Mバイト（最大64Mバイト）
フロッピーディスクドライブ	3.5インチ（1.44Mバイト／720Kバイト）
ハードディスクドライブ	3.5インチ　4.3Gバイト
CD-ROMドライブ	最大24倍速　薄型
イーサネットカード	Realtek　RTL-8139A PCI（100BASE-TX／10BASE-T）
キーボード	BTC　101配列準拠80キーミニキーボード
消費電力	35W（アイドル時　約23W）
外形寸法（mm）	78（W）×288（D）×279（H）（本体のみ、突起部除く）
重量	約5.0kg
OS	Vine Linux 1.1
キャッシュソフトウェア	Squit 2.2 STABLE5
主な付属品	キーボード、スタンド（一組）、システム修復用ディスク（FD×1、CD×1）、ユーザーズガイド（3部）、電源ケーブル

表1　COOLGREEN GigaCache Serverの主な仕様

安価なIDEハードディスクを使えるホットスワップ対応のミラーリングRAIDユニット

IntelliMirror IM-7500

ハードディスクが大容量化するにつれて、故障時のダメージも高まりつつある。IntelliMirror IM-7500を利用すれば、SCSIより安価なIDEでハードディスクを二重化し、故障によるデータの消失を防ぐことができる。

製品名	IntelliMirror IM-7500
価格	オープンプライス（市場予想価格：7万5000円前後）
問い合わせ先	株式会社ユニット TEL 03-3460-7051 http://www.myshop.co.jp/unita/

IntelliMirror IM-7500は、2台のIDEハードディスクでRAID1つまりミラーリングをすることでディスクサブシステムの耐障害性を高めるRAIDユニットである。販売元のユニットは、各種の周辺機器を開発・販売しており、本機もそうした周辺機器の1つとしてラインアップされている。

装着可能なハードディスクの条件

本機の構成は、大雑把にいえば2台のハードディスクが収納できる5.25インチ幅フルハイトのユニットと、**写真1**のフロントパネル、**写真2**のドライブキャリア、**写真3**のRAIDコントローラ回路からなる。ハードディスク自体は同梱されておらず、別途購入する必要がある。装着できるのは3.5インチ幅1インチハイトのIDEハードディスク2台だ。ただし、電源コネクタとIDEコネクタの間隔が15mmでないと、ドライブキャリア側のコネクタ位置が合わず、装着できない。実際、手持ちのWestern Digital製Expert BA（WD205BA）は寸法が合わず、装着できなかった（マニュアルにも同社製IDEハードディスクは使用できないと記されている）。また2台のハードディスクは同容量でなければならない。なるべく同一メーカー・同一型番にすべきだろう。以上の点に注意していれば、最近のUltraDMA/33あるいは66対応のハードディスクなら本機で問題なく利用できるはずだ。

専用ドライバは不要

本機に内蔵された2台のハードディスクが常に同じデータを保持するように、PCから送られてきたデータは双方のハードディスクに書き込まれる。また、どちらかのハードディスクが交換されたら、自動的にディスク間でコピーが行われ、内容を一致させる。こうした作業は、PCから隠蔽された状態で内蔵RAIDコントローラ（**写真3**）が自動的に行っている。

PCから見ると、本機は普通のIDEハードディスクにしか見えず、OS用デバイスドライバも一般的なIDE用ドライバが利用できる。つまりRAIDコントローラにつきものの専用ドライバが不要なので、Linuxでも問題なく利用できるというメリットがある。

PCと接続するIDEインターフェイスはUltraDMA/33に対応しており、最大33Mバイト/秒で転送可能だ。マスタ／スレーブもジャンパで設定できるし、他のIDEデバイスと同一ケーブルで接続可能だ。

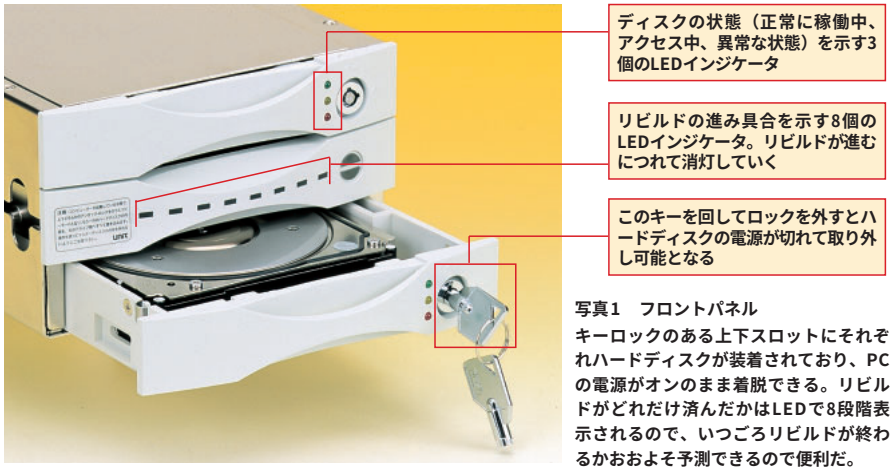


写真1　フロントパネル
キーロックのある上下スロットにそれぞれハードディスクが装着されており、PCの電源がオンのまま着脱できる。リビルドがどれだけ済んだかはLEDで8段階表示されるので、いつごろリビルドが終わるかおおよそ予測できるので便利だ。

ホットスワップやバックアップも可能

装着したハードディスクのうち1台が故障してアクセス不能になると、本機はブザー音とLEDインジケータでユーザーに警告する。そこで**写真1**の鍵でロックを解除すると、ブザー音が鳴りやんでハードディスクへの電源供給が止まり、ドライブキャリ

写真2　ドライブキャリア
ハードディスクはこのドライブキャリアに装着・ネジ止めして、本体に取り付ける。ドライブキャリアは2個付属する。キャリア側のIDEコネクタは完全に固定されているため、ハードディスク側のIDEコネクタ位置によっては物理的に寸法が合わず、装着できない場合がある。

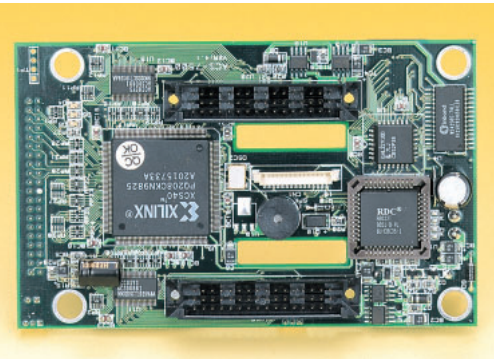


写真3　RAIDコントローラ回路基板
ミラーリングやリビルドなどの制御を行うメイン回路が搭載されている。上下2つの黒いコネクタには、ドライブキャリアが接続される。その近くに空いている穴は、この基板の背後に位置する空冷ファンが内部の空気を吸い出して冷却するために設けられたようだ。

アの物理的なロックも外れてキャリアを引き出せる。その後、ハードディスクを交換してキャリアごと本体に装着し、鍵でロックすると、無事だったハードディスクから新しいハードディスクへデータのコピーが始まる（これをリビルドという）。ハードディスク交換作業は本機だけで独自に処理され、PC側には影響を与えないのでPCの電源をオフにすることもソフトウェア的な対処もまったく必要ない。リビルドの最中はハードディスクへのアクセスが遅くなるが、それ以外についてはふだんと変わりなくユーザーは作業できる。これが本機最大のメリットといえよう。

また本機が優れているのは、リビルドの進み具合が分かりやすいことだ。リビルド処理が1/8ずつ終わるにつれてLEDが1つずつ消灯していくようになっている。さらに定常状態では、LEDインジケータ上を光が左右にゆっくり移動しつづけるので、RAIDコントローラが正常に動作しているか一目でわかる。

本機に装着しているハードディスクを取り出して、本機を介さずに直接IDEでPCと接続しても、そのまま利用できる。これは装着されたハードディスクのCHSパラメータを本機がそのままPCに知らせるため、ジオメトリ情報が一致するからだ。そのため、ハードディスクを1台余計に用意して定期的に交換すれば、ブート可能なバックアップメディアを作成・保存できる。万一RAIDコントローラが壊れても、バックアップしたハードディスクでそのまま代替できるのだ。20Gバイト以上の大容量ハードディスクが対象なら、テープドライブでバックアップするより、はるかに手軽でコストも低い。ただし1ドライブあたり1世代しかバックアップできない、空き領域まで含めてフルバックアップしかできず、時間がかかる、というデメリットもあるが。

小規模なファイルサーバなどに好適

本機の性能は、装着するハードディスクに依存する。試しにIBM製ハードディスク（DPTA-371360、13Gバイト、7200RPM）を装着し、Red Hat Linux 6.1日本語版でBonnieというベンチマークプログラムを実行してみた。ハードディスク単体の場合と比べ、本機はシーケンシャルアクセスで10%～15%ほど遅かったが、実用上その差はほとんど体感できない。

数あるRAIDユニットのなかでも本機は非常に安価な製品だ。それゆえ高価な製品に比べると、故障時にネットワーク経由でユーザーに通知できない、ミラーリングしかサポートしておらず、速度を優先したストライピング（RAID0）や、容量を増加できるRAID5が利用できない、といった点はやむを得ないだろう。こうしたデメリットを考慮すると、本機は小規模なネットワーク内のファイルサーバやワークステーションにて、IDEハードディスクと交換して手軽に耐障害性を高めたい、という用途に向いている。もっと価格が下がれば、クライアントPC全般にもお薦めできそうなのだが。

PCとのインターフェイス	IDE（UltraDMA/33）
内蔵ハードディスクとのインターフェイス	IDE（UltraDMA/33）、ただしUltraDMA/66対応IDEハードディスクも利用可能
機能	RAID1（ミラーリング）、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（内蔵CD-ROMドライブ2台分のスペース）
冷却について	空冷ファン×1個（本体背面のRAIDコントローラ回路基板の裏側に配置）
付属品	ドライブキャリア×2個、マニュアル（日本語あり）、ドライブキャリアをロックする鍵、取り付けネジ
外形寸法（mm）	146.2（W）×224.6（D）×85.1（H）
重量	1.8kg（内蔵ハードディスクを含まず）
保証期間	1年（修理・交換）

表1　IntelliMirror IM-7500の主な仕様

携帯電話／PHSを利用して携帯機器から会社のメールアカウントがそのまま使える

MM QUBE

10円メールのサーバ機能と同等なモバイルメールサーバ機能を搭載したため、ポケットボードやノートPCで手軽に速く安全に、そして安くメールの送受信が行えるというコンパクトなオールインワンサーバが発売された。

製品名	MM QUBE
価格	29万8000円
問い合わせ先	NTT移動通信網株式会社 TEL 0120-774-360 http://www.nttdocomo.co.jp/

NTT移動通信網株式会社（以下ドコモ）から、携帯電話やPHSを用いた電子メールの送受信が可能となる、モバイルメール機能を搭載したオールインワンサーバ「MM QUBE」が発売された。

ベースはCobalt Qube 2

MM QUBEは、コバルト・ネットワークス社の「Cobalt Qube 2」をベースにしていて、外観は「MM QUBE」という文字や「NTT DoCoMo」のロゴが入っている以外はQube 2と同じである。写真を見るとわかると思うが、サイコロ型の立方体で各辺とも200mm以下とコンパクトにできている。

CPUには、x86系ではなくMIPS系の64ビットRISCプロセッサを使用している。メモリは32Mバイト、ハードディスクは8.4Gバイトとなっていて増設はできない。Ethernetは10/100BASE-Tを2ポート搭載し、ファイアウォール機能を持ったルータとして使用することができる。

シリアルインターフェイスは、本体の1ポートに加え、拡張カードで2ポート増設している。このPCIスロットに挿したシリアルインターフェイスが、Cobalt Qube 2の標準とは違う点で、後述するがPDCデータアクセスユニット（以下PAU）やターミナルアダプタ（以下TA）を接続するためのものである（**写真1**）。

MM QUBEのOSはLinuxであり、POP3／IMAP4に対応するメールサーバ、Webサーバ、FTPサーバ、ファイルサーバ、ファイアウォール、DNSサーバ、DHCPサーバ、メーリングリストなどの機能を持ったオールインワンサーバとなっている。

10円メールとは？

MM QUBEには、マスターネットが行っている「10円メール」のサーバと同等の機能をもったVALUE-MAILを搭載している。すなわちVALUE-MAILを使えば、10円メール対応PDAか、専用ソフトウェアをインストールしたノートPCから、プロバイダを通さずにMM QUBEに対して直接メールを送ることができるのだ。「10円メール」は、約2Kバイトまでのメールを1回あたり10円で送受信できるサービスである。

VALUE-MAILとは、携帯電話やPHSと、ノートPC、PDA端末を使用してメールの送受信を行うためのソ

写真1　MM QUBE背面
ベースはCobalt Qube2だが、PCIスロットにシリアルインターフェイスカードを増設している。キーボードやマウスのコネクタは存在しない。すべてはLAN経由で制御する。

写真2　ポケットボードピュア
非常に軽量コンパクト（重さ199g）で、液晶は25文字（全角）×7行表示が可能。携帯電話につないでメールの読み書きができる。

写真3　PAUとルーフトップアンテナ
PAUにアクセスポイントとして携帯電話を内蔵する。アンテナは電波状況の良いところに設置する。

写真4　PAUに組み込む内蔵用携帯電話
携帯端末からこの携帯電話にアクセスすることで、MM QUBEとメールのやりとりができる。

写真5　MN128 SOHO SL11
64K PIAFSに対応したターミナルアダプタを使用することで、PHSを利用したモバイルメールが可能になる。

フトウェアで、端末がアクセスを開始してから接続するまでの時間が短く、そのため通信費が安く、セキュリティ面でも安全であるといった特徴がある。

ポケットボードがそのまま使える

VALUE-MAILは、数千円で買える携帯電話専用メール端末であるポケットボードが利用できる。**写真2**のポケットボードピュアは、大きさが166.2（W）×86（D）×23.8（H）mmで、約200gと軽量コンパクトで持ち運ぶのにほとんどじゃまにならないだろう。

もちろん、Windows 9xを搭載したノートPCも使用できるが、VALUE-MAIL専用クライアントが必要だ。それ以外にも、Windows CEや、シャープのザウルスに対応したVALUE-MAIL対応メールソフトも用意されている。それぞれの専用メールクライアントはドコモのWebページからダウンロードすることができる。

PHSを使って通信すれば、携帯電話よりも高速に通信できる。携帯電話のデータ転送速度が9600bpsなのに対して、PHSでは、32K PIAFSで約3倍、64K PIAFSなら約6倍なので、添付ファイルなどで大きなデータを転送するのならPHSを使うほうがよいだろう。

システム構成

図1が全体のシステム構成である。携帯電話からのデータ通信を受けるために、PDCデータアクセスユニッ

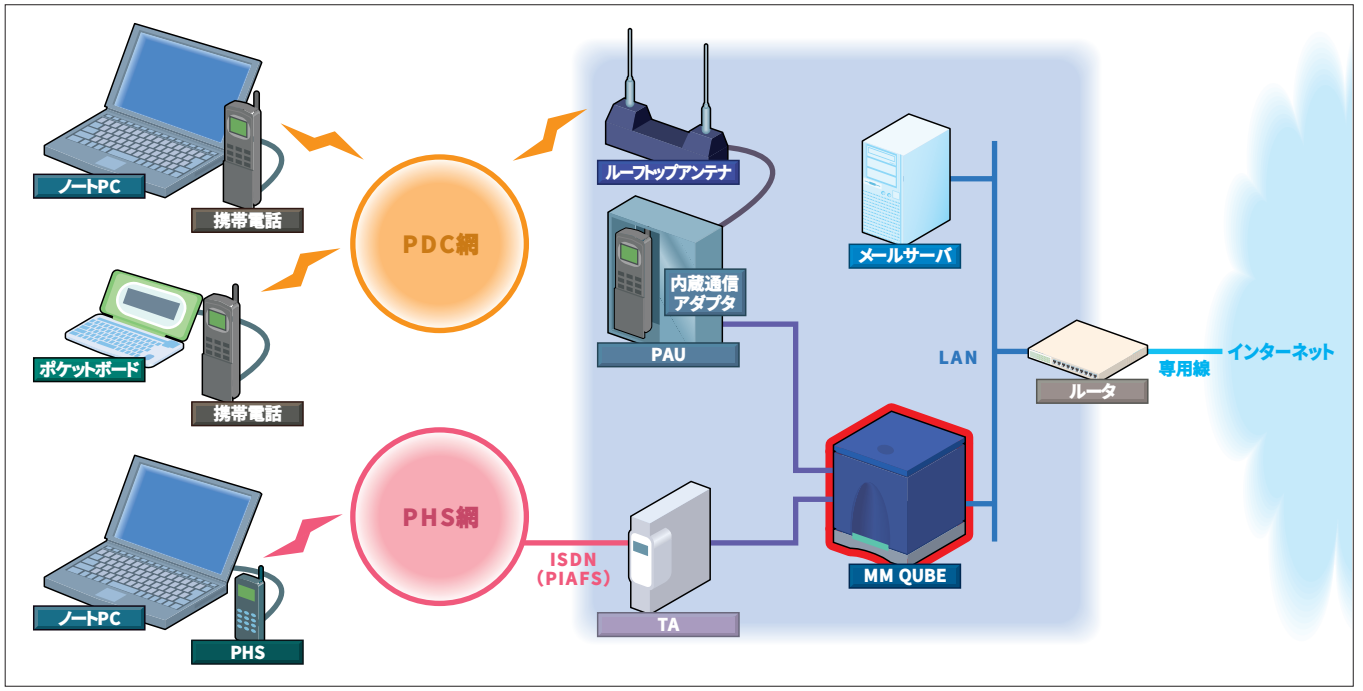


図1　全体のシステム構成

ト（PAU）を使用する（**写真3**）。MM QUBEとはシリアルインターフェイスで接続する。PAUの内部に受信用の携帯電話を内蔵し、電波の状況の良い場所に外部アンテナとしてルーフトップアンテナを設置する。なお、PAUには内蔵する携帯電話が含まれていないため別途購入する必要がある。PAUに対応する携帯電話はF206Hyper（デジタル）かF156Hyper（シティフォン）の2機種となっている。

PHSとデータ通信を行うにはISDN回線を通してTAと接続する。MN128SOHO SL11（**写真5**）は64K PIAFSに対応するTAで、MM QUBEとはシリアルインターフェイスで接続する。

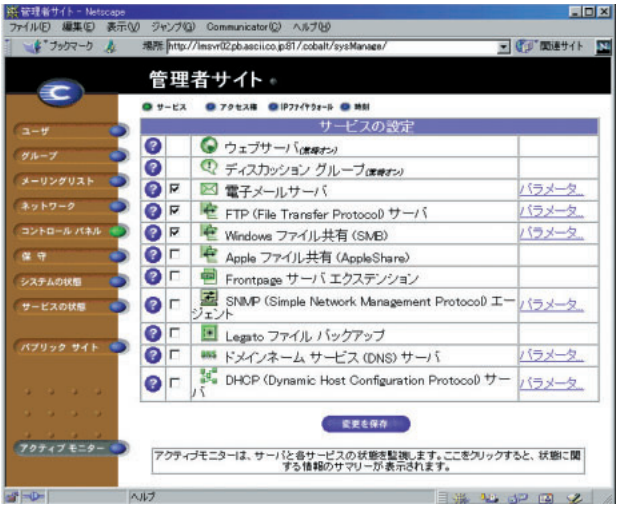
MM QUBEは単体でメールサーバにもなることができるが、すでにメールサーバが稼働している会社などでは、そのアカウントでメールをやりとりしたいところだ。そういう場合には、MM QUBEをメールの中継サーバとして使うように設定すればよい。

VALUE-MAILの仕組み

プロバイダへPPP接続した場合には、最初にモデムのネゴシエーションがあり、次にIDとパスワードの認証が行われ、それからユーザーのメールをチェックし、メールを受信するという手順を踏む。接続開始から実際にメールの受信が始まるまでには、数十秒かかることもざらである。

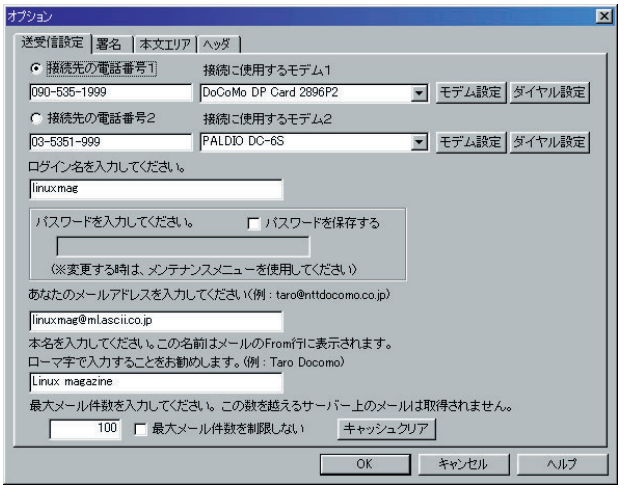
10円メールやVALUE-MAILでは、呼び出しが行われた時点で、携帯電話の発信者番号通知を利用してユーザーIDを検出し、接続する前にメールのチェックやユーザー認証のための準備をサーバ側で行っている。そのため接続してからメールの送受信までの時間を大幅に短縮できるのである。10秒以内に準備が終了し、データを送る時間が2秒だとして、9600bpsで約2Kバイトを転送できる。

ドコモの携帯電話同士で10円でかけられる秒数が、10円メールでは12秒だった時にサービスを開始したので

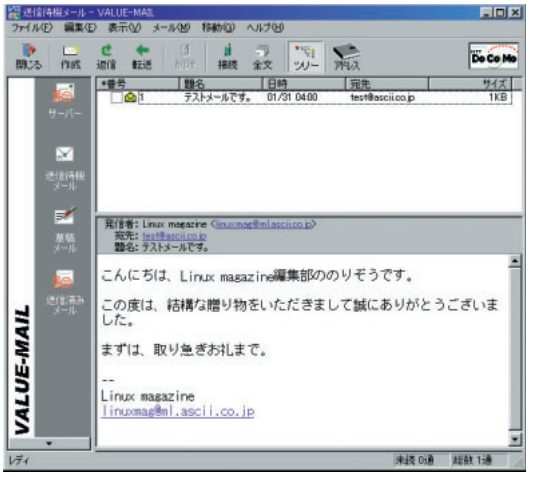


画面1　MM QUBEの管理者サイトでサービス内容を設定する。

画面2　VALUE-MAILの管理者サイトでユーザー情報を登録する。



画面3　専用メールクライアントVMWINの送受信設定で電話番号を設定する。



画面4　VMWINでメールの送受信を行う。

2Kバイトまで10円となっているが、現在では16.5秒（プランA、平日昼間）なので、VALUE-MAILではあと4.5秒分多く転送することが可能で10円で約6.5Kバイトのメールを転送できる。

もっとも、時間帯や発信者の加入している料金プランでも違うので一概には言えないが、1.5GHz帯のシティフォンでデータレートサービスを利用すれば1分15円となるので、1回のデータ転送量が多いようなら、もっと安く通信できるだろう。

VALUE-MAILの設定

MM QUBEの各種設定は、LANで接続されたほかのマシンからWebブラウザで行う。VALUE-MAILの設定の前にCobalt Qube 2の管理者サイトで、ネットワークや提供するサービスの設定を行う（**画面1**）。

次にVALUE-MAILの設定だが、トップメニューにVALUE-MAILのメニューが加わり全体として整合性がとれている。VM管理者サイトで、ユーザー情報の登録を行う（**画面2**）。中程にある登録電話番号に、発信者が使う携帯電話またはPHSの番号を記録しておく。この番号が着信時の認証に使われる。

MM QUBEのVALUE-MAILに登録できるユーザーは50名までとなっている。

専用メールソフトVMWIN

VALUE-MAIL専用端末ソフトウェアは、ドコモのホームページからダウンロードして使用する。Windows 9x用のVMWINの初期設定は**画面3**のようになっている。接続先の電話番号は、PAUに内蔵の携帯電話の番号か、TAがつながっているISDNの電話番号を指定する。一番よく使う画面はメールを送受信するための**画面4**である。

VALUE-MAIL、VMWINは、POP3はもちろんIMAP4にも対応しているので、メールのヘッダだけを取得して、必要なメールだけを読むことができる。これならば添付ファイルを含むような大きなサイズのメールを取り込まなくてもよい。また、逆に必要ならば添付ファイルを取り込むこともできる。

ふだん使い慣れたメールソフトではなく、VMWINを使わないとVALUE-MAILサーバとメールのやりとりができないというのが不便なところではある。

また、Webブラウザでホームページを見たいという要求があっても、メール以外はサポートされていないため、MM QUBEをRAS（リモートアクセスサーバ）として利用することもできないのは残念なところだ。

CPU	64ビットスーパースカラーRISCプロセッサ
メモリ	32Mバイト
ハードディスク	8.4Gバイト
LANインターフェイス	10/100Base-T ×2
シリアルインターフェイス	1ポート＋拡張2ポート
外形寸法（mm）	184（W）×184（D）×197（H）
重量	約3.1kg
最大消費電力	25W
OS	Linux 2.0（MIPS版）
ソフトウェア	モバイルメールサーバ：VALUE-MAIL（50クライアント以下） Webサーバ：Apache 1.3.3 メールサーバ：SMTP、POP3、IMAP4 ファイルサーバ：FTP、SMB、AppleShare、AppleShareIP対応 パケットフィルタリング・ファイアウォール、NAT DNSサーバ、DHCPサーバ

表1　MM QUBEの主な仕様

MM QUBEモニター募集10台！

NTTドコモはMM QUBEを2カ月間試用できるモニター10社を募集しています。下記の応募要領をご覧のうえ、お申し込みください。

MM QUBEモニター応募要領

●モニター期間
　製品受領から2カ月間。
●応募資格
　MM QUBEを利用したモバイルメールの試用利用が可能な企業（法人）。
●レポート
　使用1カ月後に利用状況のインタビューと2カ月後に、使用感や製品に対する要望などを記入シートに記述し提出していただきます。
●応募方法
　下記URLの応募専用Webページで必要事項を入力して下さい。
　http://www.mcsys.nttdocomo.co.jp/mmqube/
●応募締め切り日
　2000年2月29日。
●選考発表
　申し込みWebページで応募締め切り後、発表いたします。
●応募のご注意
　MM QUBEに接続するモバイルアクセスに必要な以下の通信機器は、モニター企業で用意できることが条件になります。

・アクセスポイント用携帯電話(1回線)
・アクセスポイント用ISDN回線とTA

　また、モバイルアクセスで必要となる通信費用はお客様負担になります。

Distribution

新着ディストリビューション

LASER5 Linux 6.0 Rel.2

日本語ディストリビューションとして人気の高いLASER5 Linux 6.0が、グラフィックス周りを中心とした各種のコンポーネントに最新版を採用し、バージョンアップされた。新バージョンのRed Hat Linux 6.1ではなく、あえて6.0をベースに改良したという「LASER5 Linux 6.0 Rel.2」を紹介する。

Storm Linux 2000

「嵐を呼ぶLinux」がいよいよ発売された。玄人好みのDebianをベースに、GUIインストーラ、オリジナルの管理ツールやパッケージマネージャを組み込むことで、堅牢性と使い勝手の両立を目指すカナダ産の新ディストリビューション「Storm Linux 2000」を紹介する。

LASER5 Linux 6.0 Rel.2

レーザーファイブから、LASER5 Linux 6.0 Rel.2が1月28日に発売された。昨年9月に発売された「LASER5 Linux 6.0」から、バグフィックス、セキュリティホールへの対応、扱えるハードウェアの範囲の拡大などが行われ、各種コンポーネントを最新バージョンに改訂したものである。

LASER5 Linux 6.0 Rel.2［デラックス］（以下デラックス）の価格は1万2800円で、90日間3件までのサポートが含まれている。インストールからX Window Systemが立ち上がるまでを、電話、FAX、Web（電子メール）でサポートしてくれる。

また、サポートと一部のソフトウェアを省いた廉価版として、LASER5 Linux 6.0 Rel.2日本語入力キット［スタンダード］（以下スタンダード）も2月1日より6800円で発売された。

デラックス、スタンダードともに、商用ソフトウェアとして、かな漢字変換のATOK12 SEとWnn6 Ver.3、商用TrueTypeフォントのDynaFont 5書体、日英辞書引きソフト（eWnn）、WebブラウザでLinuxを遠隔管理できるLinux Controller、Secure Shell（SSH1/2）が付属している。

加えてデラックスには、WindowsとLinuxのデュアルブートを簡単に行えるSystem Commander Lite、最新RDBMS（EMPRESS、ObjectivityDB、Sybase）、バックアップツールのBRUのほか、多くの製品版、体験版ソフトウェアが付属する。

なお、従来のLASER5 Linux 6.0（製品版／日本語入力キット）の登録ユーザーは、直接レーザーファイブに申し込むことで優待価格にてバージョンアップすることができる。バージョンアップは3コースあって価格は1000円からとなっている。詳しくはレーザーファイブのWebページ（http://www.laser5.co.jp/）を参照していただきたい。

あえてベースはそのままのRed Hat Linux 6.0を採用

LASER5 Linux 6.0 Rel.2は、英語版のRed Hat Linux 6.0をベースに日本語化したディストリビューションである。Red Hatからは6.1もリリースされているのだが、今回はあえて6.0ベースのままで、安定度を高めたとのことである。カーネルのバージョンは2.2.5、glibcは2.1.1と変更されていないが、セキュリティ対策のためのパッチは当てられている。

このRel.2では、Red Hat Linux 6.1に採用されている「kudzu」を導入している。kudzuは、システムの起動時にハードウェアの変更を自動検出するプログラムで、環境が変更されている場合には、変更内容を表示し、コンフィグレーションを行うことができる。

日本語化したデスクトップ環境には、GNOME 1.0.11とKDE 1.1.2を採用し、それぞれ最新のバージョンになった。

X Window Systemは、XFree86 3.3.5にバージョンアップされ新しいグラフィックスカードに対応した。

なお、Contrib CDには、Kernel 2.2.13、glibc 2.1.2、XFree86 3.3.6（X-TTパッチ済み）が納められているので、必要であればそれをインストールして使うことも可能だ。

画面1 Linux Controller
製品に付属するLinux Controllerを使って、Webブラウザからネットワークで接続したLinuxマシンを遠隔操作できる。

製品名　LASER5 Linux 6.0 Rel.2
価格　1万2800円
問い合わせ先　レーザーファイブ株式会社
03-5818-6626
http://www.laser5.co.jp/

本誌付録CD-ROM Disk 1に、LASER5 Linux 6.0 Rel.2フリー版を収録しています。インストールに際しては、243ページからの記事を参照してください。非商用のソフトウェアだけが収録されていますので、日本語かな漢字変換のATOK12 SEやWnn6、商用TrueTypeフォント、Linux Controller、System Commander Liteなどは含まれていません。

Storm Linux 2000

「Storm Linux 2000」（以下Storm Linux）は、カナダのStormix Technologies社が、'99年の12月に北米で販売を開始したディストリビューションである。Debian GNU/Linuxをベースに、GUIを用いたインストーラや、独自開発の管理システムであるStorm Administration System（SAS）、パッケージ管理システムのStorm Package Managerを採用している。

パッケージはスタンダード版の1種類のみで、49.95USドル。Star Office 5.1a（オフィススイート）、Applixware Office 4.4.2デモ版（オフィススイート）、Partition Magic SE（パーティション操作ユーティリティ）、BRUバックアップツール（デモ版）、VMware 1.1（デモ版）、Krilo（ゲームソフト）といった商用ソフトウェアが付属している。

Debianベースのディストリビューション

Storm Linuxは、Debian GNU/Linuxをベースに開発されている。カーネルはバージョン2.2.13を用いている。XFree86は3.3.5で、比較的新しいビデオカードにも対応している。またデスクトップ環境は、KDEが1.1.2、GNOMEはOctoberリリースと最新のバージョンが含まれている。デフォルトでは、KDEが起動するようになっている（**画面1**）。

GUIによるシンプルなインストール

最近発表される新しいディストリビューションは、GUIのインストーラを用意するのが当たり前のようになっているが、Storm LinuxもX Window SystemとGTK＋ライブラリを利用したグラフィカルなインストーラを採用している。またインストール時にはハードウェアの自動検出も行われる。一般的な構成のPCならば、ビデオカードの選択から適切なXサーバのインストールまで行ってくれるので、簡単にグラフィカルなデスクトップ環境を体験できる。

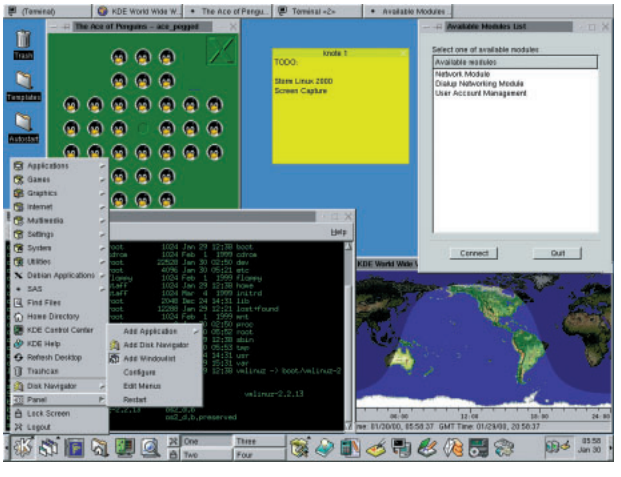
またブートローダはMBRに書き込まれ、ほかのOSが同じシステムにインストールしてある場合は、自動的にブートセレクタに加えてくれるようになっている（**画面2**）。

独自の管理システムSAS

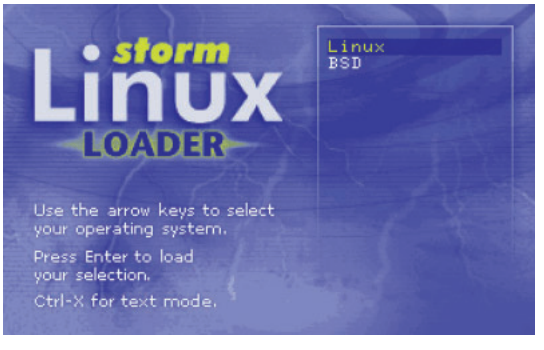
Storm Linux最大の特徴がStorm Administration System（SAS）である。SASはサーバ／クライアント方式の管理ツールで、モジュールを追加することにより、管理項目を増やせるようになっている。サーバとクライアントは、ネットワークで接続された別々のマシンに置くことも可能だ。

実際にユーザーが操作するのは、クライアントに相当するStorm Administration Tool（SAT）である。ツールバー上に用意されているアイコンをクリックすれば起動する（**画面3**）。パスワードを入力して「Connect」を押すと、使用可能なモジュールの一覧が表示される。現バージョンではネットワーク関係、ダイヤルアップ、ユーザー管理の3つが使用できる。各項目とも非常に分かりやすいインターフェイスを持ち、とまどうことなく設定が行える（**画面4、5**）。

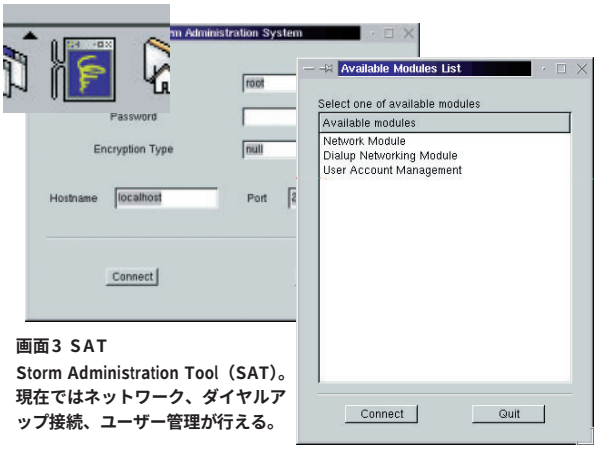
Stormix Technologies社では、このほかにもSambaやファイアウォールなどの設定が行えるSAS用モジュール



画面1 Storm Linux 2000
デフォルトのデスクトップ環境はKDEだ。もちろんインストール時に指定することで、GNOMEも利用できる。

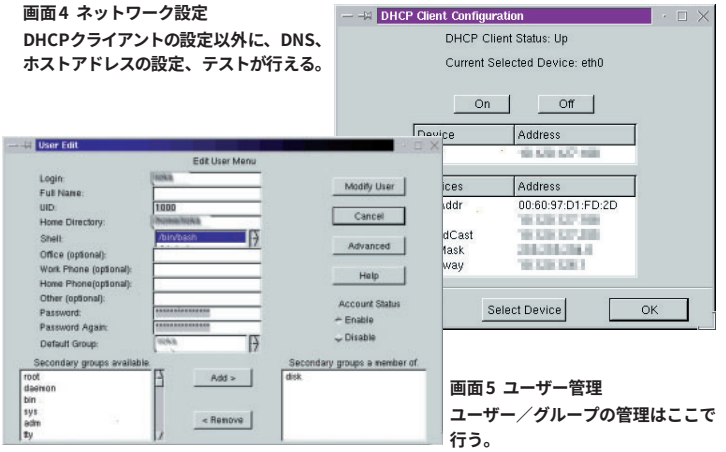


画面2 ブートセレクタ
Storm Linuxの起動画面。ほかのOSと共存させた環境では、このように起動可能なOSがリスト表示される。



画面3 SAT
Storm Administration Tool（SAT）。現在ではネットワーク、ダイヤルアップ接続、ユーザー管理が行える。

画面4 ネットワーク設定
DHCPクライアントの設定以外に、DNS、ホストアドレスの設定、テストが行える。



画面5 ユーザー管理
ユーザー／グループの管理はここで行う。

を開発する予定だという。またモジュール間の通信に用いられるプロトコルも公開されるそうなので、サードパーティやオープンソースコミュニティからモジュールがリリースされることが期待できる。

パッケージ管理システム

Debian GNU/Linuxベースのディストリビューションというと、アプリケーションのインストールがRed Hat系のRPMではなく、操作が複雑な「dselect」であることを思い出して、うんざりしてしまう人もいるかもしれない。だがStorm Linuxには、オリジナルのStorm Package Manager（コマンド名はstormpkg）が含まれている（**画面6**）。dpkgのフロントエンドであるという点では、dselectと同等の位置付けだ。しかし画面表示はわかりやすく、またマウスで操作できるのでキーボードを使い慣れていないユーザーでも安心して扱える。

日本語対応

今回発売されたStorm Linuxは北米版であり、インストールしただけでは日本語環境を使うことはできない。しかしStorm Linuxは、インストール後はDebian GNU/Linuxとほぼ同等であるため、Debian GNU/LinuxのCD-ROMやFTPサイトから必要なパッケージをインストールすれば、日本語環境を作ることができる。

Stormix Technologies社では、先日フリーズ宣言が行われたDebianリリース2.2に相当するpotatoをベースに、Storm Linuxの新バージョンを開発中だという。それをベースにした、日本語対応版のリリースも予定されているということだ。

ダウンロード版を収録

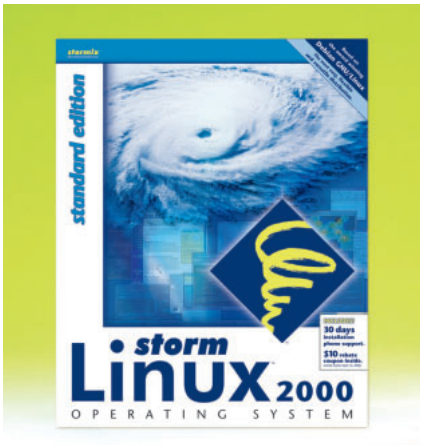
本誌付録CD-ROM Disk 2に、Storm Linux 2000のダウンロード版を収録した。また今回の収録に際し、Stormix Technologies社に、本誌読者からのフィードバック用メールアドレスを用意していただいた。

画面6 パッケージマネージャ
dselectの代わりに用いるStorm Package Manager。

Storm Linuxを使用したうえでの感想や意見、要望、再現性のある問題点などを見つけたら、下記のアドレスに送付してほしい。日本語でもかまわないそうだ。

linuxmag@stormix.com

なお、このメールアドレスは、あくまでもフィードバック用として用意されたものなので、インストール方法の質問や、サポート窓口の代わりに使うことは遠慮してもらいたい。

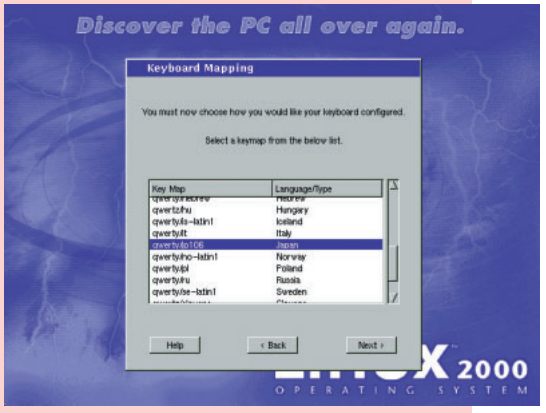


製品名	Storm Linux 2000
価格	49.95USドル
問い合わせ先	Stormix Technologies Inc. htto://www.stormix.com/

Storm Linux 2000のインストール

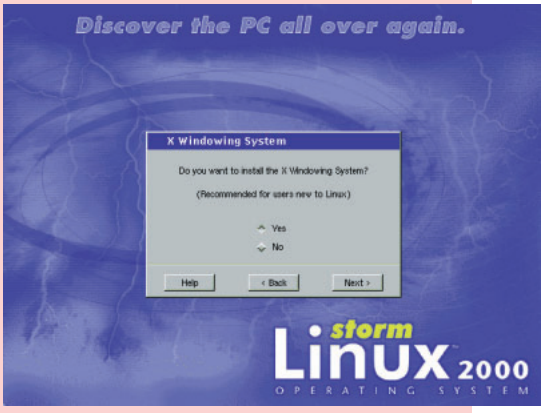
CD-ROMブートが可能なマシンでは、Storm LINUX 2000（以下Storm）のCD-ROMをドライブに入れてマシンを再起動します。CD-ROMブートできない場合は、起動用フロッピーを以下の手順で作ります。

（1）Windowsを起動し、CD-ROMをドライブにセットします。（2）DOS窓を開き、d: [Enter]と入力し、カレントドライブをCD-ROMドライブに変更します。CD-ROMのドライブ名が「D:」以外の場合は、環境に合わせて読み替えてください。(3) mkbtdk [Enter]と入力し、フォーマット済みのフロッピーをドライブに入れて、[Enter]を押します。環境によっては途中でエラーが出る場合がありますが、結果には影響ありません。



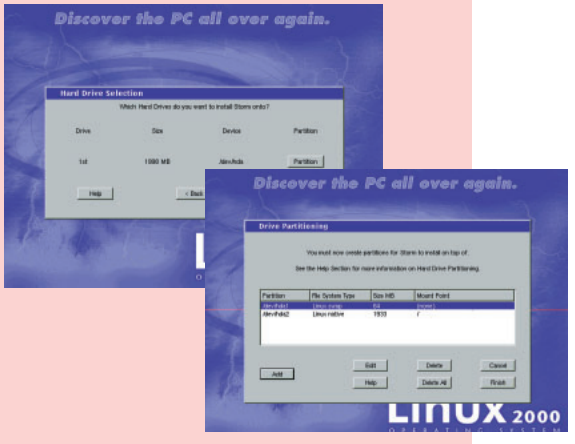
インストール開始からキーボード指定まで

CD-ROM、または上記の手順で作成したフロッピーから起動すると、Stormのインストーラが立ち上がります。「boot:」のプロンプトで、[Enter]を押すと、ライセンス条項が表示されますので、よく読んでから「Accept」を選んでください。次にマウスの検知を行います。マウスを少し動かすと自動的に判別します。次にインストールの画面表示をグラフィカルにするか、テキストで行うかを指定します。以下の画面は、グラフィカルを選択した場合のものです。続いてハードウェアの自動選択の結果が表示されます。ベンダー名やデバイス名が表示されない場合もありますが、名前が分からないだけで、デバイスとしては認識されているのでそのまま「Next >」を押して次の画面に進みます。ここでは使っているキーボードの種類を指定します。国内で多く使われているのは、106または109キーボードです。どちらも「qwerty/jp106」を指定します。



X Window System

StormでX Window Systemを使用するかどうか指定します。「Yes」を選択すると、続けて画面の色数、解像度、モニタの種類を質問されますので、自分のマシンに合わせて指定します。次にビデオカードのチェックが行われます。正しく認識されているようなら、次に進みます。検出されなかったり、検出結果が間違っていたときは、続いて表示されるリストの中から選択します。もしリスト内に使っているビデオカードがなければ、「Unsupported VGA compatible」を選択します。



パーティション設定

Stormをインストールするパーティションを設定します。「Express」を選択すると、自動的に最適なサイズのパーティションが設定されますが、既存のパーティションをすべて消去しますので、Windowsなど他のOSと共存させるつもりなら絶対に選択してはいけません。「Express」は、Storm専用のマシンを作るときだけ選択するようにしましょう。

以下、「Custom」を選択して手動でパーティションを設定する手順を示します。マシンに接続されているハードディスクの一覧が表示されるので、パーティションを設定するディスクの「Partition」ボタンを押します。次の画面で、「Edit」ボタンを押して必要なパーティションを作成します。「/」（ルート）パーティションとスワップパーティションの2つを作ればよいでしょう。ここではスワップパーティションを、実メモリと同じだけ（ここでは64 Mバイト）とり、残りすべてを「/」パーティションに割り当てています。

ネットワークの設定

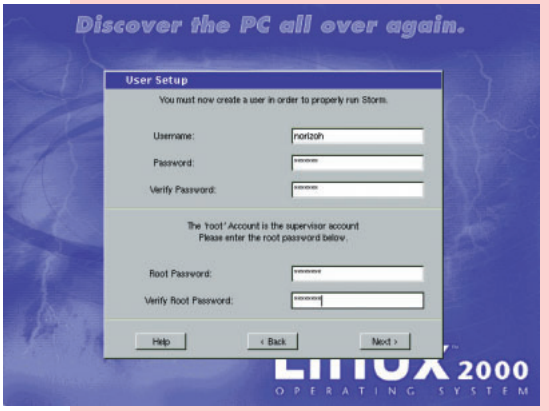
4

次に、必要ならばネットワークの設定を行います。ホスト名、IPアドレス、ネームサーバ（DNSサーバ）などを指定します。ネットワーク上のDHCPサーバからアドレスを割り当ててもらうなら、ここではネットワークの設定をせずに、インストール終了後にStormが起動したあとで、設定を行います。

ユーザーの設定

5

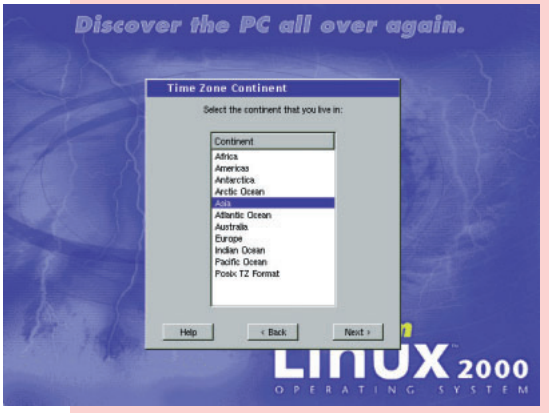
ユーザー名とパスワードを設定します。ユーザー名は、英小文字と数字の組み合わせだけが有効です。また、同時に管理者であるroot（ルート）のパスワードもこの画面で設定します。パスワードは確認のため、それぞれ2回ずつ入力します。



タイムゾーンの設定

6

タイムゾーンを設定します。日本国内なら、Asia→Japan→Tokyoと選んでいけばいいでしょう。ところでTokyoの下に、「Ishigaki」という表示がありますが、日本にはタイムゾーンが2つ（東京、石垣）あるのでしょうか。初めて知りました。



パッケージの選択

7

インストールするパッケージを選択します。最小限必要なものをインストールしておいて、使っているうちに必要になったものを順次インストールしていくのが、Stormまたはその元となっているDebianスタイルですが、よく分からなければ、とりあえず全部インストールしてみましょう。Red Hat系のディストリビューションと違って、この画面で全部を選んでも、ディスクの消費量は400 Mバイト程度です。

次の画面では、デスクトップ環境を選択します。これもデフォルトのKDEだけでなくGNOMEもインストールしておいて両方試してみるといいでしょう。

以上で設定する項目は終わりです。続いてパッケージのインストールが行われますので、終了までしばらく待ちましょう。CD-ROMを取り出してリセットボタンを押すように、というメッセージが出たら終了です。

ネットワーキングでWinもMacも快適だ!

ホームサー

ハを立てよう!

ダイヤル・オン・デマンド、IPマスカレード、Webプロキシ、Samba、Netatalkなど、LAN＋ダイヤルアップで幸福な家庭を築こう！

illustration : Aki

「ホームサーバ」にチャレンジしよう！

文：編集部　Text : Linux magazine

最近のディストリビューションは、GNOMEやKDEによる統合デスクトップ環境や、インストール直後から使える日本語対応ソフトウェアが提供されている。しかし、今までWindowsやMacintoshだけを利用してきた人が、いきなりLinuxをメイン環境にするのは難しいだろう。また、「Linuxをインストールしたけどこの後は何に使えばいいの？」という疑問を持っている人も多いようだ。

もしあなたがこれらの条件にあてはまり、複数のマシンをお持ちなら、Linuxによるホームサーバの構築をお勧めしたい。

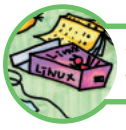
ホームサーバとは？

Linuxは、強力なネットワーク機能をサポートしており、安定性も十分なので、サーバにはぴったりだ。事実、サーバにするというのは、Linuxマシンの使われ方の中でもとてもポピュラーなのだ。

Linux magazineがお勧めするホームサーバとは、家庭の中でWindowsマシンやMacintoshとLANでつなぎ、どのマシンからもインターネットを利用できるようにしたり、WindowsマシンやMacintoshからLANを介してファイルを保存できるようにするといった、実用的なものだ。家族みんなで使えば、その便利さは倍増すること間違いなしである。

家庭で使うためには、導入コストも重要だ。家庭で使うサーバであれば、高いマシンスペックは必要ない。CPUは486クラスでも十分なくらいなので、古くなってホコリをかぶっているマシンがあれば、それを活用しよう。

この特集で解説するホームサーバは大きく分けて3つのサービスを提供する（**図1**）。このセクションでは、それら3つの概要を説明し、次のセクションから各サーバアプリケーションの設定を詳しく解説する。



インターネットにつなぐ

PCユーザーの間では、インターネットの利用はすでに当たり前となった。多くのユーザーは、プロバイダにダイヤルアップで接続してインターネットを使っているはずだ。

Linuxには、モデムやTA（ターミナルアダプタ）でダイヤルアップ接続をするためのソフトウェアが何種類かあるが、今回は、その中でも豊富な動作実績を持ち、設定もわかりやすい国産ソフトウェア、“ PPxP ”を紹介する。

PPxPには、ダイヤル・オン・デマンドという機能がある。これは、ユーザーが特別な接続操作を行わなくても、必要に応じてプロバイダに自動接続してくれる機能だ。また、インターネットの利用が終わり、一定の時間データが流れなくなると自動的に接続を切ってくれる。

そして、インターネット接続の際に、Linuxが提供する強力な機能が“ IPマスカレード ”と呼ばれるものだ。この機能は、LinuxサーバにつないだモデムやTAでインターネットに接続していれば、LAN内にあるすべてのマシンで同時にインターネットを利用できるというものだ。ダイヤル・オン・デマンドと組み合わせて使うと、LANにつないだどのマシンでもプロバイダに自動接続して、Webブラウジングを始めとするインターネット利用ができるようになるのだ。これによって、モデムやTAでもISDNルータに匹敵するインターネット環境を得ることができる。

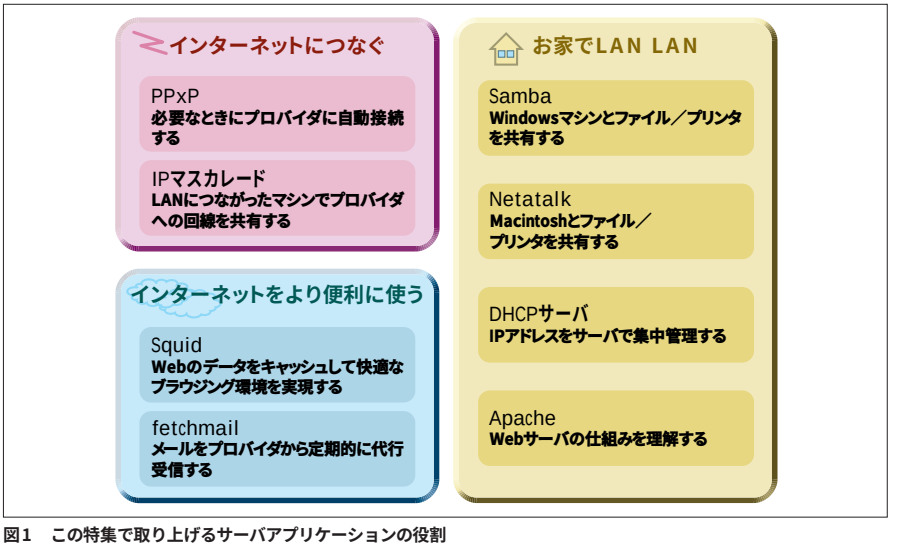
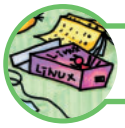


図1　この特集で取り上げるサーバアプリケーションの役割



インターネットをより便利に使う

ここまでで、インターネットを利用する一般的な環境は整うのだが、せっかくLinuxを利用したホームサーバを立てるのだから、もう一歩踏み込んで、より快適なインターネット利用環境を目指してみよう。すでにISDNルータを利用しているユーザーにとっても、きっと役に立つはずだ。

インターネットでよく利用されているサービスといえば、Webのブラウジングとメールだろう。ホームサーバの導入によって、これらのサービスをもっと快適に利用できるようになる。

Webのブラウジング環境を改善するのが“Squid”というキャッシュプロキシだ。Squidは、WebサイトやFTPサイトから流れてくるデータを一時的に保存（キャッシング）しておき、Webブラウザが再び同じデータを要求すると保存してあったデータをブラウザに返す。これにより、よく見に行くサイトのデータが蓄積され、高速なレスポンスが得られるようになる。Netscape NavigatorやインターネットエクスプローラなどのWebブラウザも、個別にキャッシュ機能を持っているが、キャッシュプロキシを使えば、キャッシュしたデータをすべてのマシンで共有するため、より効率の良いキャッシングができる。

次はメール環境の強化だ。複数のメーリングリストに加入していたりすると、プロバイダのメールボックスがいっぱいにならないかと不安になることがある。このようなときに威力を発揮するのが“fetchmail”というソフトウェアだ。fetchmailは、ユーザーが指定する時間間隔で、プロバイダからメールを代行受信する。取ってきたメールはLinuxサーバに蓄えられるので、ディスクの空き容量さえ確保しておけばプロバイダのメールボックス容量を心配する必要はない。Linuxサーバにたまったメールは、イーサネットによるLANでつながったWindowsマシンなどで読むことができる。イーサネットはISDNに比べても桁違いに速いので、たくさんのメールが溜まっていても素早く読むことができる。

お家でLAN LAN

ホームサーバが活躍するのは、インターネットに接続しているときだけではない。

“Samba”というソフトウェアをLinuxサーバにインストールすれば、WindowsマシンからLinuxサーバに接続したディスクやプリンタを利用できるようになる。Windowsだけではない。“Netatalk”というソフトウェアによって、Macintoshからも同じことができるのだ。

Linuxサーバを経由すれば、WindowsとMacintoshでのファイル交換も簡単にできるようになる。また、保存するファイルをLinuxサーバに集めて一元管理することで、バックアップ作業も格段に楽になるだろう。さらに、プリンタを共有すると、印刷のたびにプリンタのつながったマシンに移動したり、プリンタを持ってうろうろすることもなくなる。

LANでTCP/IPを利用していると、意外に面倒なのがIPアドレスの管理である。LANにつないでいる各マシンで個別にTCP/IPの設定をしていると、どのマシンがどのアドレスを使っているのかがわからなくなったりする。

このような問題を解決してくれるのが“DHCPサーバ”だ。DHCPサーバは、ネットワークにつながったマシンに対して、IPアドレスをはじめとするネットワークの設定情報を配布する。

各マシンのネットワーク設定は、DHCPでTCP/IPの設定を行うようにしておけば、それぞれ個別に設定する必要はない。LANの中で使われるIPアドレスをLinuxサーバで管理できるので、同じIPアドレスを複数のマシンに設定してしまうおそれもない。また、ノートパソコンを会社に持っていっているときなど、会社でもDHCPサーバが運用されていれば、そのままの設定で会社のLANにも家のLANにもつなぎ換えることができるようになる。

ところで、最近はプロバイダに自分のWebページを持っている人も珍しくない。本誌の読者にもWebマスターが多いのではないだろうか。Webページを運営していると、作ったページを公開する前にテストしたいことも多いだろう。そんなときは、LinuxサーバでWebサーバソフトウェア“Apache”を動かしてしまおう。そうすれば、世界中に公開する前に、自宅のLAN環境でWebページのテストをすることができるようになる。また、プロバイダとは違ってCGIやSSIの利用制限もないので、Webベースのグループウェアによる家族の掲示板を作ることも可能だ。

必要なものをインストールしよう

この特集では、上記8種類のサーバソフトウェアを紹介するが、すべてのソフトウェアを使う必要はない。欲しいものだけをチョイスしてもいいし、時間をかけて少しずつサービスを増やしてもいい。サーバ構築が初めてでも恐れずにトライしてみてほしい。苦労するだけの価値はあるはずだ。

PPxP～ダイヤルアップ接続とPPPサーバの設定～

文：清水正人　Text : Masato Shimizu

PPxPは1997年末に真鍋氏が公開を始めたPPPシステムで、設定が比較的簡単なのが大きな特徴だ。当時LinuxやFreeBSD上で利用され比較的わかりやすいと言われていたIIJ-PPPを手本に、さらに便利な機能を包括して作られおり、Vine Linuxなどでもデフォルトで採用されている。簡単にまとめるとPPxPには次のような特徴がある。

1. **操作が簡単な、ユーザープロセスPPP**
2. **コマンドライン、X用（XのGUI用とTcl/Tkを使ったものと両方あり）両方のインターフェイスが用意されている**
3. **自動（オンデマンド）接続が可能**
4. **IPパケットのフィルタリング機能**

日本語のサイト（http://www.linet.gr.jp/~manabe/PPxP/）にドキュメントがあり、専用メーリングリストも用意され、問題解決が比較的容易なことが大きなメリットだ。ちなみにPPxPはLinuxのみならずFreeBSDやNetBSDなどでも動作する。

PPxPによるダイヤルアップ接続

PPxPはカーネルに組み込まれたppp-2.xなどのpppドライバを使用せず、その代わりとしてuserlinkドライバを利用する。よってppxpパッケージとuserlinkパッケージの両方が必要となる。双方の最新版は上記真鍋氏のPPxPページからダウンロードすることができる。また以降の作業はLASER5 Linux 6.0で行ったため、ディストリビューションなどによってディレクトリ構成やファイル配置に違いがあることに注意していただきたい。

今回設定に使用したパッケージ
ppxp-0.99120923.tar.gz
userlink-0.99a.tar.gz

■userlinkモジュールの作成

入手したuserlinkを展開し、コンパイル、インストールする。

```
$ tar xvzf userlink-0.99a.tar.gz
$ cd userlink-0.99a
$ ./configure
$ make
$ su
# make install
# /sbin/depmod -a
```

インストールが終了したらmod probeコマンドを実行して、userlinkモジュールをカーネルに組み込む作業が必要になる。もしカーネルを再構築した場合には、モジュールも作り直す必要があるので注意すること。

```
# /sbin/modprobe userlink
```

/sbin/lsmodコマンドで組み込まれているモジュールが表示される。

```
# /sbin/lsmod
Module          Size  Used by
userlink        2880  0     (unused)
```

（以下省略）

このようにuserlinkが表示されればOKだ。Linuxマシン起動時に自動的に組み込むようにするには、/etc/rc.d/rc.localの最後に「/sbin/modprobe userlink」を書き加えればよい。

■PPxPのインストール

インストール前にdoc/ja_JP/INSTALL.txtファイルに目を通しておくようにしていただきたい。またQuickStart.txtにuserlinkモジュールのインストール方法、および各ディストリビューションでPPxPを利用するためのユーザー、グループの設定方法が書いてあるので、よく読んでファイルの属性などを変更しておく必要がある（ここではRed Hat系のケースを紹介する）。まずppxpをビルドする。

```
$ tar xvzf ppxp-0.99120923.tar.gz
$ cd ppxp
```

Column

userlinkの代わりにethertapを使う

カーネル2.2.xを使用している場合には、userlinkモジュールの代わりに「ethertap」というトンネルドライバを利用することができる。「Networking device support」で「Ethertap network tap」を指定してカーネルの再構築を行い、さらにデバイスを作成（スペシャルファイルとして/dev/tap0など）するなどの作業が必要になる。詳しくはPPxPのパッケージを展開した時に作られるdocディレクトリ以下にインストールされるQuickStartを参照のこと。

```
$ ./configure
$ make
$ su
# make install
```

次に、PPxPを実行するユーザーがモデムを接続するシリアルポートにアクセスできるように、デバイスファイル（ttyS＊）のアクセス権とグループを変更する。デバイスファイルのユーザー／グループの設定方法は2通りある。1つの方法はPPxPを利用するユーザーをttyグループ、およびuucpグループ両方に追加するものだ。

もう1つの方法はPPxPを利用するユーザーをuucpグループに加え、続けて/dev/ttyS＊の属性をuucpグループに変更すればよい。ここでは、後者の方法で行った。グループの追加は、

```
# /usr/sbin/vigr
```

として、エディタで「uucp::14:uucp」の行に、「uucp::14:uucp,ken」のようにユーザー名を追加する。

```
# chmod 660 /dev/ttyS*
```

```
# chgrp uucp /dev/ttyS*
```

また、デバイス名/dev/modemでアクセスできるように、リンクを張っておくと便利だろう。

```
# ln -s /dev/ttyS0 /dev/modem
```

モデムをつないだポートがCOM1なら/dev/ttyS0、COM2なら/dev/ttyS1を使用する。

現在の状態を「ls -al /dev/ttyS＊ /dev/modem」として確認する（**画面1**）。

デフォルトでインストールされるPPxP関連のバイナリは/usr/local/bin以下に配置されるので、必要があればこのディレクトリにパスを通しておく必要がある。たとえばホームディレクトリにある「.bash_profile」の「PATH=〜」の行に/usr/local/binを追加したのち、sourceコマンドを実行して設定ファイルを再読込みする。

```
$ source .bash_profile
```

■PPxPの起動と確認

まずターミナルなどから利用する方法を紹介する。ktermなどでppxpコマンドを実行するとプロンプトが「ppxp>」に変わる。

```
$ ppxp
PPxP version 0.99120923
interface: ul0
ppxp>
```

userlinkドライバが組み込まれていない場合などはエラーとなる。

■ダイヤルアップの初期設定

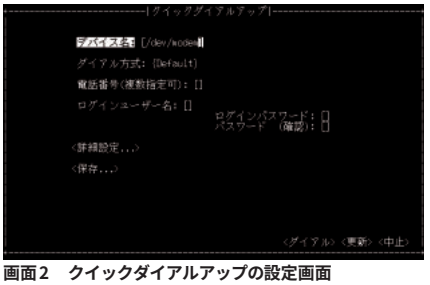
ppxp>プロンプトに続けてqdialとタイプしてEnterキーを押し「クイックダイアルアップ」画面を出し、初期設定を行う（**画面2**）。各項目の移動はTabキーで、選択する場合には左右のカーソルキーで切り替えることができる。

1. **デバイス名：モデム、TAを接続しているシリアルポートで、DOSでいうCOM1なら/dev/ttyS0、COM2なら/dev/ttyS1となる。**
2. **ダイアル方式：左右のカーソルキーで切り替わる。トーン回線（ATD）ならTone、パルス回線（ATDP）ならPulseを選択する。**
3. **電話番号：ISP（ダイヤルアップ接続先）の電話番号を入力する。**
4. **ログインユーザー名：ユーザー名を入力する。**
5. **ログインパスワード：パスワードを入力する。入力文字は表示されない。間違えたらCTRLキーを押しながら'u'キーを押して最初からやり直すことができる。確認用に再度同じパスワードを入力する。**

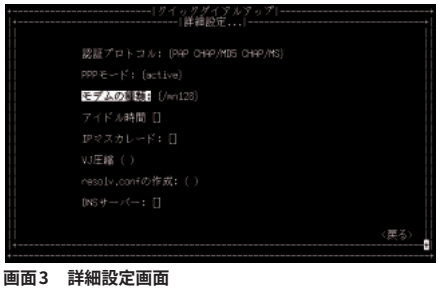
ここで<保存>ボタンを押して設定した内容を、スクリプトとして保存しておく。

```
# ls -al /dev/ttyS* /dev/modem
lrwxrwxrwx   1 root     root           10 Jan 27 21:03 /dev/modem -> /dev/ttyS0
crw-rw----   1 root     uucp       4,  64 May  6  1998 /dev/ttyS0
crw-rw----   1 root     uucp       4,  65 May  6  1998 /dev/ttyS1
crw-rw----   1 root     uucp       4,  66 May  6  1998 /dev/ttyS2
crw-rw----   1 root     uucp       4,  67 May  6  1998 /dev/ttyS3
```

画面1　モデムを接続するデバイスファイル



画面2　クイックダイアルアップの設定画面



画面3　詳細設定画面
モデムの種類、ISPのDNSサーバのIPアドレスなどを設定する。

■ダイヤルアップの詳細設定

<詳細設定>ボタンを押すと、詳細設定画面に移る（**画面3**）。

1. 認証プロトコル：通常はPAPかCHAPのいずれかを利用する。通常は「PAP CHAP/MD5 CHAP/MS」でよい。プロバイダが対応可能であればパスワードをプレーンテキストで流さないCHAPのほうが安全だ。
2. PPPモード：通常はactiveでよい。
3. モデムの種類：使用するモデムを選択する。アナログモデムであればgenericが広く対応する。このほかにもTAなどいくつか代表的な機種が用意されている。
4. アイドル時間：オンデマンド接続をする場合、TCP/IPのパケットが流れなくなってから自動切断するまでの時間（分）を指定する。0で自動切断しない設定となる。
5. IPマスカレード：IPマスカレードを利用する場合の範囲を入力する。
6. resolv.confの作成：ISPに接続した時のみ専用の/etc/resolv.confを一時的に作成して参照し、切断時にオリジナルのファイルに戻す。これを指定すれば、通常のLAN内部用のLinuxリゾルバとダイヤルアップ接続用のISPのドメイン、DNSを指定したリゾルバの両方を利用することができる。
7. DNSサーバ：ISPのDNSサーバのIPアドレスを入力する。

<戻る>を押して再度「クイックダイアルアップ」に戻り、スクリプトを保存しておく。スクリプトはユーザーのホームディレクトリ以下.ppxp/confに配置される（rootで設定した場合は/usr/local/etc/ppxp/conf/に配置される）。

■接続の確認

「クイックダイアルアップ」画面右下の<ダイアル>を押す。するとターミナルはppxp>プロンプトに戻り、接続が開始される。プロンプトがすべて大文字のPPXP>となれば接続成功となる。接続を切断するにはdisconnectコマンド、PPxPを終了するにはquitコマンドを入力する。

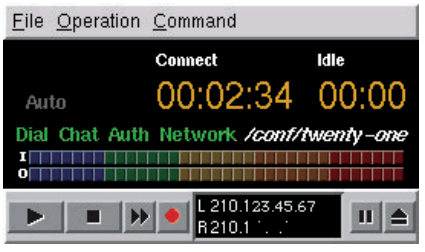
プロンプトの各文字にはそれぞれ意味があり、接続できない場合にはどの文字が小文字になっているかで、原因を推測するとこができる。

最初の「p」：ダイヤリング中（Dial）
次の「p」：接続中（Chat）
「x」：ユーザー認証中（Auth）
最後の「p」：ネットワーク設定中（Network）

```
# ppxp twenty-one
PPxP version 0.99120923
interface: ul0
ppxp> set
-r--r--r-- root     root     VERSION=0.99120923
-r--r--r-- root     root     DATE=000110
-r--r--r-- root     root     TIME=192247
-r--r--r-- root     root     FRAME=IP VJCOMP VJUNCOMP COMP IPCP LCP PAP CHAP
-rw-rw-rw- ken      ken      NAME=twenty-one
-rw-rw-rw- ken      ken      MODE=active
-rw-rw-rw- ken      ken      LINE=/dev/ttyS0
-rw-rw-rw- root     ken      SERIAL.SPEED=115200
-rw-rw-rw- root     ken      SERIAL.FLOW=crtscts
-rw-rw-rw- root     ken      SERIAL.PARITY=none
-rw-rw-rw- ken      ken      SERIAL.MODEM=/generic (Hayes AT compatible generic modem)
-rw------- root     ken      SERIAL.LOCK=<hide>
-rw------- root     root     SERIAL.LOGIN=<hide>
-rw------- root     root     SERIAL.USER=<hide>
-rw-rw-rw- ken      ken      DIAL.LIST=03-XXXX-XXXX/1
-r--r--r-- root     root     DIAL.NUMBER=03-XXXX-XXXX/1
-rw-rw-rw- ken      ken      DIAL.TYPE=Tone
-rw-rw-rw- root     ken      DIAL.INTERVAL=180
-rw-rw-rw- root     ken      DIAL.TIMEOUT=90000
(途中省略)
ppxp> connect
PPXP> disconnect
ppxp>
```

画面4
コマンドラインからPPxPを起動する

もし首尾良く接続できない場合には、電話番号、ユーザー名とパスワードなどはもちろんのこと、モデム／TAの初期化コマンドも確認してみよう。この例ではモデムスクリプトは/usr/local/etc/ppxp/modemディレクトリ以下に収められている。例として「rta50i」というYAMAHA RTA50i用のスクリプトを見ると、初期化コマンドとDTE速度を設定していることがわかるだろう。適当なファイルをコピーし、自分のモデム／TAに合った初期化コマンドに書き換えて保存する。たとえばInitialize行を"AT&F&Q5$S12"などと書き換え「mymodem」などと適当な名前に変えて保存する。続いてクイックダイアルアップの「モデムの種類」で、作成した「mymodem」を選択し、その設定自体を保存すればよい（ファイルのオーナーとグループはrootにする）。



画面5 Tcl/Tkを利用したGUIツールtkppxp
いくぶん大きめのツールだが、接続過程を段階的に見ることができ、ISP側のPPPサーバと自分のLinuxマシンに割り当てられたIPアドレスが表示されるなど、わかりやすいツールだ。

■コマンドラインでの接続

ターミナルから手動で接続をする場合には、**画面4**のように設定ファイル名を引数にしてPPxPを起動する。setコマンドで読み込んだ設定を確認することができる。

ここでconnectコマンドを入力すると接続動作が始まり、接続過程で文字が次々と大文字に替わってゆく。接続が成功するとプロンプトがすべて大文字になる。disconnectコマンドを入力すると切断される。

GUIによるダイヤルアップ接続

ここまではコマンドラインツールの説明をしたが、X上のツールも用意されている。ここではtkppxpを紹介しよう（**画面5**）。tkppxpはTcl/Tkを利用したMusicプレーヤのようなルックのGUIツールで、サイズは同じくX用GUIツールであるxppxpよりもやや大きめで見やすい。設定ファイルは、

```
$ tkppxp <設定ファイル名> &
```

などと引数に指定して読み込むか、あるいは起動後に「File」メニューの「Load configuration」を選択することで指定できる。

コマンドラインで説明した「qdial」もGUIで用意されている。「Operation」メニューの一番下にある「Quick Dialup」で呼び出すことができる。こちらのほうがデバイス名、ダイヤル形式、モデムタイプなどをプルダウンメニューから選ぶことができるので、わかりやすいだろう（**画面6**）。

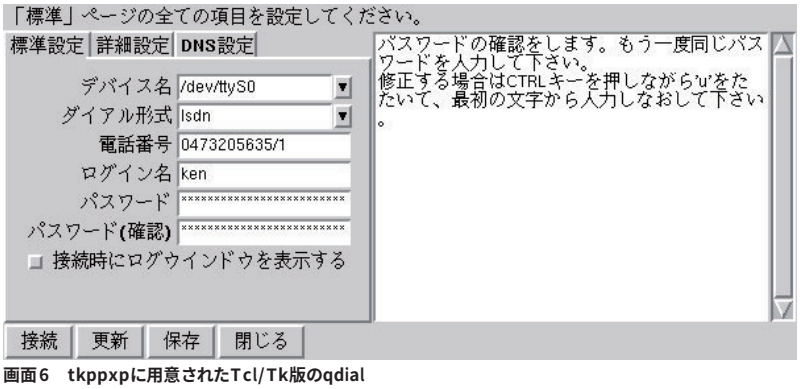
再生ボタンを押すとISPに接続を始める。接続過程がDial、Chat、Auth、Networkで順番に表示され、接続に成功すると設定ファイル名がハイライトになる。

Connectの下に接続経過時間が表示され、通信量（パケット）はデジタルのバーグラフで表示される。ISPのDNS、ローカルのIPアドレスが中央下に表示される。

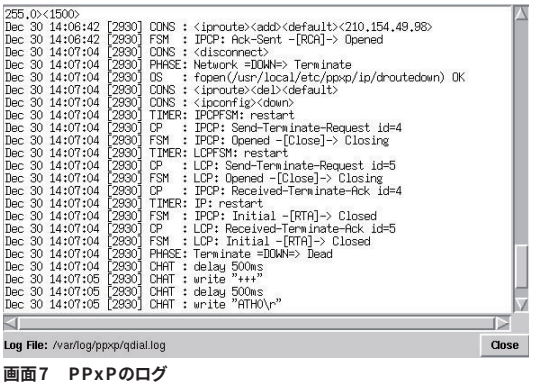
また、接続できないなどの問題があれば「File」メニューの「View Log」でログを表示して問題解決をはかることができる（**画面7**）。

オンデマンド接続

手動でISPに接続するケースでは、たとえばブラウザであるページを見たいときなど、あらかじめISPに接続す



画面6 tkppxpに用意されたTcl/Tk版のqdial
モデムタイプ、デバイス名、ダイヤル形式などはメニューから選択できるので、テキスト版よりも使い勝手がよい。



画面7 PPxPのログ
接続上の問題が出た場合には、すぐにログを見てみよう。実体は/var/log/ppxp/qdial.logファイルが表示されている。

るために「Connect」ボタンを押し、見終わったら「Disconnect」ボタンを押すといった面倒がある。もちろん切断し忘れると、そのまま課金されてしまうことはいうまでもない。

ダイヤルアップルータのように外部アドレス宛のTCP/IPパケットが発生したら自動的に接続し、一定時間パケットが流れなくなると自動切断するオンデマンドを実現するには、kerneld、dialdなどといった別アプリケーション（デーモン）もしくはpppdを利用する方法が一般的だったが、PPxPではこのオンデマンド接続も機能として組み込まれている。また設定も極めてわかりやすいのが特徴だ。

ターミナルのコマンドラインでは次のコマンドを入力する。

```
ppxp> auto on
```

GUIで設定するには、tkppxpを起動し「Operation」メニューから「Auto ON」（xppxpの場合は赤丸の録音ボタン、xppxpmの場合はメニューからAuto ONを選択）を押すと、画面にAutoの文字が浮かび上がる。オンデマンドを中止したい場合には、その下の「Auto OFF」を押せばよい。

自動切断時間を設定するには「Operation」メニューの一番下にある「Quick Dialup」の「詳細設定」で「アイドル時間」を設定する。パケットが流れなくなってから切断動作に入るまでの時間（秒）を指定する。設定ファイルには「set IDLE.INIT 60」などと秒数で記述されているので、これを書き換えることもできる。一般的には120～180秒ほどでよいだろう。0に設定すると自動切断しなくなる。

■フィルタリング機能

オンデマンド接続を利用すると問題になるのが不必要な自動接続だ。たとえば、LAN内にWindowsマシンがある場合にWindowsがまき散らすNetBIOSパケットにより自動的に接続されるケースなどがある。PPxPではこのような場合にフィルタリング機能を使って対応することができる。

IPパケットのフィルタリングついては、/usr/local/etc/ppxp/conf/filter01ファイルに定義されている。

デフォルトのfilter01ファイルでは、アイドル時間を60秒に設定し、入ってくるパケット（Income Packet）とインターネット側に出ていくパケット（Outgoing Packet）それぞれについて、ftp、www、nntp、telnetなどに分けて規定している。不具合があるようならば、書式を見ながら設定値を変えることになる。

そしてこのfilter01ファイルを、confに置いたISP接続用に作成したファイルの先頭に、

```
source filter01
```

のように追加して読み込ませればよい。

このほか詳しい情報については付属の日本語ドキュメント、Webサイト、あるいは専用メーリングリストを参照のこと。

PPPサーバの着信設定

ここまででPPxPによるISPへの接続については環境が整ったと思うが、次に着信設定をしてみよう。PPPサーバを立てれば、外出先から自分のLAN内にダイヤルアップ接続をして資源を利

Column

xppxpとxppxpm

Vine LinuxではデフォルトでuserlinkとPPxPパッケージが組み込まれている。そしてデフォルトのデスクトップ右側、上から5番目のアイコンがPPxPの小さなGUIインターフェイスであるxppxpmだ。

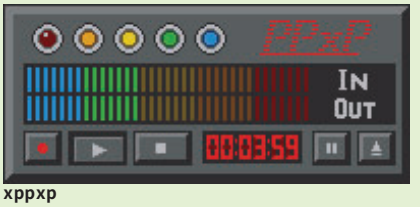
xppxpmの中央部分をマウスで左クリックするとメニューが現れる。ここでConnect、Disconnect、Bye（終了）などのコマンドを実行することができる。また、メニューの一番上にある「Auto ON」はオンデマンド接続機能を利用する場合に使用する。xppxpmはそのサイズから、デスクトップ上でのPPP接続の確認などに利用するとよいだろう。

このほか、本文で紹介したtkppxpよりも小振りでシンプルなxppxpもある。xppxpもtkppxpと同じように設定ファイル名を指定して起動できる。

```
$ xppxp <設定ファイル名> &
```

xppxpはxppxpmを大きくしたサイズで、パケットの流量メータや接続操作ボタンなどが前面に配置されている。

xppxpm
Vine Linuxのデフォルトのデスクトップ上にも現れるdockサイズのインターフェイス。マウス左クリックでメニューを呼び出し、PPP接続／切断／オンデマンド接続などが可能だ。



xppxp
xppxpmよりは大きなGUIインターフェイスで、前面にボタンがついている。

用できるので、利用価値が高いだろう。PPxPはPPPサーバとしても利用できるので、その方法を紹介しよう。

着信設定をしてPPPサーバを立ち上げるにはmgetty+sendfaxというプログラムを利用する方法が多く利用されている。mgettyを使ってモデム／TAが接続されているシリアルポート（ここではttyS0）を常時監視する。着信があったらmgettyがPPxPの本体であるppxpdを呼び出し、認証を経て接続を行う。また、認証方式はPAPをメインに考えた。

■mgetty-sendfaxのインストール

mgettyを使うには、次のパッケージをインストールする。

```
mgetty-1.1.14-8.i386.rpm
mgetty-sendfax-1.1.14-8.i386.rpm
```

最近のディストリビューションではどちらもデフォルトでインストールされている場合が多いので、

```
$ rpm -q mgetty
$ rpm -q mgetty-sendfax
```

などで確認しておこう。ここで注意することは「mgettyプログラム自体がAUTO_PPPオプションを付けてコンパイルされている必要がある」ということだ。ただし最近のRed Hat LinuxやLASER5 Linux 6.0に含まれるmgettyではあらかじめAUTO_PPPオプションを付けてビルドされているようで、そのまま利用することができた。

■mgettyの設定

LASER5 Linux 6.0ではデフォルトで/etc/mgetty+sendfaxディレクトリに設定ファイルがインストールされている。基本的には、

1. **mgetty.configファイルでポート、モデムの初期化コマンドなどを指定する。**
2. **login.configファイルでAuto PPPを有効にする。**
3. **ppxpdが接続に利用する設定ファルを作成する。**

という3つの作業を行う。

■mgetty.configの編集

コメントアウトされている以下の箇所のコメント（#）を外し、着信させるデバイスの所有者とグループを設定する。ここでは/dev/ttyS0をuucpグループに設定してあるので以下のように設定した。

```
port-owner root
port-group uucp
port-mode 0664
```

PPxPの利用と同じように、着信するユーザーも同様にuucpグループに属している必要がある。さらにその下の行で使用するポート（デバイス）と通信速度（シリアルポートとモデム間の速度）を定義する。モデムの初期化コマンドはあとで紹介する設定ファイル（ここではppp）で記述するか、あるいはsourceとして該当するモデム設定ファイルを読み込ませてもよい。

ここではmgetty.configファイルにモデム初期化コマンドを設定した。モデム初期化コマンドは通常PPxPでISPへの接続時に利用しているものでよい（/usr/local/etc/ppxp/modemディレクトリ以下にあるファイルを参照のこと）。

```
port ttyS0
speed 115200
```

画面8 PPxPのサイト
http://www.linet.gr.jp/~manabe/PPxP/
各種情報はここから得られる。特にガイド関連にはよく読むことをお勧めする。

Column

AUTO_PPPオプションの付け方

mgettyをコンパイルする時に、Makefileの該当行に次のように「-DAUTO_PPP」を書き加える。

```
CFLAGS=-2 -Wall -pippe -DAUTO_PPP
```

また、policy.hは本来環境に合わせてpolicy.h-distを編集して作成するものだが、そのままコピーしても利用できる。あとは、

```
# make
# make install
```

でインストールが終了する。Red Hat系ならばSRPMをインストールし、/usr/src/redhat/SOURCESディレクトリに展開されるtar.gzに含まれるMakefileを編集し、rpm -bbコマンドでバイナリパッケージを作成後、そのパッケージをインストールしたほうがバージョン管理の面からも有効だろう。

```
modem-type data
data-only y
init-chat " " ATZ OK ATQ0V1E1X3&Q5
OK
```

■login.configの編集

まず下の行の先頭に#を挿入してコメントアウトする。

```
#/FIDO/          uucp          fido
/usr/local/lib/fnet/ifcico @
```

次に、「#/AutoPPP/～」の行のコメントを外し（行頭の#を削除）、Auto PPPを有効にして、ppxpdを起動するように編集する。

```
/AutoPPP/          -          a_ppp
/usr/local/sbin/ppxpd -direct ppp
```

AutoPPPを有効にして、接続してきた相手がPPPであればppxpdが起動され、スクリプトpppを実行する。スクリプトpppはPPxP設定ファイルがデフォルトで置かれている/usr/local/etc/ppxp/confディレクトリに作成する。

最後に下の行もコメントアウトして、ファイルを保存する。

```
#*     -     -     /bin/login    @
```

■/usr/local/etc/ppxp/conf/pppスクリプトの作成

エディタで**画面9**のような内容のファイルを作成する。同じディレクトリにあるファイル「server」が近い内容なので、コピーして編集するとよいだろう。それぞれの行は「PPxPガイド」3章で解説されている変数の各項目をよく読んで自分の環境に合ったものに書き換える。さらに書き出すログファイルはtouchコマンドであらかじめ作成しておく。

■/etc/inittabの編集

重複しない番号（ここではs0）を割り当て、mgettyを起動するランレベル、mgettyへのパスなどを指定し、スペースで区切りデバイス名を指定する。mingettyでコンソール1～6を定義している箇所の下あたりに、次のような行を追加する。この設定ではランレベル2～5でmgettyが常時シリアルポートttyS0を監視し続ける。mgettyはいったん処理を終えても、initによって再度立ち上げられ（respawnが指定されているため）ポートの監視を続けるわけだ。

```
# Serial lines
s0:2345:respawn:/sbin/mgetty ttyS0
```

最後にinitコマンドを実行して変更を読み込ませ、ps axコマンドでmgettyの起動を確認しておこう。

```
# init q
# ps ax | grep mgetty
xxx ?   S      0:00  /sbin/mgetty ttyS0
```

■mgettyの確認

mgettyがシリアルポートを監視しているかどうかは、ログファイルで確認することができる。**画面10**のようにwaiting..と待ち受け状態になっていればよい。mgettyは60分など一定時間おきにシリアルポートにATコマンドを発行してモデムから応答があるかどうかをチェックしてくれる。

■外部からの接続

/etc/mgetty+sendfax/mgetty.configファイルでモデムを初期化した条件に合わせて、外部からttyS0に接続したモデム／TAに電話をかけて接続を試みよう。うまくいかない場合には/usr/local/etc/ppxp/conf/ディレクトリに置いた設定ファイルの「set LOG.LEVEL ～」を増やして、なるべく多く情報を収集するとよいだろう。

```
source qdial
set LOG.FILE /var/log/ppxp/ppp.log
set LOG.LEVEL PHASE FSM OS AUTH CP CHAT CONS
set IP.REMOTE 192.168.0.0/24
set IP.SLOCAL yes
set IP.NETMASK 255.255.255.0
set IP.UP hrouteup
set IP.DOWN hroutedown
set AUTH.PROTO PAP CHAT
set AUTH.SERVER unix
set IP.PROXYARP yes
set IP.RESOLV no
set IP.DNSRELAY no
```

画面9 /usr/local/etc/ppxp/conf/pppの例

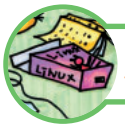
```
# less /var/log/mgetty.log.ttyS0
yS0    mgetty : experimental test release 1.1.14-Apr02
yS0    check for lockfiles
yS0    locking the line
yS0    lowering DTR to reset Modem
yS0    send : ATZ[0d]
yS0    waiting for "OK"  ** found **
yS0    send : ATQ0V1E1X3&Q5[0d]
yS0    waiting for "OK"  ** found **
yS0    waiting...
```

画面10 mgettyが正しく動作しているかログファイルで確認する

IPマスカレード～Linuxサーバを経由してインターネットに接続する～

文：清水正人　*Text : Masato Shimizu*

PPxPのAuto（オンデマンド）機能を使うことで、Linuxサーバで必要に応じて自動的にISPへの接続／切断ができるようになった。しかし、このままでは、メリットを受けられるのはLinuxサーバに限定されている。Linuxサーバに接続したモデム／TAを共有できれば、LAN内にあるすべてのコンピュータからインターネットに接続できることになる。これを実現するのがIPマスカレードだ。



IPマスカレードの仕組み

LinuxサーバがISPにダイヤルアップ接続している時は、ISPからLinuxサーバに対してグローバルアドレスという公（おおやけ）のアドレスが割り当てられている。グローバルアドレスが割り当てられるのは当然のことながらLinuxサーバだけなので、プライベートアドレスが割り当てられているLAN内のほかのコンピュータはLinuxサーバに接続されたモデム／TAを経由して外に出ることができない。

これを解決するためにIPマスカレードという仕組みを利用する。LinuxサーバでIPマスカレードを動作させることで、LAN内のプライベートアドレスを、Linuxサーバが ISPから割り当てられたグローバルアドレスに書き換えてインターネットに送り出す。

たとえば**図1**のクライアントマシンpuppyから、Webブラウザを使ってhttp://～というページを閲覧するリクエストを出したとする。puppyでは、あらかじめデフォルトゲートウェイにLinuxサーバであるcat2のプライベートアドレスを設定しておくだけでよい。

HTTPリクエストはcat2に送られ、IPマスカレードが動作しているcat2は外部宛てのTCP/IPパケットと判断すると、送信元のプライベートアドレスを自身に割り当てられているグローバルアドレスに書き換えて外部に送信しようとする。

もしcat2上でPPxPによるオンデマンド設定がしてあれば、cat2はこのタイミングでISPに対して自動接続を行うので、LAN内のどのコンピュータでもPPP接続操作をする必要はない。

リクエストに対して外部から戻ってきたパケットは、cat2が発信元puppyのプライベートアドレスに書き換えて

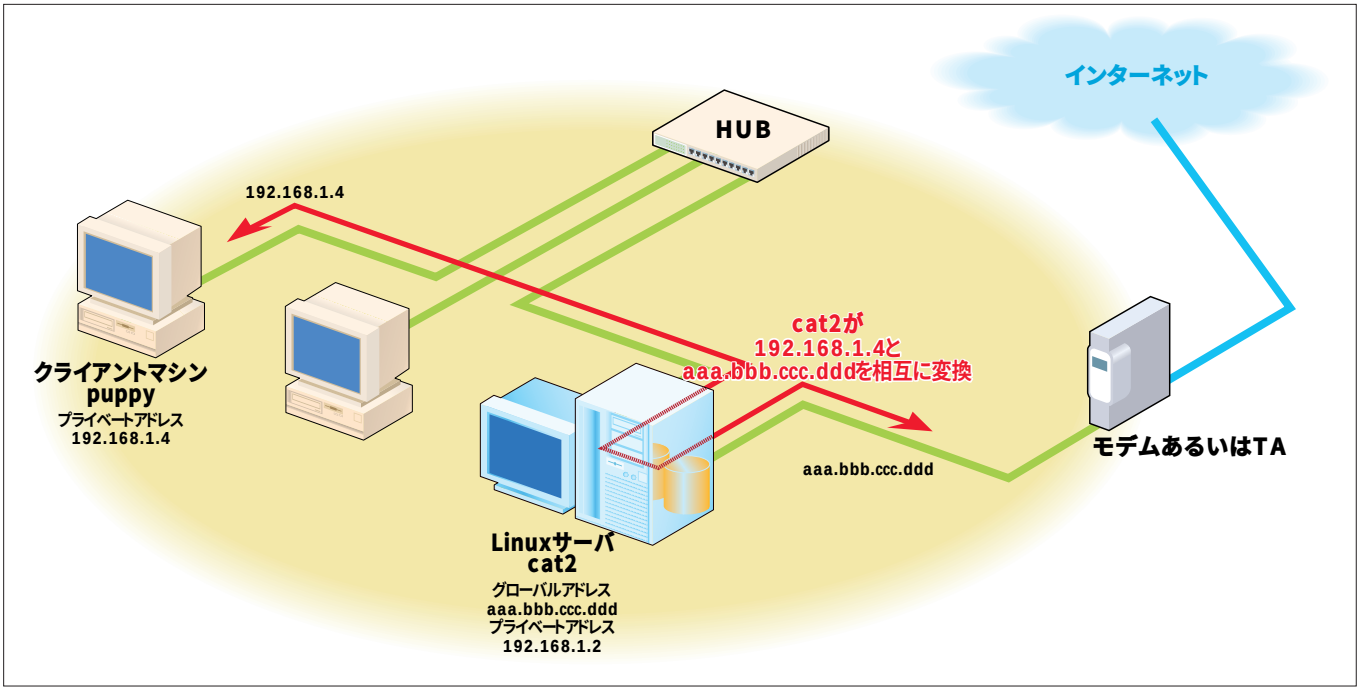


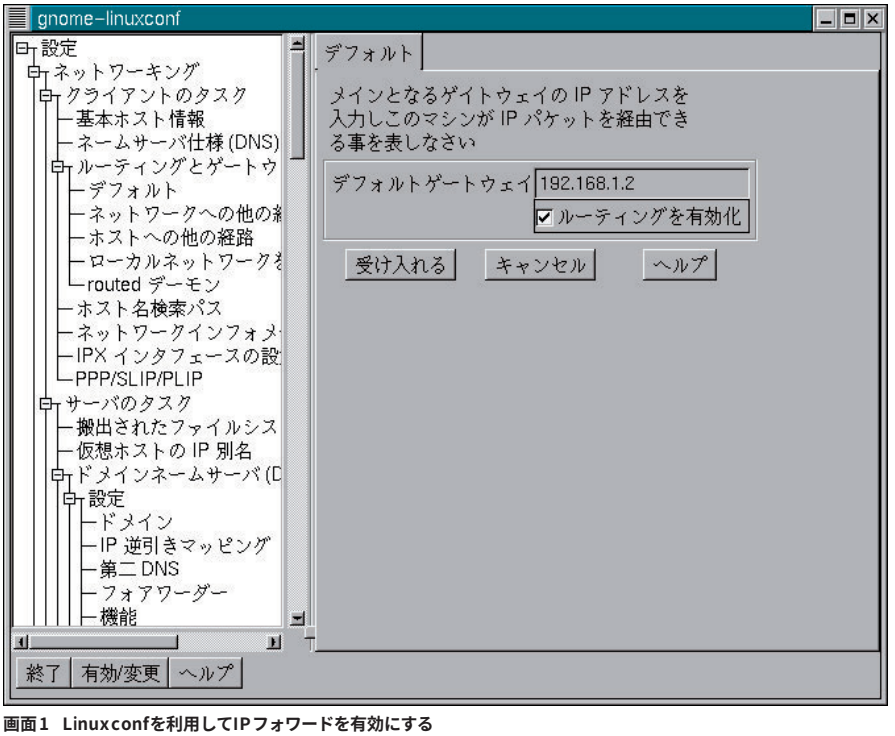
図1　IPマスカレードを利用したインターネット接続

LAN内部に送り返す。したがってpuppyはあたかも自身にモデム／TAが接続されているかのようにインターネットにアクセスすることができる。

このようにIPマスカレードを設定することで、1つのグローバルアドレスをLAN内で共有することができるのだ。

IPマスカレードには次のような特徴がある。

1. **LAN内部のコンピュータはデフォルトゲートウェイとしてLinuxサーバを指定するだけで、ほかの設定は不要。**
2. **LAN内部のユーザーは、Linuxサーバを意識することなく透過的にインターネットに接続することができる。**
3. **IPマスカレードは「1対多」のIPアドレス変換を行うため、ISPなどから割り当てられた1つのグローバルアドレスを複数のコンピュータで利用することができる。**
4. **インターネット側からLANを見ると、ゲートウェイとなるLinuxサーバのみが見える。パケットをフィルタリングする機能があるので、ファイアウォールになる。**



画面1　Linuxconfを利用してIPフォワードを有効にする

Column

TurboLinuxにはipchainsが含まれていない

TurboLinux 4.xにはipchainsが含まれていないので、ターボリナックスのFTPサイトからRPMパッケージ（ftp://ftp.turbolinux.co.jp/pub/turbolinuxj/snapshot/TurboContrib/RPMS/ipchains-1.3.8-24.i386.rpm）をダウンロードして、インストールする。

またLinuxconfもないので、IPフォワード機能の設定はturbonetcfgを使うとよいだろう。

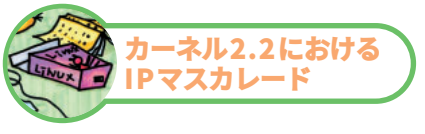
もし、ipchainsが含まれていないディストリビューションで、ソースプログラムからインストールする必要があるのなら、Linux IP Firewalling Chains（ http://www.rustcorp.com/linux/ipchains/）から、ipchains-1.3.9.tar.gzを取得して展開後、make、make installすればよい。

IPマスカレードの設定を行う前に注意することがある。ディストリビューションが採用しているLinuxカーネルの2.0.xと2.2.xとではIPマスカレード機能の一部に違いがある点だ。これは、IPマスカレードがカーネルの機能の一部として取り込まれているためである。

どちらのカーネルもIPマスカレード用にカーネルが構築されていることが前提となるが、たいていのディストリビューションに含まれる標準カーネルはそのままで大丈夫だ。

IPマスカレードを使うために、まず、IPフォワード機能をオンにする。続けてIPマスカレード上で利用するプロトコル（FTPやRealAudioなど）に合わせてモジュールをロードする。

そして最後にコマンドによってパケットの転送、フィルタリングを設定するわけだが、この設定は、Vine Linux 1.1やRed Hat Linux 5.2などのカーネル2.0.xではipfwadmコマンドを使用する。一方、最近のディストリビューションであるLASER5 Linux 6.0やRed Hat Linux 6.0以降、TurboLinux 4.2などのカーネル2.2.xではipchainsコマンドを使用する。



ここでは、ipchainsを使ったLASER 5 Linux 6.0での設定方法を紹介する。

■IPフォワード機能の設定

まずIPフォワード機能をオンにする。Linuxconfを使う方法と、エディタで/etc/sysconfig/networkファイルを直接修正する方法の2通りがあり、いずれの場合もroot権限で実行する。

- **Linuxconfで設定**

GNOMEメニューの［システム］か

らLinuxconfを起動して、［ネットワーキング］（LASER5 Linux 6.0 Rel.2では［ネットワーク経路］）の中の［デフォルト］（Rel.2では［標準ゲートウェイ］）を選択する（**画面1**）。“ルーティングを有効化”をチェックして、左下の“終了”をクリックする。

次に［実際に変更分を実行する］をクリックすると、/etc/sysconfig/networkファイルが書き換えられ、IPフォワード機能が設定される。

・networkファイルをエディタで修正

/etc/sysconfig/networkファイルは先頭から、以下のようになっている。

```
NETWORKING=yes
FORWARD_IPV4=false
HOSTNAME="cat2.home-lan.ne.jp"
```

この2行目を「FORWARD_IPV4=yes」に書き換えて保存する。

networkファイルを書き換えたら、

```
# /etc/rc.d/init.d/network restart
```

と入力しネットワークを再起動する。

設定が終了したら、catコマンドで/proc/sys/net/ipv4/ip_forwardを見て確認する。“1”が表示されればIPフォワードが設定されている。“0”と表示されたときは、正しく動作していない。

```
# cat /proc/sys/net/ipv4/ip_forward
1
```

■必要なモジュールの組み込み

IPマスカレードによってIPパケットのアドレスを変換する過程で、FTP、IRC、RealAudio、CU-SeeMeなどのアプリケーションは、そのままではアドレスを変換して通過することができない。よってこれらのアプリケーションをIPマスカレード越しに利用するには、それぞれに対応したモジュールをあらかじめ組み込んでおく必要がある。

モジュールは/lib/modules/2.2.＊/ipv4ディレクトリに置かれている。ディレクトリ名の＊部分はディストリビューション、カーネルのバージョンなどによって異なる。

FTP、CU-SeeMe、IRC、Quake、RealAudio、VDO Liveについてのモジュールがデフォルトでインストールされているはずなので、必要なものをIPマスカレードの経路設定時にロードしておく。

モジュールのロードは、depmod、modprobeコマンドで行う。

```
# /sbin/depmod -a
# /sbin/modprobe ip_masq_ftp.o
# /sbin/modprobe ip_masq_irc.o
# /sbin/modprobe ip_masq_raudio.o
# /sbin/modprobe ip_masq_cuseeme.o
# /sbin/modprobe ip_masq_vdolive.o
```

モジュールがロードされたかどうかはlsmodコマンドで確認できる。

```
# lsmod
Module          Pages Used by
ip_masq_ftp     1     0
ip_masq_irc     1     0
ip_masq_raudio  1     0
ip_masq_cuseeme 1     0
ip_masq_vdolive 1     0
userlink        1     1
tulip           6     1    (autoclean)
```

■IPマスカレードの設定

ここでは以下のような転送ポリシーを設定する。

1. **LAN内部からインターネットに接続できるように、アドレス変換を行ってIPマスカレードを利用できるようにする。**
2. **LANのプライベートアドレスは192.168.1.0/24　として、「/24」とサブ**

Column

PPxPのIPマスカレード機能

PPxPにもIPマスカレード機能がある。qdial内でアドレスを設定するだけで簡単に利用できる。

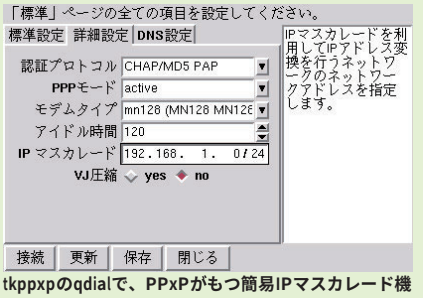
たとえば192.168.1.0〜255のアドレスを変換するには、qdialの詳細設定で、

```
IPマスカレード: [192.168.1.0/24]
```

と設定する。ターミナルのコマンドラインからは、

```
ppxp> filter ip +m0 -S 192.168.1.0/24
```

のようにする。PPxPガイドによるとあまり完成度は高くないとのことだが、簡易的に利用するには手早い方法だ。なお、PPxPのIPマスカレード機能を利用する際には、VJ圧縮（IPのヘッダ圧縮）を使わないようにする必要がある。



tkppxpのqdialで、PPxPがもつ簡易IPマスカレード機能の設定を行う。IPアドレスと「/」の後にビット数を入れるだけの簡単設定。

```
# ipchains -A forward -i ul0 -p tcp -s 192.168.1.0/0 137:139  -d 0/0 1024:65535 -j DENY
# ipchains -A forward -i ul0 -p udp -s 192.168.1.0/0 137:139  -d 0/0 1024:65535 -j DENY
```

画面2　NetBIOSパケットの外部への転送を拒否する

ネットマスクに24ビットを指定し、IPアドレス192.168.1.0～192.168.1.255の範囲でIPマスカレードを利用できるようにする。

3. 発信先はインターネットすべてのアドレスを対象にする。

4. これ以外のパケットの転送は許可しない。

下の例ではLAN内の192.168.1.0～255のIPアドレスを変換して外部と接続する。そしてそれ以外のパケットを転送しないようにする。

```
# ipchains -A forward -s 192.168.1.0/24
-d 0.0.0.0/0 -j MASQ
# ipchains -P forward DENY
```

■IPフィルタリング

ipchainsは細かくパケット転送の許可、拒否を設定することができる。たとえばLAN内部にあるWindowsマシンからNetBIOSプロトコルが発せられた場合に自動的に接続（転送）動作に入ってしまうのを防ぐ場合や、外部から特定のサービス要求を拒否する場合などに利用できる。

・ターゲット

-jに続けてMASQ、ACCEPT、DENY、REDIRECT、REJECT、RETURNを指定できる。

・アドレスの指定

-sでソースアドレス、-dでディスティネーションアドレスを設定する。また-pでプロトコルを指定する。

```
# ipchains -A input -i eth0 -s 192.168.1.0/24 -d 0/0 -j DENY
```

画面3　IPアドレスを偽って外部から入ってくるパケットを拒否する。

・インターフェイス

-iでインターフェイスを指定できる。ppp0、ppxpを利用している場合はul0（userlink）などと指定する。またポート番号はxxx:xxxのようなかたちで範囲を指定する。

これらを簡単に組み合わせると、次の例のようなことができる。

1. userlinkを使用し、LAN内からポート番号137～139のNetBIOSパケットの外部への転送を拒否する（画面2）。

2. 外部からeth0を経由して入ってくるパケットで、発信元アドレスが192.168.1.0～255（IPアドレスを偽り、クラッキング目的での侵入の可能性がある）のパケットを拒否する（画面3）。

設定を終えたら-Lオプションを付けて転送設定を確認しておく。

最後に/sbin/modprobeとipchainsの設定を、/etc/rc.d/rc.localファイルなどに書き加えておく。これでLAN内のほかのコンピュータで、ゲートウェイをLinuxマシンのIPアドレスに設定しておけばIPマスカレード機能を利用できる。

カーネル2.0.xにおけるIPマスカレード

Vine Linux 1.0／ 1.1やRed Hat Linux 5.2のカーネル2.0.36は、あらかじめIPマスカレードを利用できるように作られている。ここではVine Linux 1.1およびRed Hat Linux 5.2をベースに解説する。

■IPフォワード機能の設定

まずIPフォワード機能をオンにする。X Window System上のnetconfigツールを使う方法と、エディタで/etc/sysconfig/networkファイルを直接修正する方法の2通りがあり、いずれの場合もroot権限で実行する。

・「Network Configurator」で設定

X上のコントロールパネルから「Network Configurator」を起動してツールバー「Routing」を押す。一番上の行にある“Network Packet Forwarding（IPv4）”の先頭にあるチェックボックスをクリックして（赤色になる）、設定を「Save」する（**画面4**）。この結果/etc/sysconfig/networkファイルが書き換えられる。

・networkファイルをエディタで修正

/etc/sysconfig/networkファイルの修正の方法は、カーネル2.2.xの場合と同様であるので、前項を参照のこと。

どちらの方法でも、networkファイルを書き換えたら、

```
# /etc/rc.d/init.d/network restart
```

を実行してネットワークを再起動する。その後、catコマンドで/proc/sys/net/ipv4/ip_forwardを見て、“1”が表示されればIPフォワードが有効になって

いる。

■必要なモジュールの組み込み

カーネル2.2のipchainsの場合と同様に/sbin/modprobeコマンドを使って必要なものを組み込んでおく。詳細は前項を参照のこと。

■IP転送の設定

転送設定を行うのはipfwadmコマンドだ。ipfwadmは、

- **Linuxカーネルでの IPパケットのアカウント制御**
- **IP入力ファイアウォール制御**
- **IP出力ファイアウォール制御**
- **IP転送ファイアウォール**

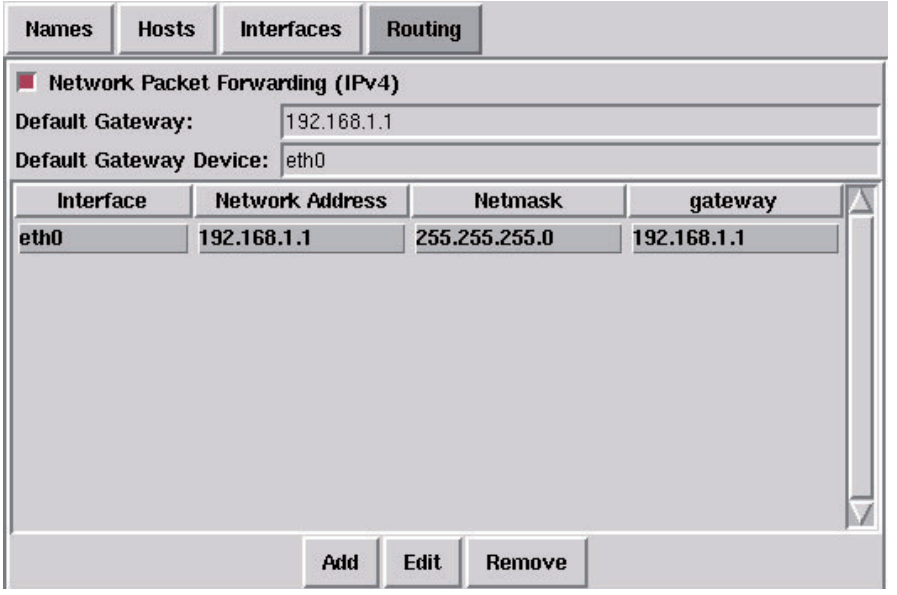
を設定する機能をもつ。それぞれの制御は順に-A、-I、-O、-Fというオプションを付けて条件を設定できる。このほかにもいくつものオプションがあるので、manを実行してオンラインマニュアルを調べてみるとよい。

カーネル2.2.xの場合と同じようにipfwadmコマンドでIPマスカレードの設定を行う。

```
# ipfwadm -F -a accept -m -S 192.
168.1.0/24 -D 0.0.0.0/0
# ipfwadm -F -p deny
```

■IPマスカレード設定の確認

「ipfwadm -F -l」コマンドで正しく設定できたか確認してみよう。LAN内部から、任意のアドレス宛にアクセスできることがわかるだろう。あとはLAN内部のコンピュータでLinuxサーバをゲートウェイに設定すればよい。うまく動作したらモジュールの組み込みとipfwadmコマンドの設定を/etc/rc.d/rc.localに追加すれば、次回Linuxサーバの起動時も有効になる。



画面4 「Network Configurator」を利用してIPフォワードを有効にする

Column

カーネルの再構築の設定

LASER5 Linux 6.0などではあらかじめ設定済みであるが、自分でカーネルを入れ換えている場合には、カーネルのコンフィグレーションで、IPマスカレードを有効にして作り直す必要があるかもしれない。/usr/src/linuxディレクトリで、make menuconfig（X上では、make xconfig）コマンドを実行したあとに、設定する場所を説明しておこう。

IPマスカレードを利用するためにカーネル2.0.xの再構築を行う時には、Networkingオプションでリスト1の項目を［y］にする。同様にカーネル2.2.xの再構築を行う時には、Networkingオプションでリスト2の項目を［y］にする。

▼リスト1

```
Networking support              (CONFIG_NET=y)
TCP/IP networking               (CONFIG_INET=y)
IP:forwarding/gatewaying        (CONFIG_IP_FORWARDING=y)
Network firewalls               (CONFIG_FIREWALL=y)
IP: firewalling                 (CONFIG_IP_FIREWALL=y)
IP: masquerading                (CONFIG_IP_MASQUERADE=y)
IP: ipautofw masquerading       (CONFIG_IP_MASQUERADE_IPAUTOFW=y)
IP: ICMP masquerading           (CONFIG_IP_MASQUERADE_ICMP=y)
IP: always defragment           (CONFIG_IP_ALWAYS_DEFRAG=y)
```

▼リスト2

```
IP:     firewalling
IP:     transparent proxy support
IP:     masquerading
IP:     ICMP masquerading
IP:     masquerading special modules support
IP:     ip fwmark masq-forwarding support (EXPERIMENTAL)
```

Squid　～Webアクセスを快適にしよう～

文：編集部　Text : Linux magazine

Squidは、WebやFTPなどに対応したプロキシサーバソフトウェアだ。プロキシサーバとは、ローカルネットワークとインターネットの間に設置して、2つのネットワークにまたがるデータ転送を仲介するサーバだ。

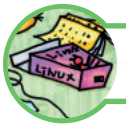
数あるWebプロキシサーバの中でも、Squidはデータのキャッシング性能が高いことで知られており、以前にアクセスしたことのあるサイトに対しては、WebやFTPなどのレスポンスを速くしたり、トラフィックを減少させることができる。

LinuxサーバやISDNルータでIPマスカレードの機能が使えるようになっていれば、LANでつないだどのマシンからもインターネットを利用できるが、これに加えてSquidも導入すれば、さらにストレスのないWebブラウジングが可能になるだろう。

Squidの動作

Squidは、Webなどのデータを転送すると同時に、そのデータをディスクやメモリに保存しておく。再び同じデータを送るようクライアントから要求されると、保存していたデータが更新されていないかをWebサーバやFTPサーバに問い合わせる。更新されていなければ保存していたデータをクライアントに送り出し、更新されていれば、サーバから最新のデータを取得してクライアントに送り出す（**図1**）。

毎回データを転送してくるのに比べ、更新の確認はわずかなデータ量で済むうえ、保存していたデータは、LANで高速に転送できる。インターネットへの接続に、ISDNや電話回線のように低速な回線を利用している場合は、キャッシュにヒットすれば数十倍の速度が得られるのだ。さらに、SquidはDNS（Domain Name System）サーバにアクセスする専用プログラムを持っており、これを複数実行することで名前解決の高速化も図っている。



インストール

Red Hat LinuxやLASER5 Linuxなどでは、ディストリビューションに含まれているので、rpmコマンドなどを使ってCD-ROMからインストールする。

TurboLinux 4.0／4.2の場合は少々面倒だ。バイナリCDにはSquidは収録されていないので、別途入手する必要がある。製品版ではコンパニオンCDに収録されているが、バージョンが古くお勧めできない。また、残念ながらターボリナックス ジャパンのWebサイトにも新しいバージョンは用意されていない。そこで、Red Hat Linux 6.1用のRPMパッケージを流用しよう。このバイナリパッケージはglibc 2.1を必要とするので、そのままTurboLinuxにインストールすることはできないが、

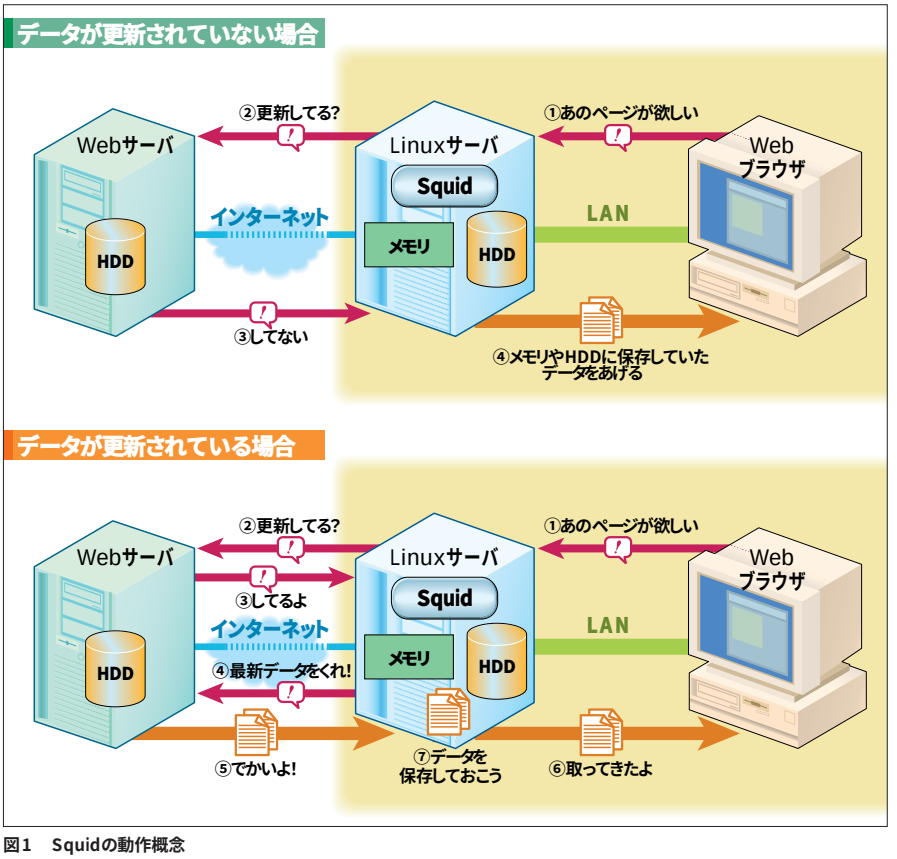
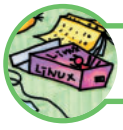


図1　Squidの動作概念

ソースRPMパッケージを入手し、リビルドすればTurboLinuxでも利用することができる。まずは、Ring Server ProjectのFTPサーバからファイルをダウンロードしよう（ftp://ftp.ring.gr.jp/pub/linux/RedHat/redhat/redhat-6.1/SRPMS/SRPMS/squid-2.2.STABLE4-8.src.rpm）。ファイルが手に入ったら、rootユーザになってリビルドする。

```
# rpm --rebuild squid-2.2.STABLE4-8.src.rpm
```

これで、/usr/src/turbo/RPMS/i386ディレクトリにバイナリパッケージが作られるので、これをインストールする。最新版は、2000年1月に発表された2.3 STABLE1だが、2.2 STABLEならまず問題ないだろう。



Squidの設定

Squidの設定は、/etc/squid/squid.confというファイルで行う。RPMパッケージではなく、オリジナルのtarボールからビルドした場合は、/usr/local/squid/etc/squid.confとなる。

このファイルでは、“＃”で始まる行はコメントとして無視される。各設定項目の詳しい説明とデフォルト設定がコメントとして書かれているので、かなり長いファイルになっている。通常は、アクセス制限を変更するだけで十分だろう。

RPMパッケージをインストールした状態ではlocalhost、すなわちLinuxサーバ自身からだけ利用できるようになっている。これでは導入した意味がないので、ローカルネットワークにあるマシンからも利用できるようにしよう。

/etc/squid/squid.confをエディタで開き、“ACCESS CONTROLS”セクションにある“acl CONNECT method CONNECT”と書かれた次の行に、

```
acl homenet src 192.168.1.0/255.255.255.0
```

と1行で書いて追加する。この例では、ローカルネットワークで192.168.1.0～192.168.1.255のIPアドレスを利用しているものとして、この範囲内のIPアドレスをhomenetという名前のアクセスリストとして定義している。次に、“http_access deny all”と書かれた行の前に、

```
http_access allow homenet
```

という行を追加する。これにより、先ほど定義したhomenetに属するマシンからのアクセスを許可する。

これでSquidを起動すれば、すぐに利用できるが、好みや環境に合わせてカスタマイズすることもできる。比較的よく使われるパラメータをいくつか紹介しておこう。

http_port 3128

Squidに接続するためのポートを指定する。3128のほか、8080や10080がよく使われる。

cache_mem 8 MB

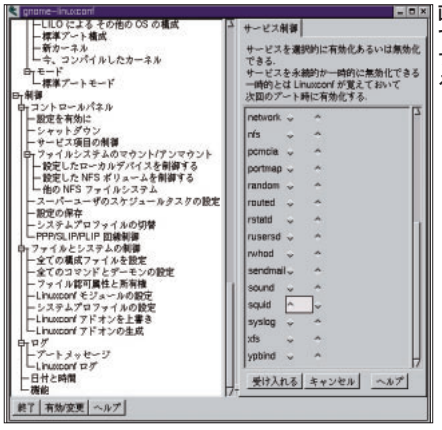
キャッシュに割り当てるメモリの量を指定する。搭載しているメモリに余裕があれば、この数字を大きくするとパフォーマンスが上がる。

cache_dir /var/spool/squid 100 16 256

キャッシュデータを保存するディレクトリと容量などを指定する。デフォルトでは、100Mバイトを割り当て、16個のサブディレクトリの下に256個ずつ

▼リスト1　/etc/squid/squid.confの例（コメント行はすべて省略した）

```
http_port 8080                          ← Squidのポートを8080にする

cache_mem  16 MB                        ← キャッシュに16Mバイトのメモリを割り当てる

cache_dir /var/spool/squid 200 16 256   ← キャッシュに200Mバイトのディスクを割り当てる

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 80 21 443 563 70 210 1025-65535
acl CONNECT method CONNECT
acl homenet src 192.168.1.0/255.255.255.0   ← ローカルネットワークをhomenetという名前のアクセスリストとして定義する

http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access allow homenet               ← ローカルネットワーク内からのアクセスを許可する
http_access deny all

icp_access allow all
miss_access allow all
```



画面1　Linuxconfでsquidを自動起動するように設定する。

画面2　TurboLinuxでは、turboserviceでSquidの自動起動を指定。

のサブディレクトリを作ってデータを保存する。

これらのパラメータを変更した/etc/squid/squid.confの例が**リスト1**だ。

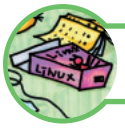
Squidの起動

Squidの設定が終わったら早速起動しよう。RPMパッケージからインストールしていれば、/etc/rc.d/init.d/squidという起動スクリプトが作られているので、

```
# /etc/rc.d/init.d/squid start
```

とすればSquidが起動する。初回起動時は/var/spool/squidディレクトリにキャッシュディレクトリを作るのでしばらく時間がかかる。

Linuxの起動とともにSquidも動作させるようにするためには、Linuxconf（**画面1**）、ntsysv、turboservice（**画面2**）などのツールを使うと簡単だ。



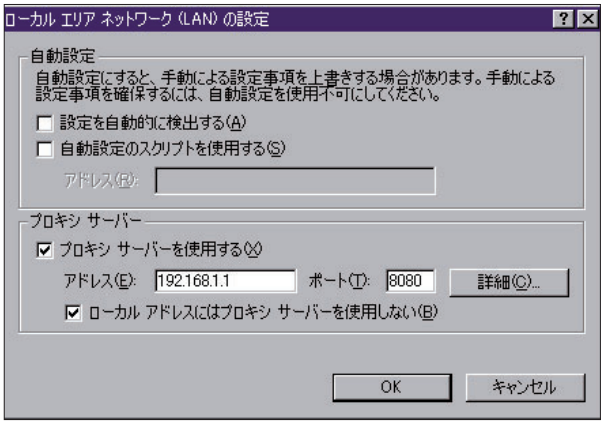
Webブラウザの設定

Squidを利用するためには、Webブラウザの設定でプロキシサーバを使うようにする必要がある。例として、Squidが動作しているLinuxサーバのIPアドレスを192.168.1.1、Squidのポート番号を8080とする。

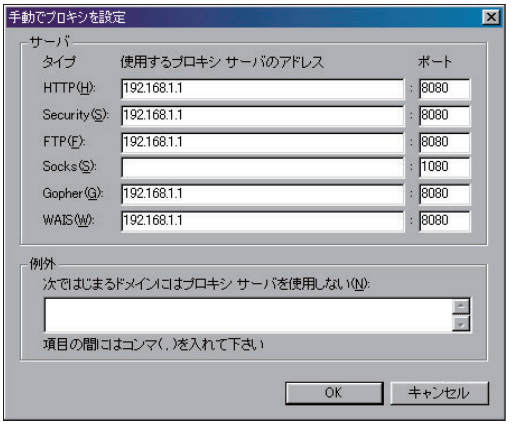
Microsoft Internet Explorer 5.0では、［ツール］メニューの［インターネット オプション］を開き、［接続］タブを選んだあと、［LANの設定］ボタンを押す。すると**画面3**のダイアログが開くので、「プロキシ サーバー」の［プロキシ サーバーを使用する］チェックボックスにチェックを入れて、［アドレス］にLinuxサーバのIPアドレス192.168.1.1を、［ポート］にSquidのポート番号8080を指定する。

Netscape Navigatorの場合は、［編集］メニューから［設定］を選び、開いたウィンドウの「カテゴリ」ツリーの詳細を展開して［プロキシ］を選ぶ。ウィンドウの右側で［手動でプロキシを設定する］ラジオボタンにチェックして、［表示］ボタンを押して現れた画面で**画面4**のように設定する。

データを蓄積してキャッシュにヒットするようになるまでにしばらく時間がかかるが、徐々に効果が現れるので気にせず使おう。Squidでキャッシュしたデータは、利用者全員で共用するため、複数のユーザーが同じサイトを見る機会が多い場合は効き目が現れるのも早くなる。



画面3　Microsoft Internet Explorer 5.0のプロキシ設定ウィンドウ。



画面4　Netscape Navigatorはプロトコルごとにプロキシを設定する。

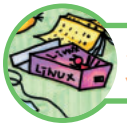
Samba　～Windowsのファイルを共有しよう～

文：山岸典将　*Text : Norimasa Yamagishi*

SambaはLinuxマシンにあるファイルやプリンタをWindowsから共有したり、LinuxからWindowsの共有ファイルフォルダやプリンタにアクセスするためのソフトウェアだ（**画面1**）。また、そのほかに、Windowsドメインのサーバとなり、パスワードの管理などを行うこともできる。

Windowsネットワークでは、NetBIOSというプロトコルを利用して通信をしている。このNetBIOSプロトコルはもともとNetBEUIという下位プロトコルを利用していたが、Windows 95／98／NTでは、NetBIOS over TCP/IPという技術により、TCP/IPを下位プロトコルとして利用することができる。そして、Windowsのファイル共有は、さらにNetBIOS上のSMBプロトコルによって行われている。

Sambaは、オーストラリアのAndrew Tridgell氏らによって開発された。Linux（UNIX）マシン上でNetBIOS over TCP/IPを利用し、サーバとしてWindowsの名前解決をするためのnmbdと、リソースを供給するためのsmbd、そしてクライアントとしてのsmbclientというプログラムから構成されている。



Sambaのインストール

最近のディストリビューションにはSambaも含まれていることが多いのだが、オリジナルのSambaは、日本語の運用については考慮されていない。これに対し、日本語のコンピュータ名や共有名、Samba管理ツールSWATからの日本語入力などに対応した日本語版を、たかはしもとのぶ氏らがリリースしているので、今回は、そちらを使ってみよう。

日本Sambaユーザ会（http://www.samba.gr.jp/）のWebページでは、コンパイル済みのバイナリファイルがRPMパッケージとして提供されている。現在提供されているディストリビューションは、Red Hat Linux　5.2／6.0／6.1、Laser5 Linux 6.0、Vine Linux 1.0／1.1、TurboLinux 4.0／4.2などだ。

これ以外のディストリビューションを使っていたり、自分の好みのディレクトリにインストールしたい場合には、自分でソースからビルドする必要がある。

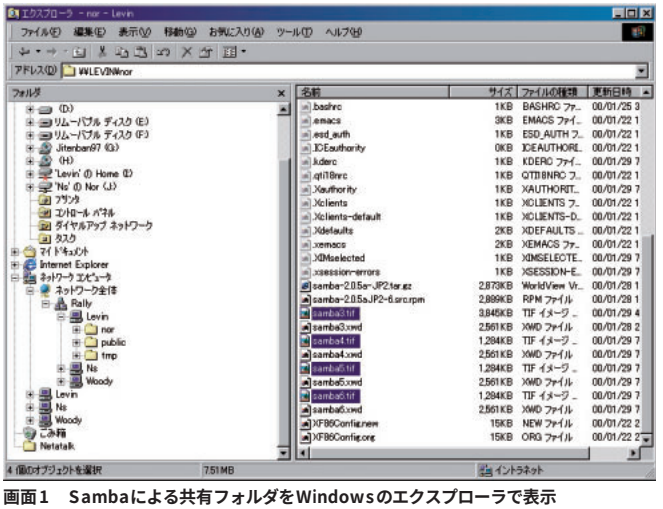
RPMパッケージが用意されている場合は、以下のファイルを入手してインストールする。

samba-common.2.0.5aJP2-6.i386.rpm
samba-2.0.5aJP2-6.i386.rpm
samba-client.2.0.5aJP2-6.i386.rpm

ソースからビルドしてインストールした場合と、RPMパッケージでインストールした場合では、それぞれ、Sambaに関係するファイルがインストールされる場所が異なるので注意が必要だ。その中でも特に重要なのが、Sambaの設定ファイルであるsmb.confの位置と、パスワードファイルsmbpasswdの位置だ（**表1**）。

Windowsマシンでの準備

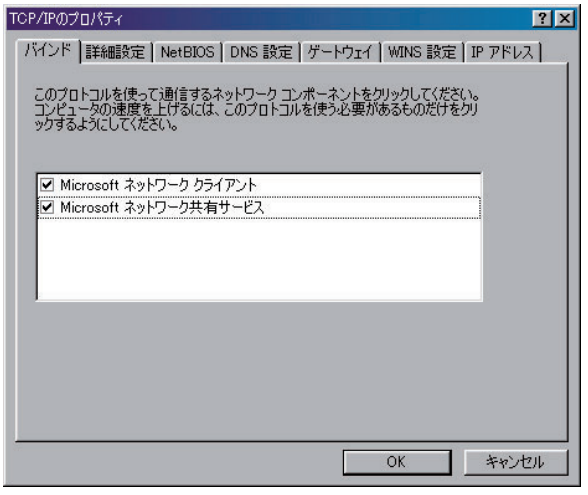
先ほど述べたように、SambaはNetBIOS Over TCP/IPを利用して、Windowsネットワークと通信している。そのため、Windowsマシンでも、Net



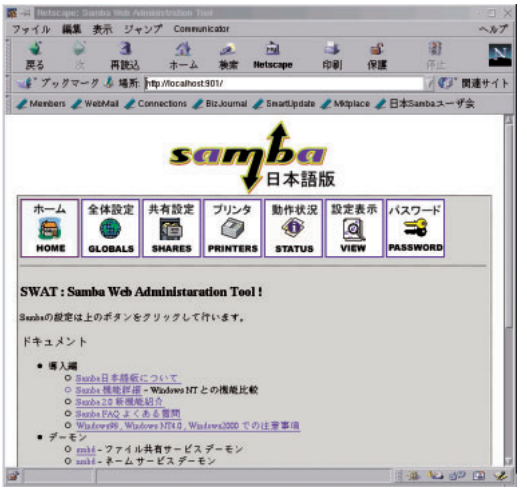
画面1　Sambaによる共有フォルダをWindowsのエクスプローラで表示

ファイル	RPMからインストール	ソースからビルドしてインストール
smb.conf	/etc	/usr/local/samba/lib
smbpasswd	/etc	/usr/local/samba/private/

表1　Sambaのファイルのインストール場所



画面2　Windows 98でのNetBIOS over TCP/IPの設定。

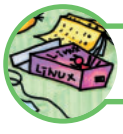


画面3　SWATのトップページ。

BIOSをTCP/IP上で利用できるように設定しておく必要がある。

Windows 95は、インストールしたままの状態だと、TCP/IP上でNetBIOSが使えるようになっていないので、ネットワークの設定をしないとSambaが使えない。また、Windows 98／NTでもTCP/IPを無効にしている場合は、Sambaを利用することはできない。

TCP/IPが有効になっているかどうかの確認は、［コントロールパネル］の［ネットワーク］、［ネットワークの設定］の「現在のネットワーク コンポーネント」の中で、TCP/IPと自分の使うネットワークボードが結ばれているか、また、［TCP/IPのプロパティ］の［バインド］タグで、［Microsoft　ネットワーククライアント］がチェックされているかどうかで行う（**画面2**）。



Sambaの設定

Sambaの設定は、smb.confというファイルを書くことにより行うが、SWATというWebベースのGUIツールも用意されている。ただし、基本的にSWATはsmb.confを書き換えるためのフロントエンドにすぎない。したがって、SWATでの設定ができれば、smb.confの書き換えも特に問題なくできるだろう。

また、SWATはよくできたツールだが、ネットワーク上をパスワードが流れるなど、セキュリティ上の問題があるので、外部に直接につながっているネットワークでは使わないほうがよいだろう。その場合は、エディタを使って直接smb.confを書くことになる。

SWATを利用するためには、ネットワークの設定を変更する必要がある。RPMパッケージをインストールした場合は、/etc/inetd.confの中から#swatで始まる行を探し、先頭にあるコメントマークの#を削除する。ソースからビルドした場合は、/etc/inetd.confと/etc/servicesに**リスト1**の行を追加する。

/etc/inetd.confを書き換えたらinetデーモンを再起動するか、Linuxをリブートする。

SWATを利用するには、Linuxサーバ上のWebブラウザで、http://localhost:901/というURLを指定し、ユーザー認証のダイアログに対してrootアカウントとそのパスワードを入力する。するとSWATの画面が現れる（**画面3**）。この画面から上部のメニューをクリックし設定を行うことになる。

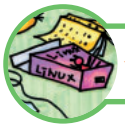
ここではsmb.confを書き換える方法を説明し、SWATではその設定がどの部分にあるかを解説する。もっとも、RPMパッケージでインストールした場合、デフォルトのsmb.confは作られない（ソースからビルドした場合は、サンプルのファイルが作成されるので、それを利用すればよい）。一度SWATを起動すると、基本的なsmb.confが作られるので、それをひな形として使用するのもよいだろう。

smb.confは、複数のセクションと、その詳細設定パラメータに分かれている。その中でも、特殊なセクションとして、［global］、［homes］、［printer］の3つのセクションがある。［global］は全体の動作の設定、［homes］はLinuxマシンにアカウントのあるユーザーのホームディレクトリの設定、

▼リスト1　ソースからビルドした場合は/etc/inetd.confと/etc/servicesに次の行を追加する

```
/etc/inetd.confに追加
swat   stream   tcp    nowait.400   root /usr/local/samba/bin/swat swat

/etc/servicesに追加
swat              901/tcp
```

［printer］はプリンタの設定だ。セクションは“［］”（大カッコ）でくくり、各パラメータとその内容は“＝”で結んで記述する。簡単なsmb.conf（**リスト2**）を例に解説しよう。



全体の設定

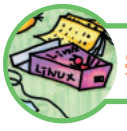
［global］セクションはすべてのセクションで共通に有効になるパラメータを指定する。

まず、ワークグループ名もしくはWindows NTのドメイン名、そしてSambaの動作しているマシンの名前などを設定しよう。smb.confでは、［global］セクションの次の項目で設定する。

`workgroup`	**ワークグループ名**
`netbios name`	**Windowsネットワークでのマシン名**
`server string`	**Windowsのエクスプローラで表示されるマシンの説明**

このほかに、ゲストアカウントを許可する場合は“guest account”パラメータに、ゲストとしてログインするときのユーザーアカウントを指定する。これらの項目は、SWATでは「全体設定」の中に存在する。

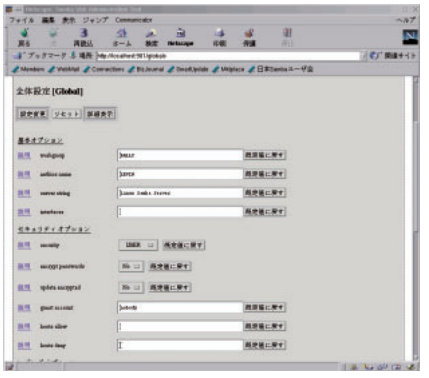
なお、［global］セクションで指定したパラメータと同じパラメータが、あるセクションで再び指定されていた場合は、各セクションで指定したパラメータが有効になる（**画面4**）。



共有フォルダの設定

次に、共有するフォルダの設定をしよう。smb.confでは、共有フォルダごとにセクションを作成することになる。ただし、特殊な共有フォルダとして［homes］がある。

［homes］セクションでは、ユーザー自身のLinux上のホームディレクトリへのアクセスを可能にする。そのため、個別に共有フォルダ指定する必要はない。［homes］で特に指定しておきたいパラメータは、以下のものだ。



画面4　SWATによる全体設定

`read only`	**noを指定しないと、書き込みができない**

それ以外の便利な設定として“browseable”パラメータを“no”にしておくとよい。このパラメータを指定すると、Windowsエクスプローラで見た場合に、「homes」と「ユーザー名」の両方に同じフォルダが表示されることを避けられる。

［homes］以外の共有フォルダを作成したい場合は、そのフォルダの名前のセクションを作成することになる。たとえば、次のようにすると/tmpディレクトリを共有する。

```
[tmp]
        comment = temp directory
        path = /tmp
        guest ok = yes
        read only = no
```

各項目の意味は次の通りだ。

`path`	**Linux上の共有するディレクトリの指定。**
`guest ok`	**yesにすると、アクセスするときにパスワードが不要になる。**
`read only`	**noにすると、書き込みが可能**

▼リスト2　SWATで作成したsmb.confの例

```
[global]
        workgroup = RALLY
        netbios name = LEVIN
        server string = Linux Samba Server
        log file = /usr/local/samba/var/log.%m
        max log size = 50
        socket options = TCP_NODELAY
        coding system = cap
        dns proxy = No
        guest account = nobody
[homes]
        comment = Home Directories
        read only = No
        browseable = No
[printers]
        comment = All Printers
        path = /var/spool/samba
        print ok = yes
[tmp]
        comment = temp directory
        path = /tmp
        guest ok = yes
        read only = no
```

になる。yesなら、読み込みのみ可能。

“ guest ok ”と同じ働きをするパラメータに“ public ”がある。また、“ writable ”は“ read only ”と反対の働きをするパラメータだ。これらの項目は、SWATでは「共有設定」の中に存在する（**画面5**）。

また、共有フォルダごとに使用できるユーザーを制限することができる。“ invalid users ”パラメータにユーザーを指定した場合、そのユーザーは確実にアクセスできなくなる。また、逆に“ valid users ”パラメータにユーザーを指定した場合、指定されているユーザーしかアクセスできなくなる。

共有プリンタの設定

Linuxマシンのプリンタを共有プリンタとして、Windowsから使うことができる。もちろんWindows 95／98でもローカルプリンタの共有はできるが、プリンタを共有しているマシンは電源を切ることができなくなるし、また大人数でプリンタサーバとして利用することはライセンス上の問題もある。プリンタの共有はサーバマシン上で行いたい。

共有プリンタの設定は［printers］セクションで次のように行う。

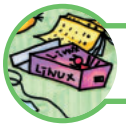
```
[printers]
        comment = All Printers
        path = /var/spool/samba
        print ok = yes
```

各項目の意味は次の通り。

`path` **Linuxのプリントスプールディレクトリを指定する。**

`print ok` **yesにすると、pathで指定したパスにスプールファイルを書き込む。**

共有フォルダと同様に、使用できるユーザーの制限も可能だ。SWATでは、これらの項目は「プリンタ設定」に存在する（**画面6**）。



アカウントの設定

Sambaはもともとは、Linuxのユーザーアカウントを利用して認証を行っていたが、Windows 98／NT 4.0では、パスワードをネットワークで送信する際に暗号化するようになったため、Sambaでもアクセス用に暗号化したパスワードファイルを別途作成する必要がある。

初期のWindows 95を使用している場合には、Sambaのユーザー認証はLinuxのアカウント情報をそのまま使用するので、ユーザーアカウントをLinux上に作ればよい。

暗号化パスワードを使用している場合は、Windows側でパスワードを暗号化しないで送る方法と、Samba側で暗号化パスワードを利用できるようにする方法がある。当然、セキュリティ的には、暗号化パスワードを使用した方が良いので、ここではそちらの方法を説明する。

1. smb.confの変更

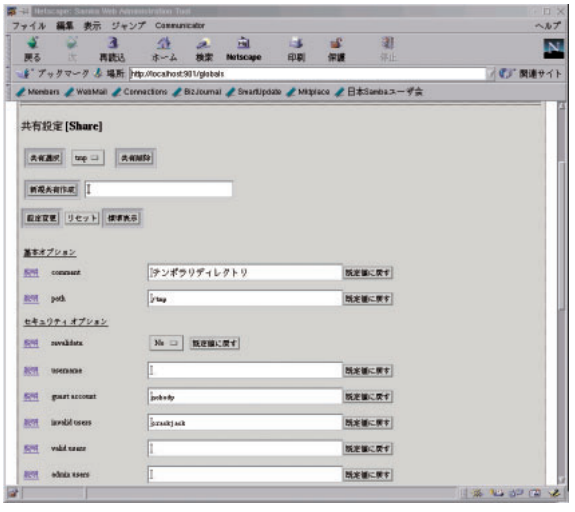
smb.confの［global］セクションに以下の指定を行う。

```
security = user
encrypt passwords = yes
```

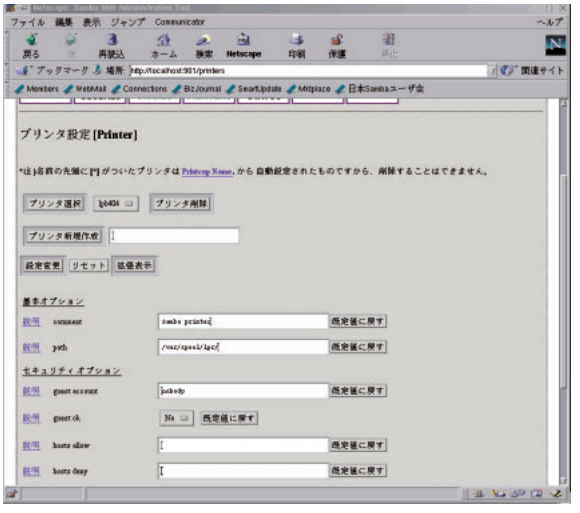
これらの項目は、SWATでは「全体設定」の中に存在する。

2. パスワードファイルの作成

パスワードファイルをLinuxのユー



画面5　SWATによる共有設定



画面6　SWATによるプリンタ設定

ザー定義ファイルから作成するスクリプトが用意されているので、それを利用し、パスワードファイルを作成する。その後、パスワードファイルをスーパーユーザー以外がアクセスできないようにする。

```
# cat /etc/passwd | mksmbpasswd.sh> /etc/smbpasswd
# chmod 600 /etc/smbpasswd
```

3.ユーザーの新規登録

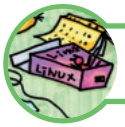
以下の［password］の部分は、パスワードを入力する。kenchanは、新しく登録するユーザー名。

```
# smbpasswd -e kenchan
New SMB password: [password]
Retype new SMB password: [password]
User kenchan enabled.
```

また、Windows NTが存在するならば、Windows NTのユーザー認証を利用する方法もある。ドメイン名がlinuxmagでドメインコントローラとしてlm1、lm2が存在するのであれば、smb.confの［global］セクションを以下のように設定する。

```
security = server
workgroup = linuxmag
password server = lm1 lm2
```

これらの項目は、SWATでは「全体設定」の中に存在する。ただし、“password server”に関しては、「詳細表示」をしないと、表示されない。



Sambaの起動

Sambaのファイルをインストールし、設定が終わっても、Linuxが立ち上がると自動的にSambaサーバが立ち上がるわけではない。

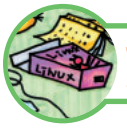
とりあえず、起動するには、スーパーユーザー（root）になって、コマンドラインでサーバプログラムを実行する

```
# nmbd -D
# smbd -D
```

しかし、実際にSambaが動作することが確認できたら、Linuxが動いているときは常にSambaを使える状態にしておきたい。

RPMパッケージからインストールした場合は、Linuxconf、ntsysv、turboserviceなどでLinuxとともにSambaも起動するようにしておこう。

一方、ソースコードからビルドした場合は、/etc/rc.localの中に起動するコマンドを書いておけば、Linuxとともに自動起動するようになる。

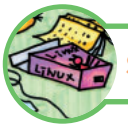


そのほかのSambaの機能

このほかにも、SambaにはLinuxマシンからWindowsの共有ファイルにアクセスしたり、共有プリンタに印刷するクライアント機能がある。ただし、LinuxからWindowsのプリンタに印刷する場合には、プリンタドライバがLinux側に必要となる。すなわちLinuxからWindowsのプリンタドライバを使って印刷することはできない。

また、Windowsドメインのメンバサーバとなったり、WINSサーバとして動作させることもできる。さらに、Windows 95／98のためのドメインコントローラとして動作することもできる。

このほか、Windows95／98に対しては、Sambaサーバからプリンタドライバの自動インストールをすることも可能だ。詳しくは、付属のドキュメントを調べてみてほしい。



Sambaの問題点

最後に、いくつか注意しておきたい点をあげておこう。

まず、Linux上のファイル名は、同じスペルで大文字／小文字が違うものを作成できるが、そのディレクトリにSambaでアクセスしたときには、正常な動作をしない。たとえば、Linux上では“Mail”というファイルと“mail”というファイルを、同じディレクトリに置くことができる。しかし、Windowsでは、表示上は大文字／小文字を区別してファイル名を付けられるが、実際にはどちらも同じファイルとみなされ、同じディレクトリに作成することはできない。そのため、もしLinux上でこういったファイルを両方作り、Windowsからアクセスしようとすると、実際にあるファイルにアクセスできなくなってしまう。

また、ファイルに対するアクセス権もWindows NTのほうがSambaよりも細かく設定できるので、ファイル単位のアクセス権の設定をしたい場合には少々物足りないかもしれない。具体的には、Windows NTではファイルのアクセス権を、任意のユーザーや、グループに対して自由に設定できるが、Linuxではそのファイルの所有者に対するアクセス権、属するグループに対するアクセス権、全員に対するアクセス権の3つしか設定できない。そのため、Sambaによる共有ファイルにもその制限が存在する。

もっとも、このような問題点はSambaを使う利便性から見れば些細なことだろう。

Netatalk～Macintoshとのファイル／プリンタ共有～

文：清水正人　*Text : Masato Shimizu*

ネットワーク内にMacintoshがある場合には、簡単な設定でLinuxをMacintosh用ファイルサーバ、プリントサーバに仕立てることが可能だ。UNIX系OSとMacintoshの間でファイルを共有するためにはいくつかの方法があるが、Linuxではここで紹介するNetatalkを利用するのが一般的に行われている方法だ。

Netatalkのインストール

Netatalkの本体はAFPD（Apple Talk Filing Protocol Daemon）で、これがファイル共有をつかさどる。同時に起動されるPAPDはプリンタ共有を実現するためのデーモンプロセスだ。

MacintoshではAppleShareというAppleTalkを利用したファイル共有機能を利用している。そしてNetatalkはカーネルレベルでMacintoshのネットワークプロトコルであるAppleTalkをサポートするプログラムだ。

よって前提条件としてAppleTalkがカーネルレベルでサポートされている必要がある。Vine Linux、Turbo Linux、Red Hat Linux 5.x以降、LASER5 Linuxなど最近のディストリビューションではAppleTalkサポートがカーネルに組み込まれている。そうでない場合には、カーネルの再構築時のメニュー「Networking options」で「Appletalk DDP」を選択してカーネルの再構築をする必要がある。

Macintoshでは MacOS 8.1以降「AppleShare IP」というTCP/IPを利用した共有サービスを利用している。これに対応するためNetatalkに対してパッチ当てが継続されており、簡単な設定でAppleShare IP対応のサーバをLinux上で構築することができる。

Netatalkの最新版、Adrian Sun氏のパッチが当てられたものをインストールしよう。RPMパッケージとしてはglibc用ではnetatalk-1.4b2+asun2.1.3-6.i386.rpmが原稿執筆時点での最新版で、libc5ユーザーはnetatalk-1.4b2+asun2.1.0-5.i386.rpmを使うか、

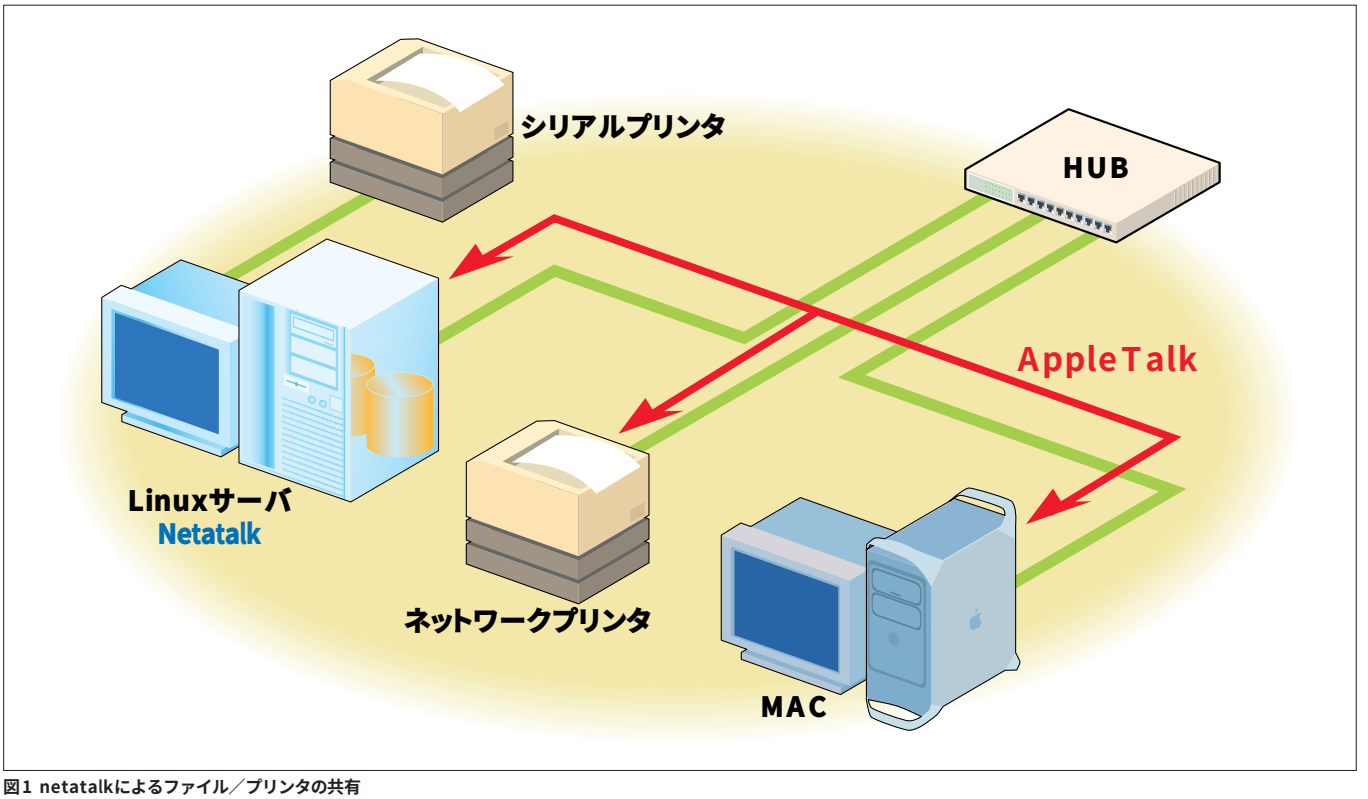


図1 netatalkによるファイル／プリンタの共有

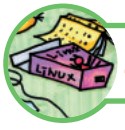
あるいはsrc.rpmやtar.gzからビルドすることになる。

Adrian Sun氏のサイト（http://thehamptons.com/anders/netatalk/）から、オリジナル（ftp://ftp.u.washington.edu/public/asun/netatalk-1.4b2+asun2.1.3.tar.gz）をダウンロードできる。

ここではRed Hat LinuxにRPMパッケージをインストールして設定する方法を紹介しよう。

```
# rpm -ihv netatalk-1.4b2+asun2.1.3-6.i386.rpm
```

以上でインストールは終了する。



Netatalk 設定ファイルの編集

設定ファイルは/etc/atalk以下に配置される。それぞれの設定ファイルの内容を説明する。

■config

AppleShare関連の基本設定ファイルである。このファイルは特に必要がないかぎり変更の必要はないだろう。最大クライアント数（デフォルトは5）、Macintoshクライアントのセレクタやネットワークブラウザに現れるサーバ名（デフォルトはLinuxマシンのホスト名）、AFPD・PAPDデーモンを起動するかどうか（デフォルトでは両方を起動）を指定するファイルだ。Netatalkは起動時にゾーンをサーチするために1分程度の時間がかかるので、デフォルトのバックグラウンド動作のままにしておくとよい。

■AppleVolumes.system

ファイルマッピングの設定を行う。lessなどで表示してみるとわかるが、ファイル拡張子、ファイルタイプ、クリエータ（作成アプリケーション）、説明が列記されている。この拡張子とクリエータの対応により、Macintosh側からブラウズした時にどのアプリケーション・アイコンで表示されるのかが決まる。一般には編集の必要はないが、ある拡張子のファイルを特定のMacOS上のアプリケーションに対応づけたい場合には、編集作業が必要になる。

■AppleVolumes.default

AFPD起動時に、各ユーザーのホームディレクトリにAppleVolumes（またはAppleVolumes）というファイルが存在すれば優先的に読み込まれ、それらが存在しなければ/etc/atalk/AppleVolumes.defaultファイルが読み込まれて共有接続するユーザー特定の環境設定となる。すべてのユーザーに共通の共有項目はAppleVolumes.defaultファイルで設定する。

viエディタを使って開いてしまうとわからないので、ターミナル上からlessなどのコマンドで表示してみると、コメント行が並び最後にひとつだけ「~」（チルダ）が書かれている。すぐ上にあるコメント文を読むとわかるように、チルダはLinuxにアカウントを持つユーザーのホームディレクトリを意味する。

たとえばデフォルトの設定ではNetatalkを起動してMacintosh側からLinuxサーバにアクセスする場合、アカウントを持つユーザーblackがMacOS側からアカウント名とパスワードを入力すると、blackというディレクトリ（実体は/home/black）がMacOSのデスクトップにマウントされるのだ。

■共有ディレクトリの設定

AppleVolumes.defaultではこのほか一般に公開するディレクトリ、グループのみに公開するディレクトリなども設定できる。ここでは例として次のような構成に沿って共有ディレクトリを設定してみよう。

1. **アカウントをもつユーザー個々のディレクトリ→/home/各ホームディレクトリとする。**
2. **LAN内部で公開するディレクトリ→/home/ftp/publicとする。**
3. **LAN内部の特定のグループ（workとする）にのみ公開するディレクトリ→/home/work1とする。**

あらかじめそれぞれのディレクトリのパーミッションを設定しておくことを忘れないように。AppleVolumes.defaultの最後に、次のように追加編集する。

```
~                       HOME
/home/ftp/public        PUBLIC
/home/work1             WORK
access=@work
```

HOME、PUBLIC、WORKはデスクトップにマウントされるアイコンに付けられる名称で、実体はそれぞれ各自のホームディレクトリ（たとえば/home/blackなど）、/home/ftp/public、そして/home/work1となる。

WORKの後に「access=@」に続けてアクセス権を持つグループを指定している。@を付けるとグループになり、単に「access=」とすると特定のユーザーになる。

このグループを有効にするには、あらかじめworkグループを作成し、グループに属するユーザーを指定（/etc/groupファイルにて）しておく必要がある。また/home/work1ディレクトリはグループworkが必要な作業をできる

ように読み書きなどの属性を付けておく必要もある。もしこの方法が面倒であれば、Netatalk専用のユーザーを作成して、ディレクトリの属性を設定しておくほうが楽だろう。

Netatalkの起動

RPMパッケージで導入される/etc/rc.d/init.d/atalkスクリプトをstartオプションを付けて実行する。

```
# /etc/rc.d/init.d/atalk start
Starting AppleTalk services:
(backgrounded)
```

実際にAFPDが起動するのには30秒～1分程度かかるが、バックグラウンドで処理されるためすぐにプロンプトが戻ってくる。少々待ってから、MacOS側から動作確認をしてみよう。

■MacOS側からのログイン（マウント）

まずMacintosh側ではコントロールパネルでAppleTalkの経由先を「Ethernet」側に設定しておく。

次に「アップルメニュー」の「セレクタ」から「AppleShare」を選択する。現れるウィンドウ右側にLinuxサーバ（ここではcat2）が表示される（**画面1**）。もし表示されない場合には、NetatalkはAppleShare IP対応なのでウィンドウ下にある「サーバのIPアドレス」ボタンを押して直接IPアドレスを入力するか、あるいはDNSでサーバのアドレスが登録済みであればFQDN（**画面2**のようにホスト名.ドメイン名）を入力してもよい。

Netatalkサーバ（ここではcat2）を選択すると、**画面3**のようにユーザーアカウントとパスワード入力画面になるので、正しく入力する。

ユーザー名とパスワードの認証が通ると、続けて**画面4**のように/etc/atalk/AppleVolumes.defaultで設定したディレクトリが表示される。マウントしたい共有ディレクトリ名にチェックを入れて「OK」ボタンを押すとMacOSのデスクトップ上にマウントされる。すべてをマウントした例が**画面5**だ。それぞれ独立した地球儀アイコンで表示される。

■MacOSのネットワークブラウザからのログイン

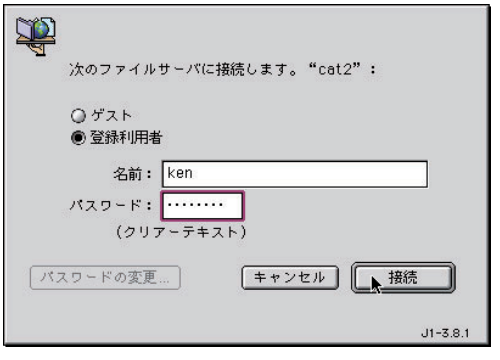
MacOS 8.5などでは、よりTCP/IPへの対応強化が進み「ネットワークブラウザ」というアプリケーションが用意されている。こちらからもNetatalkサーバを開く（マウントする）ことができる（**画面6**）。

「アップルメニュー」の「セレクタ」から「ネットワークブラウザ」を選択する。

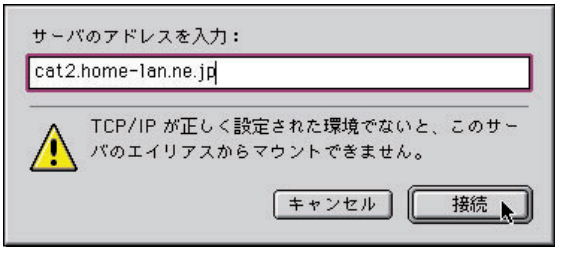
「セレクタ」と同じように「AppleShare IP」に対応したTCP/IPによるLAN内にあるサーバが表示される。アイコン左側の三角タブをクリックすると「セレクタ」と同じようにユーザー名、パスワード入力画面になる。あとは同じように共有ディレクトリをオープンして利用することができる。

また選択した共有ディレクトリは

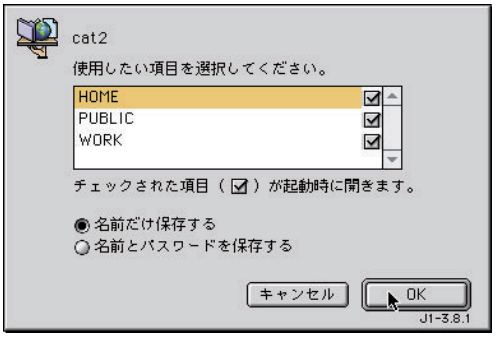
画面1　セレクタでAppleShareを選択するとファイルサーバが表示される。



画面3　Linuxサーバにアカウントを持つユーザー名、パスワードを入力する。



画面2　DNSが設定されている場合にはFQDNを入力することも可能。



画面4　デスクトップにマウントする共有名を選択。

「よく使う項目」として登録することもできる。

■ファイル拡張子とアイコンのチェック

Netatalkによる共有ディレクトリ内のファイルの見え方（アイコン表示）をチェックしてみよう。たとえば/home/ftp/public（共有名PUBLIC）内で適当な拡張子のファイルを作成し、それをさらに別の拡張子のファイルとしていくつかコピーし、MacOS側からのアイコン表示を確認してみよう。

```
$ cd /home/ftp/public
$ echo test > test.txt
$ cp test.txt test.doc
$ cp test.txt test.psd
```

このようにファイルをコピーした後で、デスクトップにマウントされた「PUBLIC」を開いてみる。**画面7**のように/etc/atalk/AppleVolumes.systemで記述された拡張子とクリエータに対応したアイコンで表示されることがわかる。

■プリンタの共有

Netatalkはファイルだけでなく、プリンタも共有することができる。プリンタの設定は/etc/atalk/papd.confファイルで行う。書式はLinuxでのプリンタ設定を行う/etc/printcapファイルとほぼ同じで、プリンタ名を記述し、prとしてlpを指定する。

/etc/atalk/papd.conf例

```
Linux-BJ-Printer:\
        :pr=lp:
```

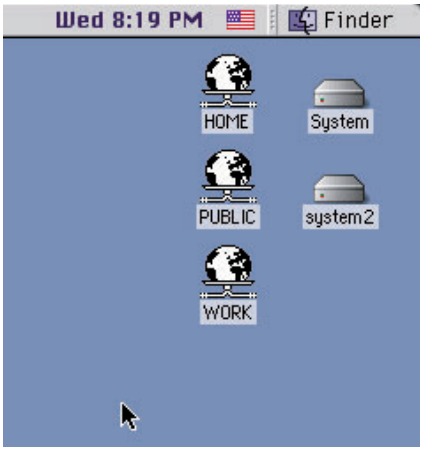
設定を終えたら、次のようにしてNetatalkを再起動する。

```
# /etc/rc.d/init.d/atalk restart
```

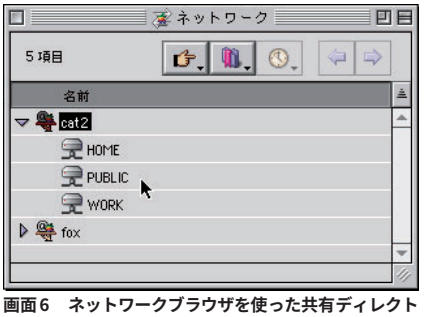
これで設定は終了だが、Linux側に接続・利用でき、かつMacOS用に用意されているプリンタドライバの種類には限りがあるので、実際にはPostScriptプリンタとして利用するのが最も現実的だろう。

MacOSのセレクタでプリンタが表示されるかどうか確かめる。プリンタが現れたら、プリンタ記述（PPD）ファイル選択ウィンドウで「一般設定を使用」を選ぶ。

Linuxサーバ側で日本語Ghostscript、日本語TrueTypeフォントが利用できるように設定しておくと、MacOS側からEPSファイルをプリンタアイコンにドロップするだけで印刷できるようになる。ただし、印刷するテキストはLinux側で理解できる漢字コード、すなわちEUCに変換しておく必要がある。

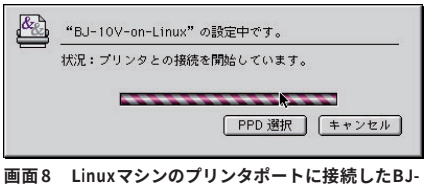


画面5　AppleVolumes.defaultでの設定例による共有ディレクトリ3つをがデスクトップにマウントされた。



画面6　ネットワークブラウザを使った共有ディレクトリのオープン。

画面7　中身は同じファイルだが、拡張子とAppleVolumes.system内で記述されているアプリケーション・クリエータの対応づけにしたがって、ファイルがアプリケーション・アイコンで表示される。



画面8　Linuxマシンのプリンタポートに接続したBJ-10VをMacintosh側から使うために、MacOSのセレクタで選択後プリンタ設定を行う。

画面9　MacOSのデスクトップにLinuxサーバのBJ-10Vがマウントされた。

fetchmail ～プロバイダからメールを自動取得する～

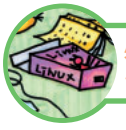
文：編集部 Text : Linux magazine

fetchmailは、プロバイダなどから定期的にメールを取得するフリーソフトウェアだ。これを使えば、プロバイダのメールボックスがあふれるのを防いだり、複数のメールアカウントを一元管理することができる。

fetchmailの動作概要

最初に、fetchmailの一般的な動作を説明しておこう。

fetchmailは、POP3（Post Office Protocol 3）などのメール転送プロトコルでプロバイダのメールサーバからメールを取得し、そのメールをローカルサーバで動作するsendmailのようなMTA（Message Transfer Agent）に渡す。そのLinuxサーバでPOP3デーモンを動かしておけば、LANで接続したほかのマシンから取得しておいたメールを読むことができる（**図1**）。



必要なソフトウェアのインストール

必要なソフトウェアは、次の4つだ。多くのディストリビューションには、すべて含まれているだろう。

- **fetchmail（本体）**
- **fetchmailconf（設定ツール）**
- **sendmail（MTA）**
- **ipop3d（POP3デーモン）**

ipop3dは、imapというパッケージに含まれている。TurboLinux 4.0／4.2では、コンパニオンCDか、FTPサイト（ftp://ftp.turbolinux.co.jp/pub/turbolinux/updates/4.2/RPMS/i386/imap-4.6-3.i386.rpm）から入手できる。

ソフトウェアのインストールがすんだら、ps axコマンドを実行してsendmailが起動しているか確認しておこう。起動していなければ、linuxconfやturboserviceコマンドで自動起動するようにする。

次に、ipop3dを有効にする。エディタで/etc/inetd.confを開き、#pop-3で始まる行の先頭にあるコメント記号“＃”を削除し、inetdを再起動するか、Linuxを再起動して設定を反映させる。

ローカルネットワーク内メール環境設定の確認

fetchmailの設定をする前に、ローカルネットワーク内のメール配送を確認しよう。まず、mailコマンドで、自分宛にメールを出してみる。ここでは、一般ユーザーhikaruを例にする。

```
$ mail hikaru
Subject: Test mail
これは試験です。
.
Cc:
```

メールのタイトルを入力し、続けて本文を入力する。本文の終了は、行頭に“.”を1文字だけ入力する。Cc: は、Enterだけ入力すればよい。送信がすんだら、mailコマンドを実行して、メールが届いていることを確認する。

今度はLANでつながったクライアントマシンでメールが読めるかどうかチェックする。クライアントマシンのメ

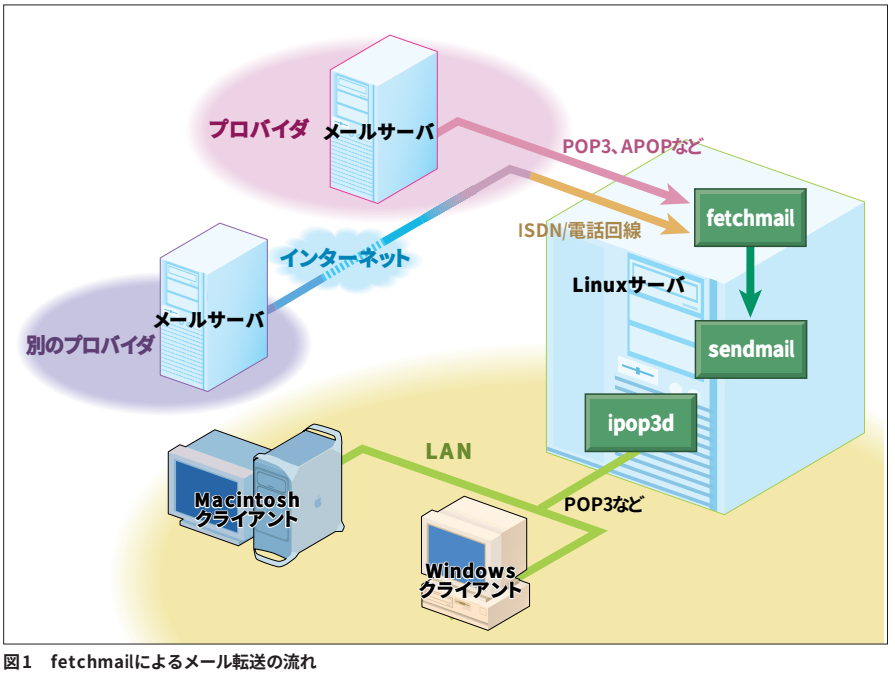
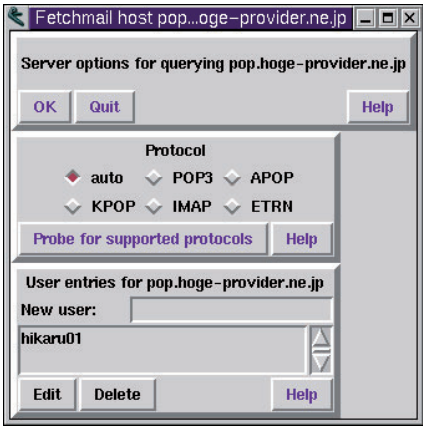
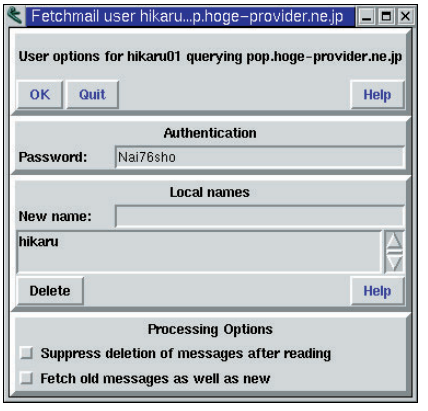


図1 fetchmailによるメール転送の流れ

画面1　メール取得間隔とPOPサーバを指定。



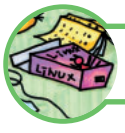
画面2　プロトコルとメールアカウント設定。



画面3　メールパスワード設定とローカルユーザー指定。

ールソフトでPOP3サーバをLinuxサーバに、メールアカウントをhikaruに設定する。パスワードは、Linuxサーバのログインパスワードだ。

Linuxサーバのコンソールからhikaru宛にメールを送り、メールソフトで受信できればOKだ。



fetchmailの設定

いよいよfetchmailの設定だ。fetchmailの設定は、ホームディレクトリの.fetchmailrcというファイルで行う。X Window Systemで動作する設定ツールfetchmailconfを使うと簡単に.fetchmailrcを作成できるので、これを利用しよう。

fetchmailconfを起動したら、[Configure fetchmail]ボタンを押し、次のウィンドウの、[Novice Configuration]ボタンを押して設定を始める。

新しく開いたウィンドウの、Poll interval:にメールの自動取得間隔を秒単位で指定する。たとえば、3時間おきなら10800を指定する。New Server:には、プロバイダのメール（POP）サーバの名前を入力してEnterキーを押す（**画面1**）。続いて、登録したメールサーバをクリックしてリストから選び、[Edit]ボタンを押すと**画面2**が開く。

ここでは、メールを取得する際のプロトコルと、プロバイダのメールアカウントを指定する。プロトコルは、初期設定のautoのままでもよいが、プロバイダに接続した状態で[Probe for supported protocols]ボタンを押すと適切なものを設定してくれる。New user:には、プロバイダのメールアカウントを記入してEnterキーを押す。そして、アカウント名をリストから選び[Edit]ボタンを押すと**画面3**が開く。

画面3では、Password:にプロバイダのメールパスワードを、New name:にメールを受け取るLinuxサーバのユーザー名を設定する。

この例では、プロバイダのhikaru01というアカウントに届いたメールを、3時間おきにpop.hoge-provider.ne.jpから取得し、Linuxサーバのユーザーhikaruに届ける。

最近のバージョンのfetchmailは、MIMEエンコードされたメールヘッダをデコードするが、サブジェクトなどに余計な改行が混入することがあるので、no mimedecodeというキーワードを追加して、デコードを抑制する。なお、.fetchmailrcには、メールパスワードがそのまま書き込まれるので、ほかのユーザーから見られないようにchmod 600する必要がある。fetchmailconfは、自動的にchmod 600にしてくれるので安心だ。こうして完成した~/.fetchmailrcは**リスト1**のようになる。

実行は、コンソールからfetchmailと入力するだけだ。fetchmailはデーモンプロセスとして動作し、指定した時間ごとにメールを取得する。

接続したプロバイダ以外からインターネット経由でメールを取得する場合には、POP3ではなく、暗号化したパスワードを用いるAPOPのようなプロトコルを使ったほうがよいだろう。

▼リスト1　完成した.fetchmailrc

```
set postmaster "hikaru"
set bouncemail
set properties ""
set daemon 10800
poll pop.hoge-provider.ne.jp with proto POP3
       user "hikaru01" there with password "Nai76sho" is hikaru here
       no mimedecode
```

DHCPサーバ〜IPアドレスをサーバで集中管理する〜

文：編集部　Text : Linux magazine

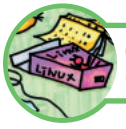
Linuxマシンを、WindowsマシンやMacintoshとEthernetを使ったLANで接続するには、TCP/IPというプロトコルを使用する。TCP/IPではIPアドレスで通信相手を指定するので、それぞれのマシンに違ったIPアドレスを割り当てる必要がある。

IPアドレスのほかにも、DNSサーバ（ホスト名をIPアドレスに変換する）の指定、デフォルトゲートウェイ、サブネットマスクなど、ネットワーク接続のための各種パラメータを設定しないと通信できない。

この作業の負担を軽減するため、ネットワークパラメータをサーバ側で管理して、クライアントマシンはサーバからその情報を得るという機能を実現するのが、DHCP（Dynamic Host Configuration Protocol）である。クライアントは最初のネットワーク接続時に、DHCPリクエストのメッセージをLAN上にブロードキャストで送り、DHCPサーバがそれに答えて情報を送るというものである。

クライアントマシンのネットワーク設定をDHCPサーバから取得するようにしておくと、ネットワークの変更があってもサーバ側の設定を変更するだけで、クライアントマシンは何も変更する必要がないので、多数のクライアントマシンを接続している環境でも手間が省ける。

また、ノートPCを持ち運んで会社と自宅で使う場合など、ネットワーク環境が違っていても、PCの設定を変えずに使えて便利である（ただし、どちらもDHCPサーバが稼働している必要がある）。



dhcpdのインストール

ほとんどのLinuxディストリビューションには、dhcpdというDHCPサーバのプログラムが含まれている。Red Hat系のLinuxの場合には、RPMパッケージで提供されており、LASER5 Linux 6.0の場合はdhcp-2.0b1pl6-6.i386.rpm、TurboLinux 4.2の場合はdhcp-2.0-4.i386.rpmがCD-ROMに入っている。

もし、使っているディストリビューションに入っていなければ、dhcpdのソースプログラムをRINGサーバなどのFTPサイト（ftp://ring.gr.jp/pub/net/dhcp/ISC/dhcp-2.0.tar.gz）から入手して、インストールする。

RPMパッケージをインストールしても、dhcpdの設定ファイル/etc/dhcpd.confは作成されないので、エディタで作成する。**リスト1**が今回の設定例だが、それぞれの項目の簡単な説明を**表1**にまとめておいた。optionで始まる行は、基本的にLinuxサーバでLANの設

▼リスト1　/etc/dhcpd.confの設定例

```
option domain-name "home-server.gr.jp";
option domain-name-servers 192.168.1.1;
option routers 192.168.1.1;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
# 60sec*60min*24hour = 86400 seconds
default-lease-time 86400;
max-lease-time 604800;
subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.128 192.168.1.254;
        host notepc {
                hardware ethernet 00:60:97:8a:a4:76;
                fixed-address 192.168.1.100;
        }
}
```

割り当てるIPアドレスの範囲

固定IPを割り当てるときだけ追加する

名前	パラメータの説明
option domain-name	ドメイン名
option domain-name-servers	DNS名（DNSサーバのIPアドレス）
option routers	ルータアドレス
option subnet-mask	サブネットマスク
option broadcast-address	ブロードキャストアドレス
default-lease-time	デフォルトのDHCP貸し出し時間（単位：秒）
max-lease-time	最大DHCP貸し出し時間（単位：秒）
subnet	IPアドレスを割り当てるネットワークの指定
range	割り当てるIPアドレスの範囲
host	固定のIPアドレスを割り当てる場合のそのマシン名
hardware ethernet	クライアントのMACアドレス
fixed-address	割り当てる固定IPアドレス

表1　dhcpd.confの設定情報

定をするときに指定した値をそのまま指定すればよい。

subnetで始まるブロックがIPアドレス割り当ての部分で、DHCPサーバはrangeで指定した範囲の値を、衝突しないように各クライアントに配布する。また、hostブロック内で指定しているように、hardware ethernet行でMACアドレス（イーサネットカードがROMに持つ識別用の固有の番号）を書いておくと、固定のIPアドレスを割り振ることも可能だ。

default-lease-timeは、DHCPによって割り当てられたIPアドレスの有効時間を秒単位で指定する。その時間が過ぎるとクライアントは、またDHCPリクエストを行うことになる。

domain-name-serversは、プロバイダのDNSサーバを指定すればよい。モデムやTAではなくISDNルータを使用している場合、そのルータのIPアドレスを指定したほうがよいこともあるので、ISDNルータのマニュアルを参照して設定する。

dhcpd.confは複雑な設定が可能であり、ここでは全部説明できないので詳細は、man dhcpd.confで調べていただきたい。

最後に、dhcpdが配布したアドレスを記録しておくためのファイルを、

```
# touch /etc/dhcpd.leases
```

と入力して作成しておく。

dhcpdサーバの起動

最初にdhcpdを起動する際には、次のコマンドのようにデバッグメッセージを表示して、正しく動作するか確認するとよい。

```
# /usr/sbin/dhcpd -d -f
```

■クライアントマシンの設定

Windows 95マシンのネットワークの設定は、コントロールパネルのネットワークを起動し、［ネットワークの設定］から「TCP/IP →（ネットワークカード名）」をクリックすると、**画面1**のように［IPアドレス］の設定画面になる。「IPアドレスを自動的に取得」を選択し、OKボタンを押す。［ネットワークの設定］ウィンドウに戻ってOKを押し、再起動すれば、設定は終了である。

再起動後、スタートメニューから「ファイル名を指定して実行」を選択し、winipcfgを起動する（**画面2**）。ここで「すべて解放」、「すべて更新」を押すことで、DHCPサーバからネットワークパラメータを再取得することができる。もし、IPアドレスとサブネットマスクが、「0.0.0.0」になっていたら、DHCPサーバからアドレスを取得できていないので、もう一度Linuxマシンの設定を確認する。

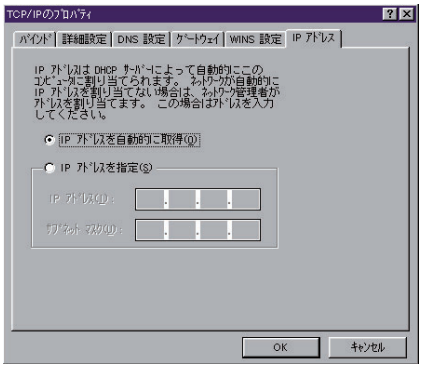
Macintoshのネットワーク設定は、コントロールパネルから［TCP/IP］を開き、［経由先］を「Ethernet」にし、［設定方法］を「DHCPサーバを参照」にすればよい。

Linuxマシンの場合は、インストール時のネットワークの設定でブートプロトコルにDHCPを選択すればよい。なお、DHCPはBOOTPプロトコルを含んだものなので、BOOTPを選んでもかまわない。

もし、DHCPでネットワーク設定を取得できない、または、取得するまでに非常に時間がかかるというときは、DNSサーバの設定を確認してみよう。DHCPサーバの/etc/resolv.confファイルにnameserverが指定してあった場合、その指定されたサーバを調べるため帰ってこない答えを待っている可能性がある。

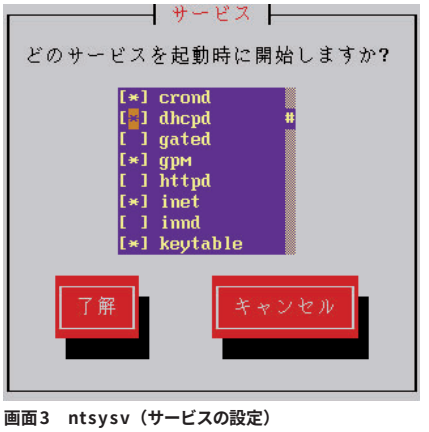
■dhcpdを自動で起動させる

dhcpdが正しく動作したら、システム起動時に自動的に実行するように設定しておく。Red Hat系のLinuxなら、ntsysvコマンドを実行して、dhcpdサービスを［＊］にする（**画面3**）。TurboLinuxでは、turboserviceコマンドで同様に行う。



画面1　Windows 95でのネットワーク設定では、「IPアドレスを自動的に取得」にする。

画面2　Windows 95でwinipcfgを実行したところ「すべて解放」で取得したデータを無効にし、「すべて更新」でネットワーク設定をDHCPサーバから取得する。



画面3　ntsysv（サービスの設定）
LASER5 Linux 6.0のコンソールから、ntsysvコマンドで起動時に実行するサービスを設定する。

Apache　～家族で使おうWebサーバ～

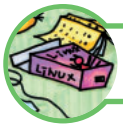
文：山岸典将　Text : Norimasa Yamagishi

Webサーバとは何かについてはもはや説明は不要だろう。世界中で最も使われているWebサーバがApacheだ。実際に運用するにも、関連情報もかなり多く、フリーということを除いたとしてもお勧めできるサーバと言ってよいだろう。

Apacheの概要

WebサーバにはApache以外にもOS付属のものや、商用のものがある。では、なぜApacheが選択されるのかといえば、その理由として、豊富な実績と高い機能が挙げられるだろう。

Apacheは、さまざまな機能をモジュール化し、自由に追加できることで、トータルで優れた機能を実現している。標準機能の多くも実はモジュールで実現されている。また、このモジュールも、バージョン1.3からは必要に応じてダイナミックに組み込むことが可能となった。そのため新たな機能が必要になっても、コンパイルしなおす手間は不要だ。CGIでよく使われるperlのランタイムをApacheに組み込むmod_perl、SQLデータベースを呼び出しHTMLを作成するためのPHP3など非常に有用なものも多い。



RPMパッケージからのインストール

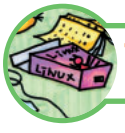
ほとんどのLinuxディストリビューションにはApacheが収録されているので、rpmコマンドなどを使い、CD-ROMからインストールしよう。

RPMパッケージからインストールした場合、Apacheの起動スクリプトもインストールされるので、

```
# /etc/rc.d/init.d/httpd start
```

とすればApacheが起動する。停止する場合は、次のようにする。

```
# /etc/rc.d/init.d/httpd stop
```



tarボールからのインストール

1月22日に最新版の1.3.11がリリースされた。せっかくなので、ここでオリジナルのソースからインストールする方法を説明しておこう。Apacheの最新版は、FTPサイトftp://ftp.apache.org/で配布されている。

次の手順で、tarボールからソースを展開し、makeする。その後、スーパーユーザーになってインストールする。

```
$ tar zxvf apache_1.3.11.tar.gz
$ cd apache_1.3.11
$ ./configure
--prefix=/usr/local/apache
$ make
$ su
# make install
```

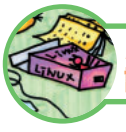
3行目の“ --prefix= ”で指定している“ /usr/local/apache ”はApacheをインストールするディレクトリだ。

Apacheは、apachectlコマンドで起動する。

```
# /usr/local/apache/bin/apachectl start
```

停止する場合は、“ apachectl stop ”とする。

ただし、インストール時には、最初に表示されるHTMLドキュメントが用意されておらず、さらに次に説明するポート番号の設定も一般的なものとは異なっている。このままでは、Webブラウザで Linuxマシンの URLを指定してもエラーになるはずだ。ポート番号の指定と、最初に表示されるドキュメントに関しては、後述の「Apacheの設定」を読んでほしい。

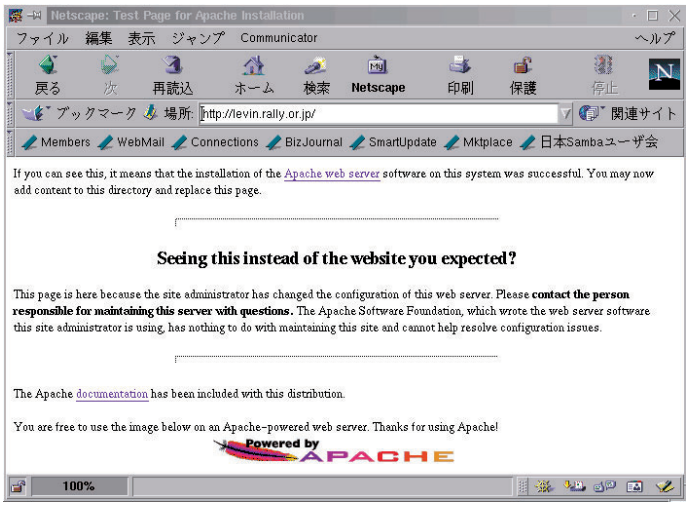


Apacheの設定ファイル

RPMパッケージからインストールした場合、Apacheの設定ファイルは、/etc/httpd/conf以下にあるhttpd.conf、srm.conf、access.confとなる。

tarボールから最新バージョンをインストールしていれば、/usr/local/apache/conf（インストール時に設定したディレクトリの下のconfディレクトリ）にあるhttpd.confが設定ファイルとなる。最近のバージョンでは、MIMEタイプを指定するmime.types以外はhttpd.confにまとめられているのだ。実際にはsrm.conf、access.confといったファイルも存在するが、それらの中には、何も設定されていない。この場合、以下の説明においても、設定ファイルはhttpd.confのみになるので

画面1　インフォサイエンス社のjapacheページ
(http://japache.infoscience.co.jp/japache/)



画面2　Apacheのindex.html.en

注意してほしい。

Apacheの設定

Apacheの設定項目はディレクティブと呼ばれ、まずディレクティブ名を書き、タブまたはスペースで区切って設定値を記述する。行頭が“＃”になっている行はコメントだ。Apacheには膨大な設定項目があり、ここですべては解説しきれないため、特に重要なディレクティブについてのみ解説しよう。なお、インフォサイエンス社ではApacheに関する情報の日本語翻訳（http://japache.infoscience.co.jp/japache/）を提供している（**画面1**）。英語が苦手な方は参考にするとよいだろう。

以下、重要なディレクティブの設定を順に解説する。変更したApacheの設定ファイルに間違いがないかチェックするには、“apachectl configtest”を実行すればよい。

・Port

RPMパッケージをインストールしていれば問題ないが、tarボールからビルドしたApacheでは、ポート番号の指定が8080になっている。Webではポート80の使用が基本なので、このままではアクセスできない。“Port”セクションを8080から80に変更しよう。ただし、root以外のユーザーでApacheを起動する場合はポート80を利用することはできない。

・User、Group

httpd.confで設定する。Apacheを実行する権限をユーザー、グループについてそれぞれ指定する。ここで言う実行権限とは、Apacheを起動する権限ではなく、Apacheがファイルを読んだり、プログラムを実行したりするための権限だ。すなわち、HTMLファイルを用意しても、ここで指定されたユーザーにそのファイルを読む権限がないと表示されない。

また、通常はnobodyになっているので、実際にnobodyというユーザーが存在する必要がある。もし、nobody以外のユーザーを指定する場合は、セキュリティ上、できる限り特殊な権限を持っていないユーザーを指定すること。rootなどを指定するのは厳禁だ。理想的なのは、必要最低限の権限しか持たないApache専用のユーザー、グループを作成し、ここで指定することだ。

・DocumentRoot

RPMパッケージでインストールした場合は、srm.confで設定する。HTMLドキュメントのルートディレクトリを指定する。すなわち、サーバ名のみを指定された場合にアクセスするディレクトリとなる。RPMパッケージをインストールしたときのデフォルトは/home/httpd/htmlとなる。一方、tarボールからビルドした場合は、/usr/local/apache/htdocs/となっている。

サーバ名だけを指定してアクセスした場合に表示させたいファイルは、ここで指定したディレクトリの中に、“DirectoryIndex”で指定したファイル名で作成する。“DirectoryIndex”についてはあとで説明する。実は、デフォルトではこのディレクトリの中にindex.html.enといった各国語でのindex.htmlとなるファイルが置かれている（**画面2**）。

・DirectoryIndex

RPMパッケージでインストールした場合は、srm.confで設定する。URL指定がファイル名でなく、ディレクトリでアクセスされた場合に呼び出されるデフォルトのファイルを指定する。通常、

http://xxx.co.jp/というURLを指定した場合、index.htmlというファイルが呼び出されるのは、DirectoryIndexにIndex.htmlが指定してあるからだ。

・UserDir

このディレクティブもRPMパッケージでインストールした場合は、srm.confで設定する。サーバマシン上にアカウントを持つユーザーが、HTMLファイルを公開するためにファイルを置く場所を、各自のホームディレクトリからの相対パスで指定する。デフォルトはpublic_htmlとなっている。

ここに指定したディレクトリをホームディレクトリの下に作成することにより、“ http://サーバー名/~ユーザー名 ”という形で、ユーザーが自由にWebページを公開できるようになる。ただし、このときに“ User ”、“ Group ”で指定したユーザー、グループが読み込みアクセスできるように、各自がアクセス権を変更しておく必要がある。

また、通常のユーザーのホームディレクトリとは別にWeb用のディレクトリを作成したい場合には絶対パスで/usr/wwwのように記述すればよい。この場合は~fooで参照されるのは、/usr/www/~fooとなる。Windowsユーザーが多い場合などはSambaを使い、/usr/wwwを公開してもよいだろう。

・アクセス制御

RPMパッケージでインストールした場合は、access.confで設定する。<Directory ディレクトリ名>～</Directory>で、そのディレクトリに関するアクセス制御をすることができる。

以下の例は、1行目で禁止、許可設定の読み込み順序を指定している。そして、2行目ですべてのドメインからのアクセスをいったん禁止し、3行目ではascii.co.jpからのアクセスのみを許可するという内容となっている。

```
<Directory />
        order deny, allow
        deny from all
        allow from ascii.co.jp
</Directory>
```

ディレクトリと同様に、<Files ファイル名>～</Files>で、ファイルに対するアクセスの制御も可能だ。

また、ディレクトリごとに.htaccessというファイルを作成し、ディレクティブと設定値を書くことによって、ディレクトリごとにアクセス制御を行うこともできる。逆に各ユーザーによる.htaccessを無効にしたい場合は、以下のようにする。

```
<Directory />
        AllowOverride None
</Directory>
```

MIMEタイプの設定

Webがこれだけ普及した理由のひとつは、画像、動画、音などのマルチメディアが扱えるからだろう。MIMEタイプとはマルチメディアの種類の規定だ。

Webサーバは、クライアントから要求されたファイルがどのようなMIMEタイプであるかを、そのファイルを送る前にクライアントに渡す。その際にMIMEタイプを決定するために使われるファイルが、/etc/mime.typesだ。種類の判定には拡張子を使う。

mime.typesでは、まずMIMEタイプを書き、タブまたはスペースで区切って拡張子を記述する。

```
text/html                       html htm
```

CGIを使う

Web上の掲示板など、インタラクティブなページの作成にはCGI（Common Gateway Interface）が多く利用されている。

ApacheでCGIを使用するためには、2通りの方法がある。RPMパッケージでインストールした場合は、srm.confで設定する。

1. 設定ファイルにCGIとして扱う拡張子を設定する。

```
Addtype application/x-httpd-cgi .cgi
```

もしくは、以下のように設定する。

```
AddHandler cgi-script .cgi
```

厳密には、この2つが意味する内容は異なるが、どちらも.cgiという拡張子がついたファイルをCGIプログラムとして扱う。

また、CGIプログラムを置くディレクトリは、前述のアクセス制御でCGIを有効にする必要がある。

RPMパッケージでインストールした場合には、/usr/local/apache/cgi-bin/の部分を/home/cgi-bin/に読み替えていただきたい。

```
<Directory /usr/local/apache/cgi-bin>
        Options ExecCGI
</Directory>
```

2. 設定ファイルにCGIプログラムを置くディレクトリを指定する。

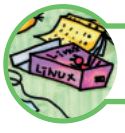
以下のように設定すると、http://

www.xxx.co.jp/cgi-bin/foo.cgiというURL指定でアクセスされるCGIプログラムを、/usr/local/apache/cgi-bin/に置くことができる。

```
ScriptAlias /cgi-bin/ /usr/local/
apache/cgi-bin/
```

この場合は任意の拡張子でCGIプログラムが実行できる。

これら2つの方法で、安易に選択されがちなのは、自由な場所にCGIプログラムを置くことができる1の方法だが、セキュリティの観点から言えば、2を選択したほうがよい。CGIプログラムは通常、クライアントからアクセスされる場所とは別の部分に置かれるべきだからだ。実際、1の方法を取ると、各ユーザーが自由にCGIプログラムを作成、利用できる反面、うっかり危険なCGIプログラムを動作させてしまうこともありうる。



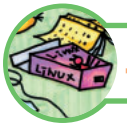
SSIを使う

SSI（Server Side Include）とは、HTMLファイル内に別のファイルの内容や、プログラムの実行結果を挿入する機能だ。一般に、shtmlという拡張子を持つファイルがSSIを利用したファイルであることが多い。SSIは、RPMパッケージでインストールした場合は、srm.confで設定する。以下のように設定することで、shtmlという拡張子を持つファイルがSSIの処理の対象となる。

```
Addtype text/html .shtml
AddHandler server-parsed .shtml
```

また、SSIプログラムを置くディレクトリには、前述のアクセス制御でSSIを有効にする必要がある。RPMパッケージでインストールした場合には、/usr/local/apache/htdocs/の部分を/home/httpd/html/に読み替えていただきたい。

```
<Directory /usr/local/apache/htdocs>
        Options +Includes
</Directory>
```



ユーザー認証

Apacheではユーザー名とパスワードで認証を行い、情報へのアクセスを制限することができる。認証の方法はいくつかあるが、ここでは一番簡単に設定できるBasic認証について解説する（**画面3**）。RPMパッケージでインストールした場合は、access.confで設定する。

1. 認証をさせたいディレクトリもしくはファイルに対して、アクセス制限の方法を設定ファイルでAuthTypeと指定する。

2. ユーザー名とパスワード入力時のメッセージを、AuthNameで指定する。このメッセージは、"（ダブルクォーテーション）で囲む必要がある。

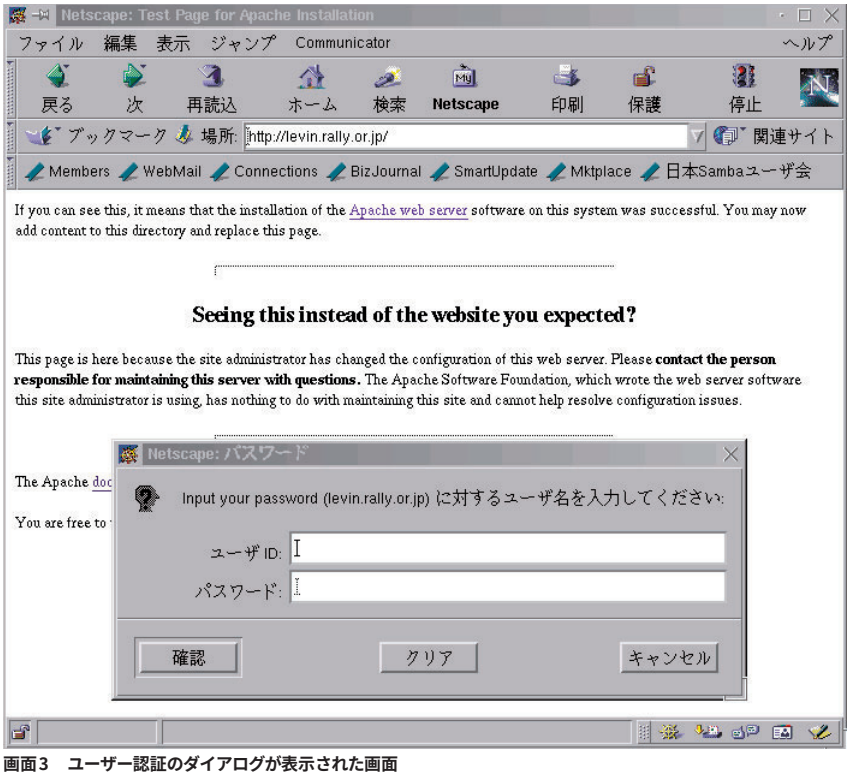
3. htpasswdコマンドでユーザー名とパスワードのデータを作成する。ただし、最初に作成する場合は、“ -c ”オプションをつける必要がある。

```
# /usr/local/apache/bin/htpasswd
-c /usr/local/apache/conf/
userpass ken
```

RPMパッケージからインストールした場合は次のようになる。

```
# /usr/bin/htpasswd -c /etc/httpd/
conf/userpass ken
```

4. 作成したデータファイルの位置を



画面3　ユーザー認証のダイアログが表示された画面

AuthUserfileで指定する。

5. 最後にrequireでアクセスを許可するユーザーを指定する。

実際の記載例は**リスト1**に示すようになる。

ここまでの設定で、Apacheをホームサーバで利用するには十分だ。最近では、フリーで使えるCGIの掲示板などがたくさんあるので、これらを使って家族の伝言板を作るのもよいだろう。

また、Apacheなどと組み合わせて利用するWebベースグループウェアも比較的低価格で販売されている。家庭で使うには大げさかもしれないが、会社の部課単位で使うにはちょうどよいかもしれない。

しかし、Webサーバといえば、インターネットを介して世界中につながっているからこそ面白いという方も多いだろう。LANの中だけで、ホームサーバとして利用するという目的からは少々はずれるのだが、最後に、Webサーバ管理に関するトピックをいくつか紹介しよう。

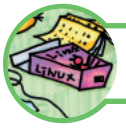
▼リスト1　Basic認証の設定

●tarボールからビルドした場合

```
<Directory /usr/local/apache/htdocs/secret>
        AuthType Basic
        AuthName "Input your password"
        AuthUserfile /user/local/apache/conf/userpass
        require user ken jiro satoshi
</Directory>
```

●RPMパッケージをインストールした場合

```
<Directory /home/httpd/html/secret>
        AuthType Basic
        AuthName "Input your password"
        AuthUserfile /etc/httpd/conf/userpass
        require user ken jiro satoshi
</Directory>
```



ログファイル

インターネットにつながったWebサーバを立ち上げたら、どれくらい利用者がいるかを知りたいという要求は多いだろう。Apacheではリクエストとその結果を記録するaccess_log、エラー情報を記録するerror_log、プロセスIDを記録するhttpd.pidという3つのログファイルを作成する。この中でも興味があるのはaccess_logファイルだろう（もちろん問題が発生していないかを知るためにも、error_logファイルもチェックするようにしたいが）。

これらのログファイルは、RPMでインストールした場合は/var/log/httpd以下に、tarボールからビルドした場合は/usr/local/apache/logs以下に作られる。

access_logファイルは比較的わかりやすい形式になっている。アクセスしてきたホスト、ユーザー（通知があった場合）、日付、要求されたファイルなどが順番に記録されている。

```
210.160.90.115 - -
[29/Jan/2000:10:38:37 +0900] "GET
/index.html HTTP/1.1" 200 1358
```

もっとも、比較的わかりやすいというだけで、すべてのログに目を通すのは面倒だ。そのため、実際には何らかのツールを用いて、ログ解析を行うことになる。自分でperlやawkを用いて解析してもよいが、よく使われているのがanalogというツール（http://www.statslab.cam.ac.uk/~sret1/analog/）だ。どのファイルに、どこからアクセスがあったのかの統計をHTMLファイルにして見ることができる。

ロボットへの対策

インターネット上の検索エンジンは、ロボットと呼ばれるプログラムを利用してWebのデータベースを作成する。こうして作られたデータベースをもとに検索を行っているのだ。しかし、希望しないのに検索の対象にされたWebページもあるだろう。その場合、DocumentRootで指定したディレクトリにrobot.txtというファイルを以下のような内容で作成する。

```
user-agent: *
Disallow: /
```

これはすべてのロボットに対する、“/”以下のディレクトリのデータを収集しないで欲しい、という依頼を意味する。ここで依頼と言ったが、実は残念ながら、ロボットのアクセスのみを排除することは、現実にはできない。なぜならば、Webサーバ側でロボットであるかどうかを判断できないからだ。このrobot.txtというファイルは、「A Standard for Robot exclusion」というロボット作成のための自主的な決まりで規定されているものだ。

Linuxでつくる3Dグラフィックス

Shade for Linux Preview Kit

国内最大のユーザー数を誇る3DCGソフトウェア「Shade」のLinux版が9800円で発売された。
Linux用の3DCGソフトとしては、LASER5 Linux 6.0にバンドルされている「BLENDER」などが有名であるが、今回の「Shade」の移植により、Linuxでの3DCG制作環境が大いに広がったといえる。
3DCGの道は奥が深く、ソフトを使いこなすのにも一苦労するのが常なのだが、初心者からプロフェッショナルまで幅広いユーザーを持つ「Shade」には、市販のマニュアル本やテクニック集などが充実しており、このあたりが大変心強い。
この「Shade for Linux Preview Kit」で、あなたも華やかな3DCGの世界へ飛び込んでみてはいかがだろう？

文：かざぐるま + 香山明久
Text : Kazaguruma + Akihisa Kayama

テライユキ・有栖川麗子・優月まりな…。これらのアイドルの名前をあなたはご存じだろうか。すべて知っているというあなたはなかなかのもの。実は、彼女たちはいずれも、今インターネットなどで人気を集めているバーチャルアイドル。つまりは3DCGでモデリングされた架空の存在なのだが、中には写真集やCDまでリリースしているキャラクターも存在するのであなどってはいけない。

こうした社会現象をどう見るかは別にして、近頃巷にあふれる3DCGは、一昔前に比べ驚くほど精巧になってきた。当然、これにはハードウェア環境の進歩によるところも大きいのだが、低価格・高機能な3DCGソフトウェアがホビー市場に流通するようになったという要因も見逃すことはできない。中でも「Shade」は、その操作性の高さから国内で最も多くのユーザーから支持され、かつラインナップも豊富に用意された秀逸な3DCGソフトだ。

今回は、この「Shade for Linux Preview Kit」の機能を紹介しながら、Linux環境下で簡単な3DCG制作にチャレンジしてみることにする。

動作環境

「Shade for Linux Preview Kit」は、正式版に先だつプレビュー版であり、日本語には未対応となっているが、ユーザーインターフェイスなどは現行のWindows版およびMacintosh版の「Shade」との高い互換性を保っている。

「Shade」の魅力のひとつに、市販の素材集やサンプルデータなどが充実しているという点を挙げることができるが、これらの資産はWindows版CD-ROMからLinux版へそのまま流用することができる。当然、これまでに他のOSで作成したデータをLinux上で読みとることも可能だ。

Windows版およびMacintosh版には「Shade Professional」「Shade Personal」「Shade Debut」「myShade」などのラインナップが用意されているが、今回のPreview Kitを見た限りでは、製品版「Shade for Linux」の仕様は、プラグインおよびスクリプトをサポートするミッドレンジの「Shade Personal」に相当するものとなりそうだ。

動作環境は、「カーネル2.2のLinuxが稼働するPC/AT互換コンピュータで、正式に動作を保証するディストリビューションはRed Hat Linux 6.0（英語版）、KDE 1.1が必要」となっている。また、LASER5 Linux 6.0（日本語版）およびTurboLinux日本語版4.0においても動作は確認されているが、これはあくまでもメーカーによる動作確認であり、動作保証はされていない。さらに、TurboLinuxの場合は、

glibc-2.1.1-6.i386.rpm
libstdc++-2.9.0-12.i386.rpm

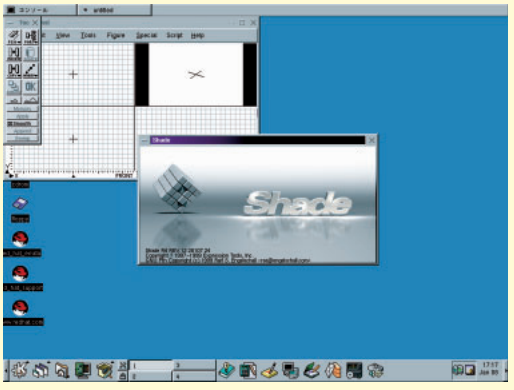
によるアップデートが必要となる。

今回のテストにはRed Hat Linux 6.1（日本語版）を使用したが、インストールや動作に特に問題はなかった。

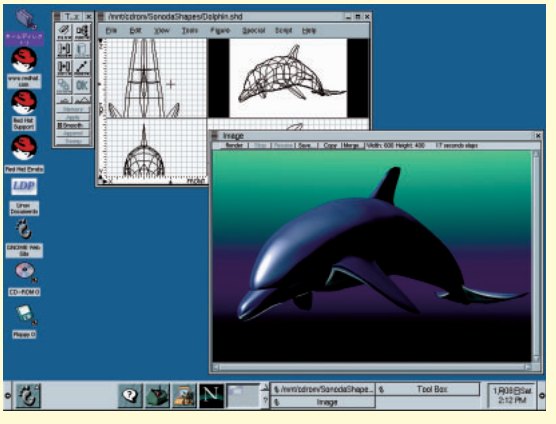
ハードウェア環境については特に記載はなかったが、Windows版Shade Personalの場合、空きメモリ容量16Mバイト以上（推奨32Mバイト以上）、ハードディスク空き容量30Mバイト以上（推奨60Mバイト以上）を要求する。もちろんインストールにはCD-ROMドライブが必要で、ソフトの性格上フルカラー環境も必須だ。3DCGはヘビーな処理を要求するため、マシンスペックは高いほど良いだろう。

Shadeのインストール

「Shade for Linux Preview Kit」はRPMパッケージでCD-ROMに収録されているため、インストールは容易だ。マニュアル類はすべてPDF形式のオンラインマニュアルとしてCD-ROMに収録されているため、閲覧にはAcrobat Readerが必要となる（製品CD-ROMにはAdobe Acrobat Reader 4.0が収録されている）。



画面1　Shadeの起動画面（KDE-1.1.2）



画面2
マニュアルにはKDE-1.1が必要とあるが、GNOMEでの起動や動作にも、特に問題は見受けられなかった

Shadeのインターフェイス

「Shade for Linux Preview Kit」のインターフェイスは、Windows版のそれとほぼ同じであり、ショートカットキーなども共通するところが多いため、既存ユーザーにとっては大変馴染みやすい仕様となっている。もちろん、はじめてShadeを触るという人でも、3DCGの基礎さえ理解できれば操作はさほど難しくない。このあたりの使いやすさが、Shadeが多くのユーザーに支持されている理由のひとつなのだろう。

3DCGソフトで画像を制作する場合、最初に三次元立体形状データの入力を行わなければならない。これが一般に「モデリング」と呼ばれる作業で、この操作を行うために、Shadeのメインインターフェイスは、三面図＋透視図の4つに分割された図形ウィンドウで形成されている（**画面3**）。

ウィンドウ中央を十字に走る境界線の左上に位置するのが上面図（Top）で、以下同様に、左下が正面図（Front）、右下が側面図（Side）となっている。これは設計図などに用いられる三面図と同じものなので、設計図面を見慣れた人には容易に理解できるだろう。右上にあるのが透視図（Pers）となっており、三面図上で入力された形状をこの画面で確認することができる。

図形ウィンドウ内の三面図および透視図には、それぞれマウスポインタの移動に合わせて動く十字カーソル（三次元カーソル）が表示される。モデリングはこの三次元カーソルを基準に行うことになるが、この三次元カーソルが指すポイントは、常に三面図および透視図に示される三次元空間内の同一点を指していることを意味している。

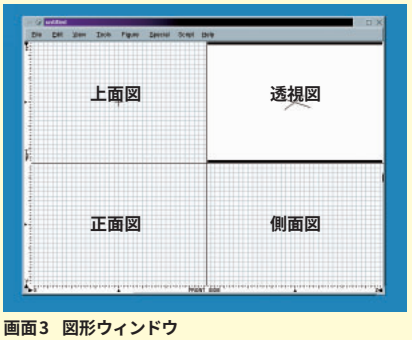
3DCGを制作する場合、まずはこのあたりの概念を理解するのが必要となるのだが、理屈よりも、まずはいろいろと触ってみて「慣れてみる」ことがいちばんかもしれない。

それでなくても狭いモニタをさらに4分割して作業を行うため、細かなモデリングはかなり窮屈になりそうなもの。しかし、この三面図と透視図の占有率はマウスのドラッグで容易に変更可能だ。また、メニューバーから任意の図面を選択し、その図面のみを表示することもできるので、実際の作業では、図面を切り替えながらモデリングを進めることになる（**画面4**）。

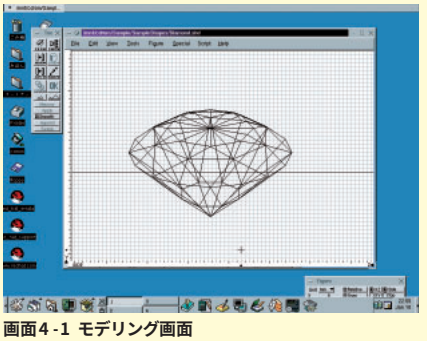
図形ウィンドウ内でのさまざまな操作は、ツールボックス（Tool Box）およびメニューバーから行う。また、必要に応じてメニューバーから任意の操作ボックスを起動しなければならない。このあたりは一般のグラフィック系ソフトと大差はない（**画面5**）。3DCGを制作するためには、それなりの複雑な操作が必要となるため、用意されているショートカットキーをうまく使うことも必要となりそうだ。

こうした操作はやはり慣れていくしか仕方ないのだろうが、Shadeの場合、メニューバーからの操作を要求されるケースが多いため、英語表示オンリーではオペレートが結構つらく感じる。使い込んでゆけば解決する問題なのだろうが、英語が苦手なユーザーのためにも、日本語に対応した正規版の早期リリースが望まれるところだ。

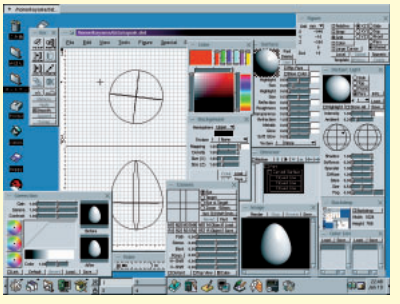
Shadeの基本操作画面



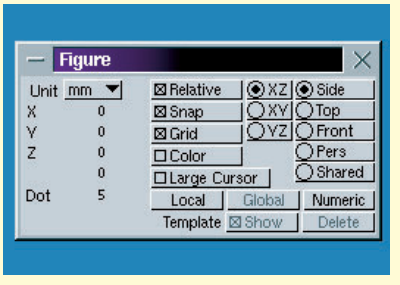
画面3　図形ウィンドウ
図形ウィンドウには、上面図（Top）、正面図（Front）、側面図（Side）と透視図（Pers）があり、各図面内に表示されている三次元カーソルでモデリングなどの基本操作を行ってゆく。各図面の占有率は変更することも可能。



画面4-1　モデリング画面
図形ウィンドウ三面図や透視図から任意の図面のみを表示することも可能。切り替えはメニューバーから図形（Figure）コントローラを起動して行う。



画面5　ツールボックス
操作に必要な基本機能はツールボックスのプルダウンメニューから選択する。そのほか、必要に応じてメニューバーから操作ボックスを起動する。画面はモデリングに必要な操作ボックスをすべて起動したもの。結構な数だ。



画面4-2　図形コントローラ

モデリング

3DCGで物体を表現するためにはいくつかの手法がある。代表的な手法としては、物体を面の集合体（多角形）で表現する「ポリゴン」や、あらかじめ用意されている積み木のようなパーツを使って物体を組み立てる「プリミティブ」などを挙げることができる。

Shadeの場合、こうした方法とは異なり、モデリング作業にベジェ曲線を使った「自由曲面」で物体を表現するという独自の手法を用いる。これがShadeの最大の特徴であり、人体などの複雑な曲面構成の物体を描画する際でも柔軟なモデリングが可能となっている。ポリゴンなどを用いる他の3DCGソフトとは少々勝手が違うのだが、Illustratorのようなドロー系ソフトに近い操作感を持つため、こうしたソフトを使い慣れた人にはかえって習得しやすいかもしれない。

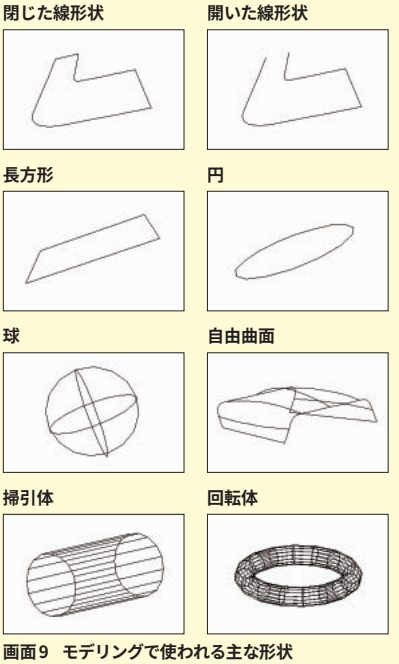
モデリングは3DCG作成の最も基礎となる操作であり、作りたい物体のイメージを細かなパーツに分解し、それぞれの形状を作成していく根気のいる作業を要求される。「Shade for Linux Preview Kit」の製品CD-ROMにはA4判で193ページ相当のチュートリアルが収録されているので、モデリングの基本を理解するためには、これに従ってサンプルを作成してみるのがよいかもしれない。

ほかのソフトウェアで作成したDXF形式またはEPSF形式のファイルから、形状データを読み込むことも可能なので、CAD等で作成したデータを流用してもよいだろう。また、DXF、RIB、Animation Master、VRML2.0、POV-Ray 3.0形式のフォーマットで出力することが可能となっている。

作成された各パーツは、ブラウザボックスというツール上で管理される。

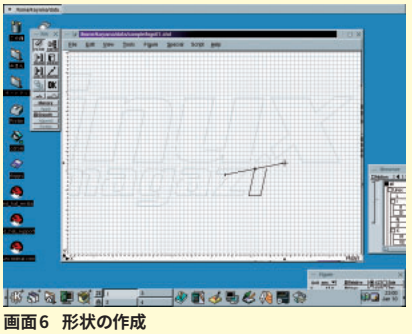
Shadeのモデリングで扱われる主な形状には以下のようなものがある。

- **閉じた線形状（Closed Line）**
  平らな面として認識される形状
- **開いた線形状（Open Line）**
  線として認識される形状。一般にモデリング操作の材料となる
- **長方形（Rectangle）**
  4つのポイントを持つ閉じた線形状
- **円（Disk）**
  厚さのない円形の形状
- **球（Sphere）**
  球形の形状
- **自由曲面（Curved Surface）**
  自由度が高く、複雑かつ微妙な曲面を扱える形状
- **掃引体（Extrude）**
  線形状または円を直線的に押し出した立体形状
- **回転体（Revolve）**
  線形状または円を軸に沿って回転させた立体形状

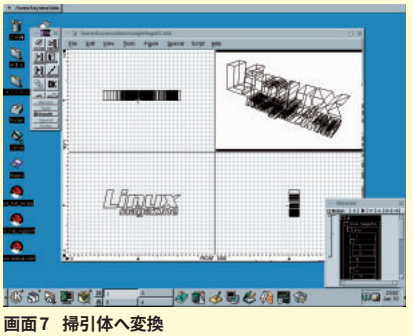


画面9　モデリングで使われる主な形状

モデリング　立体のモデリングにはさまざまな手法があるが、ここでは正面図上でLinux magazineのロゴを作成し、掃引体ツールを用いて立体に変換してみることにする。



画面6　形状の作成
PENボックスから閉じた線形（Closed Line）を選択しロゴを描画する。曲線の描画にはベジェ曲線を用いる。作成されたそれぞれのパーツの関係は、ブラウザボックス（右下に表示）で管理される。



画面7　掃引体へ変換
作成したロゴを掃引体へ変換する。変換にはツールボックスのSOLIDツールから掃引体（Extrude）を選択し、厚みを付けたい方向へポイントをドラッグすればよい。

画面8　作成された立体図
作成したロゴをレンダリングしてみた。質感などがまだ付いていないので味気ないが、立体感はそれなりに表現されている。

表面材質の設定

モデリングが完了したら、次は物体の表面の色や質感などを設定していく作業に入る。この作業は一般に「アトリビュート」などと呼ばれるが、これにより、味気なかった三次元立体がよりリアルなものへと変身する。

Shadeでは、物体の色・材質・光沢などの質感を細かなパラメータで指定できるため、より実物に忠実な表現が可能となる。物体の表面に質感を貼りつけるマッピング作業も、あらかじめ用意されているテクスチャから凹凸感や材質を選択できるため、操作は簡単。既存の質感のほかにBMPやPICTなどのイメージデータを用いたマッピングももちろん可能だ。5階層までの多重マッピングにも対応しているので、かなり複雑な表面材質でも表現することができるだろう。

こうした作業は、作成した形状ごとに行っていく必要があるが、それぞれの形状をパーツごとに管理しておけば作業がはかどる。こうしたパーツはモデリングの際にブラウザボックスで管理していくことになる。このブラウザボックスでそれぞれのパーツをうまくコントロールすることが、Shadeを使いこなすひとつのコツといえる。

設定できる表面材質の項目には以下のようなものがある。

・基本色

物体の色を決定する。あらかじめ用意されているテクスチャを選択してマッピングを行う場合にも、この基本色が反映される。色の設定には、カラースライダーを用いる方法やダイアログボックスから任意の色を選択するなどの方法がある。

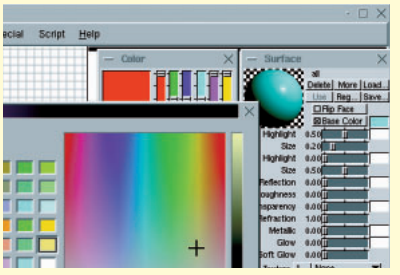
・陰影付け

材質によって異なる光沢や光の反射などの効果を形状に反映させる。材質が木の場合は光沢が弱くなり、鏡の場合は光沢や反射が強くなる。設定項目には、光沢（Highlight）・反射（Reflection）・透明度（Transparency）・屈折率（Refraction）などがあり、パラメータは各々のスライダーで設定する。

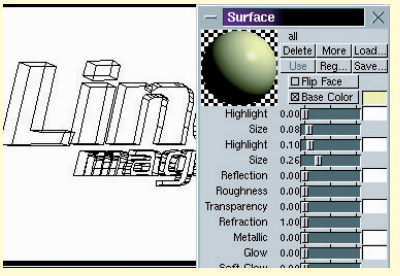
・マッピング

物体の表面にテクスチャを貼りつける。あらかじめ用意されているテクスチャには、ストライプ（Stripe）・チェック（Check）・大理石（Marble）・木目（Woody）・凹凸（Bumpy）・しわ（Wrinkle）などがある。マッピングされるテクスチャのサイズ・適用度・乱れ・方向などは任意に選択可能。

基本色



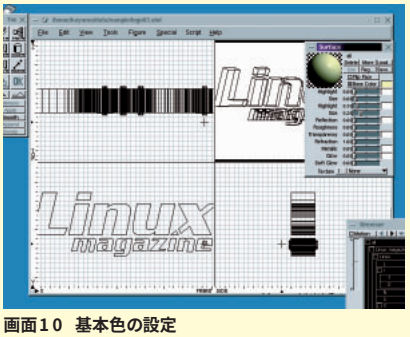
陰影付け



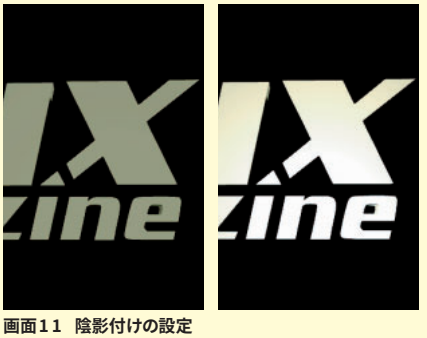
マッピング

画面13　設定できる表面材質

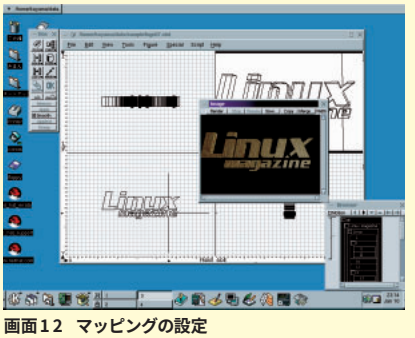
マッピング　物体の表面材質を設定することにより、立体はよりリアリティのある3Dモデルへと姿を変える。ここでは、ロゴに木目の材質をマッピングしてみる。



画面10　基本色の設定
木をイメージして基本色を設定する。木目をマッピングする場合は、仕上がりが基本色よりもかなり濃い目になるため、少し薄い色を設定するほうがより現物に近い仕上がりとなる。



画面11　陰影付けの設定
次に物体の光沢などを設定する。左は光沢を抑えたサンプルで、逆に右は光沢や反射を強調したサンプル。木の場合は光沢を抑えた設定にしておけばよい。



画面12　マッピングの設定
最後に木目のテクスチャをマッピングする。実際の物体をイメージしながら木目の大きさやテクスチャの適応度、木目の流れなどを設定する。イメージを確認するためにレンダリングを実行してみた。

視野および光源の設定

3Dモデルが完成したら、次は透視図を確認しながら視野の設定を行う。この際、透視図内に見えるワイヤーフレーム画像が、そのままレンダリングの際の画像イメージとなる。

Shadeで視野を設定するためにはカメラウィンドウを使用するが、これはその名の通りカメラアングルを決定するための操作である。実際に写真を撮影する場合、いくらモデルが美人でも、カメラマンの腕がイマイチでは良いポートレートの仕上がりは期待できない。それと同様に、視野の設定は3DCGの仕上がりを決める大切な操作となる。よりリアリティの出るようなアングルをじっくりと吟味しよう。

視野は実際にカメラを構えてファインダを覗くような感覚で、変化を与えながら操作することができる。カメラウィンドウの仮想ジョイスティック内をドラッグすることで、視点(Eye)、注視点(Target)、視点&注視点(Eye &Target)、ズーム(Zoom)の設定が可能だ。カメラウィンドウで示されるズームの数値などは、実際のカメラのレンズサイズに合わせて表示されるので、ある程度のカメラの知識があれば直感的に操作できる。また、気に入った視野の設定は、5パターンまでシステムに記憶することもできる。

視野の設定と同様に重要となるのが光源の設定だ。光源がなければ、レンダリングを行っても形状のイメージを得ることはできない。Shadeには、点光源(Point Light)、スポットライト(Spotlight)、無限遠光源(Distant Light)の3種類の光源が用意されており、それぞれ細かな設定を行うことができる(**画面17**)。

これまでの解説の中でレンダリングを行ってイメージを得ることができたのは、実は無限遠光源が設定されていたためなのである。

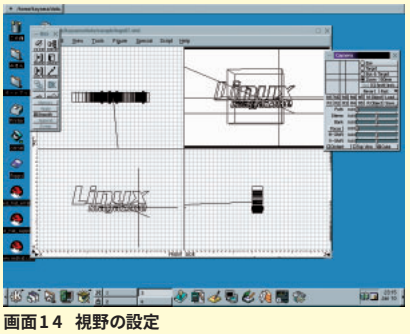
無限遠光源とは、太陽光の平行光線をシミュレートしたものを意味する。同様に点光源を使えば裸電球、スポットライトではその名の通りスポットライトと同じ光効果をシミュレートすることができる。こうした光源の設定ひとつで画像の感じは全く異なるもととなる。いろいろな設定を試して、最も仕上がりイメージに近い光源を選ぶことも3DCG制作では重要だ。

視点と光源の設定が済めばひととおりの作業は完了する。ただし、このままでは地面(床)や壁などが存在しないことになるので、必要な場合はこうした背景も三次元空間内に設定してやらなければならない。既存の画像イメージを背景に貼りつけておいてもよいだろう。

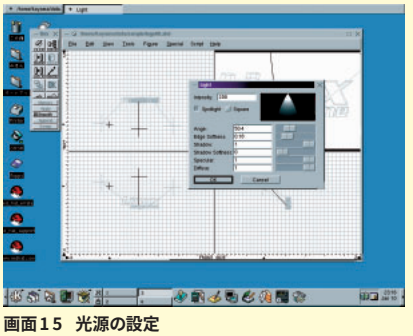
画面17 光源の種類

画面18 背景の貼り付け
Linux magazineのロゴに空のイメージを貼りつけてみた。なんだか、どこかのOSの起動画面のようになってしまった。

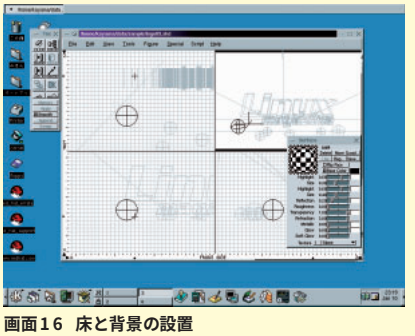
シーン構成　実際の撮影と同じで、カメラアングルや光源の設定を行うことは3DCGイメージの作成にとって重要な要素だ。バッチリと決めてレンダリングに移ろう。



画面14 視野の設定
視野の設定は透視図を確認しながらカメラウィンドウで行う。設定は実際にカメラを構えてファインダを覗くような感覚で操作できる。カメラについての知識があれば設定も容易だろう。



画面15 光源の設定
光源にはスポットライトを選択。3カ所からそれぞれ異なる色の光を当てる設定にしてみた。スポットライトだけでは暗そうなので無限遠光源も弱めに残しておく。



画面16 床と背景の設置
ロゴだけでは寂しいので、大理石の床とコンクリートの壁を配置する。小物も欲しいので、とりあえず透明の球(ガラス玉をイメージ)を数個床に配置してみた。

レンダリング

以上の作業が完了すると、あとはレンダリングを行うだけだ。「Shade for Linux Preview Kit」の場合、レンダリングできるイメージの最大解像度は4000×4000ピクセルとなっている。Windows版およびMacintosh版の「Shade Professional」では作成できるイメージサイズは無制限だが、ミッドレンジの「Shade Personal」に相当するであろうLinux製品版も、おそらくこのままの仕様を継承するものと思われる。しかし、4000×4000ピクセルのサイズがあれば商業印刷などの用途にもたいていの場合は対応可能だろう。

Shadeでは、レンダリング手法をスキャンライン法（Scan Line）・レイトレーシング法（Ray Tracing）・分散レイトレーシング法（Distribution Ray Tracing）という3種類の方法の中から選択できる。

スキャンライン法を用いれば高速なレンダリングが可能となるが、物体の影や映り込み、屈折による透過像の歪みなどは表現できない。レイトレーシング法は光線追跡を行うため、物体の影や映り込み、屈折による透過像の歪みの表現が可能となる。分散レイトレーシング法はレイトレーシング法の精度をさらに高めたもので、物体の柔らかな影や被写界深度、表面材質の粗さによる反射像の乱れなども表現でき、モアレの発生も低減することができる。ただし、精度が高い反面、分散レイトレーシング法はレンダリングに大変長い時間を要するため（マニュアルによると、レイトレーシング法の約10倍以上のレンダリング時間が必要）このあたりは目的に応じた使い分けが必要となるだろう（**画面22**）。

高速なレンダリング処理のためには、浮動小数点演算を高速に処理できるPCが必要となる。もちろんメモリ容量も多いにこしたことはない。メモリが不足すると、処理はHDD内の仮想メモリに取って代わられてしまう。こうなると演算処理のパフォーマンスは極端に低下し、ほとんど実用には適さない。ちなみに、PCに装着されているグラフィックボードはレンダリング速度にはほとんど影響しないそうだ。

レンダリング手法のほかの設定としては、曲面の分割方法・アンチエイリアシングの有無・影の表示・背景の表示・背景の反映などの項目を選択することができる。当然、レンダリングの精度を上げたり表示オプションを増やしたりすると、レンダリングに要する時間は長くなる。

これらの設定を済ませれば、レンダリングそのものは3DCGソフトとPCが行ってくれる。あとはコーヒーでも飲みながら一服して、画像の完成を待つだけである。

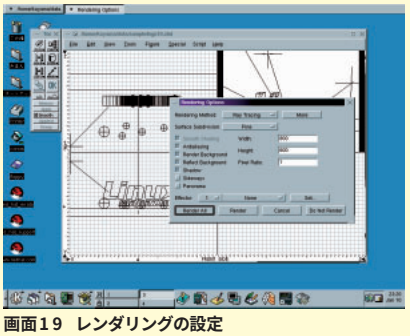
スキャンライン法

レイトレーシング法

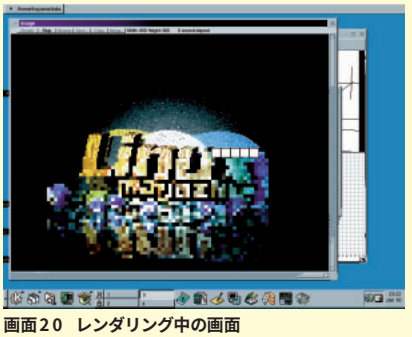
分散レイトレーシング法

画面22　レンダリング手法による生成画像の違い

レンダリング　レンダリングの設定を済ませればあとは画像の完成を待つだけ。「Shade for Linux Preview Kit」では最大解像度4000×4000ピクセルのイメージをレンダリングすることができる。



画面19　レンダリングの設定
影の映り込みなどを反映させたいので、レンダリング手法はレイトレーシング法を選択。曲面の分割は最も細かいを選び、影や背景もすべて反映させるように設定した。



画面20　レンダリング中の画面
レンダリングを開始するとモザイク状の画像の精度が徐々に上がっていく。レンダリング速度はPCの性能やメモリ容量に大きく左右される。また、画像サイズが同じでも画面が細かいほど時間はかかる。

画面21　完成
レンダリングが終了すれば画像は完成。画像はBMP・PICT・Targaのフォーマットで保存することができる。さらに細かな作業にはGIMPなどのフォトレタッチソフトを用いるとよい。

その他の機能

以上、紹介したのはShadeが持つ多彩な機能のうちのごく一部である。

製品には、印刷したマニュアルは付属していないので、CD-ROMに収録されているPDF形式のマニュアルを利用する。

PDFマニュアルの内容は、以下のように膨大で、すべてに目を通すだけでも一苦労だ。

- **ユーザーガイド（249ページ）**
- **チュートリアル（193ページ）**
- **アニメーション・チュートリアル（119ページ）**
- **プラグインユーザーマニュアル（29ページ）**
- **コマンド／プロパティ一覧（45ページ）**
- **スクリプトリファレンス（213ページ）**
- **クイックリファレンス（4ページ）**

■アニメーション

「Shade for Linux Preview Kit」は、もちろんアニメーションにも対応している。機能的には、形状に組み込まれたジョイントの動きによるジョイントアニメーションと、カメラモーションによるフライスルーアニメーションの2つの操作が可能だ。モーション設定インターフェイスでは、無制限のキーフレーム、複数ジョイントの同時表示／変更／連動、ジョイント可動範囲の設定、インバースキネマティクスによるモーション設定、自由曲面内の線形状単位でのジョイント値適用などが可能で、3D形状間の変形や表面材質の変化を補間するモーフィングも行える。

アニメーションレンダリングした結果は、フレームごとの番号の付いたイメージファイルとしてBMP・PICT・Targaのいずれかの形式で保存できる。

■スクリプト

Shadeには外部から制御できるスクリプト機能が搭載されている。これにより、直接スクリプトを記述することで、モデリングなどの高度な処理を自動化することも可能となる。なお、スクリプト言語はTclをサポートしている。

■プラグイン

Shadeは、形状作成／変形を自在に行うモデリングプラグイン、多様なフォーマットをサポートするインポート／エクスポートプラグイン、イメージ作成やエフェクトなどの画像処理プラグインなどを追加できるプラグインインターフェイスをサポートしている。

さらに、メタボール、正多面体作成、3D形状の自由変形／ラティス変形、VRML2.0、POV-Ray 3.0エクスポートなどさまざまなプラグインが標準で添付されている。

また、付属のプラグインSDKを用いることで、C/C++によるユーザー独自のプラグイン開発や配布も可能だ。今回のLinux版リリースの背景には、こうした開発者向けツールとしてのポジショニングを強く感じる。

あまりに多彩な機能とマニュアルのボリュームにちょっと気後れしてしまいそうだが、今回Shadeを使ってみた感想は、初心者にも決して敷居の高い3DCGソフトではないということだ。

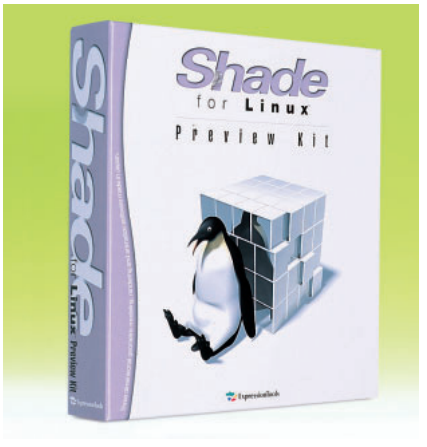
このあとリリースされる製品版はミッドレンジの「Shade Personal」相当になると見られるが、せっかくのLinux版なのだから、フラッグシップモデルに搭載されているネットワークレンダリング機能などにも対応してもらいたいものだ。

Linux magazineロゴ（画面21）のレンダリングには約24分を要した。

○テスト環境
システム：自作マシン
CPU：AMD-K6 233MHz
メモリ：64Mバイト
ハードディスク：6.4Gバイト

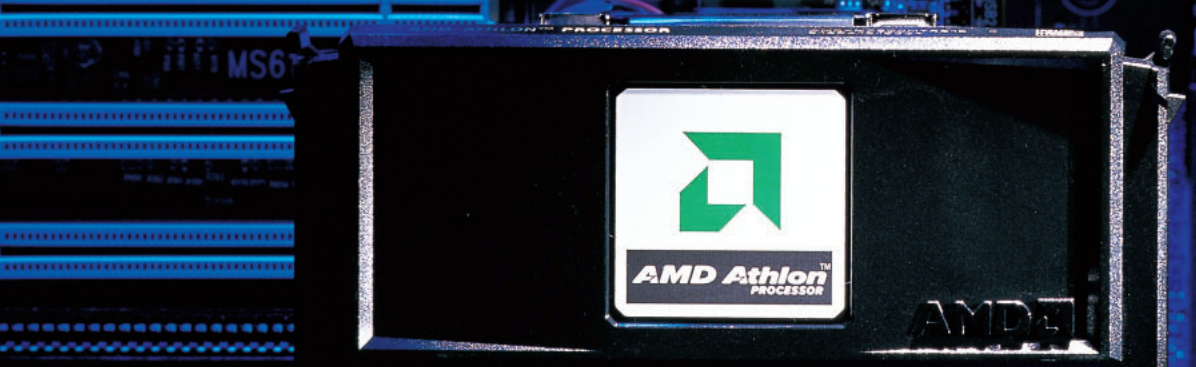
レンダリング時間：1394秒
Size：800×600pixels
Pixel Ratio：1
Rendering Method：Ray Tracing
Surface Subdivision：Verify File
■Smooth Shading
■Antialiasing
■Render Background
■Reflect Background
■Shadow
Effect：None

「Shade for Linux Preview Kit」
製品情報ページ
http://www.ex-tools.co.jp/linux/
動作が確認されたディストリビューション、デスクトップ環境などの最新情報はWebページ上に公開されている。



「Shade for Linux Preview Kit」
9800円
エクス・ツールス株式会社
「Shade」の正規登録ユーザーは、『簡易パッケージ版Shade for Linux Preview Kit』をユーザー優待価格の4900円で同社の通信販売サイトより購入することができる。

最新プロセッサ&マザーボードを選ぶ



１９９９年はいろんな意味で激動の年だったが、ＰＣ互換機のハードウェアについても大きく激しい変化が生じた。特にプロセッサの性能は大幅に向上しつつも価格は下がり、互換製品を含むプロセッサ／チップセットの勢力地図は大きく塗り変えられた。製品数も増えたため、ユーザーにとっては選択肢が増えた分、かえって選び難くなったように感じられる。そこで今月は、最新x８６プロセッサとそのマザーボードについて整理・解説し、Linuxでの活用方法を探ってみた。ＰＣ自作派の方だけではなく、メーカー製ＰＣのユーザーにもハードウェアのアップグレードなどに役立てば幸いである。

Photo : Kinouchi Kenji (Dee)

プロセッサ&マザーボード最新事情

最も多いLinuxプラットフォームであるIBM PC/AT互換機は、唯一（現実的に）エンドユーザーでもプラットフォームを自作できるマシンといえる。Linuxを動かすためにPCを自作するユーザーも多いことだろうが、それには最も重要なパーツであるプロセッサとマザーボードの知識が欠かせない。ここではデスクトップPCを対象に、x86プロセッサとマザーボード、メモリなどの最新事情をまとめてみた。予算に合った最新Linuxマシン作りにチャレンジしてみよう。

三つ巴のx86プロセッサからどれを選ぶ？

現在デスクトップPC用x86プロセッサは、大雑把にいって、Intel製のPentium II／III、CeleronのグループとAMD Athlon、そしてAMD K6シリーズを中心とするSuper7[*1]対応プロセッサのグループと3つに分類できる。ただしPCを自作する場合、Super7マザーボードの種類は少なく、互換性問題も比較的多い。特に理由がない限り、自作PCには残りの2グループからプロセッサを選ぶことになる。

■Pentium IIIかCeleronか？

現在Intelのプロセッサを購入するなら、Pentium IIIかCeleronが対象になる（Pentium IIは旧製品で入手も難しい）。一般に性能重視ならPentium III、価格重視ならCeleronを選ぶ。ただ、両者の元になったP6アーキテクチャ[*2]は同一である。

Pentium IIIについては、1999年秋に登場した新型Pentium III（**表1-1**にてコードネームがCoppermineであるプロセッサ）と、それ以前の旧型Pentium III（コードネームKatmai）が市場で混在している。新型は旧型に比べて、プロセスルール[*3]を0.25μmから0.18μmに下げ、2次キャッシュ[*4]の容量を半分にしてプロセッサコア[*5]に統合した。性能は新型のほうが良いし、旧型はそのうち市場から消えるはずだ。しかし、現時点では同じクロックで新旧2種類のPentium IIIが存在するから、間違って購入しないよう気を付けたい。クロック表記が「600EMHz」という具合に「E」が付いているのが新型Pentium IIIである。同様に「B」はFSB[*6]が133MHzであることを意味する。新型Pentium IIIはコア電圧[*7]が下がっており、またマザーボードのBIOS更新も必要なので、旧型Pentium IIIが動くマザーボードでそのまま新型Pentium IIIが動くとは限らない。

現在のCeleron（コードネームMendocino）も新型Pentium IIIと同じく2次キャッシュを内蔵するが、プロセスルールは0.25μmで旧型Pentium IIIと同じだ。また新型Pentium IIIに比べて2次キャッシュの容量は半分、かつ、プロセッサコアとの接続バス幅も狭く速度も劣る。FSBのクロック周波数も新型Pentium IIIが100MHz～133MHzなのに対し、Celeronは66MHzである。またマルチメディア命令のSSEもCeleronには実装されていない。当たり前だが性能も同クロックならCeleronのほうが劣る。Pentium IIIは2個までのSMP[*8]に対応しているが、Celeronは基本的にSMP非対応である（**写真1-2**の変換アダプタ次第では可能だが）。

しかし、まもなくIntelは新型Pentium IIIと同じアーキテクチャのCeleronを出荷するはずだ。2次キャッシュ容量こそ128Kバイトのままだが、2次キャッシュの速度向上とSSEの実装がCeleronに受け継がれ、新型Pentium IIIとの差別化要因は少なくなるはずだ。

■パッケージの違い

表1-1のように、Pentium IIIやCeleronには複数のパッケージがある。**写真1-1**のSlot1に装着できるのは「S.E.C.C.」と「S.E.C.C.2」、「S.E.P.」の3種類である。一方、**写真1-2**のSocket370に装着できるのは「FC-PGA」と「PPGA」である。ただしFC-PGAのPentium IIIとPPGAのCeleronでは、信号ピンの配置が一部異なるので、すべてのSocket370マザーボード／変換アダプタでPentium IIIが動作するわけではない。空冷ファンについても、この5種類のパッケージごとに寸法など

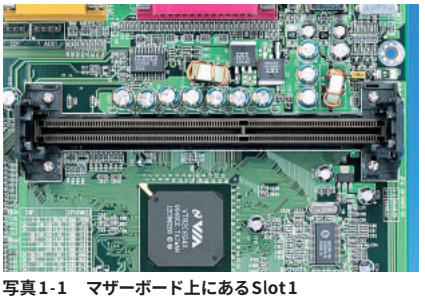


写真1-1　マザーボード上にあるSlot1
Pentium IIとともに登場したSlot1は、そのままの形状でPentium IIIでも利用されている。SC242とも呼ばれる。SMPはデュアルまで対応する。パッケージ形状によってスロット両脇に配置されているガイドレールの形状を変える必要があるので注意したい。

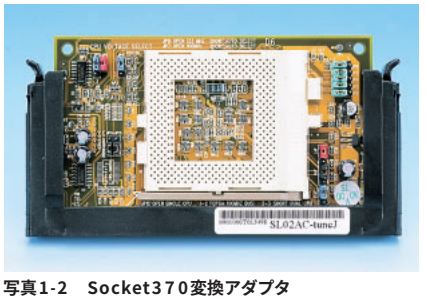


写真1-2　Socket370変換アダプタ
PPGAやFC-PGAのプロセッサをSlot1に装着するのに用いる。写真はSOLTEK製SL-02A++で購入価格は2980円。FC-PGAのPentium IIIにも対応する。公式にはSocket370でSMPはサポートされていないが、実際にはこうしたアダプタで実現できる場合がある。

が異なるので、購入時に間違えないよう注意したい。

■AMD Athlonの台頭

1999年、x86プロセッサで最大の話題はAMD Athlonだろう。これまでローエンドでしかシェアを確保できなかったAMDが、ハイエンドでもAthlonで一定の成功を収めたからだ。クロック周波数だけでもAthlonはPentium IIIに優ることがたびたびあったし、これまでのAMD製プロセッサの弱点とされていた浮動小数点演算も、Athlonは同クロックのPentium IIIと同等以上の性能を発揮する。またPentium IIIのSSEに相当するEnhanced 3DNow!というマルチメディア命令も実装している。このようにPentium IIIと比べて、Athlonの性能や機能は特に遜色なく、大手メーカー製PCにも採用され始めている。現時点ではSMP対応チップセットがないため、デュアルAthlon構成のマシンを組めないのが惜しい。

AthlonのパッケージはPentium IIIのS.E.C.C./S.E.C.C.2によく似ているが、コネクタ形状や信号の電気的特性はまったく異なる。AthlonにはSlotAと呼ばれるコネクタを装備した専用マザーボードが必要だ（もちろんPentium IIIやCeleronは装着できない）。

Athlonで自作PCを組む際に問題となったのは、高い消費電力のせいでマザーボードや電源ユニットの種類によっては起動できないというトラブルだった。購入時にはAthlonで動作するか事前に調べておきたい。ただし、Athlonのプロセスルールは当初0.25μmだったのが現在は0.18μmに置き換わりつつあり、消費電力も下がってきたので、こうしたトラブルは生じにくくなっているはずだ。購入するなら新しい0.18μmの製品を選びたい。

Glossary

＊1　Super7
Pentiumの時代に標準プラットフォームだったSocket7をベースに、バス速度を高めるなどの拡張が施されたプラットフォーム。AMD K6-2／IIIなど互換プロセッサが対応する。

＊2　P6アーキテクチャ
Pentiumに続いて開発されたPentium Proというx86プロセッサのコアアーキテクチャ。

＊3　プロセスルール
半導体集積回路におけるトランジスタ間の間隔．一般にこの数値が小さいほど集積度が高く、高速あるいは低消費電力になる。

＊4　2次キャッシュ
メモリシステムの高速化技法であるキャッシング（caching）に使われるメモリのうち、プロセッサコアから見て2番目に近いキャッシュを指す。

＊5　プロセッサコア
命令の解析・実行を司るユニット，レジスタなど、プロセッサの中核部分のこと。単一の半導体集積回路にまとめられることが多い。

＊6　FSB（Front Side Bus）
プロセッサから見て、メインメモリやI/Oデバイスを接続する側のバス。チップセットとの接続バスと考えればよい。

＊7　コア電圧
プロセッサコアを駆動する電力の電圧。外部インターフェイスの電圧とは別々に設定できる。

＊8　SMP（Symmetric MultiProcessor）
複数のプロセッサが基本的に同等なものとして振る舞うことができるマルチプロセッシングの方法。

内部クロック	名称	FSBクロック	パッケージ	2次キャッシュ	コードネーム
Pentium III					
450MHz	Pentium III-450MHz	100MHz	S.E.C.C./S.E.C.C.2	外付け、512Kバイト	Katmai
500MHz	Pentium III-500MHz	100MHz	S.E.C.C./S.E.C.C.2	外付け、512Kバイト	Katmai
	Pentium III-500EMHz	100MHz	FC-PGA	内蔵、256Kバイト	Coppermine
533MHz	Pentium III-533BMHz	133MHz	S.E.C.C.2	外付け、512Kバイト	Katmai
	Pentium III-533EBMHz	133MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
550MHz	Pentium III-550MHz	100MHz	S.E.C.C./S.E.C.C.2	外付け、512Kバイト	Katmai
	Pentium III-550EMHz	100MHz	S.E.C.C.2／FC-PGA	内蔵、256Kバイト	Coppermine
600MHz	Pentium III-600MHz	100MHz	S.E.C.C./S.E.C.C.2	外付け、512Kバイト	Katmai
	Pentium III-600BMHz	133MHz	S.E.C.C.2	外付け、512Kバイト	Katmai
	Pentium III-600EMHz	100MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
	Pentium III-600EBMHz	133MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
650MHz	Pentium III-650MHz	100MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
667MHz	Pentium III-667MHz	133MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
700MHz	Pentium III-700MHz	100MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
733MHz	Pentium III-733MHz	133MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
750MHz	Pentium III-750MHz	100MHz	S.E.C.C.2／FC-PGA	内蔵、256Kバイト	Coppermine
800MHz	Pentium III-800MHz	100MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
	Pentium III-800EBMHz	133MHz	S.E.C.C.2	内蔵、256Kバイト	Coppermine
Celeron					
266MHz	Celeron-266MHz	66MHz	S.E.P.	なし	Coppermine
300MHz	Celeron-300MHz	66MHz	S.E.P.	なし	Coppermine
	Celeron-300AMHz	66MHz	S.E.P./PPGA	内蔵、128Kバイト	Mendocino
333MHz～433MHz	Celeron-333MHz～433MHz	66MHz	S.E.P./PPGA	内蔵、128Kバイト	Mendocino
466MHz～533MHz	Celeron-466MHz～533MHz	66MHz	PPGA	内蔵、128Kバイト	Mendocino

表1-1　Pentium IIIとCeleronの種類

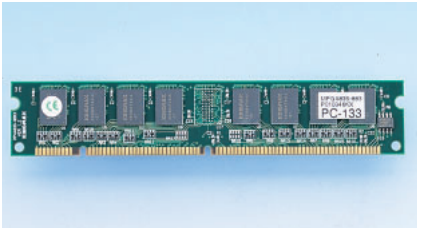


写真1-3　PC133 SDRAM DIMM
写真は実際に購入したノーブランド品。128Mバイトの価格は2000年1月下旬の時点で平均1万7000円前後。PC100の平均1万5000円前後に比べ、そう割高ではない。DIMMの右端にある小さなチップはSPDと呼ばれるROMの一種で、各種パラメータを保存している。

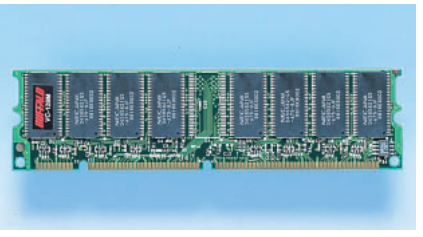


写真1-4　PC133 VC SDRAM DIMM
写真はメルコ製VC-128Mで、容量は128Mバイトで ECCはない。実売価格は3万4000円〜3万9000円程度（2000年1月下旬時点）と、PC133 SDRAM DIMMの2倍前後もする。以前は2万7000円程度だったが、最近値上げになった。

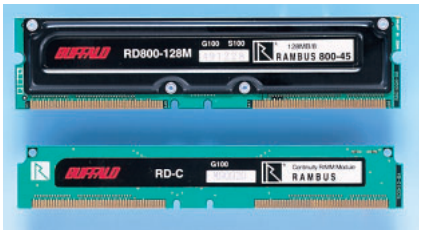


写真1-5　PC800 RIMM（上）とContinuity RIMM（下）
どちらもメルコ製でRIMMはRD800、Continuity RIMMはRD-C。Continuity RIMMはマザーボードに付属することが多い。RD800の実売価格は、8万6000円〜9万円前後とSDRAM DIMMなどに比べ非常に高価であり、このままでは普及は望めないだろう。

PC100 SDRAMの後継となるメモリは？

マザーボードに装着するデバイスのうち、プロセッサと並んで重要なパーツといえばメインメモリである。現時点でデスクトップPCに使えるメインメモリ用のメモリモジュールは、SDRAM系のDIMMと最近登場したDirect RDRAMのRIMMという2種類に大別できる。

■主流はSDRAM DIMM

最もよく使われているのはPC100 SDRAM DIMMである。PC100とは規格の名称で、100MHzのクロックで駆動するSDRAM DIMMの仕様を規定している（最大転送レートは800Mバイト/秒）。現在流通しているほとんどのマザーボードはこれに対応しており、一番つぶしが効くメモリモジュールといえる。これに追随するように133MHz駆動のPC133 SDRAM DIMMも広まり始めている（**写真1-3**）。というのも、FSBが133MHzの新型Pentium IIIにチップセットが対応すると同時に、PC133も使えるようになったからだ（最大転送レートは1.06Gバイト/秒）。

PC100／133ともに、容量およびECC[*9]の有無、CASレイテンシ[*10]の値、Unbuffered／Registered[*11]どちらのタイプか、といった点を購入時に指定する必要がある。もっともデスクトップPC用ならタイプはUnbufferedで間違いないし、また現時点でCASレイテンシはPC100ならCL=2、PC133ならCL=3と指定すればよい。

SDRAM系で変わり種なのは、**写真1-4**のVC SDRAM DIMMである。これは日本電気が開発したVirtual Channelという技術を適用したSDRAMを使用しており、通常のSDRAMより高速に動作するのが特徴だ。ただし、チップセットおよびBIOSがVC SDRAMに対応していないと速くならないこと、またメモリチップのベンダーが限られているため、現時点ではSDRAMの次の主流になるとは考えにくい。ただ、Virtual Channel自体はSDRAMに依存しないので、次の主流メモリにも組み込まれる可能性はある。

■Direct RDRAMは普及するか？

IntelとRambusが協力して開発したDirect Rambusというメモリバス規格がある。これを採用してSDRAMの次世代メモリを狙ったのがDirect RD

メーカー名	AMD	Intel	Intel	Intel	VIA Technologies
名称	AMD-750	440BX	i810E	i820	Apollo Pro133A
メモリコントローラ	AMD-751	82443BX AGPset	82810E GMCH	82820 MCH	VT82C694X
I/Oコントローラ	AMD-756	82371AB PIIX4E	82801 ICH	82801 ICH	VT82C686A
対応CPU	Athlon	Celeron／Pentium II／III	Celeron／Pentium II／III	Pentium II／III	Celeron／Pentium II／III
FSB	200MHz	66／100MHz	66／100／133MHz	100／133MHz	66／100／133MHz
メモリ	SDRAM（PC100）	SDRAM（PC100）	SDRAM（PC100）	Direct RDRAM（PC600／700／800）	SDRAM（PC100／133）、VC SDRAM（PC100／133）
AGP	2x	1x	—	4x	4x
PCIバス	Rev.2.2	Rev.2.1	Rev.2.2	Rev.2.2	Rev.2.1
IDE	UltraDMA/66	UltraDMA/33	UltraDMA/66	UltraDMA/66	UltraDMA/66
USB	4ポート／OHCI	2ポート／UHCI	2ポート／UHCI	2ポート／UHCI	4ポート／UHCI
AC'97	—	—	○	○	○

表1-2　各チップセットの主な仕様
メモリコントローラはNorth Bridge[*13]、I/OコントローラはSouth Bridge[*13]に相当する

RAMで、最大転送レートは1.6Gバイト/秒とPC100 SDRAMの2倍の性能を持つ。対応チップセットはIntelがすでにデスクトップPC向けにi820、ワークステーション向けにi840を出荷中だ。

Direct RDRAMをメモリモジュール化したのがRIMM（**写真1-5**）で、現在はPC600／700／800という3種類の規定がある。「PC」に続く数値は1ワード16ビットのデータを転送する際の転送レートに相当し、PC800の転送レートは800Mワード／秒×16ビットで1.6Gバイト／秒となる。また、**写真1-5**にあるContinuity RIMMとは、バス結線用のダミーカードで、RIMMを挿さないスロットには必ずこれを装着する必要がある。

RIMMが次の主流になるかというと、デスクトップPC用メモリとしては現時点で対応チップセットがi820しかない、RIMMの価格が高すぎる（**写真1-5**参照）、2スロットしか使えない、など問題が残っている。DRAMメーカーの動向やチップセットの対応次第では、DDR SDRAM*12に主流の座を奪われるかもしれない。

440BXの後継となるチップセットは？

次は、マザーボードの機能と性能を大きく左右するチップセットの動向をチェックしてみる。

Celeron／Pentium II／III用チップセットとしては、過去2年間、Intel製の440BX（**写真1-7**）が標準的に使われてきた。**表1-2**でほかのチップセットと比べると、さすがに仕様は古くさいが、133MHz FSBやPC133といった新しい機能が必要ないなら、古いが安定性に優れる440BXマザーボードを選ぶという手はある。別個にIDEコントローラを搭載してUltraDMA/66をカバーしている例もある。

写真1-6　AMD-750
これまで唯一のAthlon用だったAMD製のチップセット。FSBは200MHzと高速だが、メモリはPC100までである。SMPには対応していない。

写真1-7　440BX
2年近く前から現役という珍しく長生きしているIntel製チップセット。UltraDMA/66非対応と仕様は古くさいが、安定性には定評がある。

写真1-8　i820
初めてDirect RDRAMをサポートしたIntel製チップセット。最新の各種規格をサポートするほか、PC100 SDRAMもオプションで接続できる。

写真1-9　Apollo Pro133A
PC133 SDRAMをサポートする数少ないチップセット。VIA Technologiesが開発・製造しており、AGP 4xやUltraDMA/66、USB×4ポートにも対応する。

写真1-10　i810E
今回唯一のグラフィックス統合チップセット。Intel製で、133MHzのFSBやUltraDMA/66対応など、i820の機能を先取りしていた。

1998年登場の440BXに対して、1999年に登場した**写真1-10**のi810（E）はIntel初のグラフィックス統合チップセットである（末尾のEは133MHz FSBに対応したモデル）。これはコスト最優先のサブ1000ドルPCによく採用され、ローエンド向けチップセットとしての地位を確立した。なお、SiS620／630のように競合するチップセットがIntel以外からも登場している。

一方、ミッドレンジからハイエンドをカバーすべく開発されたのがi820（**写真1-8**）で、Direct RDRAMやAGP 4xなどを（Intelとして）初採用しているのが目玉だ。しかし前述のようにRIMMの価格が高すぎるため、現時点で440BXの後継としてミッドレンジをカバーするには至っていない。i820はMTH（Memory Transfer Hub）というチップを追加してPC100 SDRAMを接続できるが、PC133にはまだ対応していないというデメリットもある。

440BXの後継となり得るチップセットには、皮肉にも互換チップセットメーカーであるVIA Technologies製の

Glossary

＊9　ECC（Error-Correcting Code）
ビット化けなどのエラーを訂正するために，本来のデータとは別に付加されるコードのこと．

＊10　CASレイテンシ
SDRAMへのアクセスタイミングを決めるパラメータのひとつで、カラムアドレスを与えてから実際にデータが転送されるまでの時間。短いほど高速である。CL=？という具合に、クロック数で表記される。

＊11　Unbuffered／Registered
メモリチップとメモリコントローラの間に何も回路を入れないのがUnbufferedタイプで、レジスタと呼ばれる回路を挿入したのがRegisteredタイプ。

＊12　DDR SDRAM（Double Data Rated SDRAM）
クロック信号の1周期で2回データを転送する方式で、クロック周波数を上げずにデータ転送レートだけを高めたSDRAM。

＊13　North Bridge／South Bridge
チップセットのうち、CPUのバス（FSB）とPCIなどの拡張バスとの間を橋渡しするチップをNorth Bridgeと呼ぶ。South BridgeはPCIバスなどからISAバスやペリフェラルバスの橋渡しを司る。メモリコントローラはNorthに、USBやIDEのコントローラはSouthに含まれることが多い。

Apollo Pro133A（**写真1-9**）が最も近いといえる。133MHz FSBはもちろんPC133 SDRAMやAGP 4x、UltraDMA/66をサポートし、その割に採用マザーボードの価格も高くはない。ALiやSiSといったほかの互換チップセットメーカーも、同じセグメントのチップセットを開発または出荷しており、i820が足踏みしている間に互換チップセットの勢力が拡大しているといえそうだ。

Athlonについては、これまで唯一のチップセットだったAMD-750（**写真1-6**）に加えて、VIA TechnologiesからApollo KX133というチップセットが量産出荷中である。

IntelのX86プロセッサもAthlonも、重要なチップセットは今年半ばに登場しそうだ。IntelやVIAは、グラフィックス統合チップセットながらAGPインターフェイスも装備するチップセットを出荷予定しているという。どちらもPC133 SDRAMに対応するらしく、PC自作派にとっては魅力的なチップセットになりそうだ。Athlonについても、VIAは同様なチップセットを予定している。またAMDはハイエンド向けにSMPをサポートしたAthlon用チップセットを予定しているという。

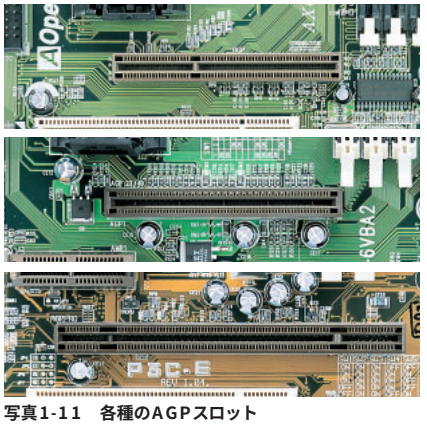


写真1-11　各種のAGPスロット
一番上が最初に登場したAGPスロットで、3.3Vの信号振幅に対応する。真ん中は1.5Vと3.3Vの両方に対応したAGPスロットで、AGP 4xをサポートする（4xは1.5V振幅でのみ利用できる）。一番下はワークステーション向けに規定されたAGP Proのスロット。

そのほかのマザーボード選びのポイント

ここではチップセット以外にマザーボード選びのポイントとなるものに触れておこう。

■フォームファクタ

マザーボードの寸法および形状には規格がある。代表的なのはATXで、日本では最もケースの種類の多いフォームファクタだ。小型ケースにしたいならmicroATXが向いている。もっと薄型のケースを望むならNLXという手もあるが、これはケースとセット販売がほとんどで、自由度が低い。

■拡張バス

最近の傾向としては、ISAスロットを廃してPCIスロットを少しでも増やす製品が多く、ISAカードをどうしても使いたいユーザーは注意したい。また440BXの場合、PCIスロットが6本あると、特殊な工夫がない限り1本はバスマスタで動作しない。しかし今時バスマスタなしで動くPCIカードは珍しく、これも注意が必要だ。

最近、AGP 4xモードをサポートしたグラフィックスカードが増えてきたが、4xを利用するにはAGPスロットが1.5Vの信号振幅をサポートしている必要がある（**写真1-11**）。

ISAスロットがなくなった代わりにAMRスロット（**写真1-12**）を搭載する機種も多い。しかし、AC'97*14対応サウンドコントローラはチップセット内に、またAC'97対応コーデック回路はマザーボード上に実装という例が多く、AMRスロットはモデムカード専用だったりする。つまりモデムが不要ならAMRに執着する必要はない。

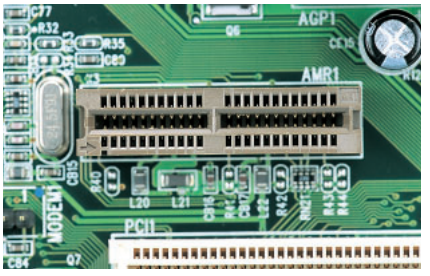


写真1-12　AMRスロット
AMRとはAudio/Modem Riserの略。サウンド／モデム機能のうちアナログ信号を扱う回路を拡張カード上に装備することで、デジタルノイズがアナログ信号に入り込むのを避けるしくみだ。AC'97対応チップセットと組み合わせて使うのが一般的だ。

■ジャンパ設定を減らす設計

プロセッサのクロックやコア電圧などの設定は、なるべく自動で最適に設定できるようにし、手動変更はBIOSセットアップから行う、というマザーボードが増えている。これでマザーボード上のジャンパで設定する手間が省けるので扱いやすくなる。特にPC自作の初心者には便利だ。

■マザーボードのリビジョン

細かい修正を重ねてリビジョンを高めていくのは、ソフトウェアだけではなくマザーボードにも該当する。同じ型番のマザーボードでも初めて世に出たころに比べ、それより新しいリビジョンでは、より多くの機能が使えたり、プロセッサやメモリ、拡張カードなどアドオンデバイスのサポート範囲が拡大されたりすることがある。具体的には、BIOS用フラッシュメモリの容量を増やしたり、電源回路を強化して新型プロセッサに対応したり、といった例がある。基本的にはマザーボード購入時にリビジョンもチェックし、あまり古いものを選ばないよう注意すべきだ。

Glossary

＊14　AC'97（Audio Codec '97）
Intelが定めたサウンド機能の実装に関する規格。サウンド機能をデジタル／アナログに分離して取り扱っている。

テストした5機種のマザーボード

プロセッサやチップセットの種類が増えてきたせいもあって、マザーボードの種類はリテール向けに限っても非常に多い。今回は、95ページで紹介した5種類のチップセットごとに1枚ずつ選んでテストしてみた。

Athlonマザーボード

2000年1月下旬の時点でAthlonの形状は1種類（**写真2-1**）だけ、対応チップセットもつい最近までAMD-750だけという状況が影響しているのか、Athlon対応マザーボードはどれも構成がよく似通っている。異なるのは、ATXにまざってmicroATXがわずかに存在することや、South Bridge相当のチップがAMD-756だったりVIA VT82C686Aだったりするぐらいだ。ハイエンド向けマザーボードによくあるSCSIやEthernetインターフェイスを搭載したタイプも見かけないし、チップセットが未対応なのでデュアルプロセッサ対応タイプもない。しかしVIAが新しいAthlon用チップセットApollo KX133をすでに量産出荷しており、採用するマザーボードも増えるだろう。少なくともユーザーにとって選択肢は広がるはずだ。

Athlon用マザーボードの価格はおおよそ1万5000円～2万5000円ほどで、440BX搭載のslot1マザーボードより高めだ。しかし、Apollo KX133の投入で価格が下がる可能性もあるので期待したい。

Athlon用マザーボードのなかには、OEM向け製品をよく見かけるが、できれば保証などの条件がいいリテール向け製品を購入したい。また、AMDが検証したマザーボードや電源ユニットなどリストが同社Webサイトに掲載されているので、購入前に参照しておきたい（http://www.amd.com/japan/products/cpg/athlon/config.html）。

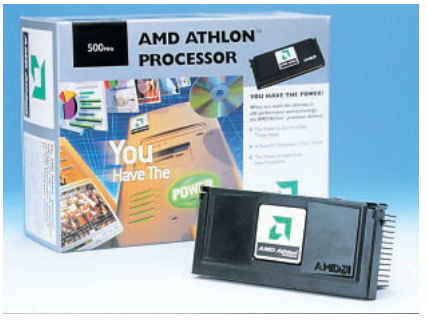


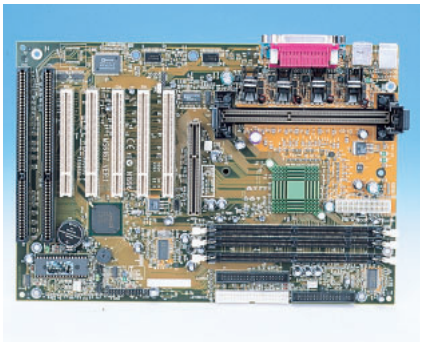
写真2-1　購入したリテール版Athlon
500MHzのモデルを2万5700円で購入した。パッケージには本体とファン付きヒートシンク、その留め具、マニュアルが同梱されている。リテール版よりバルク品＋ヒートシンクのほうが安価なこともあるので、どちらを選ぶか悩ましいところだ。

Micro-Star International

MS-6167

Athlon
AMD-750
ATX

購入価格：1万7800円
URL：　http://www.msi-computer.co.jp/



Micro-Star International（MSI）のMS-6167は、Athlonの発売開始当時、唯一の入手可能な対応マザーボードでもある。国内ではエム・エス・アイ コンピュータジャパンが日本向けリテール版を取り扱っており、今回はそれを使用した（日本語マニュアルは付属しない）。AMDの推奨マザーボードリストには、本機が800MHz対応製品として記されている。

チップセットには、AMD純正であるAMD-750が採用されており、South Bridgeも同社製（AMD-756）が使われている。メモリスロットはPC100対応のDIMMスロットが3本とごく普通の仕様だ。BIOSはAward Software製である。特徴的なのはUSBで、一般的なバックパネルの2ポートに加えて、フロントパネル用にも2ポートを備える（利用するには、外部コネクタを引き出すケーブルが必要だが）。IDEはUltra DMA/66をサポートしているが、AGPは2xモードまでで最新の4xには対応していない。サウンドなどの付加機能は装備しておらず、比較的ベーシックな構成といえる。

今回の記事には間に合わなかったが、MSIからAthlon用マザーボードとして、新たにK7Proが発表・出荷されている。MS-6167が6層基板で設計されているのに対して、K7Proは安価な4層基板で設計されており、よりコストパフォーマンスの高い製品になると思われる。

2000年1月下旬の時点でエム・エス・アイ コンピュータジャパンのWebサイトにはMS-6167およびK7Proに関する情報は掲載されていない。詳しい情報は、MSI（台湾）のWebサイト（http://www.msi.com.tw/first.htm）を参照されたい。

Slot1マザーボード

現時点でもっとも種類の豊富なマザーボードといえば、このSlot1マザーボードだろう。Celeronと同様Pentium IIIも、今後は**写真2-2**のようなソケット装着タイプに移行していくようだが、これは変換アダプタを用いればSlot1マザーボードに装着できる。その逆は無理なので、現時点でCeleron／Pentium III用ならSlot1マザーボードが無難な選択ともいえる。また、ここでは取り上げていないがSCSIやEthernetなどを搭載したタイプやデュアルプロセッサのタイプもあるので、選択肢は広い。フォームファクタもATXやmicroATXはもちろん、ベビーATやNLXなどもまだ見つけることができる。

主流のチップセットといえば、440BXとi820、そしてApollo Pro133（A）の3種類だろう。このうち440BXは最も古く製品数も豊富だ。しかし標準ではUltraDMA/66にも対応しないので、最近はUltraDMA/66対応IDEコントローラを別途搭載している製品をよく見かける。ただこうしたオンボードデバイスが増えるとIRQの共有によるトラブルが解消しにくいので、メリットばかりではない。

i820搭載マザーボードは、RIMMあるいはDIMMを装着する2つのタイプに分類できる（両方に対応したタイプも存在する）。あまりにもRIMMが高価な現状ではDIMMタイプを選ぶのが妥当だが、性能が落ちてしまいメリットが小さくなるのが難点である。

その点Apollo Pro 133（A）搭載マザーボードは、PC133 SDRAM DIMMや133MHz FSBを正式にサポートしているなど、現時点でi820や440BXより優位な点が目立つ。互換チップにありがちな互換性問題もほとんどないようで、最近急に採用マザーボードが増えている。ちなみに133Aと133の違いはAGP 4xに対応しているか否かだけといってよい。

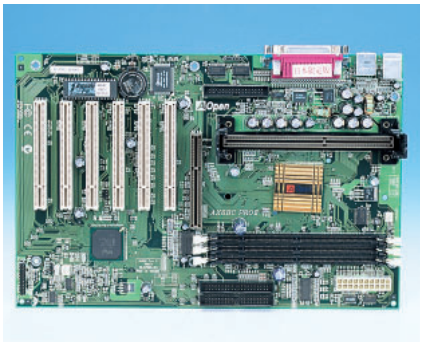
写真2-2　購入したリテール版Pentium III
FC-PGAで500MHzのモデルを2万5980円で購入した。Slot1に装着するには変換アダプタが別途必要なので、S.E.C.C.2を選ぶほうが割安である。このリテール版は化粧箱を一見しただけでは、何種類もあるPentium IIIを区別しにくいので注意したい。

AOpen

AX6BC TypeR V.spec II

Celeron
PentiumII／III
440BX
ATX

購入価格：1万5800円
URL：http://www.acer.co.jp/home/AOpen/



AOpenのAX6BC TypeR V.spec IIは、台湾など海外ではAX6BC Pro IIという名称で販売されているマザーボードである。今回は日本エイサーが日本語マニュアルをセットして販売している製品を試用した。本機はPentium II／IIIとCeleronをカバーし、新Pentium IIIもBIOSアップデートで対応できる。チップセットには定評のある440BXを採用するが、IDEはUltraDMA/33まで、AGPは2xまでしか対応していない。メモリスロットはPC100 DIMMスロットが3本と一般的な仕様だ。BIOSはAward Software製である。サウンド回路は搭載されていないが、PCIスロットは6本と多い。

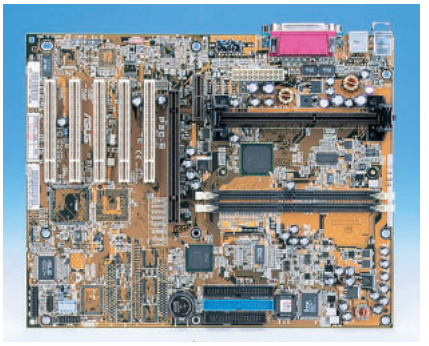
AX6BC TypeR V.spec IIの特徴のひとつにジャンパレスによる設計が挙げられる。FSBのクロック周波数やプロセッサ内部のクロック倍率はすべてBIOS上で設定可能であるほか、デフォルトの設定ではプロセッサのコア電圧が1.3～3.5Vの間で自動設定されるので、初めて自作するユーザーには扱いやすい仕様といえる。FSBのクロック周波数は標準的な66MHzと100MHzのほか、最大153MHzまで設定可能で、合計16種類の設定がある。ただし、搭載されている440BXのFSBはクロック周波数100MHz以下しか保証されていないので、それを超える周波数帯で動作させる場合は注意が必要だ。プロセッサのクロック倍率は、1.5倍から0.5刻みで最大8.0倍まで設定できる。また、最悪の場合プロセッサを破壊することもあるものの、プロセッサのコア電圧を通常時より0.1Vないし、0.2V増加させることも可能だ。そのほか、FSBのクロックに対してAGPのクロックを手動で1／1または2／3に設定するジャンパも用意されている。

ASUSTeK Computer

P3C-E

Pentium II／III
i820
ATX

購入価格：2万5700円
URL：http://www.asus.co.jp/



ASUSTeK ComputerのP3Cシリーズは、チップセットにi820を採用したマザーボード群である。試用したP3C-Eはオンボードデバイスを省いたマザーボードだ。基板サイズがほかのATXマザーボードにくらべて大きいのは、SCSIやEthernetを搭載するマザーボードP3C-L／S／LSと同じ基板を使い回しているせいである。

本機はSlot1を採用し、新旧Pentium IIIとPentium IIを利用できる。FSBのクロック周波数は、デフォルトの状態ではディップスイッチによって100～180MHzの間の29通りから選択できる。ジャンパ設定を変更すれば、BIOS上からソフトウェア的にFSBのクロック周波数を変更することも可能である。注意すべきは、100MHz未満のFSBクロック周波数がサポートされていないことだ。つまりFSBクロック周波数が66MHzである現在のCeleronやPentium II-333MHz以下のプロセッサは、動作保証範囲内で利用できない（むりやり100MHzで駆動できれば動かせないこともないようだが）。

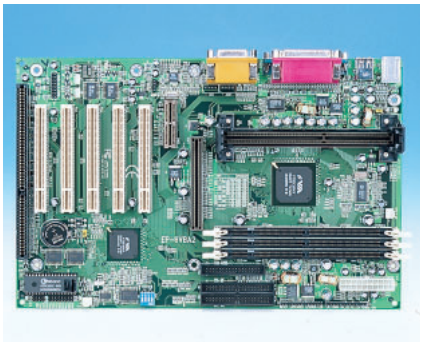
本機にはi820の特徴がよく表れている。まずメモリはPC800 RIMMを2枚まで装着できる。同社が販売するDR2 DIMM Riserを使えば、PC100 DIMMを2枚まで装着できる（RIMMとの同時使用は不可）。またAGPスロットも、AGP 4xだけではなくAGP Proにも対応している。ISAスロットは廃して、PCIスロットを6本と多めに装備している。もちろんIDEはUltraDMA/66対応だ。そのほか、チップセット内蔵のサウンド機能のために、音声入出力端子が標準装備されている。こうしたi820の機能を安価なDIMMと組み合わせて使うなら、DIMMを直接装着できるP3C2000を選ぶべきだろう。

EPoX Computer

EP-6VBA2

Celeron
Pentium II／III
Apollo Pro133A
ATX

購入価格：1万7480円
URL：http://www.mustardseed.co.jp/



EPoX Computerがラインアップする Slot1マザーボードのうち、このEP-6VBA2はVIA製チップセットを搭載した中では最新の製品である。今回は、EPoX Computer製マザーボードの国内総販売元であるマスタードシードが取り扱っている日本語版を使用した（日本語マニュアルが付属する）。

対応するプロセッサはCeleronとPentium II、新旧Pentium IIIである。プロセッサのクロック周波数は、ジャンパによってFSBのクロック周波数を、ディップスイッチによって倍率を指定することで設定する（ただし、倍率については、現在市販されているIntel製x86プロセッサだと内部で固定されているので、ディップスイッチでの設定は無視されてしまう）。FSBクロック周波数は、66～150MHzまでの12通りの中から選択できる。また、BIOS上でもFSBクロック周波数を設定可能で、ジャンパでの設定より優先される。

本機の特徴のひとつは、さまざまなタイプのDIMMが使えることだ。PC100 DIMMはもちろんPC133 DIMM、そしてVirtual Channel SD RAM DIMM（PC100およびPC133）も利用できる。またFSBが133MHzのときにメモリを100MHzで駆動できるなど、仕様は柔軟で使い勝手はよい。

本機もAGP 4xやUltraDMA/66はサポートしている。USBについては、合計4ポート装備しており、2ポートずつバックパネルとフロントパネルに装着できる（フロントパネルに取り付けるには外部コネクタを引き出すケーブルが別途必要）。またチップセットにはAC'97対応のサウンドコントローラが内蔵されているため、音声入出力端子がバックパネルI/Oに標準装備されている。

Socket370マザーボード

Celeronが登場した当初は、Pentium IIと同様にSlot1に装着できるパッケージ（S.E.P.）が用いられていた。しかしその後、コストダウンのために、S.E.P.の基板（サブストレート）を省いたPPGAパッケージのCeleron（**写真2-3**）が登場し、次第にS.E.P.パッケージを置き換えていったのである。その結果、PPGAパッケージを装着するSocket370は安価なCeleronの専用ソケットとして広まった。

こうした経緯から、Socket370マザーボードはCeleronとの組み合わせで安価なPCを組むことを主眼に置いた製品が多い。たとえば、Slot1マザーボードに比べると、チップセットにi810Eのようなグラフィックス統合タイプを採用する製品をよく見る。またmicroATXを採用して拡張スロット数を減らし、かわりにサウンドやEthernetをオンボードに実装する例も目立つ。いずれも性能や拡張性、構成の柔軟性より低価格を重視した仕様だ。

しかし、今後はこの傾向が変わってくるはずだ。なぜなら、Pentium IIIにもSocket370に（物理的に）装着できるFC-PGAパッケージが登場したからだ。今後はコスト重視の市場だけではなく、Pentium IIIが目指すハイエンド向けにもSocket370マザーボードが登場するようになるだろう。さらに、Pentium IIIは次第にFC-PGAに移行する予定なので、Slot1マザーボードを置き換えていくものと思われる。ただ、将来のPentium IIIがSocket370以外のソケットを採用する可能性もある。

また、CeleronとFC-PGAのPentium IIIの信号ピン配置は異なるため、Celeronしか想定していない従来のSocket370マザーボードでは、Pentium IIIは動作しない。これから購入するならPentium III対応製品を選びたい。

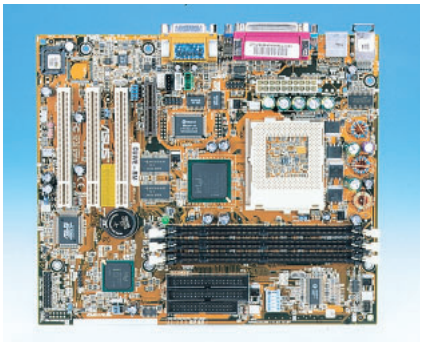
写真2-3　購入したリテール版Celeron
PPGAパッケージで500MHzのモデルを1万4800円で購入した。Pentium III同様、バルク品＋ヒートシンク付きファンよりリテール版のほうが安価なことが多いが、リテール版に同梱のファンは回転速度を検出できない廉価なタイプである。

ASUSTeK Computer

CUWE-RM

購入価格：1万7800円
URL：http://www.asus.co.jp/

Celeron
Pentium III
i810E
microATX



ASUSTeK ComputerのCUWE-RMは、Socket370に対応するmicroATXフォームファクタのマザーボードである。今回は並行輸入品を購入してテストした。ATXフォームファクタのCUWEという姉妹製品もラインアップされている。

プロセッサは、FC-PGA版のPentium IIIとPPGA版のCeleronに対応している。プロセッサのクロック周波数は、デフォルトではマザーボード上のディップスイッチで設定するが、ジャンパセットを変更すればBIOSからソフトウェア的に設定することもできる。FSBクロック周波数は66～166MHzまで24通り設定可能だ。

チップセットにはグラフィック機能が統合されたi810Eを採用し、画面描画に使われる4MバイトのSDRAMとともに実装されている。グラフィックスカードを別途必要としないのでコストを下げられる半面、グラフィックス回路を変更できないというデメリットもある。PC100対応のDIMMスロットは3本あり、最大512Mバイトまで増設できる。またIDEはUltraDMA/66に対応している。

PCIスロットの脇には、ヤマハ製サウンドチップを用いたサウンド回路の配線パターンがあるが、試用した製品では空きのままだった。標準状態では、i810内蔵のAC'97対応サウンド機能を利用することになる。

2000年1月下旬の時点では、ASUSTeKのWebサイト（http://www.asus.com.tw）にある製品紹介一覧から本機の製品ページにリンクが張られていない。しかし、同社のWebサイトでCUWE-RMをキーワードとして検索すれば、紹介ページは見つけられるので注意されたい。

最新マザーボードでLinuxをセットアップ

Linuxディストリビューションのインストーラは安全な設定を採用しているので、最新のハードウェアを採用したパソコンでもインストール自体は成功する場合が多い。しかし、実際にLinuxを起動した後になんらかのトラブルに遭遇する可能性は、最新のハードウェアの場合ほど多いのもまた事実だ。ハードディスクのUltraDMAを有効にできない程度ならばいいが、場合によってはハングアップする場合もありやっかいだ。

だが、この種の問題はLinuxカーネルを最新版にアップデートすることで解決できる場合が多い。たとえば新型Pentium III（以下、コードネームのCoppermineと記す）は、バージョン2.2.13までのカーネルでは正しく検出されず、2次キャッシュ容量が0Kバイトと認識されてしまう。しかし、安定版カーネルの最新バージョン2.2.14からCoppermineを検出するコードが追加されて、この問題は解決している。PCの自作市場でCoppermineと人気を二分するAMD Athlonも、バージョン2.2.12でサポートされた。

プロセッサと密接な関係をもつのがマザーボード上のチップセットだが、これも最新の製品だと同様の問題に直面することがある。PC133 SDRAM対応チップセットとして人気のVIA Technologies製 Apollo Pro133Aは、Linuxと組み合わせるとISA DMAに不具合が発生する場合があるとされ、これが解決されたのはカーネル2.2.13からである。

カーネルもプロセッサ＆マザーボードも最新に！

今回テスト対象に選んだハードウェアは、幸い最新のカーネル2.2.14においてほとんどサポートされている。標準では残念ながらサポートされないIDEコントローラのUltraDMA機能も、Unified IDEパッチと呼ばれるカーネル2.2.14へのパッチを適用することで、一部利用できるようになる。

また低価格帯PCでCeleronと組み合わせられることの多いi810Eチップセットはグラフィックス機能を内蔵しているが、これを利用できるXサーバは、一部のディストリビューションを除き、標準ではインストールされない。今回は、こうした問題への対処方法について解説する。

今回はあえてUSBサポートを有効にしていない。というのは、USB関連のドライバは開発版カーネル2.3系列でも頻繁に更新されていて、安定しているとはいいがたいからだ（カーネル2.4のリリースが遅れている一因でもある）。

Linuxを最新のデバイスを採用したPCに対してインストールする場合は、なるべく新しいディストリビューションを選びたい。Linuxカーネルをアップデートするには、カーネルが必要とするソフトウェアも、対応したバージョンをインストールしておく必要があるからだ。今回テストに用いたLinuxディストリビューションは、Debian GNU/Linuxをベースに開発されたCorel LINUX OSだ。しかし、今回の記事で扱う範囲内では、ほかのディストリビューションでも同様の手順でアップデート可能である。なお、Linuxカーネルが必要とするソフトウェアとそのバージョンは、Linuxカーネルを展開した場合に得られる/usr/src/linux/Documentation/Changesファイルに記述されているので、確認してみるといいだろう。

ファイル・情報	URL
カーネル2.2.14	ftp://ring.gr.jp/pub/linux/kernel.org/kernel/v2.2/linux-2.2.14.tar.bz2
Unified IDEパッチ	ftp://ring.gr.jp/pub/linux/kernel.org/kernel/people/hedrick/
i810E対応Xサーバ	http://support.intel.com/support/graphics/intel810/linuxsoftware.htm

表3-1　使用したファイルや情報の所在

■カーネルソースの入手と展開

LinuxカーネルはThe Linux Kernel Archivesから配布されている。ftp.kernel.orgは日本国内ではRINGサーバプロジェクトなどにミラーされているので、これらの入手先からカーネルソースをダウンロードする（**表3-1**）。

例として、/tmpにダウンロードしたカーネルを展開する場合を示す。作業はすべてroot権限で行う必要がある。

```
# cd /usr/src
# mv linux linux-old
# bzip2 -cd /tmp/linux-2.2.14.
tar.bz2 | tar xvf -
# mv linux linux-2.2.14
# ln -s linux-2.2.14 linux
```

■カーネルのコンフィグレーション

Linuxカーネルの再構築のため、必要な情報を用意するのがコンフィグレーションの目的だ。コンフィグレーション情報は/usr/src/linux/.configに書き込まれ、コンパイル時に参照される。このファイルは基本的に直接編集はせず、次のコマンドで起動されるコンフィグレーション画面で設定する。

```
# cd /usr/src/linux
# make distclean
# make menuconfig
```

menuconfigはコンソールベースの設定ツール（**画面3-1**）で、メニュー形式で表示されるため設定項目が一目でわかる。menuconfigの代わりにxconfigを指定するとX Window Systemベースの設定ツールが起動する（**画面3-2**）。xconfigを実行するにはTcl/Tkを必要とするので、もしxconfigを指定してエラーが発生した場合は、Tcl/Tkをインストールしているか、まず確認してみていただきたい。

■コンフィグレーション項目の選び方

Apollo Pro133AとCoppermineの組み合わせの場合を例に、デフォルトの設定から変更する必要のある項目の中から、主要なものを**表3-2**に示した。使用していないハードウェアの設定をするとメモリの無駄遣いになるので、不要な機能ははずすようにしよう（もちろんわかる範囲内だけに止めておくべきだ）。

同じハードウェア構成で、プロセッサだけCeleronに変更した場合も、設定は変わらない。チップセットに440BX、i810Eあるいはi820を搭載したマザーボードも同様である。

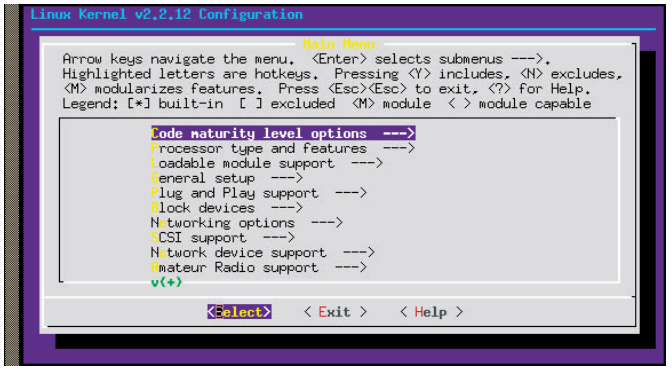
Athlonの場合、問題となるのはProcessor familyで何を選べばいいのか、ということだ。もっともプロセッサの検出はここの設定とは独立して行われるので、どのプロセッサを指定したとしても、Linuxが起動しなくなるということはない。カーネル2.2系列では「Pentium/K6/TSC」を選んでおけば間違いはないだろう。なお開発版カーネル2.3.xには独立した項目としてAthlonが設けられており、Athlon向けの最適化が加えられていることがわかる。

コンフィグレーションが完了したら、「Save and Exit」を選択して.configファイルをアップデートする。

■まずフロッピーにインストール

作成したカーネルに不具合が存在したため起動に失敗したり、Linuxのブートローダ LILOの更新に失敗したり、など起動しない場合もあるので、新しいカーネルはまずフロッピーにインストールして、動作を確認するとより安全だ。新しいカーネルを作成するには、フロッピーを挿入した状態で、次のコマンドを実行する。

```
# fdformat /dev/fd0H1440
# cd /usr/src/linux
# make dep
# make clean
# make bzdisk
# make modules
# make modules_install
```



画面3-1　make menuconfigの起動画面

画面3-2　make xconfigの起動画面

項目名	選択値
Processor type and features	
Processor family	PPro/6x86MX
Maximum Physical Memory	1GB
Math emulation	n
MTRR (Memory Type Range Register) support	y
Symmetric multi-processing support	n
Loadable module support	
Enable loadable module support	y
Set version information on all symbols for modules	y
Kernel module loader	y
General setup	
Parallel port support	y
PC-style hardware	y
Advanced Power Management BIOS support	y
Power off on shutdown	y
SCSI support	
SCSI support	n
Network device support	
EtherExpressPro/100 support	n
VIA Rhine support	y

表3-2　カーネルコンフィグレーションの例
EPoX製EP-6VBA2（Apollo Pro133A搭載）マザーボードとPentium III（Coppermine）、そしてVIA Rhineチップを搭載したEthernetカードの組み合わせで作成したPCの場合の、主なコンフィグレーション項目を示している。

このコマンド列を実行すると、新しいカーネルが作成され、フロッピーに書き込まれる。また同時にカーネルモジュールが/lib/modules/2.2.14の下にインストールされる。フロッピーを挿入したままshutdownして、新しいカーネルが起動するかどうかまず確認しよう。再起動に成功したら、

```
# cat /proc/cpuinfo
```

と入力して、プロセッサの情報が正しく表示されているかどうか確認してみよう。カーネル2.2.14ならば、Copper mineのキャッシュサイズが256Kバイトと正確に検出されているはずだ。

■LILOに新しいカーネルを登録する

毎回フロッピーで起動するのは面倒なので、ブートローダLILOの設定を変更して、新しいカーネルがハードディスクから起動するよう設定しよう。カーネルコンパイル時に「make bzdisk」の代わりに「make bzlilo」を指定すると、LILOの設定は自動で行われる。しかし今回はフロッピーを作成してテストをした、という前提で話を進めるので、直接LILOの設定ファイル/etc/lilo.confを編集する方法を用いる。

フロッピーに書き込んだカーネルと同じものが、/usr/src/linux/arch/i386/boot/bzImageに保存されているので、lilo.confを**リスト3-1**のように書き換え、次のコマンドを入力する。

```
# cp /usr/src/linux/arch/i386/boot
/bzImage /boot/bzImage-2.2.14
# /sbin/lilo
```

その後再起動して、ハードディスクから正しく起動するかどうか試してみよう。失敗したとしても、先ほどのフロッピーを用いて復旧すればよい。

なお、ディストリビューションによっては/etc/lilo.confに「initrd=」と指定されている場合があるが、これは起動時に必要なドライバモジュールを組み込んだRAMDISKイメージを指している。ディストリビューションは複数のハードウェアに対応する必要があるのでこの方法を採用しているのだが、今回は起動時に必要なドライバをカーネル再構築時に組み込んでしまっているので、initrdを用いる必要はない。initrdについてはmkinitrdのマニュアルを参照していただきたい。

▼リスト3-1　/etc/lilo.confの例

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
vga=normal
default=linux
image=/vmlinuz
  label=linux.old
  root=/dev/hda1
  read-only
image=/boot/bzImage-2.2.14
  label=linux
  root=/dev/hda1
  read-only
```

UltraDMA対応を有効にするには

Celeron／Pentium II／IIIシステムでもっとも多く使われているのは、440BXチップセットであり、そのIDEコントローラ82371（**写真3-2**）ならば、Linuxカーネル2.2標準のドライバにおいてすでにUltraDMA/33に対応している。しかし、最新のマザーボードの採用するIDEコントローラ（**写真3-1**、**3**、**4**）のドライバは、カーネル2.2にはほとんど含まれておらず、開発版カーネル2.3.xにおいてテストされている。開発版カーネルをインストールするには、カーネル以外のソフトウェアの入れ替えも必要となり、敷居が高い。そこで、カーネル2.3からIDEドライバを含むBlock Devicesセクションのみ抜き出して作成された、カーネル2.2用のUnified IDEパッチを適用して、安定版カーネル2.2.14に最新のIDEコントローラサポートを追加する方法を紹介する。

なお、このパッチで提供されるのは、いずれは安定版あるいは開発版のカーネルに取り込まれる拡張ではある。しかし、現時点でカーネルにマージされていないということは、テストが十分に行われていないということを意味している。導入に伴うリスクについてよく考えてから、パッチを当てるか否か判断していただきたい。

写真3-1　AMD-756

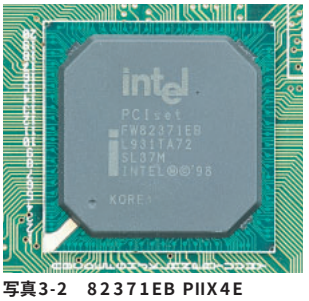


写真3-2　82371EB PIIX4E

写真3-3　82801AA ICH

写真3-4　VT82C686A

■パッチの入手

Unified IDEパッチはカーネルと同様、kernel.orgから入手可能だ（**表3-1**）。今回はカーネル2.2.14に対応した2000年1月15日版を用いた。パッチは日を置かずに更新されているので、できるだけ最新版を利用するようにしたい。

Unified IDEパッチの適用は次の手順で行う。ここでは/tmpにUnified IDEパッチをダウンロードした場合の例だ。

```
# cd /usr/src/linux
# make distclean
# cd ..
# bzip2-cd /tmp/ide.2.2.14.20000115
.tar.bz2 | patch -p0
```

■カーネルのコンフィグレーション

パッチをあてた後、make xconfigを実行すると、カーネルのコンフィグレーション画面が表示される。**画面3-3**は、Unified IDEパッチをあてた後の「Block Device」セクションを表示したところだ。Unified IDEパッチを適用する前と比べると、Promise製Ultra 66 IDEカードの採用するPDC20262チップなど、UltraDMA/66に対応したドライバが追加されていることがわかる。

今回テストに用いたマザーボードが採用するデバイスでは、Apollo Pro133AチップセットのIDEコントローラVT82CXXX（**写真3-4**のVT82C686Aを含む）やAMD-751チップセットのAMD-756（**写真3-1**）のUltra DMAサポートが追加されている。残念ながらi810（E）／i820のIDEコントローラ82801（**写真3-3**）については、まだUltraDMA機能はサポートされていない（検出できるようにはなった）。

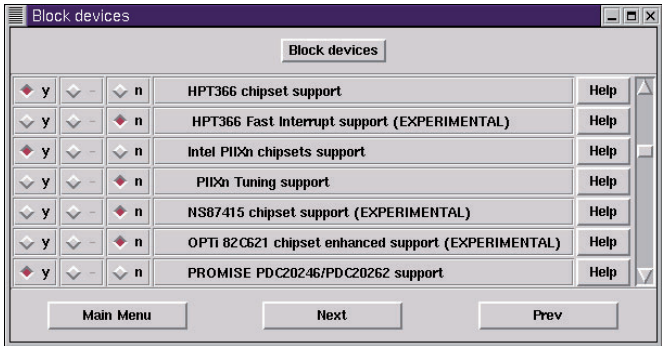
画面3-3からわかるように、各項目ごとに「y」「m」「n」の3つの選択肢（「y」と「n」のみの場合もある）が用意されている。「y」はドライバをカーネルイメージ自体に組み込む場合、「m」はドライバをモジュールとしてインストールする場合に選択する。しかし、起動デバイスであるIDEのドライバは、モジュールにするべきではないので「y」を選ぶ。

Apollo Pro133Aチップセットを採用したマザーボードの場合、「VIA82CXXX chipset support (EXPERIMENTAL)」を「y」に変更する。Intel製チップセットを採用したマザーボードの場合は、IDEチップセットのPIIX4Eに対応した「Intel PIIXn chipsets support」を「y」に、AMD-751を採用したAthlon対応マザーボードの場合は、「AMD7409 chipset support (EXPERIMENTAL)」を「y」に変更する。

Athlon対応マザーボードの中には、AMD-756の代わりにVT82C686Aを South Bridgeに採用したもの（たとえばASUSTeK製K7M）が存在することに注意したい。この場合は「VIA82CXXX chipset support (EXPERIMENTAL)」を「y」にすることでUltraDMAサポートを有効にできる。

「Use DMA by default when available」は、Linuxの起動時にIDEのDMA転送（UltraDMAを含む）を自動的に有効にするという設定だ。これも「y」に設定する。ほかはデフォルトの設定のままでよい。

あとは前述のカーネルリコンパイルと同じ手順で、新しいカーネルを作成しインストールする。



画面3-3
「Block Devices」セクションの設定画面

```
# mv /usr/include/linux /usr/include/linux.old
# cp -a /usr/src/linux/include/linux /usr/include
# cd /tmp
# mkdir tmp_i810
# mkdir tmp_XFCom
# cd tmp_i810
# tar zxvf I810Gtt-0.1-5.src.tar.gz
# make ; make install
# cd ../tmp_XFCom
# tar zxvf XFCom-i810-glibc2.1-1.00.tar.gz
# ./INSTALL
# cd /usr/X11R6/bin
# mv XF86_SVGA XF86_SVGA.old
# ln -s XFCom_i810 XF86_SVGA
```

画面3-4　i810（E）用Xサーバをインストールするためのコマンド実行手順

実際にドライバを組み込んで各マザーボードでテストしてみたが、VIAとAMDどちらのドライバも自動ではUltraDMA機能が有効にならず、次に示すhdparmコマンドを用いて、モードを変更している。

```
# hdparm -d1 -X66 /dev/hda
(UltraDMA/33モードへ変更)
# hdparm -d1 -X68 /dev/hda
(UltraDMA/66モードへ変更)
```

テストしたところAMD-756ドライバはUltraDMA/33までの対応だったが、これはドライバ付属のドキュメントにも記されているので、正常な動作といえるだろう。VIA VT82CXXXドライバは、hdparmコマンドを用いてUltra DMA/66モードに変更すると警告メッセージを表示したが、これもUltra DMA/66のテストが十分ではないとする、ドライバのドキュメントの記述通りだ。どちらもExperimental（実験的）サポートのドライバであり、なるべくなら組み込みたくないところだが、最近のIDEハードディスクはUltra DMAモード以外では性能がひどく低下してしまう（転送速度20Mバイト/sec強から数Mバイト/secまで低下する場合もある）。安定性と性能のどちらを取るか難しいところだ。

i810Eの内蔵グラフィックスをXで使うには？

i810EやSiS630のように、グラフィックチップ機能を統合したチップセットを採用する低価格帯PCが人気を集めている。というよりも、人気の低価格PCがこれらの統合型チップセットを採用する傾向にあるといえるだろう。XFree86の統合型チップセットのサポートは最新のXFree86 3.3.6でも1世代前のSiS620止まりで、まだこれからといったところだ。

実はXFree86 3.3.6からi810Eサポートは追加されているはずなのだが、バイナリでのリリースは2000年1月下旬の時点でまだ行なわれていない。そこで今回はi810（E）専用のXサーバを別途入手してインストールする方法を紹介する。このドライバはPrecision Insightによって開発されたもので、Intelのホームページから入手可能だ。なお、一部のディストリビューションはこのドライバを標準で搭載している。

■i810（E）対応Xサーバを導入する

ほかのXサーバと比較してi810（E）対応Xサーバが特殊なのは、AGPのサポートを必要とする点だ。i810はメインメモリの一部をグラフィックス用メモリとして使う設計になっている。そのため、起動時に最低限の画面表示に必要なグラフィックス用メモリを自動的に1Mバイトだけ確保する。ただ1Mバイトではサーバなどで多色高解像度環境を実現できないので、GART（Graphics Address Reaping Table）と呼ばれる手法に基づいて、メインメモリの一部をダイナミックにAGPメモリに割り当て、足りない分のグラフィックスメモリを確保するようになっている。カーネル2.2は標準ではGARTに対応していないので、i810（E）対応Xサーバをインストールするには、GARTをサポートするモジュールも同時に用意する必要があるのだ。

i810（E）対応XサーバとGARTモジュールは、**表3-1**に示したIntelのWebサイトからダウンロードできる。詳細なドキュメントも用意されているので、一度目を通しておくとよいだろう。ダウンロードする必要のあるのは、

```
XFCom-i810-glibc2.1-1.0.0
I810Gtt-0.1-5.src
```

という名前で始まる2つのファイルで、Red Hat系のディストリビューション（Red Hat Linux 6.0、LASER5 Linuxなど）の場合は拡張子がrpmのファイルを、それ以外のディストリビューションではtar.gz形式のファイルをダウンロードする。なお、i810（E）用のXサーバは、XFree86 3.3.5に対応したXサーバなので、事前に今使用しているXFree86のバージョンは調べておく必要がある（「X -version」コマンドで3.3.5と表示されるか確認しておく）。

画面3-5 「Card」セクションの設定画面

■インストールの手順

ここではCorel LINUX OSを例に、TAR形式のパッケージをインストールする場合のコマンドラインを**画面3-4**に示した。ここでは必要なファイルは全て/tmpディレクトリにダウンロードしたものとしている。Corel LINUX OS（あるいはDebian系ディストリビューション）で問題となるのは、起動時にモジュールをロードする設定方法がほかのディストリビューションとは異なる点だ。Corel LINUX OSの場合、/etc/modulesファイルに、

```
agpgart
```

の1行を追加する。Red Hat系やSlackware系のディストリビューションでは、インストーラが/etc/conf.modulesあるいは/etc/modules.confにagpgart.oを自動的に登録するので、この作業は必要ない。Linuxを再起動したのち、lsmodコマンドを実行して、agpgartモジュールがロードされていることを確認する。

i810（E）対応Xサーバの設定方法は複雑だが、ここではXF86_SVGAファイルと置き換えて、XFree86の設定ツールXF86Setupで設定できるようにした。XF86Setupは、コマンドラインで、

```
# XF86Setup
```

と入力して起動する。

画面3-5はXF86Setupコマンドを実行して、「Card」-「Detaild Setup」を選択した場合の表示だ。ここで「Server」を「SVGA」に、「Video RAM」は「Probed」を選択しておく。

「Card」以外の項目だが、「Mouse」「Keyboard」はそれぞれ自分の使用しているタイプを指定する。「Mode selection」では、i810（E）対応Xサーバは32bppカラーに対応していないので、それ以外の色深度と、利用したい解像度を選択する。「Monitor」では使用しているモニタのマニュアルを確認して、適切な値を入力しなければならない。しかしよくわからなければ、17インチ以上ある最近のディスプレイの場合、「Multi-frequency that can do 1280x1024@74Hz」ならば表示可能なので、これを指定しておくとよいだろう。全て設定したならば「Done」をクリックするとXが立ち上がるはずだ。起動したならば、「Save the configuration and exit」を選択し、設定を保存する。これでi810（E）対応Xサーバが立ち上がるようになり、高解像度で表示されるようになる。

最新プロセッサ＆マザーボードの性能を探る

今回紹介した5種類のマザーボードや各種メモリ、プロセッサをLinux環境でベンチマークテストにかけてみた。

テストに使ったLinux用プログラム

ベンチマークテストとして使ったのは、次の3種類のプログラムである。Corel LINUX OSにてカーネルをアップデートしたり必要なパッチを当てたりした状態で、これらのプログラムを実行している。

■午後のこーだ for Linux

国産のMP3エンコーダで、Windows版がよく知られているが、Linux版も元のエンコーダに対するパッチの形で配布されている。Pentium IIIのSSEやAthlonのEnhanced 3D Now!、そしてSMPなどに対応しており、エンコード速度は数あるMP3エンコーダの中でも最速の部類に入る。また、-testオプションを付けることによってベンチマーク（午後べんち）を計測する機能がある。今回は-testと-nopsy（心理解析なし）というオプションを指定して測定を行った。一次配布元はhttp://homepage1.nifty.com/herumi/である。

■Quake III Arena Demo

Quake III Arenaはid softwareによる3Dアクションゲームである。Windows版のほかにLinux版が存在し、実行にはMesa（フリーのOpenGL互換ライブラリ）を必要とする。また実用的な実行速度を得るにはグラフィックスハードウェアによるアクセラレーションが欠かせない。ベンチマークを実行するには、Quake IIIのコンソールモードで「timedemo 1」「demo demo001」などと入力する。一般にQuake IIIのベンチマークはグラフィックスの性能差を比較するために用いられるが、ここではプロセッサとメモリサブシステ

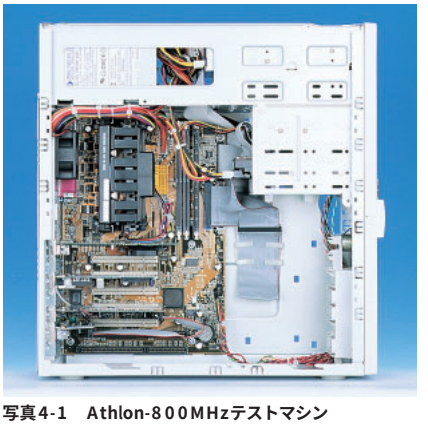


写真4-1　Athlon-800MHzテストマシン
Athlon-800MHzのテストについては、日本AMDより借用したこのリファレンスマシンを使用した。マザーボードはAMD-750採用のGigabyte Technology製GA7IXで、グラフィックスはRIVA TNT2 Ultra、サウンドはSound Blaster Live!である。

ムの性能を見るため、グラフィックスをnVIDIA製RIVA TNT2 Ultraに統一して測定した。Xサーバには、nVIDIAが2000年1月にリリースした最新のGLX/Mesa対応版を使用している。デモプログラムはftp://www.cdrom.com/pub/idgames/idstuff/quake3/linux/などから入手できる。

■LMBENCH

Bit Moverが公開しているベンチマークプログラムである。データ転送時の帯域幅（連続でデータを転送するときの速度）とレイテンシ（データが要求されてから実際に転送が始まるまでの遅延）を計測可能で、メモリ操作のほかにネットワークやファイルシステムへのアクセス、プロセス操作などの速度を計測できる。今回はメモリ帯域幅の測定に用いた。LMBENCHはhttp://www.bitmover.com/lmbench/から入手できる。

ベンチマーク結果について

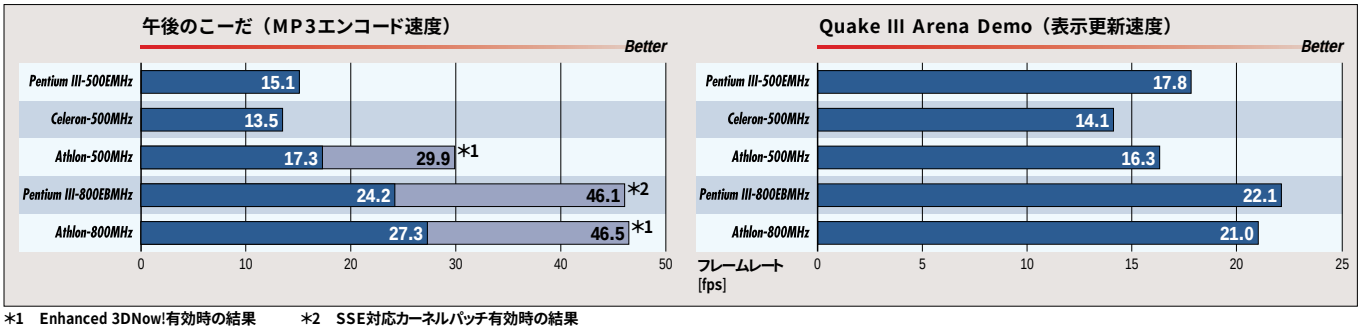
ベンチマークテストの結果は**グラフ4-1〜4-4**の通りである。いずれもマザーボードのプロセッサ／メモリ関連の設定は、BIOSセットアップのデフォルトのままテストしている。

■マルチメディア命令の効果あり

まずプロセッサごとの性能を比較してみよう。**グラフ4-1**のように、テストには午後のこーだとQuake III Arena Demoを使用した。前者は、午後べんちのインデックス値を掲載している。またIntel製プロセッサはEPoX製EP6-VBA2にPC100／133 SDRAM DIMMを、Athlon-800MHzにはAMD-750チップセット搭載マザーボードにPC100 SDRAM DIMMを組み合わせてテストしている。500MHzのプロセッサは普及価格帯クラスでの性能を、また800MHzは最速クラスでの性能をチェックするのが目的だ。

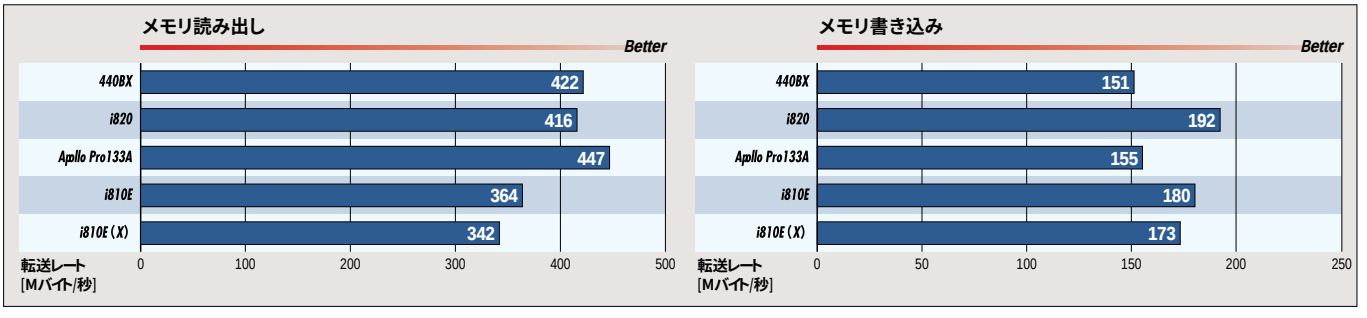
グラフ4-1を見ると、MP3エンコード速度ではAthlonが、一方Quake IIIのフレームレートではPentium IIIが優勢であることがわかる。どちらでもCeleronは最下位だが、性能差はそれほど大きくない。ところで午後のこーだのAthlonやPentium III-800EBMHzの結果が2つあるのは、それぞれのマルチメディア処理命令を有効／無効にしてテストしたからだ。どちらのプロセッサでもその効果は絶大だが、Athlonが安定版カーネル2.2ですでにEnhanced 3DNow!を利用できる環境にあるのに対し、Pentium IIIではカーネル2.2／2.3ともにまだSSEを正式には利用できない状態だ（テストではカーネル2.2にパッチを当てて無理やりSSEを使えるようにした）。つまり安全に午後のこーだで高速化できるのは、現状ではAthlonだけといえる。

グラフ4-1　各プロセッサの性能比較



*1　Enhanced 3DNow!有効時の結果　　*2　SSE対応カーネルパッチ有効時の結果

グラフ4-2　各チップセットの性能比較



■Apollo Pro133Aが優勢

グラフ4-2は、Intel製プロセッサと組み合わせるチップセットのメモリサブシステムについて、性能を比較した結果である。テストにはLMBENCHを用いた。マザーボードは97～100ページで紹介した製品である。Athlonを含めていないのは、まだチップセットがAMD-750だけで比較対象がなかったからだ。プロセッサはPentium-500EMHzでメモリはPC100 SDRAMかPC800 Direct RDRAMを用いている。

一般的なアプリケーションへの影響が大きいメモリ読み出しについては、Apollo Pro133Aが最も高速だ。このチップセットの場合、FSBが133MHzならPC133 SDRAMを使うと、さらに差が広がるはずだ。メモリ書き込みでi810E／i820が速いのは、メモリの種類に関係なく、チップセットのアーキテクチャによるものだろう。

グラフ4-2で「i810E（X）」とあるのは、X Window System上でLMBENCHを実行した結果だ。X上では解像度や色数が向上する分、i810E内蔵のグラフィックス回路がメインメモリをアクセスする頻度が高まり、プロセッサが使える帯域幅が狭まるはずだ。実際、コンソールで実行した「i810E」の結果と比べると、あきらかにX上では性能は落ちていることがわかる。パフォーマンスを重視する場合は、ほかのチップセットから選びたいところだ。

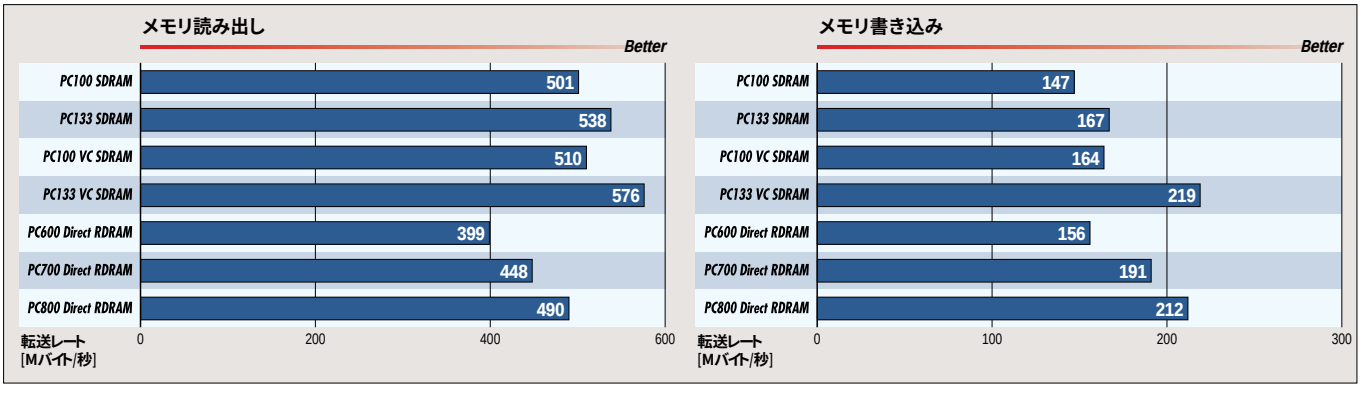
■VC SDRAMの効果は高い

グラフ4-3は各メモリの性能を、また**グラフ4-4**はFSBとメモリの速度による性能差をLMBENCHで測定した結果である。**グラフ4-4**と**グラフ4-3**のSDRAM系の測定にはマザーボードにEP-6VBA2（Apollo Pro133A）を、またDirect RDRAMの測定にはASUSTeK製P3C-E（i820）を使用した。プロセッサは**グラフ4-3**でPentium III-733MHz（FSBは133MHz）を、**グラフ4-4**では600EMHzと600EBMHzを用いている。

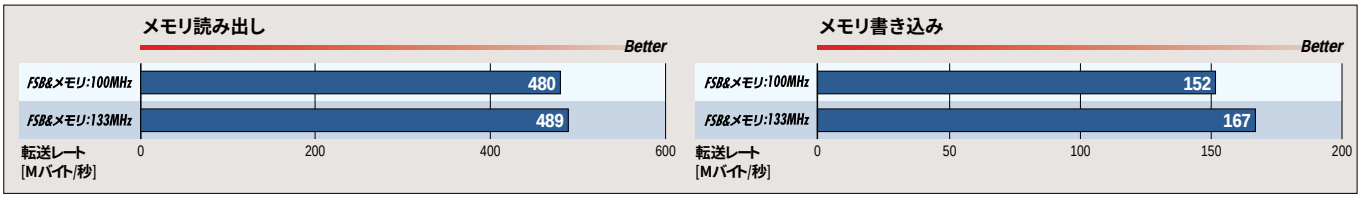
さて**グラフ4-3**を見ると、PC133 VC SDRAMの結果が読み書きともに頭ひとつ飛び抜けていることがわかる。価格が下がり入手しやすくなれば有望なメモリなのだが。一方、Direct RDRAMは最速のPC800でも読み出しでPC100 SDRAMに劣ってしまっている。理論上の帯域はDirect RDRAMのほうが圧倒的に高いので、プロセッサとI/Oの両方でメモリの帯域幅を消費するようなアプリケーションなら、Direct RDRAMの有効性が表れる可能性はある。また、書き込みの性能が高いのは、前述のようにチップセットによるものかもしれない。

グラフ4-4では、133MHzの効果があまり表れていないように見える。しかしQuake IIIのテストでも、その効果は表れていた。またPentium III-733MHzで試した**グラフ4-3**のほうが読み出しで差が開いていることから、より高速なプロセッサが登場する今後、133MHzの真価が表れるのだろう。

グラフ4-3　メモリの種類による性能差



グラフ4-4　FSBとメモリのクロック周波数による性能差



日本語環境

漢字

わかって、使える

日本語環境構築ガイド

とかくLinuxの日本語環境はわかりづらい。
読む、書くといった日本語環境の基本をすべて解説！
あなたのLinuxが「日本人」になります。

Japanese Environment

構築ガイド

Section1

日本語対応とは？

この特集の副題は「あなたのLinuxが日本人に」となっている。しかし、読者が使っているディストリビューションの多くは、すでに「日本人」だ。だから、インストール後に特別な設定をしなくても、いきなり日本語の表示や入力ができるはずだ。それは、ディストリビューターが事前に設定しているからであり、それが「日本語ディストリビューション」が「日本語」を名乗る所以なのだ。

では、読者は日本語環境の構築方法について何も知らなくてもいいか、というとそんなことはない（と思う）。

Debian GNU/Linuxのように日本語環境が設定されていないディストリビューションもあるし、SuSE LinuxやCorel LINUXのように洋物ディストリビューションを使いたくなることもあるかもしれない。さらに、日本語関連の知識を得ることで、標準設定だけでは気づかなかった、より便利な使い方も発見できるかもしれない。あなたが日本人である以上、より便利な日本語環境を構築するには、いろいろな知識が必要になるのは間違いないのだ。

というわけで、ここではLinuxの日本語環境はどのような仕組みになっていて、環境を構築するにはどのようなことをすればいいか、といったことを解説していく。そういう意味では、ディストリビューションを「日本人にする」のではなく、なぜ「日本人なのか」を探る道といえるだろう。ブラックボックス化したOSと違い、仕組みがわかって、環境をカスタマイズできる楽しみを知ってもらいたい。

なお今回は、解説するプログラムがすべてインストールされている「TurboLinux Pro 日本語版 4.2」を対象に解説していく。ディレクトリ構成などいくつか違う点もあるだろうが、設定法そのものは基本的に同じなので、適宜自分の環境に読み替えながら、読み進んでほしい。

日本語対応ということ

ひと口に「Linuxで日本語を使う」といっても、その対応のレベルはさまざまある。「日本語対応」の要件として、たとえば、

- **日本語の表示ができる**
- **日本語の入力ができる**
- **日本語の文書が印刷できる**
- **日本語の文書を送受信できる**
- **プログラム内で日本語が使える**

といったことが挙げられる。最後のプログラム内での日本語対応についても、日本語のコメントが使えるといったものから、日本語で関数が定義できるというものまでさまざまである。

限られた紙幅でこれらをすべて網羅して解説するのは難しいし、ごく一般的なLinuxユーザーには、とりあえず必要になるのは日本語の表示／入力だろう。

というわけで、ここでは主に日本語の表示／入力を中心に解説していく。「何だ、その程度か」と思う読者もいるかもしれないが、表示についてだけでも

- **文字コード**
- **ロケール**

などについて知識（かなり簡単なものでいいのだが）が必要となる。さらに入力についても、前述の2つに加えて、数多くある、

- **日本語変換サーバ**
- **日本語入力クライアント**

などについての知識が必要となる。しかも、コンソールとX Window Systemではこれらの実現方法が異なる。

「外人」のLinuxには「その程度か」も、たやすいことではないのだ。

では、まず日本語環境を構築するのに必要な基礎知識から紹介していこう。ここは、子供が日本語を使えるようになるまでの段階にたとえるなら、まず「ひらがな」を覚えましょう、といったところだ。

文字集合

コンピュータで文字を表現するときは、「符号化」と呼ばれる、文字とコードの対応づけが必要となる。これらをある基準に従って集めた表を「文字集合」と呼び、ISOによって定義されている。日本語の文字集合はJIS規格によって定められており、

- **「JIS X 0201-1997」ローマ字など**
- **「JIS X 0208-1997」漢字など**
- **「JIS X 0212-1990」非漢字、補助漢字など**

という3つの文字集合がある。ただし、開発者ならともかく、一般ユーザーが文字集合の詳細を意識しなければならないときはあまりない。そのため、ここでは詳しくは触れない（文字集合についてより詳しく知りたい方は、本誌201ページからの「日本語環境」を参照してほしい）。ただ、1バイト（正確には7ビット）で表現されるローマ字と、2バイトで表現される漢字といった複数の文字集合を切り替えながら、利用しないといけないということだけは覚えておいてほしい。

文字コード

日本語の文字集合が複数あることはすでに説明したとおりだ。ここに、

私はLinuxユーザーです。

という文があったとしよう。これを表示するには、「Linux」という1バイトの文字集合とそれ以外の2バイトの文字集合を切り替える必要がある。しかし、ただ文字コードを順番に並べただけでは、どこまでが1バイト文字で、どこからが2バイト文字なのかがコンピュータには判別できなくなってしまう。

そのため、文字集合の切り替えを何らかの方法でシステムに教えてあげなくてはならない。この文字集合の切り替え方法によって、次のような3つの文字コードが混在している。

■JISコード

文字集合を「ESC (B」や「ESC $B」といったコード列（「エスケープシーケンス」という）によって切り替える文字コードである。文字すべてが7ビット以内に収まっており、電子メールなどのように通信を通してデータ転送を行う際の文字コードとして利用されている。

■シフトJISコード

JISコードの切り替え方法では、文字列のある部分だけを取り出した際、その部分の文字集合が何であるかがわからない。また、エスケープシーケンスが挿入されるため、データが大きくなってしまう。シフトJISコードは、こういったJISコードが持つ欠点を補うべく、Microsoftとアスキーによって考案された文字コードで、2バイト文字の1文字目（8ビット）が1バイトカナ文字のエリアと重ならないように、JISコードを移動（シフト）させたものである。WindowsやMacintoshなどで利用されている文字コードである。

■EUCコード

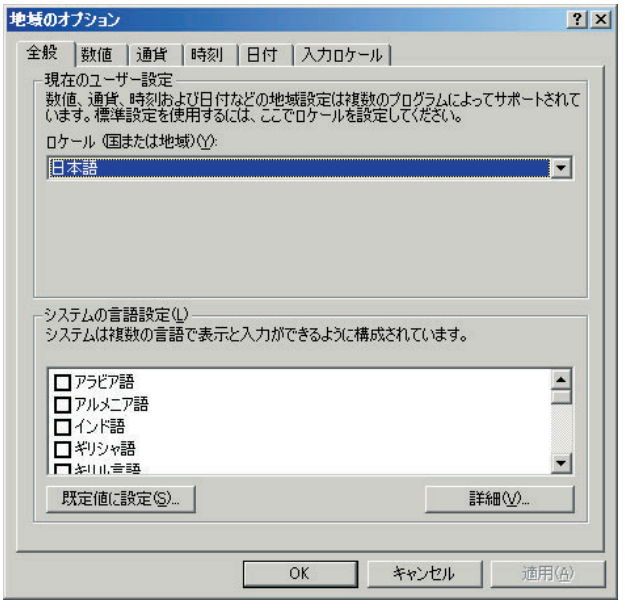
シフトJISコードは、ISO 2022の標準規格から外れていることや2バイト目にC言語のエスケープ文字がくることなどの理由から、UNIXの日本語化において新しいコード体系が考案された。これがEUC（Extended Unix Code）で、その名のとおり、主にUNIXで利用されている文字コードである。文字集合の切り替えに8ビット目を利用する。

これらの各コード系の切り替え方式など、あまり詳しい仕組みを覚えている必要はないが、表示／入力しようとしているファイルの文字コードが何なのかは把握しておく必要はある。

ロケール

Linuxのように世界中で利用されているOSでは、国や地域ごとの言語や習慣などの部分をプログラムから切り離してデータベース化しておき、ロケール（locale）と呼ばれる設定によって、それぞれの国や地域に合った言語や習慣を表示するようになっている。

具体的には、環境変数LANGやLC_ALLに“ja_JP.ujis”（日本語EUC）などを指定しておく。このロケールが設定されていないと、ページャであるlessやマニュアル表示コマンドのmanを利用した際、ちゃんと日本語表示されない場合がある。



画面1　Windows 2000のロケール設定
Windows 2000は、バイナリが1つに統合されており、ロケールの切り替えによって世界中の各国に対応するなど、かなり進んだ国際化がなされている。Linuxも、開発中の次バージョンglibc 2.2で完全な国際化対応が期待されているが、このレベルの達するのはいつになるか…。

Section2

表示

さて、つたないながらも「日本人」に必要な「ことば」がわかったはずだ。では、次の段階として「日本語を読んで」みよう。

前準備

Linux上で日本語を表示させるのは、日本語入力ほど複雑ではない。ただし、正しく表示させるには、コンソール、X Window System、いずれの環境でもロケールの設定が必要になる。次のように、

```
$ echo $LANG
```

としてみて、LANG=C（英語環境）となっていたら、

```
$ export LANG=ja_JP.ujis
```

として、日本語環境に合わせた設定をしておく。あとは対象とするファイルの文字コードに気をつけておけばよい。

日本語で書かれた文書の参照

ファイル内容を参照するには、cat、head、tailといったコマンドやmore、lessといったページャコマンドなどがある。TurboLinuxでは、いずれのコマンドでも日本語で書かれた文書を参照することが可能だ。

■コンソールの場合

コンソール上で日本語を表示させるのは簡単だ。ログインしてから、

```
$ kon
```

と入力して、漢字コンソールエミュレータ「kon」を起動してから、

```
$ less xxx.txt
```

としてファイルを参照すればよい（**画面2**）。

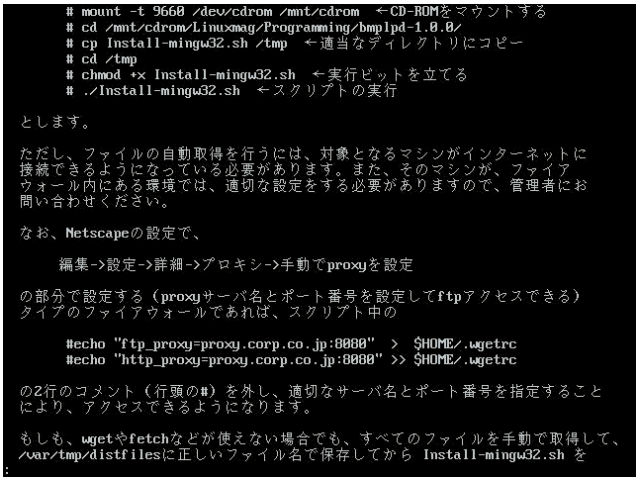
ここで注意しないといけないのは、konは/etc/kon.cfgファイルによって利用できる文字コードが設定されていることだ（多くの場合は、日本語EUCとなっている）。そのため、シフトJISコードで書かれた文書ファイルをcatコマンドやmoreコマンドで参照しようとすると、文字化けを起こしてしまう。

■X Window Systemの場合

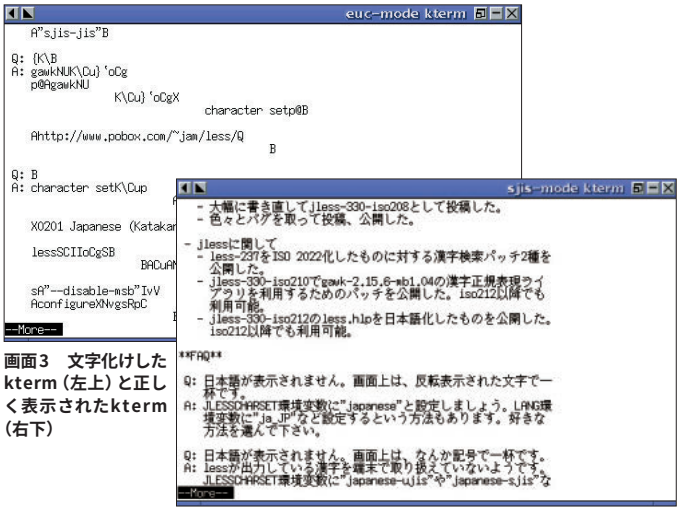
コンソールでの表示は非常に簡単だったが、X Window System上で日本語の文書ファイルを表示するには、いくつか気をつけないといけないことある。

まず、ターミナルエミュレータはktermを利用しなければならない。さらに、表示したいファイルが記述されている文字コードに合わせて、ktermの-kmオプションで文字コードの指定をしなければならない。たとえば、表示したいファイルがシフトJISで記述されているなら、

```
$ kterm -km sjis &
```



画面2　kon上のlessによる日本語文書の表示



画面3　文字化けしたktermと正しく表示されたktermの比較（左上）と正しく表示されたktermの（右下）

として、ktermが表示する文字コードを明示的に指定しなければならない(**画面3**)。

なお、日本語EUCコードの場合は、-kmの引数として「euc」を指定してもかまわないが、ktermの標準の文字コードは日本語EUCとなっているディストリビューションがほとんどなので、

```
$ kterm
```

として、-kmオプションを指定しないで起動しても問題はない。

また、JISコードのファイルはシフトJISモードのkterm、EUCモードのktermのいずれでも参照することができるので、あえて“-km jis”を指定する必要はないだろう。

これで初めて正しく日本語表示ができるようになる。といっても、最近のコマンドはよくできたもので、lessコマンドなら文字コードを自動判別して表示してくれる。そのため、ロケールさえちゃんと設定してあれば、ktermの設定に関わらず、ちゃんと表示させることができる。

しかし、このような機能に頼っていては、いつまでたっても正しい知識や設定方法を覚えることができない。正しい方法が身につくまでは、面倒でもこのような方法で表示させるよう心がけることをお勧めしたい。

■コード変換

いちいち文字コードを意識しながらファイルを参照するのが面倒だということなら、コード変換コマンド「nkf」によって、ファイルそのもののコード変換をしてしまえばよい。たとえば、シフトJISで記述されたファイル(japanese.sjs)をEUCコードのファイル(japanese.euc)に変換するなら、

```
$ nkf -e japanese.sjs > japanese.euc
```

とする。

ここで紹介した以外にも、nkfコマンドには便利なオプションがたくさんあるので、一度nkfコマンドのマニュアルを参照するといいだろう。

日本語マニュアルの表示

最近の日本語ディストリビューションには、JM Project(http://www.linux.or.jp/JM/)の成果物である、日本語オンラインマニュアルがバンドルされていることが多い。この日本語オンラインマニュアルを参照するには、

```
$ man コマンド名
```

とすればよい(**画面4**)。manコマンドが環境変数LANGを参照し、日本語の設定(ja_JP.ujisなど)となっていれば(かつ日本語manファイルが用意されているもの)、日本語表示を行うようになっている。うまく表示されないようなら、環境変数LANGの値が“ja_JP.ujis”や“Japanese”など、日本語ロケールに設定されているか確認してほしい。

また、ディストリビューション(たとえば、Vine Linuxなど)によっては、日本語表示を行うためのmanコマンド、jmanコマンドが用意されている場合もある。

TurboLinuxでは…

さて、表示に際してロケール設定がいかに重要かはわかってもらえたと思う。しかし、今回サンプルとして利用しているTurboLinuxでは、環境変数LANGの設定をしなくても、ちゃんと日本語表示がされている。これはどうしてだろうか?

答えは簡単。起動時に自動的に設定されているから、である。

bashは起動時に、真っ先に/etc/profileを読み込む。この/etc/profileを見てみると、

```
# Load system default language.
if [ -f "/etc/sysconfig/lang" ]; then
        . /etc/sysconfig/lang
fi
```

という部分がある。これは、「/etc/sysconfig/langがあれば、そのファイルを読み込む」という働きをするものだ。次に、その/etc/sysconfig/langを見てみると、

```
export LANG=ja_JP.ujis
export LANGUAGE=ja
```

となっているので、このファイルが/etc/profileに読み込まれることによって、LANG設定が起動時に完了しているのだ。このあたりの仕組みは、各ディストリビューションによってさまざまだが、何らかの方法で起動時に環境変数LANGを設定している場合が多い。

画面4　日本語manの表示

日本語のファイル名

TurboLinuxにバンドルされているcp、mvといったファイル操作系のコマンドは、日本語ファイル名を扱うことができる。touchコマンドなどで日本語のファイル名を作成することも可能であり、bashならTABキーによるファイル名補完も行うことができる。

以前は、ファイルを表示するlsコマンドが日本語ファイルをうまく表示することができなかったのだが、TurboLinuxにバンドルされているlsコマンドでは、日本語ファイルを問題なく表示することができた（**画面5**）。最近のディストリビューションには、GNU fileutilsのコマンドをバンドルしているものが多いので、TurboLinux以外のディストリビューションでも大丈夫だろう。

もしも、使用しているディストリビューションのlsコマンドなどが日本語のファイル名に対応していないものであっても、Emacs系エディタ（Mule、XEmacs、Emacs20など）のファイラモードであるdiredを使えば、表示からコピー、削除といったファイル操作、さらにファイル名補完まで行うことができる。

しかし、GUIでファイル操作を行うことが多いWindowsやMacintoshと異なり、LinuxではCUIでのファイル操作が多い。

そのため、ファイル名を入力するたびごとに日本語入力システムをオン／オフするのはかなり煩雑な作業となるため、日本語のファイル名をつけるのは、あまりお勧めはできない。

画面5　日本語のファイル名の操作

Column

日本語のファイル名とSamba

Linuxでも日本語ファイルが使えることは、本文でも紹介したとおりだ。しかし、Sambaを利用して、Windowsマシンにリソースを公開しているLinuxマシン上で日本語ファイル名を使う際は、注意しておかないといけないことがいくつかある。

まず、文字コードの問題である。Linuxは文字コードとして日本語EUCを利用しているが、Windowsは文字コードにシフトJISを使っているため、ディレクトリ名やファイル名の表示が文字化けしてしまう（**画面c-1**）。

これを防ぐには、Sambaの設定ファイルである/etc/smb.confに次のようなエントリを追加すればよい。

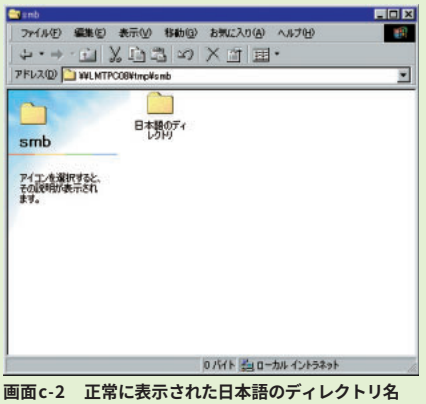
```
client code page = 932
coding system = euc
```

これは、クライアントに日本語版Windows（932）、Sambaサーバを稼動させているLinuxのファイルシステムで使う日本語コードとしてEUCを設定することを意味している。

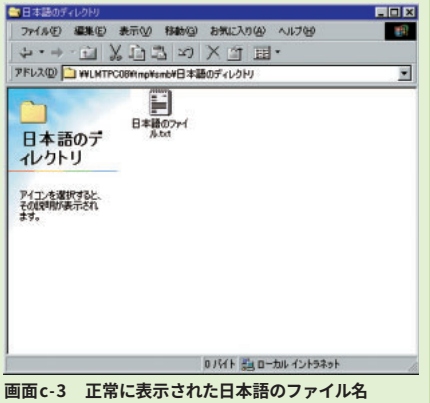
これで、標準設定では文字化けで参照できなかったディレクトリやファイルも参照できるようになる（**画面c-2、c-3**）。

また、これらの日本語のファイル名は、共有名には利用できないので、注意が必要である（裏ワザはあるが、お勧めできない）。

画面c-1　文字化けした日本語のディレクリ名



画面c-2　正常に表示された日本語のディレクトリ名



画面c-3　正常に表示された日本語のファイル名

Section3

入力

さて、「ことば」がわかって、「読める」ようになったなら、次はやはり「日本語を書く」だろう。この「書く」というのが、自然言語と同様でなかなかたいへんな作業だが、ここを乗り切れば、一応は「日本語が扱える」というレベルということができるはずだ。

Linuxで「日本語を書く」ためのシステム、「かな漢字変換システム」には数多くの種類がある。ここでは、よく知られているものを中心に紹介していく。

クライアント／サーバ方式のかな漢字変換システム

多くのUNIXアプリケーションがそうであるように、UNIXの日本語入力システムも、クライアント／サーバ方式を採用している。

日本語入力のサーバは変換を担当しており、クライアントから送られてきた文字列をかな漢字混じりの文にして、クライアントに返す。クライアントは、ユーザーからの入力などユーザーインターフェイス部分を受け持つ。クライアントとサーバはプロセス間通信によってやり取りされており、同一マシン上で動作している必要はない。

このことによる利点は、

- **複数クライアントからのアクセスが可能**
- **柔軟なシステムを構築できる**
- **ネットワーク越しの利用が可能**

といったものが挙げられる。しかし、その柔軟さゆえに、

- **設定がわかりづらい**

という欠点があるのも事実だ。

かな漢字変換システムの各種方式

UNIXに日本語を入力するには、方法がいくつかある。

■アプリケーション組み込み型

まず、最初に挙げられるのが「アプリケーション組み込み型」と呼べるものである。これは、アプリケーション自身が日本語入力の仕組みを持つものである。

■FEP型

これは、仮想端末という仕組みを利用して、ターミナルとアプリケーションの間で日本語入力を制御するシステムである（**図1**）。このようにすることで、アプリケーション側には、あたかもユーザーが日本語を入力しているかのように見せかけることができる。FEP型にはWnn用のuumやCanna用のcanuumなどがある。

■XIM

X11R5以降、XではXIM（X Input Method）と呼ばれる、国際化実現のためのライブラリが提供されている。このXIMを利用することにより、日本語入力において共通化されたユーザーインターフェイスを得ることができる（**図2**）。このXIMを利用して変換サーバとのやり取りをするのが、xwnmoやkinput2といった「入力サーバ」と呼ばれるプログラムである。xwnmoは変換サーバとしてjserverを利用し、kinput2はjserver、cannaserverの両方を利用する。

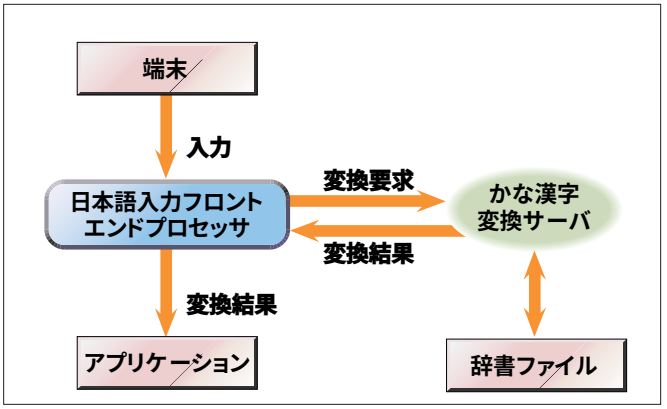


図1　FEP型の日本語入力

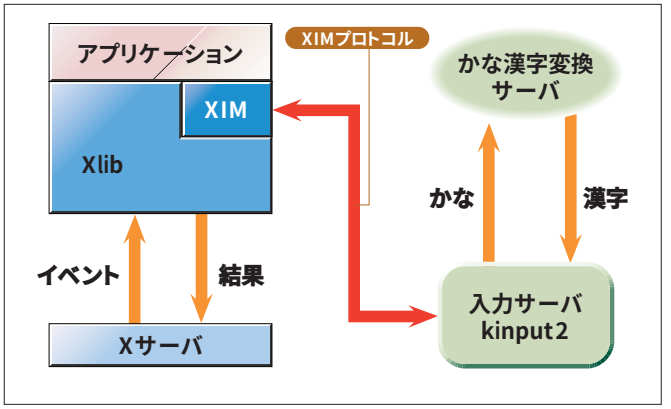


図2　XIM経由の日本語入力

Canna

Canna（かんな）は、NECによって開発されたクライアント／サーバ方式の日本語入力システムである（http://www.nec.co.jp/japanese/product/computer/soft/canna/）。

かな漢字の変換サーバには「cannaserver」が担当し、フロントエンドプロセッサ「canuum」、Xのインプットメソッド「kinput2」、マルチリンガルエディタ「Mule」などから利用できる。どのような環境からでも統一的なインターフェイスによって入力ができるという特徴を持っており、カスタマイズファイルである.cannaも一元化されている。

Cannaの設定方法

Cannaを利用するには、まず変換サーバであるcannaserverを起動し、次に入力クライアントを起動する。入力クライアントとして何を選ぶかは、何に日本語を入力するか、ということによって変わってくる。

ktermやviに入力するならcannumかkinput2になるし、Muleに入力するなら「かんな/emacs」を利用する。それぞれの方法を紹介しておくので、これらの中から、読者の目的に合ったものを選んでほしい。

■変換サーバの起動

まず、Cannaのかな漢字サーバ「cannaserver」を起動する。

```
$ cannaserver &
```

とする。

もしも、cannaserverコマンドが見つからないというエラーが表示された場合、Cannaがインストールされていないか、PATH設定がされていないということである。Cannaがインストールされている場所はディストリビューションごとに異なっているので、

```
$ find /usr -name canna*
```

などとして探してみてほしい。

また、**画面6**のようなエラーが表示された場合、すでにcannaserverが起動されていることを意味している。

cannaserverへの接続状況（cannaserverが起動しているか、だれが接続しているかなど）を知りたいなら、cannastatを実行してみればよい（**画面7**）。

また、あまり必要になることはないと思うが、意図的にcannaserverを停止したいなら、

```
# cannakill
```

とすればよい。

なお、TurboLinuxでは、/etc/rc.d/rc?.dディレクトリにS98cannaスクリプトが登録されているので、システムの起動と同時にcannaserverが起動するようになっている。

■FEPからの利用

canuumは、cannaserverのFEPとして機能する。ktermなどX Window System環境の入力としても利用できるが、XIMがサポートされた現在はkinput2などがあるため、使う必然性はあまりない。そのためか、TurboLinuxにはcanuumはバンドルされていない。

canuumの起動はいたって簡単で、

```
$ canuum &
```

と入力するだけだ。canuumが起動すると、最下行が変換行として使われる。また、次のように、

```
$ canuum -D remotehost
```

と、-Dオプションのあとにマシン名を指定するか、環境変数CANNAHOSTに指定しておけば、リモートマシン上で動作しているcannaserverを利用することもできる。

canuumを終了するには、

```
$ exit
```

```
$cannaserver
ERROR:
   Another 'cannaserver' is detected.
   If 'cannaserver' is not running,
   "/tmp/.iroha_unix/IROHA" may remain accidentally.
   So, after making sure that 'cannaserver' is not
running.
   Please execute following command.

              rm /tmp/.iroha_unix/IROHA
```

画面6　cannaserverを二重起動した際のエラーメッセージ

```
$ cannastat
Connected to unix
Canna Server (Ver. 3.5)
Total connecting clients 2
USER_NAME   ID  NO U_CX          C_TIME   U_TIME I_TIME HOST_NAME  CLIENT
mitsug-k     0   0    2  Mon 24  7:25pm        0      4 lmtpc71.pb kinput2
root         8   1    2  Mon 24  7:29pm        0        lmtpc08.pb kinput2
```

画面7　cannaserverへの接続状況を調査

と入力するか、CTRL＋Dを入力する。

■インプットメソッドの利用

Cannaのインプットメソッドにはkinput2がある。このkinput2を利用するには、次の手順を実行する。

1.入力サーバの指定

まず、入力サーバの指定が必要である。たとえば、kinput2を利用するなら、環境変数XMODIFIERSを設定しておく。

```
$ export XMODIFIERS="@im=kinput2"
```

2.リソースファイルの設定

ktermに限らず、X上で動作するアプリケーションはすべて「リソース」と呼ばれる設定ファイルによって見栄えや挙動が決められている。ktermに日本語入力を行うためには、ユーザーごとのリソースファイルである.Xdefaultsに、**リスト1**のように設定しておかなければならない。

▼リスト1　ktermのリソースファイルの設定

```
KTerm*VT100*translations:       #override \
 shift<Key>space:               begin-conversion(_JAPANESE_CONVERSION)
KTerm*allowSendEvents:          true
```

なお、TurboLinuxでは各ユーザーの.Xdefaultsではなく、システム全体のデフォルト設定が記述されている/usr/X11R6/lib/X11/app-defaultsディレクトリのKTermファイルの先頭に、**リスト1**と同様の設定がなされているので、ユーザーは設定をしなくても日本語入力が可能となっている。

3.入力サーバの起動

kinput2を起動するには、

```
$ kinput2 -canna &
```

とする。

4.Shift＋Spaceキーを押す

これで、kterm（**画面8**）やvi（**画面9**）などの日本語入力をしたいところでShiftキーとSpaceキーを同時に押すと、カーソル位置の上に**［あ］**というインジケータが表示され、日本語入力が可能になる。

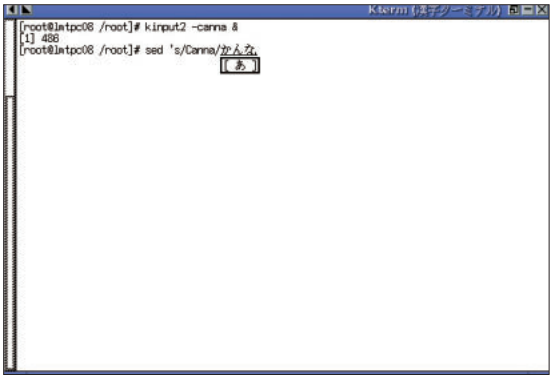
■Muleからの利用

Muleは、「かんな/emacs」を経由してcannaserverと直接やり取りする。MuleからCannaを利用するには、~/.emacsに次のような設定を書いておく。このリストの意味については、Emacsの書籍を参照してほしい。

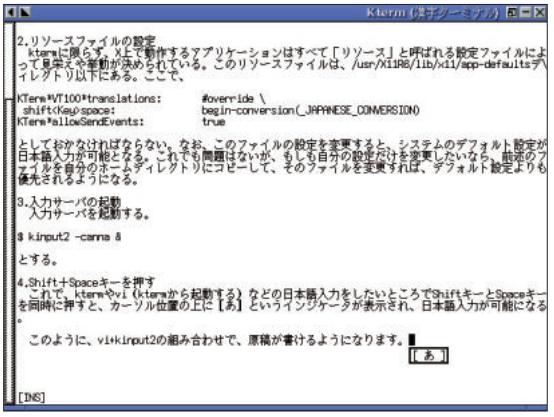
```
(if (and (boundp 'CANNA) CANNA)
    (progn
     (load-library "canna")
     (canna)
))
```

Muleが起動したら、Ctrl+oキーを押して、モード表示が**［あ］**に変われば、日本語入力が可能である（**画面10**）。

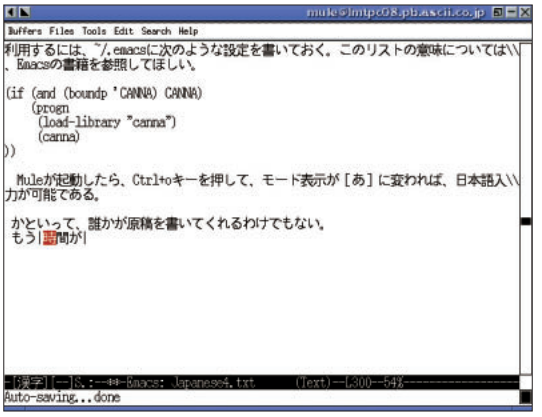
アルファベット入力モードに戻るときは、もう一度Ctrl＋oを押す。



画面8　kinput2＋Cannaによるktermへの日本語入力



画面9　kinput2＋CannaによるviへのJ本語入力



画面10　Mule＋かんな/emacsによる日本語入力

Cannaのカスタマイズ

Cannaは、初期化ファイル~/.cannaに各種設定を書いておくことでカスタマイズできる。

たとえば、アルファベット入力モードから日本語入力モードへの切り替えキーを、デフォルトのCtrl＋oからCtrl＋\に変更するなら、.cannaファイルに**リスト2**のように書いておく。

さらに、Cannaにはユーティリティ機能があり、記号入力、コード入力、部首入力、単語登録、環境設定など、各種機能を利用できる。このユーティリティ機能は、~/.cannaに次のように設定を書いておけば、kinput2などでcannaを利用しているときに、F1キーを押すことで呼び出すことできる（**画面11**）。

```
(global-set-key "\F1" 'extend-mode)
```

また、Muleからは、M-x canna-extend-modeで呼び出すことができる（**画面12**）。

Cannaの辞書管理

Cannaには、全ユーザーが利用する「システム辞書」、特定のグループに属するユーザーだけが使用できる「グループ辞書」、個人ごとに単語を登録しておける「ユーザー辞書」という3つがある。デフォルトでは、「システム辞書」はバイナリ形式（拡張子は.d）、「グループ辞書」「ユーザー辞書」はテキスト形式（拡張子は.t）となっている。なお、拡張子が.kpとなっているファイルは、ローマ字かな変換テーブルが格納されている。

▼リスト2　.cannaによるキーバインドのカスタマイズ

```
(set-key 'alpha-mode "\C-o" 'self-insert)
(set-key 'alpha-mode "\C-\" 'japanese-mode)
(set-key 'empty-mode "\C-o" 'undefined)
(set-key 'empty-mode "\C-\" 'alpha-mode)
```

画面11　F1キーで呼び出されたCannaの拡張モード

画面12　Muleから呼び出されたCannaの拡張機能

また、Cannaには、辞書の作成（mkdic）、削除（rmdic）、複製（cpdic）、辞書名の表示（lsdic）、辞書名の変更（mvdic）、内容表示（catdic）、単語の一括登録／削除（addwords/delwords）、辞書形式の変換（mkbindic/dpbindic）、ローマ字かな変換テーブルの変換（dpromdic、mkromdic）など、辞書を管理する多くのコマンドが用意されている。これらのコマンド名は、同じ働きをするUNIXコマンド名にdicをつけただけのコマンドがほとんどなので、わかりやすいだろう。

では、この中でも利用する機会が多いと思われる、辞書の登録と単語の一括登録の方法を紹介しておこう。

■辞書の登録

辞書は、新規に作成するよりも、他人の辞書やCannaで用意されている拡張辞書など、既存の辞書を登録するケースのほうが多いだろう。既存の辞書を登録するには、辞書の新規作成コマンドと同じ、mkdicコマンドを-lオプションつきで利用する。

```
$ mkdic -l another.t another
New dictionary "another" is created.
Please change customize file.
```

■単語の一括登録

単語を一括登録するには、addwordsコマンドを利用する。

```
$ cat newword
あーかいぶ #T30 アーカイブ
あーきてくちゃ　　#T35　アーキテクチャ
　　…
$ addwords user < newword
Addwords has done on "user"
```

Wnn

Wnn（うんぬ）は、京都大学、オムロン、アステックによって共同開発された、クライアント／サーバ方式の日本語入力システムである。Wnnという名前は、「Watashino Namaeha Nakanodesu（私の名前は中野です）」を一発で変換できるシステムということから、この文の文節頭のアルファベットをとって名づけられた。

かな漢字変換サーバには「jserver」が担当し、フロントエンドプロセッサ「uum」、Xのインプットメソッド「kinput2」「xwnmo」、マルチリンガルエディタ「Mule」などから利用できる。Wnn4とWnn6という2種類あり、それぞれ異なる特徴を持っている。

■Wnn4

Wnn4は、1989年にリリースされたもので、各種ヨーロッパ言語、日本語、中国語（簡体字、繁体字）、韓国語に対応した変換サーバ（日本語変換サーバ部分はjserver）と各言語用の辞書が含まれており、マルチリンガルな入力システムという特徴を持っている。なお、Wnn4.2をGPLに即した形に再整備したFreeWnnがFreeWnnプロジェクト（http://www.freewnn.org/）によって公開されている。

■Wnn6

Wnn6 for Linuxは、オムロンソフトウェア社によって販売されているクライアント／サーバ方式の商用かな漢字システムである（http://www.omronsoft.co.jp/SP/pcunix/wnn/index.html）。最新バージョンは3.01。

Wnn6では、フリーで配布されていたWnn4からさらに変換効率やカスタマイズ機能が強化されている。具体的には、Wnn4に文節区切りを判断させるアルゴリズムを追加して、文節区切りの間違いによる誤変換を減らしたことや、「FI関係辞書」という用例を記述した辞書によって、同音異義語の選択を最適化した「FI変換」と呼ばれる機能、さらに、ユーザーの確定情報を学習しておく「FI学習」という機能が追加されたことにより、変換効率がWnn4よりも飛躍的に向上している。

さらに、ユーザー辞書や頻度ファイルなどを再配置して変換効率の向上を図ったり、不必要な単語を削除してディスク、メモリ資源の確保を行う、「オフライン学習機能」という機能もある。

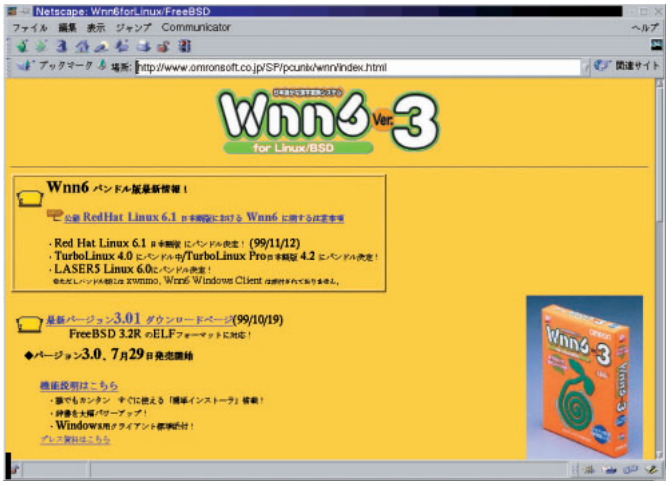
Wnn6は、TurboLinux 4.x、公認Red Hat Linux 6.1日本語版、LASER5 Linux 6.0、OpenLinux 2.3日本語版、Vine Linux 1.1CRW、Linux MLD4といった数多くのディストリビューションの製品版にバンドルされている。さらに、個人向けのパッケージは、「1ユーザー、同時に5セッションまでの接続」というライセンスで、9800円で単体販売されている。

Wnnの設定方法

前述のとおり、Wnnにはフリー版と製品版という2種類が存在している。最近の日本語ディストリビューションの製品版にはWnn6がバンドルされていることが多い。そのため、Wnn4はインストールされていないことが多いようだ。以下、Wnn6の設定方法を示すが、どちらも格納されているディレクトリが異なるだけで、設定方法には大差ない。

■変換サーバの起動

まず、かな漢字サーバを起動する。Wnnの場合は、




```
$jserver

Nihongo Multi Client Server (Wnn6 R3.00)
[socket_init_in] Transport endpoint is not
connected(Jserver already running)
```

画面13　jserverを二重起動した際のエラーメッセージ

```
$ wnnstat
ユーザ名:ホスト名        (ソケットNo.)    環境番号
mitsug-k:                  (1)              1
mitsug-k:unix              (2)              1
```

画面14　jserverへの接続状況を調査

```
$ jserver
```

とする。ここで、**画面13**のようなエラーが表示された場合は、すでにjserverが起動していることを意味する（最近のディストリビューションの多くは、/etc/rc.d以下に設定しておき、自動的にjserverが起動するように設定されていることが多い）。

jserverへの接続状況などを知りたいなら、wnnstatコマンドを実行する（**画面14**）。この例では、2人（といっても同一ユーザー）がアクセスしていることを表している。

ちなみに、「ソケットNo.1」のmitsug-kはリモートマシンから、「ソケットNo.2」の「mitsug-k」はコンソールから、それぞれjserverにアクセスしていることを表している。

また、あまり必要になることはないと思うが、意図的にjserverを停止したいなら、

```
# wnnkill
```

とすればよい。

■FEPからの利用

uum（この名前はWnnのアルファベットを180度回転させたところから由来している）は、jserverのFEPとして機能するコマンドである。Cannaの項で説明したcanuumは、このuumを元に作られているため、当然ながらuumの挙動や起動オプションはcanuumとほとんど同じである（**画面15**）。したがって、ここでは利用方法については紹介しない。

なお、Wnn6にはuumコマンドはバンドルされていない。どうしてもuumを利用したいようなら、Wnn4からuumコマンドを持ってくる必要がある。

■kinput2からの利用

Wnnのインプットメソッドには、xwnmoとkinput2がある。kinput2は、cannaと同様に起動時に

```
$ kinput2 -wnn &
```

と入力するだけで、あとはCannaと同じなので、詳細については説明しない。

なお、kinput2のバージョンが3.0以前のものは、Wnn6に対応していない。具体的には、「公認Red Hat Linux 6.1日本語版」にバンドルされているkinput2は、Wnn6に対応していない。そのため、前述のオムロンのWebサイトなどからバージョン3.0以降のkinput2パッケージを入手し、インストールする必要がある。

■xwnmoからの利用

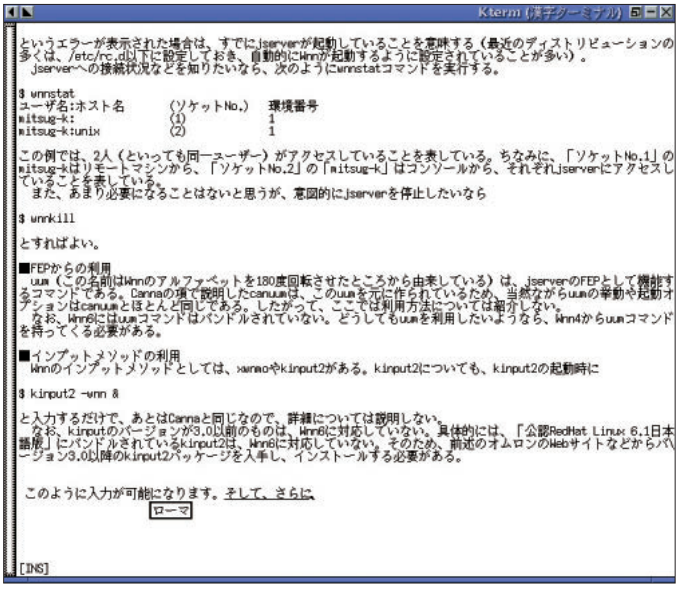
Wnn6のもうひとつのインプットメソッドとしてxwnmoがある。xwnmoは、

```
$ xwnmo
```

でデスクトップ左下の隅にアイコンの形で起動する（**画面16**）。このアイコンをクリックすると、ユーザー辞書の登録、学習頻度設定、キーカスタマイズといった、さまざまなカスタマイズをグラフィカルなユーティリティから設定できるようになっている（**画面17**、**18**、**19**）。

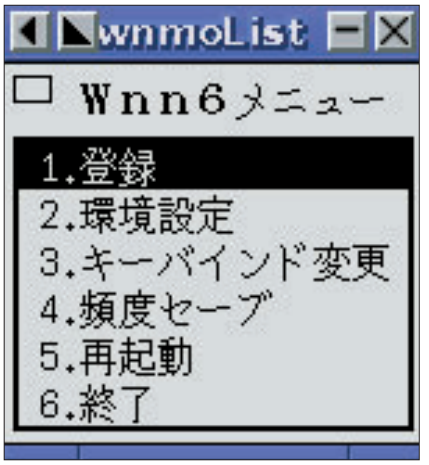
■Muleからの利用

Muleは、「たまご」と呼ばれるインターフェイス（実体はegg.elというEmacs Lispファイル）からjserverと直接通信



画面15　kinput2＋viによるviへの日本語入力

画面16　xwnmoの環境設定メニュー



画面17　Wnn6メニュー

し、日本語入力環境を実現している。

・変換サーバの指定

MuleからWnnを利用するには、利用するjserverをM-x set-wnn-host-name（ESCキーのあとにxキーを押して、set以降の文字列をMuleの最下行のミニバッファに入力し、Enterキーを押す）を実行して表示される、「Host name: 」にマシン名を入力して設定する。ここで指定するのは、ローカルホスト、リモートホストのどちらのマシン上で動作しているjserverでも指定することができる。

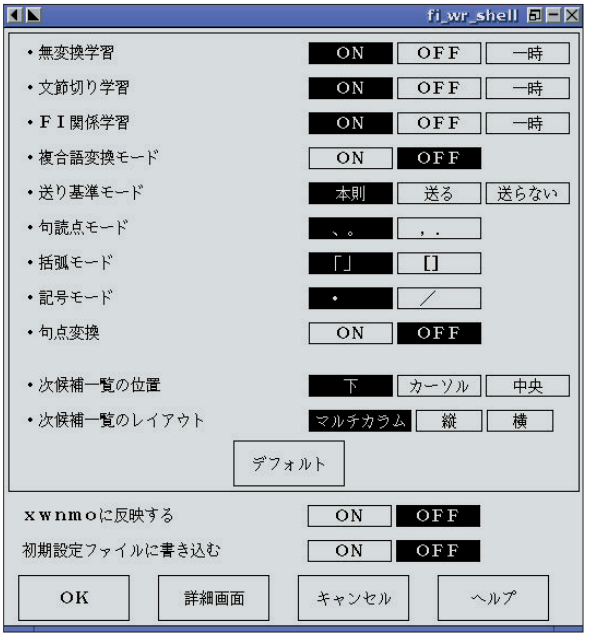
なお、毎回このような設定をするのが面倒（　いつも特定のjserverに接続する）ようなら、環境変数JSERVERを~/.bash_profileに指定しておくか、次のように.emacsに設定しておけばよい。

```
(if (and (boundp 'WNN) WNN)
    (progn
     (set-wnn-host-name "localhost")
))
```

Muleが起動したら、Ctrl+\キーを押してモード表示が**［あ］**に変われば、日本語入力が可能である（**画面20**）。

・単語登録

画面18　xwnmoの［キーバインド変更］メニュー



画面19　xwnmoの［環境設定］メニュー

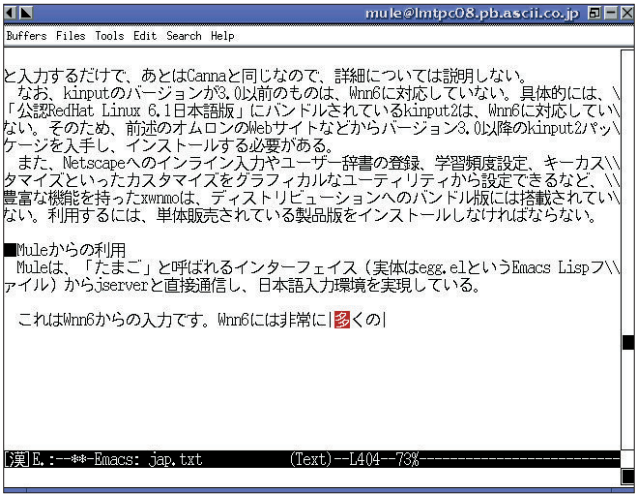
単語を辞書に登録しておくには、登録したい語句をMuleバッファの中でリージョン指定（具体的な指定方法は、Muleの書籍を参照してほしい）しておき、M-x toroku-regionを実行し、ミニバッファに表示される指示（登録する辞書や品詞の指定）にしたがっていけばよい。

・記号などの入力

Ctrl＋^と入力すると記号入力のモードに移り、ミニバッファから入力する記号の種類を選択できるようになる。ここで、選べるのは、

- **JIS入力**
- **記号入力**
- **英数字**
- **ひらがな**
- **カタカナ**
- **ギリシャ文字**
- **ロシア文字**
- **罫線**
- **部首入力**
- **画数入力**

などである。また、名前がつけられている記号は、“ @ゆうびん ”と入力して変換すれば、“ 〒 ”となる。



画面20　muleからのWnnの利用

ATOK12 SE for Linux

ATOK12 SE for Linux（以下、ATOK12 SE）は、ジャストシステムによって開発された日本語入力システムで、WindowsやMacintosh環境で人気の高いATOK（Advanced Technology of Kana Kanji Transfer）のLinux版である。ATOK12 SEは、WindowsなどのATOK12に比べ、いくつかの機能が限定されているものの、変換効率や入力補助機能（「ら」抜き言葉の指摘など）はそのまま利用できる。

現在は、単体販売されずOEM提供だけとなっているため、利用するには、バンドルされているTurboLinux 4.x、公認Red Hat Linux 6.1日本語版、LASER5 Linux 6.0、OpenLinux 2.3日本語版を購入しなければならない。

■ATOK12 SEの設定方法

ATOK12を利用するには、次の手順で設定を行う。

1.変換サーバの状態を調べる

変換サーバが起動していないか、次のコマンドで調べる。

```
$ /etc/rc.d/init.d/atok12se status
```

これで、「atok12x 633 is running ...」と表示されれば、すでに変換サーバは起動している（atok12のあとの数字はプロセスIDなので、毎回変わる）。ここで、「atok12x is stopped」と表示された場合は、変換サーバが起動していないということなので、

```
$ /etc/rc.d/init.d/atok12se start
```

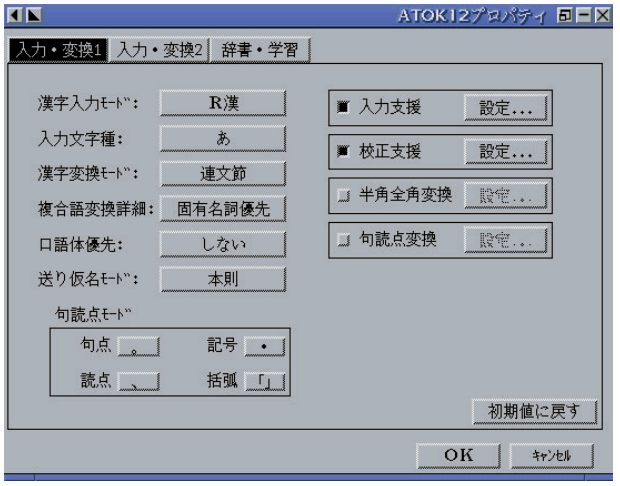
として起動すればよい。

2.入力サーバの起動

入力サーバを起動する。

```
$ kinput2x &
```

とする。ここで、入力サーバとしてkinput2ではなく、ATOK用に拡張されたkinput2である、kinput2xを利用することに注意が必要である。CannaやWnnの変換サーバとATOK12 SEの変換サーバは共存可能だが、CannaやWnnの入力サーバであるkinput2とkinput2xは同時に起動することができないディストリビューションがあるので、注意してほしい。



画面21　ATOK12 SEのプロパティ設定

3.Shift＋Spaceキーを押す

ここで、ShiftキーとSpaceキーを同時に押すと、カーソル位置の上に**［あ 連R漢］**というインジケータが表示され、日本語入力が可能になる。記号などを入力するにはF10キーを利用するところや標準のキーバインドは、DOS/Windows版などと同じ仕様なので、ATOKを使ったことがあるユーザーにはわかりやすいだろう。

■ATOK SEの環境設定

ATOK SEの環境設定は、

```
$ atok12prx
```

で起動する、Xベースのアプリケーションから行う（**画面21**）。

ここでは、入力方式（ローマ字入力かカナ入力か）、変換モード、送り仮名モード、句読点モードといった設定から、ATOK独自の入力支援機能（校正、誤り自動修復など）の設定ができる。また、辞書ファイルの設定や学習方式の設定もここから行う。

■Muleからの利用

ATOK12 SEは、kinput2x経由、つまりktermやviなどのうえでしか利用できない。

しかし、ATOK12 SEを利用したいMuleユーザーのために、武 靖浩氏によって「えせかんな」というツールが作られている（http://plaza.harmonix.ne.jp/~redstar/esecanna.html）。これは、ATOK12 SEの変換サーバをcannaserverに見せかけることで、Muleからは「かんな/emacs」経由でやり取りすることができるというものだ。変換効率に不満を感じているMuleユーザーは、一度試してみてほしい。

SKK

CannaやWnnは、UNIXには古くからある日本語入力システムなので、これらの名前を聞いたことのある読者も多いだろう。またATOKも、DOS/WindowsやMacintoshなど、多くのプラットフォームで定評のある日本語入力システムなので、よもや知らないというユーザーはいないはずだ。では、これから紹介するSKKはどうだろうか？ ご存じない方も多いのではないだろうか？

■SKKとは？

SKKは、現京都大学の佐藤雅彦氏によって開発された日本語入力システムである（http://skk.kuis.kyoto-u.ac.jp/skk/）。Emacsのマクロ言語である「Emacs Lisp」だけで記述されており、Emacs系エディタから利用することが可能である。逆にいえば、Emacs以外のエディタやターミナルエミュレータ上からは利用できないということを意味する。

SKKの名前は、「Simple Kana Kanji conversion program」の頭文字をとって名づけられたもので、GPLに則ったフリーソフトウェアである。このsimpleという名前のとおり、文法解析をいっさい行わず、漢字変換部分や送り仮名の始まりを、ユーザーが大文字で入力することにより指定するという、一風変わった特徴を持つ。

■SKKの設定

では、SKKを実際に体験してみよう。ただし、TurboLinuxにバンドルされているMuleにはSKK用のEmacs Lispが用意されていないようである。前述のサイトからダウンロードすれば利用できるようになるが、SKKを体験するだけなら、SKK用のEmacs Lispが用意されているXEmacsを利用してみよう。~/.emacsに**リスト3**のような設定を書き足しておき、XEmacsを再起動する。

■入力の実際

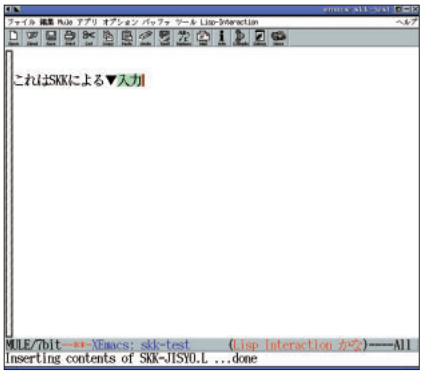
たとえば、XEmacsのバッファ上で、

これはSKKによる入力です。

と入力するなら、まず、Ctrl＋X、Ctrl＋Jと入力して、ローマ字かな入力モードにする。そして、「koreha（これは）」と入力したところで、「l」キー（小文字のエル）を押してSKKモードになってから、「SKK」と入力する。そのあと、Ctrl＋Jを押して再びローマ字かな入力モードになってから「niyoru（による）」を入力したところで、「Q」（大文字のQ）を押して、▽モード（読みを入力するモード）になって「nyuuryoku（入力）」と入力してから、Spaceキーを押して「入力」と変換する。そのあと、再び、Ctrl＋Jを押して、「desu.（です）」と入力する。これをまとめて書くと、**画面22**のようになる。また、XEmacs上での入力の様子は、**画面23**のようになる。

■根強い人気を持つSKK

一見、面倒な印象を持つかもしれないが、ユーザー自身が変換を完全にコントロールできるため、ストレスが少なく、ヘビーなEmacsユーザーには根強い人気がある。開発当初は、Emacs Lispだけで記述されたものだったが、現在は共有辞書にアクセスするサーバ「skkserv」も作られており、こちらも利用できるようになっている。また、フロントエンドプロセッサ「skkfep」やXのインプットメソッド「skkinput」なども開発が進められている。



画面23　SKKによるXEmacsへの日本語入力

koreha L SKK Ctrl + J niyoru Q nyuuryoku SPACE Ctrl + J desu.

画面22　入力の手順

▼リスト3　SKK用の.emacs設定例

```
(setq skk-large-jisyo "/usr/lib/xemacs-20.4/etc/skk/SKK-JISYO.L")
(global-set-key "\C-x\C-j" 'skk-mode)
(global-set-key "\C-xj" 'skk-auto-fill-mode)
(global-set-key "\C-xt" 'skk-tutorial)
(autoload 'skk-mode "skk" nil t)
(autoload 'skk-auto-fill-mode "skk" nil t)
(autoload 'skk-tutorial "skk-tut" nil t)
(autoload 'skk-isearch-mode-setup "skk-isearch" nil t)
(autoload 'skk-isearch-mode-cleanup "skk-isearch" nil t)
(autoload 'skk-check-jisyo "skk-tools" nil t)
(autoload 'skk-merge "skk-tools" nil t)
(autoload 'skk-diff "skk-tools" nil t)
```

Section4

チャレンジしてみることが大切

読者のLinuxは、無事「日本人」になれたであろうか？　今回は、「表示と入力」というテーマで日本語環境の構築方法を説明してきた。しかし、読む（表示）や書く（入力）だけでは、「日本人」といえるかもしれないが、子供のようなものだ。表現（印刷）やコミュニケーション（通信）などができるようになって、初めて「大人の日本人」といえるだろう。このあたりについては、また機を改めて紹介するつもりだ。

日本語変換システムは何を選ぶ？

ところで、［入力］の項では、いろいろな日本語変換システムを紹介したが、何を選べばいい、と悩んでいる初心者も多いと思う。もちろん答えは「好み次第」となるのだが、筆者の独断ながら、選び方の基準などについて少し補足しておこう。

■選び方のポイント

ktermでも使いたいし、Emacsでも使いたいし、という欲張りな環境を求めるなら、WnnかCannaという選択になる。さらにこの中でも、フリーですべて環境を構築したいならCannaかFreeWnnとなり、有料でも変換効率を求めるなら、Wnn6という選択肢になるだろう。

また、「Emacsはエディタではなく環境である」という「Emacs命」なユーザーなら、SKKも検討材料に入るだろう。SKKは、

- **文節区切りの間違いが防げる**
- **root権限がなくてもインストールできる**

といった利点がある。多少、クセのある（というよりも馴染みの薄い方式の）変換システムではあるが、慣れれば問題はない。

なお、今回紹介した中で筆者が独断で選ぶなら、

- **高い変換精度**
- **使い慣れている**

という理由から、ATOK12 SEを推したい。

ただ、ATOK12 SEは単体で販売されていないので、「ATOK12 SEがバンドルされている製品」という狭い範囲内でディストリビューションを選ばなくてはならない。いくらATOK12 SEがよくても、これでは本末転倒な気がする。

■筆者は？

というわけで、本当はATOK12 SEを使いたいが、ふだんはメディアラボのLinux MLD4を利用している筆者が採っている方法は、LinuxマシンにWindowsからtelnet接続し、そこにMS-IMEから入力するというものである（**画面24**）。もちろん、Windowsはフリーではないが、まだまだビジネス環境では不可欠であり、筆者のメイン環境は依然としてWindowsである。というわけで、どうせWindowsマシンがもう1台あるなら、そこからLinuxにtelnet接続して、日本語入力をすれば、

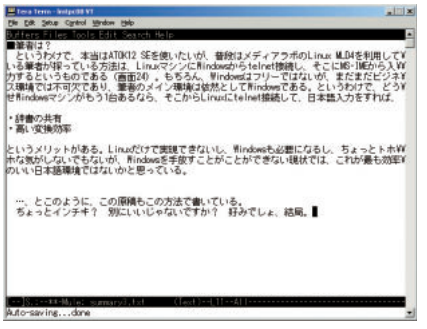
- **辞書の共有**
- **高い変換効率**

というメリットがある。Linuxだけで実現できないし、Windowsも必要になるし、ちょっとトホホな気がしないでもないが、Windowsを手放すことがことができない現状では、やむなしの策という感じである。

でも結局は慣れ

とはいえ、何事も「慣れ次第」という部分はかなり大きい。いくら使いやすいといわれているものでも慣れていないと使いづらく感じるだろうし、その反対の場合もある。

つまり、とりあえず使ってみればいいのでは、と提案したい。Windowsとは違い、Linuxではいろいろ試すのにお金はかからないことが多い。いろいろ苦労しているうちに、自分にしっくりくるものが見つかるはずだ。慣れてしまえばこっちのものだし、周りから「なぜ、そんなものを…」といわれるのも、快感に思えてくるはずと筆者は思っている。



画面24　Teraterm＋IME 2000によるMuleへの日本語入力

x86だけがLinuxじゃない!

MACINTOSH DE LINUXPPC

Linuxは、一人のハッカーが「x86の勉強をする」ために作り始めた。そしてオープンソースコミュニティの力により、CPUアーキテクチャの壁を軽々と飛び越え、多くのCPUで動くようになった。なかでもPowrPCは、大人気のiMacに搭載されたこともあり、Linuxが動くものとしては、x86に次いで普及しているCPUだ。PowerPCとLinuxの関係にせまってみよう。

Photo : Shuichi Mito (Dee)

Macintosh de LinuxPPC

これまでのLinuxPPC

Macintosh de LinuxPPC

もともとLinuxは、インテルのx86プロセッサを搭載したPC用のOSとして開発されたもので、当然x86特有の機能を用いていた。そのため他のプロセッサへの移植は困難だと言われてきた。しかしコミュニティのメンバーによる努力のおかげで、現在ではPowerPCをはじめとしてAlpha、SPARCなど多くのプロセッサ上でLinuxが動いている。なかでもPowerPCは、アップル社のMacintoshで採用されていることもあり、x86に次ぐLinuxプラットフォームと言っていいだろう。

PowerPC版Linux開発の歴史

PowerPC用のLinuxは、マイクロカーネルを用いたものと、x86と同様のモノリシックカーネル版に大別される。

マイクロカーネル版はMkLinuxと呼ばれ、アップル社とオープンソフトウェア財団（現The OpenGroup）との協同プロジェクトから生まれた。ハードウェアメーカーによるバックアップのもとに開発が進められていたわけだが、'98年の末にアップル社はMkLinuxの開発チームを解散した。同社の新OSであるMac OS Xの開発に、社内のリソースを集中させるためだったと言われている。以後MkLinuxは開発の主体がコミュニティに移り、現在も開発が続けられている。初代のPower Macintosh（バスの名前から「NuBus PowerMacintosh」と呼ばれる）などのモノリシックカーネル版が動作しない機種でLinuxを使いたい場合には、MkLinuxが唯一の選択肢だ。

一方、モノリシックカーネル版のLinuxは2つの系統で開発が始められたが、'97年の初頭には統合され1種類の「LinuxPPC」となった。LinuxPPCは、MkLinuxのバイナリがそのまま利用できるうえに、MkLinuxよりもパフォーマンスが高かったため、Power Macintosh用Linuxの主流となった。

また安定版カーネルのバージョンが2.2になる際に、Linux全体のカーネルソースにPowerPC版も統合され、これ以降は、ほかのプロセッサと歩調を合わせて開発が行われることになった。

新機種にも素早い対応

Macintoshは、アップル社だけが製造、販売している（一時は互換機が販売されていたこともあったが）せいか、モデルチェンジの際には、ハードウェアの構成が大きく変更されがちだ。数多くのメーカーが膨大な種類の部品を製造しているため、急激な変化が起きにくいPCとは対照的である。

昨年発売されたiMacや、俗に「青白」と呼ばれるPowerMacintosh G3（**写真1**）はその好例だろう。新しく採用されたPowerPC750（G3）（**写真2**）の高い性能だけでなく、フロッピードライブ、シリアルポート、ADB（Apple Desktop Bus）ポート、SCSIポートといった旧来のインターフェイスを廃止するという大胆さに驚いたものだ。これらの機種では、キーボードとマウスをUSBで接続しているため、当時USBをサポートしていなかったLinuxを使うことはできなかった。しかし、iMacユーザーを中心にUSBサポ

写真1 PowerMacintosh G3 400 MHz
その姿から「Blue & White」と呼ばれたり、使用しているマザーボードのコードネームから「Yosemite」と呼ばれる。

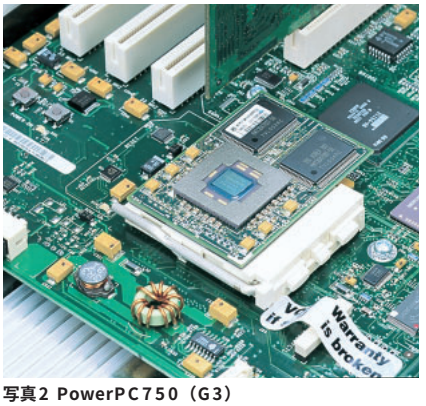
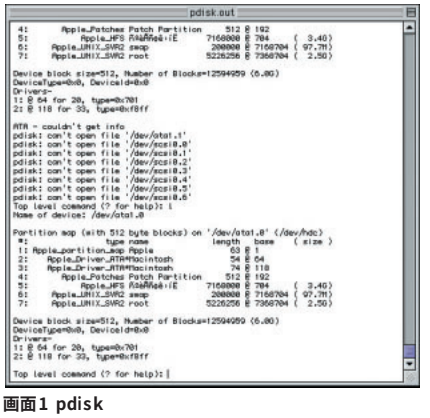


写真2 PowerPC750（G3）
「G3」は「Generation 3」、PowerPC系CPUの第3世代の意味。インテルのPentiumIIIなどと同じように、2次キャッシュ専用のバスを持つことで大幅に性能を向上させた。またx86系CPUと比較して、消費電力が低いのも特徴だ。



画面1 pdisk
fdiskとほぼ同機能のコマンド。fdiskに慣れていれば、それほど違和感なく使えるが、MacOSの操作に慣れているユーザーには、奇異に感じるだろう。

ートが進められた結果、短期間でiMacやG3に対応したカーネルがリリースされ、これらの機種でもLinuxが使えるようになった。もちろんこの成果はPC用のLinuxにも活かされており、カーネルを再構築することで安定版、開発版のどちらでもUSBデバイスが利用できる。

LinuxPPCの特徴

Linux起動後のデスクトップを眺めているかぎりは、PowerPC版もx86版も同じであり、違いはないといってよい。だがその手前、インストールや起動の際には、x86版とは別の点に気をつけなければいけない。

●パーティション作成

PCでもMacintoshでもパーティションの作成は、初心者には大変な作業だ。LinuxPPCの場合、パーティションの作成は、PC用のfdiskとほぼ同機能のpdiskというユーティリティを用いて行う。LinuxPPCのインストーラからpdiskを用いてもいいし、LinuxPPCのCD-ROMに含まれているMacOS用の同名のプログラムを用いることもできる（**画面1**）。

MacOS上からもパーティション作成ができるようになっているのは、Linuxに不慣れなMacintoshユーザーへの配慮なのだろう。だが実際の作業は、MacOSでもLinuxインストーラからでも同じで、アルファベット1文字のコマンドを入力していくというMacOSの作法とかけ離れたものだ。Macintoshユーザーによる Linuxインストールの最大の難関が、このパーティション作成だといっても過言ではない。

●他のOSとの共存

1台のPCにLinuxとWindowsを共存させるには、それなりの知識と経験が必要だ。ブートローダは、MBRかブートセクタのどちらに置くかとか、カーネルは8 Gバイトより手前の領域にあるかといったことに、気を使わねばならない。

ところが、LinuxPPCでは、BootX（**画面2**）というブートセレクタが提供されているため、マシンのスイッチを入れたときでも、MacOSの使用中でも、簡単にLinuxPPCを起動できる。また起動時にカーネルに渡すパラメータもBootX内で指定できるため、いちいちテキストファイルを編集する必要がない。

●周辺機器への対応

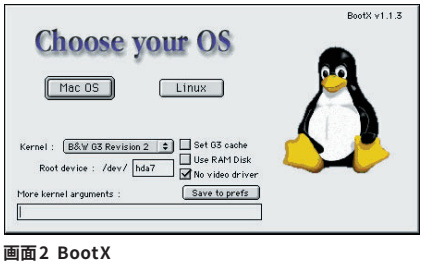
Macintoshの周辺機器や部品は、PCと比較すればそれほど多くはない。そのため、買ったままのMacintoshならば、ハイエンドのSCSIアダプタを除き、対応ドライバがなくて困るということはほとんどない。しかしサードパーティ製品については、十分にサポートされているとは言いがたい状況だ。LinuxPPCを使う際には、シンプルな構成のマシンが適当と言えるだろう。

Macintoshのマウスは1ボタンであり、3ボタンが前提となっているX Window Systemでは使いにくい。そこで標準のマウスを使う場合は、足りないボタンをキーボードで代用することになるのだが、使い勝手は良くない。

ディストリビューション

国内では、アミュレットからLinuxPPC日本語版（**写真3**）が、マインドからLinux for PPC Japanese Edition 2（**写真4**）が発売されており、日本語対応したLinuxPPCが利用できる。

また、これ以外にMkLinuxをベースにしたディストリビューションとしては、PowerLinuxが日本フォトグラフィックスから発売されている。



画面2 BootX
MacOSの起動時に現れるBootXのウィンドウ。MacOS使用中にも呼び出すことが可能だ。

写真3 LinuxPPC日本語版
4800円
株式会社市川充商店（アミュレット）
http://www.amulet.co.jp/ppclinux/

写真4 Linux for PPC Japanese Editon 2
9800円（オフィシャル版）
株式会社マインド
http://www.pc-mind.co.jp/

Macintosh de LinuxPPC

LinuxPPC 日本語版

Macintosh de LinuxPPC

LinuxPPC日本語版は、アミュレットから発売されているPowerPC搭載マシン用のLinuxである。米国LinuxPPC社（**画面3**）のLinuxPPC 1999 Q3に、日本語化キット「陽炎」をセットしたものだ。

Macユーザーに配慮したインストール手順

MacOS以外の環境に慣れていないMacintoshユーザーでも、Linuxのインストールができるように、いろいろな面に配慮がされている。実際のインストール時と同じ順序でそれらの工夫を紹介しよう。

●MacOS上のインストーラ

LinuxPPCをインストールする際には、MacOS上でいくつかのファイルをハードディスクにコピーする必要がある。手動でもできる程度の作業だが、LinuxPPC日本語版には、MacOS用市販ソフトにも用いられているインストーラが付属しており、インストーラ起動後は、デフォルトのボタンを選んで押していくだけで、インストール終了→再起動まで自動で行える。

●ライブインストール

MacOS上の作業終了後、再起動すると、ブートセレクタであるBootXの画面が表示される。ここでカーネルを選択してLinuxのボタンを押すと、最初からWindow Makerを用いたX Window Systemが起動する（**画面4**）。これはX Linux Installer（XLI）と呼ばれるものだ。

LinuxPPCのインストーラを起動する際には、RAMディスク上に最小限のシステムが作成される。そのほかインストールに必要なプログラムは、CD-ROM内のlive.filesystemというファイルにファイルシステムのイメージとして納められており、これをマウントして用いている。

画面4の状態で、最低限のコマンドを備えたLinuxとして扱うことができる。たとえば、MacOSのパーティションを手動でマウントして、中身を確認することも可能だ。また、この状態からxtermを起動して、pdiskコマンドでパーティションの作成を行うこともできる。パーティションの作成はMacOS上からもできるので、各人がやりやすいと思う方法でやればいいだろう。

LinuxPPCでは、最低でもスワップ、「/」（ルート）の2つのパーティションが必要だ。これらのパーティションが用意できたら、後はXLIのボタンを上から順に押していくことでインストールが進んでいく。

インストール作業は、パーティションの選択（**画面5**）、パッケージ選択、ユーザーパスワード入力、ネットワーク設定の順で行われる。PC用のLinuxをインストールした経験があれば、作業の内容はほとんど同じだということが分かるだろう。

●マニュアル

LinuxPPC日本語版には、印刷されたマニュアルは付属しないが、「陽炎」のCD-ROM内にHTML形式のインストールマニュアルが含まれている。インストールの過程をていねいに説明してあるので、MacOSしか使ったことのない人にも扱いやすいだろう。できればインストール時には、印刷して手元に

画面3 LinuxPPC Inc.
LinuxPPC社のwebサイト。

画面4 ライブインストール
MacOSでの設定がうまくできていれば、再起動後にこの画面が表示される。最小限のコマンドが使えるX Window Systemそのものである。この時点でターミナルを開いて、パーティションの設定を行うこともできる。

画面5 パーティションの選択
パーティションのマウントポイントを選択する。またスワップ用のパーティションもここで選ぶ。通常は、このように「/」とスワップパーティションの2つだけで十分だ。

置くといいだろう。

パッケージの内容

日本語版のベースになっているLinuxPPC 1999 Q3は、昨年の9月に発売されている。そのせいか、カーネルは2.2.6、Cライブラリはglibc 2.1.1と最新のものではないが、特にパフォーマンスやセキュリティの面で問題はないだろう。必要ならばソースからコンパイルしたり、自分が持っている機種に合わせてコンパイルされたバイナリを手に入れることで、最新版に置き換えることは可能だ。

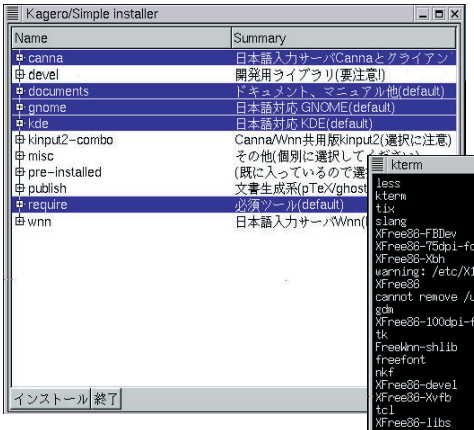
デスクトップ環境のデフォルトは、GNOME 1.0.9となっているが、同時にKDE 1.1.1も収録されており、自分の好みに合わせて利用すればいいだろう。

日本語化パッケージ「陽炎」

「陽炎」は、LinuxPPC 1999 Q3と組み合わせて、日本語環境を実現するパッケージだ。LinuxPPCの次期バージョンにも統合される予定になっている。

インストール方法は簡単で、CD-ROMをマウントして「setup」コマンドを実行するだけだ。あとはインストーラ（**画面8**）から必要なパッケージを選択して、「インストール」ボタンを押すだけでよい。余談だが、このインストーラ本体は、話題のオブジェクト指向スクリプト言語「Ruby」で書かれている。

「陽炎」をインストールすると、XFree86は3.3.5にX-TTパッチを当てたものにアップデートされるので、GNOME（**画面9**）やKDE（**画面10**）でTrueTypeフォントを用いた日本語表示が可能だ。アプリケーションもGUIでrpmパッケージを扱えるgnorpmや、menueditなどが日本語化されており（**画面11**）、Linuxに不慣れなMacOSユーザーでも安心して使えるようになっている。



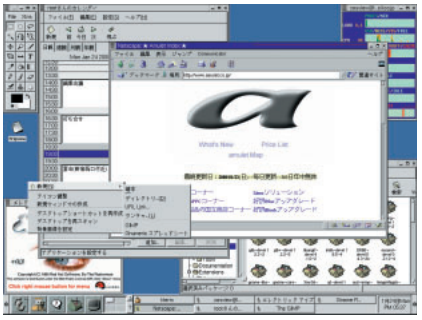
画面6 日本語化パッケージ「陽炎」
LinuxPPCに追加インストールすることで、日本語環境が実現できる。インストーラは、「Ruby」のスクリプトで書かれている。

JISキーボードに問題あり

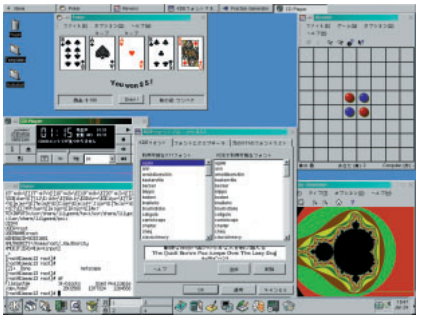
LinuxPPC日本語版のパッケージに含まれるものだけで、日本仕様のMacintoshをほとんど問題なく使用することができたが、唯一JISキーボードの扱いだけがうまくいかなかった。これは、JISキーボード用のキーマップデータを持っていないためだ。編集部でPowerMac G3にインストールした際には、以下のページを参考にして対処した。

http://www.y-min.or.jp/~nob/LinuxPPC/BlueG3/USB.html

今回、アミュレットのご厚意で、読者プレゼント用にLinuxPPC日本語版を提供していただいた。Macintoshを持っているなら、これを機会に試してみてはどうだろうか。



画面7 GNOME
デフォルトのデスクトップ環境は、GNOMEだ。



画面8 KDE
好みに合わせてKDEを使うこともできる。デスクトップ環境は、ログインダイアログから切り換えが可能。

LinuxPPC日本語版
4800円
株式会社市川充商店（アミュレット）
http://www.amulet.co.jp/ppclinux/

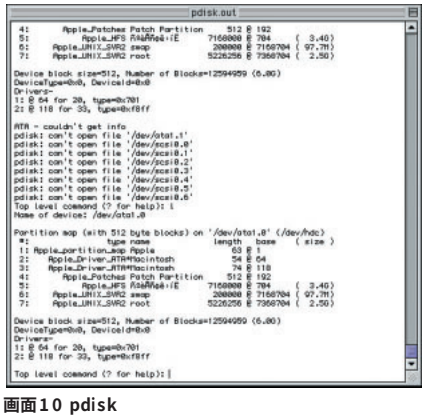
Macintosh de LinuxPPC

Linux for PPC Japanese Edition 2

Macintosh de LinuxPPC

Linux for PPC Japanese Edition 2（以下Linux for PPC）は、マインドから発売されているPowerMacintosh用のLinuxディストリビューションである。LinuxPPCをベースに日本語化を行ったスタンダード版と、商用ソフトとマニュアルを追加したオフィシャル版がある。オフィシャル版に含まれている商用ソフトは、日本語入力システムのVJE-Delta 2.5とDYNALAB社のTrueTypeフォント2種類である。以下の記述は、オフィシャル版を元にしている。

画面9 MacOS上での準備
指示に従って、ファイルを所定の場所にコピーしていけばよい。

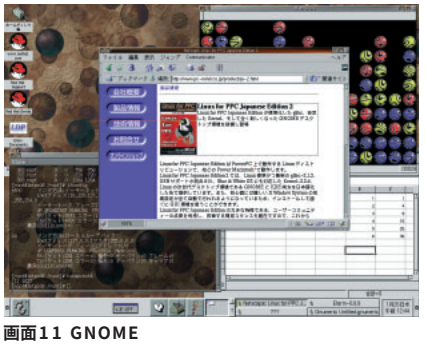


画面10 pdisk
パーティション作製用プログラム。これはMacOS上のものだが、ほぼ同じインターフェイスの同名のコマンドがLinux上にもある。分かりやすいほうを使えばいいだろう。

シンプルなインストール手順

Linux for PPCを使うためには、まずMacOS上での準備作業が必要である。1つは、CD-ROM内の「Install Files」というフォルダ内のファイルをハードディスクにコピーすることだ。それぞれコピー先が異なるのでめんどうにも思えるが、**画面9**のように表示されているので、まちがいようがないだろう。

もう1つは、Linux用パーティションの作成である。Linux for PPCに付属しているpdisk（**画面10**）というプログラムを用いるのだが、アルファベット1文字のコマンドを入力していくという、MacOSのインターフェイスとはまったく異なる操作をしなければならない。Macintoshユーザーには、これが最大の難関であろう。既存のデータを失ったりしないように、細心の注意を払おう。パーティションの切り方には、いろいろな流儀があるようだが、通常はスワップ用パーティションに100Mバイト程度割り当て、残りを「/」（ルート）パーティションに割り当てればよいだろう。



画面11 GNOME
デフォルトのデスクトップ環境はGNOMEだ。

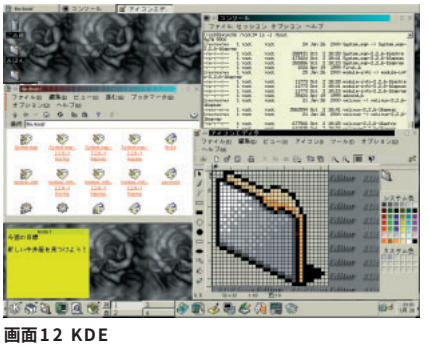
Red Hatスタイルのインストーラ

ファイルコピー終了後にMacintoshを再起動すると、ブートセレクタであるBootXが起動する。まずここで、インストールに用いるカーネルを選択する必要がある。「Japanese Edition default」か、自分のMacintosh用のカーネルを選択して「Linux」ボタンを押すと、インストーラが起動する。

インストーラは、PCでLinuxを使ったことがあればおなじみの、Red Hatスタイルである。テキストベースだが、すべて日本語化されているので、メッセージを確認しながら進めていけば、うまくいくだろう。

「普通のLinux」として使用可能

Linux for PPCは、カーネルに2.2.6、Cライブラリにglibc 2.1.2を用いている。カーネルは最新ではないが、特にセキュリティや性能の面で問題はないだろう。またXFree86は3.3.4にX-TTパッチを当てたものが採用されており、



画面12 KDE
ログインダイアログからKDEを選択することも可能。特に手を加えない状態では、KDEのほうが動作が軽快だ。

付属のDYNALAB社のフォントを用いて、日本語表示が行える。

いったんインストールが成功して、X Window Systemの画面が表示できれば、そこはもうPCのLinuxと変わらない世界だ。人気のデスクトップ環境であるGNOME 1.0.7（**画面11**）、KDE 1.1.1（**画面12**）はどちらも日本語化されており、安心して使えるようになっている。

日本語入力が可能なエディタとしては、GNOME環境用のエディタであるgEdit（**画面13**左）や、VJE-Pen（**画面13**右）が用意されている。VJE-Penは、VJE-Deltaのメーカーであるバックスがフリーソフトとして公開しているものだ。元々DOS用のプログラムであったため、操作体系は独特だが、以前DOSで使っていたユーザーには最適かもしれない。

そのほかLinuxを使用するうえでは、欠かせないNetscape Communicator 4.6やGimp（**画面14**）にも日本語リソースが追加されている。全体的にPC用Linuxと遜色ないレベルに仕上がっているといっていいだろう。

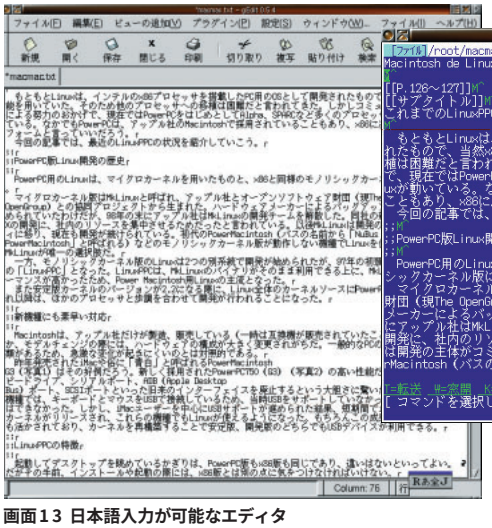
メーリングリストのログが付属

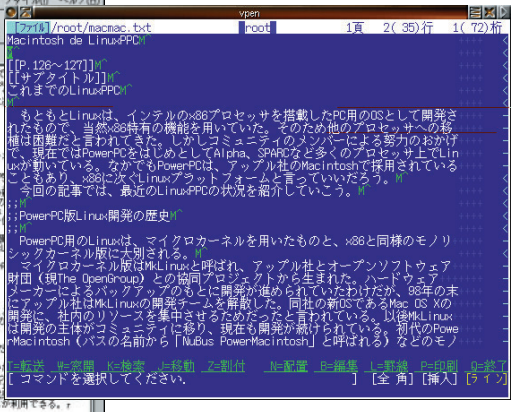
それでも使用していくうちには、機種特有の疑問点が出てくるかもしれない。MacintoshでLinuxを使っているユーザーは多いとは言えず、質問できる相手がいないということもあるだろう。

Linux for PPCのCD-ROMには、linuxppc-jp、linuxppc-jp-devなどのメーリングリストのログと、全文検索システムNamazuのGUIフロントエンドTkNamazu（**画面15**）が収録されている。TkNamazuは、Tcl/Tkのスクリプトであり、シンプルなインターフェイスから全文検索が可能だ。使用中に疑問点が出たら、ぜひ活用してみよう。自分と同じ疑問と、その解答が見つかるはずだ。

アップデートパッケージ

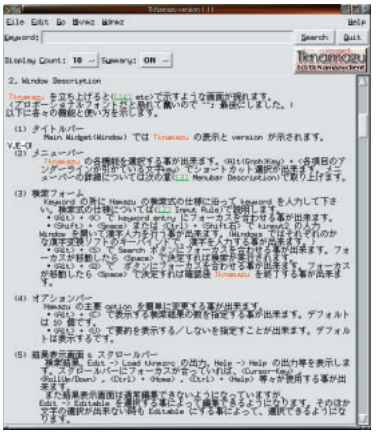
市販されているLinux for PPCのオフィシャル版のバージョンは2.0.1だが、マインドのFTPサーバで公開されている最新バージョンは、2.0.3である。同社のWebサイトでは、バージョンアップされているパッケージの一覧を公開している。セキュリティホールの解消など必須のアップデートもあるので、購入後は確実にチェックしておこう。



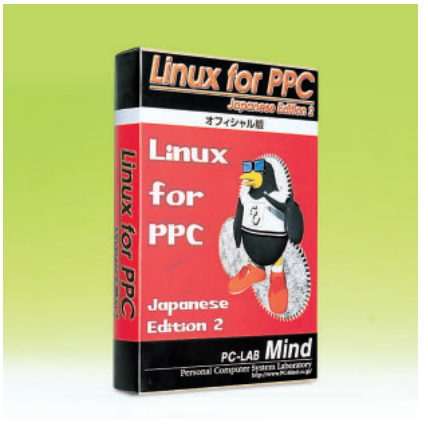


画面13 日本語入力が可能なエディタ
GNOME環境用のgEditやDOS用プログラムの移植版であるVJE-Penで、日本語テキストを作成できる。

画面14 Gimp
マスコットキャラクタWilberの助言も日本語化されている。



画面15 TkNamazu
疑問が浮かんだら、メーリングリストのログから検索してみよう。似たような質問とそれに対する答えが見つかるかもしれない。



Linux for PPC Japanese Editon 2
9800円（オフィシャル版）
株式会社マインド
http://www.pc-mind.co.jp/

Macintosh de LinuxPPC

LinuxPPC活用術

Macintosh de LinuxPPC

GNOMEやKDEのデスクトップ環境を使っているかぎりは、PC用のx86 LinuxとLinuxPPCの使い勝手に差はない。しかし、カーネルの更新やハードウェアに関連する部分では、やはりx86とは異なったテクニックが必要になってくる。ここでは、LinuxPPCを一歩踏み込んで使う際に必要となる知識や、一番気になるはずのx86とPowerPCの性能比較を紹介しよう。

USBサポート

LinuxPPCの最新版である1999 Q3は、カーネル2.2.6を採用している。これに組み込まれているUSBドライバは、UUSBDと呼ばれているものだ。しかしUUSBDは、コードが複雑すぎるという理由で、Linuxカーネルのソース群に取り込まれなかった。カーネル2.2.7以降で、UUSBDの代わりに採用されたのは、Linus Torvaldsも加わったチームで開発されたドライバ（USB）である。

UUSBDとUSBドライバには、互換性がない。カーネルを今の2.2.6から2.2.7以降のものに更新してしまうと、キーボードの割り当てが変わってしまい、操作できなくなってしまうので、注意が必要だ。

より新しいUSBドライバのほうが「ホイールマウス」に対応しているなど、機能的に優れている。しかし、腕におぼえがあるユーザー以外には、カーネル2.2.6からの更新はお勧めできない。LinuxPPCの次期バージョンで、USBドライバが正式に採用されるのを待つといいだろう。

写真5 アップル純正USBマウス
MacOSで使うには、適切な1ボタンだが、X Window Systemで使うにはまったく向いていない。

マウス

Macintoshのマウス（**写真5**）は、1ボタンが標準だ。MacOSは、1ボタンで使うように設計されているため、これでも特に使いづらいこともない。非常に優れたOSデザインと言えよう。もっともMacOS 8以降は、コントロールキー＋クリックなどの操作が加わり、多少無理があるように思えるが。

それに対してX Window Systemは、3ボタンを前提とした操作体系を持っている。2ボタンマウスが一般的であるPCでXを動かす際には、左右ボタンの同時押しで中ボタンのエミュレートをするようになっているが、1ボタンではそれさえもできない。

写真6 マウスボタンの代わりに使うキー
たいていの人が右手でマウスを扱う。ということは、これらのキーは左手で押すことになる。やってみれば分かるが、非常につらい姿勢になってしまう。

そこでLinuxPPCの場合は、マウスのボタンは左ボタンとして使い、キーボードのOption＋2を中ボタン、Option＋3を右ボタンに割り当てている。また、上で述べたUUSBDドライバを用いるマシン、つまりPowerMac G3やiMacなどでは、テンキー部分の「clear」キーが中ボタン、その隣の「=」キーが右ボタンに割り当てられている（**写真6**）。当然これらエミュレーションに用いられているキーは、本来のキーとしては使えなくなる。また実際に操作すると分かるが、Option＋2／Option＋3も「clear」／「=」も妙な姿勢を強いられてとても扱いにくく、非常に効率が悪い。片手で操作できないポインティングデバイスなど、わずらわしいだけといえよう。

3ボタンマウスの利用

やはりどう考えても、1ボタンで3ボタンマウスのエミュレーションをするのは無理がある。お勧めは、ホイールのついた3ボタンマウスを導入してしまうことだ。なおADBマウスで3ボタンの製品は、最近ではほとんど見かけないので、残念ながら以下の話はUSBマウスを用いる機種に限定されてしまう。

マイクロソフトがホイールマウスを売り出し、Windows 98でUSBデバイスが正式にサポートされて以来、USB接続のホイールマウスは数多く発売され、値段も安くなっている。MacOSで利用するためのドライバが付属している製品も多い。

普通のボタンが3つ付いたものでなく、ホイールマウスを勧める理由は3つ

ある。ひとつは、ホイールが中ボタンとしても使えるようになっており、機能的に上位互換であるということ。もうひとつは、カーネル2.2.7以降に採用されているUSBドライバがホイール機能をサポートしているため、Linuxでもいずれホイールが使えるようになるということ。最後のひとつは、ホイールマウスのほうが圧倒的に種類が豊富で、自分の手に合ったものを選択できるからだ。

市場に出回るすべてのマウスで動作確認をするのは不可能だが、編集部で試したホイール付きのUSBマウスは、いずれもLinuxPPC起動時にきちんと認識され、X Window Systemで3ボタンマウスとして利用できた。中でもエレコム社製のホイールマウスM-WULT4LGは、MacOS上でも2ボタン＋ホイールで使用でき、実売価格も3000円前後と手頃だった。

最初に純正マウスでインストールして、あとからホイールマウスに取り替えた場合、キーボードによる3ボタンエミュレーションが無効になるように、設定を変更する必要がある。エディタで/etc/sysconfig/mouseというファイルを**リスト1**のように書き換えよう。なお、この設定は、UUSBDドライバを用いたカーネル2.2.6だけで有効なものだ。USBドライバを組み込んだカーネル2.2.7以降では、使わないようにしてほしい。

現状では仕方のないことだが、市販されているホイールマウスは、Windows 98のみ、またはWindows 98／MacOSでの使用しか想定していない。メーカーや販売店に、個々の製品がLinuxPPCで使用可能かといったような問い合わせをするのは、遠慮してもらいたい。

PowerPC vs. x86

コンピュータの絶対的な性能は、なかなか実感しにくいものだ。比較の対象があるほうが理解しやすいはずだ。そこで、LinuxPPCをインストールしたPowerMacintosh G3 400 MHzと、Celeron 400 MHzを搭載したPCとで同じプログラムを走らせ、比較してみた。マシンのスペックについては、**表1**を参照してほしい。

ベンチマークプログラムや、ベンチマーク用途によく使われるプログラムはたくさんあるが、今回は以下の3つを用いた。

●x11perf／Xmark

グラフィックス関連の評価には、x11perfとその結果から算出されるXmarkを用いた。X Window Systemに含まれるx11perfコマンドは、直線や四角形といった基本的な図形の描画速度を算出するコマンドだ。この結果の加重平均を行い、総合的なXサーバの性能を示す値として、Xmarkが算出される。Xmark値は、SunマイクロシステムズのSparc Station 1（SunOS 4.1.2）の成績を1とした相対値で表される。

●nbench

CPUやメモリ回りの総合的な評価には、BYTEmarkをベースにしたnbenchを用いた。nbenchは、データのソートやフーリエ変換など、よく使われる10種類の計算を行い、メモリ、整数演算、浮動小数演算の項目別に性能を算出する。各データは、K6 233MHzを搭載したLinux PCの成績を1とした相対値で表される。

●mapmaker

実際のアプリケーションを走らせた際の評価として、mapmakerというプログラムを用いた。前の2つと違って、これはベンチマーク用のプログラムではない。遺伝子解析の分野で実際に使われているものだ（**画面16**）。計算の内容はメモリ間の転送やデータの比較が大半で、ディスクI/Oなどはあまり影響がない。tarボールに付属しているサンプルデータを処理するのにかかる時間で評価した。

PCに1日の長が

描画速度は、体感速度に直結するた

▼リスト1　/etc/sysconfig/mouseの変更。

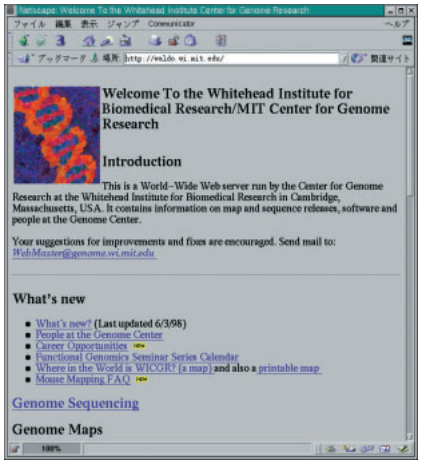
```
MOUSETYPE="Busmouse"
XMOUSETYPE="BusMouse"
FULLNAME="Universal Serial Bus 1 Button Mouse (USB)"
XEMU3=yes
```

↓

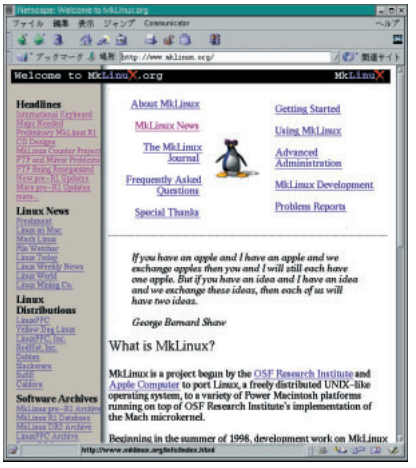
```
MOUSETYPE="Busmouse"
XMOUSETYPE="BusMouse"
FULLNAME="Universal Serial Bus 3+ Button Mouse (USB)"
XEMU3=no
```

	PowerPC	x86
プロセッサ	PowerPC750 400MHz	Celeron 400MHz
メインメモリ	SDRAM 320Mバイト（100MHz）	SDRAM 128Mバイト（66MHz）
グラフィックスカード	ATI Rage128	3dfx Voodoo3 2000
ハードディスク	E-IDE 6Gバイト	E-IDE 25Gバイト

表1 PowerMacintosh G3と比較用のx86マシンのスペック



画面16 mapmaker開発元のWebサイト



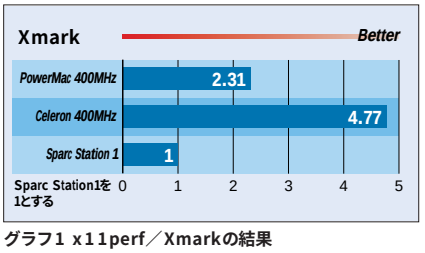
画面17 MkLinux.org

め、画面表示を高速化することは、たいへんに重要だ。x11perf／Xmarkの結果を**グラフ1**に示したが、PowerMacintoshは、同クロックのCeleronマシンに2倍の差をつけられてしまった。もちろん起動時の設定で、描画処理のアクセラレーションは有効にしてある。

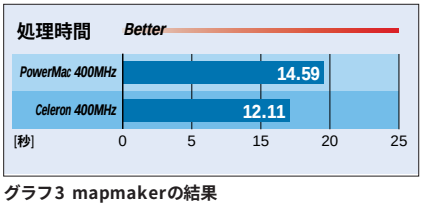
2Dの描画速度については、ほぼ飽和しており、ビデオカード間の性能差とは考えにくい。やはり開発者の絶対数がPC用Linuxよりも少ないため、ドライバのチューニングが進んでいないためと考えられる。今後はビデオカードベンダーであるATIの情報公開が進むことで、より高性能なドライバの登場が待ち望まれるところだ。

グラフ2には、nbenchの結果を示した。PowerPCは、浮動小数演算に強いと評判なので、この結果は意外だった。コンパイラのバージョンは同じもの（egcs-2.91.66）を用いているので、最適化のレベルに差が出ているとは考えにくいが、機種依存の最適化で、差がついているのであろうか。

最後は、mapmakerの結果だ（**グラフ3**）。Celeronマシンのほうが20%ほど速いという結果に終わった。実際の使用時には、気にならない程度の差だ。



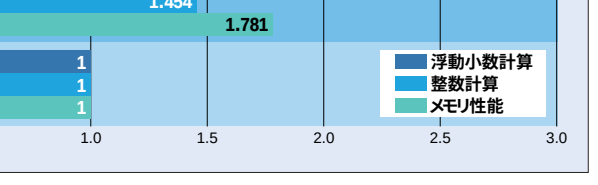
グラフ1 x11perf／Xmarkの結果



グラフ3 mapmakerの結果

ベンチマークの利用法

今回の比較では、PowerMacintoshはnbench以外はあまりふるわなかった。だからといって、LinuxPPCに失望してしまうのは、早計というものだ。実際の体感速度に近い値が出るように工夫されているとはいっても、ベンチマークというのは、かなり特殊な使い方なのだ。実作業での所要時間は、マシン性能以外にも、いろいろなことに影響されるものだ。ベンチマークのわずかな差に一喜一憂するのではなく、参考として心にとどめておくのがいいだろう。



グラフ2 nbenchの結果

MkLinuxの現状は？

'98年にMkLinux DR3（Developer Relase）が発表されてまもなく、アップル社はMkLinuxの開発チームを解散した。以来、MkLinuxはコミュニティ主導での開発が続けられており、まもなくMkLinux R1がリリースされると発表されている（**画面17**）。

MkLinuxの開発が続けられている理由は、NuBus PowerMacintoshへの対応以外に、研究のためという意味合いが強い。OSを研究対象にするとき、わざわざモノリシックカーネルを選択する研究者はいないだろう。Machのようなマイクロカーネル構造のOSを選択するはずだ。MkLinuxはPowerMacintoshだけでなく、x86 PCでも動作実績のあるマイクロカーネルのOSとして、とても有用なのだ。

MkLinuxは、広く普及することはなかったものの、現実に動作している研究対象としてこれからも使われ続けていくだろう。

Macintosh de LinuxPPC

LinuxPPCのこれから

Macintosh de LinuxPPC

内外の動き

昨年は、内外を問わずLinuxPPC関係やMacintoshに関するニュースを耳にする機会が多かった。LinuxPPCの存在感が大きくなってきたと感じたものだ。今年になってもその活発さは続いているようだ。

LinuxPPCのコミュニティは、iMac、PowerMacintosh G3といった機種に素早く対応してきた。昨年後半以降、アップル社はiBook、PowerMacintosh G4（G4）、iMacDVと矢継ぎ早に新機種を投入している。これら新機種への対応も積極的に行われており、この記事を作成している時点ですでに、iBook、G4、iMacDVに対応した開発版のカーネルが発表されている（**画面18**）。

今回のベンチマーク結果で明らかになった描画速度の問題も、ビデオドライバのバージョンアップが進むにつれて、改善されていくだろう。アップル社の現行機種には、カナダATi社のビデオカードが採用されているが、同社によるオープンソースコミュニティへの技術情報の公開も行われているようなので、ハードウェアの性能をフルに活かすことが可能になると思われる。

また日本国内では、LinuxPPC-JPというプロジェクトが動き始めている（**画面19**）。LinuxPPC-JPは、コミュニティが中心になって、標準的なディストリビューションの制作を目標にしている。現状では、非互換の複数のディストリビューションが並立しており、ただでさえ規模の小さいLinuxPPCコミュニティの力を効率良く使えているとは言えない状況だ。

そこでコミュニティ中心の「標準ディストリビューション」を作り出すことで、開発者、ユーザー、ディストリビュータ全員が幸せになれる状況を目指す運動が生まれたということだ。このような体制は、オープンソースのコミュニティではむしろ普通の状態であり、あるべき姿に戻そうという動きであるともいえよう。今までの各社の成果をなるべく無駄にすることなく、「標準化」が達成されることを願っている。

大手ディストリビューションの参入

現在のところ、メジャーなディストリビューションで、PowerPC版のLinuxを用意しているのは、Debian（**画面20**）だ。ほかにもTurboLinux社が、昨年のLinuxWorld Expoなどで試作版を展示していたが、製品化はまだのようだ。

ところが今年はじめに、ドイツのSuSE社（**画面21**）がMacintosh用のLinuxディストリビューション「SuSE Linux 6.3 for PowerPC」のベータ版を、MacWorldで配布した。これは、同社のx86用SuSE LinuxをPowrePCに移植したものだ。ヨーロッパで非常に人気のあるSuSEがPowerPC版を用意したということは、MacintoshでLinuxという組み合わせが、商売として成り立つと判断したということではないだろうか。

Macintoshユーザーは、ある意味もっともLinuxから遠いところに位置すると思われていたが、LinuxPPCのおかげで、両者の距離が縮まっていくことで、Linuxユーザーの層が一段と広がるだろう。ペンギンとリンゴのホットな関係は、ここしばらく要注目だ。

画面18 Linux/PPCプロジェクト
PowerMacintosh G4、iMacDVへの対応を報じている。

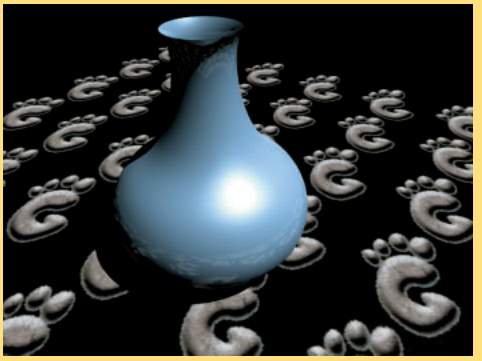
画面19 LinuxPPC-JPプロジェクト
今年になって、開発ツリーが公開された。

画面20 Debian
オフィシャルホームページ

画面21 SuSE社
MacWorld ExpoでのPowerPC版発表を報じている。

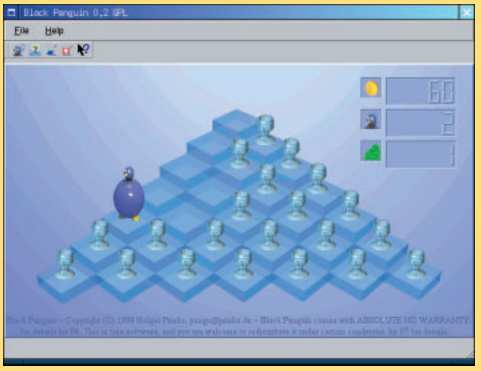
Free Application Showcase

文：出井 一
Text : Hajime Dei



Moonlight Atelier P.144

XROT P.148



Black Penguin P.149

※紹介したソフトは、すべて付録CD-ROMに収録されています。

統合デスクトップ環境を目指す日本語対応ファイルマネージャ

Cxplorer

バージョン：1.2.1　　種別：GPL

http://www.threeweb.ad.jp/~hatakeda/cxplorer/index.htm

Cxplorerは、日本語ファイル名に完全対応した国産のファイルマネージャだ。メニューやダイアログは最初から日本語で表示される。将来的にはGNOMEやKDEのような統合デスクトップ環境の構築を目指しており、現時点では簡易テキストエディタやアドレス帳、カレンダーソフトなどが付属する（ウィンドウマネージャのアルファ版も含まれる）。動作にはGTK＋1.2.1以降が必要だ。

●インストールとスタイル変更

Cxplorerは、ファイル一式をtar＋gzipしたtarボールのほか、rpmでも配布されている。ただし、現時点では最新版（1.2.1）のrpmがまだ用意されていないので、tarボールから作成することになるだろう。

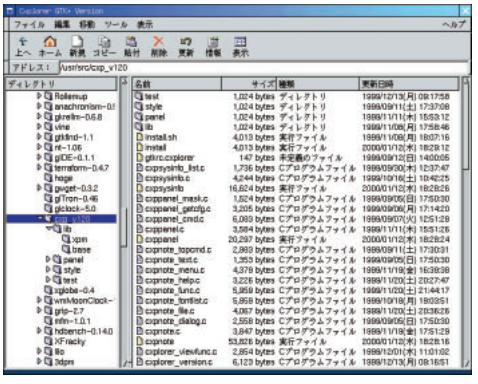
ビルドするには、tarボールの展開先で「make」とすればいい。ビルドに成功したら、スーパーユーザー（root）になった後で、インストール用のスクリプト「./install.sh」を実行しよう。実行に必要なファイル一式が/usr/local/cxplorerにコピーされ、/usr/local/binに実行ファイルのシンボリックリンクが作成される。

コマンドラインから「cxplorer」として起動すると、ディレクトリツリーとファイル一覧の2ペイン構成のウィンドウが表示される（**画面1**）。メニューやツールバー、ファイルの説明などは最初から日本語表示だ。これらが正常に表示されない場合は、.gtkrcのフォント設定や環境変数LANGの設定を確認しよう。また、メニューやツールバーはフローティング（メインウィンドウから外して独立したウィンドウにすること）が可能だ。

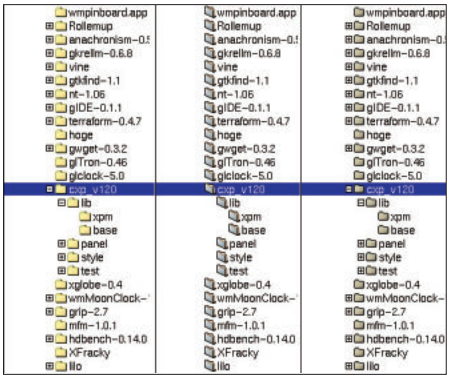
なお、ディレクトリツリーやファイル一覧で使われるフォルダアイコンは、初期設定ではMacOS風のデザインだが、Windows風、KDE風、GNOME風に変更することもできる（**画面2**）。[ツール]－[スタイル変更]を選択してダイアログを開き、好みのスタイルを選択すればいい（**画面3**）。

●基本的な使い方

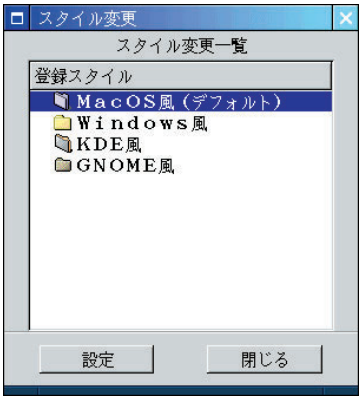
ファイルマネージャとしての操作は、GMCやKFMと似たごく一般的なものだ（ドラッグ&ドロップには未対応）。最初に、処理対象となるファイルを選択する。Shift－クリックやドラッグによる範囲選択、Ctrl－クリックによる複数選択も可能だ（**画面4**）。続いて、



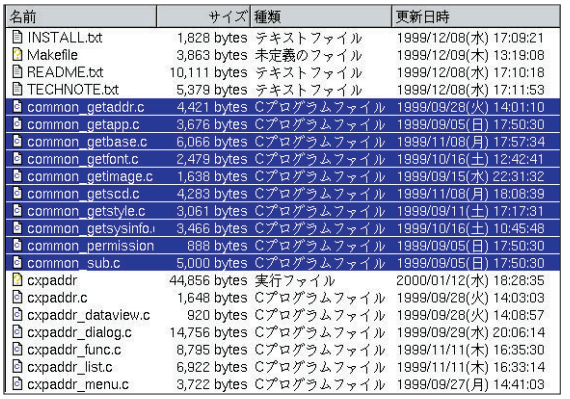
画面1
ディレクトリツリーとファイル一覧で構成されるウィンドウ。



画面2
左からWindows風、KDE風、GNOME風の表示スタイル。

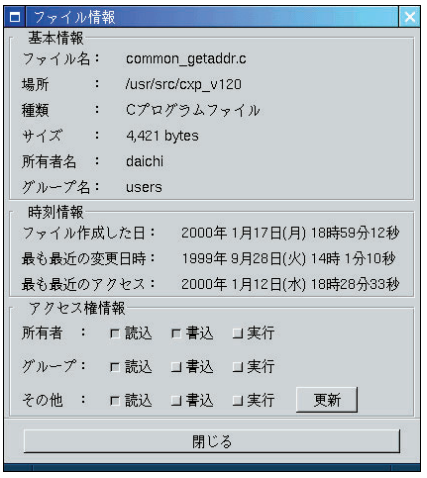


画面3
スタイルの変更はこのダイアログで行う。



名前	サイズ	種類	更新日時
INSTALL.txt	1,828 bytes	テキストファイル	1999/12/08(水) 17:09:21
Makefile	3,863 bytes	未定義のファイル	1999/12/09(木) 13:19:08
README.txt	10,111 bytes	テキストファイル	1999/12/08(水) 17:10:18
TECHNOTE.txt	5,379 bytes	テキストファイル	1999/12/08(水) 17:11:53
common_getaddr.c	4,421 bytes	Cプログラムファイル	1999/09/28(火) 14:01:10
common_getapp.c	3,676 bytes	Cプログラムファイル	1999/09/05(日) 17:50:30
common_getbase.c	6,066 bytes	Cプログラムファイル	1999/11/08(月) 17:57:34
common_getfont.c	2,479 bytes	Cプログラムファイル	1999/10/16(土) 12:42:41
common_getimage.c	1,638 bytes	Cプログラムファイル	1999/09/15(水) 22:31:32
common_getscd.c	4,283 bytes	Cプログラムファイル	1999/11/08(月) 18:08:39
common_getstyle.c	3,061 bytes	Cプログラムファイル	1999/09/11(土) 17:17:31
common_getsysinfo.c	3,466 bytes	Cプログラムファイル	1999/10/16(土) 10:45:48
common_permission	888 bytes	Cプログラムファイル	1999/09/05(日) 17:50:30
common_sub.c	5,000 bytes	Cプログラムファイル	1999/09/05(日) 17:50:30
cxpaddr	44,856 bytes	実行ファイル	2000/01/12(水) 18:28:35
cxpaddr.c	1,648 bytes	Cプログラムファイル	1999/09/28(火) 14:03:03
cxpaddr_dataview.c	920 bytes	Cプログラムファイル	1999/09/28(火) 14:08:57
cxpaddr_dialog.c	14,756 bytes	Cプログラムファイル	1999/09/29(水) 20:06:14
cxpaddr_func.c	8,795 bytes	Cプログラムファイル	1999/11/11(木) 16:35:30
cxpaddr_list.c	6,922 bytes	Cプログラムファイル	1999/11/11(木) 16:33:14
cxpaddr_menu.c	3,722 bytes	Cプログラムファイル	1999/09/27(月) 14:41:03

画面4
ドラッグやShift／Ctrlキーにより複数ファイルを選択可能だ。



画面5
ファイルに関する詳しい情報を表示。属性の変更も行える。

ツールバーのボタン、メニューバーや右クリックメニューの各項目、ショートカットキー（Ctrl＋英字キー）によって処理内容を指定する。

たとえば、ファイルを削除する場合は、削除したいファイルを選択後、ツールバーの[削除]ボタンやCtrl－Dキーを押すか、メニューバーの[編集]－[削除]や右クリックメニューの[削除]を選択する。続いて、確認用のダイアログの[削除]ボタンを押すと、実際に削除が行われる。

また、ファイルのコピーはコピー＆ペーストによって行う。まず、コピーしたいファイルを選択し、ツールバーの[コピー]ボタンを押す。続いて、コピー先のディレクトリに移動し、ツールバーの[貼付]ボタンを押すと、先ほどコピーしたファイルがそのディレクトリに作られる。なお、コピー先に同名のファイルが存在する場合、後からコピーするファイルの名前が自動的に変更される。

このほか、ツールバーの[情報]ボタンで開くダイアログでは、選択したファイルに関する詳しい情報が表示される（**画面5**）。所有者やグループ名、3種類のタイムスタンプを確認できるほか、ファイル属性の設定を（権限のある場合には）このダイアログで変更することも可能だ。

●アプリの関連付けと簡易エディタ

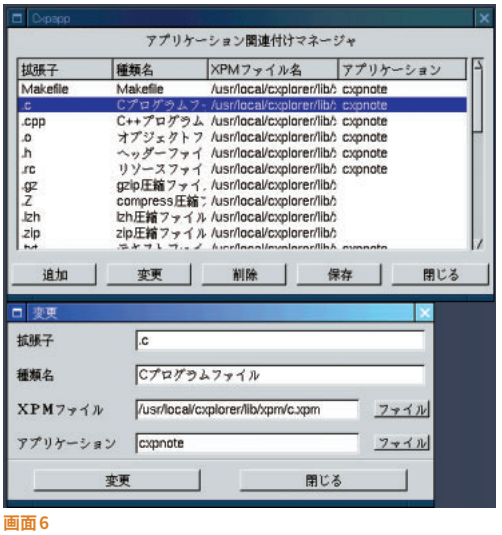
ツールバーの[ツール]メニューからは、Cxplorerと一緒に配布されている付属ソフトを呼び出せる。なお、これらは単独でも実行可能なので、以下の説明ではソフト名の後にコマンド名を併記することにする。

Cxplorerに関する設定変更も付属ソフトで行う。[ツール]－[アプリケーション関連付け]で呼び出されるCxpapp（コマンド名cxpapp）では、ファイル一覧中のファイルをダブルクリックしたときに起動するアプリを設定する（**画面6**）。

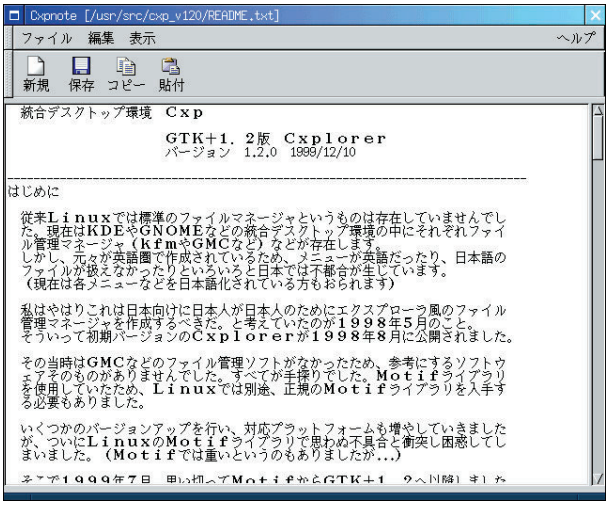
具体的には、ファイルの拡張子ごとに、一覧中に表示される種類名とアイコン画像、起動するアプリをそれぞれ設定すればいい。初期設定では、テキストファイルなどに対してCxpnote、何種類かの画像ファイルに対してxvが起動するようになっている。好みに合わせてアプリを変更したり、新たな拡張子の関連付けを追加するとよいだろう。

Cxpnote（コマンド名cxpnote）は、日本語の入力、編集が可能な簡易テキストエディタだ（**画面7**）。ファイルの読み込み・保存のほか、コピー＆ペーストなどの基本的な機能が用意され、設定ファイルなどの内容確認や書き換えには十分使えるものになっている。表示に使われる日本語フォントは固定ピッチだが、3種類のサイズ（14／16／24ドット）を切り替え可能だ。

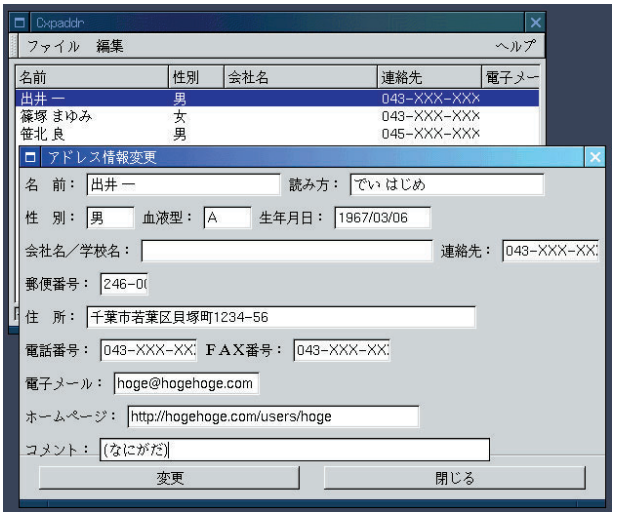
なお、キーバインドはGTK＋標準のもので、WordStar系のカット／コピー／ペースト（Ctrl－X／C／Vキー）



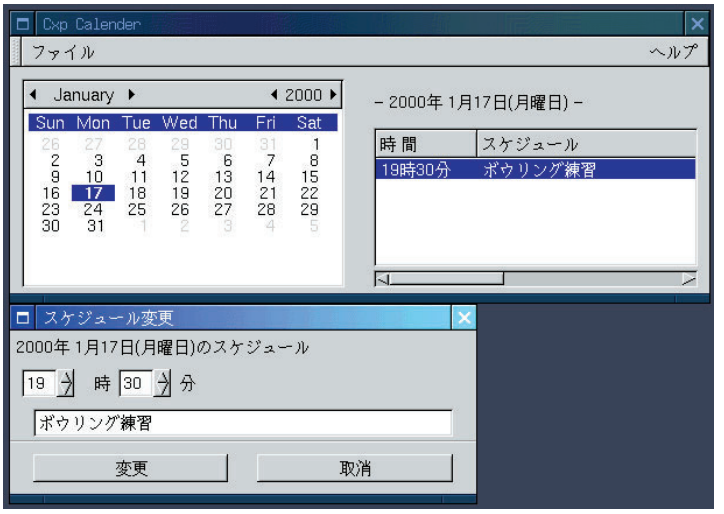
画面6
ダブルクリックで起動するアプリを設定する「Cxpapp」。



画面7
付属の簡易テキストエディタ「Cxpnote」。



画面8
メールアドレスやURLも含めたアドレス管理を行う「Cxpaddr」。



画面9
スケジュールの入力も可能なカレンダー「Cxp Calendar」。

とEmacs系のカーソル移動（Ctrl－P／N／B／Fキーなど）を合わせたものになっているため、少しとまどうかもしれない。

●充実しつつある付属ソフト

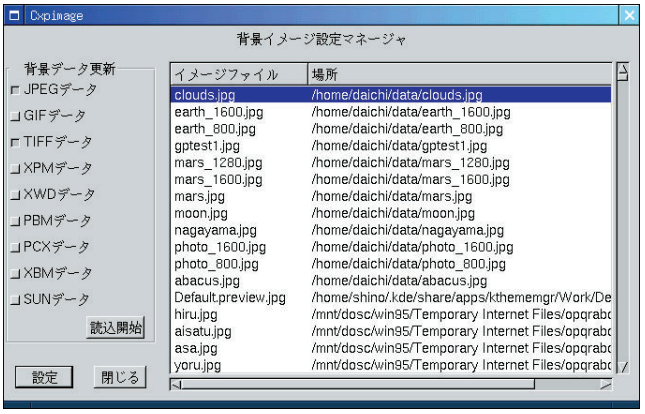
このほか、アドレス管理やカレンダー、背景イメージ変更、システム情報表示といったソフトが付属する。順番に使い方を紹介しよう。

まず、[ツール]－[アドレス管理]で呼び出されるアドレス管理用のCxpaddr（コマンド名cxpaddr）では、名前と電話番号、住所のほか、メールアドレスやWebページのURLなども設定可能だ（**画面8**）。検索機能を持たないため、小規模データの管理に向いている。

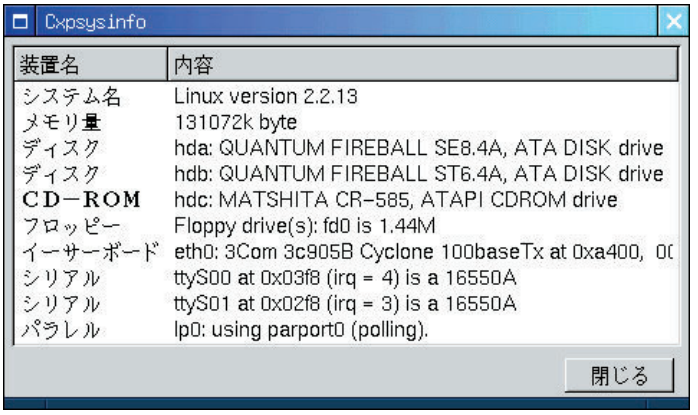
続いて、[ツール]－[カレンダー]で呼び出されるCxp Calender（コマンド名cxpcalen）では、時刻付きのスケジュールを設定できる（**画面9**）。ただし、設定した日付が太字で表示されるだけで、指定時刻になったことを知らせるアラーム機能が用意されていないため、現状ではスケジューラとして活用するのは難しいだろう。

また、[ツール]－[背景イメージ変更]で呼び出されるCxpimage（コマンド名cxpimage）は、選択した画像をXのルートウィンドウ（背景）に表示する（**画面10**）。ファイルシステム上に存在する画像ファイル（画像形式は選択可能）を検索して一覧を作成するため、最初の1回は時間がかかるものの、面倒なディレクトリパスの入力が必要ないので便利だ。なお、ルートウィンドウへの画像表示にはxloadimageを利用するため、別途xloadimageをインストールしておく必要がある。

最後に、[ツール]－[システム情報表示]で呼び出されるCxpsysinfo（コマンド名cxpsysinfo）は、システム名やメモリ量、ディスク構成、ネットワークカード、シリアルポートなどの情報を表示する（**画面11**）。



画面10
画像の一覧から背景表示するものを選ぶ「Cxpimage」。



画面11
ディスク構成などのシステム情報を表示する「Cxpsysinfo」。

30種類以上の画像形式に対応した画像ソフト

CompuPic

バージョン：4.6／5.0（ベータ版）種別：シェアウェア（39.95USドル、非商業利用はフリー）

http://linux.compupic.com/

CompuPicはWindows用ソフトとして実績のある洗練された画像管理ソフトのLinux版だ。JPEG／GIF／TIFFなど30種類以上の画像形式に対応しており、サムネイル表示などにより大量の画像を効率よく管理できるほか、スライドショウやファイル管理、画像の加工といった関連機能も充実している。現在、公式版の4.6とベータ版の5.0が公開されており、以下では5.0に基づいて機能を紹介する。

●インストールとライセンス

CompuPicのソースは公開されておらず、バイナリ配布のみ行われている。tarボールのほか、rpmなどの各種パッケージも用意されているので、自分の環境に合ったものを選ぼう。tarボールの場合は、展開先に作成されるインストーラを使って「./compupic-install」とすることで、/usr/local/compupicや/usr/local/binへのインストールが行われる。

なお、CompuPicをビジネス用途で利用する場合は、レジスト料を支払う必要がある。非商業利用の場合はフリーだが、LZW圧縮に関するライセンスの関係で、30日の試用期間を過ぎるとGIF／TIFF形式を扱えなくなる（詳細はhttp://linux.compupic.com/linux_terms.htmlを参照）。

●起動と画面表示

「compupic」として起動すると、ディレクトリツリーとファイル一覧で構成されるメインウィンドウ（**画面1**）と、使い方のヒントなどが書かれた「Tips of the Day」ウィンドウが開く。このほか、初めて起動した場合は、ライセンス認証用のダイアログも表示される。

なお、5.0はベータ版のため、動作中にさまざまな情報をコンソール（CompuPicを起動した端末）に出力する。これを見えないようにするには、起動時のコマンドラインを「compupic > /dev/null」として、出力を/dev/nullにリダイレクトすればいい。

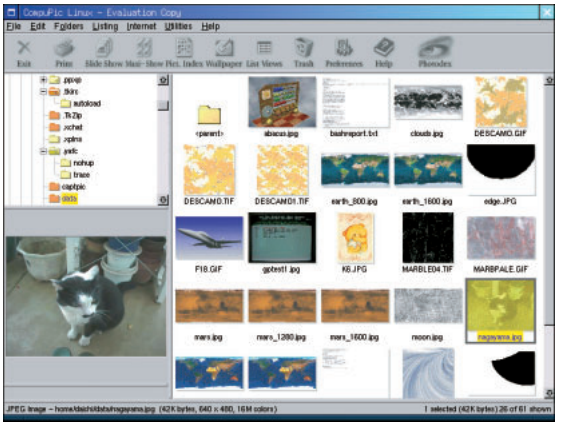
ファイル一覧の表示には、ファイル名と共にサムネイル（縮小画像）が使われる。画像ファイルの場合、最初にそのファイルを表示する際に自動的にサムネイルが作成され、以後はそれを再利用する。ディレクトリや通常のファイルは固定イメージだ。このため、大量の画像が格納されたディレクトリでも、一度訪れた後ならきわめて高速な表示が可能だ。

●基本的な操作と表示の切り替え

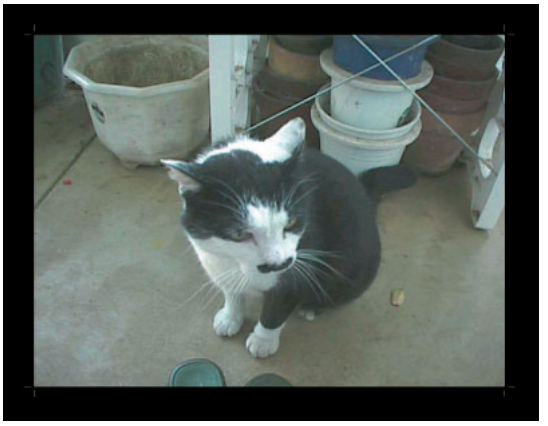
サムネイルをクリックしてファイルを選択すると、ディレクトリツリー表示の下のエリアに、少し大きめのイメージが表示される。ダブルクリックする（あるいはEnterキーを押す）と、実際の画像がフルスクリーンで表示される（**画面2**）。カーソルキーで表示位置を変更可能だ。

このほか、Shift－クリックによる範囲選択や、Ctrl－クリックによる複数選択にも対応している。選択したファイルは、コピー／削除などのファイル処理、スライドショウや各種の画像加工などの対象になる。ドラッグ＆ドロップにより、他のディレクトリへ直接移動することも可能だ。

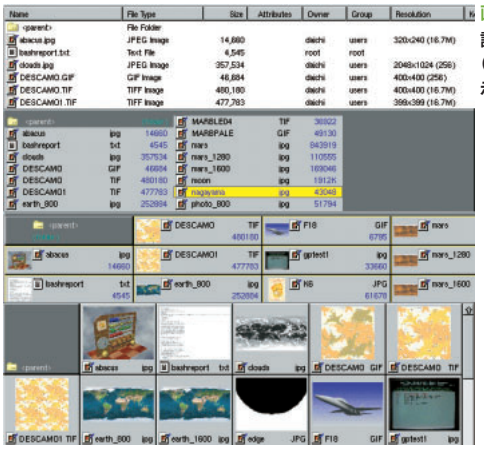
なお、ファイル一覧の表示形式は、初期設定のサムネイルのほかに、詳細、一覧、ボタン状のサムネイル（大／小）の4種類が用意されており（**画面3**）、ツールバーの[List Views]ボタンや、メニューバーの[Listing]以下の項目、右クリックメニューの[List Views]以下の項目を選択することで切り替えら



画面1
フォルダツリーとファイル一覧表示で構成されるウィンドウ。



画面2
実際の画像はフルスクリーン表示される。表示位置の変更も可能。



画面3
詳細、一覧、ボタン風（大／小）など多様な表示形式をサポート。

画面4
複数の画像をフルスクリーンに並べて表示する「マキシショウ」。

れる。

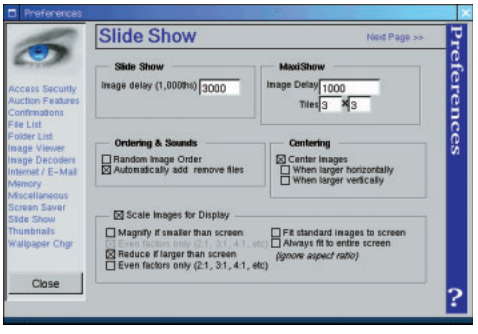
●スライドショウなど機能は豊富

画像を連続表示する「スライドショウ」を行うには、対象となる画像を選択し、ツールバーの[Slide Show]ボタンを押せばいい。ダブルクリックした場合と同様にフルスクリーンで画像が表示され、一定時間（初期設定では3秒）経過すると、自動的に次の画像に切り替わる。カーソルキーやスペースキーで前後の画像に切り替えることも可能だ。

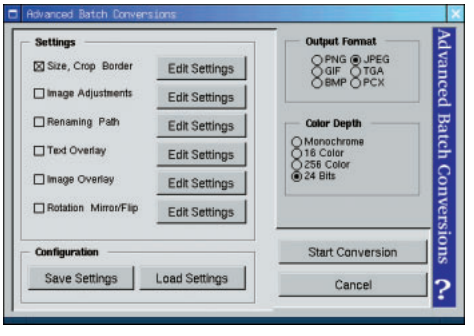
ちょっと変わった機能として、複数画像を同時に表示する「マキシショウ」がある。これは、フルスクリーンを3×3のエリアに分割し、左上から順番に画像を表示していくもの（**画面4**）。連続写真など、シリーズものの画像を眺めるのに便利だ。なお、時間間隔や画面の分割数などの設定は、ツールバーの[Preferences]ボタンで設定ダイアログを開き、[Slide Show]ページで変更できる（**画面5**）。

また、大量の画像を管理している場合は「Advanced Batch Conversions」を活用したい。画像サイズや格納パスの変更、テキストや画像のオーバーレイといった作業内容をダイアログで設定しておけば（**画面6**）、選択したファイルに対して一気にそれらの作業を行うことができる。

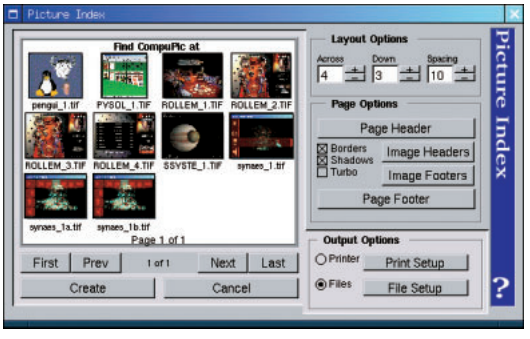
このほかにも、複数のサムネイルをまとめてインデックス画像を作成したり（**画面7**）、電子メールで送るためのグリーティングカードを作成する（**画面8**）など、ユニークな機能が豊富に用意されている。



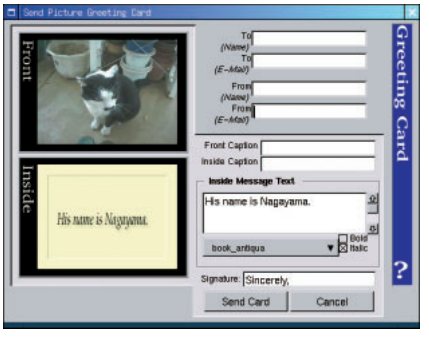
画面5
設定の変更は機能別にページ分けされたダイアログで行う。



画面6
複数画像の形式、サイズなどを一括変更するバッチ処理にも対応。



画面7
インデックス画像を効率よく作成する機能も用意されている。



画面8
画像にメッセージをつけたグリーティングカードも作成可能だ。

関数や散布図などの2次元グラフをGUIで表示する

GtkGraph

バージョン：0.6.1　　種別：GPL

http://gtkgraph.linuxbox.com/

GtkGraphは、科学技術分野で使われる2次元関数や散布図などのグラフを描画するソフトだ。定番のGnuplotと違って、散布図のデータ入力やグラフのパラメータなどをGUIで設定できるため、比較的簡単にグラフを作成できるのが特徴だ。また、日本語リソースが最初から用意されており、日本語環境ではメニューやダイアログが日本語で表示される。動作にはGTK＋が必要だ。

●ビルドとインストール

GtkGraphは、tarボールのほか、rpm／debのバイナリパッケージも配布されている。なお、バイナリパッケージはglibc2.1用に作られているので、Red Hat 6.x系以外のディストリビューション（Red Hat 5.xやVine 1.1など）では、tarボールを利用するか、付属のspecファイルに基づいてパッケージを自分で作成する必要がある。

tarボールからのビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順をとる。日本語リソースはオリジナルのtarボールにも含まれているため、日本語化のために追加作業は必要ない。

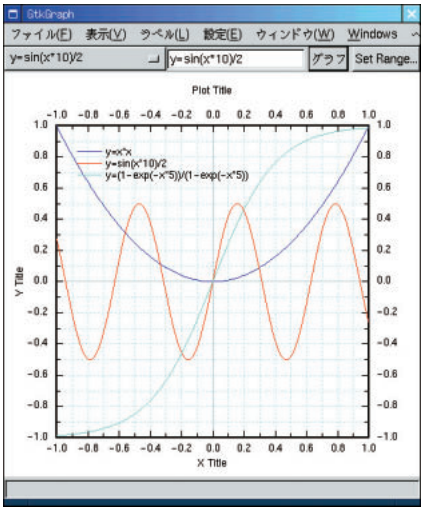
●関数のグラフを描画する

コマンドラインで「gtkgraph」と入力して起動すると、方眼紙状のキャンバスを含むウィンドウが開く。GtkGraphでは、複数のグラフを10個まで同一のキャンバスに描画できる（**画面1**）。X／Y軸の名称を設定したり、表示範囲を変更することも可能だ。

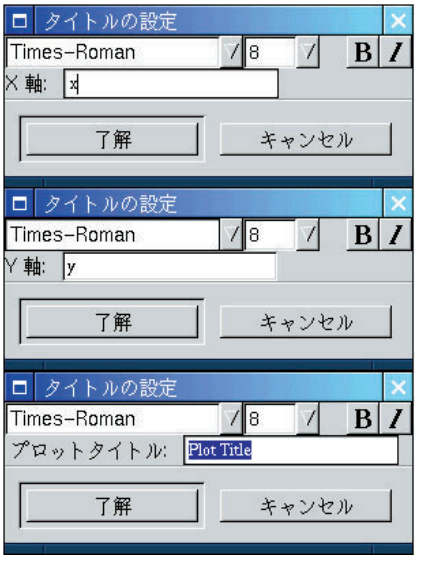
関数のグラフを描画するには、メニューバー下のエディットボックスに関数の数式を入力すればいい。数式中ではX軸の変数xや四則演算、カッコ、exp／sin／log／sqrtといった各種の関数を利用できる。たとえば、「y＝x＊x」と入力すると、2次関数のグラフが描画される。

続いて、「Untitled Funtion」（名称未設定）と表示されたボタンを押し、メニューから[Change function name]を選択して関数名を入力する。ここで入力した名称はボタン上に表示されるほか、凡例としてキャンバス上にも表示される。

新たなグラフを追加するには、同じボタンを押して、メニューから[New function]を選択し、数式の入力と名称の設定を行えばいい。複数のグラフの切り替えもこのボタンのメニューで行える。



画面1
GtkGraphでは複数のグラフを同一キャンバスに表示できる。



画面2
X／Y軸の名称やプロットタイトルも変更可能だ（日本語不可）。

●ラベルの設定と表示範囲の変更

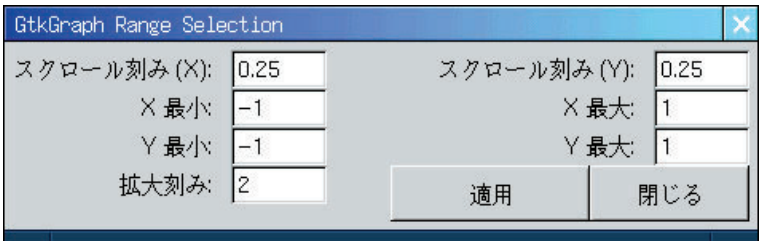
ツールバーの[ラベル]以下のメニューにより、X／Y軸の名称やプロットタイトルを設定できる（**画面2**）。なお、関数名やX／Y軸の名称、プロットタイトルに日本語を使うと、キャンバス上の表示が文字化けしてしまうので注意されたい。これはキャンバス上の表示に使われるフォントが英字フォントに限定されているためだ。

キャンバスの表示範囲を変更するには、[表示]メニュー以下の項目を選択するか、対応するショートカットキーを利用する。拡大・縮小と上下左右へのスクロールが可能だ。また、キャンバス上で拡大する範囲をドラッグ指定することもできる。

表示範囲をより正確に数値で設定するには、[Set Range]ボタンを押してダイアログを開く（**画面3**）。ここでは、X／Y軸の最大値と最小値のほか、スクロール時の移動量や拡大／縮小時の倍率も設定可能だ。

●散布図データの入力と表示

実験データで得られるような複数の

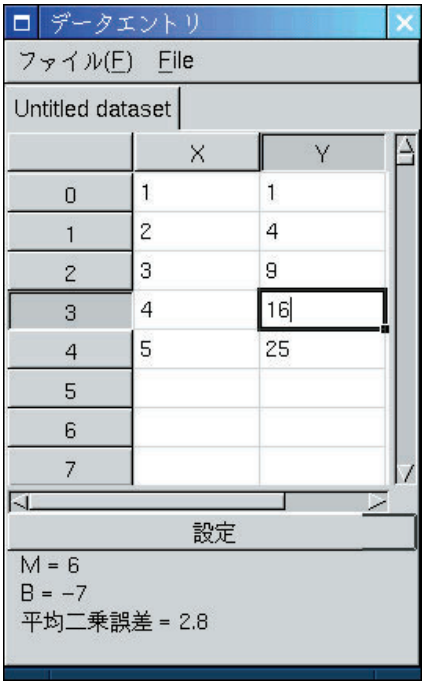


画面3
表示範囲などを数値で設定するためのダイアログ。

座標値を散布図としてキャンバスに表示するには、[Windows]－[Show Data]を選択してデータエントリ用のウィンドウを開き、「データセット」として入力する必要がある（**画面4**）。関数の場合と同様に、データセットに名前をつけたり、複数のデータセットの散布図を同時にキャンバスに表示することも可能だ。

表計算ソフトのようなグリッドに(x,y)の座標値を入力すると、キャンバス上の対応する座標に◇などのアイコンが現れる。キャンバスの表示範囲は、入力したデータに応じて自動的に調整される。また、複数の値を入力すると、線形回帰によりアイコン間に直線が描かれる（**画面5**）。また、[設定]ボタンで開くダイアログでは、アイコンや回帰の種類を変更したり、アイコン間を直線やスプライン曲線で結ぶなどの設定が可能だ（**画面6**）。

なお、ここで入力したデータは、[File]－[Save dataset]で保存できる。データ形式はX／Y座標値がタブ区切りで、1行に1組ずつ書かれたテキストだ。逆に、実験データをこの形式で作成すれば、[File]－[Load dataset]で直接読み込める。



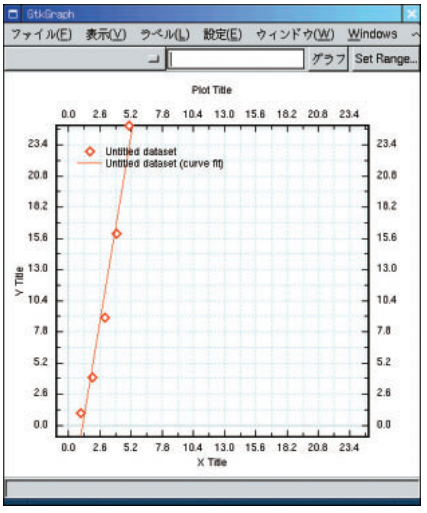
画面4
散布図を描くための座標値をデータセットとして入力する。

●グラフの印刷と保存

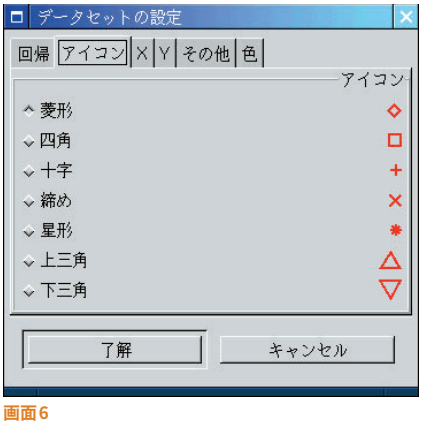
作成したグラフは、PostScript形式で直接印刷、保存できる。また、PNG／JPEG／XPM形式などのビットマップ画像を作成することも可能だ。

PostScript形式で保存する場合は、[ファイル]－[グラフをPostScriptへ出力]を選択し、保存するファイル名を指定する。出力されたファイルはgvなどによりプレビュー、印刷が可能だ（**画面7**）。

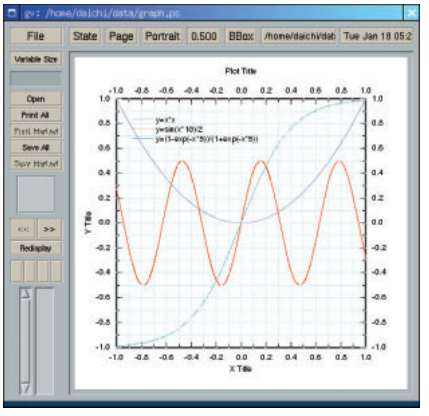
一方、ビットマップ画像を作成する場合は、[ファイル]－[グラフをイメージへ出力]を選択し、保存するファイル名を指定する（画像形式は拡張子により自動的に判断される）。なお、作成される画像の大きさは、画面に表示されるキャンバスと同じ500×500ドットで固定だ。



画面5
データセットの座標値と線形回帰による直線が描かれる。



画面6
アイコンの形状や、回帰の種類などはこのダイアログで設定する。



画面7
PostScript形式で出力したグラフをgvでプレビュー。

本格的な3Dモデリング・レンダリングソフト

Moonlight Atelier

バージョン：0.9.1（ベータ版） 種別：フリー

http://www.moonlight3D.org/

Moonlight Atelier（以下Moonlightと表記）は、本格的な3DCGを作成できるフリーソフトだ。NURBS（不均一有理Bスプライン）による自由曲線・曲面でオブジェクトをモデリングでき、レイトレーシングに加えて、ラジオシティやBRDF（双方向反射分布関数）などの高度なレンダリングをサポートしているなど、高価な市販ソフトにひけをとらない内容だ（アニメーションは将来対応する予定）。

●専用のインストーラが付属

Moonlightのソースは現時点では配布されておらず（いずれは公開される予定）、現在はtarボールによるバイナリ配布のみ行われている。このため、カーネル2.2とglibc2.1を採用したディストリビューション（Red Hat 6.xやLASER5など）でのみ動作する。また、インストーラがGTK＋を利用するため、GTK＋1.2も必要だ。

tarボールを展開すると、ファイル一式を内蔵したインストーラが展開される。あとは、「./moonlight-setup-0.9.1-beta」としてこれを実行し、質問に答えていけばいい（**画面1**）。初期設定のままだと、実行ファイルやサンプルデータなどすべてのファイルが/usr/local/moonlight以下にインストールされる。

インストール後、実行ファイル（moonlight）のシンボリックリンクを/usr/local/binに作成しておくと便利だ。「ln -s /usr/local/moonlight/bin/moonlight /usr/local/bin/moonlight」とすればいい。

なお、3Dオブジェクトの表示にはOpenGL互換ライブラリのMesaが必要だ。もっとも、tarボールにはMoonlight専用のMesaライブラリが含まれており、通常はこれが利用されるため（変更可能）、Mesaを別途インストールする必要はない。

●機能的デザインのウィンドウ

「moonlight」として起動すると、美しいタイトルグラフィックが表示され（**画面2**）、キーかマウスボタンを押すとフルスクリーンのウィンドウが開く（**画面3**）。GNOME／KDEのパネルなど重なって使いづらいなら、「moonlight -medium」として起動すれば、より小さなサイズのウィンドウが開く。ウィンドウサイズは変更できない。

ウィンドウ内は、左からメニューボタン、四面図（上面図・正面図・左面図・パース図）、ダイアログ表示部の3つの領域に分かれている。なお、四面図やダイアログは、ウィンドウマネージャではなく、Moonlight自身がウィンドウ風に表示しているもの。このため、タイトルをドラッグしても位置は変わらず、互いに重ならないようにタイル表示される。

●サンプルデータで基本操作を学ぶ

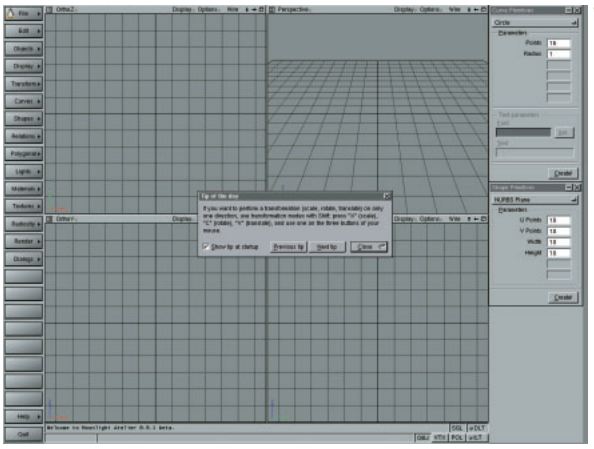
まずは、付属のサンプルデータを読み込んで、基本的な操作方法を説明する。メニューボタンの[File]－[Load]でダイアログ（**画面4**）を開き、/usr/local/moonlight/samplesにあるknot.mlkをロードしよう。

すると、中空のチューブで作られた結び目が表示される。左側は上面図と正面図のワイヤーフレーム、右側はパース図のプレビューとワイヤーフレームだ（**画面5**）。このように、四面図の表示内容は固定されたものではなく、各タイトルバーのメニューやボタンにより変更可能だ。

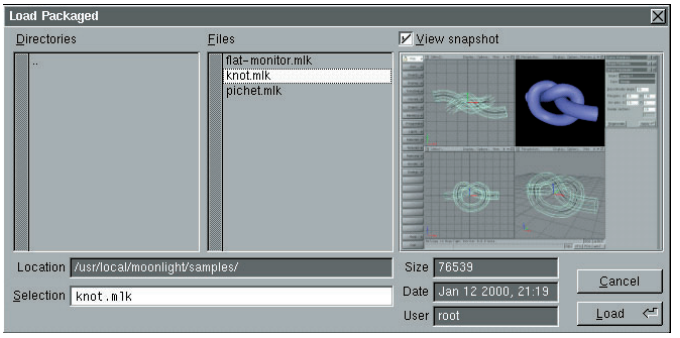
Moonlightでは、オブジェクトの選択や移動、拡大・縮小、制御点の位置修正、カメラの回転、移動、ズームといった操作を、特定のキー（モードキー）を押しながらマウスを操作（ドラッグ・クリック）することで行う。モードキーを押している間はカーソルが青に変化し、離すと赤に戻る。

画面1
インストールは専用のインストーラを利用する。

画面2
起動時に表示される美しいグラフィックタイトル。



画面3　ウィンドウ内には、擬似的なウィンドウがタイル状に並ぶ。



画面4　オープンダイアログでは、読み込むデータのプレビューが可能。

たとえば、オブジェクトを回転させるには、cキーを押しながら四面図上でマウスをドラッグすればいい。マウスの動きに応じて結び目が回転するのがわかるはずだ（背景のグリッドは動かない）。一方、カメラを回転させるoキーを押しながらドラッグすると、今度は結び目が背景のグリッドと一緒に回転する。つまり、グリッドの動きにより、オブジェクトとカメラの回転を区別できるわけだ。

なお、アンドゥ・リドゥはuキーがモードキーになっている。左ボタンクリックでアンドゥ、右ボタンクリックでリドゥだ。複数回の操作の取り消しが可能なので、失敗を恐れずにさまざまな操作を行える。

こうしたモードキーの一覧は、メニューボタンの[Dialogs]－[Modes list]で開くダイアログに表示される（**画面6**）。なお、このダイアログのラジオボタンをクリックしてモードを切り替えることも可能だ。モードの解除にはEscキーを使用する。

●オブジェクトを作成する

今度は、白紙の状態からオブジェクトを作成してみよう。サンプルデータをロードしたままの場合は、メニューボタンの[File]－[Clear all]で、起動直後の状態に戻る。

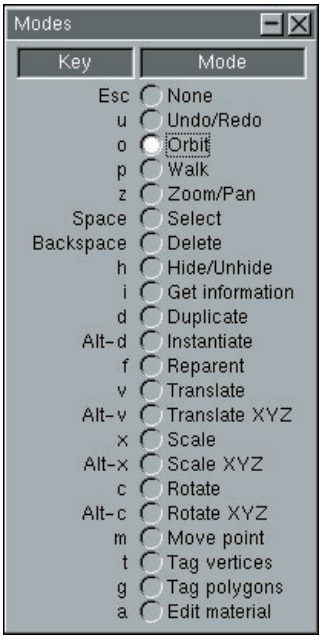
Moonlightでは、NURBSによる曲線・曲面を利用してオブジェクトを作成する。こうした曲線・曲面は自由度が高く、制御点を動かすだけである程度思い通りの形状に変形できる。円・矩形・平面・箱・球・トーラスといったプリミティブが用意されているほか、自由曲線から回転体やチューブを作成することも可能だ。

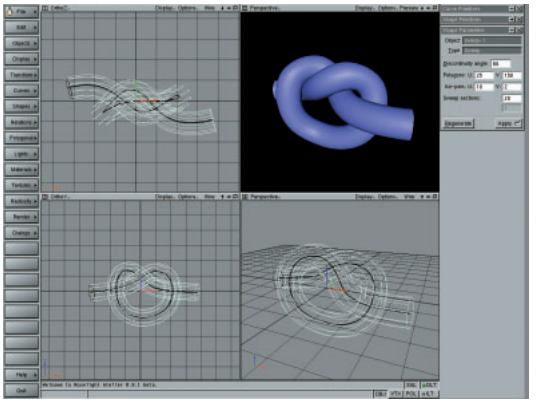
ここでは、壷のようなものを作成してみよう。まず、メニューボタンの[Curves]－[Draw NURBS curve]を選択して、曲線描画モードに入る（モードキーはない）。正面図に、壷の断面の曲線をマウスの左ボタンクリックで書いていこう。クリックした位置に制御点が置かれ、NURBS曲線が描かれる。最後の制御点は右ボタンクリックを利用する（**画面7**）。

続いて、制御点の位置を修正して望みの形に近づける。制御点の移動のモードキーはmキーだ。左ボタンのドラッグで制御点が移動する。[Edit]以下のメニューにより制御点の追加や削除も可能。さらに、制御点の重み付けといったNURBSのパラメータを変更するダイアログも用意されている。

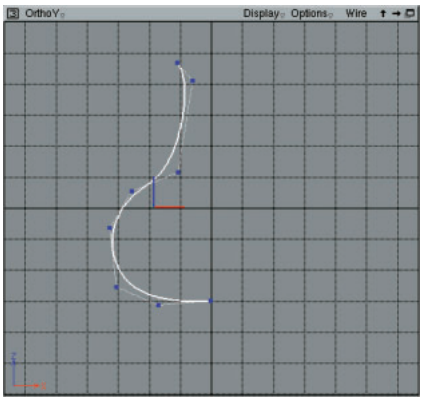
修正が終わったら、[Relations]－[Revolution]－[Revolution around Z]を選択して、曲線をZ軸に対して回転

画面6
モードキーの一覧ダイアログ。モード自体の切り替えも可能だ。

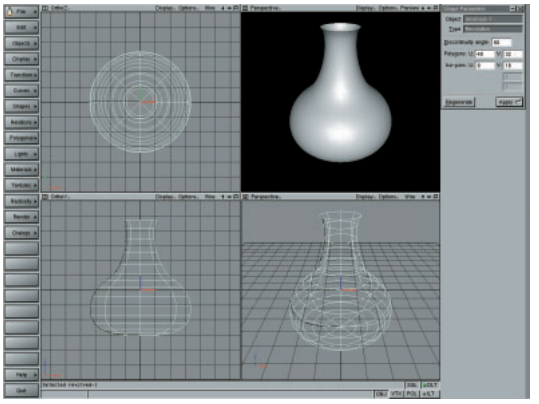




画面5
サンプルデータの「knot.mlk」はチューブで作られた結び目だ。

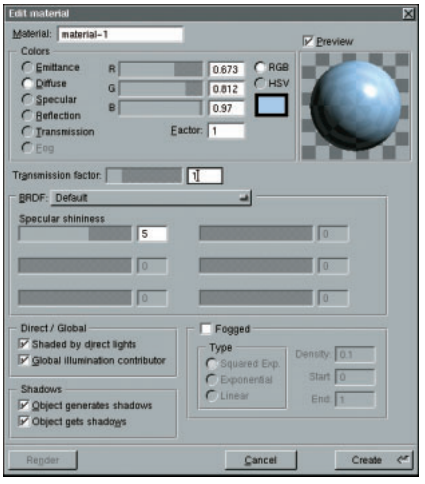


画面7
NURBS曲線を使って、正面図に壷の断面の曲線を描く。

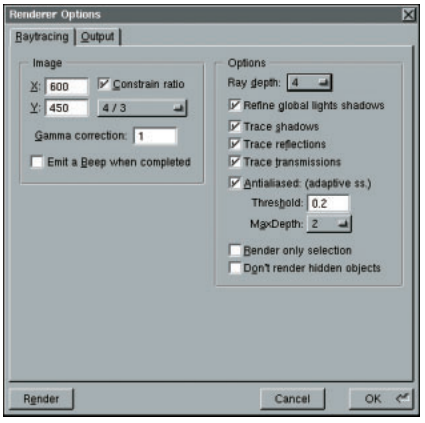


画面8
NURBS曲線をZ軸に対して回転させて作成した回転体。

させた回転体を作成しよう。また、壷の内側も表示されるように、[Shapes]－[Double Sided]をチェックする。これで、壷のようなものが表示されるはずだ（**画面8**）。さらに、曲面から生成されるポリゴンの数を増やしてなめらかさを調整したり、回転体をNURBS曲面に変換して、非対称な変形を行ったりすることも可能だ。



画面9
表面の色や材質を設定するマテリアルダイアログ。



画面10
画像サイズやレイトレーシングに関する設定を行う。

●マテリアルとテクスチャの設定

続いては、回転体のマテリアルとテクスチャを設定しよう。マテリアル編集のモードキーであるaキーを押して、編集したいオブジェクトをクリックし、aキーを離す。続いて、[New material]ボタンを押すと、マテリアル編集用のダイアログが開く。

ここでは、オブジェクトの表面の色や材質を、拡散光（Diffuse）や鏡面反射光（Specular）、反射率（Reflection）、透過率（Transmission）などのパラメータにより設定する。球体のプレビューで実際の効果を確認しながら設定可能だ（**画面9**）。

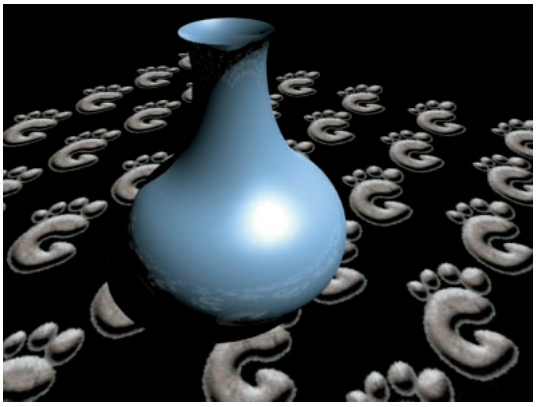
このほか、オブジェクトの表面に画像を貼り付けるテクスチャマッピングは[Textures]、オブジェクトを照らす光源は[Lights]以下のメニューでそれぞれ設定できる。

●レンダリングとデータの保存

最後に、レイトレーシングによるレンダリングを行う。まずは、[Render]－[Renderer options]で、生成する画像のサイズや反射を何回まで追跡するかといったパラメータを設定する（**画面10**）。続いて、[Render]－[Render]を選択すると、ウィンドウ内が一時的に黒く塗りつぶされ、レイトレーシングによるレンダリング画像が表示される（**画面11**）。

レンダリング画像は、JPEG／PNGなどの画像形式で保存できる。また、オブジェクトの3Dデータは、DXF／POV／Rayshade／VRML／OFF形式などでのエクスポート、DXF／3DS／OFF／PLYなどでのインポートにも対応している。

なお、Webサイトにサンプルデータの作成方法を解説したチュートリアルが用意されているので、ぜひとも参考にしてほしい。



画面11
レイトレーシングにより作成されたレンダリング画像。

ファイルシステムの宇宙空間を旅しよう

Xcruise

バージョン：0.22　　種別：GPL

http://tanaka-www.cs.titech.ac.jp/~euske/prog/index-j.html

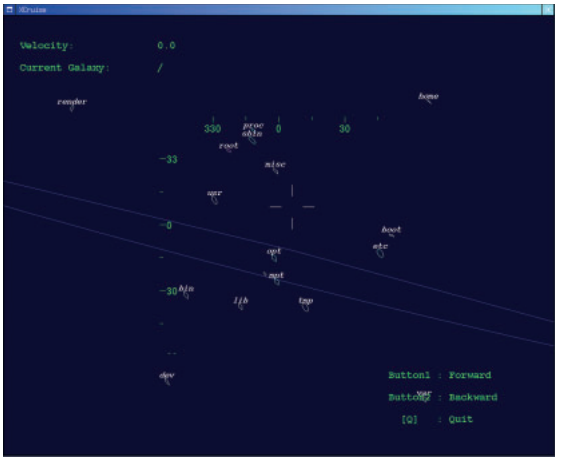
Xcruiseは、一風変わった方法でファイルの一覧表示を行うソフトだ。ディレクトリは銀河、その中のファイルは惑星といった具合に、ファイルシステムを宇宙空間に見たてて3次元に構成する。惑星の色や大きさはファイルの属性やサイズに対応しており、自由な航海を楽しみながら、ファイル構成に対する理解を深められる。なにより、単に宇宙をさすらっているだけで楽しいソフトだ。

●ビルドと基本操作

Xcruiseは、ファイル一式をtar＋gzipしたtarボールで配布されている。ビルドは、「xmkmf -a」→「make」という手順で行う。「make install」としてインストールすると、/usr/X11R6/binに実行ファイル（xcruise）がコピーされる。実行に必要なのはこのファイルだけなので、手動で/usr/local/binなどにコピーしても構わない。

「xcruise」として起動すると、ウィンドウが開いて宇宙空間が表示される。最初は、ファイルシステム全体を俯瞰する位置から始まるので、ルートディレクトリを示す大きな楕円と、いくつかのディレクトリの銀河の楕円が表示されるだけだ（**画面1**）。

宇宙空間の移動にはマウス操作を利用する。画面中央部の十字カーソルを上下左右に移動させることで進行方向を変え、左ボタンを押すと前進、中ボタンを押すと後退だ。終了する際はQキーを押せばいい。



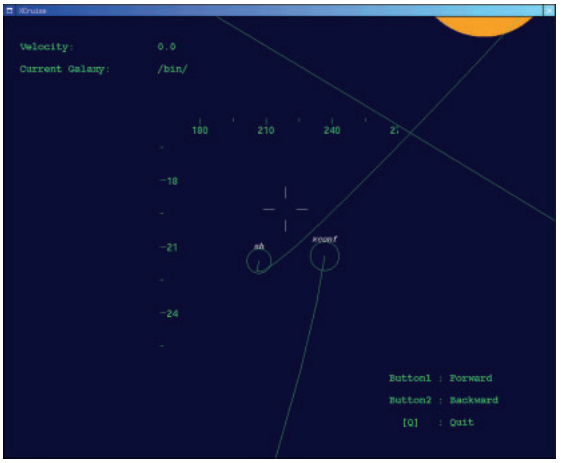
画面1
起動時にはルートディレクトリの構成が表示される。

●ファイルが惑星として表示される

銀河にある程度接近すると、その銀河に入ったと見なされ、ディレクトリ内のファイルが惑星として表示される。惑星の大きさや色、位置などはランダムに与えられるのではなく、一定のルールに基づいて決められている。このため、接近したディレクトリの内容によって、さまざまな銀河の概観を楽しめる（**画面2**）。

具体的には、各惑星の大きさ（質量）はファイルサイズに対応しており、惑星の色はファイル属性により決定される（読み込む権限を持たないファイルは紫色など）。また、銀河内での惑星の位置はファイル名の長さや類似度に応じて決められる。なお、シンボリックリンクはワームホールとして扱われ、リンク元のファイルとは緑の曲線で結ばれる（**画面3**）。

画面2
ディレクトリに接近すると、ファイルが惑星として表示される。



画面3
ワームホールからはリンク先のファイルへ曲線が伸びている。

画面を回転させて球をゴールまで運べ

XROT

バージョン：1.3.2　　種別：フリー

http://www.ci.cs.meiji.ac.jp:8150/siraisi/xrot.html

XROTは、画面を回転させることにより、落下する球をうまくゴールまで運ぶアクションゲームだ。ゴールまでの時間を競うため、壁に衝突することにより生ずるタイムロスをできるだけなくすようにプレイしなくてはならない。慣れてくると、キー操作により球の速度を上げたり、跳ね返りを小さくすることができるようになる。動作にはPseudo Color（256色）表示をサポートしたXサーバが必要だ。

●ビルドとインストール

XROTは、ファイル一式をtar＋gzipしたtarボールでのみ配布されている。Imakefileを利用しているため、「xmkmf」→「make」→「make install」という手順でビルドとインストールを行う。インストール先の初期設定は/usr/X11R6/binだ。

なお、XROTはPseudo Color環境で動作するため、XFree86では256色モードでのみ動作する。ハイカラーやフルカラーでXFree86を動作させている場合は、いったん256色モードで起動しなおす必要がある。

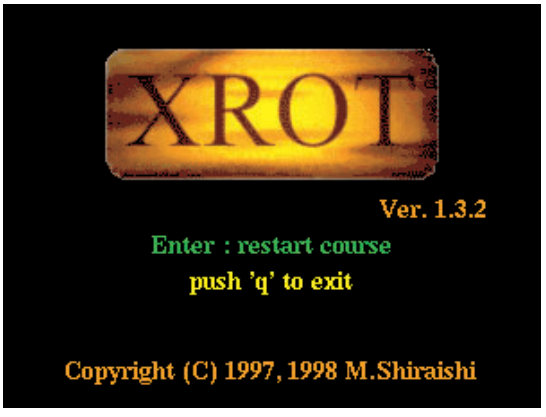
●あなたの挑戦を待つ7つのコース

「xrot」として起動すると、ロゴが回転するタイトル画面が表示される（**画面1**）。スペースキーを押して7つのコースのいずれかを選択しよう。

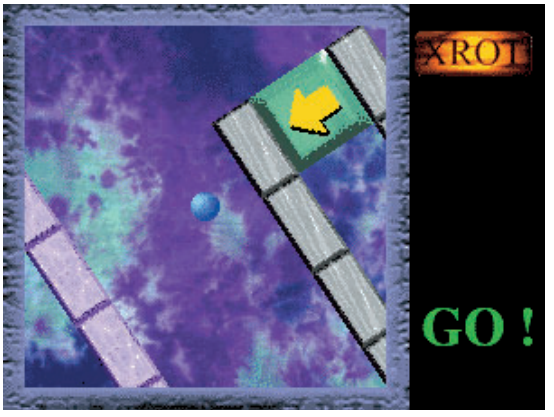
操作の基本は、←／→キーで画面を回転させて、常に下方向に動こうとする球をゴールまで誘導することだ（**画面2**）。壁に衝突すると、球の落下速度が鈍ってしまうので、できるだけ壁に当てないようにしよう。

このほか、スペースキーで球の速度アップ、↓キーで衝突時の跳ね返りの緩和、（球の速度が小さい場合に限り）↑キーでジャンプが可能だ。コースの難易度が上がるにつれ、通路の幅は狭くなり、コース上の障害物も増えてくるので、これらのキーをうまく使いこなそう（**画面3**）。

通常コースの通路は一本道で、進行方向を示す矢印が壁に描かれているので迷うことはない。しかしスペシャルコースは、分岐や斜め方向の通路があって、かなり難易度が高くなっている（**画面4**）。



画面1
ロゴが回転するタイトル画面。256色モードでXを起動しよう。



画面2
落下する球を画面を回転させてゴールまで運べ。

画面3
コースの番号が上がるにつれ通路の幅が狭くなる。

画面4
斜めの通路や迷路など、特に難易度が高いスペシャルコース。

ペンギンが主人公のアクションゲーム

Black Penguin

バージョン：0.2　　種別：GPL

http://www.priebs.de/blackpenguin.html

Black Penguinは、その名の通り黒いペンギンが主人公のアクションゲームだ。ルールは単純で、3角形のフィールドを移動しつつ、すべてのアイテムを獲得すればいい。面が進むにつれて敵キャラの「悪のウィンドウ」（笑）も登場し、ゲームを盛り上げてくれる。レイトレーシングで作成された各キャラのデザインも美しい。実行にはQtが必要だ。

●ビルドとインストール

Black Penguinは、ファイル一式をtar＋gzipしたtarボールでのみ配布されている。tarボールには、ビルド済みの実行ファイル（blackpenguin）が含まれているので、ソースからビルドする必要はない。インストールは、/usr/local/gamesなどに実行ファイルを手動でコピーする。

なお、ソースから作成する場合、gcc 2.7.2を採用しているシステム（Vineなど）では、ビルドが正常に行われないので注意が必要だ。Black PenguinのWebページの「Troubleshooting」に、この問題を修正するパッチが用意されている。

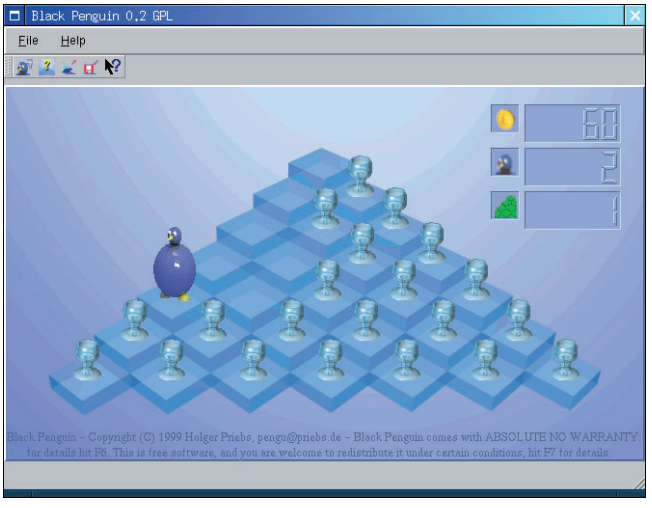
●フィールド上のアイテムをゲット

「blackpenguin」として起動すると、3角形のフィールドを持つウィンドウが表示される。ツールバー左端の[New Game]ボタンでゲーム開始だ。フィールドの頂点に表示されたペンギンをカーソルキーで操作して、フィールド上に置かれたカクテルグラスなどのアイテムをゲットしよう（**画面1**）。アイテムの獲得にはキー操作は必要なく、単に置いてある場所に移動するだけでいい。すべてのアイテムを獲得すると次の面に移る。

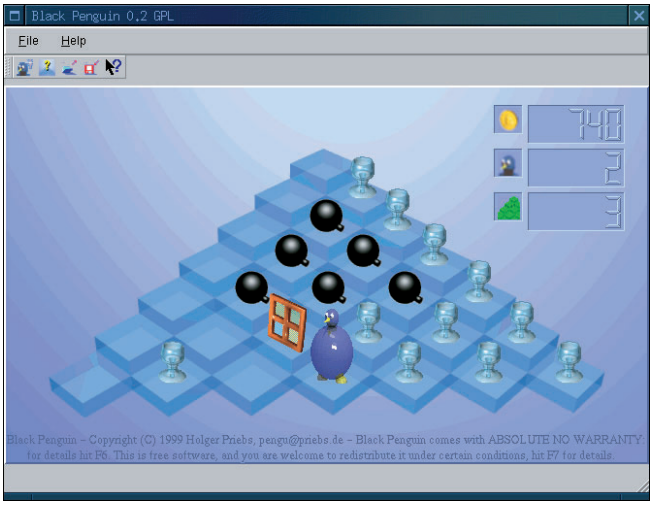
ただし、フィールドから外れるような移動を行うと、ペンギンが1匹失われてしまう。また、面が進むと、フィールド上を自由に動き回る敵キャラ「悪のウィンドウ」や、取ってはいけないアイテム「爆弾」が登場する（**画面2**）。当然、これらに触れるとペンギンが1匹失われる。2匹のペンギンを失うとゲームオーバーだ。また、中には二度触れないと獲得できないアイテムもあるので、状況はしだいにシビアになっていく。

ところで、このゲームでは斜め方向の移動を上下左右のカーソルキーで行うので、方向が90度ずれて感じられる人もいるはずだ。また、viやEmacsになじんでいる人は、カーソルキーなど使いたくないに違いない。[File]－[Configure Keys]でダイアログを開き、4方向のボタンのいずれかと使用したいキーを押すと、操作キーを変更できる（**画面3**）。

画面3
ペンギンの移動に使うキーはカスタマイズ可能だ。



画面1
カーソルキーでペンギンを操作してアイテムをゲットしよう。



画面2
面が進むにつれ、「悪のウィンドウ」や「爆弾」などが登場する。

GUIでポートスキャンを行う

Kmap

バージョン：0.7.1　　種別：GPL

http://www.islc.net/~icszepp/index.html

Kmapは、ポートスキャナであるnmapをGUIで利用するためのフロントエンドだ。自分の管理しているマシンが開いているポートをチェックし、不要なサービスを起動しないようにするのはセキュリティ保護の基本だ。Kmapを使うと、nmapのさまざまなコマンドラインオプションをGUI上で設定でき、スキャン結果を整理した情報を個別のウィンドウで確認できる。動作にはQtとKDEが必要だ。

●ビルドとインストール

Kmapは、ファイル一式をtar+gzipしたtarボールでのみ配布されている。「./configure」→「make」→「make install」という一般的な手順でインストールできる。なお、ビルドに失敗する場合は、Qtのヘッダファイルがインストールされているか、環境変数QTDIRが正しく設定されているかを確認しよう。

また、ポートスキャンはnmapを内部で起動することにより実現しているので、nmapをあらかじめインストールしておく必要がある。現時点での最新版は2.3BETA12だ（http://www.insecure.org/nmap/）。

●スキャンの実行と情報の表示

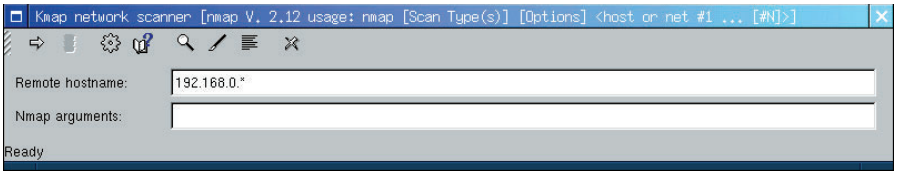
コマンドラインで「kmap」として起動すると、シンプルなウィンドウが開く（**画面1**）。基本的な使い方は簡単で、[Remote hostname]に、ポートスキャンの対象となるホスト名かIPアドレスを入力する。IPアドレスの場合は、複数のホストを一括スキャンすることも可能だ。

ツールバー左端の[Start scanning]ボタンを押すと、nmapによるスキャンが実行される。ステータスバーに「Nmap: exited normally」と表示されればスキャンは終了だ。スキャン結果は、3種類のウィンドウで表示される。なかでも、ポートとサービス名などを一覧表示する「Port View」と、ホストごとにポートをまとめた「Information View」が重要だ（**画面2**）。新たなスキャンを実行すると、これらの表示は自動的に更新される。

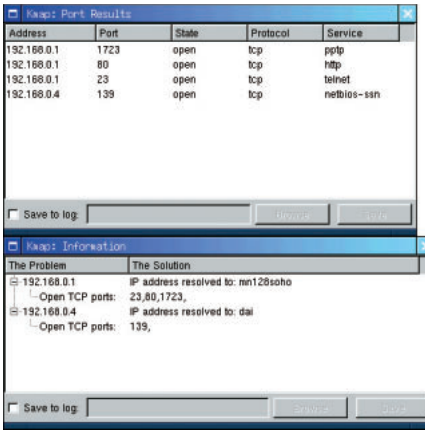
Kmap自身やnmapに対するオプション設定は、ツールバーの[Setup options]ボタンで開く設定ダイアログで行う（**画面3**）。設定項目は多岐にわたるため、詳細はnmapのマニュアルを参照されたい。

不要なサービスが有効になっていることが分かったら、対応するデーモンプロセスが起動しないように、/etc/inetd.confの修正などの対策をとったほうがよい。Red Hat系ディストリビューションの場合は、/etc/rc.d/rc?.d（?は数字1文字）以下の不必要なシンボリックリンクを消すようにする。

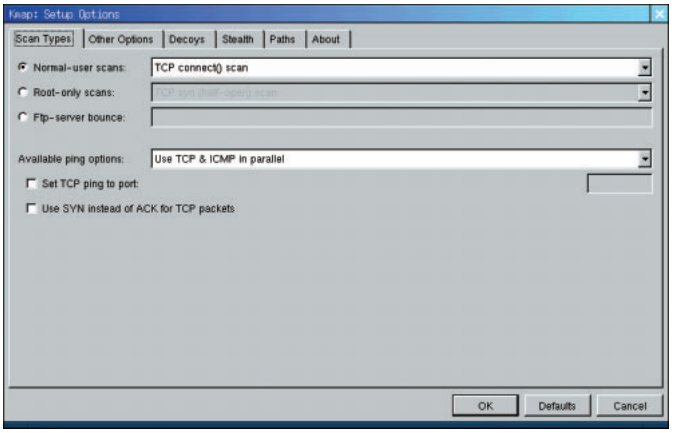
なお、不用意に自分の管理外のマシンに対してポートスキャンを行うと、それだけでクラックと見なされる危険があるので注意されたい。



画面1
スキャンしたいホスト名かIPアドレスを入力する。



画面2
スキャン結果を表示する「Port View」と「Information View」。



画面3
Kmapやnmapの設定はこのダイアログで変更可能だ。

GUIで文字列検索を行う

reXgrep

バージョン：1.2　　種別：GPL

http://rexgrep.tripod.com/rexgrepmain.htm

reXgrepは、指定した文字列を含む行を複数ファイルから抽出して、表示するgrepのフロントエンドだ。文字列やオプションをGUIで設定でき、正規表現を直接指定するほか、初心者により理解しやすいPhrase／All／Some／Noneといった簡単な検索モードを備えている。また、サブディレクトリ以下のファイルを検索対象に含めたり、検索された前後の行をあわせて表示することもできる。動作にはGTK＋が必要だ。

●ビルドと日本語対応の修正

reXgrepは、ファイル一式をtar＋gzipしたtarボールでのみ配布されている。ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。

ダイアログや検索結果で日本語を表示するには、ソースを2個所修正する必要がある。まず、rexgrep.cの272行目の「gtk_init（&argc, &argv）;」の前に「gtk_set_locale（）;」という行を挿入する。また、display.hの87行目の「gdk_font_load（…）」を「gdk_fontset_load」に変更する。

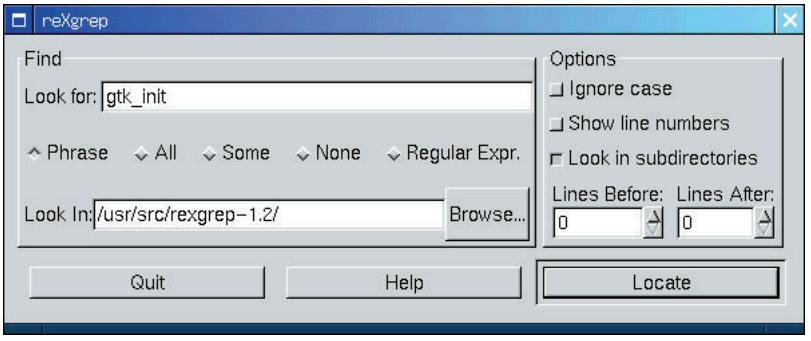
なお、検索はgrepを内部で起動することで実現している。サブディレクトリ以下を再帰的に検索する機能は、GNU grep 2.3以降で追加されたため、それより古いバージョンのgrepでは利用できないことに注意されたい。

●GUIで文字列検索を行う

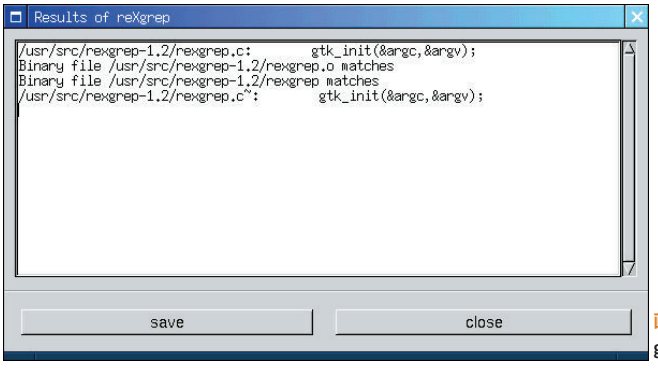
「rexgrep」として起動すると、検索文字列やオプションを設定するためのウィンドウが開く（**画面1**）。使い方は簡単で、[Look for]に検索する文字列、[Look in]に検索対象となるファイル（ワイルドカード可）を指定して、[Locate]ボタンを押せばいい。reXgrep内部でgrepが起動され、検索結果が別ウィンドウに表示される（**画面2**）。こうした検索結果をファイルに保存することも可能だ。

検索モードは、5つのラジオボタンで切り替える。これらの違いは文字列中の空白の扱いだ。初期設定の[Phrase]では、空白も検索文字列に含まれるのに対し、[All][Some][None]では空白を区切り文字と見なして、複数の文字列に対する検索を行う。たとえば[All]の場合は、空白で区切られた複数の文字列をすべて含む行だけが検索される。このほか、正規表現を直接指定する[Regular Expr.]も用意されている。

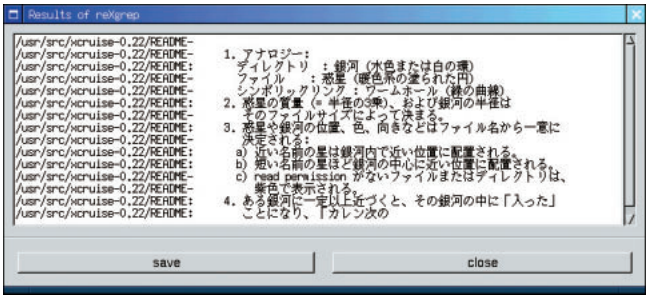
オプションとして、大文字・小文字を無視する[Ignore case]、検索結果に行番号を追加する[Show line numbers]、サブディレクトリ以下も検索する[Look in subdirectories]、検索行の前後も表示する[Lines Before／After]が用意されている（**画面3**）。



画面1
検索文字列と検索ファイル名を指定して[Locate]ボタンを押す。



画面2
grepによる検索結果を表示するウィンドウ。保存も可能だ。



画面3
検索行の前後の何行かをあわせて表示することもできる。

隠喩としてのコンピュータ

1984年に断絶されたもの

文：豊福 剛
Text : Tsuyoshi Toyofuku

Macintosh誕生に至るまでのコンピュータの歴史には、GUIをめぐる技術の系譜があったことは、よく知られている。1979年にゼロックス社パロアルト研究所（PARC）を訪問したAppleのスティーブ・ジョブスは、そこでAltoと呼ばれるワークステーションを見て、衝撃を受けた。ジョブスの目には、Altoが体現するビットマップのGUIは、パソコンの未来を啓示する革新的な技術に見えたのだ。

このAltoの開発に携わった中心人物が、アラン・ケイである。ただしケイは、まったくのゼロからコンピュータの未来を発明したわけではない。GUIの系譜をさらにさかのぼっていくと、ケイに圧倒的な影響を与えた人物に行き当たる。それがダグラス・エンゲルバートである。エンゲルバートがSRIで開発したNLSのプレゼンテーションを見たケイは、「コンピュータとは結局ディスプレイのことなのだ」という啓示を受けたのだ。

GUIの啓示は、あたかもエンゲルバートのNLSからケイのAltoへ、そしてAltoからMacintoshという流れに沿って、継承されてきたように見える。ただし、GUIにかぎらず、何かの系譜を考えるとき、それが現在の視点から過去を解釈していないか注意しなければならない。系譜においては、何かが継承されるだけではなく、何かが喪失され忘却されるものだからだ。MacintoshのGUIの原イメージを、AltoやNLSの中に見出そうとするとき、現在との「連続性」ばかりが強調される。しかし、継承されたものではなく、むしろ、継承されなかったものも、過去から発見しなければならない。

アラン・ケイとSmalltalk

Altoにオブジェクト指向のプログラミング言語「Smalltalk」が実装されていたことは、重要なポイントといえるだろう。ケイによって発明された、このSmalltalkというプログラミング言語は、GUIと切っても切れない関係にあったように思える。それは、GUIを構成するウィンドウシステムに対して、オブジェクト指向プログラミングが非常に有効であるからだけではない。ケイにとってSmalltalkは、開発者だけが利用するものではなく、ユーザーによるプログラミングに対しても開かれたものでもあったからだ。

このSmalltalkは、ケイにとってどのような意味を持っていたのか？　このことを考えるとき、ケイに影響を与えたもうひとりの人物が注目される。それがシーモア・パパートだ。

MITの教育心理学者であるパパートは、1967年にLOGOという児童教育用のプログラミング言語を開発している。LOGOでは、スクリーンにタートル（カメ）と呼ばれるカーソルが表示されており、このタートルを移動させるコマンドをプログラミングしていくことで、線を描画できる。この描画コマンドそのものは単純なものだが、再帰的に実行させることで、かなり複雑な図形でも描画できる。パパートの指導のもと、子供たちがLOGOを使って、複雑な描画プログラムを自由に操作している光景に感動したケイは、LOGOよりもさらに強力なプログラミング言語を開発しようと思い立ったのではないだろうか？

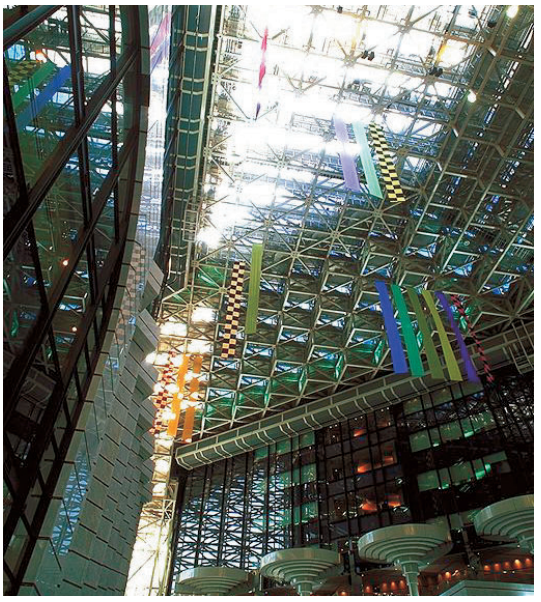
ケイが構想したDynabookの思想に、このことが端的に反映されている。Dynabookは、「ノートあるいは本のようなコンピュータ。1台のマシンとして単独で使えるコンピュータ」として知られており、この「個人のためのコンピュータ」という概念は、Dynabookによって初めて打ち出されたものといわれている。

ケイは、当時のIBMメインフレームに代表されるコンピュータを「厳密に管理された鉄道」とみなし、それに対してDynabookを「自家用車」のようなものとして発想した。「鉄道」に対比される「自家用車」の隠喩は、個人が自由に操作できる点を強調する意図から生まれた表現なのだろう。と同時に、ケイはLOGOを「自転車」に喩えている。DynabookはSmalltalkを実装したマシンとして構想された。「粗雑でゆっくりした動きのワイヤーフレームや、方形波でつくった楽音」といった程度の処理能力では子供は満足しない。ケイにとって、「自転車」であるLOGOの処理能力をより増幅したのが「自家用車」であるSmalltalkなのだろう。

Dynabookについて書かれたケイの論文を読むと、それは子供たちが使うことを目的としたマシンとして構想されたことがわかる。そこには、子供たちがSmalltalkを使って、さまざまなシミュレーションをしたり、グラフィックスや音楽の創作を楽しむ姿が、未来への期待として書かれている。

Macという断絶が意味するもの

このように、AltoとDynabookにおけるSmalltalkプログラミング環境を、ケイは必要不可欠のものとして考えていた。これに対してMacintoshは、Smalltalkのようなユーザープログラミング環境を排除したコンピュータとして



誕生してしまった。Macintoshは、Altoが体現するGUIの表面的効果だけを継承したのだ。

MacintoshからSmalltalkのようなユーザープログラミング環境が排除されたのは、当時のパーソナルコンピュータの貧弱なハードウェア環境による技術的な制約が理由なのかもしれない。仮にそうだったとしても、結果としてユーザープログラミング環境が排除されたパーソナルコンピュータの誕生は、それ以前とそれ以後のコンピュータの概念に決定的な「断絶性」をもたらしたのだ。この断絶の以後においては、どんなにMacintoshのハードウェア環境が向上したとしても、かつて排除されたものが再び取り戻されることにはならなかった（Hyper TalkやApple Scriptを、Smalltalkの代わりに導入されたユーザープログラミング環境と考えるのは、あまりに短絡的だろう）。

このMacintosh以前と以後における「断絶性」の意味は、より広い歴史的文脈から解釈することができる。1984年は、Macintoshが誕生した年であると当時に、リチャード・ストールマンがGNUプロジェクトを開始した年でもある。1984年以前のコンピュータ文化を考えるとき、人工知能に関する研究開発が存在していたという事実は、ともすれば忘れてしまいがちである。1984年以前においては、ストールマンのバックグラウンドはMITのAIラボだったのだし、またアラン・ケイがPARCに移る前は、スタンフォード大学の人工知能プロジェクトに参加していた。人工知能の時代は、プログラミング言語をめぐってさまざまなアプローチが展開した時代でもあり、ケイのSmalltalkもそのような時代背景から誕生したのである。すると、Macintosh以後、ユーザープログラミングが重要視されなくなったことと、コンピュータ科学における研究の重点が人工知能からユーザーインターフェイスへと転換していったこととは、直接的な関連はないにしても、何か共通する時代の変化として解釈することができるだろう。

インターフェイスとしての価値

シェリー・タークルは『接続された心』(Sherry Turkle著、日暮 雅通訳、早川書房刊)で、この「断絶性」を、「計算の文化からシミュレーションの文化への転換」と解釈し、この「転換」を「モダニズムからポストモダニズムへの移行」と関連づけている。

タークルは、コンピュータに対するモダニズムの価値観には、プログラミングが中心にあることを指摘している。

プログラミングとは正しい方法で行なったか否かのどちらかしかない技術的能力で、方法の「正しさ」はコンピュータの計算機的要素によって決まるものであり、リニアで論理的なものである。複雑化した事柄を単純な部分によって分析できるとする還元主義、あるいは世界の働きを説明する統一的概念がありえるという前提、つまり「大きな物語」が、このようなコンピューティング観を可能にしている。人工知能の背景にある考えもこれと共通していて、コンピュータは人間の知性を投影し拡張したものになりえるという考えは、大衆文化においても一般的だったと指摘する。

しかし、今日のポストモダニズムの価値観においては、コンピュータは個人の肉体的な存在を拡張するという考えが採用されているとし、「計算とルール」を本質とするプログラミングのかわりに、「シミュレーションやナビゲーション、インタラクション」が中心にあると指摘する。もちろん、コンピュータの内部で計算が行われていることに変わりはないのだが、ポストモダニズムにおいては「そのことを特に意識することが重要ではなくなった」のだ、と。

このような、モダニズムとポストモダニズムの価値観の違いを、タークルはIBM PCとMacintoshにおける「透明性」の概念の違いに対応させている。IBM PCの透明性とは、その基本的メカニズムが透明でオープンなマシンであり、中にあるもののメカニズムが理解できることである。これに対して、Macintoshの透明性とは、文書やプログラムが魅力的で解釈しやすいアイコンで表されていることであり、操作方法がたやすくわかることを意味するようになった、と指摘する。

タークルは、ポストモダンをシミュレーション文化への移行であるとしたうえで、心理的態度の変化に注目する。Macintoshのユーザーは、現実を代理するもので現実を置き換えることに居心地のよさを感じるようになり、画面上に見えるものを「(インター)フェイスバリュー」(フェイスバリューは額面どおりの価値)でとらえる。これを、「もはや深奥とメカニズムの探求はむだであり、起源や構造に手をつけるよりも変わりやすい表層の世界を探索するほうがより現実的である」と提唱するポストモダンの美的価値観に対応させている。

Macintoshを礼賛するタークルは、シミュレーション文化における居心地のよさばかりを強調しているが、そこに同時に存在する居心地の悪さを感じる感性は持ち合わせていないようだ。次回、タークルのインターフェイス論に対するスラヴォイ・ジジェクからの批判を検討する。

Profile

とよふく つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』(オライリージャパン)『GIMPパーフェクトガイド』(エムディーエヌコーポレーション)などがある。

オープンソース
・
アクティビズム

知的所有権という古い奇妙な因習について

文：安田幸弘
Text : Yukihiro Yasuda

この年末年始は、2000年問題とやらで業界は本気で緊張していたらしい。ぼくが関わっている非営利プロバイダ（？）でも、一人前に2000年問題特別対策チームを結成、寝袋持参で事務所に泊まり込みということをやったけど、人命にかかわるようなシステムを納入しているわけでもなく、特別対策といったって、どこかでチェック漏れのCGIか何かが100/01/01なんて間抜けな日付を表示したら「特別に」速攻で直してやろう、という程度のものである。

「何も起きないなあ」「本当の2000年問題はこれから何年も続くんだぜ」「日付計算がらみのトラブルは、2000年とは別の一般的な品質管理の話だと思うよ」「結局そうなんだけど」などと言いながら、応援（？）に駆けつけた会員からの大量の差し入れや、本場仕込みのキムチ鍋、林立する酒瓶ビール瓶を処理することが最大の「仕事」だった。こんな特別対策なら、来年もやってもいいかな？？

オープンソースの苦労は買ってでも…

こんな調子で明けた2000年だったのだが、一応「継続的にシステムを監視し、ユーザーからの問い合わせに対応する」ということになっているのであまり遠くに遊びにも行けず、結局、正月休みは計画中の新しいサーバに組み込むためのツールのテストでつぶすことになってしまった。まあ、これはこれで面白い仕事なので、わりと嬉々としてやってたわけだが、当然、家族からは不平の嵐。「2000年問題だからしょーがないの」で済ませてしまったけど、何だか「2000年問題」って、言い訳のネタに乱用されている気がしないでもない。

ところで、新しいサーバに使うプログラムはすべてオープンソースやGNUから選ぶことにしたのだが、実際にやってみると結構これはなかなか厳しい。オープンソースと言うのは簡単だけど、標準的な仕様のサーバはともかく、純粋にオープンソースだけで市販の「何とかサーバ」のような細かい機能――たとえばアクセス制御やセキュリティ関連――を実用的なレベルで実現しようとすると、機能や使い勝手、仕様といった面で、苦労することになる。

ところが、オープンソースだのフリーソフトウェアだのにこだわらなければ、NetscapeやLotusあたりのサーバを入れてしまえば全然苦労せずにクリアできる部分だったりする。それでも「オープンソース」なのは、つまり足りない機能を自分で補充する苦労が楽しいから、なのかもしれない。苦労が好きな世界中のハッカーたちがよってたかってハックすれば、どんなことだってできるような気がしてくるのが楽しいのだ。

オープンソースに潜むソフトウェア特許

ただ、正月早々あわてさせられたのは、FreeType Projectの成果物が特許がらみの問題を抱え込む可能性があるというニュースだった。

最近はFreeTypeライブラリも十分安定してきて、筆者もXでTrueTypeを使うようになり、「これは結構イケるな」と思っていたところだったので、このニュースには驚いた。正式な結論が出るまでにはかなりの時間がかかりそうだということだが、「もしかするとクロ」かもしれないツールを前提として構成を決めてしまうわけにもいかない。白黒はともかく、さしあたりプロジェクトからFreeType関連は除くことに決めたのだが、いまでも「こんなもん、特許のうちかなあ」という思いは拭えない。

もっとも公開されたソフトウェアが何かの特許なり著作権なりに抵触するという理由で問題が起きることは、珍しくはない。4.4BSDに含まれていたAT&Tのコードのときのように、かなり敵対的な目的で知的所有権が行使されるケースもあれば、GIFの圧縮アルゴリズムに使われ

ているUNISYSのLZW特許のように、「特許は特許、違法状態は放置できない」といった調子でやんわりとながら執拗に知的所有権の主張が続いているケースもある。今回の件が、もし権利者側からのロイヤリティの要求、または公開の停止を求めるものになるとしたら、オープンソース・サイドからの反発は半端じゃないだろうなと思う。

逆に、特許権の行使を引っ込めたら、オープンソース陣営からは賞賛を受けるだろうが、ビジネス的には権利者がつらい立場に追い込まれる可能性がある。おそらく、この特許も他社の特許とのバーターを含み、それなりの収益を生んでいるはずだからだ。権利者側はきっと今頃、「困ったことになったなあ」と頭を抱えているに違いないと思うんだけど、どうだろう？

特許なんてつまらんものを取るからいけないんだ、というのはおいとくとして、おそらく似たようなことは今後も続くのではないだろうか。インターネットの技術はどんどんオープンな方向に向かい、クローズドな技術はデファクト・スタンダードになり得なくなりつつある。アマチュアプログラマーが公開されている情報を元に作ったプログラムが結果的に何かの特許に抵触してしまうということは大いにあり得る。そもそも、ソフトウェア特許なんてもの自体が間違いなんじゃないかというラジカルな——GNUなら当然の、かもしれないが——疑問さえ浮かんでくる。

「知的所有権」のいかがわしさ

どういう理由でソフトウェアの特許などという珍妙な概念が成立し、大手を振って通用するようになってしまったのか、その詳しいいきさつはよく知らない。ただ、他人には簡単にマネできない知識を元にさんざん苦労して作り上げたソフトウェアだかアルゴリズムだかに対しては、それなりのギャランティが欲しいなあ、という職業プログラマーの自然な感情を理解しないとは言わない。

釈然としないのは、公開情報から作られた互換プログラムが本家の特許を侵害してしまうとしたら、その特許って、保護するに値するような独創的な内容だったのだろうかということ。プログラムに限らない。電話はベルとエジソンの2人がほとんど同時に考え、タッチの差でベルが特許を取得したという話だけど、するってーとどこかに3人目、4人目の発明者がいたんじゃないか、その時代の技術者なら、誰にでも考えつくような技術でしかなかったんじゃないかと思うのだ。

しかし、現実に世界の製造業は特許を始めとする知的所有権で動いているのだから、いまさらそんなことを言っても始まらないような気もする。だが同時に、ソフトウェアの世界、ネットワークの世界では、オープンソースという動きが19世紀的な特許という考え方を陳腐化させようとしているのかもしれないとも思う。

メーカーが山のような特許と企業秘密を保有し、メーカー同志がトランプのカードのように、特許をやりとりしながらモノを作る、ソフトを作るのが、残り1年を切った20世紀的な産業界だったとすれば、特許も秘密もあればこそ、開けっぴろげなオープンソースが地球規模のネットワークでソフトを作る来たる21世紀の時代には、古めかしい特許だの企業秘密だのといった概念のモノ作りは、過去の遺物になってしまうのではないだろうか。

新しい時代ってやつは、いつも古い奇妙な因習を滅ぼしながらやってくるものなのだ。

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

さて、どうやらY2Kで、停電になることもなく、ロシアから核ミサイルも飛んでこなかった。Y2Kの影響は、こしばらく続くとのことだが、とりあえずひと安心。PC業界もY2Kなど、忘れつつあるような感じ。

Windows2000ついに最終版へ

業界のウワサによると、MicrosoftのWindows 2000の製造工程向けリリース（RTM）は1月15日だという。つまり、これが、販売されるWindows 2000のマスタになるわけだ。すでに販売日は2月17日（日本では18日）となっているので、製造に約1カ月かかるということだ。日程からして、日本語版もほぼ同時にRTMが完成する予定と思われる。

長い年月がかかったが、ようやくNTユーザーもUSBやPlug＆Playが使えるようになるのである。筆者は今、原稿執筆用にNT4.0を使っているが、周辺装置が使えなかったり（最近はUSBのものが多い）、ハードウェアの変更が面倒（もっとも、頻繁には変えないが……）といった点がある程度解消されるのではと思っているが、いつものことで心配なので、切り替え（決してアップグレードではない）はもう少し待ってから（できればService Pack1か2が出るぐらいまで）にしたいと思っている。アップグレードにしない理由は、Windows 2000が大丈夫と確信できるまでは、せっかくちゃんと動いている現在の環境を崩したくないのと、元の環境を残しておけば、Windows 2000を使い始めたあと、最悪の事態が起こっても元の環境を使ったマシンを使えばいいだけだからである。どうして、こういうやり方になるのかは、いままで苦労してきた方ならご存じとは思うが……。筆者の場合、原稿書きのマシンは大事な商売道具なので、安全策を取らねばならない。いままでの経験からすると、こうしたやり方が一番確実というわけだ。

ところで、そのWindows NT4.0だが、昨年12月にようやくC2セキュリティレーティングを獲得した。NTのC2レーティングは、1994年に出荷されたWindowsNT3.5で獲得したのが最後で、その後ずっと獲得できないでいた。それが、今頃ようやく認定されたというわけだ。

SUNやはり評判悪し

SUNは12月6日、ECMA（European Computer Manufacturers Assocation。ヨーロッパのコンピュータ関係の標準化団体）に提出していた、Javaの国際標準化を取りやめた。この標準化はずいぶんともめていて、IBMなどは標準化を継続するようにSUNに圧力をかけているとか。また、SUN抜きで標準化を進めようという動きもあるらしい。

これとは別に、SUNのJava政策に対する不満もあちこちであるようだが、そんななか、SUNは、Inprizeと共同開発したLinux用Java2をJava Business Conferenceで発表。しかし、これまでLinux用Javaを開発してきたBlackdownの功績について一言も触れないという失態を演じた。このJava2も、多くはBlackdownのコードに依存しており、その意味では、BlackdownあってのLinux用Javaなのだが、プレスリリースなどでその点にまったく触れなかった。その後、SUNの幹部がBlackdownに謝罪したとの話だが、これについてのリリースは出ていないようである。

また、Java2の企業システム向けの

J2EE（Java 2 Enterprise Editon）も同コンファレンスで発表されたが、これについてSUNは、ブランド利用料をベンダーから徴収し始めた。このJ2EEに対してIBMは非採用を決定し、この点でSUNと対立した。SUNは一般ユーザー向けのJ2SE（Java 2 Standard Editon）のライセンスを無料にしたため、J2EEや小型機器向けのMicro Editonからお金を取るということになるのだが、IBMがJ2EEを否定したことでどうなることやら。今年ももめそうである。ものはいいんだけどねぇ。

PDA市場は？

1月のMac World Expoでは、Macintoshとは基本的には関係のないHandSpringのVISORが人気だったとか、ニュースといっても、Jobsのiが取れた（Interim：暫定CEOという意味でiCEOと呼ばれていた。もちろん、iMacに引っかけてのネーミング）のが最も大きなニュース（いつも基調講演の最後に大事なニュースを発表するのである）というぐらいなので、VISORの人気もわからなくもない。VISORは、Palm Computing（現3Com。分社化が決定されている）の創設メンバーが創業したHandSpring社のPDAで、もちろんPalm-OSをライセンスしている。ただ、Palm ComputingのPalmシリーズと違うのは、SpringBoardと呼ばれる拡張スロットを備えている点。しかも、透明カラーで5色といかにもiMacと並べて使うのに具合のよさそうなカラーリングである。

このVISORに対して、Palm Computingはソニーと提携。Memory Stickを搭載する予定なのだとか。また、年末に日本でも行われたPalm Source（Palmの開発者向け会議）では、カラー対応の次期OSもデモしていた。

モトローラもPalm-OSをライセンスする（もっとも、現在のPalm-OSが動いているのは、モトローラ製の68000ベースの1チップCPUである）という話もあり、急に流行ってきた感じがある（少なくともWindowsCEよりは人気があるようだ）。

それで、Mac World Expoで、来場者がVISORに群がったのは、もうひとつのウワサも影響しているのかも。それは、AppleがPalm-OSを採用したPDAを開発中というウワサである。かつて、Palm Computingの買収をJobsは考えていたという話もあり、ウワサにはいくばくかの信憑性がある。Jobsは、自分を追い出したスカリーの独自プロジェクトであるNewtonをことごとく嫌っていて、iCEOになるやいなや、開発を中止してしまった。だが、PDAには気があるようで、そこで選ばれたのがPalmというわけ。Comdexぐらいまでには、何か出るのでは？　という気がするのだが……。

インテルはどう？

年末のクリスマス休暇前ぎりぎりに世界最高速800MHz版PentiumIIIを発表し、トップランナーに返り咲いたインテルだが、年明けのCES（Consumer Electronics Show）でLinuxベースのWeb端末を公開した。これは、インテルブランドとして製造され、プロバイダ経由などで販売されるのだとか。日本ではNECの運営するBiglobeの名前が挙がっている。

インテルブランドの完成品コンピュータ機器というのもすごいが、このWeb端末、PC/AT互換機ではなく、Web端末専用のハードウェアで、しかもLinuxベースというのだからまた驚き。こんなものが簡単に手に入る（どう考えてもPCよりも高くなるとは思えない）のだから、まあ、いろいろなことが起こるものである。CPUやチップセットメーカーであるIntelがセットビジネスに参入するとなると、PCメーカーもあまりいい気持ちはしないだろうし、Linuxと聞くとMicrosoftもいい気持ちがしないであろう。もっとも、PCメーカーにできるのはせいぜいAMDを採用するぐらいだし、Microsoftもインテルの自社ブランド製品にまで文句を付けられる筋合いでもないのだろうが……。このハードウェア、拡張などはUSBで行うのだとか。個人的にはLinuxのUSBサポートにインテルが全面的に協力してくれると、対応が進むと思うのだが。

HandSpringのPDA、VISOR。スケルトン5色でiMacユーザーにはうけそう？
http://www.handspring.com/products/vindex.asp


Intelが年明けに発表したLinuxベースのWeb専用端末。

正しきタコへの道

初級Linuxer養成講座

第6回　「ひとり管理者」の心得

Linuxを使うにあたって、避けて通ることができないのが「システム管理者」としての作業である。しかし、個人が自分専用に自由に使っているLinuxマシンでも「管理作業」は必要なのだろうか？ 結論からいえば、難しいことを覚えたり、面倒な日常作業をする必要はないのだが、自分のマシンの機能を最低限維持したり、新しいアプリケーションを使えるようにするためには、「rootユーザー」としての知識が少しだけ必要になるのだ。

文：竹田善太郎
Text : Zentaro Takeda

新聞やテレビなどの一般マスコミも巻き込んで、世界中が大騒ぎをしていた「2000年問題」だが、予想されていたような大混乱は起こらずに済んだようだ。まだ「2月29日」に大きな問題が起こる可能性もあるのだが、とりあえずはひと安心といったところだろうか。しかし、小さなトラブルはいろいろと起こっているようで、筆者の身の回りでもいくつかトラブルがあった。

たとえば、ケーブルTVのホームターミナルが、**電源を入れるとエラーメッセージ**を出すようになってしまった（**写真1**）。早速CATVの会社に問い合わせると、他の契約者でも同じトラブルが出ていたようで、原因は分からないがメーカーに問い合わせ中だという。2000年問題が原因なのかどうかは明言していなかったが、**年が明けたとたん**のトラブルなので、どうも疑わしい。結局、2日後くらいに直ったが、ホームターミナル側ではなく、CATVの送出局側の機器に問題があったようだ。

写真1
CATVの「2000年問題（?）」の証拠写真。家の中のホームターミナル（2台）が、どちらも同じ症状を示すようになったのだ。

また、マンションの玄関にある宅配受け取りボックスの操作画面が、**ハングアップ**して使えなくなってしまった。こちらは、三流メーカーの製品なので以前からトラブルが多かったのだが、今までのトラブル（主にカードリーダ部分の機械的故障）とは症状がまったく違ったので、やはりY2Kがらみではないかとにらんでいる。

このほかの、家の中の家電製品（ビデオのタイマーなど）については、なんのトラブルもなかった。PC関連でも、サーバとして使っているLinuxマシンは起動したまま年を越させたが、まったく問題は起こらなかった。

結局、Y2K関連で問題が起こったのは、ふだんからどうも信頼がおけないと感じていた部分だった気がする。CATVについては、以前から送出中断や音声不良などのトラブルが多かったし、インターネット接続サービスも、「始めるぞ」とアナウンスしてからずいぶん経つのに、一向に始まる気配がないばかりか、気がつくと**集合住宅は除く**などという、いまどきにしては間抜けな注意書きが告知のWebページに追加されていたりする。宅配ボックスについても、毎月2、3回はトラブルが起こっていて、中の荷物が取り出せなくなることが多かったし、サービスマンを呼んでも修理に来るのに時間がかかることが多く、困っていたのだ。

ブランドだけを頼りにものを選ぶのは馬鹿げているのだが、こういうことが多いとやっぱり「有名ブランド」の製品を優先的に選びたくなる気持ちは理解できる。ふだんから安心感をもてる製品やサービスは、いざというときにも頼りになったり、たとえトラブルがあっても**気持ちよく**解決できるというものなのだろう。とはいえ、ここ数年は、一流メーカーの製品にも裏切られることが多くて、油断ならないのもまた事実である。

システム管理とセキュリティ

さて、Linuxの初心者向けの書籍や

雑誌記事では、必ずといってよいほど「Linuxではシステム管理作業が必要になる」というようなことが書かれている。WindowsやMacintoshなどのPC用のOSとは違って、なにやら**面倒な**雑用をこなさなければならないらしい。「ユーザーアカウントの設定」、「ネットワーク環境の維持管理」、「ファイルのアクセス権の設定」、「バックアップ」、「アプリケーションのインストール」など、難しそうな題目が並んでいるのに面食らったことはないだろうか。

「UID」、「GID」、「SUID」、「アクセス権」などの見なれない単語も多く出てきて、単に個人的に使いたいだけなのに、どうして**こんなにも面倒な**ことを覚えなければならないのか、といやになった人もいるかもしれない。

また、面倒だからと一般ユーザーのアカウントを作らないで、rootユーザーのままでLinuxを使っていると、「**そんな使い方をしてはいけない！**」と横やりを入れてきて、「**そもそもUNIXのrootユーザーとは**……と講釈を始めるUNIX「オタク」にうんざりした経験のある人も多いだろう。

筆者もそんな「UNIXオタク」の一種なのかもしれないが、会社で仕事に使うPCならともかく、個人が自分の財布から出したお金で、趣味に使っているLinuxマシンについてまで、いろいろと**ウルサイ**ことを押しつけるのはちょっと変なのではと思っている。また、管理作業をおろそかにすると「**ハッカーが侵入してきて**、マシンをめちゃめちゃにするかも」という心配を口にする人もいるが、少なくともダイヤルアップ回線でごくたまにインターネットに接続するだけの使い方なら、そんな心配はいらない。

もちろん、この先（いつになるかさっぱり見当もつかないが）「常時接続」が一般家庭のユーザーにも気軽に使えるようになったら、多少はそういう心配もしなければならないのだろうが、実際にそうなった場合には、セキュリティの心配をするのは一般のユーザーの仕事ではなくて、接続業者の側で対処すべき仕事だと思う（そのために高い接続料を払っているのである）。

とりあえずは、ADSLを含めた常時接続回線が「普及」（東京都内や一部大都市でしか使えないなどという、**時代錯誤な一極集中**の状況では、「普及した」というべきではない。およそ電話線が引かれているところなら、**どんな山の中**でも使えるようになって初めて「普及した」と呼ぶことにしたい）するのは遠い将来の話だから、自宅のコンピュータのセキュリティについて余計な心配をするのはやめて、それよりも**クレジットカードの管理**に気を配るとか、信頼できない相手には自分の住所や電話番号を教えないとか、**エンジンをかけたまま**乗用車を放置しないといった、もっと**身近なセキュリティ**に心を砕くべきだろう。

「root」って何だ？

さて、Linuxの管理というと、必ず出てくるのが「rootユーザー」という言葉である。rootユーザーとは、いわばLinuxの「デフォルトのユーザーアカウント」とでもいうべきもので、Linuxをインストールした状態では、基本的にはrootユーザーのアカウントしか存在しない。実際には、ほとんどのLinuxディストリビューションでは、インストール時にroot以外のユーザーアカウントを作成する画面が出てくるので、たいていのユーザーはここで**自分専用のアカウント**を作っているだろう。そして、通常はここで作成したアカウントでログインして、Linuxを使うことになる。だから、rootユーザーというものの存在を知らなかった読者も、あるいはいるのかもしれない。

rootユーザーは、難しく言うと「ユーザーID（UID）がゼロ（0）のユーザー」で、そのマシンの上で**どんなこともできる**ユーザーのことである。実際には、UIDが0であればユーザー名は何でもよくて、UNIXの世界ではroot以外に「toor」とか「sysdiag」とか「admin」などのユーザーIDを作って、そのUIDを0するといったこともよく行われていた。これらのユーザーIDは、すべてrootユーザーと同じ効力を持っていて、システム内でどんなことでもできるようになっている。

「どんなこともできる」とはどういうことかと言えば、マシン内のすべてのファイルやデバイスを自由に読み書きしたり、ファイルの削除や変更したり、すべてのプログラムを実行することなどが、自由に行えることを意味する。すなわち、システムの大事なファイルを不用意に書き換えてしまえば、マシンを**使えなくしてしまう**ことだってできてしまうのだ。もちろん、そんなことをわざわざやるユーザーはいないだろうが、ちょっとした不注意でLinuxが使えない状態になってしまう可能性は大いにある。

root以外のユーザー、すなわち「一般ユーザー」は、基本的に自分の作成したファイルの読み書きと削除しかできず、rootユーザーや自分以外のユーザーが作成したファイルについては、読み書きに制限があったり、削除などのファイル操作はできない。

Linuxの場合、システムの動作を決める重要なファイルについては、root

ユーザーやその他の特殊なユーザーアカウントでないと変更や削除ができないようになっているので、逆に、一般ユーザーのアカウントで作業している限りは、これらのファイルを**壊してしまう**心配はないといえる。だから、通常、Linuxでアプリケーションを使うなどの作業を行うときは、root以外の一般ユーザーとしてログインすることになっているのだ。

rootだけでことは足りるか？

rootユーザーなら、マシンを使うのに一切の制限がなく、アプリケーションの使用にも支障がないというのであれば、1人で使うマシンに一般ユーザーのアカウントなど必要ないのではと考える人もいるだろう。ある意味でこれは正しい。Linux以外のPC用OS、たとえばWindows 98では、一応「ユーザーアカウント」を作ることができて、ユーザーごとに設定を変えたりもできるのだが、ファイルの読み書きや削除については、どんなユーザーでも自由にできるようになっているし、アプリケーションの使用などについても、何の制限もない。Windows 98では、**すべてのユーザーがrootユーザー**であると考えることもできるのだ。だから、同じように個人が1人で使っているLinuxマシンなら、rootユーザーのアカウントだけで使い続けるのも、それほど不自然ではないかもしれない。

そういう意味では、rootユーザーだけでも十分ということもできるのだが、先に述べたように、rootユーザーのままで作業していると、ちょっとしたミスでLinuxが使えない状態になってしまう危険がある。また、前回説明した「telnet機能」を使って、ほかのマシンからLinuxマシンにログインしようとする場合、**rootではログインできない**ことが多い。これは、「ハッカー」がrootユーザーのパスワードを盗んでマシンにログインして、**悪さ**をすることのないように、という配慮から、このように設定されているのだ。だから実際には、root以外のアカウントを作って、ふだんはそれを使うようにしておいたほうがよい。

ユーザーアカウントの追加

業務に使っているUNIX（Linux）マシンなら、システム管理者の主要な仕事のひとつがユーザーアカウントの追加や変更などの作業だ。しかし、個人で使っているマシンの場合は、ユーザーアカウントの追加が必要になることは少なくて、特別なアプリケーションをインストールする場合を除いて、通常はLinuxをインストールしたときに、インストーラが問い合わせてくるままに、自分用のアカウントを作っておけばそれでよい。

実験用に別のユーザーアカウントを作ったり、家族用にアカウントを追加したい場合には、useraddコマンドを使ってアカウントの作成を行えばよい。useraddコマンドは、Red Hat系のLinuxディストリビューション（TurboLinuxなどを含む）で使えるが、これ以外のディストリビューションでも同様のコマンドが用意されていることが多いので、それについては各自で調べてみてもらいたい。

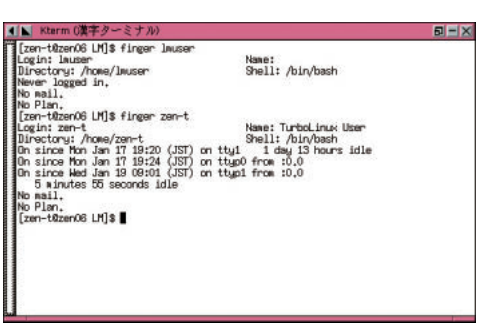
useraddコマンドを使って、新しいユーザーID（たとえば「lmuser」というユーザーID）を追加する場合は、rootユーザーとしてログインした状態（あるいは後述する「suコマンド」を使って、一時的にrootユーザーになる）で、次のようにコマンドを実行すればよい。

```
# useradd lmuser
```

すると、lmuserという名前のユーザーアカウントが作成され、このユーザー用のホームディレクトリの作成、シェルの設定ファイルなどが自動的にコピーされる。昔のUNIXでは、UIDやGIDの割り当て、/etc/passwdの編集、ホームディレクトリの作成、ホームディレクトリの所有権の設定など、めんどうな作業がいろいろと必要になったのだが、useraddコマンドはこのような面倒な作業をすべて自動的に行ってくれる。ただし、ログインパスワードの設定だけは行ってくれないので、useraddコマンドを実行しただけでは、

画面1　ユーザーIDの追加からパスワードの設定まで

ユーザーIDの追加は、X Window System上で動作するGUIツールでもできるようになっていることが多いが、コマンドラインのuseraddを使った方がずっと楽である。



画面2　fingerコマンドでユーザーIDを確認

ユーザーIDがきちんと作成されたかどうかを確認するなら、fingerコマンドを使うのがてっとり早い。ちなみに、fingerコマンドを使えば、そのユーザーがメールを読んでいるかどうか、現在ログインしているかどうか、最後にログインしたのはいつかなどの情報もわかる。

作成したユーザーIDは、パスワードなしでログインできる状態になっている。そこで、useraddコマンドに続いて、passwdコマンドを実行する。

```
# passwd lmuser
```

すると、

```
New UNIX password:
```

というプロンプトが表示されるので、このユーザーIDに設定したいパスワードを入力する。さらに、

```
Retype new UNIX password:
```

というプロンプトがさらに表示されるので、先に入力したパスワードを繰り返し入力する。すると、

```
passwd: all authentication tokens
updated successfully
```

と表示され、パスワードの設定が成功したことがわかる（**画面1**）。

ユーザーIDが正しく作成されたかどうかは、そのユーザーIDでログインしてみればわかるが、fingerコマンドでも確認できる（**画面2**）。

いったん作成したユーザーIDを削除したくなったら、次のようなコマンドを実行すればよい。

```
# userdel lmuser
```

もし、このユーザーのホームディレクトリも一緒に削除したいのなら、-rオプションをつけてuserdelコマンドを使えばよい。

```
# userdel -r lmuser
```

suコマンドとrootログイン

ユーザーIDの追加や削除、次回に説明するファイルのアクセス権の設定、あるいはアプリケーションのインストールといった作業を行う場合は、rootアカウントでログインした状態で作業する必要があるのだが、ディストリビューションの種類によっては、rootアカウントでログインすると、「**rootでログインするな！ suコマンドを使え！**」というような意味の警告メッセージが表示されることがある。もちろん、さっきから述べているように、個人で使っているマシンなら、rootユーザーとしてログインして使っても何の問題もないので、このようなメッセージはあまり気にしなくてもよい。

しかし、一般ユーザーとしてログインしている状態で、何らかの事情から管理者としての作業を行う必要が生じた場合、いちいちログアウトしてからrootでログインし直すのは面倒である。このようなとき、一般ユーザーとしてログインしたままで一時的にrootユーザーの権限で作業したい場合は、suコマンドを使えばよい（**画面3**）。

suコマンドは「Substitute User」の略なのだが、rootユーザーの別名が「スーパーユーザー」ということもあって、これを「Super User」の略だと勘違いしている人も多いようだ。それはともかく、suコマンドは、ログインユーザーIDを**一時的に**変更するためのコマンドである。一般ユーザーとしてログインしている状態で、

```
$ su
```

と入力すると、パスワードを訊かれるので、rootユーザーのパスワードを入力する。すると、これ以降実行するすべてのコマンドは、rootアカウントでログインした場合と全く同様に、rootユーザーの権限で実行されるようになる。

rootユーザーの状態から抜けて、元のユーザーIDに戻るには、

```
# exit
```

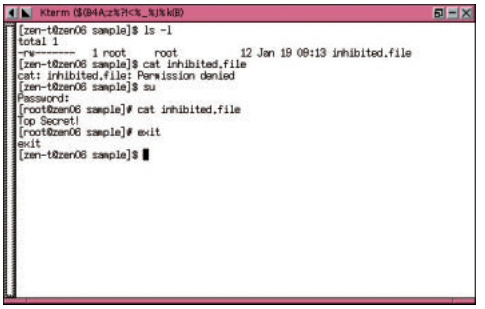
と入力すればよい。ちなみに、suコマンドはrootユーザー以外のユーザーIDを一時的に使うときにも使える。たとえば、lmuserというユーザーIDの権限を一時的に使いたい場合には、

```
$ su lmuser
```

とすればよい。

●

次回は、管理者としてのもっとも面倒な作業「バックアップ」について、その必要性と手段を考えてみよう。



画面3 suコマンドの使い方
rootユーザーしか読めないファイルを、一般ユーザーの状態で読もうとすると怒られる。そこで、suコマンドを使って「一時的に」rootユーザーになりますと、そのようなファイルも読めるようになる。ちなみに、コマンドプロンプトをよく見ると、「$」（これはbashの場合。csh系シェルの場合は「%」）というシンボルが「#」に変化して、rootユーザーの状態になったことを示している。

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台

～Linux.comの翻訳記事/韓国WDBと提携～

日刊アスキー Linuxでは、今月もまた新しいコンテンツを追加した。米国Linux.com（http://www.linux.com/）の「Security」コーナーの連載である。さらに、韓国のWeb Data Bankと提携し、韓国のLinux関連ニュースも送ってもらうことになった。

今回は、上記2つのトピックと、ここ1カ月で取り扱ったニュースの中でアクセス数が多かった記事の紹介、さらに、今後日刊アスキー Linuxで実施する予定の新企画についてお話する。

Linux.com

1月21日現在、日刊アスキー Linuxには、Linux.com（http://www.linux.com/、**画面1**）との提携のもと、2本の記事が掲載されている。「イントラネットの基本」と「Linux Capaility」だ。

日刊アスキー Linuxは,Linux Todayともすでに提携しており、Linux界の動きや話題などがメインの「Feautures and Opinion」（http://features.linuxtoday.com/）の記事を翻訳/掲載している。このコーナーは、Eric S, Raymond氏やRichard Stallman氏といった著名人が、思い思いのトピックを語るコーナーでかなりの人気がある。

画面1 新たに提携したLinux.com（http://www.linux.com/）

こうした状況の中でLinux.comとの提携話が持ち上がったわけだが、読み物ページはすでにLinux Todayコーナーがあるので、今回のLinux.comは役に立つ技術情報がいいだろう、ということになった。Linux.comにアクセスしてみるとわかるが、同サイトには、「Resources」と題して16のコーナーが作られている。その中から、Securityコーナーを翻訳することになったのは、読者の関心が高そうであることと、実はセキュリティに関するノウハウは、Webや雑誌などで取り上げられる頻度が少ないのではないか、という理由からだ。「セキュリティなんて地味っぽいし人気がないのでは？」と思われるかもしれないが、大手書店チェーン、書泉の古田島氏による「Linux関連書籍今これが売れている!」（http://www.ascii.co.jp/linux/bookstore/column/）で、翔泳社の「PC-UNIXサーバのためのクラッカー撃退計画」がランキング上位を占めていたことからもわかるとおり、切実な問題だけあってじつは意外と人気がある。

当初は、Securityコーナーではなく、初心者向けの「Jump Start」にしようかという案もあったのだが、「○○のインストール」や「○○の使い方」といったHow Toモノは、それこそLinux magazineなどでも多く連載されているので見送ることにした。

では内容を紹介しよう。まずは、「Linux Capabilities」（原題も。著者はJim Hewlett氏）。この記事は、開発が進められている「ケーパビリティ」によるセキュリティモデルの解説だ。このセキュリティモデルは、現在一般的に使われているrootによるセキュリティモデルと比較すると、かなり細かい設定が可能になる。rootも含めて、ユーザー（プロセス）ごとに細かい実行権限が設定され、rootですら権限が与えられないプロセスも存在可能になるのだ（それでは困るのでは？ と思われるだろうが、困らないような仕組み

が用意されている)。つまり「rootでログインされてしまったらクラックされ放題」といった弱点を改善することが可能になるのだ。記事中では、ケーパビリティモデルの基本的な概念から始まって、現時点でケーパビリティモデルを実行する方法が、ソースコードのサンプル付きで示されている。かなり歯ごたえがあるので、実際にこのセキュリティモデルをテストする方は少ないだろうが、将来ケーパビリティモデルが完成した時のために知識を蓄えておいてもいいだろう。また、単に読み物として新しい仕組みにザッと目を通してみるのも面白いだろう。

次の「イントラネットの基本設定」(原題「Basic Network Configuration for an Intranet」。著者はBrad Marshall氏)はイントラネットにおける、クライアントとルータに関する基本設定を扱ったもので、IPマスカレードの方法などが記述されている。便利なGUIツールなどは記事に登場しないが、そのぶん深い知識を身につけることができるだろう。シリーズものなので、ほかの関連記事も随時掲載していく予定だ。

韓国Linuxニュース

そろそろ年末という時期に、韓国でサーバのホスティング事業などを手がけるWEB DATA BANK(http://www.wbd.co.kr/、以下WDB)のプレジデント、Kim Dea Sin氏が来日した。日刊アスキー Linuxのスタッフが、LinuxWorld Expo/Tokyo '99の展示会場でKim氏に面会しており、今回の来日でも会うことができた。そのときに決まったのが、WDBと日刊アスキー Linuxとの提携話だ。WDBは、ホスティング事業のほかに、「qLinux」というディストリビューションや、TurboLinuxの販売を手がけていたり、Linuxの技術者を雇い入れるなど、Linux関連の事業にも力を入れているのだ。

余談だが、韓国では政府のコンピュータ普及運動によって、郵便局でも4万円程度の低価格PCが売られていて、Linuxマシンがあるそうだ。Linux magazineの2000年2月号154ページ〜155ページ、安田幸弘氏による「オープンソース・アクティビズム」にも、このあたりの話が書かれているので、ご記憶の方も多いだろう。Kim氏の話によると、韓国という国は政府が「やろう」といった目標を掲げると、みんなでその目標に向って邁進するそうで、急速にパソコンの普及が進んでいるようだ。日本のコンピュータの普及率は、経済企画庁「消費者動向調査」によると、1999年3月時点で29.5%と低くはなさそうだが、インターネットの普及率のほうは人口比10.8%だそうだ(INTERNET Watch『「インターネット白書'99」で見る日本のインターネット事情(5)』http://www.watch.impress.co.jp/internet/www/article/1999/0705/hakusho5.htm)。人口が多いぶん絶対数は高いものの、普及率はオーストラリアやシンガポール、台湾よりも低いという。韓国は4.6%だそうだが、今後急激に高まるかもしれない。

話題を元に戻す。WDBはLinuxのニュースサイトを運営する計画があり、その記事をこちらにも掲載することになった。現在は、WDBの編集部がまだ試験稼動中なので、とりあえずは2本ほど記事を送ってもらっている。http://www.ascii.co.jp/linux/news/today/article/article359392-000.htmlに掲載した「韓国Linuxニュース:韓国のLinux商標権紛争」と、http://www.ascii.co.jp/linux/news/today/article/article351904-000.htmlの「韓国Linuxニュース:生越昌己氏インタビュー」がそれだ。WDBのWebサイトは、2月にオープンする予定とのことだ。

ちなみに、「Linux」をハングルで書くと、**写真1**のようになるそうだ。1文字目が“li”、2文字目が“nuk”、3文字目が“s”ということである。これは、WDBの李さんという方からメールで教えていただいた(ハングルは画像で送ってもらったのだ)。李さん、ありがとうございます。

TransmetaのCPU発表へ

1月もいろいろあったが、この1カ月間でアクセスの多かったニュースを少し振り返ってみようと思う。

現在はまだ最終的なアクセス数の集計が出ているわけではないが、瞬間的に多くのページビューを獲得したのが、Linus Tovaldsが在籍する米Transmeta社の新CPU、Crusoe(**写真2**)の記事

写真1 ハングルで書いたLinux

だ。この記事は、TransmetaのWebサイト（http://www.transmeta.com/）にアクセスして書いたのだが、なかなか先方のWebサイトにアクセスできずに困った。現地時間で1月19日にWebサイトをオープンする、という話はすでに広まっていたため、それこそ世界中からアクセスが集中していたらしい。ガランとした早朝の編集部で、ひたすらWebブラウザの更新ボタンを叩きつづけるのは、かなり侘びしかった。

そうしているうちに、ようやくTransmetaのWebページが現れたので、すぐに記事を書いて、「Linus Tovalds氏在籍の米Transmetaが新CPUについてプレスリリースを！」http://www.ascii.co.jp/linux/news/today/article/article363008-000.htmlとして公開した。コラムにも書いたが、目新しい機能も満載で、なんとも気になるCPUである。Linux関連のトピックならば、やはり同社がインターネット端末のためにOEM提供する、「Mobile Linux」も、気になるところだ。このMobile Linuxは、オープンソースとしてリリースされるという（TransmetaのFAQページを参照のこと）。

さて、このTransmetaのWebページだが、今までの沈黙のお詫びというわけでもないのだろうが、盛りだくさんである。Crusoeのデータシートやホワイトペーパーといった各種資料、モバイル端末の写真まで公開されている。また、Transmetaの発行する「Going Mobile News」や「Crusoe Developer News」の申し込み、CrusoeのDeveloperサイトに登録する事も可能だ。そして、別の意味で注目なのが、「About Transmeta」（http://www.transmeta.com/about/、**画面2**）ページの中に貼られている集合写真だ。きっと世界中で同じことをやっている人が何人もいるだろうなあ、と思いながら、ついついLinus Tovalds氏を探してしまった。写真は、1348×419ドットと、かなり大きいのだが、その中に147人（実際に数えた）ものスタッフが写っているので、1人の顔はせいぜい40ドット四方の大きさなのだ。しかしなんとか判別してみるに、おそらくは中央の前から3列目、黒いシャツの人の後ろに立っている、小豆色のシャツを着た方ではないだろうか？ 皆さんはどう思われますか?

Tovalds氏絡みでもうひとつ。TransmetaのFAQページ（http://www.transmeta.com/about/faq.html）には、Tovalds氏についてのFAQも掲載されている。「What is Linus Torvalds's role at Transmeta?」という箇所で、その答えは、Code Morphing Softwareの開発に携わっている、となっている。Code Morphing Softwareは、Crusoeの最重要テクノロジーであり、Torvalds氏がCOMDEXで「ソフトウェアとともに開発されたCPU云々」と発言していたことが伝えられているから、ナルホドと納得。Crusoeに関しては、近々続報も行う予定である。

写真2 Linus Tovaldsが在籍する米Transmeta社の新CPU、Crusoe

フリーのWebグループウェア「TrueOffice」も人気

Transmetaの記事が人気があったのはわかるが、もうひとつ記録的とも言っていいアクセスがあったのが、「フリーのWebグループウェア『TrueOffice』登場」（http://www.ascii.co.jp/linux/news/today/article/article359840-000.html）だ。これは、QUPA.com（http://tr.qupa.com/）というところ（会社組織ではない）が作成したWebグループウェアで、フリーの製品となっている（**画面3**）。読者からの感想メールはいただいていないため、どういった原因でアクセスが増えたのかは分からない。ただ、グループウェアは有料の製品が多い分野なので、フリーのTrueOfficeは目を引いたのかもしれな

画面2 Transmeta社の社員全員集合！ Linus Tovaldsはどれだ！

い。TrueOfficeの機能は、「グループ管理」「スケジュール管理」「行き先表示」「施設予約」「掲示板」「ToDo管理」「おしらせ」「WEB LINK」と、一般的な内容なのだが、次期バージョンでは、①ユーザー独自の処理ルーチンの組み込み、②携帯情報端末、携帯電話との連携、③メールサーバとの連携、を図るという。

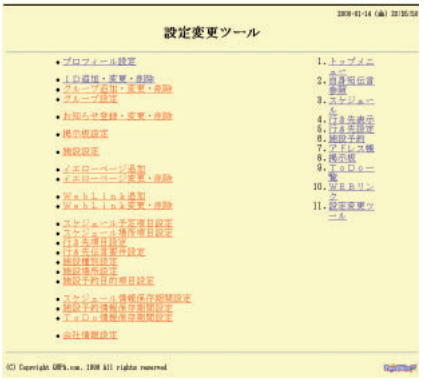
現在同社のWebページでは、デモンストレーションとして、実際にTrueOfficeにインターネットからアクセスすることが可能になっている。Apacheが使えるLinuxマシンを持っているならば、ダウンロードして稼動させてみるのもいいが（インストールから試用するまで、10分とかからないだろう）、それも面倒ならば、http://tr.qupa.com/にアクセスすればよいというわけだ。また、同サイトでは、アクセス解析機能が付いたカウンタのサービスや、ホームページビルダーのユーザーコミュニケーションページも運用している。

最後に、ちょっとお遊びネタだったのだが、意外と人気が出たのが、「Microsoftがソースコードを公開！？」（http://www.ascii.co.jp/linux/news/today/article/article356224-000.html）だ。じつは全然Linuxとは関係がないお話だったのだが、http://www.microsoft.com/の中で、サンプルではないソースコードが公開されていたので、ニュースにした。「Linuxと関係ないじゃないか!」というお怒りのメールは来なかったので、そこそこ皆さんに「ふうん」と思っていただけたかもしれない。

●

2月の初旬あたりから、日刊アスキー Linuxではまた別のコーナーを立ち上げようとしている。どちらかというと実用的なコーナーになる予定である。お楽しみに！

（日刊アスキーLinux編集部・吉川）



画面3 フリーのWebグループウェア「TrueOffice」

Column

TransmetaのCrusoe

TransmetaのCrusoeは、ホワイトペーパー「The Technology Behind Crusoe Processors」（各種資料は、http://www.transmeta.com/press/にアクセスするとよい）を読むと、興味深い仕組みのプロセッサだといえる。

●2種類の製品ラインナップ

Crusoeは大きく分けて2種類の製品が発表されている。Windowsプラットフォームが動作するような、小型軽量のモバイルPCをターゲットとした「TM5400」と、“Mobile Linux”を搭載するような、小型のインターネット端末で使うための「TM3120」である（写真2）。

TM5400は、動作周波数が500～700MHz、L1キャッシュが128Kバイト、L2キャッシュが256Kバイトで、パワーマネジメント時でも、後述するLongRun Power Managementにより、DVD再生ソフトといったアプリケーションをスムーズに使うことができる。価格は500MHz版が119ドル、700MHz版が329ドルとなっている。配線プロセスは0.18mmだ。なお、166MHzのDDR-SDRAMに対応している。製品は2000年中ごろに登場予定である。

TM3120は、動作周波数が333MHz～400MHzまで。L1キャッシュを96Kバイト搭載し、L2キャッシュは搭載していない。配線プロセスは0.22mm。価格は333MHz版が65ドルで400MHz版が89ドル。TM3120は、製品がすでに用意されている。このTM3120と、TransmetaがOEM用に作成したディストリビューション“Mobile Linux”を組み合わせて、タッチパネルによるWeb端末が作られる。このWebPadは、たとえばNetscapeNavigatorのプラグインソフトなどもきちんと動作するという。

●革新的テクノロジー

Crusoeに使われているテクノロジーは、主なものでCode Morphing SoftwareとLongRun power managementがある。

Code Morphing Softwareは、x86用の命令セットをRISCライクに並列処理を実行するCrusoeのVLIW（Very Long Instruction Word：超長命令語）エンジン用命令にダイナミックに変換するための仕組みだ。

Code Mophing softwareはプロセッサがブートしたときにロードされる。そして、CrusoeのVLIWエンジンと、BIOS、OS、アプリケーションすべてを隔離、x86命令をネイティブのVLIWに変換してVLIWエンジンに供給する。

いったんx86命令を変換するわけだから、そのオーバーヘッドに関する仕組みも備えている。キャッシュを用意し、いったん変換処理した命令がもう一度来ると、変換せずにキャッシュから命令が実行される。キャッシュに入る命令も、よく使われるものの優先順位が高くなっている。このキャッシュサイズは、起動時に設定できる。

●LongRun Power Management

従来のパワーマネジメントは、単純な言い方をすれば、フルパワーのモードとローパワーのモードの2種類しか存在しない。これでは、モバイル時のローパワーモードの時には、当然実行速度が落ちてしまう。しかしCrusoeの場合は、動作周波数を、使用するアプリケーションによって変化させる。あるアプリケーションが90%のプロセッサースピードしか要求しないとすると、Crusoeは10%スピードを減速し、10%の省電力を図るという。

のりぞうPresents

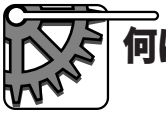
Linuxぽりぽりチューニング

文：のりぞう
Text：Norizoh

あなたのLinuxマシンをチューンしよう。といっても、ハードウェアのアップグレードではない。確かに、最近は高性能なCPUやハードディスクが信じられないほど安価になっている。マシンの処理速度を向上させることが目的なら、これらのパーツを購入するのが最も効果的で、かつ確実だ。

しかし、それでいいのか飽食日本。消費文化は必ず破綻をきたす。この問題に対し、のりぞうはソフトウェアによるソリューションをアピールしたい（相変わらず大げさ）。なにしろ、フリーソフトと頭を使えば、経済的な負担なしに、より便利な環境を手に入れたり、マシンの性能を引き出すことができるのだ（要するに貧乏なんだね）。

今回は、前半で一般的なソフトウェアのインストール方法をまじめに説明して、その後でちょいとおバカな（オマケにちょっと危険な）チューニングの話をしようと思う。どうか最後までおつきあいいただきたい。



何はともあれインストール

みなさんご存じのように、ソフトウェアはインストールしなければ使えない。Windowsでは、インストーラ付きのアーカイブが用意されていて、Setup.exeかなんかをダブルクリックすれば、あとはいくつかの質問に答えるだけで、インストーラがよしなにやってくれる。

●RPMパッケージのインストール

一方Linuxではどうだろうか。RPM（Red Hat Package Manager）を採用しているディストリビューションでは、X Window Systemで動作するgnorpmやglintを使えばGUI操作でパッケージのインストールができる。また、rootユーザーになり、rpmコマンドを使えばソフトウェアのインストール／アップグレードは簡単にできる。たとえば、次のようにするとhogeというパッケージのバージョン1.2.3をインストール／アップグレードインストールする。

```
# rpm -Uvh hoge-1.2.3-4.i386.rpm
```

しかし、実行ファイルの入ったバイナリRPMパッケージが用意されていても、標準Cライブラリ（libcやglibc）のバージョンが合わないなどの理由で、あなたの環境ではインストールできないかもしれない。このような場合は、187ページからの「賢く使うUNIX」を参照して、ソースRPMパッケージから自分の環境に合ったバイナリパッケージを作成しよう。バイナリパッケージを作るにはコンパイラのgccを始めとする開発環境をインストールしておく必要がある。

●tarボールによる配布

世界中すべてのソフトウェアがRPMパッケージとして提供されているわけではない。むしろRPMパッケージで配布されているものはごく少数なのだ。

では、どのような形で配布されているのだろうか。ほとんどの場合、ソースコードをtarというアーカイバで1つのファイルにまとめ、それをgzipやbzip2というツールで圧縮して配布されている。これらのファイルは、それぞれ

hoge-1.2.tar.gz、hoge-1.2.tar.bz2というようなファイル名をもち、俗にtar（ター）ボールと呼ばれている。また、実行ファイルをtarボールにして配布することもある。

Windowsでは、ソースコードでソフトウェアを配布するのはまれだが、LinuxやUNIXでは一般的だ。ひとくちに、LinuxやUNIXといっても非常に多くの種類や環境があるため、配布する人が個別にバイナリパッケージを作成するのは困難だ。そこで、ソースコードで配布し、ユーザーが自分の環境に合った実行ファイルを作るのが一般的になったのだ。もちろん、オープンソースへの流れに象徴されるように、それだけが理由ではないんだけど。

●tarボールの展開とインストール

さて、tarボールを用意したら早速展開しよう。tar.gzなら、次のようにすればファイルが展開される。

```
$ tar xvzf hoge-1.2.tar.gz
```

tar.bz2なら次のようにしよう。

```
$ bzip2 -dc  hoge-1.2.tar.bz2 | tar xvf -
```

多くのソフトウェアは、ディレクトリ付きのアーカイブになっているので、hoge-1.2のようなディレクトリが作られ、その中にソースコードやドキュメント、場合によっては実行ファイルが展開される。

さて、展開されたファイルをインストールするにはどうすればいいのか。実行ファイルとして配布されていれば、それを適切なディレクトリにコピーすればよいのだが、ソースコードで配布されている場合は、実行ファイルを作らなければならない。これをビルドするなんていう。ビルドするには開発環境が必要だ。さて、そのビルドなのだが、「どんなソフトもこうすればおっけー」という方法があるわけではない。世の中そんなに甘くはないのだ。比較的よくあるパターンはいくつかあるが、正しい方法を知るにはドキュメントを読むしかない。README、INSTALL、BUILDなどというファイルがあったら読んでみよう。インストール方法が書かれているはずだ。とは言ってもこれだけではナニなので、よくあるパターンを説明しておこう。

まず、configureというファイルがある場合、これを実行する。

```
$ ./configure
```

こうすればライブラリなどの環境を調べてMakefileを作ってくれる。また、Imakefileがある場合は、xmkmfを実行すればMakefileが作られる。Makefileというのは、コンパイラやリンカに対する指示書のようなものだ。

Makefileができたらmakeコマンドを実行すれば、コンパイルやリンク作業の後、実行ファイルが作られる。

```
$ make
```

プログラムの中には、最初からMakefileが用意されているものもあり、環境に合わせてオプションを与えたり、内容を書き換える必要があることもある。

実行ファイルができたら、作ったファイルを適切なディレクトリに配置すればインストールは終了だ。これもmakeコマンドによって行えることが多い。よくあるパターンは、rootになって、

```
# make install
```

とする方法だ。しつこいようだが、正しいインストール方法を知るには、ソフトウェアに付属するドキュメントを読む必要がある。やっぱり世の中は甘くないのだ。

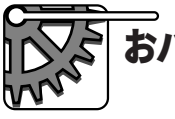
さて、なぜrootになるのかというと、ほとんどの場合、ファイルをコピーするディレクトリは、一般ユーザーには

画面1　freshmeat.netはLinuxのソフトを検索できる代表的なWebサイトだ（http://freshmeat.net/）。

書き込み権限がないからだ。実は、ここに1つ落とし穴がある。rootといえば、そのシステム内では全知全能の存在であり、できないことはない、すなわちシステムの破壊だってできるのだ。もし、Makefileの記述に間違いがあったら大変なことになる……かもしれない。makeコマンドには、実際の操作はしないで、どのような操作を行うかを表示するオプションとして-nが用意されているので、make installなどを行う前に操作内容を確認しておきたい。

```
# make -n install
```

ここまでで、ほとんどのソフトウェアはインストールできるようになったはずだ（え？ debパッケージ…… 今回は誌面の都合上パス _0_）。もう恐いものはない、Linuxプログラムがぎっしり詰まったWebサイト、freshmeat.net（**画面1**）が宝の山に見えるはずだ。



おバカなチューンとは

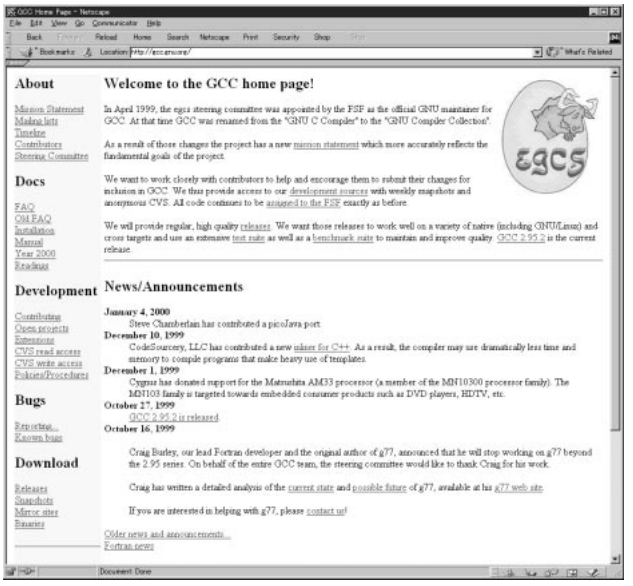
さて、いよいよチューニングの話に突入だ。プログラミングの知識がないからといってビビることはない、のりぞうもプログラムは作れない（えっへん）。

先ほどのビルド作業をした人なら、makeを実行したときにgccというコマンドが呼び出されていたのを目にしただろう。これは、GNU C／C++ コンパイラだ（厳密にいうと現在ではGNU Compiler Collectionという）。他の多くのコンパイラ同様、gccには最適化オプションというのがある。最適化とは、コンパイラがオブジェクトファイルを作るときにコードの無駄を省き、実行速度が速くなるようにしたり、プログラムサイズを小さくしたりする機能だ。

●gcc／egcsの最適化オプション

gccの最適化オプションは、-Oに1桁の数字をつけて指定する。数字が大きいほうが最適化の度合いも強くなる。ただし、あまり強い最適化を行うとできあがった実行ファイルが正しく動かなくなることもある。まさに諸刃の剣なのだ。

また、最近のディストリビューションに含まれるgccは、旧来からあるgcc 2.7.2ではなく、さらに高速なコードを出力するよう改良されたegcs（Experimental/Enhanced Gnu Compiler System）であることが多い。ところが、1999年からegcsプロジェクトが正式にgccのメンテナーとなり、最新版のgcc 2.95はegcsのコードを採用してgccと統合したものとなっているから話はややこしい（**画面2**）。実際に利用しているgccの素性を知るためには、gcc -vと入力する。



画面2 gccのオフィシャルWebサイトで最新版のgcc 2.95が公開されている（http://gcc.gnu.org/）。

```
$ gcc -v
Reading specs from /usr/lib/gcc-lib/i386-redhat-
linux/egcs-2.91.66/specs
gcc version egcs-2.91.66 19990314/Linux (egcs-1.1.2
release)
```

これはLASER5 Linux 6.0での実行結果だが、gccとして起動されるのはegcs-2.91.66（egcs-1.1.2 release）だとわかる。

egcsやgcc 2.95はCPUに合わせた最適化も行うことができる。たとえば、Pentium用の-mpentium、Pentium Pro用の-mpentiumpro、AMD K6シリーズ用の-mk6といったオプションが用意されている。さらに、-march=CPU指定 というオプションを使えば、指定したCPU固有の命令も駆使して最適化を行う（そうして作った実行ファイルは他のCPUでは動かなくなるけど）。

これで本稿でいう“チューン”の意味がおわかりだろう。そう、これらの最適化機能を使ってプログラムを再構築するのだ。

●さらに強力なgcc系コンパイラ

これだけではつまらないので、コンパイラをもう2つ紹介しよう。その1つは、egcsをベースにPentiumファミリ

画面3　pgccのWebサイトでは、Linux以外のOSに対応したpgccもgetできる（http://www.goof.com/pcg/）。

ー向けに強力な最適化を行うpgcc（Pentium-optimized gcc）だ（**画面3**）。最新版は、gcc 2.95をベースとする2.95.3だ。

glibc 2.0.6以上に対応するバイナリをhttp://www.goof.com/pcg/binaries-linux.htmlから、tarボールの形で入手できる。インストールはrootになって、

```
# tar xvpzf <tar-file.tar.gz> -C /
```

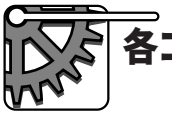
で行う。ただ、このバイナリは、のりぞうのLinux専用Pentiumマシンでは動作せず、ゲロゲロとcoreを吐きまくった。編集部のCeleronマシンでは動くので悔しさ100万倍だが、CPUの種類に起因するものかはわからなかった。

もし動作しない場合は、http://gcc.gnu.org/gcc-2.95/gcc-2.95.2.htmlを見て、最寄りのサイトからgcc 2.95.2のソースを入手してから、pgcc 2.95.3にするためのパッチ（差分）をあて、ビルドする必要がある。パッチファイルは、http://www.goof.com/pcg/source.htmlからダウンロードできる。また、パッチのあて方もこのページに書かれている。パッチをあてたらドキュメントをよく読んでビルドしよう。ね、ドキュメントは大事でしょ。

最後に紹介するのは、先頃Red Hatが買収したCygnus Solutionsの販売する「GNUPro Dev Kit for Linux v2.0」だ（**写真1**）。これは、egcsをベースにPentiumファミリー向けの最適化を強化したコンパイラとデバッガをパックにした開発キットで、79USドル販売されている。のりぞうは「1回だけ飲みに行かなければ買える！」と思

写真1　GNUPro Dev Kit for Linux v2.0は、低価格の開発入門キットだ（ただし日本から注文すると送料が高い）。

って買ったのだ（プログラムも書けないくせに）。ちなみに、送料が4000円ほどかかって激しく後悔したのはここだけのナイショだ。



各コンパイラの実力

とっても長い前振りだったが、いよいよチューンの効果のほどを見てみよう。

以下すべてのテストには、Celeron 300MHzのマシンを使用した。

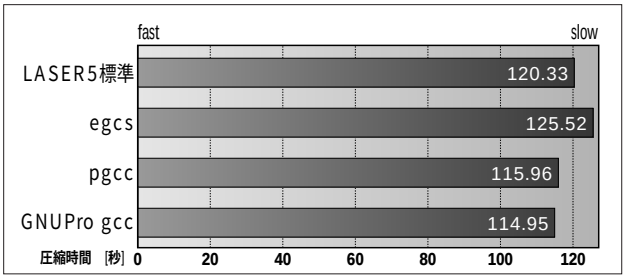
●gzip圧縮のスピード対決

最初のテストは、gzipでファイルを圧縮するのにかかる時間を比べるというものだ。LASER5 Linux 6.0に標準で含まれるgzip 1.2.4と、次の3種類のコンパイラを使ってチューンしたgzip 1.2.4でテストした。

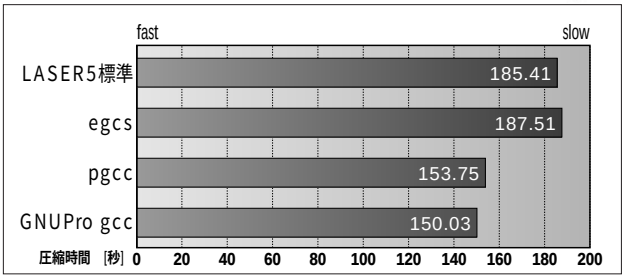
- egcs-2.91.66（以下egcs）
- pgcc-2.95.3（以下pgcc）
- gcc 2.9-codefusion-990706p1（以下GNUPro gcc）

LASER5 Linux 6.0標準のgzipはegcsでコンパイルされているが、コンパイルオプションは不明だ。これに対し、上記3つのコンパイラでgzipをビルドする際には、MakefileのCFLAGSに-mpentiumpro -march=pentiumpro -O6をセットしてコンパイルオプションを指定した（Celeron専用のオプションはないので、CPUの種別はpentiumproを指定する）。

こうしてできた4つのgzipで、Linuxカーネル2.2.14のアーカイブlinux-2.2.14.tar（67747840バイト）を圧縮す



グラフ1　gzipでの圧縮にかかった時間。



グラフ2　bzip2での圧縮にかかった時間。

る時間をbashの組み込みコマンドtimeで測定した。

```
$ time gzip -9 linux-2.2.14.tar
```

-9は圧縮率を最大にするgzipのオプションだ。こうして得られた結果は**グラフ1**のようになった。

うーむ、egcsが遅い。試しにオプションの-O6を-Oにするとと LASER5標準のgzipと同じくらいの速さになった。この環境のgzip圧縮では、egcsの-O6オプティマイズはかえって足を引っ張っているのだ。pgccと商用製品のGNUPro gccはさすがに速い。送料の件でへこんでいたのりぞうはちょっとウレシイ。

●bzip2圧縮のスピード対決

次はgzipの代わりにbzip2という圧縮プログラムで同様のテストを行った。bzip2は、gzipに比べて低速だが高い圧縮率を実現している。gzipとはまったく異なるアルゴリズムで圧縮を行うのでegcsも名誉を挽回するかもしれない。テストに使ったbzip2のバージョンは、LASER5 Linux 6.0に含まれる0.9.0c。最新版は0.9.5dなのでちょっと古い。時間計測はgzipの時と同じtimeコマンドで行った。

```
$ time bzip2 -9 linux-2.2.14.tar
```

さて、結果はというと**グラフ2**のようになった。gzipのときとほぼ同じ傾向が見られる。egcsは今度も成績が悪い。pgccとGNUPro gccはegcsに大差を付け、爆速の結果を叩き出している。pgccとの差はわずかだが、今回も一番速かったのはGNUPro gccだ。のりぞうは喜んでいる。

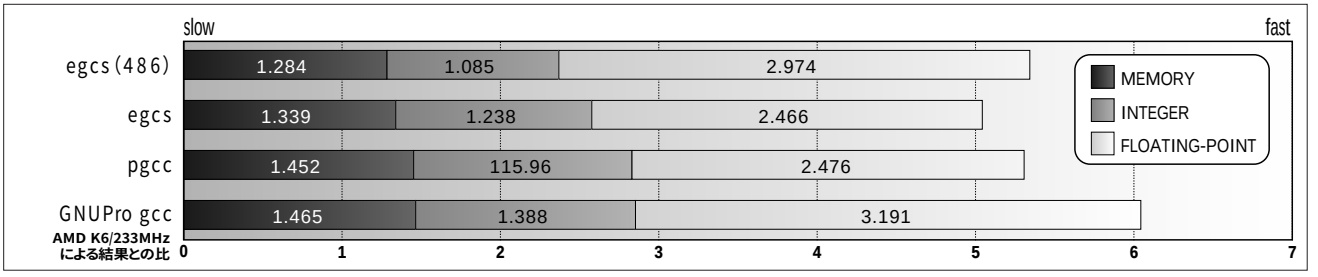
●nbenchによる各種アルゴリズムルーチンのベンチ

今度は趣向を変えて、nbench（http://members.xoom.com/Gavrilov_Con/nbench/index.html）というベンチマークプログラムでテストしてみた。

このベンチマークテストは、米国BYTE MagazineのBYTEmarkベンチマークをLinuxやUNIXに移植したものだ。数値ソート、文字列ソート、ビット操作、ハフマン圧縮など、一般的によく使われるアルゴリズム10種類を組み合わせてCPUの能力を計る。各アルゴリズムごとに詳細な結果が出力されるが、メモリパフォーマンス（MEMORY）、整数演算（INTEGER）、浮動小数点演算（FLOATING-POINT）のそれぞれについてAMD K6 233MHzを1としたときの比率で結果を見ることもできるので、ここではこの数値を比べてみよう。

CPUの能力計ってどーするという突っ込みを入れたくなったアナタ、ちょっと待った。nbenchは、ソースコードで配布されており、ユーザーが各自でコンパイルしてから使うのだ。そして、コンパイラの能力によって結果が変わってくる。したがって、同じマシンを使っていれば、コンパイラのパフォーマンスを計ることもできるわけだ。

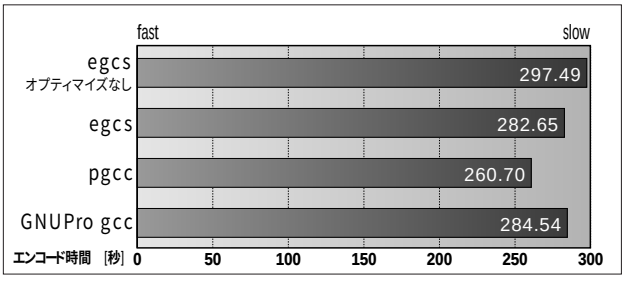
LASER5 Linux 6.0にはnbenchが含まれないので、今回はegcsを使い、-m486オプションでコンパイルした実行ファイルをリファレンスとした。残りの3つは-m486の代わりに、-mpentiumpro -march=pentiumproをオプショ



グラフ3　nbenchの結果。

ンとした。

テストの結果は**グラフ3**に示したとおりだ。egcsでPentium Proオプティマイズをときの結果が芳しくないのは相変わらずだ。また、pgccは浮動小数点演算が苦手だという結果が出ている。nbenchによる勝負では、GNUPro gccの圧勝だ。のりぞうは有頂天になっている。ほっほっほ〜。



グラフ4　LAMEによるMP3エンコードにかかった時間。

●MP3のエンコードスピード対決

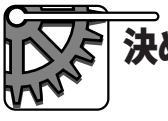
最後にMP3のエンコード速度を比較した。エンコーダソフトウェアは、LAME（http://www.sulaco.org/mp3/）を選んだ。LAMEはすべてCで書かれているため、エンコードの速さはコンパイラの能力にかかっている。

MP3エンコードする音楽データは、ANN LEWISの「恋のブギ・ウギ・トレイン（groove that soul mix）」（7分33秒）を使用した。

このテストでも、各コンパイラで-mpentiumpro -march=pentiumpro -O6オプション付きでコンパイルした実行ファイルと、リファレンスのために、オプティマイズオプションを付けずにコンパイルした時の実行ファイルによるエンコード時間を計測した。

結果は**グラフ4**のようになった。今まで行った3つのテストとはずいぶん違った結果になった。ここまでマイナス方向へばかり働いていたegcsのオプティマイズがまともに利いているではないか。プログラムの内容によってオプティマイズによる効果が変わるのだ。これはGNUpro gccにもあてはまる。ここまで無敗を誇っていたGNUPro gccもLAMEのコーディング、あるいはアルゴリズムは苦手だったようだ。

pgccがダントツの1着、GNUPro gccの完敗だ。のりぞうはへこんだ。ぺこん。



決めるのは あなた自身

4つのテストの結果を見ておわかりのように、コンパイラによって、得手不得手があるものの、pgccやGNU Pro gccは、egcsに比べるとおおむね良い結果を出すようだ。のりぞうのようにプログラムを作らない（正しくは、作れない）のに商用コンパイラを購入するのはお勧めしないが、pgccにトライしてみるのはいいかもしれない。

ただし、うまい話に裏があるのは世の常で、最新版のpgccはgcc 2.95をベースとしていることもあり、ちょっと前までLinuxの標準だったgcc 2.7.2や、egcsと100%の互換性があるわけではない。たとえば、Linuxカーネル2.2.14のリリースノートをみるとgcc 2.95でビルドするためのパッチは取り込まれているが、gcc 2.7.2やegcs 1.1.2に比べ十分なテストが済んでいないので気を付けるようにと記載されている。gcc 2.95どころか、pgcc 2.95.3でビルドするなんて無謀だといえよう。

とかいいつつ、のりぞうがサーバにしているマシンのLinuxカーネルはGNUPro gccでビルドしている。カーネルの再構築を行うときに、make xconfigなどでさまざまな設定をする。この中でCPUの種類を選ぶことができるが、これはコンパイラの最適化オプションを変えるわけではない。ここでどのCPUを選んでも最適化オプションは-m486が指定される。最適化オプションを変えたいときは、linux/arch/i386/Makefileを直接書き換える。ただし、その結果Linuxが動かなくなっても誰にも文句は言えないのだ（これだけ焚きつけておいていまさらなにを…）。最初はtarボールからgzipやbzip2のような小物をpgccでビルドしてみてはどうだろうか。

●おまけ：rpmコマンドでの最適化

最後に、小技を1つ。rpmコマンドを使ってソースRPMパッケージからからバイナリRPMパッケージをリビルドする場合、最適化などのコンパイルオプションを指定する方法だ。/usr/lib/rpmrc、あるいは/usr/lib/rpm/rpmrcというファイルの中に“optflags: i386”で始まる行があるはずだ。LASER5 Linux 6.0では、/usr/lib/rpm/rpmrcに、

```
optflags: i386 -O2 -m486 -fno-strength-reduce
```

という行がある。i386の次の部分、この例でいえば-O2からがコンパイルオプションとしてgccに渡されるのだ。従って、このファイルを書き換えれば簡単に最適化済みバイナリパッケージが作れるというわけだ。

冒頭でも書いたように、こんなことしなくても、CPUやハードディスクを換えたほうが確実に速くなるんだけどね。

落ちこぼれなしのスーパー連載

Emacsはじめました

第1回　超入門編

Emacsを使い出すと、やめられないという。彼の国ではメールやニュースもこれひとつでOK、あれやこれやのアプリケーションウィンドウが騒々しく開く喧燥をはなれ、操作方法の統べられた美しい環境で暮らせるのだとか。そうか、そんなに幸せになれるなら出かけてみよう、Emacsワールドへ。行く手には拡張の湖が待っている。

Illustration : Manami Kato

文：佐々木太良
Text : Taroh Sasaki

みなさま、またお目にかかります。『viはじめました』が終わったあと、多くの方から「次はEmacsでしょう?」との声をいただき、UNIX二大エディタのもう一方を引き続き連載することになりました。

さて、viとEmacs、エディタはどっちかでいいんじゃないの?　という声もありそうです。実際、フリー系UNIXを使い始めたばかりの方は、どちらを使おうかと迷ったのではないでしょうか。しかし、文章書きには日本語との相性がよいEmacs、プログラムやシステム設定ファイルにはviとケースバイケースで使い分けている人も多いようです。

Emacsの特徴

Emacsの特徴を簡単に紹介しましょう。

- **カスタマイズの余地が大きく、初心者から上級者まで幅広いユーザーが習得度に応じて快適に使える。**
- **基本用途は、文章の執筆やプログラム作成などだが、メールやニュースなどほかのアプリケーションと組み合わせていろいろなことができる。**
- **洗練された日本語入力。フリー系UNIXでは日本語を取り扱うためにはほとんど不可欠。**
- **ワープロではなくエディタだけど、WYSIWYGライクな装飾で仕上がりを表示させることも可能。**
- **UNIXだけでなく、Windows環境でも使える。**

■Emacsは初心者向きか?

viに比べてEmacsは「やさしい」とよくいわれます。X Window System上であればコマンドを知らなくてもカーソルキーとマウスで操作できるので、まるでWindowsの「メモ帳」と同じように見えます。それでもEmacsは「初心者にはムリ」と多くの人が信じているのは、パニックに陥りやすい誤操作もあるためでしょう。安心してください、この連載ではまっさきにパニックを回避する方法を伝授します。

しかし、文章をばしばし書く人にとって、カーソルキーはホームポジションから遠くて使いにくいし、マウスに持ち替えるなどもってのほか。達人にとって、Emacsはすべての操作がキーボードでできるうえ、Emacs本来のキー操作はもちろん、あらたな操作を好きなキーで使えるようにカスタマイズできるのも魅力です。

■カスタマイズと「なんでもツール」

誰しも使う道具は自分好みに育てたいもの。達人ほどこの傾向は強くなります。Emacsのカスタマイズ性は、通常考えられる範疇をはるかに超えています。エディタ自体がEmacs Lisp言語でプログラムされており、外部のアプリケーションプログラムと連携して、組み版、マニュアルの執筆や閲覧、メール／ニュースの読み書き、チャット、ゲーム、シェルの使用まで、驚くようなことができるのです。

筆者はふだん、メールの読み書きにEmacsを使ってい

図1　Emacs起動時の画面
Laser5 Linux 6.0のemacs起動時の画面。Emacs20.3.1＋cannaの組み合わせを使用している

ます。メールの文面とファイルのやりとりが簡単でテキストの使い回しがしやすく、本文の編集機能が豊富で、日本語入力の親和性もよいので、他のメールツールを使う気にならないくらいです。

■日本語入力

日本語入力は、Linuxユーザーにとって一番痛い点かもしれません。kinput2など、入力メソッド（かな漢字変換）と日本語を扱えるアプリケーションソフトの橋渡しを行うものもありますが、Windowsのアプリケーションに慣れたユーザーから見ると、安定度がいまひとつだったりします。

UNIXの社会では流通する漢字コードが統一されておらず、アプリケーションがJIS、シフトJIS、EUCのどんな文字コードでも平等に扱える美しさを保つのは、たやすいことではありません。しかし、Emacs上にある数多くのマクロを活用して、日本語の表示と入力の窓口をEmacsにすれば、Linuxでも不自由なく日本語が取り扱えます。

EmacsとMule

もともと米国で開発されたEmacsは、日本語を始めとするマルチバイトコードの扱いが不得意でした。MuleはEmacsを多国語対応にしたバージョンです。Muleの改良はEmacsにも採り入れられ、今ではcannaなどと組み合わせてEmacsでも日本語が入力できます。そんなわけで、Linuxのディストリビューションによって Emacsが入っていたり、muleが入っていたりします。手元の環境でemacsコマンドが見つからなかったら、muleコマンドで試してください。今回は、EmacsをX Window System上で使うことにします。

超入門

ではさっそく、Emacsを起動してみましょう。コンソールのコマンドラインから“emacs”と入力すると、X Window System上で**図1**のウィンドウが開きます。

なにか入力してみましょう。表示されているメッセージは、文字をタイプすると（あるいは一定時間経過すると）消えてしまいます。消えたメッセージにはじつは重要なことが書いてあったので、見のがしたら再起動……、いえいえ**図1**を見てください。

さて使い方ですが、キーボードを叩けば文字がカーソル位置に挿入され、マウスで好きな位置をクリックすればカーソルはそこに移動します。ほかのGUI対応のエディタを使ったことがあれば、マウスでなぞった場所が反転し、メニュー［Edit］から［Copy］や［Cut］を選べば（以降［Edit］－［Copy］のように書きます）コピーまたはカットされ、その内容は［Edit］－［Paste Most Recent］でペーストされることが見てとれます。なんと、これではWindowsの「メモ帳」と同じではありませんか（笑）。この

Column

X Windows SystemとEmacs

Emacsは、X Window Systemでなくても使えます。文字端末（telnetで遠隔利用している場合など）からEmacsを立ち上げると、メニューバーと右側のスクロールバーが表示されないほかは同じように使えます……と言いたいところですが、今回の超入門編ではマウス操作を利用するので、「Xが立ち上がらないんだよ～」という方は、まずX Window Systemのセットアップをお勧めします。

さてこのX上のEmacsですが、Emacs19の場合はさほどGUI化されていませんが、Emacs20以降は積極的にXの特徴を活かし、GUIならではの機能がたくさん盛り込まれています。たとえばXEmacs上のWebブラウザなどは、Webページに指定されている文字装飾に応じてフォントが変わったり、写真や図も表示されます（エディタの画面ですよ、これは）。もちろん、すごく重いです。

ともあれ、この記事ではキー操作で使えるようにしていきます。

まま［Files］－［Save Buffer As...］で名前を付けてセーブし、［Files］－［Exit Emacs］で終了できれば、この連載はいらないではありませんか（泣）。

Emacsのおやくそく

しかし、とりあえずはこれでよいとしても、この怪物エディタの機能をおいしく使うには、キー操作をある程度使いこなさないといけません。マウスの操作だけで使えるEmacsの機能は限られたものです。まずは基本的なキー操作や用語のお約束からみていきましょう。

■メタキーとコントロールキー

Emacsでは、メタキーとコントロールキーを多用します。たとえば「M-x」と書いてあったら「メタX」と読み、「C-c」と書いてあったら「コントロールC」と読みます。これらは、シフトキーと同様に「メタキーやコントロールキーを押しながら文字キーを押す」ことで別の文字コードを入力するものです。C-aは、

1. 〈Ctrl〉キーを押しつつ
2. 〈A〉を押して
3. 〈A〉を離したら
4. 〈Ctrl〉キーを離す

という操作です（〈Ctrl〉キーを押したまま〈A〉キーをちょんと押す）。メタキーも同様です。

「バカにするなよ、知ってらい」。あ、失礼しました。ですが、メタキーの入力動作で間違う人がよくいるのです。

メタキーはEmacsに特有ともいえる入力キーで、PCで動いているX Window Systemでは、通常〈Alt〉キーに割り当てられていることが多いようです。しかし、M-xを入力する際に〈Alt〉キーを小指で押しながら〈X〉キーを薬指で押そうとして指がつった人が続出し、社会問題になったため、現在は〈ESC〉キーで入力を代用する人が多いようです（うそ。端末によっては完全にメタキーがない場合があるからです）。〈ESC〉キーは通常の文字キーですので、〈ESC〉キーを使ってM-xを入力するためには、

1. 〈ESC〉キーを押して
2. 〈ESC〉キーを離したら
3. 〈x〉キーを押して
4. 〈x〉キーを離す

という2文字の当たり前の入力手順になります。メタキーと混同して〈ESC〉キーを押したままにすると、〈ESC〉キーを連続して押したこと（M-ESC）になり、いきなりわけのわからない状態になって初心者はパニックしてしまうことがあります。

パニック！

さてたとえば、勢いよく（長い間）〈ESC〉キーを押してESC文字が2文字入力されたとしましょう（〈ESC〉〈ESC〉と入力してこの状態にしてみてください）。画面下部に“ ESC ESC-"と表示されて、「ほらほら次を入力するんだよ」と急かされているのがわかります。

こんな状態になると、はじめての人は「わ〜っ、次どうすればいいんだろ〜」と驚くものですが、慌てず騒がず**C-g**を押してみてください。うまくいかない場合は、2度押してみましょう。C-gは何回押しても困ったことにはならないので、「あれっ?」と思ったらC-gを叩くのはEmacsの

Column

Emacs操作の表記

Emacsの操作上では、C-aやM-xのAやXに大文字と小文字の区別はありませんが、UNIXシェル流ではなぜか「^A」と大文字で表記し、Emacs流では「M-x」と小文字で表記する習慣があります。Emacsはメタ文字の大／小文字を同じように扱うだけで、じつはM-xとM-Xは別の入力文字になります。メタキーは「文字の上位1ビットを立てて入力する」キーです。そこでちょっといたずら。kterm（EUC表示）を開き、8ビット入力可能なシェルでメタキーを押したまま、“ :4!9LZB@NI ”（M-:、M-4、M-! ...）と入力するとあら不思議、日本語入力メソッドもないのに、次のように表示されます（笑）。

佐々木太良

このほか、Emacsでは特殊キーの次の表記にしばしばお目にかかるので、覚えておくとよいでしょう。

キー	他の入力法	Emacsでの表記
〈TAB〉	C-i	TAB
〈Enter〉	C-m	RET
〈SPACE〉		SPC
〈ESC〉	C-[	ESC
〈DEL〉		DEL
〈INS〉		INS

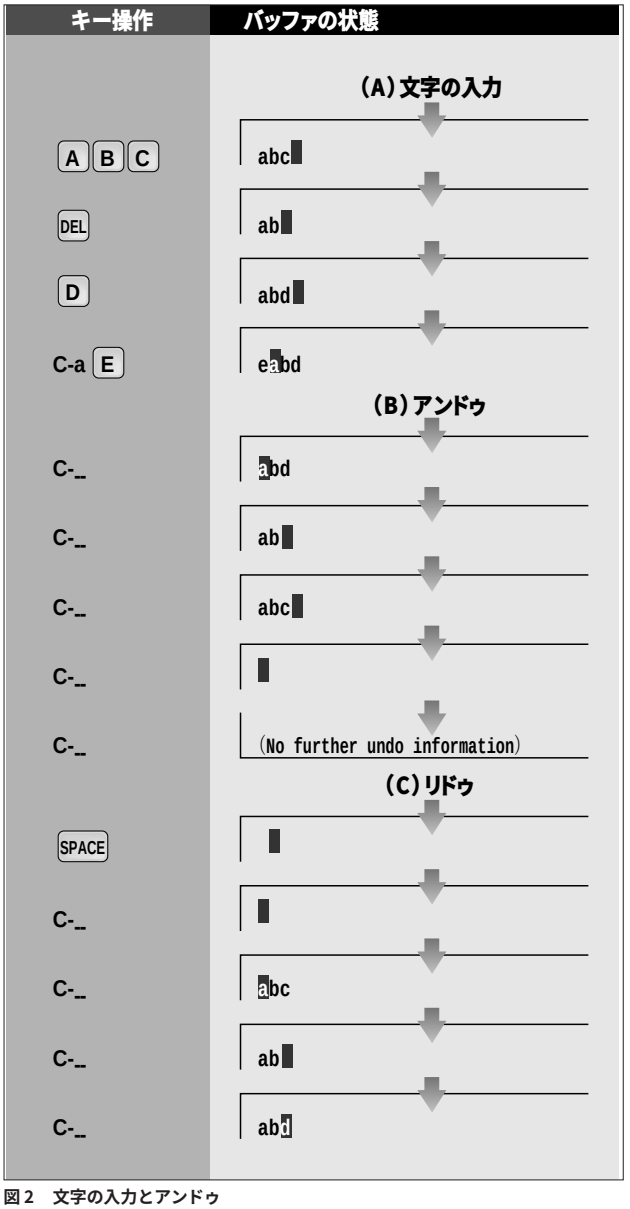


図2　文字の入力とアンドゥ

達人でも同じです（いや、達人ほど慌てず騒がずC-gを叩きます）。

■説明のとおりに動かない

ディストリビューションによっては、いろいろとくふうした.emacsという設定ファイルが用意してあって、操作結果がこの記事のとおりにならないことがあります。オリジナルの動作を試したいときにはホームディレクトリのこのファイルを削除すればよいのですが、いつでも元に戻せるように、.emacs.ORGなどのように別の名前に変えておくのがよいでしょう。このファイルについては、あらためて説明することにします。

■やりなおし

Emacsは原則的に、無限にアンドゥ（操作の取り消し）ができます。「しまった!」と思ったら、冷静に1ステップずつアンドゥを繰り返せば、かならず元に戻せます。

アンドゥは**C-_**ですが、ASCII配列のキーボードで〈Ctrl〉＋〈Shift〉＋〈_〉を入力するのは手が疲れるため（うそではありませんが、原理的にこの組み合わせを入力できない端末ソフトがある）、別に**C-x u**という2ストロークのアンドゥ操作も用意されています。かなりの文章を書き進めてから「あ、やっぱりやめた」などというときは、連続して（キーボードのオートリピートで）アンドゥすることもありますので、どちらかというと2ストロークのC-x uは不便だと筆者は感じます。

起動直後の状態から何か文字を入力して（さらにカット&ペーストなども試していればよりお楽しみいただけます）、アンドゥ操作を試してみてください（**図2-B**）。複数文字の入力、削除など、「えっ？　どうしてこれが1ステップなの?」というような操作が一括して取り消されることもあります。

「アンドゥしすぎちゃったよ〜」というときは、なにか入力してからさらに取り消せば、アンドゥの過程を逆にたどる（リドゥ）こともできます（**図2-C**）。

つまりEmacsはアンドゥ中の操作も「undoバッファ」というものに記録しているのです。いざというときにこれがあるとないとでは大違い。「しまった」ときの時間が節約できるだけでなく、二度と生まれないアイデアを復活できるかもしれません。

チュートリアル

Emacsには、操作方法を教えてくれるチュートリアル機能があります。これは**C-h t**で表示されますが、muleには日本語、ハングル語、タイ語のチュートリアルも用意されています。日本語の場合は**C-h T**とタイプし、言語の選択のところで“Japanese”と入力します（後で述べる補完機能が働くので、〈J〉〈SPACE〉〈Enter〉を順に押せばOKです）。

いくらチュートリアルがあっても、使い方がわからなければしょうがありません。それで、起動直後の画面に「あんた初心者ならC-h tやC-h Tでチュートリアル読めよ」と表示されるのです。英語のメッセージを見ると知らないふりをする人がいて困っちゃいますが、辞書を引いてでもメッセージは読む習慣をつけましょう。

■ミニバッファとモード行

もう一度起動時の画面を見てみましょう。下部の黒く反転している行は、モード行といって各種の情報が表示されることになっています（**図1**）。日本語入力が使える場合には現在の入力文字の種類、ファイルの漢字コードの種類、テキストを最後にファイルに書き込んだあと変更の有無（ダーティ／クリーン）、バッファ名、時刻やCPUの負荷、編集中のファイルの種類、行数、現在の表示が全体のどのあたりに位置するかのパーセント表示などがところ狭しと表示されます。

なかでも重要なのは、ダーティ／クリーンの表示です。Emacsはかなり安全にできていて、もし編集中にラップトップPCの電池が切れても30秒くらい前の状態に回復できます。しかし転ばぬ先の杖と申します。ここをちらっと見て“＊＊”（ダーティ）になっていたら、**C-x C-s**の2ストロークでセーブしてからトイレに立つ習慣にしましょう。けっして悲しい思いをしません。

さて、モード行の下は、ミニバッファといってEmacsからのメッセージが表示されます。これは、どんなときでも1行だけです。場合によっては、セーブするファイル名や検索する文字列など、Emacsへの入力にも使うことがあります。

下の狭い空間がミニバッファということは……上の広い空間はバッファと呼ばれるんです。じつは、Emacsではバッファとウィンドウとファイル名が同じものではなく、微妙な使い分けがあるのですが、これは次回から少しずつ触れていきます。

基本中の基本の編集コマンド

ここからの説明は、チュートリアルを終えた方には不要かもしれません。それに、これくらいは知っていないと、具合が悪い。編集長、大丈夫でしょうか？　え？　しないとダメ？　しくしく……。

■終了

まずはじめに終了の仕方です。そうでないと、続きを明日にすることもできませんね。**C-x C-c**を試してください。このとき、ダーティなバッファがあれば、セーブするかどうか聞いてきます（例外もある）。これに「n」と答えると、ほんとうによいのか確認されます（しつこいですね）。ここで「yes〈Enter〉」と答えないと終了できない仕掛けになっています。

■ファイルの名前を指定する

起動直後のミニバッファには＊scratch＊というバッファ名が表示されていました。アスタリスクに囲まれているバッファは一時的な用途に使うもので、意図的に保存しないと黙って捨てられてしまいます。そこで、Emacsを起動をしてなにか編集を開始するときは、まずfind fileを実行してください。［Files］－［Open file...］からも選べますし、メニューの後ろの説明からわかるとおり、キーボードから**C-x C-f**で実行できます。キー操作に慣れない最初のうちは、メニューを見てキー操作を見つけるのもひとつの方法ですね。ここで、ミニバッファにファイル名を指定するように指示されます。

```
Find file: ~/Genko/
```

ここには既存のファイル、これから作る新しいファイルのいずれでも指定できます（**コラム「補完を活用しよう」**をちょっと眺めてください）。中止したければ万能中止コマンドC-gを使います。

■ポイントと移動

Emacsでは他のエディタのカーソルに相当するものを、「ポイント」といいます。これは、じつは「文字と直前の文字のあいだにあるもの」を指しているものと思われます。

ポイントの移動には、**C-f**（右）、**C-b**（左）、**C-p**（上）、**C-n**（下）を使います。これらのコマンドは、forward、back、previous、nextにちなんだもので、けっして人間工学的に使いやすいとはいえないキー配置ですが、この基本コマンドをカスタマイズしてしまうと、他人（よそのマシン）のEmacsが使えなくなってしまうこともあって、筆者は指が覚え込んでしまいました。

このほかよく使うのは、行頭に移る**C-a**、行末に移る**C-e**、1画面下にスクロールする**C-v**、1画面上にスクロールする**M-v**です。筆者はメタキーはESC派ですが、C-vとM-vは組みで使うことが多いため、M-vにはほんとうのメタキー〈ALT〉を使います。

■文字の削除

もともとのEmacsでは、ポイントの前の文字を削除するには〈DEL〉、ポイントの後ろの文字を削除するには**C-d**を使います。ちょっと戸惑う人がいるかもしれません。通常のエディタは、〈BS〉（またはC-h）で前の文字を削除し、〈DEL〉で後ろの文字を削除します。ところが、C-hは

Emacsのデフォルトではヘルプ（help）機能を起動します。

C-hをバックスペースとして使いたければ、〈BS〉と〈DEL〉の機能を交換するマクロを利用するとよいでしょう。**M-x**に続けて“load-library〈Enter〉”（“M-x loa〈TAB〉l〈TAB〉〈Enter〉”でもOK）とタイプしてください。「Load library: 」と尋ねてくるので、“term/bobcat〈Enter〉”と答えます（このM-xで始まるコマンドは重要なので、次回にあらためて解説します）。

毎回これを設定するのは面倒なので、Emacsの初期設定ファイルを作ってそこに設定しておけばよいでしょう。

~/.emacsというファイルに、

```
(load "term/bobcat")
```

という内容を書いてみてください。次に起動したときにはこの設定が有効になるはずです。

■文字の挿入

これは文字をタイプするだけですので、特別なことはありません。でも、C-xとかC-gのようなコマンドを実行するためのキー操作文字をファイルに挿入するにはどうすればよいのでしょう？　そういうときには、その前に**C-q**をタイプします。C-q、C-gとタイプすると、ファイル中に“^G”という文字が挿入されたでしょう？

■ファイルの書き出しと読み込み

さて、Emacsでの編集作業が済んだら、**C-x C-s**をタイプしましょう。最初に指定したファイル名でセーブされます。

```
Wrote /home/Taroh/Genko/column.txt
```

ただし、クリーン（前回セーブしてから変更されていない）の場合は何も起きません。また、**C-x C-w**をタイプし、ミニバッファで別のファイル名を付けてセーブすることもできます（バッファの名前も変更され、以後のC-x C-sではその別名が使われます）。

さて次回は……

次回は、日本語の入力と検索を説明します。Emacsのエディタとしての機能がだいたい使えるようになったら、その先に広がる魑魅魍魎界へみなさまを誘う予定ですので、ご期待ください。

ではhappy hacking!

Column

補完を活用しよう

ミニバッファに入力されるコマンドやファイル名などの大部分は、補完（completion）機能で楽ができるようになっています。カレントディレクトリに以下のファイルがあったとしましょう。

```
% ls
19990214.ps    hoge.mid     src/
b.gif          hoge1.txt    tmp/
bigcity.eps    hoge2.txt
```

ここでC-x C-fをタイプするとミニバッファに“Find file: ~/”と表示されますが、〈SPACE〉をタイプするとウィンドウが分割され、右図のように＊Completions＊というバッファが開いて選択候補が表示されます（慌ててはいけません。これは中止したり必要がなくなると消えるバッファです）。

また、hのあと〈SPACE〉（または〈TAB〉かC-i）をタイプすると、hで始まるファイル名の候補が可能な限り補完され、下表のようにミニバッファの表示が変化していきます。

キー操作	ミニバッファの表示
C-x C-f	Find file: ~/
〈h〉	Find file: ~/h
〈SPACE〉	Find file: ~/hoge
〈1〉	Find file: ~/hoge1
〈SPACE〉	Find file: ~/hoge1.txt
〈Enter〉	（確定）

〈SPACE〉は拡張子の切れ目まで、〈TAB〉（C-i）は可能な限り長く補完します。〈TAB〉での補完は、tcshなどの賢いシェルやgnuplotなど（GNUが作っているコマンドライン入力ライブラリを使ったツール）でも利用できます。

＊Completions＊バッファ

```
In this buffer, type RET to select the completion near point.
Possible completions are:
#me#           ../
./             19990214.ps
b.gif          bigcity.eps
hoge.mid       hoge1.txt
hoge2.txt      src/
tmp/
J_:-----Mule: *Completions*    (Completion List)--L1--Top-
```

Linux日記

第6回　ファイルシステム（前編）

Linuxは、UNIXの作法に従い、ディレクトリとファイルからなる階層化されたファイルシステムを採用しています。今回からは、この階層化ファイルシステムがどのような仕組みで実現されているのかを説明します。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

連載のタイトルがLinux日記なのに、内容がほとんど日記らしくないなぁと思っていたのだが、編集部もそのことに気付いたらしい。よく言えばUNIXオペレーティングシステム概説、実の所はUNIX雑談である。タイトルを変えたほうがいいかもしれない。まぁ、少しは日記的なことも書かねばと思う。

ここしばらく、地球外知的生命体探査プロジェクトSETI@home（http://setiathome.ssl.berkeley.edu/）の数字を上げることを頑張っていた（実際に頑張っていたのはプロセッサだが）。これはひたすら浮動小数点演算指向の処理なので、ふだんは動かしていないマシンも稼働させ、ひたすら計算した。とはいっても、速いマシンはあまり持っておらず、最速がこの連載のためと称して買ったPentium III 500MHz、次に速いのがVAIOノートという状態である。それでも7台のマシンを動かし、1日6個弱程度の数字にはなった。寒い季節だからできる構成である。とりあえず暖房の電気代だと思えば、こんだけ動かしてもどうにかなるかなと思ったのだ。実際、暖房を入れていない部屋より、仕事部屋は数度温度が高くなっている。夏場だと、これに加えてエアコンの電気代が発生するので、ちと恐ろしくてできない。

しかし、問題が発生した。計算データを供給し、結果を受理するサーバにつながらないという事態に陥ったのだ。これは思った以上に恐ろしいことだった。接続に失敗したクライアントが、接続のリトライを繰り返し、ISDNルータがとんでもない回数の接続を行っていたのだ。今は遅いマシンを止めて、ちょっとペースを落としている。

さて、この連載とは別の仕事の絡みで、わが家のマシン構成を大幅に変更することになった。Windows 2000のサーバとワークステーションを動かすことになったのである。いろいろやりくりを考えた結果、編集部からの借り物の実験マシンをわが家のメインNTサーバにすることにした。どうせ今さら編集部も返せとは言わないだろう。

Column

SETI@home

SETIというのは、Searching for Extraterrestrial Intelligenceというプロジェクトで、電波望遠鏡で全天観測したデータを解析して、文明的な信号の兆しがないかを調べるというものだ。

これには莫大なCPUパワーが必要になるが、それをインターネット上で実現したのがこのプロジェクトである。サーバ上からデータブロックをダウンロードし、その解析結果をサーバに送り返すという仕組みで動く（プロジェクトの言うところでは）世界最大のスーパーコンピュータシステムである。本原稿を執筆している時点で、参加者総数は150万人、総CPUタイムは12万年である。詳しくはhttp://www.vacia.is.tohoku.ac.jp/~s-yamane/articles/setiathome/home_japanese.html（日本語版）を見てほしい。

Column

借り物

筆者のような仕事をしていると、実験などでどうしても複数のマシンが必要になる。そのため、世間平均のユーザーよりも多くのマシンを持っているのだが、それでもネットワークなどの原稿を書く時など、マシンが足りなくなる。「これやるにはマシンが足りないんだよぉ」などとグズグズ言うと、結構編集部がマシンを貸してくれるのである。このPentium Pro×2マシンも借り物であるが、仕事が終わった後も返せともいわれないので使っている次第である。

何を隠そう、わが家のメール／DNSサーバのBSD/OSは、今は亡きSuper ASCII編集部から8年くらい前に借りた486DX 33MHzマシンを自腹で拡張したものである。これも今さら返せとは言われないだろう。お払い箱にしたときは返すつもりだが。

このマシンは、Pentium PRO 150MHzという今となってはどうというスペックではないのだが、わが家で唯一デュアルCPUマシンなのである。SETIの計算も、速くはないものの、画面表示をしても遅くならないという優れものであった。サーバといえば24時間稼働が原則だが、家の場合、ユーザーはほとんど筆者だけで、たまに奥さんが家計簿のExcelファイルをサーバ上にバックアップするだけなので、止めようと思えばいつでも止められる。そこで、このNT サーバをデュアルブート環境にして、Linuxもインストールした。今までLinuxが動いていたマシンは、Windows 2000 サーバになった。

というわけで、デュアルプロセッサでLinuxが動くようにはなった。ストーリー的には、前回までのプロセスの話に続いて、マルチプロセッサ環境、実行スレッドといった話の展開になるのが自然である。しかし残念ながら、原稿を書けるほどにはいじっていないので、この話は先送りである。

さて、それでは今回の原稿はどうしたものかと考えたのだが、ファイルシステムなんてどうだろう。プロセスに続いてファイルシステム。なんかUNIXのエッセンスを正しくトレースしているようではないか。

■ファイルシステム

UNIXに限らず、オペレーティングシステムの重要な仕事の1つに、ファイルシステムの実装がある。当初、UNIXのファイルシステムは、直接接続されていたローカルディスクしかサポートしていなかった。これをローカルファイルシステムという。その後（1980年代中頃くらいだったと思う）、ネットワークを介してリモートマシン上のファイルシステムにもアクセスできるようになった。これがリモートファイルシステム、あるいはネットワークファイルシステムと呼ばれるものだ。本稿では、ローカルファイルシステムに対して総称的にリモートファイルシステムという用語を使っていく。

オペレーティングシステムは、ローカルファイルシステムを拡張する形で、リモートファイルシステムのサポートを提供している。あるマシン上のプログラムは、ローカルファイルシステムに対するのとまったく同じシステムコールを使って、リモートファイルシステムにアクセスすることができる。つまり、通常のプログラムからは、ローカルファイルシステムもリモートファイルシステムも完全に等価なのである。違いはすべてオペレーティングシステムが吸収してくれる。

リモートファイルシステムについて

Column

NFSとRFS

現在UNIXで主流になっているリモートファイルシステムは、NFS（Network File System）である。これはSun Microsystems社がBSDベースのUNIX上に実装したものである。これとは別に、ATTがSystem V上に実装したRFS（Remote File System）というものもあった。NFSはステートレス（状態情報をサーバとクライアントの間でやり取りしない）のに対して、RFSはステートフル（サーバとクライアントの間で状態情報をやり取りする）で、信頼性はRFSのほうが高いと言われていたが、実際にはNFSの簡易性が受け入れられ今日に至っている。

ディスクレスワークステーション

ディスクが安価になった今では、ファイル共有は、ホームディレクトリやプロジェクトディレクトリなど、まさに複数のマシンからアクセスしたいファイルのために使っているが、ディスクが高価だった時代には、それとは別にさまざまな使い方があった。

たとえば、ほとんど更新されない/usrパーティションなどを共有することで、ディスク容量を節約するといった使い方があった。これをさらに進めたのがディスクレスワークステーションである。自身ではローカルディスクを持たず、ファイルシステムすべてをサーバに依存するというシステムである。ディスクレスワークステーションがどのように自身をブートさせるかというのは、興味深いところだ。暇だったら考えてみるといいだろう。今の一般的なPCでは不可能だが、ROM BIOSがちょっと頑張れば、さほど難しいものではない。実際、ROM BIOSを搭載できるネットワークカードも多数ある。

はまたいつか説明するとして、今回は、ローカルファイルシステムの仕組みについて見てみよう。

■ディレクトリとi-node

ディスク上の個々のセクタを、ファイルとして扱うためにはどうすればいいか。UNIXに限らず、ディスク上の領域をファイルという形でアクセスするために、いくつかの管理情報が必要になる。

まずは、ディスク上のファイル名の一覧を収めた表が必要だ。UNIXの場合、ディレクトリがこのファイル名一覧表になる。UNIXのファイルシステムは階層化されているので、ファイル名は多数のディレクトリに分散して収められている。

次いで、ディレクトリ中に記録されている個々のファイルが、ディスク上のどのブロック（連続した数個のセクタからなる管理単位）から構成されているかを示す位置情報（インデックス情報）が必要である。また、ファイルの属性、長さ、タイムスタンプ、オーナー、パーミッションといった付帯的な情報も必要だ。UNIXは、この情報をi-node（インデックスノード）というデータ構造に収めている。各ファイルにi-nodeが1つ割り当てられ、ファイルの内容がディスク上のどこにあるか、ファイルの各種属性が識別される。i-nodeテーブルは、ファイルシステムを構成するパーティションの特定の位置に置かれている。

ディレクトリ中の個々のファイル名のエントリは、ファイル名とi-node番号を収めている。そしてディレクトリとi-nodeを組み合わせることで、ファイル名が指定された時に、それがディスクのどこにあるかがi-nodeを介してわかり、それを読み込むことができる。また、ディレクトリやi-nodeの情報を正しく更新しながらディスク上にデータを書き込めば、ファイルの作成や更新を行うことができる。

要するに、UNIXでは、ファイルアクセスは、ディレクトリ→ファイル名→i-node→ディスク上のブロックという形になるわけだ（**図1**）。

UNIXのファイルシステムは階層化されており、多数のディレクトリが木構造に配置されている。これを実現するための構造は、意外なほど簡単だ。lsコマンドを実行すると、ファイルとディレクトリが一緒に表示される。データやプログラムの実行イメージを収めた通常のファイル（このようなファイル、つまり、ディレクトリでもスペシャルファイルでもシンボリックリンクでもないファイルのことをレギュラーファイルという）と比べると、ディレクトリは特殊なものに思えるが、実はそんなに複雑なものではない。ディレクトリはファイルの一種であり、その内容は、そのディレクトリに含まれるファイル名とi-node番号の一覧である（もちろん、その内容をcatやviで見ようとしても、エラーとなってオープンできないが）。あるディレクトリ中にサブディレクトリを表現するディレクトリファイルを作成することで、木構造ができあがるのである。これで、あ

Column

i-nodeの数

i-nodeテーブルは、mkfsによってファイルシステムが構築される時に作成される。i-nodeの数は、ファイルシステムのサイズに基づいて自動的に決定される。この計算は、UNIXの平均的なファイルサイズなどに基づいて行われ、普通はこれで困ることはない。

しかし、場合によっては、i-nodeが不足することもある。たとえば数百バイト程度の小さなファイルでファイルシステムを埋めつくすと、ディスク領域を使い切る前にi-nodeを使い切ってしまうことがある。この場合、既存のファイルを大きくすることはできるが、新規のファイルを作れなくなってしまう。

かつて、ニュースシステムを動かすとi-nodeが足りなくなるという事態がよく発生した。ニュースシステムが、個々のアーティクルを1つのファイルとしてスプールしたためである。このような場合、管理者は、i-nodeの数を多く指定してmkfsを実行することで、小さな多数のファイル向けのファイルシステムを構築することができる。

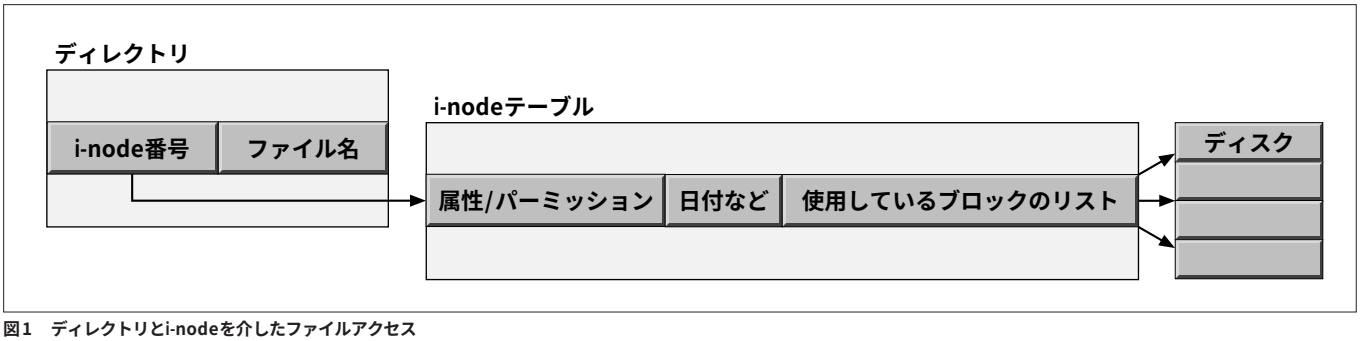


図1　ディレクトリとi-nodeを介したファイルアクセス

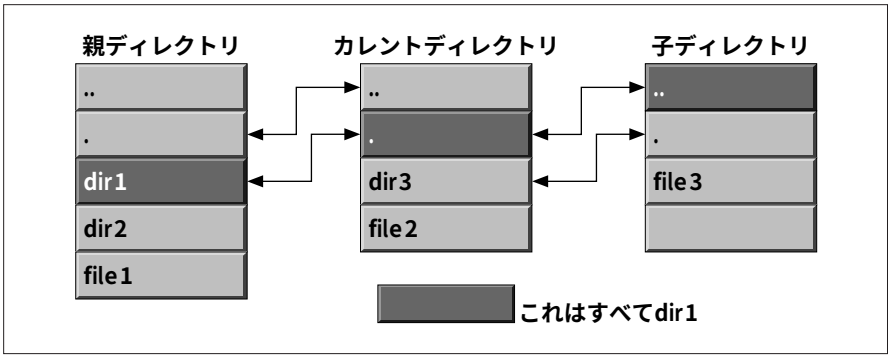


図2　各ディレクトリの関係

るディレクトリからそのサブディレクトリ（子ディレクトリ）を見たり移動することができる。

ただし、これだけでは十分ではない。親（ルート）のほうにさかのぼっていくこともできなければならないからだ。そのため、すべてのディレクトリには、「..」というディレクトリファイルが存在している。これは親ディレクトリを表している。すべてのディレクトリにおいて、その親は一意に定まるので、個々の名前で識別する必要はない。そのためUNIXでは、親ディレクトリは常に「..」という名前で表すことになっている。また、ls -alで見てみると、「..」とは別に「.」というディレクトリがあることもわかる。これは自分自身のディレクトリを表すファイルである（**図2**）。「.」ディレクトリの存在は冗長なのではないかと思うかもしれないが、そんなことはない（**リスト1**）。

パーミッションのフィールドの左端のdは、このファイルがディレクトリであることを示している。

さて、階層化ファイルシステムを構築するうえで、もう1つ考えておかなければならないことがある。ここまでの説明で、ツリーの一部分の親子関係はわかったと思うが、木構造ファイルシステムを使うためには、最初に何らかの方法でルートディレクトリにたどり着かなければならない。ルートディレクトリを識別できれば、あとはツリーのどこにでも移動できる。

UNIXはこれを単純な方法で解決している。ファイルシステムのルートディレクトリは、常にi-node番号が2なのである。mkfsでファイルシステムを構築すると、各種管理情報と共に、i-node番号が2のルートディレクトリが作成される。

また、ルートディレクトリには名前がないということも重要である。上記のlsの例からもわかるように、各ディレクトリの名前は、親ディレクトリ中で定義される。ルートディレクトリには親がないから、定義のしようがない。しかし、1つしか存在しないものを名前で識別する必要はないので、名前がなくても困ることはない。ルートディレクトリを表すために、しばしば/という表記を使うが、/はルートディレクトリの名前ではないということに注意すること。

名前がないために便利という点もあるのだ。UNIXでは、ファイルパスは相対指定と絶対指定があり、絶対指定は/という文字で始まり、相対指定は先頭が/でない。たとえばwork/doc/memo.txtというパスは、カレントディレクトリから始めてwork、docという

▼リスト1

```
$ cd dir1
$ ls -al カレントディレクトリ（dir1）の内容。
total 8
drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 .
drwxr-xr-x   4 masa     users        1024 Dec  9 04:40 ..
drwxr-xr-x   2 masa     users        1024 Dec  9 04:41 dir3
-rwxr-xr-x   1 masa     users        4171 Dec  9 04:41 file2
$ ls -al dir3     子ディレクトリ（dir3）の内容。
total 2
drwxr-xr-x   2 masa     users        1024 Dec  9 04:41 .
drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 ..
-rw-r--r--   1 masa     users           0 Dec  9 04:41 file3
$ ls -al ..       親ディレクトリ（..）の内容。自身の名前が親ディレクトリ中にある。
                  親ディレクトリの名前はもう1段上まで登らないとわからない。
total 5
drwxr-xr-x   4 masa     users        1024 Dec  9 04:40 .
drwxr-xr-x   4 masa     users        1024 Dec  9 04:39 ..
drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 dir1
drwxr-xr-x   2 masa     users        1024 Dec  9 04:40 dir2
-rw-r--r--   1 masa     users          90 Dec  9 04:40 file1
$ ls -al .        カレントディレクトリ（.）の内容
                  最初のlsと同じ結果が表示される
total 8
drwxr-xr-x   3 masa     users        1024 Dec  9 04:41 .
drwxr-xr-x   4 masa     users        1024 Dec  9 04:40 ..
drwxr-xr-x   2 masa     users        1024 Dec  9 04:41 dir3
-rwxr-xr-x   1 masa     users        4171 Dec  9 04:41 file2
```

ディレクトリをたどりmemo.txtに行き着く。ここであえて、ルートディレクトリからの相対パス指定を考えてみよう。これは絶対パス指定にほかならないが、あえてルートディレクトリに対する相対パスとして考えてみるのである。仮にルートディレクトリに<root>という名前を付けてみよう。相対パス指定なので、先頭に/は付かない。

```
<root>/usr/bin/vi
```

実際にはルートディレクトリには名前はないので、<root>は存在しない。

```
/usr/bin/vi
```

あーら不思議。相対パス指定で書いたはずなのに絶対パスになっている。こう考えてみると、ルートディレクトリに名前がないほうが便利だということがわかるだろう。

■リンク

UNIXの階層化ファイルシステムを実現するうえで、もう1つ重要な要素がある。1つのファイルの実体を複数の場所から参照するリンクである。UNIXには、シンボリックリンク、ハードリンクの2種類のリンクがある。今日、広く使われているのはシンボリックリンクで、ハードリンクなんて知らない人のほうが多いかもしれない。

もともと、UNIXがサポートしていたのは、ハードリンクであった。シンボリックリンクの登場はずっと後だ。だが、現在のLinuxなどのインストレーションを見ると、ほとんどのリンクがシンボリックリンクで、ハードリンクは廃れてしまった感がある。しかし、ユーザーが知らないだけで、UNIXのファイルシステムは、今も昔もハードリンクなしでは成り立たないのである。

■シンボリックリンク

ハードリンクの話の前に、シンボリックリンクの話をしておこう。WindowsやMacintoshのユーザーなら、ショートカット（あるいはエイリアス）という種類のファイルを知っているだろう。Windowsであれば、エクスプローラを使ってファイルやディレクトリのショートカットファイルを作ることができる。たとえば、C:ドライブの下のよく使う特定のディレクトリやプログラムファイルのショートカットを作成し、それをデスクトップに移動することができる。これをダブルクリックすると、元のアプリケーションプログラムを起動したり（ショートカットが.exeを指す場合）、特定のフォルダをオープンする（ショートカットがディレクトリを指す場合）ことができる。つまり、ファイルの実体を移動したりコピーしたりすることなく、複数の場所で1つのファイルを参照できるようになるのである。

多くのユーザーは、UNIXでもこれができることを知っているだろう。そう、これがシンボリックリンクである。UNIXのシンボリックリンクは、特殊なファイル属性の1つで、ファイルの内容をファイルパスとして認識するという仕組みである。cdによるディレクトリの移動とか、ファイルのオープン（そして読み書き）の対象としてシンボ

Column

rootのホームディレクトリ

システム管理用のユーザーアカウントであるrootのホームディレクトリは、今は/rootが一般的であるが、以前は/であった。そのため、/中にroot用の各種プライベートファイル（.profileやmboxなど）があり、ルートディレクトリが結構グシャグシャしていた。また、個々のファイルがパーミッションで保護されていても、ディレクトリそのものは公開されているので、セキュリティ上の問題もあった。/rootにすることによって、その辺の問題は解決したのである。

しかし、よく考えてみよう。

一般に、UNIXのユーザー名とホームディレクトリ名は同一である。ユーザーmasaのホームディレクトリは、/home/masaというようになる。なぜUNIXのシステム管理アカウントの名称はrootになったのか？　これは筆者の想像であるが、ホームディレクトリがルートディレクトリだったからではないのか？　ただ、ルートディレクトリには名前はないので、rootという名称にしたのだろう。/rootを作ることで、ユーザー名とホームディレクトリ名が一致したように思えるが、名前のないルートディレクトリ中にrootという名前のディレクトリがあるというのは、美しくない。また、ルートディレクトリの下に名前を持ったサブディレクトリを用意するのであれば、rootという名前にこだわる必要はまったくなくなる。adminでもmanagerでもなんでもいいのである。

DOS／Windowsのディレクトリ

MS-DOS／Windowsでは、サブディレクトリには「.」と「..」があるが、ルートディレクトリにはこれらがない。これはバージョン1.XのDOSとの互換性のためであるが、いかにも美しくない仕様であるし、このファイルシステム上でディレクトリなどをいじるツールを書くプログラマーはこれで苦労した。ルートディレクトリだけ例外的に扱わなければならなかったからだ。

また、DIRコマンドなどで見ると、なぜかディレクトリの大きさが表示されない。実際には何らかの長さのファイルであるにも関らずだ。互換性のために、何とも美しくないファイルシステムになっているのだ。

リックリンクファイルが指定された場合、これらの操作は、シンボリックリンクファイルそのものではなく、シンボリックリンクによって指定されたターゲットファイルに対して行われる。

ハードリンクと対比するために、シンボリックリンクの特徴を挙げておこう。

- **リンクされるターゲットファイルと、リンクファイルが明確に区別される。**
- **ターゲット側は、シンボリックリンクによって参照されていることを認識していない。**
- **シンボリックリンクは、任意の種類のファイル（ディレクトリも含む）を参照できる。**
- **シンボリックリンクはパーティションをまたいで参照できる。**

■ハードリンク

最初に説明したUNIXファイルシステムの基本的な構造を思い出してもらいたい。ディレクトリファイル中のエントリは、ファイル名とi-node番号を対応付けるものである。i-nodeは、そのファイルの属性やディスク上の位置を示す情報であるが、ファイル名とか、どこのディレクトリから参照されているといった情報は持っていない。同じディレクトリ中の複数のファイル名エントリ、あるいは異なるディレクトリ上のファイルエントリが同じi-nodeを参照した場合はどうなるだろうか？これにより、異なるパス名の指定で同じファイルの実体がアクセスされることになる。実はこれは、UNIXでは完全に合法なことなのである。というか、UNIXはこのハードリンクを使って階層化ファイルシステムを実現しているのである。

異なるディレクトリエントリが同じi-node（つまり同じファイルの実体）を参照することを、シンボリックリンクに対してハードリンクという（もともとはリンクという用語を使っていたのだが、シンボリックリンクの登場により、区別するためにハードリンクと呼ばれるようになった）。

i-nodeは、どのディレクトリエントリから参照されているかという情報は持っていないが、いくつのディレクトリエントリから参照されているかという情報を、属性の1つとして持つ。これをリンクカウントという。ls -lの出力の、パーミッションのフィールドの次の数字がリンクカウントである。

ハードリンクは、i-node番号に基づいて行われる。つまり、i-node番号が一意にファイルを識別するスコープでのみ使えるということだ。簡単に言えば、同一ファイルシステム上でのみ可能ということである。たとえば、/と/usrが別パーティションのファイルシステムの場合、/ファイルシステム上には、/usr中のファイルにリンクするハードリンクは作成できないということである。また、UNIXでは、一般ユーザーはディレクトリのハードリンクを作成できない。rootのみこれを行える。

効率という点では、ハードリンクに分がある。シンボリックリンクの場合、ファイルアクセスが2度手間、つまり、まずリンクファイルにアクセスし、次にその内容のパス名のファイルにアクセスすることになるが、ハードリンクの場合は、1アクションでファイルにアクセスできる。また、ハードリンクの場合、参照している別のディレクトリエントリがmvで（同一ファイルシステム中で）移動されたり、名前が変えられてしまっても、リンク関係は維持される。移動や名前変更は、i-nodeの情報に影響しないからだ。一方、シンボリックリンクのように、実体とリンクを明確に区別することはできない。こういった点がメリットかデメリットかは、用途次第だ。

Column

リンクの解除

プログラマーしか知らないことであるが、ファイルを削除するシステムコールがunlinkであるというのも、これが理由だ。ファイルの削除は、ディレクトリファイルからそのエントリを削除し、対応するi-nodeをクリアし、ディスクブロックを開放する処理なのであるが、i-nodeのクリアとディスクブロックの開放は、ほかのハードリンクが残っている間は行われない。実際にこれらの処理を行うのは、ほかのハードリンクが残っていない、つまりリンクカウントが1の時だけである。リンクカウントが1より大きい場合は、ファイル削除の処理は、ディレクトリエントリを抹消すること、そして、i-node中のリンクカウントを1だけ減らすことである。それゆえ、UNIXのファイル削除システムコールは、unlink、つまりリンク解除なのである。

これに対して、シンボリックリンクの削除は異なった処理になる。シンボリックリンクファイルそのものの削除は普通に行われる。i-nodeにはどこかからシンボリックリンクされているという情報は含まれておらず、シンボリックリンクの削除はほかに何の影響も与えることなく行うことができる。逆の場合は困ったことが起こる。i-nodeにシンボリックリンクされているという情報は何も含まれていないので、シンボリックリンクされている側のターゲットファイルは、自由に削除できるのである。シンボリックリンクの対象が削除、移動、名前変更されても、システムは何も検出しないし、何も関知しない。シンボリックリンクを参照した時に、そのファイルが見つからないというエラーを返すだけである。

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（後編）―

Linuxの標準シェルであるbashのコマンドラインを中心として、Linuxの便利な使い方について紹介していく本連載。今回は、前回に引き続きRedHat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（RedHatパッケージマネージャ）について取り上げ、ソースパッケージの作成方法について解説する。

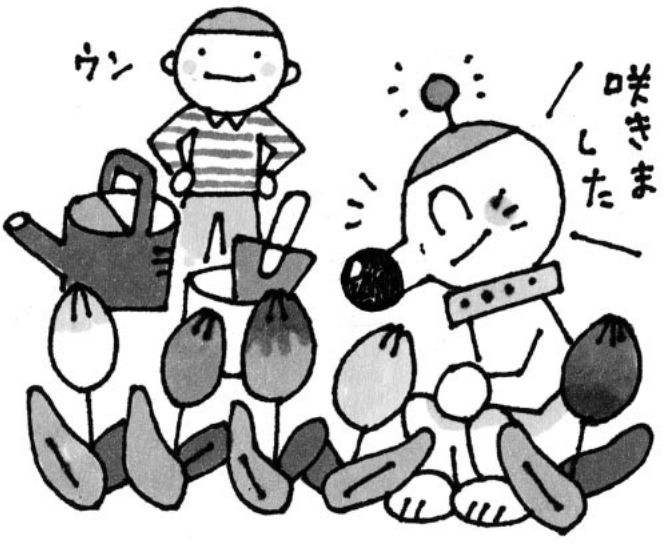


Illustration : Manami Kato

今月のお題

tarボールからRPMパッケージを作成する

文：**大池 浩一**
Text : Kouichi Ooike

先月号はお休みをいただいたため、2カ月ぶりの登場となる今回は、前回に引き続いて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）を取り上げる。まずは復習がてら、RPMについておさらいしてみよう。

RPMの特徴は、インストールやアンインストールが容易なことに加え、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こ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のだ。たとえば、他のパッケージが利用しているコマンドやライブラリを含んだ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て通常はアンインストールできない。

このように便利な特徴を持つRPMだが、バイナリパッケージのインストールしか使ったことがないという人も結構いるようだ。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てしまい、「--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強引にインストールした結果、システムをおかしくしてしまった経験を持つ人もいるはずだ。

RPMのソースパッケージを活用しよう

RPMを本当の意味で使いこなすには、バイナリパッケージをインストールするだけでなく、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。そこで前回は、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の使用方法と、その応用例を説明したのだった。

今回は、rpmを利用してバイナリパッケージを作成する方法を説明しよう。まずは、(a) 既存のソースパッケージからバイナリパッケージを作成する「リビルド」、続いて(b) 既存のソースパッケージの内容を一部修正してカスタマイズされたソースパッケージを作成する方法、さらには(c) 自分で新たなソースパッケージを作成する方法という具合に、難易度の低いものから高いものへと順を追って説明する。(a) については、プログラミングに関する専門知識がなくても大丈夫なので安心してほしい。(b) と (c) についても、小規模なソフトのtarボールを展開して、READMEを参照しながらビルド、インストールできる程度の知識があれば大丈夫だ。

●ソースパッケージをリビルドする

RPMのバイナリパッケージのインストール方法を覚えた初心者がついやりがちなのが、WebサイトやFTPサイトにあるバイナリパッケージをやみくもに取得してはインストールしてしまうことだ。ここで注意すべき点は、特定のディストリビューション用に作られたバイナリパッケージが、他のディストリビューションでそのまま動作するとは限らないということだ。

たとえば、Red Hat 6.x系ディストリビューション（Red Hat 6.1やLASER5 6.0など）では、システムライブラリにglibc2.1が採用されている。これらのディストリビューションで作成されたバイナリパッケージを、glibc2.0を採用したディストリビューション（TurboLinux 4.xやRed Hat5.2系のVine 1.1など）にインストールしようとしても、ライブラリの依存関係のチェックではねられてしまうのだ（**画面1**）。さらに、ライブラリのバージョンが共通の場合でも、ディレクトリ構成が微妙に異なるためにインストールできない場合もある。もちろん運良く動作する場合だってままあるのだが、基本的にはそのディストリビューションのために作られたバイナリパッケージ以外は利用しないほうがいい。解決方法はちゃんと用意されているのだから。

解決方法とは、あなたが使っているディストリビューション用のバイナリパッケージを自分で作成してしまうことだ。白紙の状態からパッケージを作るのは結構大変なので、最初は別のディストリビューション用のソースパッケージを流用する。ファイル名の末尾が「.i386.rpm」のバイナリパッケージではなく、「.src.rpm」のソースパッケージを探すのだ。

ソースパッケージは、バイナリパッケージを作成することを目的としたパッケージで、実行ファイルを作成するのに必要なソースのtarボールや修正用のパッチファイルのほか、バイナリパッケージの作成手順やインストールするファイルリストなどが記述された「specファイル」が含まれている。このspecファイルがソースパッケージのかなめであり、パッケージの作成とspecファイルの作成はほぼ同義である。specファイルの内容を修正できるようになればRPM使いとしては中級者、白紙の状態から作成できるようになれば上級者だといえよう。

初心者は手始めに、既存のソースパッケージに含まれるspecファイルをそのまま利用してバイナリパッケージを作成する「リビルド」の方法を覚えよう。「su -」などとしてスーパーユーザ（root）になった状態で、--rebuildオプションとソースパッケージファイル名を指定してrpmを起動すればいい。

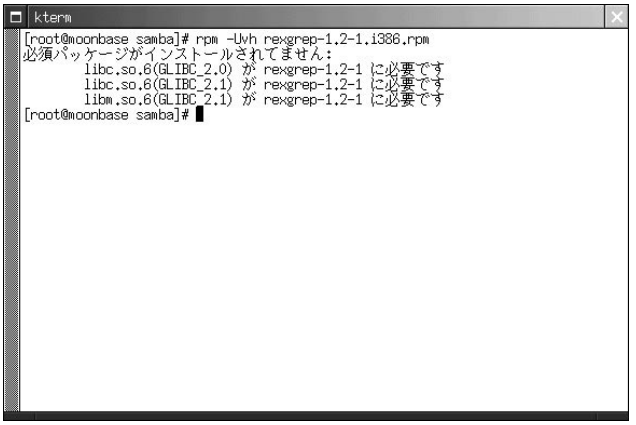
たとえば、ソースパッケージのファイル名が「hoge-1.2.3-4.src.rpm」だとすると、

```
# rpm --rebuild hoge-1.2.3-4.src.rpm
```

とする。ktermなどの端末画面に、ずらずらとリビルド中の処理の様子が表示されるはずだ（**画面2**）。

リビルドで行われる主な処理は、ソースのtarボールの展開、パッチファイルによるソースの修正、configureスクリプトによる各種調査とMakefileの作成、makeによるコンパイルとリンク、仮想インストール、バイナリパッケージに必要なファイルの収集などだ。問題が発生した時点でリビルドは中断される。最後まで問題なく処理が進むと、バイナリパッケージが/usr/src/redhat/RPMS/i386に作成されている。作成されたバイナリパッケージは、WebやFTPから入手したものと同様に、-Uvhオプション付きのrpmでインストールできる。

Web／FTPサイトから入手したバイナリパッケージではライブラリの依存関係のチェックでインストールに失敗し



画面1 バイナリパッケージのインストール失敗例

画面2 ソースパッケージをリビルド中の様子

たのに、同じサイトから入手したソースパッケージをリビルドすると、正常にインストールできるバイナリパッケージが作成されるのはなぜだろうか。それは、ソースのコンパイルやリンクといった作業も含めて、バイナリパッケージの作成をあなたのマシン上で行ったからだ。

ライブラリの依存関係は、バイナリパッケージ作成時にRPMによって自動的に調査され、パッケージの情報に追加される（一方、パッケージ間の依存関係はパッケージ作成者が設定する）。つまり、自分のマシンで作成したバイナリパッケージについては、glibcのバージョンが異なるといった問題は起きようがないのだ。

もちろん、ソースパッケージによっては、リビルドに失敗したり、リビルド自体は成功してもバイナリパッケージのインストールに失敗することもある。また、configureスクリプトやmakeに特定のオプションを追加したい場合や、ソースにパッチを当てて修正したい場合もあるだろう。こうしたケースでは、specファイルの内容を修正して、ソースパッケージ自体を作り直す必要がある。

●specファイルの展開と概要

specファイルの内容を修正するには、いったんソースパッケージからspecファイルを取り出さなくてはならない。そのためには、ソースパッケージをインストールする必要がある。具体的には、スーパーユーザー（root）になった状態で、-iオプションとソースパッケージファイル名を指定してrpmを実行する。たとえば、

```
# rpm -i hoge-1.2.3-4.src.rpm
```

とすればいい。

ソースパッケージでは、tarボールやspecファイルのインストール先が固定されている。トップディレクトリ（通常は/usr/src/redhat）のSPECSディレクトリにspecファイル、tarボールやパッチファイルなどはSOURCESディレクトリに展開される。こうしたディレクトリの構成については図1を参照されたい。

specファイルのファイル名は、「パッケージ名-バージョン番号.spec」という書式だ（バージョン番号は省略可能）。内容は、大まかにいって「データ定義部」「スクリプト部」「ファイルリスト部」の3つで構成される。詳しい書式などについては後ほど説明することにして、ここでは各部の概要を簡単に説明しよう。

・データ定義部

パッケージ名やバージョン、リリース番号、tarボールやパッチのファイル名、依存するパッケージやそのバージョンなど、パッケージの作成時やインストール時に利用されるさまざまな情報が列挙されている。

・スクリプト部

パッケージの作成方法が記述された重要部分。tarボールの展開やパッチ当てが行われる「%prepセクション」、configureスクリプトの実行やmakeによるコンパイル・リンクが行われる「%buildセクション」、仮想インストールが行われる「%installセクション」など、さらに細かなセクションに分かれている。

・ファイルリスト部

スクリプト部で展開、作成されたファイルのうち、バイ

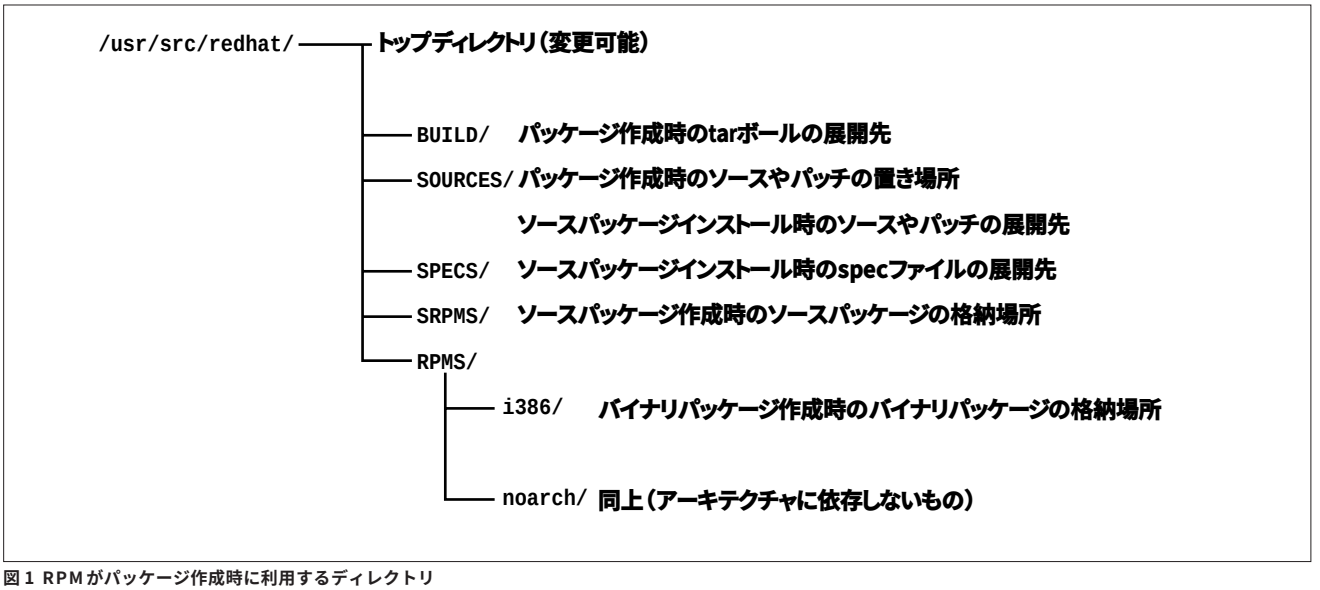


図1 RPMがパッケージ作成時に利用するディレクトリ

ナリパッケージに含めたい実行ファイルやドキュメント、データファイルなどが列挙されている。バイナリパッケージのインストールやアンインストールは、このリストに基づいて行われる。

●specファイルを修正してパッケージを作成する

specファイルは単なるテキストなので、Emacsなどのエディタで内容を参照・変更可能だ。シェルやPerlなどのスクリプトと同様、「#」以降から行末まではコメントとして扱われる。

修正する機会が多いのはスクリプト部だろう。この部分の内容は、%prefや%buildといったセクションごとにRPMによりシェルスクリプト化され、bashによって実行される。「$変数名」（後に文字列が続く場合は${変数名}）という書式で変数の参照も可能だ。RPMにより内容が設定される変数（RPM_BUILD_ROOTなど）がいくつか用意されており、スクリプト部で利用できる。

一方、specファイル内の各部で利用可能なマクロも用意されており、「%名前」という書式で参照される。変数と異なるのは、specファイル上で値の展開が行われる点だ。「%define 名前 内容」として設定するほか、あらかじめ内容が設定された組み込みマクロが多数用意されている。引数を使った関数的な動作も可能で、実際に「%setup」（tarボールの展開）や「%patch」（パッチ当ての実行）など、specファイルの記述の単純化に役立っている。

以上のような基本知識があれば、スクリプト部の修正は比較的簡単だ。たとえば、configureスクリプトに特定のオプションを追加する場合には、%buildセクションに含まれる「./configure…」や「%configure…」の行に、目的のオプションを追加すればいい。makeにオプションを追加する場合も同様だ。

一方、ソースにパッチを当てたり、依存するパッケージ名を変更する場合には、データ定義部に新たなタグを追加したり、タグの設定内容を修正する必要がある。また、こうした変更によりインストールされるファイルが増減した場合には、ファイルリスト部の記述を対応させなくてはならない。具体的な作業については後で説明しよう。

さて、修正を加えたspecファイルが完成したら、それに基づいたソースパッケージとバイナリパッケージを作成することができる。スーパーユーザー（root）になった状態で、-baオプションとspecファイル名を指定してrpmを実行すればいい。

たとえば、トップディレクトリ（/usr/src/redhat）がカレントディレクトリで、SPECディレクトリに「hoge-1.2.3.spec」があるなら、

```
# rpm -ba SPECS/hoge-1.2.3.spec
```

とすると、リビルドの場合と同様に、端末画面にずらずらと処理の様子が表示される。すべての処理が問題なく終了すると、新たなソースパッケージがSRPMSディレクトリに、バイナリパッケージがRPMS/i386ディレクトリにそれぞれ作成される。なお、-baではなく-bbオプションを使ってバイナリパッケージのみ作成することも可能だ。しかし、specファイルやtarボール、パッチファイルを散逸させないためにも、バイナリパッケージと同時にソースパッケージも作っておいたほうがよい。

ところで、RPMの設定を一部変更し、specファイルの書き方を工夫すれば、一般ユーザーのままでパッケージの作成やリビルドを行える（コラムを参照）。特に、頻繁にパッケージ作成作業を行ったり、他人に配布するパッケージを作成するような場合は、一般ユーザーのまま作業することをお勧めする。

Column

一般ユーザー権限でパッケージを作成するには

本文中では、スーパーユーザー（root）になった状態でパッケージの作成やリビルドを行っている。その理由は、パッケージ作成時に使われる/usr/src/redhat/RPMS/i386などのディレクトリには、rootでしかファイルを作成できないからだ。しかし、rootにはあらゆるファイル・ディレクトリに対する書き込み、読み込み権限が与えられているため、specファイルの記述ミスが原因で、予想外のディレクトリにファイルをインストールしたのを見逃してしまったり、重要なファイルを含んだディレクトリを誤って削除してしまう危険がある。

そこで、RPMには一般ユーザーのままでパッケージを作成する方法が用意されている。上記のような事態になっても、そのディレクトリに書き込み権限がない一般ユーザーならば、エラーメッセージが表示されるだけですむので安全だし、間違いにも気がつきやすい。

一般ユーザーのままパッケージを作成するには、トップディレクトリを/usr/src/redhatから、書き込み権限のあるディレクトリ（~/rpmなど）に変更し、図1と同じサブディレクトリを作成しておく必要がある。変更方法はRPMのバージョンにより異なり、2.5では~/.rpmrcの「topdir:」、3.0では~/.rpmmcarosの「%_topdir」に、それぞれ変更後のディレクトリの絶対パスを設定する。

今月のお題

☑ tarボールから RPMパッケージを作成する

後半は毎回ひとつのテーマに絞り、それを実現する方法を説明する。今回のお題は、

tarボールからRPMパッケージを作成する

というもの。tarボールでのみ配布されているソフトのspecファイルを新たに作成して、RPMのソースパッケージとバイナリパッケージを作るまでの過程を、順を追って説明する。パッケージ化により、インストールやアンインストールが容易になり、依存関係のチェックの恩恵も受けられる。また、作成したパッケージを提供して、ディストリビューションに貢献することも可能だ。

例として、本誌「Free Application Showcase」で紹介している「reXgrep」（151ページ）というソフトを取り上げる。これは、指定した文字列を含む行を出力するgrepのフロントエンドで、検索文字列や検索ファイル名をGUIで設定できるのが特徴だ。

●tarボールの展開とパッチの作成

まずは、トップディレクトリ（通常は/usr/src/redhat）以下のSOURCESディレクトリに、reXgrepのtarボールをコピーする。

```
# cp rexgrep-1.2.tar.gz /usr/src/redhat/SOURCES
```

SOURCESディレクトリは、パッケージの作成に使われるソースのtarボールやパッチファイルの置き場所だ（複数のtarボールやパッチファイルを組み合わせることも可能だがここでは触れない）。

さて、オリジナルのreXgrepでは、GTK+のロケール設定が行われていないため、検索結果の表示で日本語が文字化けしてしまう。この問題を解決するには、ソースを2カ所修正する必要がある。RPMでは、こうした場合、オリジナルのtarボール内のファイルを変更する代わりに、変更内容が書かれたパッチファイルを別途用意して、パッケージ作成時にパッチを当てることを推奨している。オリジナルとの違いが明確になるし、バージョンアップにも対応しやすいからだ。

パッチファイルを作成する際、オリジナルのままのファイル一式と、変更を加えたファイル一式をそれぞれ別のディレクトリに置くといい。BUILDディレクトリに移動して、さきほどコピーしたtarボールを展開する。

```
# cd /usr/src/redhat/BUILD
# tar zxvf ../SOURCES/rexgrep-1.2.tar.gz
```

すると、rexgrep-1.2というディレクトリにファイル一式が展開される。このディレクトリを「rexgrep-1.2.orig」に変更したうえで、もう一度tarボールを展開する。

```
# mv rexgrep-1.2 rexgrep-1.2.orig
# tar zxvf ../SOURCES/rexgrep-1.2.tar.gz
```

これで、BUILDディレクトリには同じファイル構成のrexgrep-1.2.origとrexgrep-1.2が作成された。続いて、rexgrep-1.2のrexgrep.cとdisplay.hを修正する（修正内容は151ページを参照）。

本来なら、ここでいったんビルドして動作を確認するところだが、誌面の都合で省略してパッチファイルの作成に移る。BUILDディレクトリに移動した状態で、diffを-uNrオプション付きで実行する。

```
# diff -uNr rexgrep-1.2.orig rexgrep-1.2 > ../SOURCES/rexgrep-1.2-locale.patch
```

これで、2カ所の修正部分がパッチファイルrexgrep-1.2-locale.patchとしてSOURCESディレクトリに作成される。lessなどで内容を確認しておこう（**画面3**）。バックアップファイルやバイナリファイルなどに関する不要な情報が含まれている場合には削除する必要がある。

画面3 パッチファイルの内容をlessで確認

●データ定義部を作成する

specファイルを書く準備は整った。テキストエディタで「rexgrep-1.2.spec」を編集しよう。まずは、データ定義部から記述していこう（**リスト1**）。「タグ:内容」という形式で1行に1つずつデータを記述する。

(1) Summary

パッケージの簡単な説明を英語で書く。specファイルのタイトルも兼ねて1行目に書かれることが多い。

(2) Name、 Version、 Release

パッケージ名、バージョン番号、リリース番号をそれぞれ記述する。パッケージのファイル名は、これらの情報に基づいて作成される。

(3) Source、 Patch

SOURCESディレクトリに置いたソース（tarボール）とパッチファイルの名前を記述する。パッケージ作成時に参照されるのはファイル名の部分だけだが、入手先を示すためにURLで記述することが望ましい。複数のソースやパッチがある場合には、「Source0:」「Source1:」…、「Patch0:」「Patch1:」…のように0から始まる番号を付ける。

(4) Copyright、 Group、 URL、 Packager

パッケージのライセンス（GPLなど）、カテゴリー、ソースの情報を提供しているURL、パッケージ作成者の名前／メールアドレスをそれぞれ記述する。

(5) Buildroot

パッケージ作成時に仮想的なルートディレクトリ（/）と見なすディレクトリを指定する。「/var/tmp//パッケージ名-root」が使われることが多い。具体的な使用方法は後で説明しよう。

(6) Requires

パッケージが動作するのに必要とするパッケージ名やバージョンを記述する。複数のパッケージをカンマ区切りで並べることも可能だ。なお、実行に必要なライブラリの情報はRPMが自動的に追加してくれるので、specファイルで記述する必要はない。

reXgrepの場合は、内部でgrepを呼び出しているためgrepパッケージが必要だ。また、再帰的な検索機能を持つバージョン2.3以降が必須なので、「grep >= 2.3」という記

▼リスト1 specファイル（データ定義部と説明）

```
Summary: Graphical frontend to the grep command + locale patch
Name: rexgrep
Version: 1.2
Release: 1
Source: http://rexgrep.tripod.com/rexgrep-1.2.tar.gz
Patch:  localhost:${RPM_SOURCE_DIR}/rexgrep-1.2-locale.patch
Copyright: GPL
Group: Applications/Text
URL: http://rexgrep.tripod.com/rexgrepmain.htm
Packager: Koichi OIKE <daichi1@gol.com>
Buildroot: /var/tmp/rexgrep-root
Requires: grep >= 2.3

%description
rexgrep is a graphical frontend to the grep command, which has been
written by Rohit Singh <Rohit.Singh@ieee.org> and distributed under
the terms of the GNU GPL.
rexgrep combines the power of the 'grep' command with the convenience
of a very easy to use graphical user interface, while not cmpromising
on its functionality.
```

述になる。

(7) %description

rpmの-qiオプションで表示されるパッケージの説明を記述する。「%」で始まることからもわかるように、データ定義部には含まれない独立したセクションだが、データ定義部と並べて書くとわかりやすい。次のセクションの直前までが内容となるので、1行だけのSummaryと違って複数行にわたる長い文章を書ける。

なお、Summaryと%descriptionは国際化機能を持っている。通常のSummaryと%descriptionに英語の説明、Summary（ja）と%description -l jaに日本語の説明を書いておけばいい。環境変数LANGやLANGUAGEの内容を参照して、日本語環境では日本語のメッセージが表示される。

●スクリプト部を作成する

スクリプト部は、パッケージの作成方法が記述された重要部分で、「%prep」「%build」「%install」などいくつかのセクションに分かれている（**リスト2**）。各セクションはbashのシェルスクリプトとして実行されるため、bashの組み込みコマンドや外部コマンド、ifなどの制御構文、${変数名}による変数参照などが可能だ。

(1) %prepセクション

tarボールの展開とパッチ当てを行う。tarボールの展開は「%setup」、パッチ当ては「%patch」という組み込みマクロで記述するのが普通だ。

「%setup」は、データ定義部のSourceタグで指定されたソース（tarボール）を、tarを使ってBUILDディレクトリに展開し、展開先ディレクトリをカレントディレクトリとする。-qオプションにより、展開時のファイル名表示を抑制可能だ。tarボールに含まれるディレクトリが「パッケージ名-バージョン番号」と食い違う場合、-nオプションに続けてディレクトリ名を明示しなければならない。reXgrepでは両者が一致しているので指定は不要だ。

「%patch」は、データ定義部のPatchタグで指定されたパッチファイルを、外部コマンドのpatchに入力し、展開されたソースに対するパッチ当てを行う。%setupを実行した時点で、カレントディレクトリがBUILD/rexgrep-1.2になっているため、パッチのファイル名情報に含まれるディレクトリ部分を1レベル無視する-p1オプションを指定している。なお、パッチファイルがgzipで圧縮されている場合には自動的に展開される。

(2) %buildセクション

configureスクリプトの実行により、動作環境の調査が行われ、環境に合ったMakefileが作成される。続いて、makeによるコンパイル・リンクで実行ファイルが作成される。カレントディレクトリはBUILD/rexgrep-1.2だ。

configureスクリプトの実行は、マクロを使わずに直接

▼リスト2 specファイル（スクリプト部）

```
%prep
%setup -q
%patch -p1

%build
./configure --prefix=/usr
make

%install
rm -rf $RPM_BUILD_ROOT
make prefix=$RPM_BUILD_ROOT/usr install

%clean
rm -rf $RPM_BUILD_ROOT
```

Column

RPM 2.5と3.0の違いとは

現在使われているRPMには、Red Hat 5.x系ディストリビューションに含まれる2.5と、Red Hat 6.x系ディストリビューションに含まれる3.0の2種類ある。3.0では、configureスクリプトを「--prefix=/usr」オプション付きで実行する%configureマクロなど、いくつかのマクロやタグが追加されている。また、設定ファイルの構成も一部変更されている（rpmmacrosの追加など）。

RPM 3.0用のソースパッケージを2.5でリビルドする際、3.0でしか利用できないマクロがspecファイルに含まれているとリビルドに失敗するが、たいていはspecファイルをわずかに修正すれば流用可能だ。しかし、追加されたマクロ一覧などの解説が用意されていないため、tarボールで配布されている3.0のソースを読む必要がある。

RPMを開発・配布しているRed Hat Softwareでは、Red Hat 5.x系でも3.0を使うことを推奨している（http://www.redhat.com/support/errata/rh52-errata-general.html#rpm）。ただし、2.5と3.0ではデータベースの構造が異なる。データベース自体は、rpmを--rebuilddbオプション付きで実行することで再生成できるものの、ディストリビューション付属のバイナリパッケージに使えないものが出てくるので注意が必要だ。

「./configure」としている。「--prefix=/usr」オプションにより、実行ファイルやマニュアルなどのインストール先が/usr以下になる（省略時は/usr/local以下）。RPM 3.0ではこうした設定を自動的に行う「%configure」オプションが用意されている（コラム参照）。

(3) %installセクション

実行ファイルやマニュアルなどのインストール（ファイルのコピー）を行う。組み込みマクロは用意されていないので、「make install」のようにコマンド名を直接記述するのが普通だ。

ただし、そのままインストールしたのでは、パッケージ化されていないファイルを直接/usr/binなどに置くことになる。さらに、前のバージョンのパッケージがインストールされていた場合は、それらを上書きしてしまう。こうした事態を回避するため、実際のルートディレクトリ（/）とは別のディレクトリを仮想的なルートディレクトリと見なしてインストールする方法が使われる。

仮想的なルートディレクトリは、データ定義部のBuildrootタグで指定する。ここで注意すべき点は、Buildrootタグを指定しても、自動的にインストール先が変更されるわけではないということだ。

実際には、Buildrootタグの内容が設定された変数RPM_BUILD_ROOTを利用して、インストール先が仮想的なディレクトリになるように、specファイルを記述しなければならない。configureスクリプトを利用するreXgrepの場合は、「make install」のオプションとして「prefix=$RPM_BUILD_ROOT/usr」を指定する。

(4) %cleanセクション

ここでは、パッケージ作成後の処理を行う。具体的には、Buildrootタグで指定した仮想的なルートディレクトリを、サブディレクトリも含めて削除する。なお、%installセクションでインストールを実行する前にも、これと同じ処理を行っている。

▼リスト3 specファイル（ファイルリスト部）

```
%files
%defattr(-,root,root)
%doc AUTHORS COPYING ChangeLog INSTALL NEWS
README TODO
/usr/bin/rexgrep
/usr/man/man1/rexgrep.1
```

●ファイルリスト部の作成と動作チェック

ファイルリスト部には、バイナリパッケージに含める実行ファイル、ドキュメント、データファイルなどを列挙する（**リスト3**）。ディレクトリ自体をリストに含めると、アンインストール時にディレクトリごと削除される。また、組み込みマクロの「%attr」「%defattr」により、ファイル属性、所有者、グループを変更可能だ。

ファイルはフルパスで1行に1つずつ記述する。実際にインストールするディレクトリを記述することに注意されたい。ただし、ソースのtarボールに含まれるドキュメント類（READMEなど）は、「%doc」に続けてファイル名だけを並べて記述する。これらは、/usr/doc以下の「パッケージ名-バージョン番号」ディレクトリにまとめてインストールされる。

これでようやくspecファイルが完成した。最後に、-baオプションとspecファイル名を指定してrpmを実行し、ソースパッケージとバイナリパッケージを作成する。カレントディレクトリがトップディレクトリなら、

```
# rpm -ba SPECS/rexgrep-1.2.spec
```

とすればいい。

説明の都合上、一気にspecファイルを作成してしまったが、本来は%prepセクションまで作成した時点で-bp、%buildセクションまでなら-bc、%installセクションまでなら-biといった具合に、段階的にパッケージ作成処理を行うオプションを使って、各セクションの記述内容に誤りがないか確認しながら作業を行う。また、他人に配布するパッケージの場合は、作成したのとは別のマシンにインストールして、動作チェックを行ったほうがいい。

最後に、パッケージ作成の参考となるURLをいくつか紹介しよう。日本語で書かれたものでは、

- **Making RPM（http://vinelinux.org/rpm/MakingRPM/index.html）**
- **rpm-packaging HOWTO（http://www.ns.musashi-tech.ac.jp/~inoue/Pages/Linux/rpm-packaging.howto.html）**
- **src.rpmを作ろう（http://www.best.com/%7Ehoshina/linux/redhat-50.html）**

がわかりやすい。さらに詳しい内容を求めるなら、英語で書かれた第一級資料、

- **Maximum RPM（http://www.rpm.org/）**

がある。400ページ以上ある書籍の内容がPostScript形式とLaTeX形式で配布されている。

Webサーバ構築術(第7回)

アクティブなページやインタラクティブなページというと、CGIを使うものである。しかし、そこまでしなくても、SSIである程度のことならできてしまう。管理者にはサーバが重くなるとかセキュリティが守れないという意味で嫌がられるSSIだが、SSIによるメリットもある。そんなSSIの一面を今回は紹介する。

SSIで動的なページを扱う

文:中島昌彦
Text : Masahiko Nakajima

ファイルの更新日表示をしているサイトがあるが、そういう仕組みを自分のサイトにも欲しいと思ったことはないだろうか。このような機能を手軽に実現するのがSSIである。Webでのパブリッシング作業がこれまでのような負荷をかけずにできるようになる。

SSIもCGIもサーバ側で処理するプロセス

SSIとCGIを混同するケースは非常に多い。SSIとCGIは、どちらもサーバ側で処理する機能だが、SSIはServer Side Includeの意味で、HTML内に書き込まれたSSIのタグを処理するものだ。

コンテンツを作成する側からすると、CGIであれば、ある程度のプログラミング知識が必要になる。しかし、SSIであれば、タグを入れ込む感覚で利用できる。そして、入れたタグはWebブラウザが解析して表示するのではなく、Webサーバが特定タグで指定している内容に合わせて置き換えてから、閲覧者に送るという作業をする。それは、今の時間であったり、サーバ上に置かれている特定のファイルのタイムスタンプだったりする。

CGIでも、テンプレートのHTMLファイルを作り、CGIで置き換えるための目印の文字列を書き込んでおき、その必要な部分だけを置き換えてHTMLとして出力するという処理を行うこともあるだろう。SSIの場合は、その置き換え機能をApache自体が処理をしてくれる。だからこそ、ブラウザが受け取ったHTMLのソースを見たとしても、そこには動的なページの仕組みを知るためのヒントを見つけられない。JavaScriptのように、クライアントマシンには負荷をかけないし、ましてや処理内容もばれることはないのだ。

わかりやすくSSIの特徴をいってしまうと、SSIはサーバが処理してくれる拡張タグのようなものだ。しかも手軽に使えるからこそ、SSIを提供するだけで、多くの人が最終ファイル更新日付処理ぐらいはしてくれるはずだ。

SSIを使うには、Apache側でもそれなりの設定が必要だ。デフォルトの設定では、ApacheはSSIを使えないようになっている。

まず拡張子であるが、SSIの場合はshtmlが使われることが多い。Webサイトで拡張子が.shtmlであるファイルと出会うことがあるが、これはSSIを使ったページである可能性が非常に高い。

拡張子をhtmlと明確に区別して.shtmlにする理由は、サーバ自体にかかる負荷を減らすためである。通常のhtmlファイルであれば、サーバはリクエストのあったファイルを開いてその内容をアクセス元に戻すという作業をする。しかしSSIを使うと、リクエストのあったファイルを開いて、その中を調べていき、SSIのタグがあれば、その部分を処理して置き換える。アクセス数の少ないサイトであれば、ファイル全体のサーチはそれほど大きな負荷につながるわけではないが、アクセス数の多いサイトでは無視できない負荷になる。

こんな理由から、通常のHTMLファイルとSSIタグ付きHTMLファイルとを区別するために、拡張子として.shtmlを使うという区別が一般的に行われて

いる。しかし、セキュリティの関係で、SSIを使っていること自体をオープンにしたくないようなときには、.htmlをSSI用ファイルとして定義してもなんら問題はない。

httpd.confでshtmlを有効にする

SSIの設定を有効にするためには、httpd.confでSSI対象ファイルは.shtmlであるという定義をする。この設定をすることで、Webサイト全体でSSIが利用できる。

設定としては、/usr/local/apache/conf/httpd.confの680行目と681行目にある、

```
#AddType text/html .shtml
#AddHandler server-parsed .shtml
```

という2行の行頭にあるコメント（#）をはずすだけだ。

SSI用のhtmlファイルのコンテンツタイプは、閲覧者にとってみれば、あくまでもtext/htmlである。このため、.shtmlという拡張子には、text/htmlのコンテンツタイプを指定する。そして、AddHandlerを使い、拡張子.shtmlファイルはSSIとしてサーバ側で分析（server-parsed）処理するものとして定義する。この2行がそろって、ようやく.shtmlファイルがSSI対象のファイルとして認識されるわけだ。

ところが、準備はこれだけではない。SSIは、ディレクトリオプションとして、Includesが指定されていなければ動作しない。そこで、httpd.confの315行目を、

```
Options Indexes FollowSymLinks Includes
```

というようにIncludesを加えてから、Apacheを再起動する。これで、SSIを利用する準備が整う（**リスト1**）。

そして、実際にどんな表示になるのかが**図1**だ。この例は、アクセスした時間と、ファイルの最終更新日時をアクティブに返すSSIだ。HTMLと対比するとわかるが、CGIを使わなくてもある程度アクティブなページが作れるわけだ。

▼リスト1 httpd.confの修正部分

```
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/usr/local/apache/htdocs">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
#    Options Indexes FollowSymLinks
    Options Indexes FollowSymLinks Includes

#
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
```

ちょうど中央部分が該当部分。Options設定でIncludesを加える。SSIの拡張子を.htmlにする場合は、.htaccessを併用し、ディレクトリ単位でOptions設定をするケースが多い。

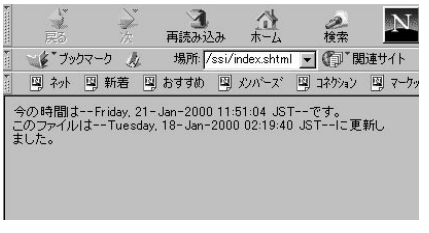
SSIは<!--#で始まり、-->で終わる

具体的に、SSIでどんなことができ、どういう結果になるのかという例を挙げよう。

SSIの書式は、

<!--#エレメント -->

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
        <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=EUC-JP">
        <TITLE>SSI SAMPLE</TITLE>
</HEAD>
<BODY>
<P>今の時間は--<!--#echo var="DATE_LOCAL" -->--です。<BR>
このファイルは--<!--#echo var="LAST_MODIFIED" -->--に更新しました。
</BODY>
</HTML>
```



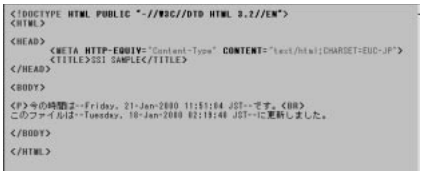
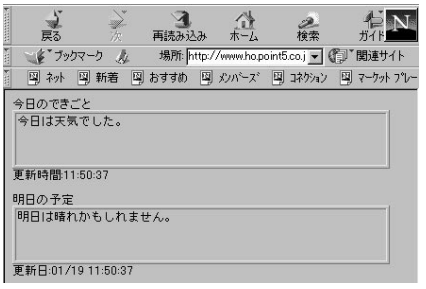


図1 簡単なSSIの例
左上が実際のソース。右上が閲覧者のブラウザに表示された画面。そして、右下がブラウザで読み込んだソース。SSIタグの部分はApacheによって置き換えられているため、表示した画面からはどんな仕組みでアクセス時間を表示しているのかはわからない。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML><HEAD>
        <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=euc-jp">
        <TITLE>TODAY</TITLE>
</HEAD>
<BODY>
今日のできごと<BR>
<TABLE BORDER="1" WIDTH="400">
        <TR><TD WIDTH="100%"><PRE>
<!--#include file="ssi/today.txt"-->
        </PRE></TD></TR>
</TABLE>
更新時間:<!--#config timefmt="%X"--><!--#flastmod file="ssi/today.txt"--><BR>
<P>明日の予定<BR>
<TABLE BORDER="1" WIDTH="400">
        <TR><TD WIDTH="100%"><PRE>
<!--#include virtual="/ssi/ssi/tomorrow.txt"-->
        </PRE></TD></TR>
</TABLE>
更新日:<!--#config timefmt="%m/%d %X"--><!--#flastmod file="ssi/tomorrow.txt"-->
</BODY>
</HTML>
```

図2　SSIのincludeとconfig、flastmodの利用例
もっともよく使われるSSIの例だ。includeはファイルを読み込む命令で、ニュースサイトやWhat 's Newページでよく使われている。configによる時間フォーマットの指定と、flastmodによる最終更新日の表示は、ページの下部に最終更新日という形で記述されていることが多い。



という形式をとる。単純に見れば、ただのタグのコメントアウトだ。このため、同じHTMLをSSI非対応のサーバに持っていっても、思ったように動作しないだけで、大きな影響は出ない。ブラウザやOSによって大きな差が出るクライアントベースのスクリプト言語よりも、問題が起こりにくいだろう。

エレメントのあとには、エレメントに合わせてなにかしらのオプションがつき、タグの最後は半角スペースが1つ入ってから、-->で終わるという基本構造を持つ。タグの最後にある半角スペースが重要で、これを入れないと動作しないサーバもある。

別ファイルの読み込みもできる

SSIの機能は充実しているが、よく使われるものとして、ファイルのタイムスタンプから最終更新日時を表示したり、広告用バナー組み込み用に、別に作成したファイルを読み込むというものだ。もしSSIを使わずにページの最終更新日時を表示しようとすると、各ページを更新するたびに、最終更新日時も忘れずに修正しないといけない。口でいうと簡単だが、この作業は結構忘れやすい。そもそも、個人のWebサイトは、最終更新日がいつのものかわからないものが多いが、このあたり、SSIの機能をISPが提供していないからかもしれない。

広告用バナー表示をするようなサイトなどは、外部の特定ファイルをSSIが参照する方式にしておかないと、広告用バナーを交換するたびに、既存のHTMLページすべてを修正していかなければいけない。バックグラウンドでデータベースが動いているようなニュースサイトなどでは、それなりの仕組みができているので、苦労せずにバナーの張り替えができるはずだが、個人サイトでバナー配信会社のシステムを使ってバナー広告を組み込んでいるようなときには、HTML中にダイレクトでタグを書き込んでいるケースがある。こんなとき、バナー配信会社を変更したとか、特別バナーを設置しなければいけないというようなとき、いくつものHTMLファイルを修正していく作業が必ず出てくる。そんなときでも、SSIの仕組みが使えるならば、手間をかけずにバナーを入れ換えられるだろう。

図2のブラウザ画面とソースを見比べてほしい。SSIのinclude命令を2個所で使っている。include命令は、指定した場所に指定したファイルを読み込むというコマンドだ。これを使えば、HTMLファイルとは別に、表示専用のファイルを置いておけば、HTMLファイルを毎回更新せずに、特定ファイルを更新するだけでページの更新が終了する。頻繁に書き換えが生じるニュース系サイトや日記サイトなど、この仕組みを使えば手間をかけずにページの更新ができるはずだ。

includeエレメントでは、ファイルの指定方法により、fileとvirtualの2つの

方法に分かれる。

図2のソースで最初に出てくる#include file=""は、HTMLファイルを読み出したカレントディレクトリから相対パスでドキュメントを指定する。このサンプルhtmlの置き場所は、/usr/local/apache/htdocs/ssi/today.shtmlである。#include file="ssi/today.txt"と指定した場合は、/usr/local/apache/htdocs/ssi/ssi/today.txtの内容を読み込み、それをHTMLファイル内に組み込んで閲覧者に送るという作業をする。

図2の中には、#include virtual="ssi/ssi/tomorrow.txt"という記述もある。この場合は、ドキュメントディレクトリからの相対パス指定だ。virtual指定をした段階で、Apacheのドキュメントディレクトリ（この場合は/usr/local/apache/htdocs/）からの指定となる。このHTMLをApacheが読み出すと、include命令に従い、ssi/today.txtとssi/tomorrow.txtを読み出して所定の位置に埋め込み、ブラウザに戻すという動作になる。

この場合はテキストファイルをincludeしているが、HTMLファイル自体も指定できる。たとえば、バナー広告用のファイルとして、**リスト2**のようなファイルを作っておき、HTMLファイルにこのファイルを読み出すようにSSIを埋め込んでおく。こうしておけば、バナーを交換するたびに関係するHTMLファイルをすべて書き替える必要がなくなる。バナーを交換するときは、SSIが読み出すファイルの中身を変更するだけで、それを参照しているすべてのHTMLファイルがバナーを切り替えられるわけだ。

リスト2　バナー広告用HTML

```
<A HREF="http://www.????.co.jp/product/">
        <IMG SRC="http://www.????.co.jp/product/newpro.gif" ALT="新製品○○○、8万円を切って17インチディスプレイ付き">
</A>
```

このような、HTMLタグをincludeすれば、ごく普通のHTMLタグとして表示される。広告用のバナー表示として、この方法はよく使われる。

フォーマット指定によるファイルの時間表示

図2のHTMLファイルには、ファイルの更新日を表示するSSI命令も使っている。それがflastmodだ。flastmodを使った日付表示は、ssi/today.txtとssi/tomorrow.txtファイルの更新日を元に表示している。書式としては、

```
<!--#flastmod file="ファイル名"-->
```

という形式になる。ただし、いきなりflastmodを使うと、

曜日，日-月-年 HH:MM:DD タイムゾーン

という形式での表示になる。この表示フォーマットでは、曜日と月の表示が英語表記になるうえに、年月日の表示が日本の習慣とは逆である。そこで、日本らしいフォーマットに定義し直して表示するという作業が必要になる。

flastmodのタイムフォーマットを変えるためには、configコマンドを使う。たとえば、ssi/today.txtの更新日フォーマットを定義するときには、flastmod指定をする前に、config timefmtで時間表示フォーマットを指定する。その部分が、

```
<!--#config timefmt="%X"-->
```

である。同様に、ssi/tomorrow.txtの更新日フォーマットとしては、

記号	意味	記号	意味
%a	曜日（Sun、Mon…の省略形）	%p	午前、午後（am/pm表記）
%A	曜日（Sunday、Monday…の形式）	%P	午前、午後（AM/PM表記）
%b	月（Jan、Feb…の省略形）	%r	12時間表示時間（HH:MM:DD AM/PM表記）
%B	月（Januaryの形式）	%R	時間（HH:MM表記）
%c	日付（曜日 月 日 時間 年の書式）	%T	時間（HH:MM:SS表記）
%d	日（数字1〜31）	%u	曜日番号（日-土を0-6で表記）
%D	日付（MM/DD/YY）	%U	曜日番号（日-土を00-06で表記）
%e	日（数字1〜31）	%w	週番号（年頭の最初の日曜日を0とする）
%G	年	%W	週番号（年頭の最初の月曜日から数え、2桁表記）
%h	月（Jan…の省略形式）	%x	MM/DD/YY表記
%H	時（00-23の24時間形式）	%X	HH:MM:表記
%I	時（00-11の12時間形式）	%y	西暦下2桁
%j	1月1日からの経過日	%Y	西暦
%k	月（1〜12）	%z	付加時間
%m	月（01〜12）	%Z	タイムゾーン

表1　timefmtで指定する文字列

```
<!--#config timefmt="%m/%d %X"-->
```

を指定している。

timefmtで指定したパラメータは、%Xがファイルの時刻をHH:MM:SSで表示するフォーマットである。%mは、月を2桁表示し、%dは日を2桁表示する。ssi/today.txtの更新日表示指定は、HH:MM:SSの形式であり、tomorrow.txtの更新日表示指定は、MM/DD HH:MM:SSの形式になる。

timefmtで指定する文字列は、C言語の関数strftimeで定義されている文字列と同じである。man strftimeで参照できるが、代表的なものを**表1**にまとめておいた。

SSIを使ったファイルの時間表示は、サイトの更新日表示のときによく使われる方式だ。What's Newページの更新日時を自動挿入したり、メニューページに最終更新日時として挿入するといった使い方が多い。SSIを使わないと、各HTMLファイルを変更していかなければいけないが、SSIを使ってしまえば、更新日時を気にする必要がなくなる。ただし、よくある失敗が、修正をしていないファイルまでまとめてアップロードしてしまうようなときだ。このようなとき、実際は内容を更新していなくても、ファイルを上書きすることで、ファイルの更新日付が変わってしまう。特に、最近のよくできたHTMLエディタは、ローカルのドキュメントディレクトリをそっくりFTPするケースがある。余計なファイルの更新だけには十分気をつけたい。

CGIのようなアクティブなページを作る

SSIの欠点といえば、細かいプログラミング制御がきかないことだ。とはいっても、プログラミングで必須ともいえる条件分岐機能はSSIも持っている。ほかのプログラムやスクリプティング言語のように、ややこしい分岐処理はできないが、単純なものならばSSIでも十分だ。その例が、**図3**となる。これは、アクセス元のIPアドレスを調べ、それをもとに、192.168.1という文字列にマッチすれば社内ユーザーだとして、社内ユーザー向け特別ページのリンクを表示するというものだ。

```
<!--#if    expr="$REMOTE_ADDR    =
/^192.168.1./" -->
(表示する文字列、タグなどを記述)
<!--#endif -->
```

この部分の最初の行では、ブラウザが渡した変数REMOTE_ADDRの内容が、ローカルのプライベートアドレスである192.168.1とマッチするかどうかを探している。もしマッチしたときには、<!--#endif -->までの文字列を表示し、マッチしなっかたときには<!--#endif -->以降を処理する。マッチしなかった場合には、ブラウザでソースを閲覧してもこの3行は表示されない。

非常に単純なものだが、不要な人には不要なものを見せないというスタンスが重要だ。本来ならば、Perlあたりで記述することなるが、SSIでもできないわけではない。しかも、HTML中にそのまま記述できるため、メンテナンスの手間がかからない。SSIを経由すれば、ソースを閲覧しても非表示部分が見えないため、セキュリティレベルもちょっと上がる。

分岐では、

```
<!--#if ... -->
<!--#elif ... -->
<!--#else ... -->
<!--#endif ... -->
```

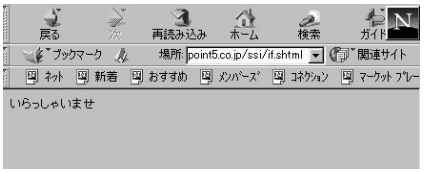
という条件分岐ができるため、多少は複雑なこともSSIで記述できる。

なお、/^192.168.1./という形でマッチ条件を指定している。スラッシュでくくった場合は、正規表現指定となる。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML><HEAD>
        <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=euc-jp">
        <TITLE>if</TITLE>
</HEAD>
<BODY>
        いらっしゃいませ<BR><BR>
        <!--#if expr="$REMOTE_ADDR = /^192.168.1./" -->
        <A HREF="secret/">社内ユーザ特別ページ</A>
        <!--#endif -->
</BODY>
</HTML>
```

図3　ifによる条件分岐の例
192.168.1.＊というプライベートの内部アドレスの時は、右上の画面となり、それ以外のアドレスからアクセスしたときは、右下の画面表示となる。

クライアントマシンが192.168.1.＊のIPアドレスからアクセスしたときの画面

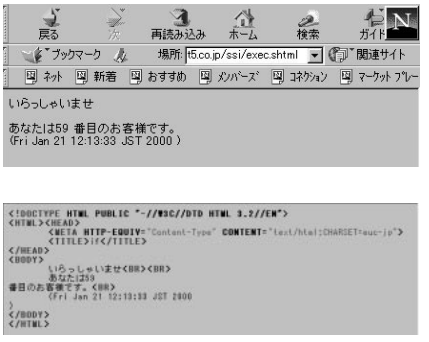


クライアントマシンが192.168.1.＊以外のIPアドレスからアクセスしたときの画面

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML><HEAD>
        <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=euc-jp">
        <TITLE>if</TITLE>
</HEAD>
<BODY>
        いらっしゃいませ<BR><BR>
        あなたは<!--#exec cgi="/cgi-bin/count.sh" -->番目のお客様です。<BR>
        (<!--#exec cmd="/bin/date" -->)
</BODY>
</HTML>
```

```
#!/bin/sh
echo $REMOTE_ADDR >>/tmp/count.txt
wc -l /tmp/count.txt | sed 's/^ *\([0-9]*\) .*/\1/g'
```

図4　execを使った例
左上のソースが元のHTML。左下がシェルスクリプトで作った簡易アクセスカウンタ“count.sh”。この結果が右上のHTML画面となる。



外部プログラムを実行させる

SSIの中でもっとも強力であり、しかも融通のきく機能が外部プログラムを動かし、その結果をHTML内に組み込むというものだ。ふつうはCGIプログラムで実現するのだが、SSIを使うと、HTMLファイルを読み込んだときに特定部分でプログラムが呼び出され、その結果を表示することができる。

図4がSSIから外部コマンドを呼び出した例である。HTMLからは、

```
<!--#exec cgi="/cgi-bin/count.sh"
-->
```

というように、/cgi-bin/count.shを呼び出している。この場合、動かすコマンドタイプはCGIであるため、Apacheのドキュメントディレクトリ、もしくはcgi-binディレクトリにあるものが実行対象となる。絶対パスを指定したときのトップディレクトリは、Apacheのドキュメントディレクトリであり、Apacheが管理するURI構造がそのまま受け入れられる。つまり、/bin/や/usr/local/binにある実行ファイルを動かそうとしてもできない。

これに対して、

```
(<!--#exec cmd="/bin/date" -->)
```

というように、#exec cmdで動かすと、/bin/shを通してLinux上の指定プログラム（この場合は/bin/dateコマンド）を起動することができる。すべてのファイルにアクセスできるわけだ。

NOEXECによるセキュリティ確保

SSIはたいていのISPでは利用できない設定になっている。その理由は、悪用目的で利用できるからだ。たとえばincludeを使い、/etc/passwdを公開できたり、#exec cmdで危険なプログラムを実行できる可能性を持っている。CGIが使えないようなところは、たいていSSIも使えないはずだ。

そうとはいえども、すべてのSSIを殺すことはない。flastmodによる更新日付表示まで使えなくなる必要はない。このために、IncludesNOEXECというオプションが指定できる。

IncludesNOEXECは、Option指定で追加するものだ。NOEXECで指定するには、httpd.confの315行で、

```
Options Indexes FollowSymLinks
IncludesNOEXEC
```

という形で指定すると、SSIのexec指定やinclude指定はすべて実行できなくなる。

これに加えて、.htaccessによる制限もかけておかないと、利用者単位で.htaccessを使ってSSIを有効にできてしまうので注意が必要だ。httpd.conf内で、

```
AllowOverride None
```

もしくは、

```
AllowOverride FileInfo AuthConfig
Limit
```

として、.htaccessによる処理を無効にするか、Optionの指定を無効にする。

とはいえ、制限をかければかけるほど、サーバは利用しにくくなる。ISPのように不特定多数の利用者にサーバを提供する場合にはSSIの利用は問題があるが、社内サーバや部内サーバの場合は、利用方法を徹底させて、制限をかけるよりも、使い方をレクチャーしたほうがよさそうだ。

Linuxの日本語環境

文：富樫 秀昭
Text：Hideaki Togashi

文字コード

この連載も、今回でとうとう最終回になる。ところが、「日本語環境」と題した連載にもかかわらずこれまで一度も文字コードの話に触れてこなかった。それを、ここでまとめておくことにしよう。

まとめるといっても、奥深い文字コードの話を網羅するゆとりもないし、そのつもりも毛頭ない。ここでは、これまで説明せずに使ってきた各文字コードについて、現在PC-UNIXを日常使ううえで必要と思われる最低限の説明をし、その扱い方について実践的な側面から触れて、終わりとするつもりである。文字コードについてより深い理解を求める向きには、文中で参考文献を挙げるので、そちらをご覧いただき、さらに探求を進めていただきたい。この記事がその足がかりを提供するものになるなら、幸いである。

文字コードとひとくちに言うが、実際には、利用される文字集合である文字セットを定義した規格と、その文字セットに対して実際に文字コードを割り当てるエンコーディング・メソッドを規定した規格と、大きく2種類に分けて考えることができる。まずは基本となる文字セットの規格から見ていこう。

文字セット

●ASCII、JIS X 0201、ISO 646

ASCIIについては特に説明は必要ないだろう。1963年に誕生し、1986年にANSIが規格化した米国で標準的に使われる文字セットである。7ビットで表現できる0x00-0x7fが用いられており、その中のさらに制御コードに割り当てた0x00-0x20、0x7fを除いた0x21-0x7eの94文字が**図1**のように割り当てられている。0x20の空白が制御文字であることに注意してほしい。0x7fはDELコードである。このASCIIが、すべての文字コードの基本となる。

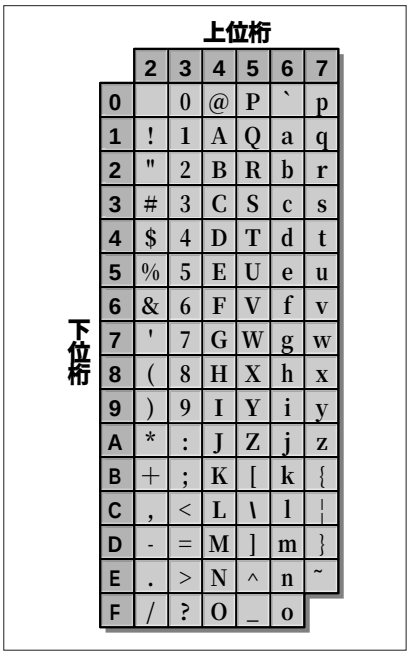
ASCIIは、必ずしも米国以外の国の実情に合ったものではない。1973年には、ISO 646でASCIIの領域のうち、0x23、0x24、0x40、0x5b-0x5e、0x60、0x7b-0x7eの12文字を、各国でよく使う文字に割り当てることを定義しており、多くの国でさまざまなバージョンが作られた。結果として、同一のコードでも環境の違いによって異なる文字として表示されてしまうことになるが、その取り扱い方を定めたのが後述のISO 2022であり、ISO 646と同じ1973年に制定されている。

ISO 646の日本語版は、

0x5cにASCIIの「/」の代わりに「\」
0x7eにASCIIの「~」の代わりに「‾」

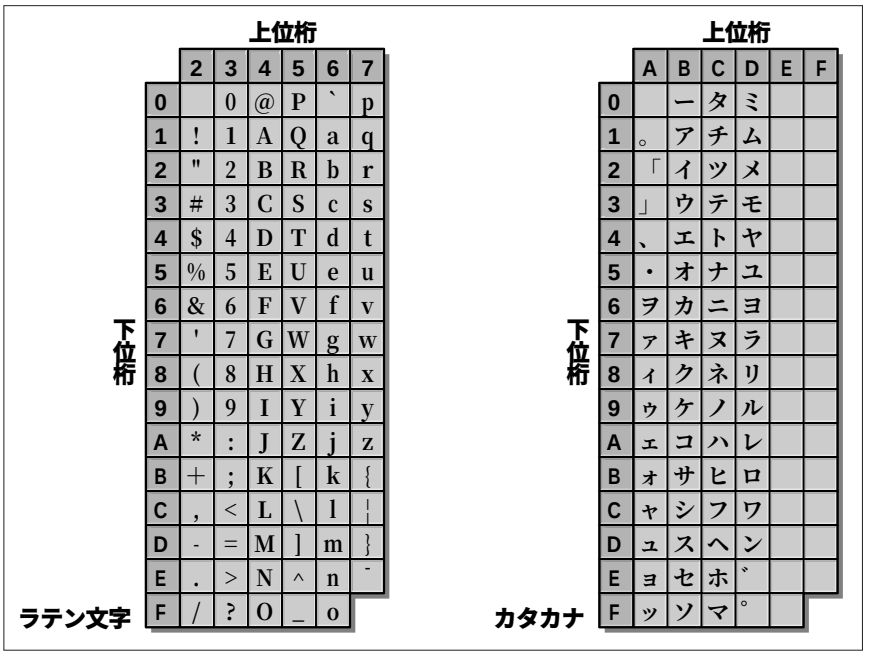
が使われている。

日本では1976年にJIS X 0201が定義され、これが日本の文字コードの基本となる（**図2**）。これは、ラテン文字集合とカタカナ文字集合を8ビットコードに割り当てたもので、ラテン文字集合をASCIIと重なる0x21-0x7eに割り当て、カタカナ文字集合は0x21-0x7eの最上位ビットを立てた0xa1-0xfeに割り当てたものだ。カタカナは94文字もないので、実際には0xe0-0xfeまでは未定義になっている。



下位桁＼上位桁	2	3	4	5	6	7
0		0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
A	*	:	J	Z	j	z
B	+	;	K	[	k	{
C	,	<	L	\	l	|
D	-	=	M	]	m	}
E	.	>	N	^	n	~
F	/	?	O	_	o	

図1　ASCII



下位桁＼上位桁	2	3	4	5	6	7
0		0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
A	*	:	J	Z	j	z
B	+	;	K	[	k	{
C	,	<	L	\	l	|
D	-	=	M	]	m	}
E	.	>	N	^	n	~
F	/	?	O	_	o	

ラテン文字

下位桁＼上位桁	A	B	C	D	E	F
0		ー	タ	ミ		
1	。	ア	チ	ム		
2	「	イ	ツ	メ		
3	」	ウ	テ	モ		
4	、	エ	ト	ヤ		
5	・	オ	ナ	ユ		
6	ヲ	カ	ニ	ヨ		
7	ァ	キ	ヌ	ラ		
8	ィ	ク	ネ	リ		
9	ゥ	ケ	ノ	ル		
A	ェ	コ	ハ	レ		
B	ォ	サ	ヒ	ロ		
C	ャ	シ	フ	ワ		
D	ュ	ス	ヘ	ン		
E	ョ	セ	ホ	゛		
F	ッ	ソ	マ	゜		

カタカナ

図2　JIS X 0201

JIS X 0201のラテン文字集合は、ISO 646の日本語版と同等であり、ASCIIとは異なっている。それにも関わらず、日本ではASCIIとJIS X 0201ラテン文字集合の違いを意識せずに利用することが多いようだ。

以上が1バイト256文字で表現できる文字セットである。

●JIS X 0208

2バイトで表現することによって、膨大な量の漢字をコンピュータで扱えるようにした規格が、JIS X 0208だ。漢字を含む日本語を表現する文字コードの中核となる文字集合で、すべての日本語の文字コードにこのJIS X 0208が含まれていると考えてよい。この規格にならって、アジア各国もそれぞれの文字セットを規格化している。

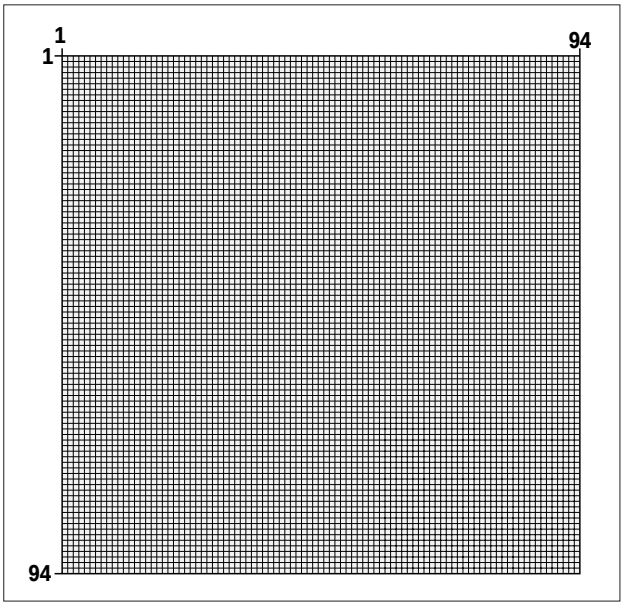


図3　区点コードの空間

1978年に、約6800文字からなる最初の規格JIS C6226-1978（通称：旧JIS）が公開され（後に改名）、1983年と1990年に文字の入れ替えと追加を含む大幅な改訂がなされている。1983年改訂への移行期にベンダーによって対応が異なるなどの混乱があり、現在でも旧JISを使う漢字コードがある。

JIS X 0208は文字セットを定義するのに、94×94のコード領域を用いている（**図3**）。94という数字がASCIIの文字数と一致するのは偶然ではない。1バイト目に相当する最初のコードを「区」、2バイト目に相当する後ろのコードを「点」と表現するので、この配置法は「区点コード」と呼ばれる。区点コードは、実際に文字コードとして使われることはないが、エンコーディングに関わらず文字を特定するには有用である。

このJIS X 0208に含まれる漢字のうち、16区から47区に配置されたものが第1水準の漢字であり、48区から83（後に84）区に配置されたものが第2水準の漢字である。1区から配置されているのは、記号・アルファベットなどである。

●JIS X 0212

1990年にJIS X 0208に定義されていない約6000文字を、JIS X 0208と同じ形式で定義したもので、「補助漢

字」と呼ばれる。JIS X 0208と同様に、記号・アルファベットなどが1区から、漢字が16区から配置されている。

日本語EUCとISO-2022-JPはこれに対応して拡張されたが、後述する理由でシフトJISではこの補助漢字を使うことはできない。そもそも利用頻度の高くない文字を集めたものであるためか、この文字セットそのものの利用頻度もあまり高くないものと思われる。

ちなみに、先ごろ選定されたJIS X 0213:2000 第3水準漢字、第4水準漢字には、JIS X 0212に含まれる文字も重複して定義されている。第3水準漢字、第4水準漢字はJIS X 0208と共に使うことが想定されているので、JIS X 0208の文字セットは重複して定義されることはない。

エンコーディング

JIS X 0208の文字セットを実際に文字コードとして用いるためのエンコーディング・メソッドは、主として3種類ある。それらについて触れる前に、これまで利用されてきたエンコーディング・メソッドの基本的な枠組みとなっているISO 2022について触れておこう。

●ISO 2022

1973年にISO 646と同時に制定されている。ASCIIの0x21-0x7eの領域とその最上位ビットを立てた0xa1-0xfeの領域に、必要な文字セットをエスケープ・シーケンスで呼び出して使うというのが、ISO 2022の発想である。それによって、常に複数の文字セットを切り替えて利用することができ、複数の文字セットで同一のコードに異なる文字を割り当てていても、そのコードで誤ってほかの文字を表示してしまうようなことがなくなるわけである。これによってISO 646の各国の文字セットを正しく扱うことができるようになるのだ。

当初ISO 2022はこの2つの領域だけを使うものであったが、その後拡張されて4つの領域を扱うことができるようになった。図4がその拡張版のISO 2022のエンコーディングの概念を示したものである。4つの領域を同時に扱うことができるようにするために、文字セットの呼び出しが2段階に分けられている。

現在使用中のコードの0x21-0x7eの領域を「GL」、0xa1-0xfeの領域を「GR」と呼ぶ。「G」はGraphic CharacterのGである。「G0」、「G1」、「G2」、「G3」に文字セットを呼び出し、さらにG0、G1、G2、G3をGL、GRに配置して使うのだ。文字セットをG0、G1、G2、G3に呼び出すエスケープ・シーケンスは、国際的な機関であるECMA（Standardizing Information and Communication Systems）が管理している。

G0はGLにしか配置することができず、配置するには「0x0f」という手続きが必要となる。G1はGLにもGRにも配置することができ、GLに配置するには「0x0e」、GRに配置するには「0x1b 0x7e」という手続きを行う。これらは、文字セットをまるごと呼び出して、その領域に新しい文字セットを配置するまで有効となるロッキングシフトである。

G2、G3を配置するのに通常使われる手続きはシングルシフトと呼ばれ、一瞬だけ配置するとすぐに元の配置に戻るというものだ。G2を配置するには「0x8e」もしくは「0x1b 0x4e」が使われ、G3を配置するには「0x8f」もしくは「0x1b 0x4f」が使われる。たとえば0x8eの直後はG2が配置され、次の文字コードにはG2の文字セットが用いられる。そしてその次の文字コードには0x8e以前にロッキングシフトで配置されていた元の文字セットが使われるという具合になる。

図4は8ビットの場合であるが、ISO 2022には7ビット

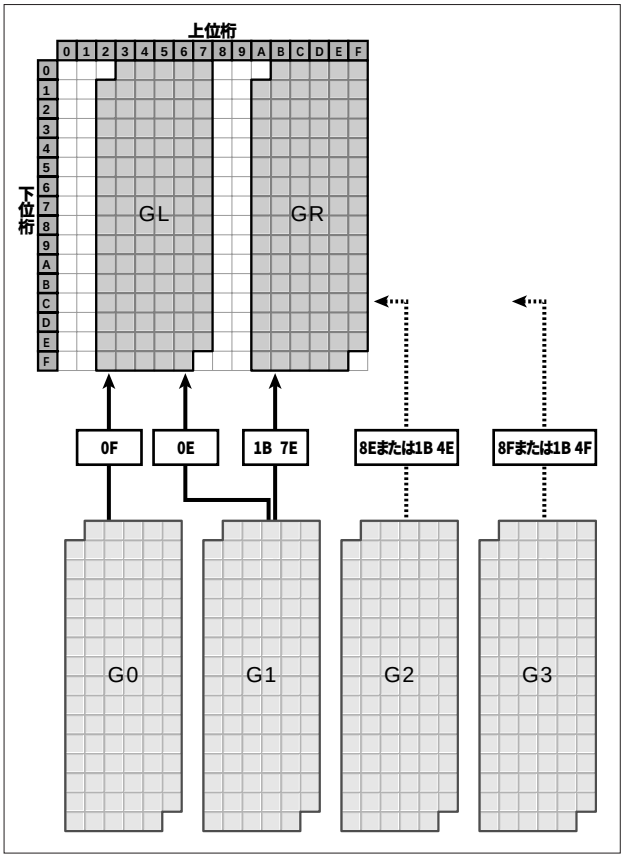


図4　拡張版ISO 2022

用のエンコーディングもあり、7ビットの場合はGRの領域は用いられず、G2、G3のシングルシフトには7ビットの「0x1b 0x4e」と「0x1b 0x4f」が用いられる。これで、すべての文字セットを7ビットだけで表現できることになるわけだ。

●JIS7ビットコードとISO-2022-JP

ISO 2022のエンコーディングにしたがってJIS X 0201ラテン文字集合（もしくはASCII）とJIS X 0208を切り替えて利用する文字コードのことを、一般にJISコードと呼んでいる。JISコードには8ビットのカタカナを用いるエンコーディングもあるが、よく使われるのは7ビットのカタカナを用いて、すべての文字を7ビットで表現したJIS7ビットコードである。

JIS7ビットコードは、ISO 2022の7ビット用のエンコーディングであり、G0を常にGLに配置した状態で用いるものだ。GLに配置されたG0に、適宜必要な文字セットを呼び出して利用するのである（**図5**）。G0への呼び出しには、以下のエスケープ・シーケンスが使われる（以下ESCは0x1b）。

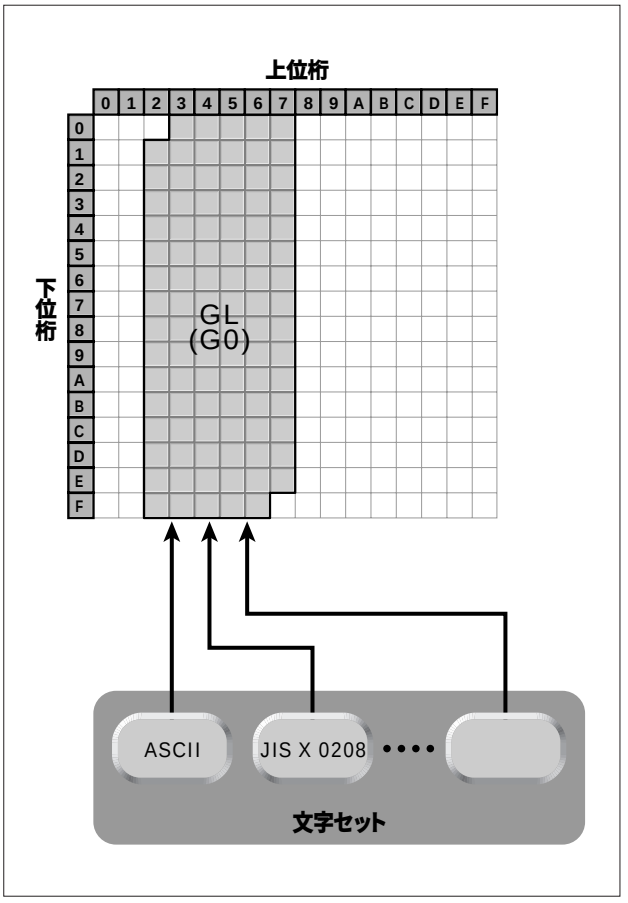


図5　JIS7ビットコード

文字セット	エスケープ・シーケンス
ASCII	ESC (B
JIS X 0201ラテン文字	ESC (J
JIS X 0201カタカナ	ESC (I
JIS X 0208	ESC $ B
JIS X 0212	ESC $ (D

実装によっては異なるシーケンスが使われる場合もある。

ISO-2022-JPは、日本のインターネット・メールの標準コードであり、RFC 1468として登録されたもので、JIS7ビットコードのサブセットである。以下の文字セットとエスケープ・シーケンスが使われる。

文字セット	エスケープ・シーケンス
ASCII	ESC (B
JIS X 0201-1976ラテン文字	ESC (J
JIS X 0208-1978	ESC $ @
JIS X 0208-1983	ESC $ B

通常は、ASCIIとJIS X 0208-1983を切り替えて使う。さらに、初期状態はASCIIが指示されているものとし、各文字列はASCIIで終わることになっている。JIS X 0208は、0x21-0x7eの領域で使われるため、区点コードのそれぞれに0x20を加えたものがそのまま使われる。よって、たとえば、

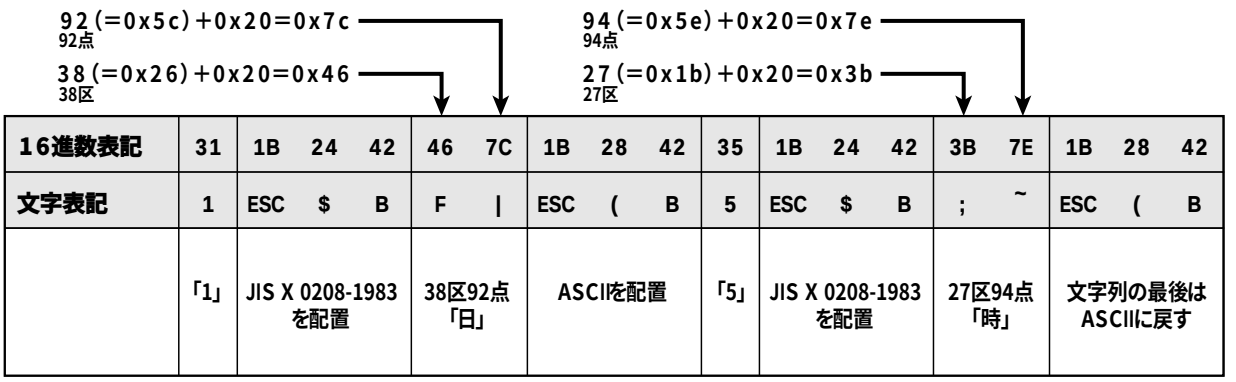
1日5時

は、

ASCIIの「1」
JIS X 0208の38区92点「日」
ASCIIの「5」
JIS X 0208の27区94点「時」

であるから、38区92点の「日」は、38（＝0x26）＋0x20＝0x46、92（＝0x5c）＋0x20＝0x7cで、27区94点の「時」は、27（＝0x1b）＋0x20＝0x3b、94（＝0x5e）＋0x20＝0x7eとなり、文字列全体は**図6**に示すコード列になる。

頻繁に登場するエスケープ・シーケンス「ESC $ B」と「ESC (B」は、それぞれ「漢字イン」と「漢字アウト」と呼ばれることもある。



16進数表記	31	1B 24 42	46 7C	1B 28 42	35	1B 24 42	3B 7E	1B 28 42
文字表記	1	ESC $ B	F |	ESC (B	5	ESC $ B	; ~	ESC (B
	「1」	JIS X 0208-1983を配置	38区92点「日」	ASCIIを配置	「5」	JIS X 0208-1983を配置	27区94点「時」	文字列の最後はASCIIに戻す

図6　ISO-2022-JPによる「1日5時」のコード列

JIS X 0212の制定にともない、前述のエスケープ・シーケンスに、

文字セット	エスケープ・シーケンス
JIS X 0212	ESC $ (D

を加えたISO-2022-JPの拡張版ISO-2022-JP-1が生まれている。また、JIS X 0212だけでなく、日本以外の国の文字コードもサポートするISO-2022-JP-2でも同じエスケープ・シーケンスが使われている。

ISO-2022-JPのコードマッピングを、**図7**に示した。7ビットですべての文字を表現できるところが特徴である。

●日本語EUC、EUC-JP

日本語EUCはUNIX SYSTEM Vのコード体系として考案されたEUC（Extended UNIX Code）に、日本語の文字セットを割り当てたものである。「UJIS（UNIXized JIS）」や「AT&T JIS」とも呼ばれる。SYSTEM V系だけでなく、事実上のUNIXの標準のコードセットとしての地位を築いている。JIS X 0212の制定にともなって、JIS X 0212補助漢字も用いることができるように拡張されたのが、EUC-JPである。

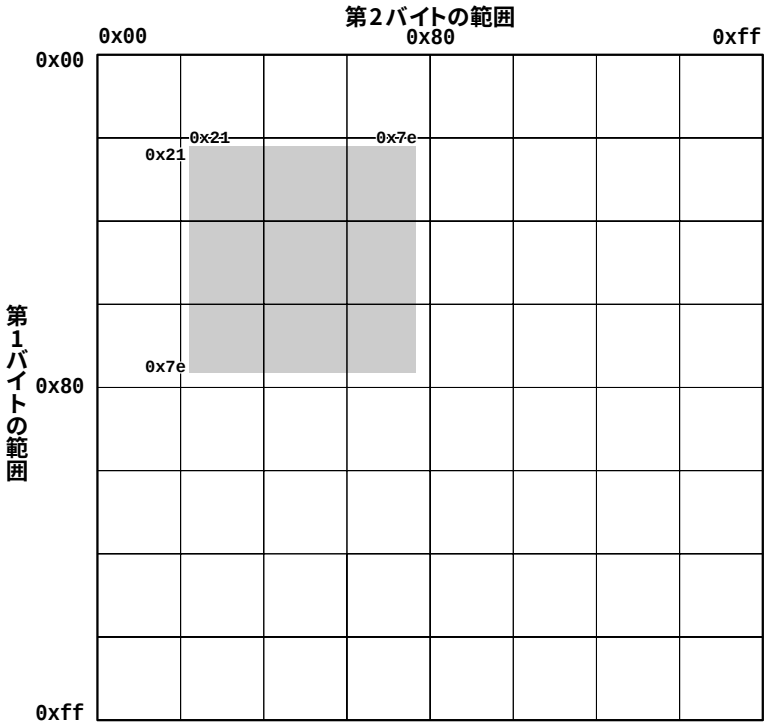


図7　ISO-2022-JPのコードマップ

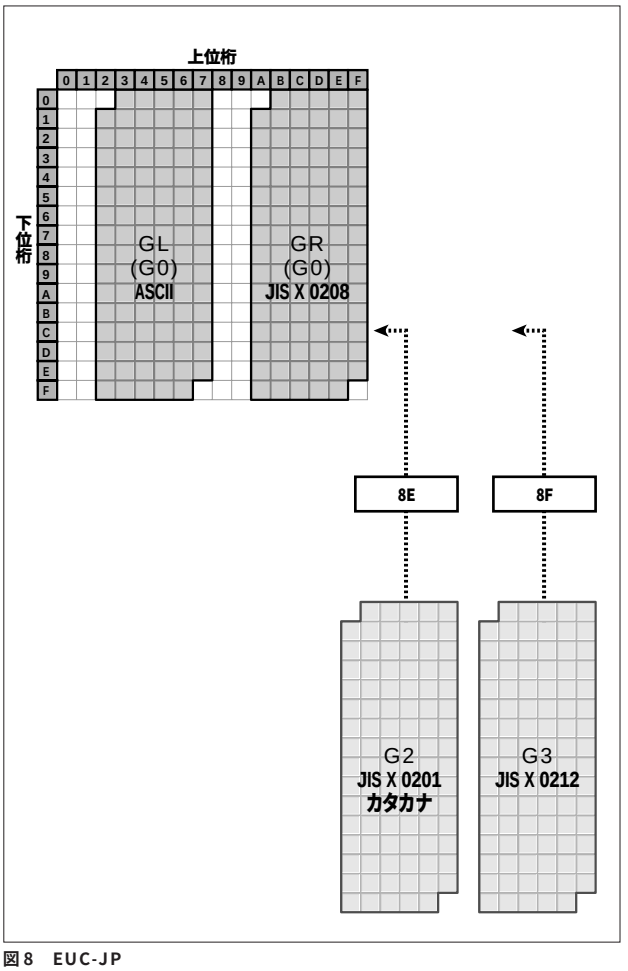


図8　EUC-JP

EUC-JPは、ISO 2022のG0にASCII、G1にJIS X 0208、G2にJIS X 0201カタカナ、G3にJIS X 0212を常に呼び出し、G0（ASCII）をGLに、G1（JIS X 0208）をGRに常に配置した状態で用いられるエンコーディングである。よって、エスケープ・シーケンスによるモードの変更は通常行われないが、G2のJIS X 0201カタカナ、G3のJIS X 0212補助漢字をシングルシフトでGRに配置するのに、それぞれ「0x8e」と「0x8f」が用いられる（**図8**）。

JIS X 0208は区点コードのままGRに配置される。つまり、区点コードのそれぞれに0x20を加え、さらにその値の最上位ビットを立てた値にマッピングされる。JIS X 0212も同様である。

よって、シングルシフトを含むすべてのASCII以外の文字は最上位ビットが立った形で用いられることになる。**図9**がEUC-JPのASCIIとJIS X 0208ののコードマッピングである。JIS X 0201カタカナはシングルシフトを含めて2バイトで表現され、1バイト目がシングルシフトの0x8e、2バイト目が0xa1-0xdfになる。JIS X 0212はシングルシフトを含めて3バイトで表現され、1バイト目がシングルシフトの0x8f、2バイト目と3バイト目にJIS X 0212の区点コードに0x20を加え、最上位ビットを立てた値が使われる。

たとえば、

1日5時カ

は、

ASCIIの「1」
JIS X 0208の38区92点「日」
ASCIIの「5」
JIS X 0208の27区94点「時」
JIS X 0201カタカナの「カ」
JIS X 0201カタカナの「ナ」

であるから、38区92点の「日」は38（＝0x26）＋0x20＋0x80＝0xc6、92（＝0x5c）＋0x20＋0x80＝

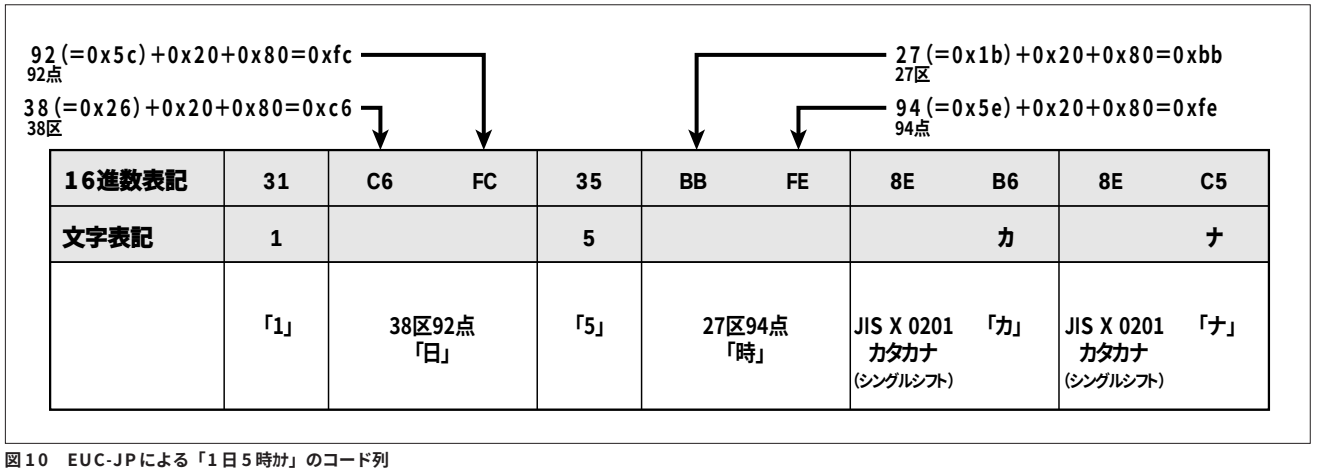


図10　EUC-JPによる「1日5時カ」のコード列

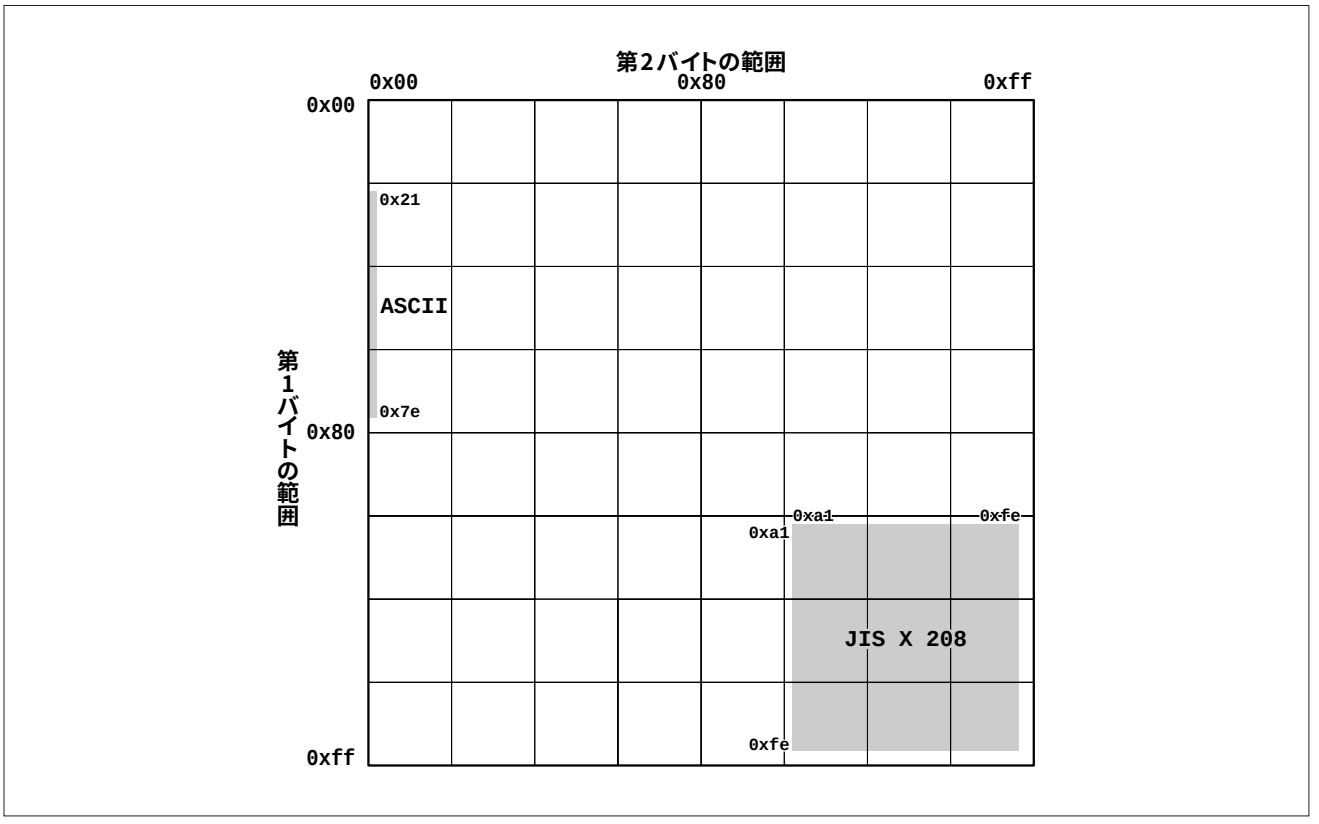


図9　EUC-JPのコードマップ（一部）

0xfcで、27区94点の「時」は27（＝0x1b）＋0x20＋0x80＝0xbb、94（＝0x5e）＋0x20＋0x80＝0xfeとなり、文字列全体は**図10**に示すコード列になる。EUC-JPのJIS X 0208は結局、JIS7ビットコードのJIS X 0208の最上位ビットを立てたものと言うこともできる。

●シフトJIS

シフトJISは、MS漢字と呼ばれることもあり、MS-DOSの流れを汲むPCのOSで広く使われている。また、Macintoshでも採用されている。JIS X 0208-1997の「附属書I」で規格化されているが、ISO 2022とは無縁の日本独自のローカルな規格である。

シフトJISは、JIS X 0201のラテン文字、カタカナしかなかった時代の資産を生かすために、JIS X 0201のラテン文字とカタカナを避けてJIS X 0208を配置したエンコーディングだ。

図11が、そのコードマッピングである。JIS X 0208の1バイト目がJIS X 0201のカタカナと重ならない領域に配置されている代わりに、2バイト目がASCIIの領域の一部と重なる形で配置されている。具体的には、0x40-0x7eの63個、つまりASCIIの約2/3と重なっている。

1バイト目をJIS X 0201のカタカナを避けるようにマッピングしたために仕方がなかったのだろうが、この変則的なマッピングのためにJIS X 0212補助漢字に対応することもできず、シフトJISを正しく解釈しない外国製のプログラムの多くで、このシフトJISの2バイト目の0x40-0x7eは問題になった。この点については後述するが、たとえば、MS-DOSのディレクトリ記号に割り当てられた0x5c（ASCIIのバックスラッシュ、JIS X 0201の円記号）の問題は有名である。UNIXでは、0x5cをエスケープ文字に使うことも多い。2000年1月20日に公示されたJIS X 0213:2000では、NEC選定IBM拡張文字部分、NEC選定NEC拡張文字部分、外字領域などのWindowsなどで利用されている部分が第3・第4水準漢字領域として使われている。シフトJISの今後が注目される。

参考文献

ここまでの部分について、主な参考文献を挙げておこう。この項を書くにあたっても参考にさせていただいた。

・Ken Lunde著「日本語情報処理」ソフトバンク、1995年
ISBN 4-89052-708-7

非常に広範にわたって、なおかつ詳細に日本語の処理について書かれている。

・清兼義弘・末廣陽一編著「国際化プログラミング I18N ハンドブック」共立出版、1998年

ISBN 4-320-02904-6

国際化プログラミングには欠かせない本であるが、日本語の文字コードについてもよくまとめられている。

・安岡孝一・安岡素子著「文字コードの世界」東京電機大学出版局、1999年

ISBN 4-501-53060-X

教科書として書かれたものだけに、煩雑にならずに必要な情報が体系的によくまとめられている。

日本語文字コードの処理

ここでは、ユーザーレベルで現在遭遇しうる問題を中心に見ていこう。

そもそも、米国で開発されASCIIを前提としたプログラムが日本語を正しく扱えないのは仕方がないにしても、かつては8ビットクリーンでさえないものが多かった。ASCIIは7ビットで表現できるため、最上位ビットをたとえば通信エラーを検出するためのパリティ・ビットに用いたりなどして、コードをマスクして7ビットだけを文字データ用に取り出すということが、よく行われていた。そういった処理を行わずに、8ビットのデータを正しく処理できることを、8ビットクリーンと呼ぶ。米国圏の最初の国際化といっていいだろう。経路中に古いメールサーバが1つでも残っていれば、そのMTAが8ビットクリーンではない可能性は依然としてある。8ビットコードを文字コードに用いる日本のような国ならそんなことはないかもしれないが、全世界を分散ネットワークで結ぶインターネットの世界ではどこにそういったホストが残っているかわからないという状況は無視できないのである。メールに7ビットのISO-2022-JPを用いることが勧められるのは、そのためである。

一方、通常の文字列処理では、7ビットコードはASCIIと重なるため、そのプログラムがエスケープ・シーケンスを正しく処理してくれなければ、逆にISO-2022-JPを用いたときにとんでもない結果が出力されかねない。8ビットクリーンなプログラムなら、日本語をすべて8ビットで表現してASCIIとは区別しているEUC-JPを使えば、日本語

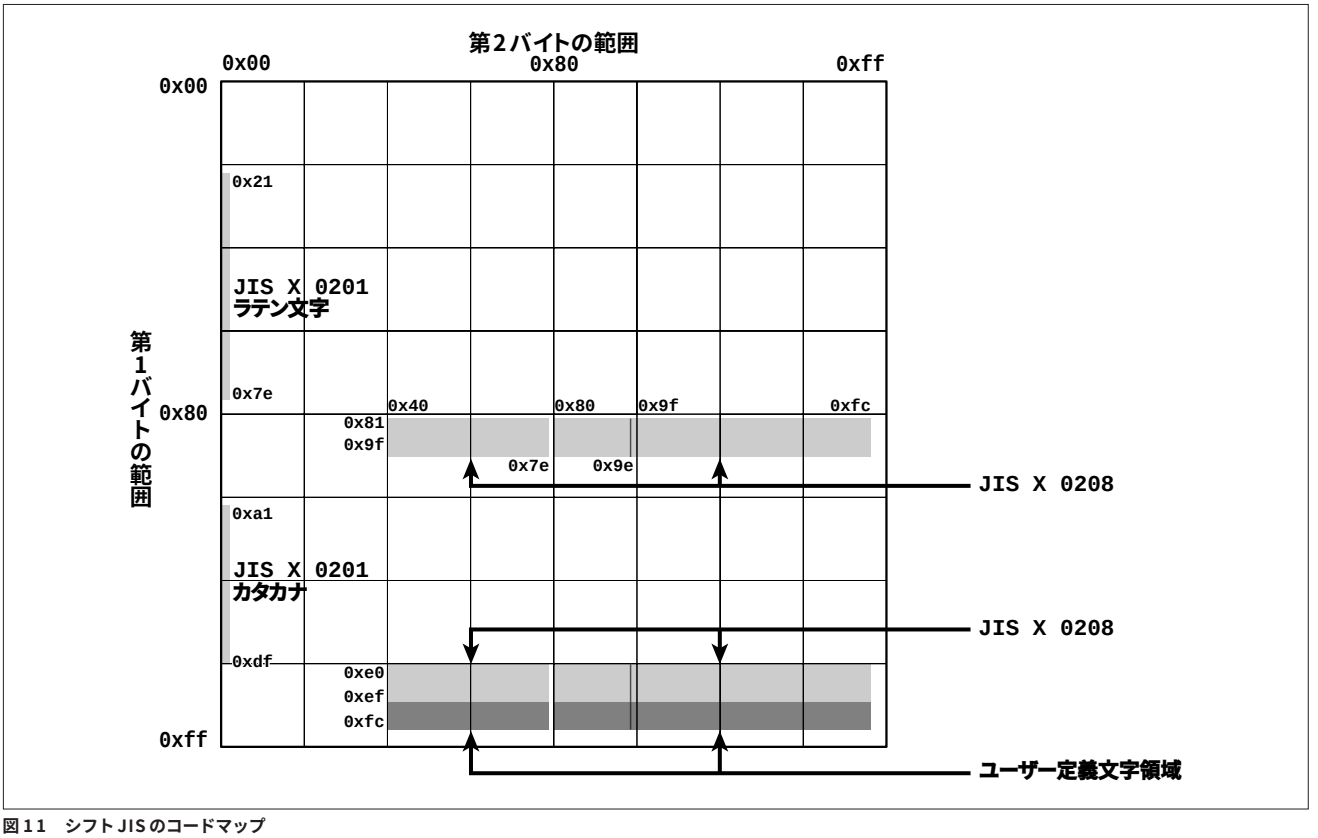


図11　シフトJISのコードマップ

文字列を正しく処理しないにしても最低限日本語をASCIIと誤認されることだけは避けることができ、なおかつ日本語は処理されずにそのまま残るのだ。実際に、EUC-JPを使えば、たとえ日本語対応していなくても8ビットクリーンであるために期待した結果を得ることができるケースが数多くある。これは、EUC-JPを用いる大きな利点と言えよう。それでも、多くのプログラムを組み合わせて使うことの多いUNIX上では、そのうちの1つのプログラムが8ビットクリーンでないだけで、全体の結果が正しくなくなってしまう可能性があるので、十分に留意する必要がある。

そういった場合に対処するためにも、現状では、最低限の文字コードの知識は、ユーザーの自衛手段として必須であるといわざるをえない。問題が文字コードによるものだとわかれば、対処の仕方もそれなりに決まってくる。筆者の経験上、問題の原因が文字コード処理のミスにあったというケースは、依然としてまだまだあるのだ。

特にシフトJISの扱いには、十分に注意が必要だ。漢字コードの2バイト目がASCIIの領域と重なるため、ことさら事故が起こりやすい。基本的に日本語を処理する際は、いったんEUC-JPに変換してから処理するのが無難だ。Perlの場合は、jcode.plを用いてEUC-JPに変換してから処理するのがよいだろう。シフトJISのテキストをやり取りする場合も、十分なチェックが必要になる。JIS X 0201のカタカナ、つまりいわゆる半角カナは基本的に扱えないものと考えて、特にチェックする必要がある。筆者もこれには何度も痛い目にあっている。そもそもISO-2022-JPでは、JIS X 0201カナはサポートされていないので注意してほしい。WindowsのFTPクライアントの中には親切なプログラムもあって、半角カナを全角カナに変換してくれるものもあるようだが、それに頼りすぎて別のクライアントで半角カナをそのまま渡してしまったり、知らずに全角カナに変換されてしまっていたりということもあるようだ。EUC-JPの項で説明したように、EUC-JPではこのいわゆる半角カナを扱うことができるのだが、実装によっては半角カナを表示できない場合もあるようだ。後述するように、文字コード変換ツールのnkfでも特にオプションで指定しない限り、半角カナはないものとして動作する。これらの変換ツールは非常に有用だが、その動作を理解しておくべきだろう。

●文字コード変換ツール

そこで、ここで文字コード変換ツールの使い方をいくつか紹介しておこう。

・Mule／emacs

文字コード変換ツールとして紹介されることはあまりないが、多言語エディタとしての豊富な機能を自在に操ることができるということは、文字コードを自在に操ることとイコールだろう。

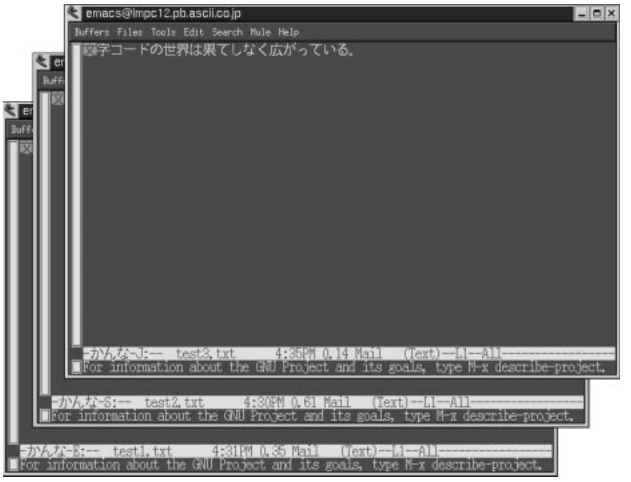
まず、モードライン左端近くに表示される文字コードの表示が重要である。これは、現在のバッファの文字コードを示してくれている。

```
E       EUC-JP
J       ISO-2022-JP
S       シフトJIS
```

となっている（**画面1**）。ついでになるがその右隣の表示は、改行コードを表している。テキストを扱う場合には、改行コードにも注意が必要だ。

ファイルの読み出し、書き込み時に「C-u」を先に打って文字コードを指定する技は、非常に便利である。めったにないことだが、C-x C-f（ファイルのオープン）でファイルを読み出した際に文字コードが間違って解釈された場合は、「C-u C-x C-v」で文字コードを指定して読み直す。そもそもMule／emacsで文字コードを誤って解釈された場合は、ファイルのどこかで文字化けが起こっている可能性を真剣に検討したほうがよい場合が多い。diredからファイルを読む際も、「C-u」を先に打って文字コードを指定することができる。

そして、文字コードの変換に使うことのできる技が、C-x C-w（ファイル名を指定するファイルの書き込み）だ。このコマンドも「C-u」を先に打つことによって、文字コ



画面1　emacs 20でISO-2022-JP、シフトJIS、EUC-JPのファイルを開いた。モードライン左端近くにそれぞれ「J」、「S」、「E」と文字コードが表示される

ードを指定することができる。これは立派な文字コード変換だ。

文字コードの指定に使うニーモニックは、emacs 20以降では変更されているが、古いものにせよ新しいものにせよ、覚えておく必要はない。わからなければその場でスペースキーを押して、表示されるリストの中から必要なものを使えばよいのだ。日本語の文字コードのニーモニックは、たいてい見当がつくだろう。文字コードのニーモニックの後ろについている「unix」、「dos」、「mac」は、改行コードを表すものだ。

・nkf

nkf（Network Kanji code conversion Filter）は、もっともよく使われている漢字コード変換ツールだろう。

```
$ nkf [option] filename.txt > outputfile.txt
```

もしくはパイプで、

```
$ cat filename.txt | nkf [option] > outputfile.txt
```

というように使う。

通常オプションには、出力する漢字コードを指定する以下のものが使われる。

-j	7ビットJIS（デフォルト）
-s	シフトJIS
-e	EUC-JP

先ほど触れたようにデフォルトでは、シフトJISの半角カナは入力には含まれていないものとして動作している。理由は自動認識の精度を上げるためだ。以下のオプションでシフトJISの半角カナつまりJIS X 0201カタカナを処理してくれるが、自動認識の精度は落ちることになる。

-X	入力に半角カナが含まれるものとして動作する。半角カナは JIS X 0208のカナ、いわゆる全角カナに変換される。
-x	入力に半角カナが含まれるものとして動作し、全角カナに変換しない。
-S	入力が半角カナを含むシフトJISであるものとして動作する。7ビットJISも入力できる。-xオプションなしでは、全角カナに変換される。

文字コードに関するちょっとおもしろいオプションとしては、

-Z	JIS X 0208の全角アルファベットをASCIIに変換する
-B	入力がエスケープ文字（0x1b）の欠けた7ビットJISであるものとして動作する。

などというものもある。

・iconv

最後に、XPG5のiconvコマンドを取り上げておこう。XPG5（X/Open Portability Guide 5）に準拠したシステムで提供されているコード変換ツールである。glibc2.2が実装される際には、Linuxでも標準で提供されることになるだろう。

```
$ iconv -f from_code -t to_code filename.txt \
   > outputfile.txt
```

もしくはパイプで、

```
$ cat filename.txt | iconv -f from_code -t to_code \
   > outputfile.txt
```

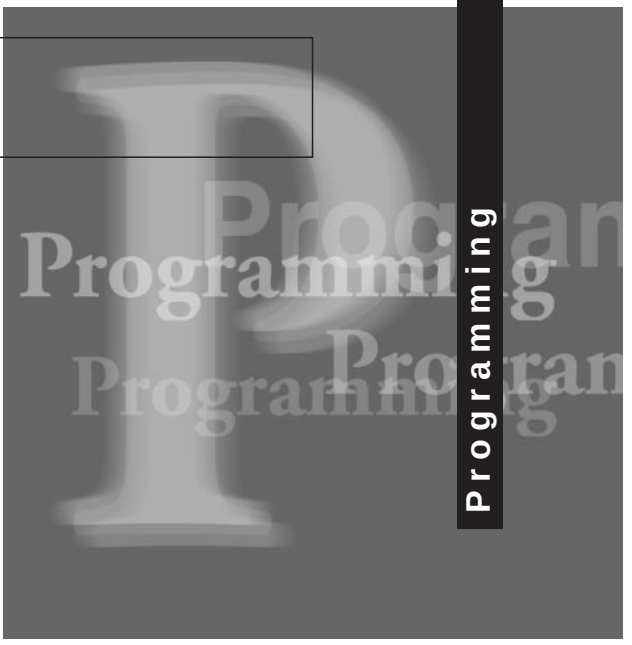
のように使う。文字コードのニーモニックは実装によって異なるが、たとえば、

```
$ iconv -f SJIS -t eucJP sjis.txt > euc.txt
```

という具合に使うことになる。文字コードの自動判定はしてくれない。

さて、これでこの日本語環境の連載も終わりとなった。そもそもこういった記事など必要なく、普通に日本語を使うことができるようになれば、それに越したことはないのだが、PC UNIXは残念ながらまだそこまで到達していない。苦労もあるかもしれないが、過渡的状況だからこその楽しみも、そこにはある。この楽しみは今でなければ味わえない楽しみでもあるのだ。読者諸氏のさらなるご活躍を願ってやまない。短い間ですが、ありがとうございました。

プログラミング工房



印刷はLinuxの不得意分野のひとつといえるだろう。しかし、ここは逆転の発想でWindowsが得意なものはWindowsに任せてしまうというのも、立派な解決策のひとつだ。というわけで、今回はWindowsでプリントサーバプログラムを動かし、Linuxから画像ファイルを印刷するプログラムを紹介する。

第5回　Windows経由のプリンタ利用

文：藤沢敏喜
Text : Toshiki Fujisawa

今月は、Windowsからしか利用できないプリンタ（ここでは「Windowsプリンタ」と呼ぶ）をFreeBSDやLinuxから簡単に利用するようなプログラムを作成する。具体的には、プリンタが接続されたWindowsマシン上にプリントサーバとなるプログラムを動かしておき、ネットワーク接続されたLinuxやFreeBSDマシンのコマンドラインからそのマシンにプリント命令を発行することで、画像などを簡単に印刷するというものである。

いつもなら、作成するプログラムの仕組みを個別に説明していくのだが、今回作成するプログラムは、今まで紹介してきたプログラムに比べるとサイズが大きいため、一部分だけをながめていても全体の動き方がわかりづらい。

そこで今回は、プログラムの詳細よりも、プログラミングを始める前の設計方法や、プログラムを作成するうえで考慮すべきこと、さらにプログラムを作ったあとにどのような改良を加えるべきか、といった重点を置いて解説していくことにする。例によって付属CD-ROMに、今回紹介するプログラムのソースコードがすべて収録されているので、具体的なコーディング方法が知りたい方は、記事を読み終わったあとに、そちらを参照してほしい。

Windowsを経由したプリンタの利用

一般的にLinux（というよりもUNIX）は、WindowsやMacintoshといった他のOSに比べ、マルチメディア系に弱いといわれている。各種ハードウェアやソフトウェアのサポートの数などから見ても、それはそのとおりといえる。しかし、そんなLinuxでも、昨今はデジタルカメラからの画像データの取り込み、フラットベッドスキャナやフィルムスキャナからの画像スキャンなど、マルチメディア（特にグラフィックス）に対応できるような環境が整いつつある。

これならWindowsやMacintoshと遜色ない、といいたいところだが、その前に壁がある。それは印刷だ。高価なPostScriptプリンタを持っている人はともかく、手が届きそうな価格のプリンタは、WindowsやMacintoshには対応していても、Linuxからは利用できないものが少なくない。Ghostscriptを利用するという手もあるが、Ghostscriptがサポートしているプリンタはあまり多くはない。そのため、せっかくLinuxに取り込んだ画像データも、いざ印刷という段になると、Windowsへ転送し、Windowsからプリントアウトするような面倒な作業が必要となる。こうなってくると、Linuxで画像データを扱う必然性そのものが怪しくなるというものだ。デジカメ全盛の昨今、これではLinux普及のブレーキ要因にもなりかねない。

これではまずいので、このような現状を改善するための解決策を考えてみた。よく考えてみると、プリンタは元々低速なデバイスであり、インタラクティブに使うものではない。したがって、プリンタがLinuxマシンに直接接続されていなくてもネットワーク経由で利用できればあまり問

題はないはずだ。それならということで、Windowsマシンをプリンタサーバとして利用し、ネットワーク経由でそのマシンに印刷命令を送りつけて、プリントアウトさせるという方法を思いついた。これならWindowsを意識せずに、Linuxから簡単に印刷できるはずだ。

もちろん、この方法はマシンが複数（少なくとも2台）必要になるが、筆者の自宅にはPCが何台もあるので、古いPCをプリンタサーバにしておけばよい。また、本誌のアンケートハガキの調査結果によると、PCを数台所有している読者も多いということらしいので、このプログラムを試せる方も多いはずだ。奥さん用のWindowsマシンにこっそりプリントサーバプログラムを動作させておくとか、古くなって使わなくなったノートPCをプリンタ専用サーバとして運用してもよいだろう。

作成する2つのプログラム

今回作成するのは、プリンタが直接接続されたWindowsマシン上で動作させるプリントサーバプログラム「bmplpd.exe」、このサーバプログラムとネットワーク経由で通信するクライアントプログラム「bmplpr」という、2つのプログラムである（**図1**）。bmplprは、LinuxとFreeBSDで動作することを第一目標としているが、そのほかのOSにも簡単に移植できるような形でプログラムを記述してある。

ちなみに、このプログラムの「bmplpd」と「bmplpr」という名前は、bmp、lpd、lprというキーワードに由来している。bmpはWindowsビットマップ形式のことで、lpd、lprは4.2BSDの時代に登場したプリント関係のコマンドである。ただし、今回作成するbmplpd.exeとbmplprは、lpdやlprが本来使用しているプロトコルではなく、筆者が定義した独自のプロトコルでデータをやり取りしている。これは、コードを単純にするためである。

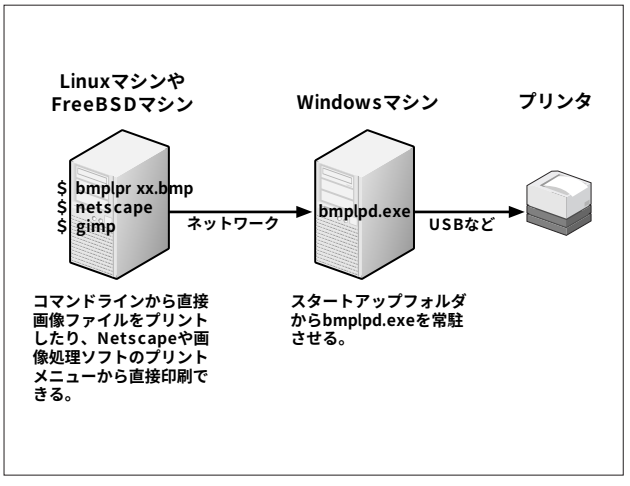


図1　今回作成するシステム

■使用方法

今回作成するシステムで、xxx.bmpという画像ファイルを印刷するときは、シェルのコマンドラインから

```
bash$ bmplpr xxx.bmp
```

または、

```
bash$ cat xxx.bmp | bmplpr
```

と、入力するだけである。

ただし、bmplprが直接扱えるのは24ビットのWindowsビットマップ形式（BMP）の画像だけ、という仕様になっていることに注意してほしい（これもプログラムを簡略化するためである）。といっても、Linuxにはいろいろな画像変換用のコマンドが標準でインストールされているので、画像形式や画像サイズの変換はさほど難しくない。たとえば

```
bash$ djpeg xxx.jpg  | ppmtobmp -Windows -24 | bmplpr
```

というようにパイプで接続することにより、さまざまな画像フォーマットの印刷が可能になる。この例では、JPEG形式のxxx.jpgという画像ファイルをdjpegコマンドでpnm（ppm）形式に変換し、さらにこのppm形式のファイルをppmtobmpコマンドで、最終的に24ビットカラーのBMP形式の画像データに変換してから、今回作成するクライアントコマンド、bmplprへ渡して印刷している。

さらに、次のように、

```
bash$ tifftopnm xxx.tiff | ppmtobmp -Windows -24 | bmplpr
bash$ pstopnm xxx.ps   | ppmtobmp -Windows -24 | bmplpr
```

tifftopnm、pstopnmコマンドなどを使用すれば、TIFF形式やPostScript形式の画像データも印刷できる。

また、最大印刷範囲が960×1280ドットのハガキ専用プリンタを使う場合は、NetscapeやGimpの印刷指定ウインドウで、Ghostscriptを呼び出し、

```
bash$ gs -q -g960x1280 -r120 -sPAGESIZE=a4 -dNOPAUSE
      -dBATCH -sDEVICE=ppm -sOutputFile=- -
      | ppmtobmp -Windows -24 | bmplpr
```

というように指定すれば、NetscapeやGimpで［ファイル］メニューの［印刷］コマンドをマウスで選択するだけで印刷することができる。

■BMP形式サポートのワケ

Win32APIを利用して印刷するときは、「DIB」（Device Independent Bitmap）と呼ばれる画像表現形式が必要となる。このDIB形式では、色数や圧縮の有無などを指定することができるが、24ビット（8ビット×3色）形式のものが多く使われる（**表1**）。このDIB形式に、**表2**に示すようなヘッダをつけたものがBMP形式である。

このため、入力にBMP以外の形式を採用してしまうと、DIBへの変換が必要になるため、アルゴリズムによっては、変換入力領域に130Mバイト（EPSONのプリンタ「PM-800C」でA4サイズのデータの場合）、変換出力領域に130Mバイト、計260Mバイトものメモリを費やしてしまうことになる。

一方、BMP形式を採用した場合は、ヘッダを取り去るだけでDIB形式にできるため、スワップのためのディスクスペースやアクセス時間も半分となる。

これらの理由から、今回のプログラムの入力形式としてBMP形式を選択した。

■通信プロトコル

bmplprとbmplpdの通信は、bmplprがbmplpdへ対してコマンドを発行し、bmplpdがそのコマンドを実行し、コマンド実行結果をbmplprへ返答するという流れで行われる（**図2**）。なお、このコマンドとその返答は、32ビットの符合つき整数で表される（**表3**）。

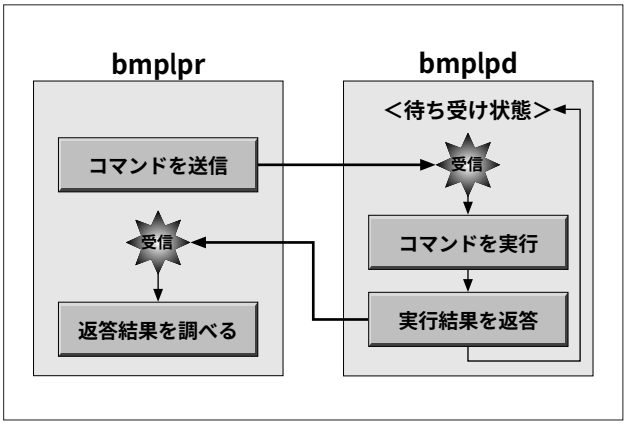


図2　通信の流れ

通信コマンドには、プリンタが印刷できるピクセル数を得るものや、サーバプログラムの終了やサーバマシンの停止、そして、DIB形式の画像イメージを印刷するものなど、4つを用意した。それらを次に示す。

1. 印字できるピクセル数を知る通信コマンド

Windows経由で印刷するプリンタの印刷可能なピクセル数をLinux上から得るときは、bmplprから通信コマンドとしてゼロ（32ビット）を、Windows上で動作しているbmplpdへ送ると、bmplpdが接続されたプリンタの印刷可能範囲を調べ、その水平方向のピクセル数（32ビット）と垂直方向のピクセル数（32ビット）を、bmplprに対して送信する。

2. bmplpdを終了させる通信コマンド

今回は、プリントサーバプログラムを簡単にするため、このプログラムを終了させるためのGUIは用意していない。そのため、bmplpdに“-1”（32ビット）を送れば、Linuxマ

フィールド	サイズ	説明
dib_header_size	32ビット	ヘッダのサイズ
width	32ビット	画像の横ピクセル数
height	32ビット	画像の縦ピクセル数
planes	16ビット	1
ビット_count	16ビット	色数（1、4、8、24ビット）
compression	32ビット	非圧縮時は0
size_image	32ビット	圧縮時のビットマップサイズ
x_pix_per_meter	32ビット	水平解像度
y_pix_per_meter	32ビット	垂直解像度
clr_used	32ビット	使われている色数
clr_important	32ビット	使われている重要な色数
image_data	8ビット×n	画像データ列

表1　Windows DIB形式

フィールド	サイズ	説明
magic	16ビット	BMという文字（BitMapの意）
file_size	32ビット	ファイル全体のサイズ
reserved1	16ビット	0（予約）
reserved2	16ビット	0（予約）
off_bits	32ビット	ファイル先頭からの画像位置

表2　Windows BMPファイル形式の先頭ヘッダ

通信コマンドの種類とその値	bmplpdがbmplprへ返答する値
最大印刷ピクセル取得（0）	X座標(32ビット)とY座標(32ビット)
DIBを送って印字する（DIBのサイズ）	0(正常終了時) または -1(エラー時)
bmplpdを終了させる（-1）	なし
サーバマシンの電源OFF（-2）	なし

表3　通信コマンドの種類とその値

シンからリモートでbmplpdを終了させられるようにした。

3. 電源を切るために、Windowsを終了する通信コマンド

このシステムでは、bmplpdを動かしているWindowsマシンをすべてリモートで操作できるようにしてある。そのため、このシステムを利用するために、Windowsマシンを直接操作する必要はない。Windowsマシンの電源切断も、“ -2 ”（32ビット）を送ればWindowsの終了処理が行われるようにしてある。

4. DIB形式の画像データを印刷する通信コマンド

上記の“ 0 ”、“ -1 ”、“ -2 ”以外の数値が送られた場合は、画像データを印刷するコマンドとして処理される。つまり、送られた数値は、印刷すべき画像（DIB）のサイズ（バイト数）として解釈されるのである。このコマンドでは、送信されたDIBサイズのメモリ領域をライブラリ関数mallocを用いて確保し、正常に確保が行われたかどうかをbmplprへ返信する（**図3**）。なお、プリンタによっては、この確保するサイズが数百Mバイトになることがあるので、メモリ確保が正常に行われたかどうかを確認することは、重要なポイントである。

メモリ確保が正常に行われた場合は、DIBの画像をbmplpdへ送信し、実際に印刷を行う。そして、正常に印刷が行われた場合は、bmplprに対して「0」を送信する。もしも、紙詰まりなどの理由で正常に印刷できなかった場合は「-1」を返信する。

インストール方法

今回のシステムで利用する2つのプログラム（bmplpd.exe、bmplpr）のソースコードは付属CD-ROMに収録してある。作成およびインストールの方法は、大まかに次のとおりである（/Linuxmag/Programmingディレクトリ のREADME.txtも参照のうえ、利用してほしい）。

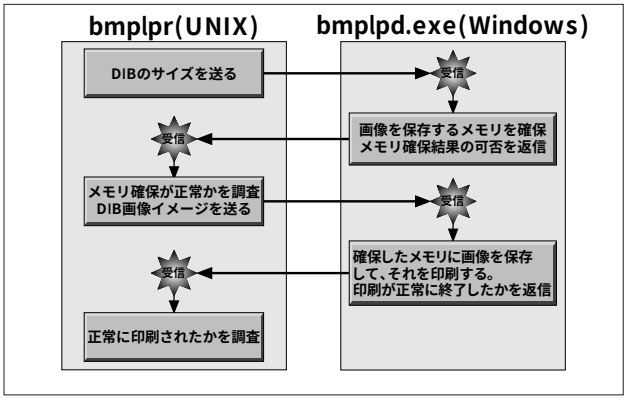


図3　DIBを印字する通信コマンド実行時の詳細

1. Mingw32のインストール

プリントサーバプログラム「bmplpd.exe」は、その名のとおりWindowsプログラムである。しかし、1月号の本連載で紹介した「Mingw32」というクロスコンパイラを利用すれば、FreeBSDやLinux上でもWindowsプログラムを作成することができる。付属CD-ROMに収録したソースコードも、このMingw32を利用して作成するようになっているため、コンパイルするマシンには、Mingw32がインストールされている必要がある。

2.bmplpr、bmplpd.exeのコンパイル

bmplpr、bmplpd.exeをコンパイルするには、付録CD-ROMをCD-ROMドライブに挿入し、次の手順どおりに行う。

```
$ su passwd
  ↑スーパーユーザーになる。passwdは正しいものを入力
# mount -t 9660 /dev/cdrom /mnt/cdrom
  ↑CD-ROMをマウントする
# cd /mnt/cdrom/Linuxmag/Programming/bmplpd-1.0.0/src
# make
```

これだけでコンパイルが始まるので、あとは終了するのを待つだけである。コンパイルが始まると、まずgccでコンパイルを実行したプラットフォーム用のクライアントプログラム「bmplpr」を作成する。これが終わると、次にMingw32（i386-mingw32-gcc）を用いて、bmplpd.exeのコンパイルを行う。最後に、おまけとしてWindows上で動作するクライアントプログラム「bmplpr.exe」のクロスコンパイルも行う。

ここで作成された3つのバイナリは、それぞれ

```
$(HOME)/bin/bmplpr
$(HOME)/smb/bmplpd.exe
$(HOME)/smb/bmplpr.exe
```

にインストールされる。

なお、このプログラムはLASER5 Linux 6.0やTurbo Linux 4.2上で正常にコンパイルできることを確認しているが、うまくコンパイルできなかった読者のために、付属CD-ROMのLinuxmag/Programmingディレクトリ下に

```
bmplpd-1.0.0/Linux/     Linux用実行バイナリ
bmplpd-1.0.0/FreeBSD/   FreeBSD用実行バイナリ
bmplpd-1.0.0/Win32/     Win32用実行バイナリ
```

が収録してある。

3. bmplpd.exeのインストール

Windows上で動作するbmplpd.exeのインストールは、プリンタサーバとするWindowsマシンの適当なディレクトリにコピーするだけである。また、スタートアップフォルダ（C:\Windows\スタートメニュー\プログラム\スタートアップ\）にbmplpd.exeをコピーしておけば、Windowsが起動するたびに自動的に実行されるので便利である。もちろん、スタートアップフォルダに入れずに、使用するときだけ起動しても構わない。

また、コンパイルしたLinuxマシンでsambaを動かしておき、$(HOME)/smb/bmplpd.exe へのショートカットファイルをスタートアップフォルダに置いても正常に動作する。この方法なら、ソースコードに変更を加えても再配布の手間を省くことができる。

ノートPCをプリントサーバとする場合は、ネットワークのプロパティで［Windowsログオン］を選択しておけば、ノートPCのフタを閉じたままの運用が可能になる。もし、Windowsログオン時にパスワードを聞いてくる場合は、\windowsフォルダにある、.pwlという拡張子のパスワードファイルをwindowsフォルダ以外に移動し、次回リブート時にパスワード欄に何も入力しないで［OK］ボタンをクリックすれば、起動時にパスワードを聞いてこないようになる。

4. bmplprのインストール

次に、クライアントプログラム「bmplpr」をLinuxマシンへインストールする。これも、bmplprをPATHの通った適当なディレクトリ（/usr/local/binなど）にコピーし、実行ビットを立てる（$ chmod +x bmplpr）だけである。なお、付録CD-ROMのMakefileでは、$(HOME)/bin/にインストールするように設定してあるので、このまま使用するなら、$(HOME)/binを環境変数PATHに加えておくことをお勧めする。

インストールが終わったら、環境変数BMPLPD_SERVERにWindowsマシンの名前（または IPアドレス）をセットする。bashを使っているなら

```
$ export BMPLPD_SERVER=printserver.ascii.co.jp
                    ↑適切なマシン名を値として入力する
```

Column

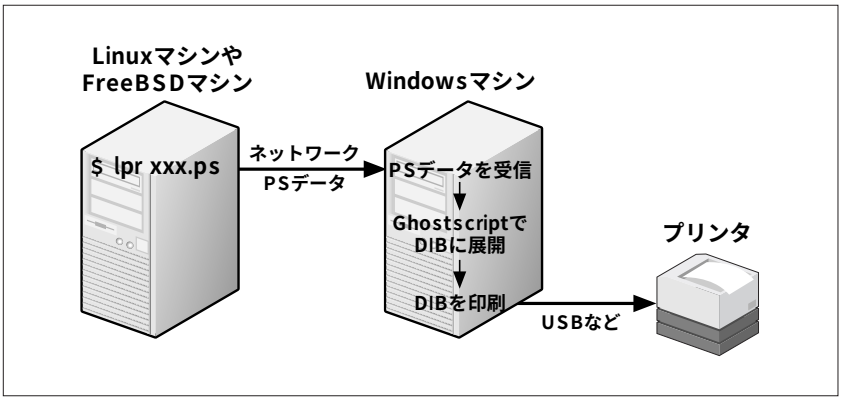
PostScriptプリンタの作成

今回作成したプログラムは、プログラムを簡単にするため、クライアント側でビットマップイメージを用意し、それをプリントサーバ側へ送るという手法を用いている。この場合、ネットワークに転送するデータ量が非常に多くなってしまい、高解像度のプリンタなどでは、プリントアウトに非常に時間がかかってしまう。

これを解決するためには、ネットワークに送るファイル形式をPostScript（PS）ファイルにしてしまうという方法が考えられる（**図C-1**）。つまり、Windows上でGhostscriptを動作させ、ビットマップに展開するというわけである。GhostscriptにはWin32版もあるので、先月号で紹介したwingpg.exeのような開発手法（main関数の呼び出し）を用いれば、単一のオブジェクトとすることも可能だろう。

また、今回紹介したような独自プロトコルではなく、LPDなどのオープンなプロトコルを用いれば、ネットワーク越しに見る限り、完全なPSプリンタとすることもできるだろう。

UNIXでは、PSプリンタが前提となっていることが多いので、この方法が実現できると非常にうれしい。筆者は、組み込み用の小型PCにWindowsを載せたプリントサーバ専用マシンを作れば、けっこう需要があるかもしれないと思っている。



図C-1 WindowsプリンタをPSプリンタに変身させる

また、環境変数を設定せずに、bmplprの実行時に-sオプションを利用して、

```
$ bmplpr -s printserver.ascii.co.jp xxx.bmp
```

と指定することもできる。出力先を複数用意して、切り替えながら使用するならこの指定方法のほうが便利だろう。

5. bmplprのテスト

インストールが終了したら、コマンドラインから

```
$ bmplpr -g
  ↑bmplprのbmplpd.exeへの接続テスト
2360 3387
  ↑正常なら標準プリンタが印刷可能なピクセル数を表示
```

と入力する。正常に動作していれば、次行にWindowsマシンに接続されているプリンタの印刷可能な横方向と縦方向のピクセル数が表示される（このピクセル数は、プリンタの用紙や、出力モードによって変化する）。

この数値が表示されない場合は、Windowsマシンのコントロールパネルで、接続されているプリンタが「通常使うプリンタ」となっているかどうかを確認してほしい。この値は、［プリンタ］アプレットで設定したデフォルトのモードが設定されるからだ。

6. bmplprを使ったプリントテスト

次に、適当な24ビットカラーのBMP形式の画像ファイルを用意し、

```
$ bmplpr xxx.bmp
```

とすれば、紙の中央に画像が印刷されるはずだ。もしも、“only support 24bit color”というメッセージが出るときは、前述の変換コマンドなどで、24ビットカラーのBMP形式に保存し直す。また、“too large image”というエラーメッセージが表示されるときは、印字可能なピクセル数を超えた画像を印刷しようとしているので、データの大きさを変更する。

また、もしもうまく動かないようなら「bmpdbg.exe」

Column

UNIXでのプリンタ事情

UNIXの印刷では、PostScript（PS）形式が標準である。PS形式をそのまま出力できるPSプリンタなら何も問題はないが、残念ながらPSプリンタは高価のものが多い。そのため、個人で使うなら、非PostScriptプリンタ＋Ghostscriptを使うのが現状では一般的な方法だろう。また、TeXを利用している人なら、DVIファイルを印刷するためにlips3dviコマンドを使っているかもしれない。

しかし本文でも述べたように、Ghostscriptが対応しているプリンタはあまり多くはない。LIPSもCanonのプリンタだけにしか使えない、閉じた世界のものなのだ。さらに、Ghostscriptで対応しているインクジェットプリンタであっても、低解像度の文字なら何とか印刷することができるが、7色のインクを使うような最新のフォト画質の印刷はできない。

MS-DOSが主流だったころは、EPSONが提唱したESC/Pのようにプリンタを制御するコマンドが公開されることが多かった。しかし、Windowsが普及するようになると、プリンタの制御コマンドが公開されなくなってしまった。これは、Windowsが市場を独占するようになった結果、メーカーは1種類のOSのプリンタドライバを作成すればよくなり、公開の必要性が薄れたという事情が大きいのだろう。

また現在、インクジェットプリンタの分野は、非常に激しい開発競争が繰り広げられる激戦区となっている。そのため、他社との競合上の理由から、画質を左右する重要な情報であるプリンタドライバのノウハウを公にしたくないという事情もあるだろう。さらに、最近ではOEM供給によってプリンタを販売している会社も多く、販売会社がLinuxへ対応させたくても、OEM元の事情によりできないこともあると思われる。

現在、筆者の自宅には、Canonのレーザープリンタ「A404G/II」とEPSONのインクジェットプリンタ「PM-800C」、そしてKonicaのハガキ専用昇華プリンタ「Q-PRINTER」という3台のプリンタがある。

A404G/IIは、Windows普及以前のもので、制御コマンド（LIPS3）が公開されているため、Ghostscriptから印刷が可能である。しかし、最近販売されている安価のレーザープリンタのほとんどは、制御コマンドが非公開となってしまっている。PM-800Cは、文字印刷程度はGhostscriptで使えるものの、フォト画質の制御方法はやはり非公開である。Q-PRINTERの場合は、ローカルプリンタポートだけの対応（ネットワーク経由で使うことができない）である。このようなプリンタは、双方向プリンタポートを特殊なモードで使用しているのではないかと思うので、FreeBSDやLinuxから直接使うのは絶望的である。

しかし、これらのプリンタ3台ともすべて、今回本文で紹介したプログラムを利用して、問題なくプリントアウトが行えているので、使えないとあきらめている自宅や会社のプリンタもよみがえらせることができるかもしれない。特に、フォト画質の印刷を頻繁に行うユーザーにとっては、かなり便利になると思う。

という、BMPファイルを直接印刷するコマンドも付属CD-ROMに収録してあるので、MS-DOSプロンプトから

```
C:¥>bmpdbg xxx.bmp
```

としてみれば、問題がネットワーク経由した場合にあるのか、それとも別のところにあるのかが、突き止められるはずだ。これが正常に動作しない場合は、bmpdbg.cやos.cを調べてみるとよいだろう。

他のオプションと注意点

プリントサーバプログラム、bmplpdを終了させるには、Linuxマシン上から

```
$ bmplpr -q
```

と-qオプションを送る。なお、bmplprが接続したままの状態では、Windows上のbmplpd.exeを削除することができないので、その際もこのコマンドを利用すればよい。

また、

```
$ bmplpr -z
```

と、-zオプションを送ると、bmplpd.exeを動かしているWindowsマシンを終了させることができる（ただし、この機能はWindows NTでは正常に機能しないことがある）。この機能は、ノートPCなどをプリントサーバとしている場合などに、ノートPCを操作しなくても（　フタをあけなくても）電源を落とすことができるので便利に使えると思う。

Windows上で動作するbmplpr.exe

前述のとおり、コンパイルの際、Windows上でも動作するクライアントプログラム「bmplpr.exe」も同時に作成する。これを使えば、Linuxだけでなく Windowsからもbmplpd.exeにプリント命令を送ることができる。

これを利用するには、次のように環境変数BMPLPD_SERVERにbmplpd.exeを動作させているWindowsマシンの名前（あるいはIPアドレス）を設定しておく。

```
c:¥>set BMPLPD_SERVER=printserver.ascii.co.jp
```

ファイル	内容	行数
os.h	OS依存関数のヘッダ	57
os.c	OS依存関数	879
prot.h	通信コマンドの定義	6
winbmp.h	BMP構造体の定義	60
bmplpd.c	bmplpdデーモン	143
bmplpr.c	bmplprコマンド	334
	合計	1,479

表4　プログラムファイル行数と内容

autoexec.batなどを利用しておけば、起動時に自動で設定できる。実際に印刷するときは、MS-DOSプロンプトから

```
c:¥>bmplpr xxx.bmp
```

として印刷したり、bmplprのアイコンにBMPファイルをドラッグ&ドロップして印刷することもできる。

改造のポイント

表4に示したように、今回のプログラムの行数は全部で1479行あり、これまでの連載で扱ってきたものと比べると比較的規模の大きなものとなっている。しかし、メインとなるbmplpd.cはわずか143行で、bmplpr.cも画像を送るsend_dib_to_printer関数はたったの65行しかない。今回は紙幅の関係で、プログラムの詳しい解説をすることができなかったが、前述した通信方法の部分を頭に入れてソースコードを読んでいけば、理解はさほど難しくないはずだ。ぜひ、ソースを読んでみてほしい。

なお、今回紹介したシステムは、あくまでもプログラミングの習得を目的としたサンプルであり、実用性を追求したものではない。そのため、当然ながらすべてのプリンタで動くということは保証はできない。もしも、自分の持っているプリンタでうまく動かないなら、原因を調べてソースコードを改造するのもいい。そのためにソースコードがついているのだ。

セキュリティについてもほとんど考慮されてない。もしも、会社のように多くの人が使っている環境で使うなら注意が必要である。なぜなら、-zオプションを利用して、悪意的にWindowsマシンの電源を落としたり、知らない間に勝手にプリンタが使われてしまったりという状況が考えられるからだ。これらを回避するには、IPアドレスによる接続制限やパスワードをつけるなどの改造を行なえばよい。

さらに、より実用的なプログラムとするためには、タイムアウト処理、紙詰まり処理、キャンセル処理、そして複数ページの対応なども必要になるだろう。

特別講座

ステップアップC言語

OSに依存しないプログラミング

いろいろなプロットフォーム上でコンパイルできるようなプログラムを作成するにはどうしたらよいだろうか？

プロットフォームが異なるということは、CPUによって数値がメモリに格納される順番（endian）が異なっていたり、コンパイラによってintのサイズが違ったりすることもある。

また、OSによって使うシステムコールの名前や引数が違っていたり、ネットワークを使う前に謎の呪文を唱えなければいけないOSもある。

このような問題について、今回作成したプログラムでの解決方法について考察してみる。

■CPUやコンパイラによる違い

Pentiumなどで数値をメモリに格納するときは、「Little Endian」（バイト0を最下位バイトにする）と呼ばれるメモリ配置を用いているが、PowerPCでは、「Big Endian」（バイト0を最上位バイトにする）と呼ばれるメモリ配置になっている。

メモリブロックをネットワーク越しに送信するとき、このendianの違いが問題になるため、ネットワークバイトオーダーに変換して送信する必要がある。今回のプログラムで定義した、os_net_write_long(os.c)関数では、ライブラリ関数htonl(Host TO Net Long)を使い、ホストマシンのバイトオーダーからネットワークバイトオーダーに変換している。

また、C言語のint型が何ビットであるかというのはコンパイラに依存する。現在ではint型が32ビットであることがほとんどであるが、DOSの時代は16ビットであったし、将来int型が64ビットとして扱われることもあるだろう。

したがって、移植性を重視するプログラムではint型でなく、サイズが明示できる、short型（16ビット）やlong型（32ビット）などを使うことをお勧めする。

■標準ライブラリ関数の違い

標準ライブラリ関数に関してもプラットフォーム間で次のような違いがあるので、注意が必要である。

まず、ファイルをオープンする際に利用するときは、UNIXではopen関数を使うが、新しいWin32ではCreateFile関数を利用する。また、古いWindowsでもコンパイルできるようにするためには_lopen関数を使う必要がある。

このような場合分けに対応するには、C言語のマクロ機能を利用すればよい。たとえば、FreeBSDやLinuxのgccでは「UNIX」というマクロが定義されており、Mingw32環境では「WIN32」というマクロが定義されている。これらを利用して、リストc-1に示したような書き方をしておけば、どちらの環境でもソースを書き換えずにコンパイルをすることができる。

しかし、ソースコードが大規模なものになってくると、open関数のように使用頻度の高いものは、ソースコードのいろいろな場所で使われることになる。その場合、その都度#ifdef文による切り替えをしてしまうと、ソースが非常に読みづらいものになってしまう。

そのため、今回はos.cというファイルで、

```
fd_t os_file_open( char *fname );
```

という関数を定義し、この中でのみ標準ライブラリ関数を使用している。したがって#ifdef文で切り替える必要があるのは、os_file_open関数を定義するos.cの中だけとなる。

そのため、新しいOSへの移植時には、os.cのみを書き換えればよく、os.c以外のファイルを書き換える必要がなくなる。

また、os.h、os.cに標準ライブラリ関数を隠蔽すると、

```
#include <windows.h>
#include <linux/blkdev.h>
```

などの各OSに依存するヘッダファイルもos.cの中に閉じ込めることができる。

■OSの能力の違い

Windowsでは、ネットワーク関係の関数(winsock.dll)を使用する前に、

```
WSAStartup(version, &wsa_data)
```

を呼び出す必要がある（もちろんUNIXではこんな面倒な呪文は必要ない）。

しかも、Winsockは、BSDのsocketインターフェイスを真似しているにも関わらず、制限が多くてプログラムが作りづらい（このあたりは、非力なcommand.comと、強力なbashというコマンドインタプリタの違いにも似ている）。

また、Winsockだけでなく、表示やキー入力なども、WindowsとUNIXでは大きく異なるため、同じようなコードで記述するのは難しい。

しかし、アプリケーションから呼び出す関数仕様を統一し、その関数の中でのみ標準ライブラリ関数を呼ぶようにすれば、どんなOSでも使用できるアプリケーションを書くことができる。

Linuxの広まりとともに、Linux用のソフトウェアが多く出回るようになってきたが、Linuxだけでしかコンパイルできないような閉じたプログラムも散見されるようになった。

せっかくプログラムを書くからには、FreeBSDやNetBSD、そして可能であればWindowsなど、さまざまなOSで使えるような移植性のあるプログラム構造にしたいものである。

▼リストc-1 マクロ定義によるプラットフォームの切り替え

```
fd_t
os_file_open( char *fname )
{
 #ifdef unix
  return open(fname, O_RDONLY );
 #endif

 #ifdef WIN32
  return _lopen(fname,(int)OF_READ);
 #endif
}
```

PostgreSQLを極める

これまで紹介してきたPL/pgSQLやトリガは、たしかに便利な機能ですが、データベースそのものがしっかり運用／管理されていないようでは、これらの便利な機能も意味がありません。そこで、今回はPostgreSQLの運用や管理に役立つ情報をご紹介します。

第5回 運用と管理 ―基礎編―

文：片岡裕生
Text : Hiroki kataoka

前回までは、主にPostgreSQLの便利な機能について紹介してきました。しかし、データベースを利用していくためには、運用や管理について知っておくことも必要になります。しっかり運用／管理がなされていないデータベースは役に立たないからです。今回は機能の解説をちょっとお休みして、PostgreSQLの運用や管理で知っておくと便利なことをいくつか紹介します。

PostgreSQLの構成

PostgreSQLをインストールすると、標準では/usr/local/pgsqlディレクトリにいくつかのサブディレクトリが作成され、その中にPostgreSQL用の各種コマンドや設定ファイルなどが格納されます。標準の構成でインストールした場合のディレクトリ構成を**図1**に示します。デフォルトのデータベース領域もこの中に含まれています。

pgsql/binディレクトリには、コマンドラインからSQL文を実行できるpsqlコマンドなど、PostgreSQL用の各種コマンドが格納されており、さらにPostgreSQLのサーバ本体であるpostmasterプログラムとpostgresプログラムも格納されています。

いきなり話が横道にそれますが、なぜPostgreSQLのサーバプログラム本体がpostmasterとpostgresの2つあるのかご存じでしょうか？　実は、PostgreSQLのサーバプロセス構成は**図2**のような構成になっており、PostgreSQLの初期化とクライアントからの接続要求の待ち受けを担当するマルチユーザーバックエンドのpostmasterと、単一のクライアントからの処理要求を実行するシングルユーザーバックエンドのpostgresにそれぞれ役割分担されています。最近のバージョンでは、この2つのプログラムはまったく同じもの（postmasterはpostgresのシンボリックリンク）なのですが、歴史的経緯もあって分けて考えるようになっています。

さて話を元に戻しまして、次にpgsql/libディレクトリを見てみましょう。このディレクトリには、主にPostgreSQL本体やコマンドの実行時に必要となるライブラリファイルなどが格納されています。しばしば「PostgreSQLがうまく動かない」という話を耳にするのですが、このライブラリディレクトリの場所が指定されていない（もしくは間違っている）ために、プログラムの実行時にライブラリファイルがロードできず、エラーになってしまっている場合が多いようです。

pgsql/includeディレクトリは、PostgreSQL用のアプリケーションを開発する際に必要となるC言語用のヘッダファイルが格納されています。C言語からPostgreSQLアクセス用のライブラリである“libpq”を利用する場合には、このヘッダファイルを利用するのですが、ふだんの運用中にはあまり使うことはないでしょう。

pgsql/dataディレクトリの直下には、PostgreSQLの動作を調節するための設定ファイルやデータベース間で共有

されるシステムテーブルなどが含まれています。そして、さらに下のpgsql/data/baseサブディレクトリには、それぞれのデータベースがディレクトリ1つに対応づけられて保管されます。ですから、たとえば“ascii5”という名称のデータベースを作成したとすると、このデータベースはpgsql/data/base/ascii5というディレクトリに格納されることになります。

最後にpgsql/manディレクトリですが、ここにはUNIX系OSのmanコマンドで閲覧することができるオンラインマニュアルが格納されており、各種コマンド群のほかにPostgreSQLがサポートしているSQL命令のマニュアルも含まれています。ほかにもpgsql/docというディレクトリがある場合がありますが、こちらにはHTML形式などのマニュアルが格納されます。

template1データベース

ところで、PostgreSQLをインストールして初期設定を終えた直後には、自動的に“template1”というデータベースが作成されています。このデータベースはいったい何のためにあるのでしょうか？

このデータベースは“template～”の名称のとおり、新規に作成するデータベースのテンプレートとして利用されるのです。たとえば、“template1”データベース内にユーザー定義関数を登録しておくと、そのあとに新規作成したデータベースには、初めからこのユーザー定義関数が登録されているようになります。しかしこのことは、“template1”データベースに何らかの損傷を与えてしまうと、そのあとに作成したすべてのデータベースにも損傷を与えてしまうことを意味します。そのため、“template1”データベースへ接続している最中は、慎重に操作を行わなければなりません。

“template1”データベースには、ほかにもデータベースを特定せずにPostgreSQLサーバに接続したい場合に「暗黙の接続先」として利用することができます。具体的には、データベースの一覧を取得したい場合などがこれに当たります。データベースの一覧を取得する場合、PostgreSQLではシステムテーブルにアクセスしなければなりません。しかし、このシステムテーブルにアクセスするためにはいずれかのデータベースに接続する必要があります。このような場合に“template1”データベースを「暗黙の接続先」として接続するのです。

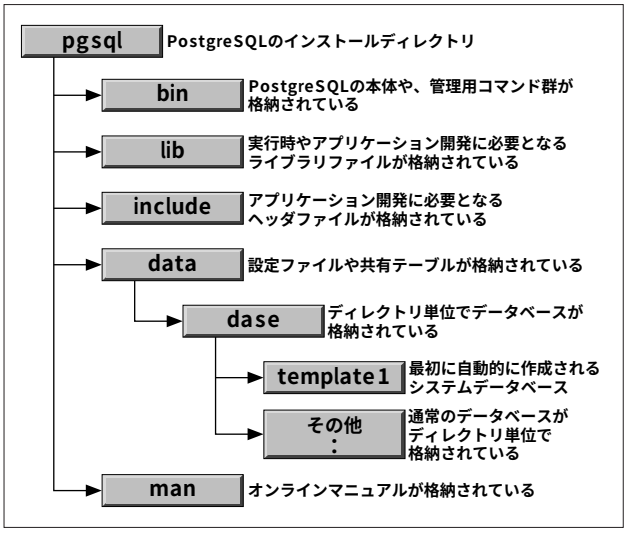


図1　PostgreSQLのディレクトリ構成

postmaster

PostgreSQLサーバを起動するには、マルチユーザーバックエンドであるpostmasterを実行します。この際、postmasterにはいくつかのオプションが指定できます。利用頻度の高いオプションを**表1**にまとめておきます。

単にPostgreSQLを起動するのなら、次のようにします。

```
$ postmaster -S
```

また、ネットワーク経由で利用する場合には、“-i”オプションを付加しなければなりません。

```
$ postmaster -i -S
```

デバッグ用のメッセージを取得したい場合には“-d”オプションを付加し、同時に“-S”オプションはつけないようにします。

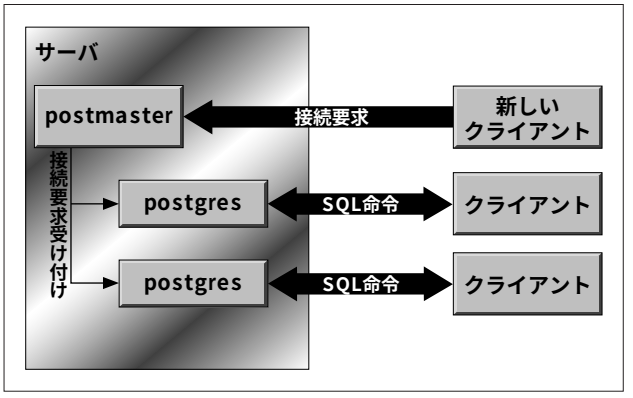


図2　PostgreSQLのプロセス構成

```
$ postmaster -i -d > postgresql.log &
```

PostgreSQLを終了させるにはpostmasterを終了させればよいのですが、これにはUNIXのkillコマンドを利用します。たとえば、postmasterがプロセス番号100番で動作していたとすると、次のコマンドでPostgreSQLを終了させることができます。

```
$ kill 100
```

このとき、postmasterの終了時に実行中だったトランザクションは、結果的にロールバックします。クライアント側では通信が切断されたというエラーになりますので、クライアントのアプリケーションが適切なエラー処理さえ行っていれば、PostgreSQLが突然終了しても重大な問題になることはありません。

psqlコマンド

対話型SQLインタプリタであるpsqlコマンドは、もっとも基本的なPostgreSQL用のクライアントアプリケーションです。このpsqlコマンドは、システムの管理者や開発者にとってはもっとも使用頻度の高いツールでもあるので、まずpsqlコマンドの簡単な使い方などを説明しておきましょう。

psqlコマンドは次のようにして起動します。

```
$ psql [<オプション>] [<データベース名>]
```

<データベース名>には接続するデータベースを指定します。データベースを指定しなかった場合には、現在のユーザー名と同じ名称のデータベースに接続します。もちろん、接続先のデータベースが存在しなかった場合には、psqlコマンドはエラーメッセージを出力して終了します。

たとえば、“ascii5”という名称のデータベースに接続するには、以下のようにします。

```
$ psql ascii5
```

psqlコマンドには多くのオプションがありますが、そのうちの主なものを**表2**に挙げます。

これを見ていただければわかるように、psqlコマンドはシェルスクリプトなどでの利便性も考慮されており、なかなか便利に利用できます。この**表2**に挙げたほかにも、SQL文の実行結果をHTMLのテーブル形式で出力するオプションなどもあります。psqlコマンドで“ascii5”データベースに接続した様子を**画面1**に示します。

“Welcome～”から始まる9行はpsqlコマンドのオー

オプション名	説明
-D <データディレクトリ>	データディレクトリ（pgsql/data）をフルパスで指定する。デフォルトのデータディレクトリは、PGDATA環境変数の内容
-N <同時接続ユーザー数>	PostgreSQLバージョン6.5以降で利用できるオプションで、同時に接続を受け付けるユーザー数（クライアント数）を指定する。デフォルトのユーザー数は32。ユーザー数が多いと必要とするリソース（同時オープン可能ファイル数、メモリ、共有メモリなど）も多くなる。共有メモリが足りないなど理由によるリソース不足でpostmasterの起動に失敗したと思われる場合には、OSの設定などを変更してリソースを増やすか、このオプションで同時接続ユーザー数を減らすといった対処をする
-i	ネットワーク経由でPostgreSQLを利用することを許可する。具体的には、INETドメインからの接続を受け付けるようになる。このオプションを指定していない場合は、ホスト名を指定した接続は受け付けない
-p <ポート番号>	ネットワーク経由でPostgreSQLを利用する場合に、クライアントからの接続要求を待ち受けるポート番号を指定する。デフォルトのポート番号はPGPORT環境変数の内容で、通常は5432
-o -F	“-o -F”と2つのオプションを続けて指定することによって、トランザクションのたびに行われるディスクキャッシュの同期を行わないようする。PostgreSQLの動作速度は劇的に向上するが、OSレベルの障害時（OSの停止や停電など）の信頼性は低下する。ちなみに、“-o”は、シングルユーザーバックエンドであるpostgresプログラムに対するオプションを指定するオプションで、“-F”がディスクキャッシュの同期を抑制するpostgresプログラムのオプション
-S	サイレントモードを指定する。つまり、postmasterは一切のメッセージを表示しなくなる。また、同時にこのオプションは、postmasterの実行を自動的にバックグラウンドに移行させる。たとえば、シェルのコマンドラインから“postmaster -S”を実行したのであれば、すぐに次のプロンプトが返ってくることになる
-d [<デバッグレベル>]	デバッグメッセージを表示する。<デバッグレベル>は省略可能で、指定する場合には1以上の整数が指定でき、数字が大きいほど、多くのデバッグメッセージが表示されるようになる。なお、-Sオプションといっしょに指定するとせっかくのデバッグメッセージが表示されなくなってしまうので、-dオプションを指定する場合には、-Sオプションは指定しないようにする。ただし、-Sオプションを指定しないとpostmasterは自動的にはバックグラウンドに移行しないので、必要に応じてシェルの機能などで代用することが必要になる

表1　postmasterの主なオプション

```
% psql ascii5
Welcome to the POSTGRESQL interactive sql monitor:
  Please read the file COPYRIGHT for copyright terms of POSTGRESQL
[PostgreSQL 6.5.2 on i586-pc-linux-gnu, compiled by gcc 2.7.2.3]

   type \? for help on slash commands
   type \q to quit
   type \g or terminate with semicolon to execute query
 You are currently connected to the database: ascii5

ascii5=>
```

画面1　psqlコマンドの起動直後

プニングメッセージで、最後の“ascii5=>”がpsqlコマンドのプロンプトです。プロンプトの左部は現在のデータベース名を表し、右側の“=>”は現在の入力状態を表します。入力状態には次の4つがあります。

=>	通常の状態
->	SQL文の入力中
'>	文字列の入力中
">	識別子の入力中

デフォルトの動作では、psqlコマンドのSQL文は“;”（セミコロン）を入力するまで実行はされません。SQL文の途中で“;”（セミコロン）を入力せずに改行した場合、次の行のプロンプトは“->”に変わり、SQL文はまだ続いているとみなされます。また“'”（シングルクォーテーション）で始まる文字列リテラルの入力中に改行した場合は、次の行のプロンプトは“'>”になり、改行も含めて文字列リテラルの入力中とみなされます。“">”も同様です。

“=>”と“->”のプロンプトの状態のときには、この直後にメタコマンドが入力できます。メタコマンドとは、SQL文のようなデータベースにアクセスするコマンドとは違い、psql自身の動作を指示するコマンドです。メタコマンドの名称はすべて“\”（円記号）（英語環境では“＼”（バックスラッシュ））から始まります。**表3**に主なメタコマンドを示します。

ユーザーの管理

では、PostgreSQLの基本的な部分を理解してもらったところで、そろそろ具体的な運用／管理の話に移りましょう。まずは、PostgreSQLのユーザーの管理方法について説明します。

PostgreSQLでは、接続認証やアクセス権限をデータベースの利用者である「ユーザー」という単位で管理しています。このユーザーはPostgreSQL内で独自に管理しており、OSが管理している「ユーザー」とはまったくの別物です。ですから、あるユーザーがOSに登録されていたとしても、そのユーザーがPostgreSQLのデータベースにアクセスできるとは限りません。逆に、OSには登録されていないユーザーをPostgreSQLに登録することも可能です。

さらにPostgreSQLでは、UNIX系のOSと同様に「スーパーユーザー」という概念があります。PostgreSQLのスーパーユーザーはすべてのデータベースに対してあらゆる操作が許されています。ですから、すべてのデータベースのバックアップなどはPostgreSQLのスーパーユーザーの権限で行います。

なお、ユーザーの管理が行えるのはPostgreSQLのスー

オプション名	説明
-h <ホスト名>	psqlコマンドは、デフォルトでは自分のホスト上のPostgreSQLサーバに接続するが、ほかのサーバに接続したい場合にはこのオプションを指定する
-p <ポート番号>	ポート番号を指定する。デフォルトのポート番号はPGPORT環境変数の内容で、通常は5432
-u	ユーザー名とパスワードによる接続認証を行う必要がある場合には、このオプションを指定する。このオプションを指定してpsqlコマンドを起動すると、psqlコマンドは起動直後にユーザ名とパスワードの入力を要求してくる
-l	データベースの一覧を表示して終了する。“-c”や“-f”オプションは無視される
-c <SQL文>	このオプションを利用すれば、SQL文をコマンドラインから指定できる。指定したSQL文を実行し終わるとpsqlコマンドは終了する
-f <ファイル名>	SQL文をファイルから読み込み、実行する。ファイルをすべて実行し終わるとpsqlコマンドは終了する
-o <出力先ファイル名>	psqlコマンドの出力結果をファイルに保存する。デフォルトの動作ではpsqlコマンドのオープニングメッセージやSQL文そのものも保存されるが、“-q”オプションによって抑制できる
-q	このオプションを指定すると、psqlコマンドのオープニングメッセージやSQL文そのものなど、SQL文の実行結果以外の一切メッセージを出力しない。“-t”オプションとともにシェルスクリプトなどで利用する際に指定する
-t	このオプションを指定すると、SQL文の実行結果にカラム名を含めない

表2　psqlコマンドの主なオプション

パーユーザーだけです。一般ユーザーでは自分のパスワードの変更さえも許されていませんので注意が必要です。

ユーザーの管理を行う方法には、コマンドを利用する方法とSQL文を利用する方法の2つがあります。ここではシンプルでわかりやすいSQL文を利用する方法だけを紹介しますが、コマンドを利用する方法については、次のコマンドに関するマニュアルなどを参照してください。

```
createuser
destroyuser
pg_passwd
```

■ユーザーの一覧

PostgreSQLではユーザーの情報を“pg_shadow”というシステムテーブルに保管しています。このテーブルには各ユーザーのパスワードなども格納されるため、PostgreSQLのスーパーユーザーだけがアクセスすることができます。その代わり、パスワードカラムが隠された“pg_user”というシステムビューが用意されていますので、こちらを利用すれば、一般ユーザーでもある程度の情報にはアクセスが可能です。

たとえば、ユーザーの一覧を取得するには次のSQL文を実行します。

```
SELECT * FROM pg_user;
```

このSQL文を実行した例が**画面2**です。主なカラムの意味を**表4**に示します。

```
ascii5=> SELECT * FROM pg_user;
usename |usesysid|usecreatedb|usetrace|usesuper|usecatupd|passwd  |valuntil
--------+--------+-----------+--------+--------+---------+--------+---------------------------
kataoka |     994|t          |t       |t       |t        |********|
postgres|     993|t          |t       |t       |t        |********|Sat Jan 31 15:00:00 2037 JST
nobody  |     995|f          |t       |f       |t        |********|
(3 rows)
```

画面2　ユーザーの一覧表示

コマンド名	説明
\connect <データベース名> [<ユーザ名>]	psqlコマンドを終了することなく、データベースやユーザーを変更して再接続する。ユーザーだけを指定したい場合は、データベース名として“-”（ハイフン）を指定する。“\c”は、このコマンドの省略形
\l	データベースの一覧を表示する
\dt	テーブルとビューの一覧を表示する
\di	インデックスの一覧を表示する
\ds	シーケンスの一覧を表示する
\dS	システムテーブルの一覧を表示する
\d [<オブジェクト名>]	テーブル、ビュー、インデックス、シーケンスを一覧表示する。<オブジェクト名>を指定した場合には、一覧ではなくそのオブジェクトについてのみ、詳細（カラム名など）を表示する
\z	テーブルやビュー、シーケンスのアクセス権を一覧表示する
\h [<SQL文>]	利用可能なSQL文の一覧を表示する。<SQL文>を指定すると、そのSQL文についての詳細ヘルプを表示する
\r	現在入力中のSQL文をキャンセルする
\w <ファイル名>	SQL文を実行した直後であれば、そのSQL文をファイルに保存する
\i <ファイル名>	<ファイル名>で指定したファイルからSQL文やメタコマンドを読み込み、実行する
\o <ファイル名>	SQL文の実行結果を<ファイル名>で指定したファイルに出力します。<ファイル名>の代わりに“|<コマンド>”とすれば、実行結果をコマンドに渡す
\copy <テーブル名> from または to <ファイル名>	データをコピーする。“from”を指定した場合はファイルからテーブルへ、“to”を指定した場合はテーブルからファイルへコピーする。ファイル上でのフォーマットはレコード区切りが「改行」、カラム区切りが「タブ」、データとしての改行文字やタブ文字はそれぞれ“\N”と“\T”になる
\q	psqlコマンドを終了する
\?	メタコマンドの一覧を表示する

表3　psqlコマンドの主なメタコマンド

■ユーザーの作成

新たにユーザーを作成するには次のSQL文を利用します。

```
CREATE USER <ユーザー名>
[WITH PASSWORD <パスワード>]
[[NO]CREATEDB]
[[NO]CREATEUSER]
[IN GROUP <グループ名> [, ...]]
[VALID UNTIL <有効期限>]
```

“ CREATEDB ”キーワードを指定した場合、このユーザーは新たにデータベースを作成することができるようになります。“ CREATEDB ”キーワードを指定しなかった場合のデフォルトは“ NOCREATEDB ”、つまりデータベースを作成することはできません。“ CREATEUSER ”キーワードを指定した場合には、このユーザーは新たにユーザーを作成することができるようになります。また、同時にPostgreSQLのスーパーユーザーの権限も持つことになります。“ CREATEUSER ”キーワードを指定しなかった場合のデフォルトは“ NOCREATEUSER ”、つまりユーザーを作成することはできませんし、スーパーユーザーの権限もありません。<グループ名>には、このユーザーが所属するグループを指定するのですが、正しく動作しないようです。グループについては、コラム「**PostgreSQLとグループ**」を参照してください。<有効期限>は、このユーザーがPostgreSQLを利用できる期限です。有効期限を過ぎると、このユーザーはPostgreSQLにアクセスすることができなくなります。

カラム名	説明
usename	ユーザー名
usesysid	ユーザーID。PostgreSQLは、このID番号によってユーザーを識別する
usecreatedb	データベースの作成権
usesuper	スーパーユーザーの区別
valuntil	ユーザの有効期限。この期限を過ぎるとデータベースに接続できなくなる。このカラムがヌル値（NULL）の場合は無期限を意味する

表4　pg_userシステムテーブルの主なカラム

Column

PostgreSQLとグループ

PostgreSQLでは、「グループ」という言葉はあまり聞き慣れないと思います。というのも、執筆時点での最新バージョン（ver.6.5.3）までのPostgreSQLにはグループを管理するためのコマンドが用意されていなかったからです。かといって、グループという管理単位がないわけではありません。実は、ずっと以前からグループはあったのですが、利用するにはシステムテーブルを直接変更するなどの手間がかかってしまうため、あまり利用されていないのでしょう。それでも、グループを利用したいという読者のために、グループの使い方を簡単に紹介しましょう。

グループの情報は、“ pg_group ”というシステムテーブルに格納されます。ですから、グループを管理するためには、このシステムテーブルを操作してやればいいのです。

このシステムテーブルには以下のカラムがあります。

groname … グループ名（英字と数字のみ）
grosysid … グループID（グループ間で重複しない任意の番号）
grolist … グループに所属するユーザーIDのリスト

たとえば、新たに“ testgroup ”というグループを作成し、“ postgres ”と“ kataoka ”というユーザーを所属させるとします。まず、“ postgres ”と“ kataoka ”のユーザーIDを調べます。

```
ascii5=> SELECT usename, usesysid FROM pg_user
ascii5-> WHERE usename IN ('postgres', 'kataoka');

usename |usesysid
--------+--------
kataoka |     994
postgres|     993
(2 rows)
```

“ testgroup ”のグループIDを“ 1 ”と決めて、グループを作成します。

```
ascii5=> INSERT INTO pg_group
ascii5-> VALUES ('testgroup', 1, '{993,994}');
INSERT 194497 1
```

以上で“ testgroup ”の作成が完了しました。これでGRANT文を利用して“ testgroup ”にアクセス権を与えることも可能です。グループの内容を変更するには今挿入したレコードを新しい内容で更新すればよく、グループの削除もレコードを削除するだけと、やることはとても単純です。

このように、グループの管理は思ったよりも簡単にできるわけですが、やはりCREATE GROUPなどのSQL文で管理できるようになることを期待したいですね。

■ユーザーの属性変更

ユーザーのパスワードや権限などを変更するには、次のSQL文を利用します。

```
ALTER USER <ユーザー名>
[WITH PASSWORD <パスワード>]
[[NO]CREATEDB]
[[NO]CREATEUSER]
[IN GROUP <グループ名> [, ...]]
[VALID UNTIL <有効期限>]
```

＜ユーザー名＞には既存のユーザーを指定しなければなりません。なお、ここで指定しなかった属性は一切変更されませんので、変更したい属性についてのみ指定します。

■ユーザーの削除

ユーザーを削除するには、次のSQL文を利用します。

```
DROP USER <ユーザー名>
```

データベースの管理

PostgreSQLでは、データベースの作成権を持っているユーザーだけがデータベースを作成することができ、そのデータベースの管理者となります。

データベースの管理を行う方法も、ユーザーの管理方法と同様にコマンドを利用する方法とSQL文を利用する方法の2つがあります。ここでもSQL文を利用する方法のみを紹介しますが、コマンドを利用する方法については次のコマンドに関するマニュアルなどを参照してください。

```
createdb
destroydb
psql -l
```

```
ascii5=> SELECT * FROM pg_database;
datname    |datdba|encoding|datpath
-----------+------+--------+----------------
template1  |   993|       1|template1
ascii5     |   994|       1|ascii5
(2 rows)
```

画面3　データベースの一覧表示

■データベースの一覧

データベースの情報は“pg_database”というシステムテーブルに格納されています。psqlコマンドには、データベースの一覧を表示する“\dl”というメタコマンドがありますが、システムテーブルを読み出すことによってもデータベースの一覧が得られます。**画面3**に、システムテーブルを利用してデータベースの一覧を取得した例を、主なカラムの意味を**表5**に示します。

■データベースの作成

新たにデータベースを作成するには次のSQL文を利用します。

```
CREATE DATABASE <データベース名>
[WITH LOCATION = '<配置>']
[ENCODING = '<エンコーディング名>']
```

＜配置＞にはデータベースを作成する場所が指定できます（詳しくはあとで説明します）。また、PostgreSQLがマルチバイト対応版の場合にはデータベース単位で言語の指定が可能です。具体的には＜エンコーディング名＞にデータベースで扱う文字列のエンコーディングを指定します。たとえば、日本語を扱うデータベースの場合には“EUC_JP”を指定します。なお、デフォルトのエンコーディングはPostgreSQLのインストール、あるいは初期設定時に指定したものです。

■データベースの削除

データベースを削除するには次のSQL文を利用します。

```
DROP DATABASE <データベース名>
```

■任意のディレクトリにデータベースを作成する

PostgreSQLでは、標準で**図1**のディレクトリ構成の

カラム名	説明
datname	データベース名
datdba	管理者のユーザーID。PostgreSQLは、このID番号でユーザーを識別する
encoding	データベースで扱う文字データのエンコーディングID。エンコーディングIDの意味は、**表6**を参照のこと

表5　pg_databaseシステムテーブルの主なカラム

エンコーディング名	ID	説明
SQL_ASCII	0	ASCII文字
EUC_JP	1	日本語EUC

表6　PostgreSQLがサポートする主なエンコーディングの一覧

pgsql/data/baseにデータベースを作成します。しかし、ディスクの負荷分散を行いたい場合やディスクの空き領域が足りない場合など、ほかのディレクトリにデータベースを作成したい場合もあるでしょう。そのような場合のために、PostgreSQLでは任意のディレクトリに追加のデータベース格納領域を作成することが可能です。ここではその方法を紹介します。

まず、新しいデータベース格納領域の場所を決め、ディレクトリを作成します。ここでは仮に/pgdata1とします。そして、そのディレクトリのオーナーおよびグループをPostgreSQLのユーザーとグループ（通常はいずれも“ postgres ”）に変更します（**画面4**）。なお、ディレクトリの作成とオーナー、グループの変更は権限のあるユーザーで行い、それ以降の作業はPostgreSQLのユーザーで行います。

次に、データベース格納領域を初期化します。これには“ initlocation ”コマンドを利用します（**画面5**）。

PostgreSQLから新しいデータベース格納領域を利用するには、そのディレクトリを環境変数に登録したのち、PostgreSQLサーバを再起動しなければなりません。この環境変数が常に有効となるように、.bash_profileなどのシェルの設定ファイルに環境変数の登録処理を追加しておきます。ここでは例として“ PGDATA1 ”という環境変数に登録することにします（**画面6**）。

そしてPostgreSQLを再起動します。どのような方法で再起動しても構いませんが、今登録した環境変数が有効になるように注意してください。**画面7**の例では手作業でPostgreSQLを再起動していますが、もしもマシンの起動時に自動的にPostgreSQLが起動されるようになっているのなら、マシンごと再起動してしまってもかまいません。

以上で、新しいデータベース格納領域の準備は整いました。例として、今作成したデータベース格納領域内に“ ascii5_test ”というデータベースを作成してみます（**画面8**）。

この**画面8**のSQL文で、“ WITH LOCATION ”に対して先ほどの環境変数名である“ PGDATA1 ”を指定している点が重要です。ここに直接“ /pgdata1 ”などと具体的なディレクトリを指定してもエラーになってしまいます。つまり、“ WITH LOCATION ”にはデータベース格納領域のディレクトリを保持する「環境変数名」を指定しなければならないのです。しかし、これはある意味で便利な仕様といえるでしょう。というのも、ディスク構成の変更などで将来データベース格納領域が変わってしまったとしても、PostgreSQLの方では環境変数の内容を変更するだけですむからです。

なお、1つのデータベース格納領域には複数のデータベースを格納することができます。

●

今回は、管理／運用に関する話の中でも、基本的な部分の紹介だけで終わってしまいましたが、定期メンテナンス、バックアップなど、ネタはまだまだあります。そこで次回は、より実践的な管理／運用法を紹介したいと思います。

```
% su
Password:
# mkdir /pgdata1
# chown postgres:postgres /pgdata1
# exit
%
```

画面4　データベース格納領域の作成

```
% initlocation /pgdata1
We are initializing the database area with username postgres (uid=993).
This user will own all the files and must also own the server process.

Creating Postgres database system directory /pgdata1/base
```

画面5　データベース格納領域の初期化

```
% vi .bash_profile
        :
PGDATA1=/pgdata1
export PGDATA1
```

画面6　環境変数の登録

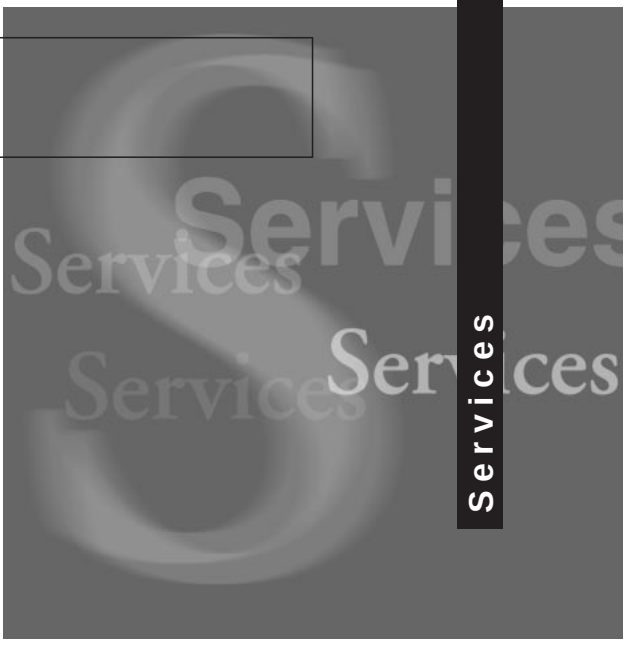
```
% kill <PostgreSQLのプロセスID> ←PostgreSQLの終了
% . .bash_profile ←新しい環境変数を有効にする
% /usr/local/pgsql/bin/postmaster -S -i ←PostgreSQLの
起動（オプションは例）
```

画面7　PostgreSQLの再起動

```
% psql ascii5
        :
ascii5=> create database ascii5_test
ascii5-> WITH LOCATION = 'PGDATA1';
CREATEDB
```

画面8　データベースの作成（格納領域を指定）

Sambaと闘う（最終回）



やり残した作業はいろいろ多いのだが、今回はその中から、企業などで使うことが多いと思われる、「NTサーバによる認証」を紹介しよう。

共有モード

文：梅原 系
Text : Kei Umehara

今回のトピックは、「Sambaのユーザー認証をどのように行うか」だ。これについては、連載の2回目（1999年12月号）でも簡単に解説した。その時は、Sambaサーバ＝認証サーバという構成についてのみ説明したが、ここではそのほかの方法も含めて紹介することにしよう。

Sambaの共有モード

Sambaの共有モードには、4つ選択肢があり、設定ファイル（smb.conf）の書き方次第で変更できる。

これらの共有モードは、大きくユーザーごとにアクセス制御を行うものと、リソースごとにアクセス制御を行うものの2種類に分けられる。

ユーザーごとにアクセス制御を行う方法では、ユーザーアカウントとパスワードの組み合わせが認証サーバによって認証されることで、リソースへのアクセスが許可されることになる。ただし基本的には、Sambaサーバマシン自体のユーザー管理とは別管理であることには注意されたい。

たとえば、NTサーバを認証サーバとして使う方法では、Sambaマシン、つまりUNIXマシンではなく、ユーザーがログオンする時に使用するNTサーバに認証を一元化できるので、管理を大幅に単純化できるというメリットがある。

逆に、ユーザー認証によるアクセス制御ではなく、公開リソース単位でのパスワードによるアクセス制御（各リソースに設定したパスワードさえ知っていれば、誰でもアクセスできる）という方法も採れる。Windows 9xをサーバとして使う場合と同じだといえば、わかりやすいだろう。NTドメインなどを構築し、ユーザーベースのアクセス管理をしている環境では不要な機能だが、古いOSを使い続けている場合には、便利なことがあるかもしれない。

securityパラメータ

Sambaの共有モードの設定には、smb.confの[global]セクションにある、securityパラメータを使う。指定できるのは、以下の4つのいずれかだ。

■ share

リソース単位でのパスワードによるアクセス制御。NTなどのクライアントからアクセスする場合は、一応ログオン処理が行われるが、パスワードなどを指定する必要はない。リソースに対してアクセスする場合に、改めてパスワード（ユーザーパスワードではなく、リソースに設定されたパスワード）が求められる。

一見単純そうに見えるshareモードだが、その実態はかなり複雑だ。というのも、ログオン時にユーザー名が渡されるとは限らないため（ログオン処理はダミーなので、クライアントOS次第ではこの処理をパスすることも可能）、

Sambaがどのユーザー名（とユーザー権限）を使ってアクセスするか、その決定方法が非常に複雑になっているのだ。詳細の解説は本連載の範囲を超えるので、Sambaの解説書などを参照してほしい。

筆者としては、shareモードは過去のシステムとの互換性を取るためのものであり、新規システムで使うものではないと考えている。

■user

現在のSamba（2.0.x）のデフォルト共有モード。以降で説明するserverモードとdomainモードも、このuserモードのバリエーションのひとつと考えることができる。

userモードでは、最初に必ずログオン処理が行われる。平文パスワードと暗号化パスワードのどちらを使うことも可能だ。ここで正しいユーザー名とパスワードが指定されない限り、Sambaはクライアントへのサービスを拒否する。リソースの一覧さえ送られないので、きちんとした設定が不可欠になる。ワークグループレベルのLANでは、このuserモードが一番現実的な選択肢だろう。

平文パスワードと暗号化パスワードについても、1999年12月号で解説しているで参照してほしい。キーになるのは[global]セクションのencrypted passwordsパラメータなので、ドキュメントや解説書では、これをキーワードに調べてみてほしい。

■server

serverモードの挙動は、基本的にはuserモードと同じだ（クライアントからは全く同じに見える）。違うのは、クライアントが接続したSambaサーバではなく、他のサーバによって認証される、ということだ。

どのサーバで認証を行うかは、password serverパラメータで指定する。基本的にはNTサーバを指定するが、仕様上は任意のSMBサーバを指定できることになっているので、複数のSambaサーバを利用している場合などには、smbpasswdファイルの集中管理に利用できるかもしれない（筆者は試していないので、各自で確かめていただきたい）。複数のサーバを指定することもできるので、PDCとBDCの両方を指定しておき、いずれかのNTサーバがダウンした場合のトラブルを減らすことも可能だろう。

画面1　Active Directoryのインストール中のダイアログボックス
現在のSambaはNT 4.0互換なので、Active Directoryのセットアップもそれに準じたものとする。

■domain

domainモードの仕様は、serverモードとほとんど同じである。違うのは、serverモードではクライアントの接続中は認証サーバへの接続も継続するのに対して、domainモードでは認証処理が終わり次第、Sambaサーバと認証サーバ間の接続が切断されるということだ。基本的に、NTサーバ（ドメインコントローラ）を認証サーバに利用する場合は、このdomainモードを使用する。

注意が1つ。現在のSambaには、日本語を含む2バイト文字がユーザー名に含まれているとdomainモードで正しく動作しないというバグがある（とマニュアルに明記されている）。つまり日本語のユーザー名を使っているドメインでは、この機能は使えないということだ。

shareモードと他の3つのモードの間には、認証の単位（リソース単位かユーザー単位か）という違いがある。それに対し、user／server／domainの3モードは基本的に同じもので、単に認証をどこで行うか、どのように認証サーバと通信するか、という違いしかないことに注意しておこう。特にserverモードとdomainモードの違いは微妙で、基本的には同じものと考えることもできる。まったく異なるレベルの設定を、1つのパラメータで切り替えるようになっているので、混乱させられないよう、注意深くマニュアルを読んでほしい。

Windows 2000 Serverを使ったユーザーの認証

前回までは、Sambaサーバが認証サーバを兼ねていたので、userモードを指定していた。今回はNTサーバでの認証ということで、domainモードを使用してみる。これだけではあまりに面白みがないので、人身御供を兼ねて、本誌が店頭に並ぶ頃には発売される、Windows 2000 Serverを認証サーバに使ってみることにしよう。テストに使ったのは、Windows 2000 Server RTM版、Sambaは

これまでと同じく英語版の2.0.6だ。

Windows 2000 Server側のセットアップ

Windows 2000 Serverのインストールで注意することは、システムのインストール後、Active Directoryもインストールしなくてはならないということだ。Active Directoryをインストールすることで、Windows 2000 Serverはドメインコントローラとして機能するようになる。

■NT 4.0互換モードの選択

ポイントは、Active Directoryのインストール中、「アクセス許可」というタイトルのダイアログボックスで「Windows 2000以前のサーバーと互換性があるアクセス許可」を選択することだ（**画面1**）。これを忘れると、Windows 2000 ServerはWindows 2000クライアントからしかアクセスできなくなるので注意したい。Sambaも、いずれはWindows 2000方式のアクセスに対応するだろうが、当面はこの作業を忘れないことだ。

そのほか、DNSの設定などについても、既存のネットワーク環境と整合性が取れるように注意しながら設定を行う。Active DirectoryではDNSが必須になるが、Windows 2000 ServerのDNSサーバを使うのか、Linux（UNIX）側のDNSサーバを使うのか、きちんと検討しなくてはならない。Active Directoryのトラブルを避けるためにはWindows 2000のDNSサーバを使うのが無難だが、Dynamic Updateと呼ばれる機能を備えた最新のBIND（LinuxなどのDNSサーバ）なら、これを使うことも可能である。

Active Directoryのインストール後はシステムの再起動が必要だ。

■Sambaサーバの登録

Active Directoryのセットアップが終わったら、【管理ツール】メニューにある【Active Directoryユーザーとコンピュータ】ツールを使って、Sambaサーバをドメインに登録する。左側のツリーから【Computers】コンテナ（フォルダのようなものだ）を右クリックして【新規作成】—【コンピュータ】を選ぶ。**画面2**のようなダイアログボックスが表示されたら、Sambaサーバのマシン名を入力し、下のほうにあるチェックボックス、【Windows 2000以前のコンピュータに、このアカウントの使用を許可】をチェックする。

以上の2つの作業で、Windows 2000 Server側の作業は終わりだ。続いて、Samba側の設定作業に取りかかろう。

Sambaサーバ側のセットアップ

Sambaサーバ側での作業は、例によってSambaを停止した状態で行うのが原則だ。作業内容をよく確認して、スマートに作業を進めてほしい。

■smb.confの書き換え

最初の作業は、smb.confを書き換えてNT（Windows 2000）を認証サーバとして使うようにすることだ。具体的には、以下の2つの[global]パラメータを追加（もしくは変更）する。

```
security=domain
password server=認証サーバのリスト
workgroup=ドメイン名
```

security=domainについては、先に説明したとおりだ。もう1つのpassword server=パラメータでは、認証サーバ（NTのPDCやBDC）のNetBIOS名を指定する。複数の認証サーバを指定する場合には、カンマで区切る。workgroup=パラメータについては、説明の必要もないだろう。

security=domainモードでは、暗号化パスワードの使用が前提になっている。必要に応じてencrypted passwords=yesも指定しよう。

ちなみに、筆者の実験環境でのsmb.confは、次のようになった。

画面2　Sambaサーバの登録
現在のSambaはNT 4.0互換なので、コンピュータの登録もそれに準じたものとする。

```
[global]
        workgroup = MYDOMAIN
        server string = Samba Test Server--domain mode
        security = DOMAIN
        encrypt passwords = Yes
        password server = SV10
        username map = /etc/smbusers
        log file = /var/log/samba/log.%m
        max log size = 50
        coding system = euc
        client code page = 932
```

■lmhostsの更新

smb.confでは、認証サーバをNetBIOS名で指定する。これをIPアドレスに変換する（名前解決）ために使われるのが、/etc/lmhostsだ。ここには認証サーバを登録しておこう。書式についてはmanを参照してほしい。

Red Hat Linux 6.0で試したところ、/etc/hostsファイルに認証サーバが登録されていれば、lmhostsには書かなくても、問題なく名前解決できるようだ。

■ドメインコントローラの登録

smb.confの書き換えが終わったら、次にsmbpasswdコマンドを使って、ドメインコントローラへのマシン登録を完了させる。先ほどActive Directoryのツールで Sambaサーバを登録したのだが、その時点ではSambaサーバのGIDなどが決まっていないため、それをSambaサーバ側から登録するという作業が必要になるのだ。

Samba関連コマンドに共通することだが、smbpasswdもなかなか多機能なコマンドで、妙なところで使うことになる。Sambaサーバをドメインに登録するための書式は、次の通りだ。

```
smbpasswd -j ドメイン名 -r ドメインコントローラ名
```

ドメイン名、ドメインコントローラ名には、それぞれ適切な名前を指定すること。登録処理がうまくいけば、次のメッセージが表示されるはずだ。

```
smbpasswd: Joined domain ドメイン名.
```

もう1つ確認してほしいのが、/etcディレクトリ（ディストリビューションによって異なることもあるが、原則としてsmb.confと同じディレクトリ）に“ドメイン名.Sambaサーバ名.mac”という名前のファイルが作成されているかだ。このファイルは、ドメインコントローラにアクセスする際に必要となる、マシンIDなどが登録されている重要ファイルだ。書き換えたりすると、すぐにSambaが使えなくなるので扱いには十分注意してほしい。万一削除してしまった場合には、ドメインコントローラ側で一度Sambaサーバをドメインから削除し、関連する作業を最初からやり直す必要がある。

テスト

基本的には、以上の作業でドメインコントローラによるユーザー認証が行えるようになる。Sambaを起動し、クライアントからアクセスできるかテストしよう。テストに当たっては、以下のような点に注意してほしい。

・設定がうまくできておらず、ドメインコントローラの認証に失敗した場合、Sambaはsmbpasswdを使った認証を行う。そのため正しく設定できているかを確認するには、smbpasswdファイルを空にしておく必要がある。

・security=domainモードでは、認証はドメインコントローラで行われるが、ファイルへのアクセスはUNIX側のユーザーアカウントが使われる。つまり、いずれにしてもUNIX側へのユーザー登録は必要ということだ。

・クライアントマシンがドメインに登録されていても、ユーザーがログオン時にドメインにログオンしていない（たとえば、ローカルマシンにログオンしているなど）と、認証に失敗してSambaサーバをアクセスできない。

・トラブルシューティングには、[global]セクションのdebug level=パラメータを使うといい。引数には数字を指定する。数字が大きいほど、詳細なメッセージがログに記録される。通常のテストでは2～3、トラブルシューティングでは9程度を指定して何が起こっているかを確かめよう。デバッグメッセージは、Sambaのログファイル（/var/log/sambaなどに格納）に出力される。

ただしデバッグメッセージを出力し続けると、あっという間にログファイルが散らかってしまう。動作を確認した後は、デフォルトに戻すのを忘れないようにしたい。

結果的に、（筆者の環境では）Windows 2000を認証サーバに使う構成は、なんのトラブルもなく動作した。

ただし、Windows 2000に対応したのはSamba 2.0.6からで、それ以前のリリースではいろいろとトラブルが起こるという話も伝わってきている。旧リリースを使っているユーザーの場合は、ドメインコントローラもNT 4.0を使ったほうが無難だろう。

after case：日本語の扱い

この連載で、日本語の扱いについて解説したことがあるが、これについて読者からの質問をいただいた。他の読者でも困っている方がいそうな事例なので、簡単にフォローしておきたい。

■ext2ファイルシステム以外へのアクセス

「（Linuxではなく）FreeBSDでSambaを使っているが、VFAT上の日本語ディレクトリが、Sambaクライアントからアクセスできない」というのが、読者からの質問であった。実は、同様の症状はLinuxでも起こることがある。

Linuxのファイルシステムはモジュール構造になっており、プログラムさえ書けば、さまざまなファイルシステムをアクセスできるようになっている。Linuxで標準的に使われているのはext2と呼ばれるLinux独自のファイルシステム（基本的には、UNIXの一般的なファイルシステムをちょっとカスタマイズしたもの）だが、DOSのFATやVFAT、CD-ROM、NTFSなどといったさまざまなファイルシステムをLinuxからアクセスできるのは、世界中のLinuxプログラマーが、これらのファイルシステム用のモジュールを開発してくれたおかげなのだ。

ところで、日本語版のWindows 9xでは、（FATや）VFATに日本語のファイル名やディレクトリ名を使った場合、ショートファイル名（8＋3文字）にはシフトJIS、ロングファイル名のエントリにはUNICODEでそれらを格納する。それに対してLinuxなどのUNIX系OSでは、デフォルトの漢字コードにはEUCを使うことが多く、UNICODEまでサポートしているものは少ない。

このため、普通にVFATなどの日本語名のファイル／ディレクトリをアクセスしようとすると、コードの混乱から対象ファイルがアクセスできない、ということになるのだ。特にASCIIコードとUNICODE間の変換は単純な機械的作業（0x00の追加削除程度）で行えるが、日本語となるときちんとしたテーブルが必要となる。

つまり、UNIX側がサポートするVFATファイルシステムドライバの多くは、ASCIIコードの範囲の文字列はなんとか扱えるようになっているものの、日本語ファイル名はまったくだめという現象が生じてしまうのだ。

LinuxのFAT、VFATのコードでも、ASCIIとUNICODE間の変換を行っている。しかし、シフトJISのコード範囲は変換できるようにはなっておらず、すべて“？（クエスチョンマーク）”として置換するようになっている。そのため、日本語名の付いたファイルでも参照することは可能だ。しかし、新たに作成するのは現実的ではないのである。

ちなみに、Linuxネイティブのext2ファイルシステムでは、Sambaの設定にもあるように、シフトJISからEUCに変換して処理することで日本語ファイル名を問題なく扱うことができるようになっている。これはファイルシステムで日本語の文字コードを処理するわけではなく、単にリードバイトなどが不正に処理されないようになっているだけだ。8ビットスルーという用語でよく表現されるので、聞いたことがあると思う。

■Samba日本語版 2.0.5aJP2

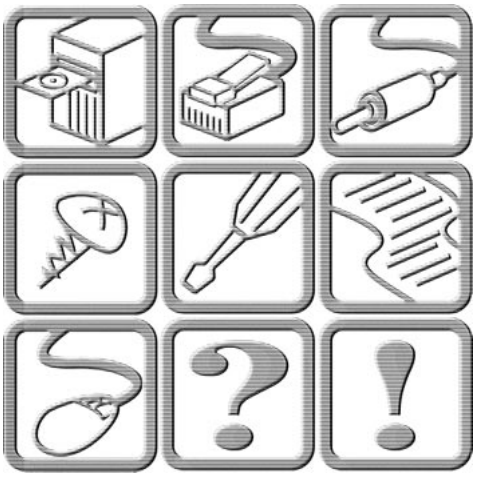
Sambaの日本語サポートについてもう1つニュースを紹介しておこう。以前にも本稿で紹介したことのある日本語版Sambaだが、先日2.0.5aJP2がリリースされた（http://www.samba.gr.jp/）。新リリースの目玉は、共有名やコンピュータ名、コメントなどでの日本語サポートだ。もちろんSWATでもこの機能に対応しており、簡単に日本語共有名が使えるようになっている。

UNIX畑の管理者だと、共有名などに日本語を使うのをためらうことも多いだろうが、企業など、一般的なWindowsユーザーにとって、これらの日本語サポートはありがたい。お試しあれ。

終わりに

基本の基本から始めたこの連載だが、今回を区切りとして、ひとまず終了ということになった。Windows側で公開しているリソースをLinux側から使う機能など、紹介しきれなかったSambaの機能もいろいろあるのだが、基本的な使いこなしについての最低限の紹介はできたと思う。短い間ではあったが、お付き合いいただいた読者の皆様に感謝したい。

Try & Try [trái] [trái]



TVチューナー付きキャプチャカード

文:政久忠由
Text : Tadayoshi Masahisa

前回の補足

まずは前回の補足から。前回、システムのお引越しについて紹介したわけだが、あとで少し端折りすぎかなと思う部分があったので補足しておきたい。それはファイルをコピーする部分だ。パーミッション（アクセスコントロール）やシンボル、ハードリンクを含めコピーする方法は紹介したとおりなのだが、マウントポイントとして存在するディレクトリの扱いについては、ほとんど触れていない。少し考えれば、たぶん分かるはずと思って省略したんだけど、/procと/mntの扱いに困る可能性、というか不毛にハードディスク容量を消費してしまっている場合があるので、フォローしておこう。

まず/procだが、これはprocファイルシステムと呼ばれるシステムのステータスをファイルシステムとしてマッピングするためのマウントポイントである。その内容は、カーネルが生成するので実際のファイルシステム上には、/procという空のディレクトリがあればよいのだ。つまり、このディレクトリに関しては、コピーする必要はなく、新しくルートファイルシステムにするディレクトリで“mkdir proc”として新規にディレクトリを作成しておくだけで構わないのだ。

また/mntに関しては、ファイルコピー時の臨時のマウントポイントとして利用していたので、これも再帰的にコピーしてしまうというわけにはいかない。一般に/mntには、cdromやfloppyのマウントポイントがあるだけなので、これまた既存のパーミッションを確認しつつ新規に作成してもいいし、臨時のマウントポイントを/1とかに変更することで対処してもいい。

/procにしても/mntにしても間違って不必要なファイルまでコピーしてしまったとしても、無駄にファイルシステムを消費するだけで、破壊的な問題となるわけではない。マウントポイントは必ずしも空のディレクトリである必要はないので、気づかないで運用してしまっているかもしれない。そこで確認のために、各マウントポイントを一度アンマウントして間違ってファイルがコピーされていないか、確認してみるとよいだろう。

人間万事塞翁が馬

あと今回の作業には、カーネル再構築の必要性があるかもしれないので、これに関連した注意点をひとつ紹介しておこう（まぁ、実際に僕が陥ったミスなんだけど…）。

よほどのことがない限り、カーネルの再構築自体はつつがなく終わると思う。で、モジュールのインストール、そしてカーネルのインストールを行うだろう。しかしながら、ふとしたことからコンフィギュレーションを変えてみようと思うことがある。僕の場合、今回のbttv関連のドライバを有効にするあたり、よせばいいのに、ネットワークオプ

ションのパケットソケットとUNIXドメインソケットの2つのオブジェクトをモジュールとしてコンパイルするようにしてしまった。しかしこの時点では、この些細なことが数分後の危機的状況を生もうとは夢にも思わなかったのである。

カーネルに変更を加え、システムをリブートしてみると、カーネルの初期化後、initがキックされてから、modprobeのエラーが表示され、初期化スクリプトがまったく処理されない状況に陥ってしまった。突然の不測の事態にパニック寸前だったが、落ち着いて記憶をロールバックし、問題の原因が何であるかを考えてみた。ところが、システムを起動する前には、カーネルの再構築だけではなく、modprobeユーティリティ群、binutils、そしてコンパイラと、さまざまなコンポーネントも新しいものにリコンパイルしたことに気づいた。そのため、問題の原因が何であるか即座に特定できない。

まずは表示されたエラーメッセージ（modprobeをキルしたといった内容）から、modprobeプログラム自体を疑ったのだが、メッセージをよく見ると、カーネルからのモジュールの要求に対して、modprobeは8回程度、タイムアウトを繰り返しながら試行しているだけであった。このことは、しばらくしてから表示されたmodules.depが見つからないというエラーメッセージで確認した（そういえば、カーネルのバージョンも上げたんだっけ）。従来の僕の環境では、最低限必要なオブジェクトはカーネルに含めるようにしていて、各モジュールの依存関係を記したmodules.depの生成は、初期化スクリプト任せにしてしまっていた。一種の慣れである。その昔は、再起動する前にdepmod -a ＜カーネルバージョン＞としてキチンと自ら生成していたのに…、でも今悔やんでもしかたない。

状況を総合的に検討してみると、どうやらネットワークのソケットドライバがこの時点で要求されているようだ。上記のソケットドライバは、ネットワークデバイスインターフェイスの開始まで必要ないと思っていたが、共有ライブラリルーチンのレベルで参照する部分があるのかもしれないし、僕の環境のようにネットワークカードのドライバをカーネルに含めていることが関係しているのかもしれない。ともかく、modules.depが生成されていれば、カーネルがkmodルーチンで適切にモジュールをロードしてくれそうな感じだ。

しかしinitで処理が行われ始めた直後に止まってしまっているので、modules.depを生成するにも対話可能なシェルがない。また今回は不幸にも複数のカーネルを起動できるようにLILOを設定しておらず、緊急用のフロッピーもなかったりする。結局のところ既存のカーネルしか選択肢がないのである。

ダメもとでランレベル1で起動してみた。しかし、予想通り何も改善されない。今の状況は、ランレベル以前のシステムブート時用のスクリプト/etc/rc.d/rc.sysinit処理の最初の段階で止まってしまっている。実はrc.sysinitでは、中盤でdepmodを実行するようになっていて、これさえ実行されていれば今のような事態にはなっていなかったのである。

つまりここでは、初期化スクリプトをスキップして、対話可能なシェルを起動する必要があるのだ。そこで、カーネルが最初にキックして起動するプログラム（デフォルトは/sbin/init）をカーネルパラメータでもって別のプログラムに変更することにした。これを変更するには、カーネルパラメータのinit=を使用する。

```
boot: Linux init=/bin/sh
```

上記のようにLILOのブートメニューの段階で指定すれば、ラベル（ここではLinux）で指定したカーネルが最初に起動するプロセスを/bin/shにすることができる。

これにより、ようやくカーネルと対話できる環境に持ち込むことができた。ただし、この場合でもlsなどのコマンドを実行するとmodprobeのエラーメッセージが表示されてしまう。でも試行のタイムアウトにより、対話は続行可能だ。

この段階でシステム復旧のための方策は、2つ考えられる。modules.depの生成とlilo.confを編集して以前の稼動実績のあるカーネルを起動できるようにすることだ。前者は、確実性は低いが簡単、後者は、確実だが面倒である。ここではlsコマンドの実行時にkmodが機能してしまったことを考慮し、よりステップの少ないmodules.depの生成を試みることにした。

```
/sbin/depmod -a
```

超簡単と思いきや、書き込みエラー。あっそうそう、起動時、ルートファイルシステムは、リードオンリーでマウントされているんだった。ルートファイルシステム（/lib/modulesディレクトリがあるファイルシステム、普通は別にしないと思うけど）を書き込み可能な状態に移行しないと変更が加えられない。これを行うために、次のコ

マンドを実行する。

```
mount -n -o remount,rw /
```

-nオプションは、/etc/mtabを変更しないためのもので、ここでは、余計な処理を発生させないために指定している。続く-oオプションでは、remount、rwという引数を指定している。見てのとおり、リード、ライト属性で、マウントし直せという指定である。

これで書き込み可能になったので、あとは先ほどのdepmodコマンドを実行するだけだ。また、当然lilo.confを修正する場合でも、マウントし直す必要がある。modules.depが作成されたのを確認したら、早速リブートといきたいところだが、このまま終了してしまうとファイルシステムのダーティフラグが立ったままで、再起動時にチェックディスク（fsck）が実行されてしまう。まぁ、そうしたい場合はそれでもよいが、時間の無駄なので、次のコマンドを実行してフラグを下げておこう（リードオンリー属性でマウントし直せばよい）。

```
mount -n -o remount,ro /
```

再起動後システムは正常に機能するようになった。それにしても、慣れや、ちょっとしたことが大きなトラブルとなることを再確認することになってしまったわけだが、「禍を転じて福となす」のことわざのしかり、トラブルは有益な経験でもありえることも認識する今日この頃である。謝謝！？

またもや、お下がりデバイス、確保

前回のハードディスクに引き続き、Linuxマシンにお下がりデバイスがやってきた。今回は、TVチューナー付きキャプチャカードだ。

僕のデスクトップ環境であるWindows NT上でテレビを、そしてあわよくばビデオデッキの代わりとして使おうと考え、以前購入してそれなりに利用していたんだけど、環境をWindows 2000にしたことや、それに伴うBIOSのアップデート、そしてその他もろもろのハードウェア設定の変更により、そのキャプチャカードにBIOS（PCIバス）の初期化の段階でハードウェアリソースが割り振られなくなってしまったのである。

もともとNT上では、オーバーレイ表示のご機嫌が悪く（できなかった）、Windows 2000でも不明デバイスとしてしか認識されないマイナーなカードで、デバイスドライバとしてどれを使ってよいのか分からない状態だった。でも一応、以前のNT4用のデバイスドライバで動作はするし、少し前のWindows 2000 DDKのサンプルコードとして付属していたBt848チップ（ビデオキャプチャのためのチップとして有名なやつ。ちなみに僕のカードはBt878搭載、基本機能レベルで互換性がある）用のデバイスドライバソースもコンパイルして利用できていた。しかし、本当の意味でのTV、ビデオの代替としては、クオリティ面で納得できるものではなかったので、TVチューナー付きキャプチャカードの当初のもくろみは空中分解してしまい、単なるBGM（BGVか？）ツールとなってしまっていた。

このような経緯もあって、カードにハードウェアリソースが割り振られなくなったとき、特別な思いはなかった。原因はなんとなく分かっていたのだが、ハードウェア的にぶっ壊れた可能性もないわけではないので、デスクトップ環境用のマシンからは退かせ、テスト用マシンに取り付けてみることにした。そう、LinuxマシンにTVチューナー付きキャプチャカードがやってきたのである。

案の定、Linuxマシンでは、TVチューナー付きキャプチャカードは、機嫌よくハードウェアリソースを割り振られていた。どうやらハードウェアレベルでの不具合はなさそうだ。

LinuxマシンとTVチューナー付きキャプチャカード

Linuxでのキャプチャカードのサポートは、結構昔から行われていて、カーネルバージョン2.0.xでは配布カーネル自体にコードは含まれていないものの利用できたし、2.2.xではカーネルに含まれる形で配布されている。Linuxでは、動画のサポートとして、Video for Linuxとしてインターフェイス（API）が規定され、現在、Video for Linux2（http://millennium.diads.com/bdirks/v4l2.htm）の開発も進められている。ちなみに、ここでは僕の使用しているカーネル2.3.39（開発版カーネル）をベースに話を進めることにする。

まずは、僕の使用しているTVチューナー付きキャプチャカードが何者で、現状Linuxでサポートされているものかどうかをチェックする必要がある。とりあえずは、カードを肉眼で見て、チップやチューナーのメーカー、そして型番を控えておく。次にPCIカードであれば、システムがPCI BIOSからディテクトした情報も見ておく。PCIカー

ドの情報は、/proc/bus/pci/devices（/proc/pciで対応文字列に変換した形で見ることもできる）を直接見てもよいが、普通の人は、この数字の羅列を見ただけでは、何のことか分からないと思うので、lspciコマンドを使用する。lspciコマンドは、PCIユーティリティのひとつで/sbin/に格納されていると思うが、もしもない場合は、http://atrey.karlin.mff.cuni.cz/~mj/pciutils.htmlから取得してほしい。通常、新しいバージョンでは、ベンダーやデバイスのIDのデータベースが拡張されている可能性があるので、チェックしてみるとよいだろう。

```
$ cat /proc/bus/pci/devices
0040    109e036e        b       e2009008
0041    109e0878        b       e200a008

$ cat /proc/pci
  Bus  0, device   8, function  0:
 Multimedia video controller: Brooktree Corporation
Bt878 (rev 2).
 IRQ 11.
 Master Capable.  Latency=32. Min Gnt=16.Max Lat=40.

$ /sbin/lspci -v
00:08.0 Multimedia video controller: Brooktree
Corporation Bt878 (rev 02)
 Subsystem: Unknown device 109e:036e
 Flags: bus master, medium devsel, latency 32,IRQ 11
 Memory at e2009000 (32-bit, prefetchable) [size=4K]
```

これらの情報から、僕のTVチューナー付きキャプチャカードは、Brooktree社のBt878チップをベースにしたものであることがわかる。ちなみに、上記にdevice 109e:036eといった表示があるのがわかると思うが、これの109eがBrooktree社、036e、0878がBt878チップを表している。カードによっては、さらにサブベンダーIDなどがあり、上記のようなSubsystem: Unknownではなく、ちゃんとそのベンダー名などがより細かくわかるようになっている。これを踏まえて、/proc/bus/pci/devicesの数字の羅列を見てみるとキチンと内容のあるものであることがわかるだろう。

しかしながら、このPCI BIOSからの情報だけでは、チューナーの詳細までは知ることができない。これは目で確認したPhilips製を信じることにしよう。

必要なデバイスドライバをコンパイル

先ほどの調査でビデオコントローラチップがBt878であることがわかったので、早速、必要なデバイスドライバを選択し、コンパイルすることにしよう。しかし、まずは情報収集ということで/usr/src/linux/Documentation/video4linux/bttv/のドキュメントに目を通しておこう。

ここで必要となるビデオコントローラ用のデバイスドライバは、カーネルのコンフィギュレーションメニューのCharacter devicesの項目に含まれるVideo For Linuxにある。ここでは、次の2つをモジュールとしてコンパイルするように選択した。ビデオアダプタの選択肢にBt878はないが、基本的にBt878はBt848のアッパーコンパチなので問題ない（以前各チップのスペシフィケーションか何かで違いを調べてみたのだが、忘れてしまった）。

```
<M> Video For Linux
<M> BT848 Video For Linux
```

早速モジュールをコンパイル、インストールして使ってみよう。そして、モジュールのインストールが完了したら、次にdepmod -aとしてモジュールの依存関係ファイルを再生成しておこう。と、いきたいところだが、実は、Bt848 Video For Linuxのモジュールbttv.oはi2cなるものに依存しているようだ。I2C（I2Oじゃない）ってなんぞやって感じなんだけど、これは、Philips社が開発した低速バスプロトコルだ。本来、低速なバス接続のデバイスと通信するために使用されるもので、PCI接続のデバイスに必要なの？と疑問に感じるが、デバイスドライバの依存関係で要求しているのだからしようがない。と思っていたら、実はI2Cは、各デバイスカード上に搭載されているEEPROM、まぁそこに記載されている情報とかファームウェアを読み取るのに結構利用されているようだ。このビデオキャプチャカードの場合もそのために使用しているし、Xサーバのドライバも利用している（これからしていくといったほうがいいのかな）。

I2Cは、Character devicesの項目に含まれるI2C supportで選択する。後ほどの実際に試した結果からするとI2C supportとI2C bit-banging interfacesを選択しておけばよいようだが、不安であれば全部をモジュールとして選択しておくとよいだろう。

```
<M> I2C support
<M> I2C bit-banging interfaces
< >   Philips style parallel port adapter
< >   ELV adapter
< >   Velleman K9000 adapter
<M> I2C PCF 8584 interfaces
< >   Elektor ISA card
<M> I2C device interface
```

カーネルコンフィグレーションで設定するのは、以上の2個所だ。選択が終わったら、次のようにコンパイルしてモジュールを導入しよう。

```
# make modules modules_install
```

次にdepmod -aと行いたいところだが、/usr/src/linux/Documentation/video4linux/bttv/のドキュメントのとおり、/devにデバイスノードを作成し、/etc/modules.confを設定しておく必要がある。

新しいデバイスノードは、ドキュメントディレクトリにあるシェルスクリプトMAKEDEVを実行するだけだ。chmodで実行属性を付けるか、/bin/bash MAKEDEVとしてこのスクリプトを実行すると、/dev/video0などが作成されるようになっている。

/etc/modules.confの設定は、とりあえずドキュメントディレクトリにあるModules.confを/etc/modules.confに追加リダイレクトして編集することにしよう。

```
# cat Modules.conf >> /etc/modules.conf
```

各オプションについては、Insmod-optionsとCARDLISTを参考に設定する。ちなみに僕の設定は次のようなものだ。

```
# i2c
alias char-major-89     i2c-dev
#options i2c-core       i2c_debug=1
options i2c-algo-bit    bit_test=1

# bttv
alias char-major-81     videodev
alias char-major-81-0   bttv
options bttv            card=0
options tuner           type=2 debug=1
```

とりあえず、ポイントとしては、僕のカードはCARDLISTファイルにもないので、bttvのオプションでcard=0としてカードタイプを自動的に判定させようとしていることと、tunerのオプションで2のPhilips NTSCタイプを指定していることぐらいだ。

/etc/modules.confの修正が終わったら、depmod -aでモジュールの依存状況ファイルを生成して、いよいよデバイスドライバをロードしてみよう。モジュールの依存状況を見ればわかるが、bttvをロードすればその他のモジュールは自動的に組み込まれるはずである。

```
# modprobe -k bttv
# lsmod
Module       Size Used by
tuner        3048 1 (autoclean)
bttv        45292 0 (autoclean) (unused)
i2c-algo-bit 7592 1 (autoclean) [bttv]
i2c-core    12444 0 (autoclean) [tuner bttv i2c-
algo-bit]
videodev     3008 2 (autoclean) [bttv]
```

正常に読み込まれているようだ。次にdmesgでドライバが出力したメッセージを見てみよう。

メッセージを見れば大体何が行われているかわかると思う。Bt878搭載カードということで、ニアリーなカードHauppauge newに見立てられ、サウンドチップのチェックが行われている。でもどれも当てはまらないようだ。チューナーは、ちゃんとPhilips NTSCとして登録されている。あとでわかることなんだけど、実は僕のこの環境ではサウンドが出力されていない。チューナーカードからのオーディオアウトをサウンドカードのラインインにケーブルで接続するタイプで、音はチューナー部で適当にデコードするのだろう思っていたのに、どうしたものだろう？（というわけで、今回、音は棚上げしちゃいます）

アプリケーションをゲット

Linuxカーネルに含まれているのはデバイスドライバまでで、実際に表示するアプリケーションは別途入手する必要がある。今回はこの手のアプリケーションとして最も有名なxawtvを使用してみようと思う。以下にその入手先と

bttvデバイスドライバのホームページのアドレスを紹介しておく。

http://www.in-berlin.de/User/kraxel/xawtv.html
http://www.metzlerbros.de/bttv.html

xawtv

xawtvは、X上でTVを表示するためのアプリケーションで、現在バージョン3.07がリリースされている。またこのホームページには、bttvドライバの0.7.15（Linuxカーネル2.3.39には0.7.13が付属）も登録されていたので同時に入手して利用することにした。それぞれ適当なディレクトリに展開してmakeする。makeが正常に終了したら、あとはスーパーユーザーとなり、make installを実行するだけだ（特に調整するパラメータもないので、makeの説明は省略する）。bttvドライバは、最初からスーパーユーザーとなり、makeを実行する。ドライバのインストールは同様にmake installを実行する。

```
$ ./configure ; make
```

xawtvには、xawtv、xawtv-remote、fbtv、streamer、v4l-confなどのアプリケーションが含まれている。xawtvは主となるTV画面を表示するためのXアプリケーションで、チャンネルの切り替えから、イメージのフレームキャプチャ、動画として保存するなどの操作が行えるようになっている。xawtv-remoteはそのxawtvをコマンドラインで操作するためのツールだ。またfbtvは、フレームバッファを利用したTV画面を表示するためのアプリケーション、streamerはコマンドラインでキャプチャ操作を行うためのツール、v4l-confはVideo for Linuxインターフェイスレベルのサイズ、色数を設定するためのツールである。

xawtvを実行する前に、少しだけデフォルト値を設定しておこう。基本的には日本用のチューナーの設定である。これらの値は~/.xawtvに記述する。

```
[global]
freqtab = japan-bcast
[defaults]
norm = NTSC-JP
source = Television
```

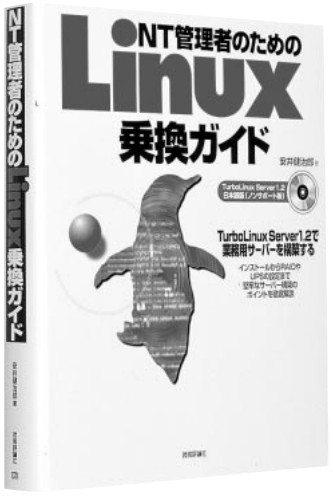
では、xawtvを実行してみよう。操作自体難しいものではないので、特に説明の必要はないと思うが、一応、画面を右クリックするか、キーボードからOを入力するとオプションウィンドウが現れるので、ここで設定の調整を行おう。またチャンネルエディタで必要な局を登録しておくと便利だ。登録したチャンネルは、チャンネルウィンドウが表示され選択できるようにもなる。

まずは、最初のテストなのでキャプチャカードを取り付けたLinuxマシンでXサーバを動かし、そこでxawtvを実行してみたわけだが、これが本当の目的ではない。僕にとってはデスクトップとして使用しているマシンに表示させないと意味がないのだ。フレームバッファを利用したfbtvに興味を示していない理由はこのためだったりする。もし、Linuxマシンをデスクトップとして使用しているのであれば、カーネルでフレームバッファモジュールを有効にして使用してみると、Xとは比べ物にならない軽さを感じられるかもしれない。またXでも、xawtvは、開発中のXFree86 3.9.x（正式リリース版の3.3.[56]あたりもサポートしてたかも？）などがサポートするDGAを利用できるようになっているようなので、試してみるといいだろう（あっ、僕のXFree86 3.9.x環境でのマウス、挙動が変だったんだけど、gpmのバージョンを上げて、インテリマウスなんだけど単にps/2マウスとしたら快調に動くようになりやがった。そんなことだったのね）。

で、肝心の僕のデスクトップ環境へのxawtvの表示だが、表示自体は問題なく行えている。しかし、サウンドはどうしたら出るのという問題もさることながら、僕の利用形態に関する根本的な問題が露呈してしまった。ネットワークの転送速度である。Xサーバの描画性能も多少影響しているものの、単にXの表示を他のマシンのディスプレイに表示するという計画は、動画に関しては無謀であった。落ち着いて考えれば、通常動画のネット配信は帯域が問題になるから、いかに圧縮して転送するかがキーポイントとされているのに、何の圧縮もせずXプロトコルでXサーバにイメージを転送しているだけなものだから、帯域をめいっぱい（10BASEの環境で、1Mバイト/秒以上のトラフィック）使っても、384×288で秒、2コマも描画できていないありさまだ。でも、192×144ではそれなりに耐えられるレベルなので、しばらくは重宝できそうだ（トラフィックは1Mバイト/秒以上のままだけど）。

というわけで、課題はサウンド出力、そしてキャプチャ、変換ツールを駆使した現実的なタイマーセットもできるビデオレコーダーとしての用途に加えて、ネットワークへの動画のストリーム配信へと広がるのであった。

Books



NT管理者のためのLinux乗換ガイド

安井健治郎 著
技術評論社
B5変形判／432ページ／CD-ROM付き　　本体価格 3200円

サーバ用途でLinuxを使う場合、Windows NTなどと違ってGUIではなく、コンソールからコマンドを入力し、エディタで設定ファイルを修正することが中心となるので、NT管理者はとまどうことだろう。また、NTとLinuxではサービスのプログラム名が違うので、セットアップするために何から手を着けてよいのかわからないかもしれない。本書は、NT Serverの知識がある人向けに、Linuxサーバを構築するためのノウハウが書かれている。

DNSやDHCP、メール、Web、FTPといった基本的なネットワークの設定はもちろんだが、Sambaを使ったWindowsマシンとのファイル共有や、Netatalk、ネットニュースサーバ、FAXサーバ、IRC（チャット）サーバなどの設定方法も解説されている。また、UPSやRAIDの使い方、バックアップの方法といったものも含めて、サーバを管理するために必要な事柄をほとんど網羅しているので、Linuxサーバ管理の基礎知識を得ることができる。

Linuxコマンドリファレンス

スタークラスター 著
ナツメ社
B6変形判／440ページ　　本体価格 2300円

最近のLinuxディストリビューションでは、GNOMEやKDEといったデスクトップ環境が整ってきて、GUIからマウスの操作でいろいろなことができるようになってきた。しかし、シェルやコマンドを使わなければいけない場面もまだまだある。Linuxを運用していくには、コマンドを自由に使いこなすことが必須なのである。

そんな時に、本書のような各種コマンドをアルファベット順に解説したリファレンスがあると便利だろう。オンラインマニュアルではわかりにくい細かなオプションまで詳細に解説しており、すべてのコマンドに具体的な使用例が掲載されている。

コマンドリファレンスだけではなく、bashやtcshといったLinuxで標準的に使われているシェルの基本操作や、viやEmacs（Mule）といったテキストエディタの操作方法も書かれているので、初心者にも実践的な入門書として役立つだろう。

できるLinux サーバー活用編

辻 秀典、渡辺高志、アクロバイト＆インプレス書籍編集部 著
インプレス
B5変形判／256ページ　　本体価格1800円

Linuxでは、メールサーバやWebサーバ、Sambaなどを利用して、高機能なサーバを構築できるが、インストールしたあとの設定は結構むずかしい。

本書では、セキュリティを強化するために、通信を暗号化するsshや、ワンタイムパスワードの導入を勧めている。そして、ファイルサーバ（Samba）をWindows NTと共存させる方法や、Sendmailの設定では、大量の同報メールを効率よく配信するためにsmtpfeedを使う方法、fmlを使ったメーリングリストサーバの導入も記述されている。Webサーバ（Apache）では、アクセス制限やパスワード認証、SSLを利用した暗号化Webサイトの設定といった高度な利用方法が解説されている。

Linuxサーバをイントラネットやインターネット向けに本格的に運用するためには、カスタマイズが必要である。本書はそのための助けとなるであろう。

Apacheハンドブック第2版

Ben Laurie、Peter Laurie 共著 田辺茂也 監訳 大川久人、三代川 信義 訳
オライリー・ジャパン
B5変形判／444ページ
本体価格 3800円

入門Perl/Tk

Nancy Walsh 著 石曽根 信、西中芳幸 訳
オライリー・ジャパン
B5変形判／440ページ
本体価格 3800円

Caldera OPEN LINUX STARTER KIT

安達昭人 著
秀和システム
B5変形判／256ページ／CD-ROM付き
本体価格 2800円

QtではじめるXプログラミング

杉田研治 著
技術評論社
B5変形判／382ページ／CD-ROM付き
本体価格 2980円

Linux徹底構築ガイド システムチューンアップ編

西尾和彦 著
テクノプレス
B5変形判／456ページ
本体価格 4200円

オープンソース・ワールド

川崎和哉 著
翔泳社
A5並製／400ページ
本体価格 1900円

最近売れてるLinux書籍は？

新年を迎え2000年になりました。今回は本の紹介ではなく、コンピュータ業界にとって、2000年がどのような年になっていくのか、現時点までの動向から占ってみたいと思います。

昨年末ごろから、Linux関連の書籍の売れ行きはすっかり沈静化してしまった感があります。特に、いわゆるインストール本はほとんど動きが止まってしまいました。ただし、このことイコールLinux人気にかげり、とは思っていません。おそらく、本誌のような専門雑誌が何冊も創刊されたことで、この手の役割が雑誌に移ってきたということなのでしょう。メディアの特性を考えると、今年もこの流れはより強まっていきそうです。

ただ少し気になるのは、Perlなどの一部のテーマを除いて、オライリーの書籍に今ひとつ元気がないことです。いや、元気がないというよりも、Linux人気に比例したような活気が感じられないのです。ひと昔前なら、オライリーのシリーズ書籍は、UNIXを志す人にはバイブルといえる必読書でした。以前とは、ユーザーの気質や利用形態が変わってきているのかもしれませんが、Linuxがブームに終わらず、定着したものになっていけるのか不安なところです。

海外では、XMLやJavaが新たに盛り上がり始めているようで、売上ランキングの上位に顔を見せ始めています。具体的にどのような形で利用されているのかはわからないのですが、今年は日本にもその流れが来そうな気配があります。ただLinuxは、XMLもJavaも他のプラットフォームに比べて乗り遅れ気味と聞いています。少し心配なところです。

さて、ようやくWindows 2000もリリースされます。これが今年最大のトピックになるでしょう。Linuxユーザーも、少なからず気になるハズです。書店としても盛り上がりを期待したいのはヤマヤマですが、Windows 95のころのような盛り上がりはないのでは、というのが正直な気持ちです。しばらくは、「模様ながめ」といった感じでしょうか？

Linux関連であまり明るい予想ができず恐縮です。まあ、予想はあくまでも予想ですから…。ただし、好材料もあります。それは、年末からファイヤウォール関連の書籍の売れ行きが好調だということです。これは、インターネットサーバとしてLinuxが本格的に利用され始めているということの証ではないでしょうか？　（書泉ブックドーム 古田島）

読者の声

俺にもいわせろ！

もうそこまで春がやってきているとはいえ、寒い日が続きますね。こんな時期はスキー、スノボに温泉が最高です。……はぁ、時間さえあればなぁ。

気を取り直して今月もまいりましょう。

2月号の特集へのお便り

今回の「フリーソフト大全」はとても役に立ちました。私はDebian（potato）を使っているのですが、ソフトの数がものすごく多くて、しかもソフトの説明が英語なので名前と中身が一致しないことがよくあったのですが、今回の記事を参考にして便利なものをインストールしてみようと思いました。

（大阪府　文野 城さん）

定番のものから知る人ぞ知るというものまで、Linuxで動作するフリーソフトは無数にありますね。Linux magazinは、これからもさまざまなソフトウェアをご紹介していきます。

今月の「ハードディスク大増設指南」、大変よくまとまっていて参考になりました。数カ月前、試行錯誤を行いながら15Gバイトハードディスクを増設したときにこの特集があったらだいぶムダな時間を使わずに済んだのに……。

（千葉県　緒方宏昭さん）

「Linux 100の疑問」はとっても参考になりました。

ハードディスクの増設もWindowsでは、ほいほいとできるのですが、Linuxではうまくいかずあきらめていたので、これを読んでもう1回トライしてみようと思ってます。

（長野県　Carlleyさん）

お役に立てて編集部一同喜んでおります。ディスクの増設といえば、Windowsなどにはドライブネームという手強い敵がいますね。このせいで、ディスクを増設しているのか、それともパズルを解いているのかわからなくなってしまった方も多いのではないでしょうか。

次期Vineへの期待高まる

Vine Linux 2.0の登場が待たれるこのごろ、職場の机上のPCにもハードディスクを増設し、区画を空けて待っています。

（山形県　＜直角＞さん）

妥協のない日本語対応で根強いファンの多いVine Linuxですが、現行のVine Linux 1.1は、一昨年リリースされたRed Hat Linux 5.2をベースにしているため、ほかのディストリビューションに比べると、ちょっと古く感じるようになりました。しかし、いよいよ3月にVine Linux 2.0がリリースされます。楽しみですね。

Linux＋Windowsの最強マシンプロジェクト

今度自作のPCを作ろうと思っております。WindowsとLinuxの2つのOSを入れたいと思います。最新のパーツを組み合わせて作りたいのですが、ぜひ特集で2大OSを組み入れたPCのお勧めをやってほしいと思います。制作費別とか、使い方別とか、色々な方面から検討したPCを見てみたいものです。ぜひお願いします。

（山形県　遠藤浩二さん）

面白そうですね。マシンを組むときには、利用目的をはっきりさせないとお金ばかりかかって中途半端なマシンになりやすいようです。え？　それは担当だけ？

待ち人来たらず

この前の記事からずっと気になるKondara MNU/Linux。ぜひ次の付録によろしくお願いします。

（大阪府　宮崎昭憲さん）

Kondara MNU/Linuxを付録CD-ROMに入れてほしい。できれば、Alpha版も入れてほしい。

（群馬県　近藤良夫さん）

このほかにも同様のお便りをいただいております。Kondara MNU/Linuxは編集部でも人気の高いディストリビューション

ですが、発売元のデジタルファクトリジャパンさんの方針で、雑誌付録への収録はできないのが現状です。製品版も6800円と比較的安価ですので、こちらのご購入もご検討ください。

時代はトランスルーセントです。

Linuxは、UNIXのべんきょーとか、Cとかのべんきょーに使っている人です。

マルチブートです。System Commander使ってます。外付けSCSI 4GバイトにLinuxがすみついてます。

Eterm -Oでスケスケです。

（千葉県　山田兼嗣さん）

ちょっと怖い感じもしますが、なんとなくリズミカルなお便りだったので思わず掲載してしまいました。スケスケはいいっすね。

虚飾を廃して質を高めよ

カラーの必要のないページが多い気がする。豪華さより充実を。

（大阪府　小川昌敏さん）

よりよい誌面づくりを目指し、編集部一同さらなる努力を続けてまいります。今後ともよろしくお願いいたします。

LinuxでのY2K問題

Linuxで定番の某ゲームのハイスコアが2000年から記録されなくなりました。意外なところで2000年問題に遭遇しました。

（北海道　泉 裕さん）

ひっそりと潜んでいるのがY2K問題の怖いところです。まだまだ気を緩めるわけにはいきません。よし、チェックのために手元にあるゲームを片っ端から……バキッ。デスク！　なんで殴るんですか？　Y2Kはまだ去っていないん……ドカッ　うぐぐ。

スペック競争の陰で

最近はどうも古い機種への設定方法などが各誌とも見受けられませんが、Linuxのウリはそれではなかったか？ Win 2000に対応できなくなる機種が中古で巷にあふれそうな最近、ぜひとも取り組んでいただきたいですね。

（大阪府　池 教男さん）

来月は古いマシンの再生を特集します。Windows 2000は、高いマシンスペックを要求しますので、池さんのおっしゃるとおり、Linuxユーザーにとってはマシン獲得のチャンス到来かもしれません。

Linuxの使い道

新ディストリビューションにWin 2000。OSの選択が増えて面白い今日ですが、速くなって安定度の増したW2Kで仕事、Win98でゲーム、となるとLinuxではなにをしようか……。

一介の文系サラリーマンでは、やはりLinuxの需要はないのだろうか…？

（大阪府　松本友宏さん）

家庭内LANでサーバにするもよし、プログラミングの勉強に使うもよし、Linuxの使い道はいくらでもあります。Windowsだけでなく、Linuxの世界もお楽しみください。Linux magazineがそのお手伝いをいたします。

最初はそういうものです

Windows雑誌の「読者の声」は“なるほど”と思いながら読むのですが、Linux雑誌の「読者の声」は記事同様難しくてよくわかりません。

「読者の声」くらいはうなずきながら読めるようになりたいものです。

（千葉県　八木橋 邦彦さん）

読者の声も役立つLinux magazineです (^_^)_v

続けて読んでいくうちに、必ず“なるほど”と思えるようになりますよ。それがLinux者への道なのです。

Slackwareは心のふるさと

先日Slackware 7.0を入手しました。業務で使用しているNFSサーバのアップグレードの準備をしています。最近、TurboやRed Hatばかりさわっていたのですが、古風なUNIX派には、Slackwareが何となく落ち着く気がします。もっとSlackwareも広めてください。

（岡山県　田中 多津男さん）

Slackwareはシンプルでいいですね。担当も久しぶりに使ってみたらなんだかほっとしました。

実は、編集部にも熱狂的なSlackwareファンがいます。彼は、自宅でSlackwareサーバを中心に、Macintosh、Linux／Windows混在の“濃い”環境を構築しています。曰く、「ふつーSlackware」だそうです。

今月の困ったおハガキ

やっぱBSDっしょ！　BSD Magazine最高！

（東京都　千葉健一さん）

(^^;;;;;;;　えーと。BSD Magazine編集部に伝えておきます。Linux magazineもよろしくお願いしますね。

では、また来月お会いいたしましょう。おハガキお待ちしております。

付録CD-ROM Disk1に収録した
LASER5 Linux 6.0 Rel.2フリー版のインストール

CD-ROMブートができる場合は、付録CD-ROM Disk1をCD-ROMドライブに入れ、CD-ROMから起動します。CD-ROMブートができない場合は、DOS、Windows環境でブートディスクを作成する必要があります。この場合、フォーマット済みのフロッピーディスクをAドライブにセットし、CD-ROMのimages\makediskディレクトリに移動して、そこにあるバッチファイルでブートディスクを作成します。通常はmkboot.batファイルでブートディスクを作成します。

ネットワーク経由でインストールする場合は、mkbootnet.batでブートディスクを作ります。また、ノートPCでPCカードを利用してインストールする場合には、mkpcmcia.batでブートディスクのほかにPCMCIAサポートディスクを作っておきます。

CD-ROM、またはブートディスクから起動し、“boot:”のプロンプトが表示されたらEnterキーを押します。日本語が正しく表示できない時には、プロンプトに対し“english”と入力し、Enterキーを押すとメッセージが英語になります。

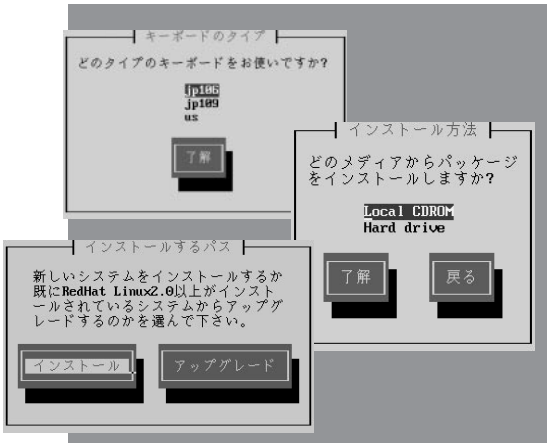
これ以降のインストーラの操作は、[Tab]キーで項目の移動、矢印キーで選択の移動、スペースキーで選択、Enterキーで決定となります。

1 キーボード、インストール方法、インストールパスの設定

最初に、使用するキーボードの種類を選択します。日本語106キーボードでは「jp106」を、Windowsキーのある日本語109キーボードでは「jp109」を選びます。英語キーボードでは「us」を選びます。

「インストール方法」の画面では、利用するメディアを選びます。ここでは「Local CDROM」を選択します。

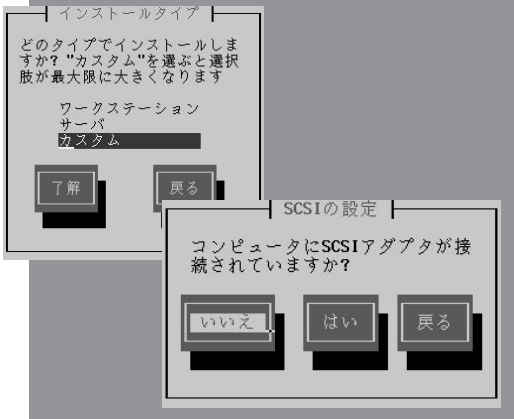
「インストールするパス」では、新規インストールをするか、アップグレードするかを選びます。Red Hat Linux 2.0以降がインストールされている場合は、「アップグレード」を選ぶことができます。ここでは「インストール」を選択することにします。



2 インストールタイプ、SCSIの設定

どのようなアプリケーションパッケージをインストールするか選択します。「ワークステーション」と「サーバ」では、規定のパッケージがインストールされます。「ワークステーション」を選ぶと既存のLinuxパーティションが、「サーバ」を選ぶとハードディスクの全領域が初期化され、パーティション設定からインストールまでが自動的に行われます。「サーバ」を選ぶと、Windowsなど、ほかのOSの領域もすべて消えてしまいますので注意してください。「カスタム」ではこれらの設定を手動で行います。ここではカスタムを選択したものとして説明を続けます。

「SCSIの設定」画面では、SCSIインターフェイスを接続している場合は「はい」を選びます。なければ「いいえ」を選択します。「はい」を選ぶとSCSIインターフェイスが自動検出されます。



3 ディスクのセットアップ（カスタムのみ）

ここでは、ディスクのパーティションとマウントポイントを設定します。Disk Druidとfdiskのどちらかのツールを利用できます。ここでは、Disk Druidを使って説明します。

パーティションは、最低限“/（ルート）”と“スワップ”の2つが必要です。8.4Gバイト以上のハードディスクを使う場合は、最初にカーネルの置かれる“/boot”を10～20Mバイト作っておきます。

パーティションを追加するには、「追加」ボタンを押して、マウントポイント、サイズなどを設定します。スワップ以外は、タイプを「Linux native」にします。スワップパーティションを追加するときは、タイプで「Linux swap」を選びます。サイズはメインメモリと同じ程度を目安として32～128Mバイトに設定します。

必要なパーティションを作成したら、「了解」を選択します。

4

スワップスペースを有効にする、フォーマットするパーティションの選択

「スワップスペースを有効にする」画面では、スワップとして使用するパーティションを選択します。“＊”がついているのが利用されるパーティションです。「フォーマット中に不良ブロックをチェックします」に“＊”をつけると、そのパーティションのディスクメディアのチェックをします。

「フォーマットするパーティションの選択」画面では、フォーマットするパーティションを選びます。ここでも、利用するパーティションに“＊”をつけます。

5

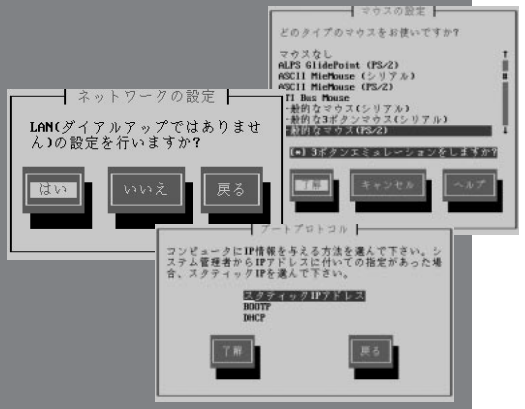
インストールするコンポーネント（カスタムのみ）、インストールログ

インストールするアプリケーションパッケージを選びます。スペースキーでインストールしたいパッケージに“＊”をつけます。「個々のパッケージを選択する」をチェックすると、パッケージの詳細な選択ができます。

このあと「解決されていない依存関係」の画面になることがありますが、そのまま「了解」を選べば、自動的に必要なパッケージがインストールされます。

「インストールログ」の画面では「了解」を押します。インストールの過程は/tmp/install.logにログとして保存されます。

「インストール状況」では、選択したソフトウェアをハードディスクにコピーします。進捗状況がグラフ表示されますので、終わるまで待ちましょう。



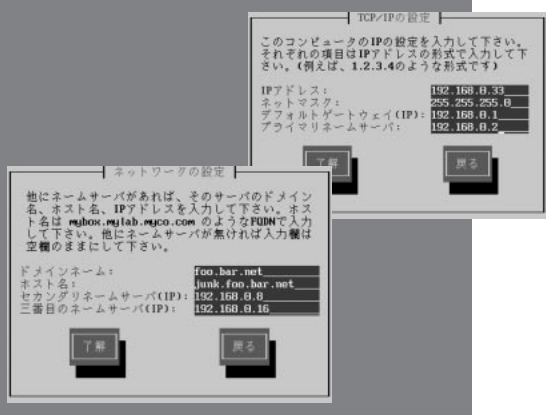
6

マウスの設定、ネットワークの設定、ブートプロトコル

マウスの種類を設定します。2ボタンマウスを利用する場合は、「3ボタンエミュレーションをしますか？」にチェックを入れます。これをチェックすると、3ボタンマウスの中ボタンを、左右ボタン同時押しで代替します。

「ネットワークの設定」画面では、LANの設定を行うかどうかを選択します。ここでは「はい」を選択することにします。「はい」を選択すると、ネットワークカードの検出を行います。自動検出できない場合は手動で選択します。

「ブートプロトコル」画面では、IPアドレスをどのように設定するかを選択します。「スタティックIPアドレス」では、手動でIPアドレスなどを入力します。「BOOTP」、「DHCP」では、それぞれの該当するサーバから必要な情報を取得します。ここでは、「スタティックIPアドレス」を選択して進めることにします。



7

TCP/IPの設定、ネットワークの設定

「ブートプロトコル」画面で「スタティックIPアドレス」を選ぶと「TCP/IPの設定」画面になります。ここでは、マシンのIPアドレスなどを設定します。設定する内容がわからない場合は、ネットワーク管理者に問い合わせてください。

「ネットワークの設定」画面では、ドメイン名、ホスト名を設定します。また、必要に応じて、セカンダリネームサーバ、ターシャリネームサーバも設定します。

設定する内容がわからない場合は、ネットワーク管理者に問い合わせてください。

タイムゾーンの設定、サービス

「タイムゾーンの設定」画面では、世界標準時からの時差を設定します。日本国内で利用するなら「Japan」を選択します。同じマシンでほかのOSも利用する場合は「ハードウェアクロックをGMTに設定しますか？」にチェックをいれてはいけません。

「サービス」の画面では、Linuxのブート時にスタートさせるサービス（デーモン）を選択します。F1キーを押すと、そのサービス（デーモン）の説明を英文で表示します。よくわからなければ、デフォルトのまま「了解」を押して次に進んでも大丈夫です。

プリンタの設定、rootのパスワード、ログイン認証の設定

「プリンタの設定」画面で「はい」を選択するとプリンタの設定が行えます。

「rootのパスワード」画面では、システム管理を行うルートユーザーのパスワードを設定します。確認のため2回入力する必要があります。なお、セキュリティを確保するため、入力した文字は画面に表示されません。

「ログイン認証の設定」画面では、ログインする際のユーザー認証方式を設定します。通常はそのまま「了解」を選べばいいでしょう。NISサーバが設置されている場合はネットワーク管理者の指示に従ってください。

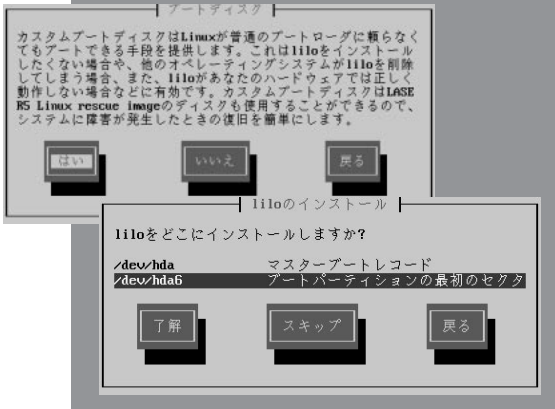
ブートディスク、liloのインストール、ブート可のパーティション

ブート用のフロッピーディスクを作成するかどうか選択します。緊急時に必要になりますので、できるだけ作成しておきましょう。

「liloのインストール」画面では、LILO（LInux LOader）のインストール先を選択します。Linuxのみ、あるいはLinuxとWindows 9Xのデュアルブートをする場合は「マスターブートレコード」を選びます。System Commanderなどのブートセレクタを利用している場合は「ブートパーティションの最初のセクタ」を選びます。

「ブート可のパーティション」画面では、LILOで起動可能にするOSを設定します。Windows 9Xは、デフォルトで設定されている“ dos ”というラベルのままで問題ありません。

続いてX Window Systemの設定になりますので、画面の指示に従って設定すれば、インストールは終了します。



LASER5 Linux ソースCD-ROMの入手方法

LASER5 Linux 6.0 Rel.2ソースCD-ROMが必要な場合は、以下の方法で入手してください。

・配布手数料：1枚1000円（送料・税込）
・有効期間：2002年2月7日まで
・支払い方法：郵便振替（振り込み料金は利用者負担）
口座：00110-0-150965
加入者名：レーザーファイブ株式会社

郵便振替払込取扱票の通信欄に「LASER5 Linux 6.0 Rel.2 ソース請求Linux magazine」と明記し、住所と氏名を記入のうえ、お申し込みください。入金確認後送付いたします。