News Express

Distribution Software Hardware Headline Event

テキストベースのインストーラはもう古い？

Red Hat Linux 6.0の発表から約半年たった10月4日に、次のバージョンであるRed Hat Linux 6.1がリリースされた。0.1だけ数字が上がったことで、マイナーバージョンアップのようだが、驚いたのは今度のインストーラが、今までのようなテキストベースのものではなくグラフィックス表示になったことだ。

すでにCalderaのOpenLinuxはグラフィカルインストーラになっていて海外では人気があるようだが、Red Hatがこれに追いついたというところだろうか。

インストーラなんて最初だけで、そうしょっちゅう使うものではないと思うのだが、そういうところにこそ力を入れて開発しているメーカーが、結局市場に受け入れられるのかもしれない。

この次はどのディストリビューションがグラフィカルインストーラになるのだろうか？

ところで、今年の3月に開かれたLinuxWorld Conferenceから、Linuxと名前が付かないPC関係の展示会にもLinuxコーナーが設けられていることが多く、毎月のようにLinuxの展示会が開かれているような気分だ。

下に掲載した今月のイベントは直接Linuxを扱ったものではないが、Linuxのビジネスをしている会社が結構出展している。Linuxの国内市場の規模がどれくらいなのか把握できないが、Windowsと比べるとはるかに小さいのは確かだろう。それでもこれだけの出展があるということは、いかにLinuxへの期待が大きいかをうかがわせる。

今月のイベント

イベント	COMDEX／Japan '99
内容	コンファレンス、展示会、セミナー
期間	1999年11月9日（火）～12日（金）
会場	幕張メッセ（日本コンベンションセンター）
お問い合わせ	ソフトバンクフォーラム TEL 03-5642-8433 http://www.sbforums.co.jp/comdex99/

イベント	NEC ExpressWorld'99
内容	WebComputing Solutions Linux特別企画コーナー
期間	1999年12月1日（水）～12月3日（金）
会場	東京ビッグサイト　東展示棟4～5ホール
お問い合わせ	日経BP社 NEC ExpressWorld事務局 TEL 03-5210-7058 http://bizevent.nikkeibp.co.jp/nec

Software　発売日　1999年11月26日

パーティショニングユーティリティ「Partition Magic 5.0日本語版」

URL　http://www.netjapan.co.jp/

発売　株式会社ネットジャパン
TEL　03-3864-5210
価格　1万5800円

ハードディスク上のデータを保持したまま、パーティションのサイズを変更できるユーティリティ「Partition Magic 5.0日本語版」がネットジャパンから発売される。本製品は、米PowerQuest社の製品をネットジャパンが日本語化したもの。

FAT16、FAT16X、FAT32、FAT32X、NTFS、HPFS、Linux ext2、Linux Swapの各ファイルシステムに対応しており、サイズの変更が可能だ。Linuxのext2ファイルシステムに対応しているため、Linuxユーザーにも有用なツールである。

以前のバージョンから可能だったFAT16からFAT32、FAT16からNTFSへのフォーマット変換に加えて、バージョン5.0からはNTFSからFAT16／FAT32への変換や、隣接したFAT16／FAT32パーティションの結合が可能になった。

また、複数のOSを1つのハードディスクにインストールし、選択して起動するBootMagicがバンドルされている。本製品により、ハードディスク全体をWindowsで使っているユーザーでも、Windowsのパーティションを縮小しLinuxをインストールして、2つのOSを共存させることが可能となる。

なおバージョン5.0は、Windows2000のNTFS5には対応していないが、来年のWindows2000リリース後に対応を予定している。

Software　発売日　1999年10月29日

3次元CGソフト「Shade for Linux Preview Kit」

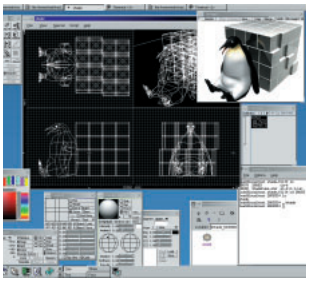
URL　http://www.ex-tools.co.jp/

発売　エクス・ツールス株式会社
TEL　092-722-4540
価格　9800円

エクス・ツールスは、3次元CGソフト「Shade」のLinux版を販売すると発表した。正式版に先だちプレビュー版「Shade for Linux Preview Kit」が10月29日に発売された。

「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として4900円で購入することができる。なお、マニュアルはPDFファイルとして提供される。

「Shade for Linux」は、国内で最大のユーザー数を誇る3DCGソフトウエア「Shade」のLinux版。X Window System上で軽快に操作できる。今回のリリースは英語版だが、ユーザーインターフェイスおよび形状データは、Windows／Macintosh版「Shade」との互換性がほぼ保たれており、同社では、ほかのプラットフォームからLinux版への移行も容易に行うことができるとしている。



「Shade for Linux」の仕様については、現在最終的な調整が行われているが、プラグインおよびスクリプトをサポートするなど、従来の「Shade」ラインナップのミドルレンジモデルに準じたものになる模様。プラグインは従来通り、C／C++での開発が可能で、ソースコードの流用も容易である。また、スクリプト言語はTclをサポートする。

動作環境は、Linux カーネル2.2のRed Hat Linux 6.0で、KDE 1.1が必要となる。

Software　発売日　1999年11月1日

Webフィルタリングソフト「日本語版Cyber Patrol Proxy for UNIX」

URL　http://www.ascii.co.jp/netmedia/

発売　株式会社アスキー
TEL　03-5351-8590
価格　98万円（2000ユーザーまで）

インターネット上の不適切なコンテンツの閲覧を規制するフィルタリングソフト「日本語版Cyber Patrol Proxy for UNIX」がアスキーから発売された。

同製品は、LinuxおよびSolarisで構築されたプロキシサーバ上で動作するフィルタリングソフトウェアである。サーバ上で一括規制を行うため、クライアントのOSやWebブラウザを問わない。

フィルタリングには、「CyberNOT（閲覧禁止）」「CyberYES（閲覧許可）」の2種類の規制リストを使用する。これらのリストは、常に最新のものがインターネット上のサーバからダウンロード可能だ。更新は1週間に1度行われる予定。それとは別に、毎日更新される閲覧禁止サイトのリスト（HotNOTリスト）も利用できる。さらに規制するカテゴリーの選択や、ユーザー自身が規制リストにサイトを追加、削除することもできるので、ユーザーの基準に合ったアクセス規制を実現できる。

閲覧禁止に設定されたサイトへのアクセスは、Linux、Solaris上で動作するSyslogサービスによって記録することができるため、不適切なサイトの閲覧を防止するだけでなく、そのようなサイトを見ようとしたユーザーの特定も可能だ。

Cyber Patrol Proxy自身の設定は、管理用の専用Webサーバプログラムにアクセスして行うため、Webブラウザが使えるマシンを用いて、ネットワーク越しにリモート管理ができる。

Hardware　発売日　1999年10月13日

コンパクトなキーボード「Happy Hacking Keyboard Lite」

URL　http://www.pfu.co.jp/hhkeyboard/

発売　株式会社PFU
TEL　0120-144-541
価格　オープン価格

PFUから、コンパクトなキーボード「Happy Hacking Keyboard Lite」が発売された。以前から発売されていた「Happy Hacking Keyboard」の操作性を継承しつつ、PS/2専用接続とし低価格化を計った製品。オープンプライスだが、PFUのWebサイトの通販では7800円となっている。

サイズは294mm（W）×38mm（H）×110mm（D）でA4判の半分となっている。重量は620g。しかしファンクション、カーソル、テンキーを持たないため、キーピッチは19.05mmと通常のキーボードと変わらない。背面のディップスイッチにより、Windowsキー、CAPS LOCKキーや変換／無変換キーの設定が可能だ。またオプションとして、キャリングケースが用意されていることからわかるように、持ち歩くことも想定している。

今後3年間で5万台の販売を見込んでいる。

Hardware　発売日　1999年10月下旬

100BASE-TX対応のプリントサーバ「Mini100」

URL　http://www.planex.co.jp/

発売　プラネックスコミュニケーションズ株式会社
TEL　0120-415-976
価格　オープン価格（参考価格1万7800円）

プラネックスコミュニケーションズから、100BASE-TX／10BASE-Tに対応したプリントサーバ「Mini100」が発売された。オープン価格だが、同社による参考価格は1万7800円。

対応OSは、Linuxをはじめとした各種UNIX系OS、Windows9x、WindowsNT4.0／3.5x、MacOS7.6以降、NetWare 3.x／4.xと多岐にわたる。TCP/IP、IPX/SPX、NetBEUI、AppleTalkの各プロトコルをサポートし、さらにプロトコル自動切り換え機能を持つため、多機種の混在したネットワーク環境でもプリンタ共有が可能だ。Auto Negotiation機能により、回線速度の切り替えは自動的に行われる。

サイズは92.8mm（W）×25.3mm（H）×58mm（D）で、設置場所で悩むこともない。付属の設定ユーティリティは、Windows9x／NT用のみだが、Webブラウザ経由の設定も可能なため、Windowsマシンがなくても各種設定を行える。

Hardware　発売日　1999年11月中旬

19インチラックにマウントできるサーバ「MS-102-3570」

URL　http://www.mitacsys.co.jp/

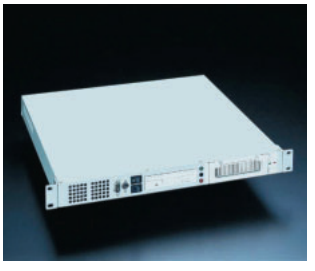
発売　マイタックシステム株式会社
TEL　03-5420-2885
価格　オープン価格

マイタックシステムは、厚さわずか44mmとうす型のサーバ用途コンピュータ「MS-102-3570」を発売する。サイズは482.6mm（W）×44.0mm（H）×482.6mm（D）で、19インチラックにマウント可能。

CPUはAMD K6-2 350MHz、メモリは64Mバイト（最大512Mバイト）、ハードディスクは内蔵3.5インチベイと前面の5.25インチベイを用いて、IDEタイプを最大2台搭載できる。VGA（S3Trio3D／VRAM 2Mバイト）、LAN（100Base-TX／10Base-T）を標準で搭載する。拡張スロットは、フルサイズのPCI／ISAカード（どちらか1枚）を搭載可能。

劣悪な環境下でも運用できるように、筐体の前面（フィルター付き）、後面に空冷ファンを合計3台設置し、さらにハードウェアモニタ機能により、CPU／システム温度／ファンなどの障害監視が行える。

同社ではWeb、DNS、FTPなどのサーバ用途などで、1年間で約1万台の販売を見込んでいる。



Hardware　発売日　1999年10月

超小型IPルータ「FutureNet CR-20」

URL　http://www.centurysys.co.jp/

発売　センチュリーシステムズ株式会社
TEL　0422-37-8911
価格　8万8800円

センチュリーシステムズは、超小型で携帯可能なIPルータ「FutureNet CR-20」を発売した。サイズは100mm（W）×28mm（H）×62mm（D）、重量も110gと軽いので、携帯して外出先で臨時にLANを組むといった使い方も可能だ。

DHCPクライアント／サーバ機能を持つため、DHCPサーバのあるネットワークに接続すれば、特に設定をすることなく利用できる。パケットフィルタリング機能を利用して、簡易ファイアウォールとして用いることもできる。SYSLOGによる運用ログの収集や、電子メールによる管理者へのログ送信も可能だ。その場合、ネットワーク上にSYSLOGデーモンやメールサーバを動作させているマシンが必要だ。詳細な設定は、本機にtelnetまたはWebブラウザで接続して行う。

同社では初年度約1万台の販売を見込んでいる。

富士マグネディスクが、アンチウイルス機能を含んだメールサーバを発売

1999年10月25日

富士マグネディスクは10月29日、ウイルス検知機能を最初から搭載したLinuxベースのメールサーバ「プロサーバ with アンチウイルス」を発売する。ギデオンが共同開発している製品で、価格は18万円（100ユーザーまで）。

プロサーバ with アンチウイルスは、同社のプロサーバ for Linuxに、米Data Fellowsのウイルス検出ソフト「FSAV Linux」を搭載したもの。1台のサーバマシンでメールサーバとウイルス検知サーバを動作させることができるため、メールサーバとウイルスチェック用サーバを別々に用意する方法よりも、ウイルス検出のためのシステム構築費用が安くすむとしている。さらに、従来のメールスキャン方式では脆弱になりがちだったSPAM、リレー、DoS攻撃に対しても、対策が練られているという

富士マグネディスク
(http://www.fujifilm.co.jp/fmd/)

Intelが Gibabit Ethernetカードで Linuxをサポート

1999年10月22日

米Intelは10月21日、Gigabit Ethernet（IEEE 802.3z）対応のネットワークカード「Intel PRO/1000」におけるLinuxの公式サポートを発表した。

Intel PRO/1000 Gigabitドライバはオープンソースとなり、IntelのWebサイト上で入手可能となる。これはLinuxコミュニティに向けて広く公開し、必要に応じてカスタマイズを可能にすることが目的という。

Intelの Network Interface Division、Marketing DirectorのTim Dunn氏は「ISPなどのWebベースの業務を行なう会社がIntelアーキテクチャのLinuxを利用しており、IntelとしてはLinuxネットワークへのソリューションを提供しなければならない。今回のGigabitネットワークのソリューションにより、広帯域の接続を提供することができる」と、語った。

Intel（http://www.intel.com/）

ATIがLinuxを公式サポート、開発情報を公開

1999年10月22日

グラフィックスカード製造の大手、カナダのATI Technologiesは10月20日、Linuxを公式にサポートする方針を発表した。ATI Technologiesは以前からXFree86を通じて、X Window Systemへの2Dにおけるサポートを行っていたが、今回の発表はさらに広く一般に開発情報を公開し、2D、3Dの分野におけるサポートを行うというもの。

具体的には、RAGE Pro、RAGE 128といった製品の3D機能、それらにRAGE IIを加えた製品群のビデオキャプチャ、テレビチューナー機能の情報を公開するという。

また今回、ATI Technologiesは米Precision InsightとLinuxサポートを強化するための契約を行った。Precision InsightはXFree86用のDRI（Direct Rendering Infrastructure）を開発し、3Dドライバを作成、2000年春にソースコードを公開することを目標とする。

ATI Technologies
(http://www.ati.com/)

IBMが、「Java 1.1.8 IBM Developer Kit for Linux」をリリース

1999年10月22日

米IBMは「Java 1.1.8 IBM Developer Kit for Linux」をリリースした。Java 1.1.8 for LinuxはIBMのJava for Windowsに劣らず高速で、現在のLinux用のJavaの中では最も高速であるという。

Java担当のSenior Marketing ManagerのJeff Robertsは「Linuxの成長は素晴らしい。このような高性能なJavaのインプリメントを、コミュニティに渡したり、DB2やWebSphereのようなJavaを使ったミドルウェア製品に役立てることを望む」と語った。

同製品は、現在IBMのWebページよりダウンロードすることができる。

IBM（http://www.ibm.com/）

ソフトボートがWebサーバログ分析&管理システムを発売

1999年10月22日

ソフトボートは10月21日、Linuxに対応したWebサーバ分析&管理システム「Webトレンド レポーティングサーバー」を11月26日より発売すると発表した。同製品は米WebTrendsが開発した製品で、価格は49万8000円。

同製品はWebサーバログ分析機能に加えて、アスセスしてきたユーザーごとに分析レポートの作成や参照の権限を設定できるユーザー管理機能、Webサーバのクラスタ対応機能を備えており、Web上のマーケティングなどに使うことができる。また、ホスティングサービスを行うISPが、同製品のレポート機能を使ってユーザーに分析レポートを提供するといった使いかたもできる。Linux版のほかSolaris版、Windows NT版が用意される。

株式会社ソフトボート
(http://www.softboat.co.jp/)

Win32アプリケーションのLinuxへの移植ツール「MainWin」発表

1999年10月20日

米Mainsoftは最小限の努力でWin32アプリケーションをUNIXに移植できる製品「MainWin」のLinux版を発表した。

MainWinは、Windowsのエミュレータや、WindowsのセマンティクスをMotifなどにマップするものではなく、UNIXにWin32 APIを実装したソフトウェア。同ソフトウェアより、Win32 APIを使用したアプリケーションをUNIXのコンパイラでリコンパイルすることができ、アプリケーションはUNIX上でネイティブに動作する。

MainWinによって、「Internet Explorer」「Outlook」「Unicenter」などがUNIX上で動作可能になるという。

製品版の出荷時期は2000年第1四半期。

なお、デモ版が数週間以内にダウンロード可能になるという。なお、同製品はすでにAIX、HP-UX、IRIX、Solarisなどのプラットフォームで動作している。

オープンソースの世界の例にもれずMainWinにも、同じ機能を実現しようとしているソフトウェアが存在する。「Wine」という名のソフトウェアは、Win32 APIを実装し、Win32アプリケーションを UNIXに移植するための開発ツールキットと、Win32アプリケーションのバイナリ互換性を提供している。

Wineはクリーンルームによる「もうひとつの」Win32 APIの実装で、MainsoftがWindowsソースコードのライセンシーであるのと対照的だ。そのぶんWineの開発速度は速いとはいえないが、Corel社がWordPerfectやCorelDRAWなどをLinuxに移植するためにWineの開発に協力を発表していることもあり、近いうちにWineも実用的なレベルに達するかもしれない。

Mainsoft（http://www.mainsoft.com/）

internet.comがLinuxTodayを買収

1999年10月20日

米internet.comは10月19日、米LinuxTodayの買収を発表した。LinuxTodayは、Linuxとオープンソースにおける最大のWebサイトの1つ。

internet.comのCEO、Alan M. Mecklerは「LinuxToday.comを我々のLinuxPlanet.comとひとつにすることによって、internet.comはオープンソースマーケットにおいてリーディングポジションを獲得することになる」、また「インターネットはオープンソースの成長を加速した。internet.comはこの開発の継続的のためのリソースを提供する」と語った。

internet.comは先週、Sun Solaris systemsの管理するSolarisGuide.comの買収も完了している。

internet.com（http://www.internet.com/）

日本SGIがLinuxなどの教育コースを開始

1999年10月15日

日本SGIは、東京都青山のカナダ大使館ビル2階に最新のコンピュータ技術を修得できる「SGI東京ラーニングセンター」（略称TLC）を開設したと発表した。

同社では今まで横浜市で研修センターを運営していたが、教育事業を強化拡充するために移転を行い、さらに将来の中核OSとして推進しているLinuxの教育コースを新設した。また定期コース以外にも、個別の教育ニーズに対応したオン・サイトトレーニングも実施していく。

今回同社では、教育および教育コンサルティング事業の強化拡充において、クリーク・アンド・リバーと業務提携を行った。それにより、SGI独自の認定インストラクタ制度もスタートさせ、Linuxでもインストラクタの認定を行い、全国規模での教育コンサルティング・ビジネスを展開していくという。

日本SGI（http://www.sgi.co.jp/）

アスキーがDTPシステムを無償公開

1999年10月15日

アスキーは10月15日、LinuxやFreeBSD上で動作するDTPシステム「EWB（エディターズワークベンチ）」を無償公開すると発表した。タグを埋め込んだテキストファイルをEWBで処理することにより、レイアウトされたPS（ポストスクリプト）ファイルを生成し、そのまま印刷用フィルムに出力することができるようになる。そのため編集時にレイアウトを考慮した編集を行うことができ、別途レイアウトを行う必要がないため、編集作業の効率化を図ることができる。さらに、PDF化などの出版物の電子化も容易になる。

EWBは今までにも、アスキー内部で出版物、マニュアルなどの製作に利用されていた実績がある。

内部的には、TeX、dvips、Ghostview、XEmacs、Tcl/Tkといったフリーのプログラムと、アスキー独自のプログラムを組み合わせたシステムとなっている。アスキーではこれまでTeXの日本語化を行い、公開していた。

これにともない、同システムの普及を狙うEWBコンソーシアムが設立された。コンソーシアムには現在、アスキー、共同印刷、共立印刷、ソフトバンクパブリッシング、大日本印刷、東京書籍印刷、図書印刷、凸版印刷、日経BPが加盟している。

アスキー（http://www.ascii.co.jp/）

三洋電機が医療用ワークステーションのOSとしてTurboLinuxを大量採用

1999年10月14日

米TurboLinuxは10月12日、三洋電機のNewveと呼ばれる日本での医療用ワークステーションのOSに、TurboLinuxが採用されたと発表した。これは三洋電機が、既存のプロダクトラインアップにTurboLinuxを有力なOSとして追加するもので、モニタ、プリンタ、および医療用ソフトウェアが付属し、価格は1台245万円から。今後4年にわたって2万台が製造される予定だという。

TurboLinuxのエンジニアによる医療用アプリケーションの移植およびテストには約1年がかけられているという。また、三洋電機によると、「TurboLinuxを選択した理由は、日本における開発チームが強力なため」だという。

TurboLinuxによると、今回のTurboLinuxの供給の規模は、Linuxとしては世界中でも最大のものだという。

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

日本Sambaユーザ会発足

1999年10月13日

1999年11月12日、「日本Sambaユーザ会」が発足することとなった。略称は「Samba-JP」。SambaはUNIX系OSのマシン上で、Windows上で使われているファイル共有、プリンタ共有などのサービスを提供するソフトウェア。年内にリリース

予定のSamba3.0では、Windowsのドメインコントローラの正式サポート、LDAPの正式サポートなどが実現される予定。

IntelとNokiaがLinuxを利用したインターネットTVの開発で協力

1999年10月13日

米Intelは10月12日にフィンランドのNokiaと、Linuxを利用したインターネットデジタルTVの開発で協力をしていくと発表した。

最初の製品は、オープンなデジタルビデオ、インターネット・プロトコル、ATVEF（Advanced Television Enhancement Forum）の制定した規格、LinuxとMozillaブラウザをもとに開発され、2000年の後半に発表される予定。

ATVEFは、コンテンツ作成者がマルチプラットフォームで動作するコンテンツを作成するための、基本的な仕様を定めることを目的とした団体。

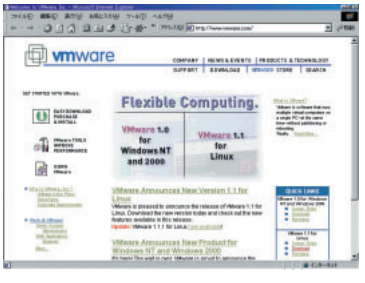
NokiaのHeikki Koskinen社長は、「Linuxのような既存のインターネットベースのOSで動作する製品を作ることにより、新たなインターネットベースのサービスのプラットフォームを提供できる」とした。

Intel（http://www.intel.com/）
Nokia（http://www.nokia.com/）

VMwareの新バージョンがリリース

1999年10月12日

米VMwareは10月11日、「VMware」の新バージョンである「VMware 1.1」のリリースを発表した。「VMware」はPCアキテクチャをLinux上でエミュレーションするソフトウェア。　今回のリリースの注目点はWindows 2000のエミュレーションのサポート。その他にも、スキャナ、プリンタ、ZIPドライブ等の周辺機器のサポートも向上したとしている。



VMware 1.1は、VMwareのWebページ上からダウンロードすることができる。

VMware（http://www.vmware.com/）

IntelほかがTurboLinuxに出資

1999年10月12日

米TurboLinuxは10月12日、米Intel、ベンチャーキャピタリスト（投資会社）AugustCapital、投資銀行Broadview International LLCから出資を受けたことを発表した。出資の詳細については発表されていない。

TurboLinuxのCEO、Cliff Miller氏は「今回の出資によって、最先端の技術を開発する環境が拡大し、ハイパフォーマンスLinuxがさらに普及することを確信している」とした。

Microsoftが、LinuxとWindows NTを比較した文書を公開

1999年10月8日

米Microsoftは10月4日（現地時間）「Linux Myths（Linuxの神話）」という文書を、同社のWebサイト上で公開した。これはパフォーマンス、信頼性、コスト、セキュリティ、デスクトップの能力、という5点をあげ、「LinuxがWindows NTに対して優位である」ことは神話であり、現実にはWindows NTの方が優れているとするもの。

同社では、以前にもLinuxとWindows NTを比較したベンチマークテストの結果を公開している。

Microsoft「Linux Myths」
http://www.microsoft.com/ntserver/nts/news/msnw/LinuxMyths.asp

日本オラクル、「Oracle8i WorkgroupServer for Linux Release8.1.5」の出荷を発表

1999年10月8日

日本オラクルは「Oracle8i Workgroup Server for Linux Release8.1.5」を、1999年11月30日から出荷開始すると発表した。価格は22万円（5同時ユーザー）からとなっている。

またNECでも、日本オラクルのビジネス・アライアンス・パートナーとしては初めて、「Oracle8i Workgroup Server for Linux Release8.1.5」のOEM販売を1999年11月30日より開始する。日本国内での製品出荷にあたり、NECと日本オラクルは「NEC Express5800シリーズ」を使用した共同評価体制を確立する。

さらに、NECソフトウェアでは、「Oracle8i Workgroup Server for Linux Release8.1.5」と「NEC Express5800シリーズ」を使用したコンサルティングおよび設計・構築サービスである「Oracle8iデータベースサーバソリューション・サービス」を製品出荷と同時に開始する。

日本オラクル（http://www.oracle.co.jp/）

漢字対応16進数表示ツール「デ変研DUMP」無償配布開始

1999年10月6日

データ変換研究所（略して「デ変研」）は、漢字対応の16進数表示ツール「デ変研DUMP」をWeb上で無償配布した（容量は301Kバイト）。

デ変研DUMPは、ファイル解析ツール。ダンプしたファイルの先頭から何バイト目に16進数があるのかを調べ、漢字（と16進数）で表示できる。たとえばMicrosoft Wordのファイルフォーマットで作られたファイル（.docファイル）を解析し、テキスト部分を抽出する、といった使い方をする（これはあくまでも例なので、実際に.docファイルに適用できるかどうかは別問題）。

また、フロッピーディスクをはじめとしたメディアのファイルシステムをダンプし、フォーマットを解析して別のファイルシステムにコンバートする、といった作業などでも使われる。

対応する漢字コードは、EUC、SJIS、JIS、Unicode（UCS2、URF-8）。日本語漢字ターミナル上で動作する。

同社では、製品として出荷されている日本語マルチコード対応エディタ「デ変研TEXT」の販促として、無償配布を行ったとしている。

データ変換研究所
（http://village.infoweb.ne.jp/~dehenken/）

Distribution ▶▶▶

「Red Hat Linux 6.1日本語版」11月12日より発売開始 緊急速報！

レッドハット株式会社は10月28日に、「Red Hat Linux 6.1日本語版」を11月12日より発売すると発表した。同製品は日本語を標準言語としたインターナショナルバージョンで、メッセージファイルが日本語となっているほかは、英語版と変わらない動作が得られるとしている。

パッケージは、バイナリCD、ソースCD、商用ソフトウェアCDの、計3枚のCD-ROMとブートフロッピーで提供され、価格は1万2800円。商用ソフトウェアには、ATOK12 SE、Wnn6 ver.3、dpNOTE、ダイナラボの商用フォント5書体がバンドルされている。

また、パッケージには、30日間の電話サポート、90日間の電子メールサポート、180日間のオンライン優先アクセス権が含まれている。日本語化されたグラフィカルインストーラも用意されている。

「Kondara MNU/Linux 1.0」が12月1日より発売

デジタルファクトリから「Kondara MNU/Linux 1.0」が12月1日より発売される。価格は6800円。Kondara MNU/Linux 1.0は、Kondara project（代表: 保科 徹）が開発したディストリビューションで、Red Hat Linuxをベースに日本語環境を整えたもの。PC/AT互換機用、Alpha PC用、ソースプログラムのCD-ROM3枚組と、フロッピーディスク1枚、マニュアル1冊（150ページ）で構成される。

kernel 2.2.13または出荷時の安定最新版を採用し、gibc2.1.2＋日本語localeデータというように常に最新バージョンを、いち早く取り入れていくディストリビューション。

Kondara MNU/Linux 1.0（http://www.kondara.org/）

米COMDEXで「Corel Linux」発表、同時にダウンロード版公開

カナダのCorelは、11月15日より米Las Vegasで開催されるCOMDEXにて、正式に「Corel Linux」を出展し、無料ダウンロード版を公開すると発表した。Corel Linuxは Debian GNU/Linuxをベースとしたディストリビューションで、ベータテストで配布されたPreview1では、わずか4ステップで終了するインストーラや、パッケージマネージャ“Get-It”などCorelが独自に開発したソフトウェアが付加されていた。

Corel（http://www.corel.com/）

「TurboLinuxアップデートRPMパック19991015」を公開

ターボリナックス ジャパンは10月15日に、TurboLinux日本語版4.0およびTurboLinux PRO日本語版4.2のシステムを最新版にアップデートする「TurboLinuxアップデートRPMパック19991015」をリリースした。登録ユーザーには実費（1000円/送料・税込み）でCD-ROMを送付する。また、雑誌付録やFTPからのダウンロードでも提供する。

主なアップデート内容は、XFree86-3.3.5、Ntool-1.1.2を追加（nmail）、APMカーネルを選択したときに電源が落ちない問題を修正、インストーラを修正（ビデオカード選択時の文字化け、PCカードの認識の問題）、samba-2.0.5a日本語対応パッケージに更新、KDE-1.1.2関連パッケージの追加及びアップデート、ALPS MD & MD2Kドライバの追加、pdf_sec.psのリプレース（暗号規制対策）、kinput2-v3にアップデートなど。

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

グラフィカルインストールが可能になった「Red Hat Linux 6.1」発表

米Red Hat Softwareは、10月4日に同社のディストリビューションの最新リリース「Red Hat Linux 6.1」を発表した。カーネルは2.2.12。インストールおよびシステム管理面の機能を向上させた。製品版パッケージにはSun MicrosystemsのStar Office 5.1をバンドルした。

新機能「Red Hat Update Agent」で、同社のFTPサイトからソフトウェアの自動更新が可能。さらに、LDAPを実装し、Wired for Management（WfM） 2.0をサポートし、WfM2.0対応のPXE（Preboot Environment）ブートによるリモート管理機能を実現した。

パッケージには、バイナリCDとソースCDのほかに、Standard版はStar Office 5.1aが追加されて29.95USドル。Deluxe版はStar Office 5.1aとアプリケーションCD2枚が追加されて79.95USドル。Professional版はアプリケーションCD5枚が追加され149.95USドル。

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

Products

Linuxで稼働するカスタマイズ可能な超小型Ethernetルータ

LAMB-RT-01（子羊ルータ）

最近広まりつつあるCATV網やxDSLによるインターネット接続で複数台のPCからインターネットを同時に利用するには、EthernetからEthernetへのルータが必要になる。LAMB-RT-01はこうしたニーズを満たすべく開発されたLinuxで稼働する超小型ルータである。

製品名	LAMB-RT-01
価格	5万5000円（税込）
問い合わせ先	有限会社ワイルドラボ lamb-info@wildlab.com http://www.wildlab.com/

CATV網やxDSLによるインターネット接続では、市販のISDN TAやモデムではなくケーブルモデムなどの専用デバイスを用いてPCと接続する。そのため複数台のPCを接続する場合、市販のISDNルータは利用できない。またケーブルモデムとPCの間はEthernetで接続されるが、一般的にIPアドレスは1つしか割り当てられないので、そのままでは複数台のPCを接続できない。

LAMB-RT-01は、PCとケーブルモデムの間に接続することで、複数台のPCからインターネットを同時に利用したり*1、外部からの侵入を防いだり、といった機能を実現するEthernet間のルータである。今回は開発途中の試作機にて評価を行った。

小型ケースにPCとLinuxを集約

本機の最大の特長は、**写真1〜3**でもわかるように、非常にコンパクトなケースを採用していることだ。専有面積は3.5インチフロッピー程度の大きさで、設置場所には困らずにすむ。また発熱が少なく空冷ファンがないほか、ハードディスクも使っていないので動作中も静かだ。家庭で使うにはありがたい。

これほど小さいにもかかわらず、本機自体はLinuxで稼働している。内蔵のフラッシュメモリにLinux kernel 2.0.37が格納されており、そこからブートしてLinuxが起動し、ルータとして働くのだ。ローカル側ネットワーク（家庭・SOHO側）とグローバル側ネットワーク（CATV・局側）の間でIPパケットを相互にやり取りするのは、LinuxのIP masqueradingである。プライベートアドレスからグローバルアドレスへの変換もサポートしており、試用機にはFTPやReal Audio、IRCなどのアプリケーションをIP masquerading経由で使うための

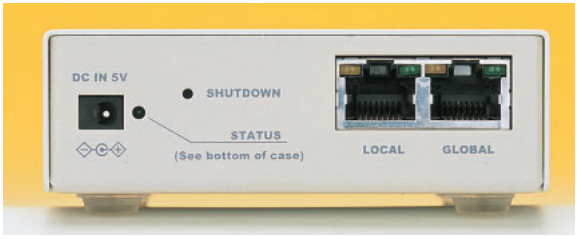


写真1
本機側面。右側からEthernetのコネクタ（10BASE-T）、シャットダウンボタンを押す穴、ステータス表示LED、電源コネクタが並ぶ。電源を接続するとLinuxがブートし、完全に起動するとステータスLEDが点灯する。

写真3
内蔵基板の表面。上に2つ並んでいるチップはNE2000互換Ethernetコントローラ。その下がコンパクトフラッシュのスロットと16Mバイトのフラッシュメモリで、ハードディスクの代わりに使用されている。なお基板上のジャンパ線は製品版ではなくなる。

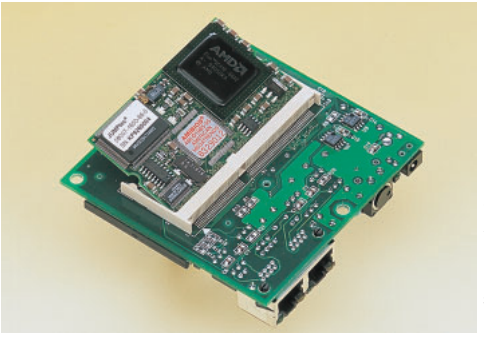


写真2
本機に内蔵の基板の裏面。本機の小型化に大きく貢献しているのが、SO-DIMMソケットに装着されているJUMPtec社のDIMM-PCというワンボードPCだ。CPUはAMDの組み込み向けi486互換チップで、16Mバイトのメインメモリと AMI BIOSを組み込んだROMを搭載する。これがLinuxを動かしている中枢である。

モジュールも組み込まれていた。

IPパケットレベルのフィルタリングにはipfwadmを採用しており、送信元／受信先アドレスやポート番号などを指定してさまざまなフィルタリングが可能だ。そのほか、ローカル側ネットワークに対してはDHCPサーバとして、グローバル側ネットワークにはDHCPクライアントとして動作できる（固定アドレスも可）。DHCPサーバは、20個のIPアドレスをダイナミックにリースできる（固定アドレスも含めると利用できるのは192.168.1.2～254まで）。ログ機能としては、Linuxのsyslogに対応しており、別のLinuxマシンにログを転送できる。

ルータの各種設定は、**画面1**のように基本的な設定はWebから操作できる。一方、IP masqueradingやパケットフィルタリングなど詳細については、telnetでログインして直接コマンドラインから設定する必要がある（FTPでファイルの転送も可能）。逆に言えば、Linuxの知識を持っているユーザーならば、本機を自由にカスタマイズして自身の環境に最適化できるわけだ。ハードウェア面でも、フラッシュメモリをより大容量のものに交換するというカスタマイズが可能だ（ただし無償保証はなくなってしまうが）。

画面1
本機の設定用Webトップページ。グローバル側では本機やDNSサーバのIPアドレスを、ローカル側ではDHCPサーバの設定を変更できる。そのほかはログ取得の設定ぐらいで、設定メニューとしては非常にシンプルにまとめられている。

SOHOでのファイアウォールに最適

Ethernetルータといえば業務用ばかりで高価なものが多かったが、本機の価格は5万円台と安く、家庭やSOHOでも比較的導入しやすい点は評価できる。またルータやファイアウォールといった用途では、PCでLinuxサーバを構築するより、本機を導入するほうがコストや占有面積の点で有利だ。Linuxの設定をよく知っているユーザーには、カスタマイズできるという点で特にお勧めできる。

＊1　CATVやxDSLの事業者によっては、複数台のPCを接続することを許可していなかったり、別の契約が必要だったりすることもある。

CPU	組み込みシステム向け486互換CPU（AMD Elan SC410-66MHz）
メインメモリ	16Mバイト
ストレージ	フラッシュメモリ16Mバイト（コンパクトフラッシュ TypeIIスロット）
Ethernet	10BASE-T ×2ポート（ローカル／グローバル）
OS	Linux kernel 2.0.37 （MLD IIIベース）
NAT機能	LinuxのIP masquerading
ファイアウォール	IPパケットフィルタリング（Linuxのipfwadm）
サーバ機能	DHCP（ローカル側）、telnet、FTP、Web（設定用）
ログ機能	syslog、Web
外形寸法（mm）	93（W）×91（D）×30（H）
電源	外付け（ACアダプタ）

表1　LAMB-RT-01の主な仕様

NEC LaVie NX（LW333D/74DA）

ノートPCでLinuxを利用してみたいけど、大手メーカーからLinuxプレインストールマシンは発売されていない。そうかといっても自分でインストールするのは大変そう、という人にNECのノートPC、LaVie NXにLinuxをセットアップしてくれるサービスが登場した。

製品名	NEC LaVie NX（LW333D/74DA） Linuxセットアップサービス
価格	オープン価格
問い合わせ先	NECパーソナルシステム株式会社 PC販売本部 TEL 03-3331-8941 http://linux.nec-ps.co.jp/

NECの関連会社であるNECパーソナルシステムでは、VALUESTAR NXとLaVie NXにTurboLinux日本語版4.0をインストールする「Linuxセットアップサービス」を行っている。これは、ユーザーがPCを購入するときに販売店で申し込むと、いったんPCを預かり、NECパーソナルシステムでLinuxをインストールしてから出荷するというものである。また、秋葉原にあるNEC Bit-INN東京に持ち込むと、その場でセットアップサービスを実施してくれるので急いでいる人は、こちらに聞いてみるのもよいかもしれない（要電話予約）。

Linuxセットアップサービスの価格はオープン価格。なお、Bit-INN東京でのセットアップQuickサービス基本料金は2万6000円である。

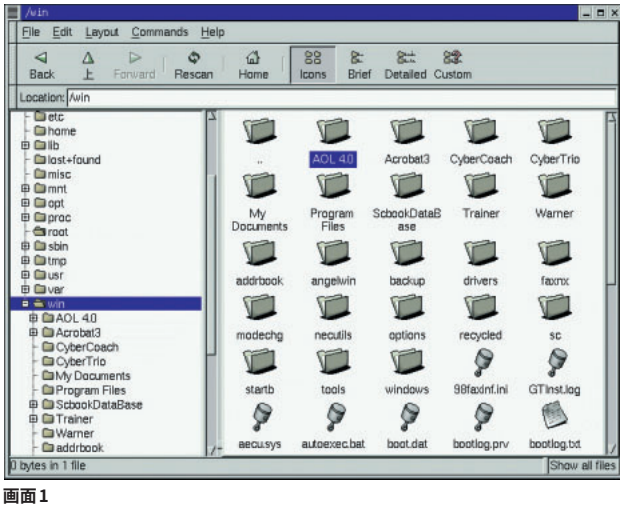
すべての機種に対応しているわけではなく、同社がX Window Systemの動作や、そのほかの周辺装置の動作状況を確認した機種だけがLinuxセットアップサービスに対応している。

Windows 98とのマルチブート環境

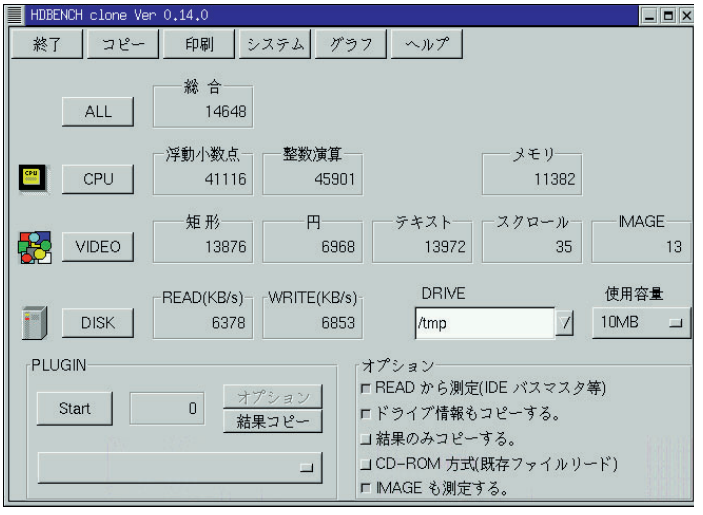
今回は、そのLinuxセットアップサービスの対象機種であるLaVie NXのA4サイズオールインワンノートLW333D/74DAを試用した。CPUはMobile Pentium II 333MHzで、メモリは64Mバイト、ハードディスクは10Gバイトと大容量である。TurboLinux日本語版4.0の製品版がセットアップされており、Windows 98とのマルチブートの環境になっている。

まず電源を入れブートすると、System Commander Liteによって、Windows 98とLinuxを選択することができる。Linuxを選ぶと、見慣れた起動画面が表示されていき、グラフィカルログイン画面になる。rootアカウントのパスワードに、あらかじめ設定されていたものを入力しログインすると、デスクトップ環境のGNOMEが立ち上がる。ルック&フィールに違いはあるものの、Windowsユーザーもそんなに違和感なく使えるのではないだろうか。

たとえば、Windowsのエクスプロ



画面1
GNOMEのファイルマネージャで見た/winディレクトリ。Windowsのパーティションが、Linuxの/winディレクトリにマウントされている。



画面2
Linux用ベンチマークプログラムHDBENCH cloneを実行してみた。

写真1
本体左側には、24倍速のCD-ROMドライブと3.5インチフロッピーを内蔵している。

写真2
PS/2、パラレル、USB、シリアル、アナログRGB出力の各コネクタが付いている。

写真3
本体右側には、TYPE IIのPCカードスロット×2と、モデム用モジュラーコネクタがある。

ーラに相当するGNOMEのファイルマネージャを使って、/winディレクトリを見たところが**画面1**である。この/winディレクトリは、基本パーティション（/dev/hda1）にあるWindowsのファイルをマウントしていて、見慣れたファイル名があるのがおわかりだろうか。

Linuxでのベンチマークテストも行ってみた。本誌104ページからの「Free Application Showcase」でも紹介されているHDBENCH cloneを実行した結果が、**画面2**である。ほかのマシンの結果の数値と比べると、CPUクロックに応じたまあまあの値といえよう。

Linux用パーティションは2Gバイト固定

10Gバイトのハードディスクは、**表1**のようなパーティションに分けられていた。ハードディスクの容量が違う場合でも、Linux用に確保されるのは、約2Gバイトと決まっていて、ユーザーの希望には応じていない。これはサポートの手間を減らすためだと思われる。

プリンタは、PICTYやTurboLinux日本語版4.0がサポートしているプリンタであれば、設定してくれる。サウンドはTurboLinux日本語版4.0の製品版に含まれているOSS（Open Sound System）がサポートしている。

すでに購入したPCを持ち込んでLinuxをインストールしてもらうことも可能だが、ハードディスクの内容をすべて初期化してから、Windows 98とLinuxをセットアップするので、必要なデータやプログラムはすべてバックアップを取っておく必要がある。

また、FAXモデムやEthernetカードも機種は限られるが、一緒にセットアップしてもらうことが可能だ（有料）。なお、セットアップサービスの料金にはTurboLinux日本語版パッケージの料金も含まれており、製品版のパッケージがそのまま提供される。

セットアップサービスが対応していないPCに、ユーザーがLinuxをインストールするのならば、NECのWebページ（http://www.pc98.nec.co.jp/linux/）に掲載されている動作確認情報を見るとよいだろう。

最近のディストリビューションでは、インストール時に認識するハードウェアが多くなってきているが、最新のビデオカードへの対応やUSB、IEEE-1394などの比較的新しい規格には対応していなかったりする。安心して使えるLinux環境を、手間をかけずに手に入れられるなら有料でも高くはないはずである。

デバイス名	割り当て容量（Mバイト）	パーティション名
/dev/hda1	3152	Win98 FAT32
/dev/hda5	136	Linux Swap
/dev/hda6	2101	Linux
/dev/hda7	4430	Win98 FAT32

表1 ハードディスクのパーティション設定

写真4
A4サイズのノートPC。キーピッチは19mm。

CPU	Mobile Pentium II 333MHz
メモリ	64Mバイト
ハードディスク	10Gバイト
CD-ROM	内蔵（24倍速）
フロッピードライブ	内蔵（3.5インチ）
ビデオコントローラ	NeoMagic NM2160（VRAM 2Mバイト）
液晶ディスプレイ	14.1インチカラーTFT（XGA 1024×768、65536色）
モデム	内蔵（最大56Kbps／FAX 14.4Kbps）　Lucent Mars2（1646）
サウンド	内蔵（PCM、FM音源、MIDI音源）　ESS Maestro2（ES1968S）
キーボード	JIS標準配列（90キー）、キーピッチ19mm
マウス	NXパッド
外部インターフェイス	PS/2、パラレル、シリアル、アナログRGB、IrDA、USB
PCカードスロット	Type II ×2スロット（CardBus／ZVポート対応）
外形寸法(mm)	305（W）×41（H）×250（D）
重量	約3.2kg
OS	Windows 98、TurboLinux日本語版4.x
価格	オープン価格

NEC LaVie NX（LW333D/74DA）ハードウェアスペック

LinuxWorld Expo／Tokyo'99

9月29日、30日の両日、東京ファッションタウンにおいてLinuxWorld Expo／Tokyo'99が開催された。2日間で約1万5000人の来場があった会場には、ソフト、ハードのベンダーや、Linuxを中心に据えたビジネスソリューションを提供する合計78の企業や団体が出展した。またこれら出展企業や団体によるワークショップや、コミュニティを代表する人々による特別講演もあわせて行われた。

この展示会は、シンボルマークのペンギンがネクタイをしていることからも分かるように、ビジネス色の濃いものである。来場者もスーツにネクタイのビジネスマンが目立った。

しかし、ペンギンのぬいぐるみがいたるところに置いてあったり、着ぐるみのペンギンまでいたりするため、会場全体がなんとなくなごんだ雰囲気に包まれていた。こういう展示会もLinuxならではだろう。

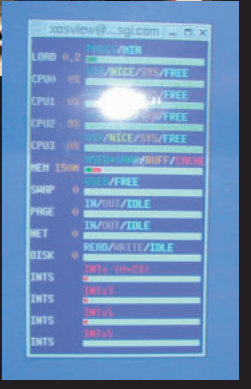
大手メーカーが揃って参加

国内の大手コンピュータメーカーは、各社揃って出展していた。特にNECや富士通は、グループ企業と合同で大きなブースを構えていた。どの会社もLinuxをインストールしたサーバ機を、単に売るだけでなく、Linuxユーザーへのサポートやインストールサービス、またLinuxサーバとOracleのようなデータベースを組み合わせたソリューションプロバイダ的ビジネスを目指しているようだ。

このようなビジネスでは、ハードウェアは自社グループの製品に限定せずに、用途ごとに最適なものが選ばれていく。たとえばNECのブースには、SOHO用サーバとしてCobalt Qube2が展示してあった。

ハードウェアでは、Linuxを利用したサーバ用の製品が多く展示されてい

国内の大手メーカーがいっせいにLinuxビジネスに参入してきた。現状は、各社とも新たなビジネスモデルを模索中といったところか。



SGI 1400L。一見普通のredhat LinuxのGNOMEデスクトップだが、xosviewにはCPUのグラフが4本!!

た。いかにもサーバ然としたごついマシンもあったが、注目が集まるのはやはりユニークなデザインのマシンであった。中でもSOHO用サーバのblue grass（アクアリウムコンピューター）やCyberCom PathNavigator（富士通テクノシステム）が、ひときわ目立っていた。

これらの小型サーバは、電源やスペースに制限のある小規模なオフィスでも問題なく使用できるため、今後流行していくだろう。

またハイエンドでは、Pentium III Xeonを最大4個搭載できるSGIの1400Lが、デザインと性能の両面から人目を引いていた。

クライアントPC用ソフトの増加

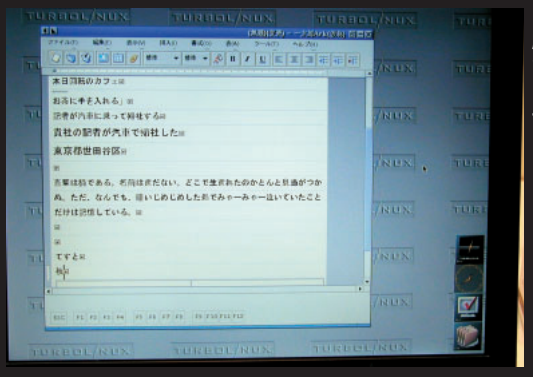
サーバが主体のハードウェアに比べてソフトウェアでは、オムロン、IBM両社が展示していた翻訳ソフトを始めとする、デスクトップ環境向けの製品が見受けられた。また参考出品ではあるが、IBMがホームページビルダー2001のLinux版を展示していた。さらにジャストシステムの一太郎Ark for Java（β版）もTurboLinux上で動作しており、これらが市場に揃うことで「デスクトップPCにLinux」という選択肢が、より現実味をおびてくると思われた。

ディストリビューション各社ももちろん出展しており、それぞれ賑わっていた。中でも日本語版の開発を発表したCaldera Systemsと、PowerPC版、Alphaプロセッサ版を参考出品していたターボリナックスジャパンは、多くの来場者を集めていた。

Linuxビジネスの定着

全体を通して感じたのは、Linux関連のビジネスが非常に幅広く多方向に展開しているということだ。ハードウェアは、ハイエンドからローエンドまでのサーバ製品に加え、Alphaプロセッサマシンが多くのブースで展示してあり、x86版と同じGNOMEのデスクトップが表示されていた。使い慣れたインターフェイスで、Alphaの強力な数値演算能力が利用可能になっている。ソフトウェアも以前からあったサーバ用途の製品に加え、デスクトップ環境向けの製品が揃いつつある。

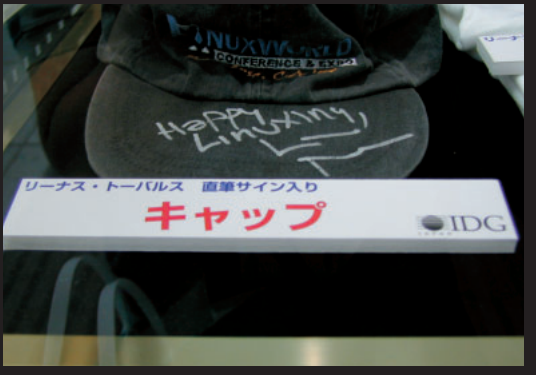
今年3月のLinuxWorld Conferenceでは、2日間で約1万人の来場があった。その半年後のLinuxWorld Expoに、その1.5倍の来場者が集まったということは、日本でのLinux人気が一時的でバブリーなものではなく、「Linuxでビジネス」という考え方が確実に定着しつつあることを表しているのではなかろうか。



一太郎Ark for Java（β版）の画面。ワープロとして必要な機能は、一通り揃っている。ESCメニューが一太郎シリーズの証である。

ホームページビルダー2001 for Linux（参考出品）。Windows版より少しだけ動作が遅いらしい。

ターボリナックスジャパンのブース。Alphaマシン、iMac、PCで同じ画面が並んでいるのは、とても不思議な感じがする。



サイレントオークションにかけられていた、リーナス・トーバルズ氏サイン入りの帽子。いくらで競り落とされたのか気になる。

Red Hat Linux 6.1（英語版）リリース

いち早くkernel 2.2、GNOMEを採用したRed Hat Linux 6.0がバージョンアップし、Red Hat Linux 6.1として登場した。最新の機能はもちろん、グラフィカルなインストーラで、よりインストールしやすくなった。

米Red Hat Softwareは、10月4日（現地時間）にRed Hat Linux 6.1を発売した。Red Hat Linux（以下Red Hatと略）は、RPM（Red Hat Package Manager）というソフトウェアパッケージ管理システムや、システム管理ツールLinuxconfなどを導入し、初心者でもLinuxのシステム管理がしやすいディストリビューションとして知られ、米国でトップのシェアを誇っている。RPMは、その扱いやすさからさまざまなディストリビューションでも採用されており、ソフトウェアパッケージ管理システムの標準になりつつある。

Red Hatは、新しい機能をいち早く導入することでも知られており、今年5月に発売されたRed Hat 6.0では、ほかのディストリビューションに先駆けてシステムにkernel 2.2、glibc 2.1を、デスクトップ環境にGNOME、KDEを採用して我々を驚かせた。

Red Hat 6.1は、6.0をベースに、さまざまなコンポーネントを最新版に置き換えるとともに、今回新たにグラフィカルインストーラ、ハードウェア自動認識機能を搭載したものだ。

日本語には対応していないので、デスクトップ環境で日常的に使うのにはややつらいかもしれないが、サーバ用途など、日本語環境が必要ない場合には、リソースを効率よく利用できるので、かえって好都合だ。

日本語版は、Red Hat Software社の日本法人、レッドハット株式会社から11月中にも発売される予定だ。

●新しいインストーラ Anaconda

Red Hat 6.0までのインストーラは、テキスト画面によるユーザーインターフェイスを採用していたが、6.1ではX Window Systemで動作するグラフィカルインストーラ「Anaconda」が利用できる。GUIによるとっつきやすさと、画面の左側に表示されるヒントが初めてインストールする人にも心強い。また、デスクトップ環境にGNOMEとKDEのどちらを利用するかを簡単に選べるようになっているのも嬉しい。

CD-ROMや、フロッピーディスクからインストーラを実行すると、ビデオカードの種類を自動検出し、適合するXサーバを起動する。不明なビデオカードが検出されると、16色VGAで表示するようになっているが、それでも表示できない場合は、従来と同様のテキストインストーラを利用することも可能だ。

また、NFSやFTPなどを介したネットワークインストールをする場合は、ネットワークインストールディスク（機種によってはPCMCIAインストールディスク）を作成する必要がある。

Anacondaのルック&フィールとインストールの詳細は、次ページからのインストールガイドを参考にしてほしい。

●ハードウェア環境を検出／設定

このバージョンで新たに採用されたのが、ハードウェア環境を自動検出するプログラム「kudzu」だ。このプログラムは、システムの起動時にハードウェアを自動検出し、環境が変更されていると、変更内容を表示する。ユーザーは、変更内容を確認し、設定することができる。

kudzuは、ハードウェア環境を/etc/sysconfig/hwconfというファイ

ルに保存しておき、起動時に得られた情報と、このファイルに保存された情報を比較することで、環境の変更チェックを行う。初回起動時など、/etc/sysconfig/hwconfが存在しないときは、/etc/conf.modulesから組み込まれるモジュール情報を、/etc/sysconfig/netwaork-scripts/以下のファイルからネットワークインターフェイスの情報を、そして/etc/X11/XF86Configからビデオカードの情報をそれぞれ収集し、/etc/sysconfig/hwconfを作成する。

●主要なコンポーネントもアップデート

Red Hat 6.0から6.1になり、システムの各コンポーネントもリビジョンアップしている。kernelは2.2.5から2.2.12に、glibcは2.1.1から2.1.2にそれぞれ変更された。また、Xのデスクトップ環境を支えるGNOMEは1.0.4から1.0.39に、KDEは1.1.1から1.1.2になり、安定度が増したようだ。XFree86も、3.3.3.1から最新版の3.3.5に入れ替えられており、より多くのビデオカードに対応した。

●市販パッケージは3種類

Red Hat Softwareが市販するパッケージは、付属するソフトウェアの量、サポートの内容に応じて、Standard（29.95USドル）、Deluxe（79.95USドル）、Professional（149.95USドル）の3種類だ。

Standardは、バイナリとソースCD-ROMの2枚組み。Deluxeは、デスクトップワークステーション用途に合わせた追加のアプリケーションCD-ROMが2枚含まれる。Professionalには、上記に加えIBMのDB2など、サーバ向けの追加アプリケーションCD-ROMが3枚バンドルされる。

米国とカナダでは、3種類のパッケージともStarOffice personal editionが付属するが、日本を含め、これ以外の国への輸出版には付かないので注意が必要だ。また、米国の法律により、輸出版では強力な暗号が使えない。

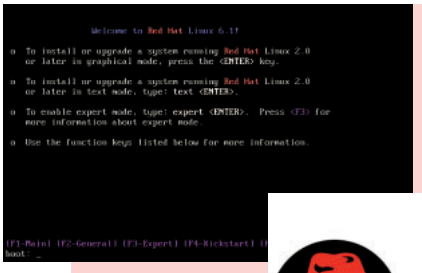
市販パッケージの内容など、詳細については、Red Hat Software社のWebサイトを参照していただきたい。

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

インストーラの起動

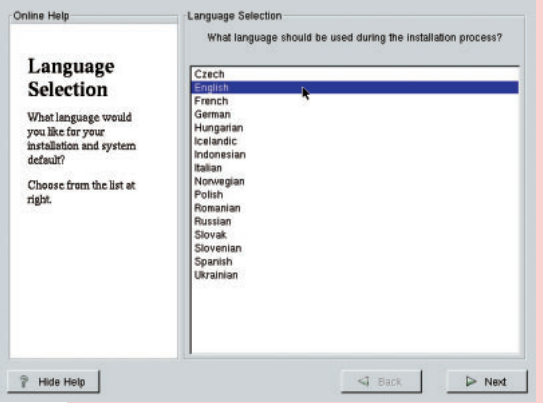
CD-ROMブートが可能であれば、Red Hat Linux 6.1のCD-ROMから起動します。CD-ROMブートができない場合は、WindowsのDOS窓などでブートフロッピーを作成します。手順は次のとおりです。①Red Hat LinuxのCD-ROMをドライブ（X:とする）にセット。②DOS窓を開き、x:［Enter］と入力。③cd \dosutils［Enter］と入力。④rawrite -f ..\images\boot.img -d a［Enter］と入力。⑤「Please insert a formatted diskette into drive A: and press --ENTER-- :」と表示されたら、フォーマットしたフロッピーディスクをAドライブに入れ、Enterキーを押す。

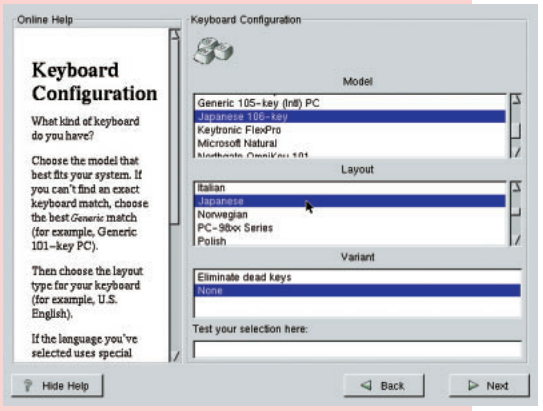
インストーラを起動すると、黒地の画面に「boot:」のプロンプトが表示されますので、通常はEnterキーを押してください。ここでtext［Enter］と入力するとテキストベースのインストーラが起動します。



使用言語の選択

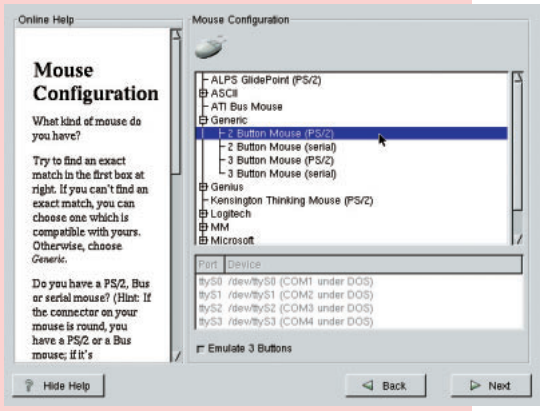
ここでは使用する言語を選択します。日本語はありませんので「English」を選択します。Anacondaでは、ほとんどの画面でヘルプが表示されますので、迷ったりよくわからないときは読んでみることをお勧めします。





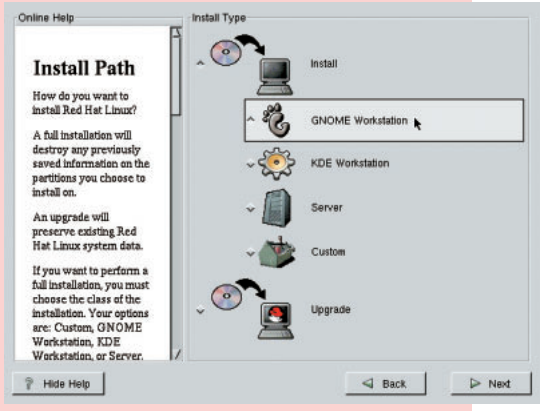
キーボードの設定

キーボードの種類を選択します。106もしくは109タイプのキーボードを使っていればModelに「Japanese 106-key」を、Layoutに「Japanese」を選択します。しかし、インストーラに不具合があり、インストール終了後に設定が反映されません。このため、Linuxを起動後、次のように設定する必要があります。まず、/etc/sysconfig/keyboardというファイルにエディタで「KEYTABLE="jp106"」と記述します。これでコンソールでのキーマップが正しくなります。同様に、X Window Systemでのキーマップは、/etc/X11/XF86ConfigというファイルのSection keyboardでXkbModelに「"jp106"」を、XkbLayoutに「"jp"」をそれぞれ記述します。



マウスの設定

現在使われているマウスのほとんどがPS/2マウスかシリアルマウスでしょう。これらのマウスの場合は「Generic」にある4種類のうちから条件に合ったものを選びます。「2 Button Mouse(PS/2)」「2 Button Mouse(PS/2)」を選択した場合、自動的に3ボタンマウスのエミュレートが有効になります（チェックをはずして無効にすることもできます）。これは、左右のボタンを同時に押すことで、真中のボタンと同等の働きをさせる機能です。

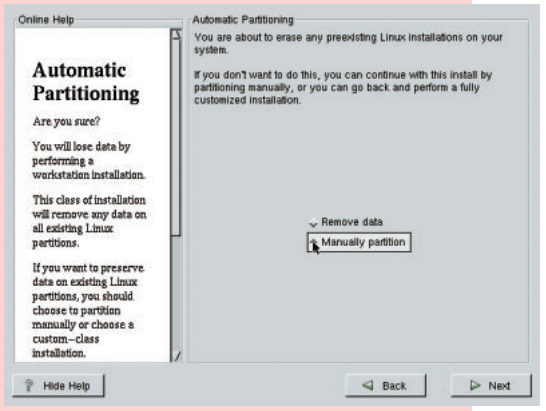


インストールパスの選択

Windowsやほかのディストリビューションとのデュアルブート環境で使うなら「Custom」を選びます。PCをRed Hat 6.1専用マシンにできるなら、デスクトップマシンとして利用するための「GNOME Workstation」「KDE Workstation」、サーバとして利用するための「Server」を選ぶと、パーティション設定やパッケージ選択をインストーラに任せることができます。ただし、ハードディスク内の既存のパーティションを消去することがあるので注意が必要です。特に「Server」はハードディスクのデータすべてを消去するので気をつけてください。

ここでは、「Custom」を選択したと想定して説明を続けます。

「Upgrade」は、Red Hat 6.0などに上書きアップグレードする場合に選択します。



自動パーティション設定（インストールパス Custom以外のときのみ）

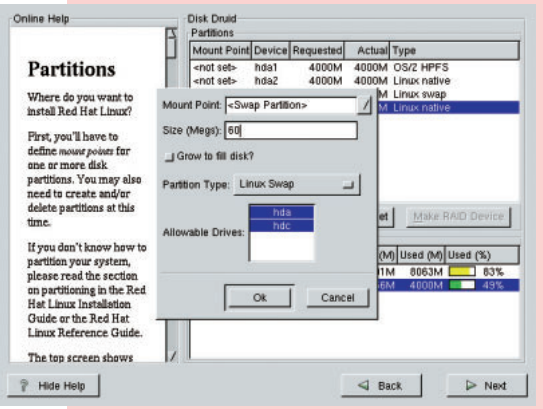
この画面はインストールパスで「Custom」以外のインストールタイプを選択したときに現れます。「Remove data」を選択すると、ディスクのパーティション構成、データをすべて削除したあと、自動でパーティション設定をするので、Red Hat 6.1専用マシンにするときのみ選択します。

ここでは「Manually partition」を選択し、自分でパーティションを設定すると想定して説明を続けます。

パーティションの設定

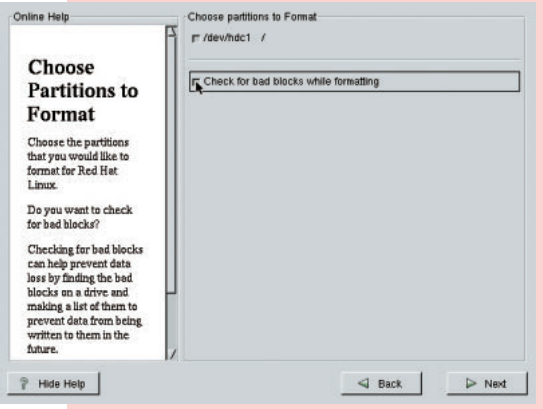
手動でパーティションを作成します。Red Hat 6.1をインストールするには、最低限でもLinuxシステム用とスワップ用の2つのパーティションが必要です。Linuxシステム用のパーティションを作るには、「Add...」ボタンを押し、小さなウィンドウが現れたらMount Pointを「/」とし、「Size」をMバイト単位で入力します。すべてのパッケージをインストールする場合は2Gバイト程度用意すれば安心です。Partition Typeは「Linux native」を選択します。

スワップパーティションは、Partition Typeを「Linux Swap」にし、64から128Mバイトの範囲でメインメモリの1～2倍程度を割り当てます。



ハードディスクのフォーマット

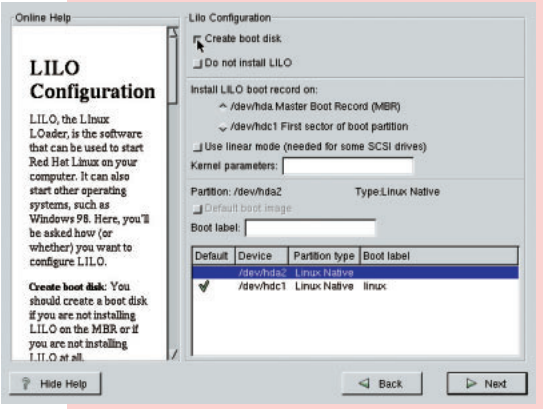
ここではLinuxをインストールするパーティションが選択され、（この画面だと/dev/hdc1です）チェックが入ったパーティションはフォーマットされます。パーティションのデータはすべて削除されますので、もう1度よく確認してください。「Check for bad blocks while formatting」にチェックを入れると、フォーマット中にハードディスクに不良ブロックがないかどうか確認します（特に理由がなければ、この欄にもチェックを入れておきましょう）。



LILO（LInux LOader）の設定（インストールパス Customのみ）

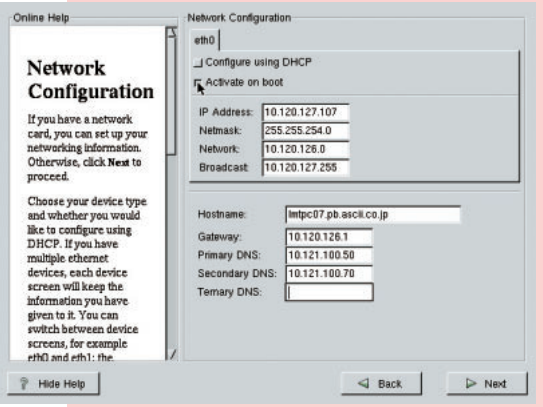
Linuxを起動するローダの設定です。「Create boot disk」をチェックすると、インストールの最後にブートフロッピーを作成します。できるだけ作成しておきましょう。「Do not install LILO」をチェックするとLILOをインストールせず、Linuxはブートフロッピーからのみ起動できるようになります。

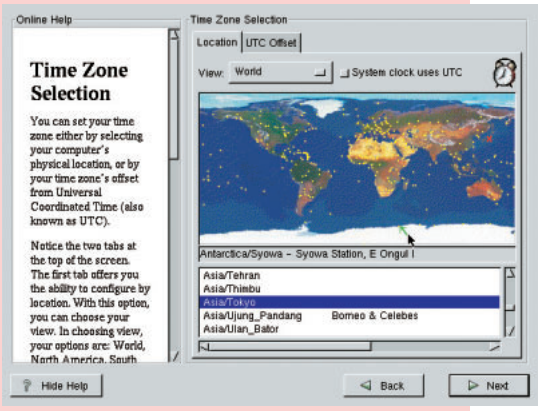
Install LILO boot record on:でLILOのインストール場所を指定します。システムコマンダーなどのブートセレクタを使っている場合は「First sector of boot parition」を選択します。Red Hatのみ、あるいはWindows 9xとのデュアルブートをする場合はMBRにインストールします。画面下半分でLILOに登録するOSを設定します。Windowsなどのパーティションを選択し、ラベルをつけると、利用するOSを選んで起動できるようになります。



ネットワークの設定

インストーラがネットワークカードを認識した場合のみこの画面になります。上段の「Configure using DHCP」をチェックすると、マシンのIPアドレスなどをDHCPサーバから取得します。固定IPアドレスを設定したい場合は、各項目へIPアドレスなどの情報を設定します。「Activate on boot」にチェックすると、起動時にネットワークカードを有効にします。特に理由がなければ、チェックしておきます。

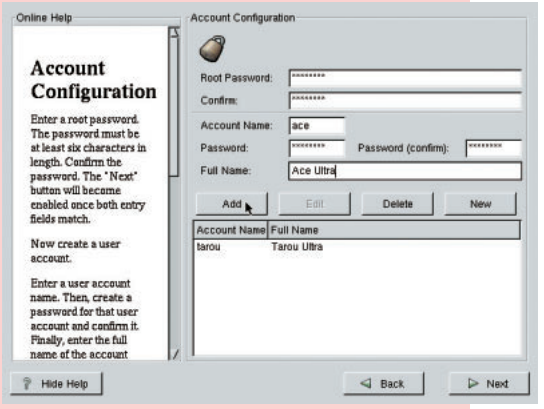




タイムゾーンの設定

マシンが設置されている場所と協定世界標準時（UTC）の時差を設定します。グラフィカルインストーラが大きな特徴となっているRed Hat 6.1では、世界地図上で設置場所をマウスで選ぶだけでタイムゾーンが設定されます。日本国内で使うなら東京を指定します。Viewで地域別の拡大表示を選ぶこともできます。

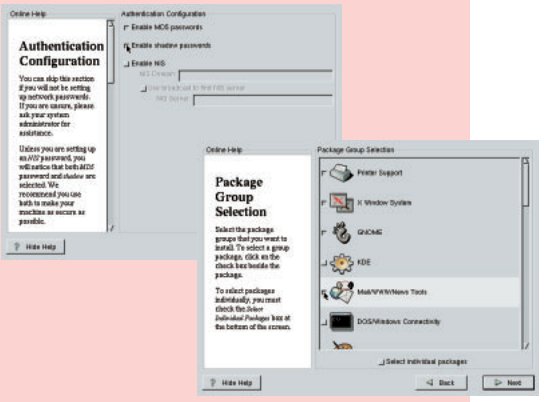
余談ですが、南極大陸の昭和基地も選べるのには驚きました。Linuxユーザーはいるのでしょうか。



ユーザーの登録

ユーザー登録とパスワードの設定を行います。rootユーザーのパスワードを「Root Password」と「Confirm」に入力して設定します。2度入力するのは間違ったパスワードを登録しないようにするためです。

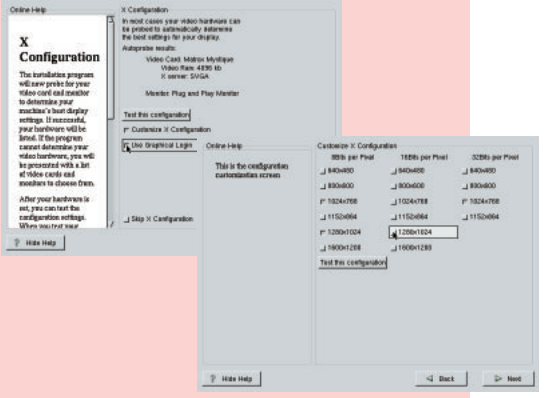
必要に応じて一般ユーザーの登録も行います。この時、ユーザー名が数字のみだと登録できません。



ユーザー認証設定とパッケージ選択（インストールパス Customのみ）

マシンへログインする際に行うユーザー認証の方法を設定します。NISはネットワーク上にあるNISサーバで認証を行う場合にチェックします。NISが導入されているかわからないときは、ネットワーク管理者へ問い合わせてみてください。NISを利用しない場合は変更せず次へ進みましょう。

パッケージ選択では用途に合わせ必要なグループを選びます。どのグループを選択してよいかわからないなら、1番下の「Everything」を選択してすべてのパッケージを導入してみてもよいでしょう。そのままではディスク資源やメモリを消費しますので、Linuxに慣れ、必要なパッケージがわかるようになったら再度挑戦しましょう。



Xの設定

インストーラがビデオカードの自動検出を行った結果がAutoprobe results:の下に表示されます。ここで正しく認識していれば問題ありませんが、そうでないときはとりあえず「Skip X Configuration」をチェックして次に進み、Linuxのインストール後に再起動し、XF86Setupなどを使って設定します。

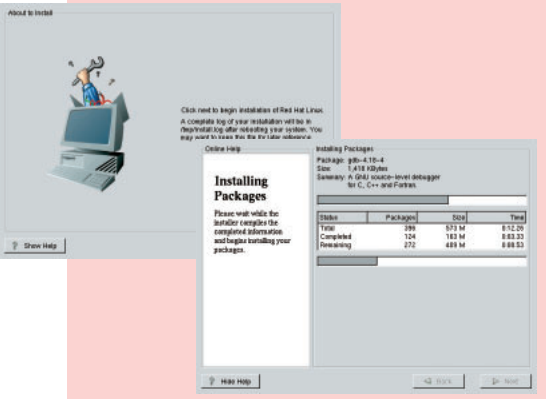
「Customize X Configuration」にチェックを入れると、次のステップで、表示する色数、解像度が選択できます。また、「Use Graphical Login」をチェックしておくと、システムの起動時にXが実行され、グラフィカルなログイン画面になります。グラフィカルログインを利用するかどうかは、好みの問題ですが、マシンパワーなども勘案して決めましょう。

15

インストール開始

インストールの設定は終了です。あとはここで「Next」ボタンを押し、インストールが終了するまでじっと待つだけです。インストールの作業内容は/tmp/install.logに出力されますので、あとから覗いてみてもいいでしょう。

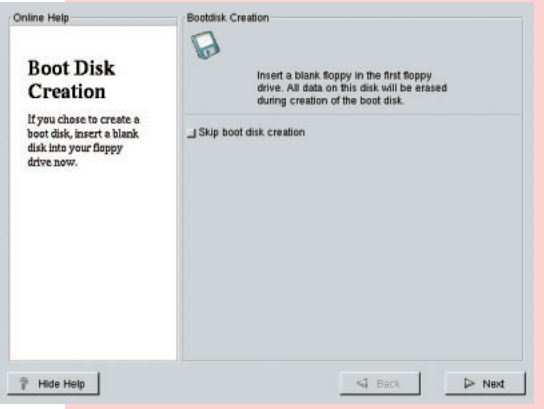
パッケージインストールの進行状況は、棒グラフで表示されます。フルパッケージをインストールすると少々時間がかかります。



16

ブートフロッピーの作成（インストールパス Customのみ）

最後に、ブートフロッピーを作成します。スキップすることも可能ですが、万一に備え、ブートフロッピーを作成しましょう。システム設定を誤り、ハードディスクからLinuxが起動できないということも起こりうるのです。ブートフロッピーを作成するには、フロッピーディスクドライブに空のフロッピーディスクを入れ、「Next」を押してください。



17

インストール終了

これでインストールは終了です。CD-ROMやフロッピーディスクをドライブから抜いてから「Exit」を押すと、PCが再起動します。

Linuxがうまく起動するでしょうか？

18

ログイン画面

Xの設定で「Use Graphical Login」をチェックしていれば、Linuxを起動するとこのようなログイン画面になります。ログイン名とパスワードを入力し、ログインしてみましょう。

グラフィカルログインをやめて、コンソールからログインしたくなったら、/etc/inittabというファイルをエディタで開き、id:5:initdefaultと書いてある行の5を3に書き換えれば、Linux再起動後、コンソールで立ち上がります。グラフィカルログインに戻すには、反対に3を5に書き換えます。

Linux on ノートPC

Linuxを持ち運ぼう！ 必ずできるノートPCへのLinuxインストール

Photo : Shuichi Mito (Dee)

日本でのノートPCの販売台数は、デスクトップPCのそれを超えている。すでにノートPCはパソコンの主流の座を占めているとも言えるのだ。当然、ノートPCをLinuxで使いたいという要求も出てくる。しかし、ノートPCはデスクトップPCとはその構成が違い、すんなりインストールできないことも多い。そこで、今回はノートPCへのインストール法からモバイル環境でのLinux活用法までを解説しよう。
さあ、モバイルLinuxへ一直線！

Section 1

インストールの前に

LinuxノートPCのメリット、デメリットとインストールの基礎知識。最初が肝腎。インストール方法のベストチョイスはコレだ!!

文：塩田紳二
Text : Shinji Shioda

近年のノートPCの小型化と性能向上には目を見張るものがある。ノートPCの発展とLinuxの開発が進んだことにより、数年前にはワークステーションでしか実現できなかったコンピュータ／ネットワーク環境を持ち運ぶことが手軽に実現できるようになったのだ。しかし、ノートPCにLinuxをインストールして使うためには数々のノウハウが必要となる。Section 1では、ノートPCでLinuxを使うメリット、インストールするために知っておきたい基礎知識について説明する。

ノートPCでLinuxを利用するメリット

Linuxをノートで利用するメリットとは何だろうか？　ひとつには、ノートマシンが持つ、一般的な性質である、

- バッテリ駆動でどこでも使える
- 持ち運びが可能
- すべての環境を持ち歩ける

といった部分があるが、これは、どのOSを使っても同じだ。Linuxというか、UNIX系のOSは、もともとがミニコンあたりから始まっているため、必ずしもノートPCというハードウェアに最適ではないが、逆に、動いてしまえば、その機能は、完全に使えるわけで、その意味では、どこでも、フル機能が利用できることになる。かつて、UNIXワークステーション全盛の時代に、数社がワークステーションをラップトップマシン化したが、それは、今でいえば、液晶デスクトップ並のサイズであり、かろうじて持ち歩けるものにすぎなかった。もっとも、当時は、PCでも、ラップトップサイズが普通で、ノート型は、非常に簡易なものであり、ハードディスクなどがノートサイズマシンでも利用できるようになったのは最近の話だ。

現状のLinuxがノートPC向けにサポートしている機能とは、

- PCカードのサポート
- サスペンド／ハイバネーション
- APM対応

などだ。PCカードのサポートは、PCMCIAサポートとローダブルモジュールによって実現され、カーネルがAPM対応になったことで、電力管理やソフトによる電源オフなどが実現されている。また、ハイバネーションやサスペンドは、BIOSの機能をそのまま使うが、APM対応カーネルにより、内部クロックのレジューム時の更新などが行われる。

このほか、APMを利用したバッテリ状態表示のためのプログラムなど、いくつかのノートマシン用ツールも存在する。

ネットワークのメリット

Linuxなどでは、ネットワーク機能が充実しており、ノートマシンと組み合わせることで、ふだん利用している環境に限りなく近い環境でどこでも作業を行うことができる。また、X Window Systemは、ほかマシンで動作しているXクライアントのウィンドウを、手もとのマシンに表示させることができる。たとえば、社内ネットワークのような環境なら、ふだん使っているデスクトップマシンそのままの環境を使い続けることもできるし、必要なディレクトリをNFSでマウントし、ファイルを利用することもできる。

逆に、ノートPCからWebやFTPサーバといったサービスも提供でき、アドレス解決が可能なら、ノートマシンへ外部からアクセスすることさえ可能になる。お遊び的には、出先で、Webサーバを運営して、リアルタイムに状況を報告するなんて使い方もできるわけだ（なんだか、アマチュア無線で移動運用のために無人島に行くような話だが……）。

OSとしての軽さ

これはデスクトップマシンでもいわれることだが、Windows 95／98などと比べると、Linuxは要求リソースが小さく、かつ、動作効率が良いので、必ずしも、Pentium IIなどの最新CPUを使わないマシンでも十分利用可能だ。これは、ノート機においても成り立つ

ことで、特にPentium以前のノートマシンは、ディスク容量などからいっても、Windows 95／98などの利用には向かない環境になりつつある。そんな環境でも利用が可能なLinuxでは、使われなくなったマシンや、安価な中古品の購入など、比較的容易にマシンが入手可能だ。

筆者の使った印象では、486DX4 75MHzのノートマシンで、X Window Systemを動かし、VJEでかな漢変換を行うと、速度的に少し不足を感じた程度だ。仕事柄、キー入力は速いほうだと思っているが、変換結果が出るのに一呼吸ある感じで、同音異義語の選択時に待たされるのに違和感を覚える程度だ。もっとも、Windows 95で使っていたときには、もっと反応は遅く、限界ギリギリという感じだった。それも、Windowsでは、書き込みが行われる学習辞書をRAMディスクに置くなどしており、かなり「対策」を施したうえでのことだ。

Linuxのプログラムは必ずしもGUIのものだけではないので、この程度のマシンでも十分使いでがある。キーボードからすべて操作できることは、デスクトップマシンに比べポインティングデバイスに制限のあるノートマシンでは大きなメリットでもある。

デメリットもないわけではない

LinuxをノートPCで使う上のデメリットの1つは、ノートPCを前提としたディストリビューションなり、設定ツールがなく、自分でチューニングしなければならないことが挙げられる。たとえば、各種Daemonの設定をどうするか？　あるいは、Disk Syncをどうするかなどは、自ら設定しなければならない。

もう1つは、最近の周辺装置や本体ハードウェア、ならびに付属の装置などのデバイスドライバがWindows用しか用意されていないことだ。たとえば、PCカード接続のフロッピーディスクドライブ、組み込みのデジタルカメラなど、独自のデバイスが組み込まれているノートマシンは多いが、これらのドライバは、Windows用のみというものがほとんどだ。また、サードパーティーの販売するPCカードもちょっと特殊なものは、ドライバはないと思ったほうがよい。

こうした制限はあるものの、どこでもLinuxが使えるというのは、ある意味楽しいことでもある。Linux自体が、大量のドキュメントを含んだシステムでもあり、電車の中でドキュメントを読みつつ、システムの理解を深めるなんてこともLinuxならではといえる。

ここでは、全般的にノートPCにLinuxをインストールすることについて解説を行うことにする。

Linuxのインストールについて

ノートPCのインストールに関して説明する前に、一般的なLinuxのインストールについて簡単に説明しておこう。Linuxのインストールは、ディストリビューションによっても違うが、基本的には、インストーラのブート方法と、

Column

LinuxのUSBデバイスサポート

LinuxのUSBサポートは、バージョン2.2.7からカーネル本体のソースに含まれるようになった。しかしながら現時点ではUSBサポートのコードは開発途上であり、一般ユーザーが使えるレベルではない。

2.2系のカーネルにはUSB接続のマウスとキーボードをサポートするためのコードが含まれているものの、カーネルのコンパイル時のオプションではデフォルトでコメントアウトされている。ただし、自分でカーネルをコンパイルしてUSBマウスを実際に使っているユーザーをインターネット上で何人か見かけたので、うまくいけば動作するものと思われる。開発版の2.3系のカーネルには、USB接続のデジカメやプリンタ、ZIPドライブなどのストレージデバイスをサポートするためのコードなどが追加されているものの、メーリングリストのコメントを見る限りでは安定度に関してはまだまだのようだ。

今回、試しにLibretto ffのUSBデバイスが使用できるか試してみたところ、カーネル2.3.21付属のUSBドライバではUSBフロッピーディスクドライブは認識されるものの、エラーが表示され使用することはできなかった。メーリングリストでは頻繁にパッチがアップロードされており、それらのパッチを適用すればうまくいく可能性はあるだろう。メーリングリストによれば、SONY VAIOのUSBフロッピーディスクドライブはRead-Onlyで動作したとあるが、Libretto ffとはUSBコントローラの種類が異なるためVAIOではうまく動作しているのかもしれない。

LinuxのUSBサポートに関する情報は、Webページ「Linux USB」(http://www.linux-usb.org/)を参照してほしい。

（細川茂一）

OSイメージの読み込みメディア（インストールメディア）を何にするのかの組み合わせとなる（**表1-1**）。

ブート方法は、

- CD-ROM
- フロッピーディスク
- DOS

の3つがあり、インストールメディアとしては、

- CD-ROM
- ハードディスク
- ネットワーク

の3つがありえる。また、ネットワーク経由のインストールでは、FTP、NFSなどが標準的だが、マイクロソフトのネットワークプロトコル（SMBプロトコル。LinuxではSAMBAで使っているWindowsマシン向けプロトコルといったほうがわかりやすいか）やHTTPが利用できることもある。

最近のディストリビューションでは、CD-ROMまたは、フロッピーディスクからブートし、CD-ROMからインストールするのが標準的だ。前者は、CD-ROMドライブ内蔵のノートPCではおそらく普通に利用できるはずだ。後者は、CD-ROMブートが不可能な機種や、ブートに対応していないサードパーティー製CD-ROMドライブとの組み合わせで可能なインストール方法だ。

DOSからのブートは、LOADLIN.EXEというプログラムとの組み合わせで、Linuxカーネルとイメージファイルを使って行う起動方法だ。具体的には、メモリ上にLinuxのファイルシステム（つまりRAMディスク）を作り、そこにイメージファイルを展開して、Linuxカーネルを起動する。原理的にはDOSが起動すればいいのだが、フロッピーディスクを使わない場合のイメージファイルは、1.44Mバイトよりも大きくなるため、通常は、ハードディスク上にDOSパーティションを作成して起動する。

現実的には、各ディストリビューションが全ての組み合わせをサポートしているわけではなく、前記の組み合わせの一部や、ブート方法、インストールメディアの一部がサポートされていないこともある。

つまり、ノートPCへのLinuxのインストールでは、ノートマシンのハードウェア構成（CD-ROMドライブからのブートの可／不可、フロッピーディスクドライブの仕様）を把握し、さらにそれらで組み合わせ可能なディストリビューションを選択する必要がある。あるいは逆に、インストールしたいディストリビューションに合わせて、CD-ROMドライブなどを入手する必要があるということだ。

なお、Linuxのインストーラは、通常、インストール用に作られたLinuxカーネルが起動し、その上で動作することになる。このため、CD-ROMや2枚目以降のフロッピーディスクは、Linuxが読み出しを行う。このため、DOSやWindowsで正常に使えるデバイスでも、Linux側の対応により、読み込みが行えない場合がある。たとえば、特殊なインターフェイスのCD-ROMドライブや3モードフロッピーディスクドライブの一部などがこれに相当する。また、フロッピーディスクによるブートでは、ディスク容量の関係から、PCMCIA関連のモジュールは、ブートフロッピーとは別のフロッピーディスクから読み込むのが普通だ。このため、前記のフロッピーディスクドライブによる問題が生じる可能性があるので注意が必要だ。

ネットワークからのインストールでは、最低でももう1台CD-ROMドライブを持つマシンが必要となる。これはデスクトップマシンでかまわないが、FTPやNFSサーバプログラムを用意する必要がある。もっとも簡単なのはLinuxが動いているデスクトップマシンを利用する方法だが、WindowsマシンでもFTPサーバプログラムを用意することで、対応が可能になる。

ブートデバイス ＼ インストールメディア	CD-ROM	ハードディスク	ネットワーク
CD-ROMドライブ	◎ ブート可能ならそのままインストール可能	× CD-ROMが使えるのならわざわざすることもない	× 時間がかかる。ほかのマシンが必要
フロッピーディスクドライブ	◎ CD-ROMからブートできない場合	○ インストールイメージの入る大きさのDOSパーティションが必要	△ CD-ROMドライブが使えず、HDDにも余裕がないときのみ
DOS／Windowsパーティション	△ CD-ROMがPCカード接続で、フロッピーディスクがPCMCIAサポートのディスクを読み込めない場合	○ PCMCIAフロッピーが読み込めず、ハードディスクに余裕があるなら	○ ほかに方法がない場合

表1-1　Linuxをインストールする際のブートデバイスとインストールメディアの組み合わせ

■ノートPCのハードウェアを確認する

デスクトップPCでは標準的なハードウェアも、ノートPCでは特殊な方法で接続されているなど、細かな点で違いがある。たとえば、PCカードを介した外付け型フロッピーディスクドライブは、標準的な接続方法と違ってくるため、そのままでは、Linuxのインストールに利用できないこともある。

また、ハードウェアを確認するもう1つの理由は、インストール後の設定でそのハードウェアの仕様がわかっている必要があるためでもある。たとえば、X Window Systemの場合には、使用されているグラフィックチップ名が必要になるが、中を開けてボードを見れば、おおよそ見当が付くデスクトップマシンと違い、ノートマシンでは、カタログやメーカーのWebサイトなどで、使用デバイスを調査しなければならない（もちろん、自信があるなら分解してチップを探すことは止めないが……）。そのほか、シリアルポートの設定（モデムやIrDAへ割当てがどうなっているかなど）もあらかじめ調べておかねばならない。

■インストール用のデバイスについて

最初のポイントは、CD-ROMドライブの有無と、接続方法、起動設定だ。CD-ROMドライブは内蔵か外付けか、また、CD-ROMからのブートは可能かどうかだ。ブートが可能かどうかを判定する方法の1つは、BIOSの設定画面で、起動デバイスの指定にCD-ROMが含まれているかどうかを見るという方法がある。ただし、これはあくまでもBIOSの設定で、サードパーティー製の外付けCD-ROMドライブを使っている場合、ドライバの関係で、ブートができないものもあるため、純正品以外の外付けドライブを使っている場合には注意が必要だ。

CD-ROMがない、もしくは、CD-ROMからのブートが不可能な場合には、フロッピーディスクを使ってインストーラを起動する必要がある。このとき、フロッピーディスクドライブが外付けの場合、PCカード経由で接続するものは、インストーラの起動ができても、PCMCIAサポート用フロッピーが読めない可能性がある。

もう1つのフロッピーディスクのチェックポイントは、3モードドライブかどうかだ。一部の3モードドライブに関しては、BIOSでのサポートの違いから、インストール用のLinuxカーネルで扱えない場合があり、これにより、PCMCIAサポートディスクを読み込むことができないことがある。このため、PCカード経由のネットワークインストールやCD-ROMからの読み込みができない可能性がある。

もし、ネットワーク経由でのインストールを想定している場合（CD-ROMドライブがない場合）には、利用できるネットワークカードのメーカーと製品名、できるならば、LANコントローラチップの名前などを調べておく必要がある。インストール時に利用できるPCカードの種類は限られており、各ディストリビューションで、使用できるすべてのPCカード型ネットワークカードが利用できるわけではないからだ。

このほかには、ハードディスクドライブの容量やパーティションの状況（DOSパーティションの有無とそれ以外のパーティションの状態）などを調べておく必要があるだろう（調査する時点では、Linux用にパーティションを確保する必要はない。実際の作業は、調査がすみ、インストール方法やディストリビューションが決定した時点で行う。先にあわててDOSやWindowsを消すとあとで困ることになるので手を着けないでおく）。

■Linuxで扱うデバイスについて

インストール後に、システムの設定を行う場合、必要になる情報としては、

- ディスプレイ関連
  - コントローラ（グラフィックチップ）のメーカー／製品名
  - ビデオメモリ容量
- ポインティングデバイス
  - ボタンの数

Column

LinuxのIrDAサポート

IrDAサポートに関しては、2.2系カーネルから含まれるようになり、シリアルポートのエミュレーションを行うプロトコルであるIrCOMMは安定して動作しているようだ。IrCOMMを使うと、IrDA対応のISDN公衆電話やIrDAポートが装備されている携帯電話とやり取りができるようになる。また、IrDAのプロトコルの一つであるIrOBEXを使うと、PalmIIIとデータ交換ができるようになるとドキュメントに記述がある。詳しくは「The Linux IrDA Project」（http://www.cs.uit.no/linux-irda/）をチェックのこと。（細川茂一）

LinuxでIrDAによる赤外線通信を使えるようにする「The Linux IrDA Project」のWebページ

接続方法（ノートPCの場合PS/2接続がほとんど）

- **内蔵モデム**
  使用シリアルポート（あるいはBIOSで割当てが行われているかどうか）
- **IrDAポート**
  使用シリアルポート（あるいはBIOSで割当てが行われているかどうか）
- **シリアル、パラレルポート**
  割り込み番号と使用I/Oアドレス
- **サウンドデバイス**
  サウンドチップのメーカー／製品名
  Sound Blaster互換かどうか
  使用割り込み番号とI/Oアドレス
  （MIDIポートやゲームポートを含む）
  MIDIポートやゲームポートのBIOSによる設定状況

などがある。これらの情報を調べる場合、Windows 95／98が動いているなら、コントロールパネルの［システム］にある［デバイスマネージャ］を使うとよい（**画面1-1**）。ここでは、情報の印刷も可能なので、すべて印刷しておくと、あとの設定が楽になる。

インストール方法を考える

ハードウェアの調査が終わったら、最初に試すべきインストール方法を決定する。CD-ROMドライブが内蔵されている機種では、ほとんど問題がないが、そうでない場合には、可能なインストール方法を検討したのち、自分の使いたいディストリビューションがサポートしているインストール方法や、デバイスを調べる必要がある。

画面1-1　Windowsのデバイスマネージャでハードウェアの情報を収集できる

インストール方法決定の手順を**図1-1**にフローチャートとしてまとめた。

なお、ネットワークからインストールする場合は、CD-ROMドライブを持ち、ネットワークに接続されたマシンが必要だ。このほかの方法としては、ノートPCのハードディスクをデスクトップマシンに取り付けてインストールする方法がある。ハードディスクがはずせない、あるいははずす自信がないならCD-ROMドライブを購入しよう。

さらには、自分で、インストーラをカスタマイズするという方法もあるが、基本的にはかなりLinuxに精通していなければ難しいと思われる。多くのディストリビューションでは、インストーラのソースを公開しており、それらを入手（インストールCDに入っていることが多い）し、稼働しているLinuxマシンなどを使って、イメージファイルを作成する。もちろん、インストーラそのものを改造することも可能だ。

さて、可能なインストール方法がわかったら、次は、ディストリビューションを選択する。このとき、ディストリビューションによっては選択したインストール方法に対応していないこともあり得るので注意が必要だ。

また、インストーラでサポートされているPCカードを持っているかどうかという問題も発生する。これに対し、ディストリビューションによる特徴もまたあり、自分の好みや利用する環境を考えると、特定のディストリビューションを選択せざるを得ない場合もありえる。

インストール前の準備

ここで簡単にインストール前に必要なことについて解説しておこう。

多くのノートPCでは、すでにDOSやWindowsがハードディスクドライブすべてを使っている状態になっていると思われる。もし、WindowsとLinuxを共存させたいなら、何らかの方法で、Linuxをインストールするための領域を確保する必要がある。もっとも確実なのは、パーティションコマンダーやPartitionMagicなどの市販のパーティション変更ソフトやLinuxに付属するFIPS.EXEを使って、Linux用のパーティション（通常領域とスワップ領域の最低2つ）を作り出す方法だ。このときには、少なくとも、必要容量以上の空きがなければならないので、Windows上で不要なファイルを削除するなどして、空き領域を作成する。このとき、移動不可能なスワップ領域などが、ディスクの末尾付近にあると、うまくパーティション作成が行えない場合があるので、その前にデフラグを行う（この前に不要なファイルはすべて削除しておく）。このとき、コントロールパネルの［システム］―［パフォーマンス］タブにある仮想メモリボタンで、「仮想メモリを使わない」を選択し再起動する。この状態でデフラグをおこなえば、ディスクの後半にファイルが残る可能性はかなり低くなる。デフラグ終了後、前記手順を使って、仮想記憶の設定を元に戻しておく。

逆にまっさらのハードディスクドライブが用意できる場合や、Windowsはもういらないという場合、完全にフォーマットし、最初にDOSパーティションを作っておく。これは、万一の場合の備えであるとともに、ノートPCがハ

イバネーション領域を確保するのに必要になるからだ（ただし、ハイバネーション機能のないノートPCもあるが……）。なんらかの問題で、CD-ROMからのインストールが不可能になった場合、DOSが起動する領域があれば、そこからインストーラを起動することもできるからだ。ハードディスクドライブに余裕がない場合には、数十Mバイト程度、さらに余裕がなければ、数Mバイト程度残しておく。

DOSを持っていないユーザーは、Windows 95の緊急用ディスクを使う（Windows 98の緊急起動ディスクはDOSではないので注意）、FreeDOSを使う（http://www.freedos.org/）、日本アイ・ビー・エムのPC-DOSを購入するなどして、DOSが起動するパーティションを作っておく。このとき、Windows 95 OSR2以降でサポートされているVFAT32形式のパーティションは、Linuxのインストールで問題が発生する場合があるので、使わないことが望ましい。

また、インストールに利用する外付けCD-ROMやネットワークカードなどは、Windowsが動いているなら、動作チェックをすませておく。これらがまともに動作していることが確認できないと、インストール時のエラーで、何が悪いのかがわからなくなってしまう。ネットワークでインストールできないと思ったら、実はケーブルが切れていたなんてことで、よけいな時間を食ってしまうことにもなりかねない。

このほか、ちょっとしたDOS用のツールなどを用意しておくと便利だ。たとえば、DOS用のバイナリファイルの分割、結合ツールなどである。フロッピーディスクを必要としない、インストール用イメージファイルは、多数のモジュールを含んでいるため、1枚のフロッピーディスクに入らない。このため、ネットワーク経由でのファイルコピーが行えない環境では、インストールイメージをCD-ROMから取り出し、こうしたツールにより、複数にわけてフロッピーディスクで転送する。また、デスクトップマシンでPCカードの読み書きができるのなら、フラッシュメモリカードなどを使うという方法もある。

また、すでにLinuxが動いているデスクトップマシンを所有しているのなら問題はないが、Windowsマシンしか

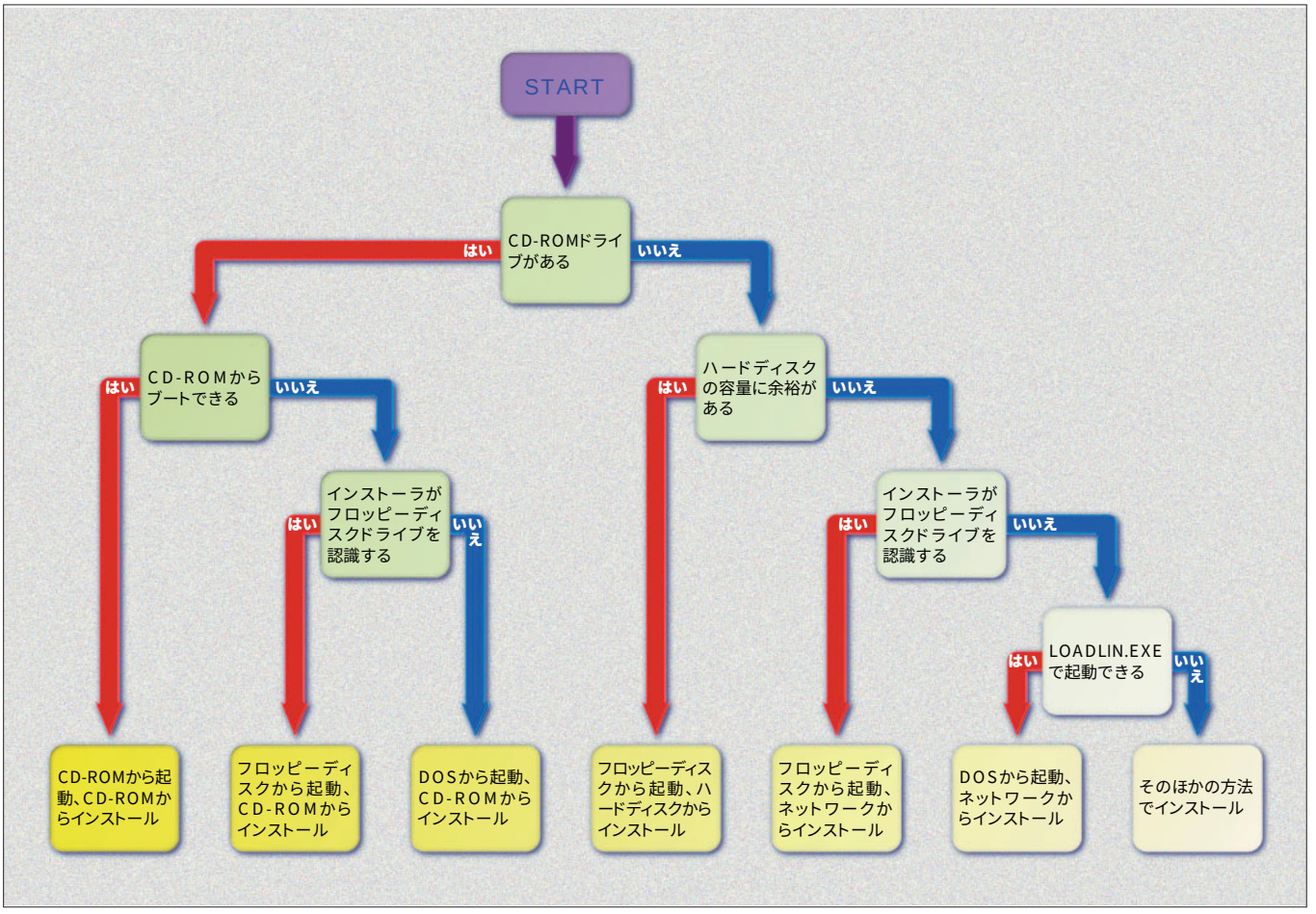


図1-1　このフローチャートで、自分の環境に合ったインストール方法ををを見つけよう

ない場合、gzやtarなどを開くことができるWindows用のアーカイブツール（WinZipなど）を入手しておくと、インストールディスクをいろいろと調べる場合に便利だ。

ディストリビューションによっては、DOSやWindowsで読むことができる形式のテキストファイル（改行がCR+LFのもの）がなかったり（UNIX標準のLFのみのテキストファイル）や、EUCやJISコードのドキュメントのみでシフトJISコードのオンラインドキュメントがない場合がある。こうしたファイルは、Webブラウザで開くとよい。

ここでは、インストールという面から見て、主要なディストリビューションの選択を考える。ここでは、各ディストリビューションの特徴面からの選択については、考慮しない。各自、カタログなどを参考にして決定していただきたい。

とりあえず、ここでは、以下のディストリビューションについて、インストール機能を調べてみた。

- TurboLinux 日本語版 4.0
- LASER5 Linux 6.0
- Vine Linux 1.1CR

なお、調査は、各ディストリビューションの市販パッケージを入手して行った。インストールは、マニュアルに従って行ったので、各ディストリビューションで、前記すべてのインストール方法を試した結果ではない。ただし、まったく何もしてないのではなく、調査として十分な数のノートPCを用意して実際のインストールの確認ができなかったということである。「最悪」条件の例として、筆者の手持ちのIBM ThinkPad 701CsでCD-ROMなしのインストールができるかどうかは実際に試してみた。しかし、この結果は、ほかのマシンに拡張可能なものではないため、あくまでも参考にしかならないものだ（とはいえ、古いマシンにインストールしようとしているユーザーに多少のなりとも情報提供はできるだろうが……）。

こうした状況であるため、ここで紹介する各ディストリビューションのインストール方法は、その完全さを保証するものでもないし、各環境でのインストールを保証するものでもない。筆者は、過去に、あるディストリビューションをローカルネットワーク内でFTPインストールしたが、FTPでのエラーにより、インストールができなかったことがあった。サーバ側（このときにはサーバにFreeBSD 2.6を利用）には特に問題があった経験はないため、おそらくはインストーラに問題があると考えられる。実際にこのようなことがあったので、インストーラが起動できても、かならずしもインストールが完了できるとは限らないことも考慮しておいたほうがよい。

なお、供給元以外によるパッチやマニュアルに記載のないインストール方法などは、今回検討の対象に含めていない。また、本誌出版までに、供給元からパッチやアップデートが出る可能性もある。

TurboLinux 日本語版 4.0

TurboLinux 4.0（ターボリナックスジャパン）のインストーラでは、以下のブート方法がサポートされる。

- CD-ROM
- フロッピーディスク
- DOS

また、インストール元メディアとしては、

- CD-ROM
- ハードディスク
- FTPサーバ
- NFSサーバ
- SMBサーバ（Windowsのファイル共有）

がサポートされる。

インストール用フロッピーには、

- インストールディスク
- 追加モジュールディスク
- パラレルモジュールディスク
- 補助ディスク

の4種類があり、「追加モジュールディ

Column

Winmodemへの対応

DSPを使いソフトウェアによりモデム機能を実現するいわゆる「Winmodem」は、Libretto ffを始め最近のノート型の内蔵モデムに採用されている。Winmodemはソフトウェアでモデム機能が実現されているので、現時点ではWindows上でしか内蔵モデムが使用できないのが実状だ。しかしながら、最近Linux上でこのWinmodemを使えるようにドライバを開発しようという声がLinuxユーザの間で高まり、「linmodem」というプロジェクトが立ち上がっている。興味のある方はhttp://www.linmodems.org/をチェックしてみてほしい。

（細川茂一）

スク」は、PCMCIAサポートや、一部のSCSIカードなど、インストールディスクでサポートされていないハードウェアを利用する場合に使う。

パラレル接続のIDE CDを使う場合には、「パラレルモジュールディスク」、FTPおよび、ハードディスクからインストールを行う場合には、「補助ディスク」が必要になる。パッケージには、「インストールディスク」のみ付属するので、必要に応じてほかのディスクを作成する必要がある。

サポートされているネットワークカードなどは、製品添付のマニュアルにディスク別に記述がある。マニュアル自体もPDFとしてインターネットから入手可能だ。

LASER5 Linux 6.0

LASER5 Linux 6.0（レーザーファイブ）のインストーラでは、ブート方法として、

- CD-ROM
- フロッピーディスク
- DOS

の3つの方法があり、インストールメディアとしては、

- CD-ROM
- ハードディスク
- FTPサーバ
- HTTPサーバ
- NFSサーバ

が利用できる。

フロッピーディスクからブートする場合、インストールメディアとしてCD-ROMを使うか、ネットワークからインストールするかで、利用するディスクが違う。インストール用フロッピーディスクは、

- CD-ROM／ハードディスク用ブートディスク
- CD-ROM／ハードディスク用拡張ディスク
- ネットワーク用ブートディスク
- ネットワーク用拡張ディスク

の4種となっている（それぞれ、対応するデバイスが違うが、すべてインストーラが起動するディスク）。これとは別にPCMCIAサポート用ディスクがあり、CD-ROMやネットワークカードをPCカードスロットで接続する場合には、このディスクが別途必要になる。なお、DOSからのブートは可能だが、PCMCIAモジュールはイメージファイルに含まれていないので、別途フロッピーディスクを作成して、起動後に読み込ませる必要がある。このため、Linuxできちんと扱えない3モードフロッピーディスクドライブを持つ機種（テスト用のThinkpad 701がこれに該当した）では、実質、DOSから起動する方法でインストールを行うことはできない。

前記、4枚のディスクがサポートするデバイスは、インストールCD（Disc1）の\images\BOOTDISK.SJISファイルに記述があるが、これは、モジュールのファイル名のみなので、ちょっとわかりにくい。パッケージ付属の「インストールガイド」のAppendix Aに、モジュール名とハードウェア名称のリストがある。

Vine Linux 1.1CR

Vine Linuxについては、市販されているパッケージである技術評論社の「Vine Linux 1.1CR」をベースに調査した。この版では、ブート方法として

- CD-ROM
- フロッピーディスク
- DOS

の3つが選択可能だ。またインストールメディアとしては、

- CD-ROM
- ハードディスク
- FTPサーバ
- NFSサーバ
- SMBサーバ

が利用可能だ。インストール用フロッピーディスクは、

- ブートディスク
- 補助ディスク

の2枚があり、PCMCIAを使う場合やFTP、ハードディスクからのインストールでは、補助ディスクを起動後に読み込ませる必要がある。

Vine Linuxは、Red Hat 5.2ベースなので、インストールで対応するデバイスなどは、Red Hatの情報が利用できると思われる（パッケージに含まれるマニュアルやディスクには、対応デバイスリストは見あたらなかった）。

どれにするか？

実際、今回調査した範囲では、インストール方法としては、多少の違いがあるものの、特に大きな差はないと思われる。ただし、ネットワークインストールを行う場合で、2枚目のフロッピーディスクが読み込めないような機種では、TurboLinuxかVine Linuxを使うしかないのが現状かもしれない。

Section 2

Linuxインストールの実際

ノートPCでLinuxとWindows 98とのデュアルブートを実現！
インストーラが自動認識しないハードウェアはどうやって設定する？

非常に多くのメーカーからノートPCが発売されているが、その大きさと用途から、おもに3種類に分けることができるだろう。

・A4サイズオールインワン

14インチ前後の大きめの液晶ディスプレイと、CD-ROM、フロッピードライブ、モデムあるいはLAN機能を標準搭載している。キーボードのピッチも十分にあり、おおむね重さ2kg以上で省スペースのデスクトップPCのように使われる。

・B5サイズノート

10～12インチの液晶ディスプレイ、20～25mmの厚さで1.2kg～1.5kgと軽量コンパクトになっている。キーピッチは少し小さめだが、持ち運べてふつうに使える実用機である。CD-ROMやフロッピードライブは外付けとなっていることが多い。

・ミニノート

7～9インチの液晶ディスプレイは横長のアスペクト比になっている。B5ノートよりふた回りほど小さいが、25～30mmと少し厚めで1kg程度。キーボードが小さいため入力がしやすいとは言えない。CCDカメラを内蔵し、遊び感覚を重視している。

今回は、それぞれの代表として、A4サイズの富士通FMV-BIBLO、B5サイズのSONY VAIO 505、ミニノートの東芝Libretto ff 1100に、Linuxをインストールした。

そして、Windowsの環境はそのまま残し、デュアルブートでLinuxと切り替えて使うことを想定している。

なお、これらのマシンはWindows 98が標準のOSであり、Linuxでの動作は保証していない。Linuxをインストールしたためになにか問題が起こっても、メーカーからのサポートは受けられないことに注意されたい。

また、本記事にもとづくLinuxのインストールに関しては、各自の責任で行っていただきたい。

Column

A4オールインワンノート FMV-6366NU4/L

省スペースでありながらデスクトップ機並みの性能を持つA4サイズオールインワンノートは、ファーストマシンとしても十分使えるため、オフィスでも家庭でも根強い人気がある。ここでは、FMV-BIBLO LIFEBOOK FMV-6366NU4/L（以下FMV-6366）にTurboLinux PRO日本語版4.2をインストールしてみた。

FMV-6366のハードディスク容量は6.4Gバイトで、購入時には、約2Gバイト（FAT16）と約4Gバイト（FAT32）にパーティション分割されている。2GバイトのパーティションにはWindows98がインストールされており、4Gバイトのパーティションは空き領域となっている。今回は時間の都合もあり、パーティションの切り直しをせずに、そのまま4Gバイトのパーティションにインストールすることにした。

「インストール」フロッピーディスクをドライブに挿入して電源を入れる。［PCMCIAサポート］には［いいえ］を選んでおく。これはインストールデバイスとして使う場合であり、今回は内蔵のCD-ROMからインストールするからだ。

インストール後のPCMCIAの設定方法については、VAIOノートのページを参照してほしい（ドライバも同一でOK）。X Window Systemの設定は残念ながらうまくいかない。仮の値を入力するか、スキップしてインストールを終えよう。

別にXが起動しなくてもLinuxなのだが、やはり昨今の風潮ではコンソールでXが起動しないと何か寂しさを感じる。しかし、ノートPC最大の難関がX Window Systemのインストールなのだ。

/etc/X11/XF86Configをいろいろと編集したが、結局正しく表示されなかった。このFMV-6366に搭載されているRAGE Mobilityは、98年12月に発表されたチップで、XFree86のサイトで3.3.5のドキュメントを調べた結果、Mobilityを含むRAGE128系のチップはまだサポートされていないことがわかった。

もし自分のノートPCのビデオチップが、XFree86でサポートされているかどうかを知りたければ、下記サイトにアクセスするといい。

```
http://www.xfree86.org/cardlist.html
```

さらに、ノートPCへのXのインストールで参考になるのが以下のサイトだ。機種ごとにXF86Configをまとめてあり、たいへん参考になる。（竹内充彦）

LCD-NOTE-PC list
http://www.yy.cs.keio.ac.jp/~sanpei/note-list.html

B5ノート SONY バイオノート505シリーズ

文：竹内充彦
Text :Michihiko Takeuchi

洗練されたスタイルと携帯性で人気のB5薄型ノートは、いわゆるモバイラーに人気の機種のひとつだ。ここでは、SONYのVAIOノートN505に、Turbo Linux PRO日本語版4.2をインストールする。N505は、CPUにモバイルCeleron 300MHz、メモリ64Mバイト、ハードディスク6.4Gバイトを搭載し、10.4インチの液晶ディスプレイはXGA（1024×768）表示が可能だ。

ハードディスクのレイアウト

べつにVAIOに限った話ではないが、B5サブノートパソコンの場合、内蔵ハードディスクは1台だ。しかも、通常はWindowsがインストールされている。パソコンを2台以上持っていて、このノートパソコンはLinux専用マシンにしてもOKという豪快な人はともかく、大半の人は、現在使っているWindowsは残したままで、Linuxを追加インストールしたいはずだ。

ここで思いつくのが、ハードディスクのパーティション分割である。WindowsにはパーティションFDISK.EXEが付属しているが、これをドライブCに対して使うには、せっかくインストールされているWindowsを消去してしまうことになる。Linuxをインストールするのも大変そうなのに、その上Windowsまで再インストールしなければならないというのは閉口である。

こんなときのために、多くのLinuxディストリビューションCD-ROMの中にはFIPS.EXEというパーティション分割ツールが収録されている。もちろんTurboLinuxにも付属しているのだ。FIPS.EXEとは、現在使用中のWindows環境を破壊することなく、空き領域から別のパーティションを切り出すという、優れモノなのだ。たとえば4.3GバイトのハードディスクにWindowsがインストールされていて、空き容量が2Gバイトあったとすると、そこから、2Gバイトのパーティションを切り出せるのだ。もちろん、残った2.3Gバイトのパーティションでは何事もなかったかのように今まで通りのWindowsが起動する。あぁ便利！ ただし、FIPS.EXEはプライマリDOSパーティション（つまりドライブC）にしか適用できない。エクステンドDOSパーティション（ドライブD以降）を分割したい場合は、やはりFDISK.EXEを使うことになる。

■FIPS.EXEでパーティション分割

FIPS.EXEを使うための前準備をしておこう。当然だが、まだWindows上での作業だ。

FIPS.EXEは比較的安全なツールだと思われるが、製作者も保証はしていない。編集部も筆者もその動作を保証しないし、入力操作で間違った値を入力すれば結局データは破壊される。不安があれば、ここで念のため既存のWindowsのデータはバックアップしておくべきだ。

まず、TurboLinuxの「インストール

写真2-1　SONY VAIO PCG-N505

CD」のディレクトリ\dosutilsの下に「Fips20.zip」というアーカイブファイルがあるので、これをWinZipなどのツールで解凍しておく（ちなみにすでに同じディレクトリにFIPS.EXEがあるが、これはバージョン1.5でFAT32に対応していないので使ってはいけない）。FIPS.EXEはマルチタスク環境下で実行してはいけない。したがって、次にフロッピーディスクを1枚用意し、「起動専用」でフォーマットする。そこに先ほど解凍したファイル群の中から「FIPS.EXE」と「ERRORS.TXT」をコピーする。

FIPSの作業に入る前にもう1つだけ。Windowsを長く使っているとファイルの書き込みや削除で、断片化されたデータがディスクのそこここにちらばっている状態になっている。パーティションを切り出す前に、念のために、これらをつなぎ合わせて、ディスクの内周近辺にまとめておく必要がある。Windowsの「デフラグ」をかけておこう。

さて、いよいよ、先ほど作成したフロッピーディスクから起動する。DOSプロンプトが表示されたら「FIPS」と入力しEnterキーを押す。英語のメッセージでわかりにくいが、対話式でパーティションを切り直していく。

パーティション分割を終えたら、Ctrl＋Alt＋Deleteキーでシステムを再起動して、Windowsを起動し、ドライブCのサイズが変更されているかどうか確認しよう。このとき「マイコンピュ

ータ」に新たにドライブDが追加されているが、切り出したばかりなので、フォーマットされていないためアクセスはできないので注意してほしい。

■ハイバネーション領域の確保

多くのノートパソコンでは、バッテリ切れに備えて、ハイバネーションという機能が用意されている。長時間電源を入れっぱなしにして放置した場合などに、メモリの内容やCPUのレジスタの内容をハードディスクに退避し、自動的に電源をオフにする機能だ。もちろん次に電源をオンにした時に、退避した情報を読み出して、電源オフの直前の状態に復帰する。

VAIOノートに搭載されているPhoenix BIOSは、このハイバネーション機能をサポートしている。ただし、購入時にはWindows用チューニングがされており、ハイバネーション領域はドライブCの「save2dsk.bin」というファイルに設定されているのだ。これでは、Windowsからは使えても、Linuxからは使えない。そこで、Windows以外のOSからもハイバネーション機能が使えるように、ハイバネーション領域を独立したパーティションとして確保する必要がある。

VAIO購入時に「プロダクトリカバリCD-ROM」というものが付属していたはずだ。その1枚目にPHDISKというツールが収められている。

せっかくだから、先ほどのFIPSをコピーしたフロッピーディスクにこのPHDISK.EXEをコピーしよう。自分のパソコンのメモリ容量を調べておくのも忘れないように。ではフロッピーディスクで起動して、PHDISK.EXEを実行する。

```
A>PHDISK /CREATE /PARTITON /RAM64
```

/RAMの後ろの64はメモリ容量だ。もし128Mバイト搭載していれば/RAM 128と指定する。これでハイバネーション領域が確保できた。ちなみにハイバネーションまでの時間設定などは、BIOSで設定できる。起動時に画面に「SONY」ロゴが表示されているうちにF2キーを押せばBIOS画面を呼び出せる。

ただし、VAIOノートでは、ハイバネーションからの復帰時にXの画面が乱れるという問題がある。動作自体には影響はないのだが…。

■ハードウェアの調査

Linux用のパーティションも用意できたところで、いよいよインストール開始といきたいところだが、ちょっと待った。まだWindowsを起動しているうちに、いくつかのハードウェア設定を調べておこう。[システムのプロパティ]—[デバイスマネージャ]を使って各設定値を調べておくといい。特に「ディスプレイアダプタ」、「PCMCIAソケット」、「サウンド、ビデオ、およびゲームのコントローラ」、「モデム」について、チップのメーカー名、IRQ、I/Oアドレス、DMAチャネルなどを調べておく。VAIOのインストールで必要になるのは、ディスプレイアダプタのVRAM容量、サウンドチップのIRQとI/OアドレスとDMAチャネル、モデムが使用するCOMポート番号などだ。

ついでと言っては何だが、ここで、ふだん利用しているプロバイダのアクセスポイントの電話番号と、アカウント、パスワードも確認しておくと二度手間にならない。

インストール開始!?

さて、いよいよインストール開始である。結論から先に言うと、VAIOノートではインストール手段が限られる。CD-ROMブートを使うしかないと言っても過

Column

初代VAIOノート505

ところで、Linuxをインストールするのに、いったいどれぐらいのハードディスク容量が必要だろうか？　TurboLinuxのマニュアルによれば「600Mバイト以上、最低200Mバイト程度」とのことだ。しかし、最低の容量ではサイテー！に違いない。本誌の読者ならば、毎号の付録CD-ROMを楽しむためにも、確実に1Gバイト以上はほしいところだ。この時点で初代VAIOノート505ユーザーはショックを受けるかもしれない。なぜなら、内蔵ハードディスク容量がちょうど1Gバイトピッタリだからだ。分割とか言ってるレベルではない（とは言え、今回は無理やり400Mバイト空けて分割してみたりもしたが）。Linux専用マシンにするか、思いきってハードディスクの交換の外科手術をするか、悩むところだろう。

旧型機のユーザーは、とにかく自分のマシンと相談して、なんとか1Gバイト以上の空き領域を捻出しよう。よーく探すと、プレインストールされているけど使ってないアプリケーションがザックザックと発掘されるかもしれない。こうしたアプリケーションはWindowsの[コントロールパネル]の[アプリケーションの追加と削除]を使ってアンインストールすると安心だ。

初代VAIOノートPCG-505

言ではない。VAIOノートでは、SONYの純正CD-ROMドライブ（PCGA-CD5、PCGA-CD51）や、メルコのCDN-D24VAなどを使うとCD-ROMからの起動が可能だ。しかもTurboLinux 4.xの「インストールCD」はブータブルになっているため、この組み合わせで、いきなりCD-ROMからインストールできるのだ。では、他の方法ではなぜダメなのか？　まずフロッピーディスクでのインストールは、CD-ROMドライブを認識させるために2枚目のフロッピーディスクを要求するのだが（PCMCIAカード認識のため）、それが正しく読み込まれない。では、マニュアルの64ページに記載されているフロッピーレスインストールはどうだろうか？　こちらは試してみたが、インストーラに問題があるようで、キーボード入力を受けつけてくれない。ちょっとトリッキーな手段として、CD-ROMの内容をドライブCにコピーしておいて、そこからインストールという手段も考えられたのだが、TurboLinux 4.xのインストーラからはその項目がなくなっている。残された手段が冒頭に紹介したCD-ROMブートである。

ようやくインストールの開始だ。ここでは純正のPCGA-CD51を使う。「インストールCD」をCD-ROMドライブに挿入し、システムを起動する。CD-ROMからTurboLinuxインストーラが起動し、「boot:」プロンプトが表示される。通常ならここで何も入力せずにリターンだが、VAIOノート＋CDブートの場合、プロンプトに続けて以下のように入力する。

```
install ide2=0x180,0x386
```

Enterキーを押せばインストール開始だ。

インストール中の選択肢で注意が必要なのは「PCMCIAサポート」だ。ここで「いいえ」を選ぶ。でないとインストールが進まなくなる。PCMCIAはインストーラ終了後に設定すればいい。

■自動パーティション設定

「ディスクの分割方法」では「自動」を選ぶ。「自動パーティション」も「OK」を選ぼう。「ディスクパーティション」が表示されたら、よく見てほしい。hda1はWindowsパーティション、つまりドライブCだ。“ 使用しない ”になっているのでそのままにしておく。hda4は“ IBM ThinkPad Hivanation ”とあるが、これがハイバネーション領域だ。当然ここも使用しない。残るhda2が先ほどLinuxのために用意したパーティションだが、現在はWindows用として認識されているため、これも“ 使用しない ”になっている。こちらとしては、ここを使用してほしいわけだ。カーソルキーでハイライトをhda2に合わせてEnterキーを押すと、メニューが表示される。そこで「このパーティションを消去」を選択する。すると、hda2の内部が“ / ”、“ /usr ”、“ /home ”などに細かく分割される。Linuxに割り当てるパーティションさえ間違えなければ、あとは自動パーティションの設定値で問題ないだろう。

「マウントテーブルの設定」に先ほど見たhda1が表示されていれば、これをWindowsディレクトリとしてマウントできるようにしておくと便利だ。“ /dev/hda1 ”にハイライトを合わせて、「編集」を選択する。マウントポイントの設定に“ /mnt/win ”と入力し、「OK」を選択しよう。

■カーネルの選択とLILO設定

「カーネルの選択」では、「APMカーネル（電源管理）」を選ぶ。これで液晶パネルを閉じると自動的にスタンバイモードに切り替わるなどの電源管理が設定される。

「LILO設定」は好み次第だが、WindowsもLinuxも頻繁に使うならば「マスターブートレコード」を選ぶ。オプション設定は何も入力せずにそのまま「OK」、起動可能パーティション」では“ /dev/hda1 ”にハイライトを合わせて、「編集」を選び、起動名に“ win ”（何でもかまわない）と入力する。

■X Window Systemの設定

ノートパソコンにLinuxをインストールする場合の最大の難関がX Window Systemの設定である。ここでは、インストール後に若干の修正を行う前提で、とりあえず仮の値を設定していく。

「ビデオカードの自動認識」に「はい」を選ぶ。検出結果は“ Neo Magic NM2200 ”になっているはずなので「OK」を選ぶ。「検出データ」では「検出値で設定」を選ぶ。

「マウス設定」では「一般的なPS/2マウス」を選べばいい。オプションの「3ボタンエミュレーション」は好みに応じてチェックしよう。

「ディスプレイ選択」では「800x600 or 1024x768 LCD」を選ぶ。「デフォルト色数」は「16bitカラー（65536色）」かそれ以上の色数を選んでおこう。画面サイズの設定は液晶パネルの表示能力に応じてチェックする（あるいはチェックを外す）。「周波数」も「自動設定」を選ぶ。「フォント解像度」は美しく表示させたいなら「100dpi」を選んでおこう。「75dpi」でも問題ない。

「設定のテスト」で「OK」を選ぼう。もしマウスカーソルが正しく表示されないなどの問題が起こったら、Ctrl＋Alt＋Backspaceを押してテストを終了し、「テスト結果」に「はい」を選んで、ここは先に進もう。「ログイン方法」の選択は「テキストログイン」を選んでお

く。「グラフィカルログイン」にしたい人は、後でXの設定を手作業で修正してから変更すればいい。

■PPP設定

「PPP設定」で「追加」を選び、プロバイダの電話番号、アカウント、パスワードを入力する。パスワードは画面に表示されてしまうので注意が必要だ。ここで「シリアルポート」を選び“ COM2 ”が選ばれているかどうかも確認しておこう。その他のオプションは通常は必要ない。ここで設定したPPPのインターフェイス名は“ ppp0 ”になる。正しく設定できていれば、インストール終了後にコマンドラインから“ /usr/sbin/ppp-on ppp0 ”と入力することでこの接続が確立され、“ /usr/sbin/ppp-off ppp0 ”と入力することで切断されるはずだ。これでFTPやWebを利用してオンラインのデータや情報をウハウハにゲットできるようになった。

インストーラ終了後の設定

サービスやユーザーアカウントの作成を終えれば、めでたくインストールが終了する。自動的にシステムが再起動する。SONYロゴが表示された後に“ LILO boot: ”プロンプトが表示される。ここで“win”と入力すればWindowsが起動し、“ linux ”と入力するか、何も入力しないで5秒たつとLinuxが起動する。Linuxが起動したら、ログインしよう。

インストール作業で先送りにした、/etc/X11/XF86Configを手作業で書き換えなければならない。一般ユーザーでログインした場合は、以降の設定作業すべてについて、suコマンドで一時的にrootになっておくこと。

ビデオチップのメモリ容量を手作業で指定した場合は、1024Kバイト単位でしか選択できなかったので、これを実際の値に変更する。2.5Mバイトの場合、以下のようにVideoRamの行を2560に書き換える。

```
VideoRam     2560
```

また、マウスカーソルが正しく表示されない場合、XF86Configにある以下のオプション行のコメントを外す(「#」を削除する)。

```
# Option "sw_cursor"
```

VAIO C1の場合、画面のモード自体が1024×480という特別なものなので書き換える部分が多い。変更箇所を**リスト2-1**にまとめておく。

設定が済んだらXを起動してみよう。コマンドラインから“ startx ”と入力すればいい。正しく起動しない場合や、

▼リスト2-1 VAIO-C1用のXF86Config設定(抜粋)

```
Section "Monitor"
    Identifier  "Custom Monitor"
    VendorName  "SONY"
    ModelName   "PCG-C1DisplayPanel"
    HorizSync   20-64          ← 水平同期周波数を変更
    VertRefresh 50-100         ← リフレッシュレートを変更

Modeline "1024x480" 65.00 1024 1032 1176 1344 480 491 493 525 -hsync -vsync
                               ← ほかのModelineを削除し、この行に置き換える
EndSection

Section "Device"
    Identifier  "AutoProbed (NeoMagic 2200:PCI)"
    VendorName  "AutoProbed (NeoMagic 2200:PCI)"
    BoardName   "NeoMagic 2200:PCI"
    VideoRam    2560           ← ビデオRAMの容量を変更
    Option      "override_validate_mode"   ← オプションを追加
    # No probed static clocks for this card (good)
EndSection

Section "Screen"
    Driver      "svga"
    Device      "AutoProbed (NeoMagic 2200:PCI)"
    Monitor     "Custom Monitor"
    DefaultColorDepth    16
    Subsection "Display"
        Depth     8
        Modes    "1024x480"    ← 解像度を変更
    EndSubsection
    Subsection "Display"
        Depth     16
        Modes    "1024x480"    ← 解像度を変更
    EndSubsection
EndSection
```

写真2-2 SONY VAIO PCG-C1

表示がおかしい場合は、Ctrl＋Alt＋BackspaceでXを終了して、もう一度XF86Configを確認してみる。うまく設定できたら、turboxcfgを起動してログイン方法を変更するのもいいだろう。このとき起動前にkonを起動しておかないと英語メッセージのturboxcfgが起動するので注意が必要だ。Xが動くと急激に世界が広がる。Gimpを使ってお絵描きができるし、Netscape Navigatorも使えるようになるのだ。

■サウンドの設定

Linuxからサウンドを使えるように設定しよう。turbosoundcfgを起動する。ここでは「SoundBlasterPro」を選び、「OK」を押す。「カード設定」に調べておいた各設定値を入力する。

```
I/Oアドレス: 0x220
IRQ: 5
8bitDMA: 1
```

テストで銅鑼の音がじゃーん！と鳴ればOKだ。これでxampでMP3プレイヤーも楽しめるし、xanimでビデオがサウンド付きで再生できるようになったはずだ。バンジャーイ！

■CD-ROMドライブの設定

インストール時に放棄したPCMCIAの設定をしておこう。ここからの作業はXを終了してコンソールで進めたほうがわかりやすいかもしれない。

まずは、カードサービスが正しいドライバを読み込むように、/etc/sysconfig/pcmciaを**リスト2-2**の通り設定する。“ irq_list= ”には、インストール前に調べた空きIRQをいくつかリストしておこう（筆者はIrDAを使わないので10番もつぶしている）。これをしないとIRQがぶつかって正しくカードが認識されなかった。

▼リスト2-2 /etc/sysconfig/pcmciaの設定例（PCMCIA）

```
PCMCIA=yes
PCIC=i82365
PCIC_OPTS="irq_list=4,6,10,11 poll_interval=100"
CORE_OPTS="unreset_delay=300"
```

▼リスト2-3 /etc/pcmcia/configの設定例（純正CD-ROM）

```
card "Sony PCGA-CD51 CD-ROM"
  version " ", "NinjaATA-", "V1.0", "AP00 "
  bind "ide_cs"
```

名前は任意（ログやcardinfoで表示される）

```
hdc: TOSHIBA CD-ROM XM-1902B, ATAPI CDROM drive
ide2: ports already in use, skipping probe
ide1: at 0x180-0x187,0x386 on irq 10
```

hdcにアサインされた

画面2-1 cardmgrがカードを認識したログメッセージ

次に/etc/pcmcia/configにカードのID情報を追加する。**リスト2-3**は純正CD-ROMドライブの場合の設定だ。すでに“ Ninja ”のエントリは記述されているので、それを検索して名称やバージョンを追記すればいい。手持ちのカードのID情報がわからない場合は、この後カードサービスを再起動してから、カードを差してみて、ログファイル/var/log/messagesの内容を確認するとわかる。

これで準備は整った。カードサービスをいったん終了させて再起動する。カードサービスの開始と終了は、/etc/rc.d/init.d/pcmciaコマンドに“ start ”か“ stop ”の引数を指定する。ここではいったん引数に“ stop ”を指定し、次に“ start ”を指定しよう。それぞれ開始と終了の1行メッセージが表示されるのでわかりやすい。

カードサービスが再起動できたら、カードをスロットに差し込む。コンソールならば、画面にcardmgrからのログメッセージが表示される。/dev/hdcにアサインされているのがわかる（**画面2-1**）。

あとは、mountコマンドで/dev/hdcを/mnt/cdromにマウントすればいい。

```
# mount -t iso9660 -o ro /dev/hdc
/mnt/cdrom
```

注意が必要なのは、ここで使用しているようなIDEカードの場合、スロットから抜くとkernel panicになってシステム自体が落ちてしまう。他のIDEカードで試しても同様の結果だった。IDE CD-ROMを利用する場合はシャットダウンするまでカードを抜くことはできない。試した結果、SCSIカードではこうしたことは起きないので、頻繁にカードを抜き差ししなければならない人は、SCSI CD-ROMを使うべきかもしれない。

以上、通りいっぺんではあるが、最低限必要なインストールは紹介した。もちろん、このままでは決して使いやすいとは言えないのも事実だ。しかし、ここまでくれば、使いやすいように環境をカスタマイズするお膳立てはできているはずだ。あとは、インターネットや本誌付録CDの有用なソフトやドライバを活用して、ガンガンカスタマイズしよう！

ミニノート 東芝 Libretto ff 1100CTA

文：細川 茂一
Text : Moichi Hosokawa

今回ミニノートへのLinuxインストールということで選ばれたマシンは、東芝Libretto ff 1100CT（以下Libretto ff）だ。このLibretto ffは800×480ドット液晶の採用、USB、スマートメディアスロット、モデムの内蔵、そしてリモコンとカメラが付属しており、従来のLibrettoとひと味違ったデバイスが満載である。Libretto ffをターゲットマシンとして、リリースされて間もないLASER5 Linux 6.0日本語入力キットのインストールにチャレンジしてみた。フリー版もインストールの手順は同じはずなので参考にしてほしい。

■FIPSでパーティション分割する

Libretto ffのハードディスクは容量3.2Gバイトで、パーティションは分割されずに、Windows98用にFAT32でフォーマットされている。Linuxをインストールする最初の作業として、FIPSコマンドを使ってパーティションを分割する。FIPSコマンドは、LASER5 Linux 6.0バイナリCD-ROMのディレクトリの/dosutilsにある。FAT32に対応したfips.exeを使用する必要があるので、今回は/dosutils/fips20に入っているバージョン2.0のfips.exeを使用した。

Linux用のファイルをまとめるためにc:\linuxというディレクトリを作成し、そこへfips.exeをコピーする。次に、MS-DOSモードで再起動する。FIPSコマンドはWindows上では動作しないからだ。cd c:\linuxでc:\linuxディレクトリに移動し、fips.exeを実行する。パーティションの分割の割合だが、今回は2GバイトをWindows98用領域として確保し、残り1.2GバイトをLinux用に使用することにする。

インストーラがUSBフロッピーを認識しない

LASER5 Linux 6.0をインストールするにあたって最初に障害となったのは、Libretto ffのフロッピーディスクドライブがUSBタイプのものであり、インストールに使用できないことであった。Libretto ffの取扱説明書には、USBレガシーエミュレーションを有効にすればフロッピーディスクから起動できると書いてあったので、その設定で通常のフロッピーディスクを使ったインストールが可能と考えていた。

しかし、インストール用ブートディスクで試したところ、LASER5 Linux 6.0の起動画面は出るもののRETURNキーを押してLinuxのカーネルを起動させた後はフロッピーディスクドライブが認識されず、結果としてフロッピーディスクを使ったインストールができなかった。

USBレガシーエミュレーションはI/Oポートのエミュレーションを期待していたのだが、実際はBIOSレベルのエミュレーションであり、BIOSを使わないLinuxのフロッピーディスクドライバでは使えない。どうやらこの問題はSonyのVAIO C1シリーズでも起きている比較的メジャーな問題のようだ。

写真2-3　東芝Librette ff 1100CT

LASER5 Linux 6.0のインストーラはPCMCIAサポートを必要とする場合に補助フロッピーディスクを要求してくるので、フロッピーディスクが使用できないとPCMCIAのSCSI CD-ROM経由やネットワーク経由でのインストールができず、LASER5 Linux 6.0のインストールガイドに記述された方法では事実上インストール不可能ということになる。

フロッピーディスクが使えないのではマニュアル通りのインストールはできない。そこでWindows98の2Gバイトのパーティションにインストールに必要なファイルをコピーし、loadlin.exeを使ってMS-DOS上からインストーラを起動してこの問題を回避することにした。最初に断っておくが、このインストール方法は正式サポートされていない方法なのでそのことをよく考え、インストールする場合は自分の責任で行ってほしい。うまくインストールできなくてもLASER5に質問することはできない。

ハードディスクにLinuxイメージをコピーする

まずLASER5 Linux 6.0の/dosutilsディレクトリに納められているautoboot.bat、autobootディレクトリの中身、loadlin.exe、それからインストールに必要な/LASER5ディレクトリ以下をc:\linuxにコピーする。インス

トールに必要なファイルは全部で570Mバイトぐらいあるので、Libretto ffにプレインストールされているいくつかのアプリケーションをいったん削除して空き領域を確保する必要がある。Windows98のエクスプローラでPCカードで接続したCD-ROM、またはネットワーク経由でほかのマシンからコピーする。

必要なファイルをc:\linuxディレクトリにコピーしたら、いよいよインストールとなるのだが、その前にPCMCIAの設定をautoからCardBus/16bitに変更しておこう。autoの設定では、LASER5 Linux 6.0のカーネルがPCMCIAの認識に失敗することがあった。PCMCIAの設定への変更は、Windows98の[コントロールパネル]―[TOSHIBA Hardware Setup]から行う。

MS-DOSからインストーラを起動

PCMCIAの設定が終わったら、いよいよインストーラの起動だ。FIPSコマンドのときと同様にMS-DOSモードで再起動した後、usコマンドでUSモードに変更する。MS-DOSプロンプトからcd c:\linuxでc:\linuxディレクトリに移動し、autoboot.batを実行する。autoboot.batを実行するとLinuxが起動し、その後LASER5 Linux 6.0のインストーラが起動する。この後はインストーラの指示に従って進んで行けばよい。インストールメディアには「Hard drive」を選択、インストールに必要なファイルの場所には、パーティションにWindows98のパーティションである/dev/hda1を指定、LASER5があるディレクトリには/linuxを指定する。

次のインストールパスではインストールを選択し、インストールタイプにはカスタムを選択する。FIPSでパーティション分割した直後は、2Gバイト+1GバイトのFAT32パーティションというディスクレイアウトになっているので、1Gバイトのパーティションを Linux用に確保し直す必要がある。ディスクのセットアップでは初心者にも分かりやすい「Disk Druid」を選択してLinux用パーティションを作成しよう。まず/dev/hda2のFAT32パーティションを削除し、次にLinux用パーティションを作成する。今回は、ルートパーティションに1050Mバイト、スワップパーティションに51Mバイトを指定した。ディスク容量が1Gバイトしかないので、細かくパーティション分割を行うと空き領域が分散してしまい、追加パッケージを入れる容量が確保できなくなる恐れがあるからだ。

パーティションの編集が完了したら、「了解」を押して完了する。次にスワップパーティションの初期化画面に移るが、ここでハードディスクからのインストール時に発生するエラーに対処するための操作を行う。この操作を行わずに普通にインストールを続けると、スワップパーティションの初期化が終わってから「Cannot open /tmp/rhimage/LASER5/base/install3.tr」というエラーが発生し、インストーラのメッセージが一部英語になってしまう。エラーの影響はメッセージだけでインストール自体に問題はないので、このままでも続行できるが、以下に示す簡単な操作を行うことでこのエラー自体も回避することが可能だ。

まず、スワップパーティションの初期化のところで、Alt+F2を押してコンソールのシェル画面に移る。次に、

```
# mknod /dev/hda1 b 3 1
```

というコマンドを実行し、続けて、

```
# mount -t vfat /dev/hda1/tmp/hdimage
```

を実行する。最後にAlt+F1で元のインストーラ画面に戻れば完了だ。FAT32パーティションをあらかじめインストーラが必要とする場所にマウントしておくことにより、メッセージファイルが読み込めないエラーを回避するものだ。

スワップパーティションの初期化が終わると、パッケージの選択に移る。パッケージの選択では、「個々のパッケージを選択する」をぜひチェックしよう。これをチェックすることで細かい指定が可能になるという利点のほかに、選択したパッケージのインストールに必要な容量が表示されるからである。この容量をチェックして、容量がオーバーしないように好きなパッケージを選択する。

インストールするパッケージの選択が完了すると、パーティションのフォーマットに移るので、表示されているデバイスを選択する。インストーラはパーティションをフォーマットした後、パッケージを次々とインストールしていく。パッケージのインストール画面に経過のグラフが表示されるので、残り時間をチェックしてここらで一服しよう。

■インストーラでの設定注意点

パッケージのインストールが完了すると、次はシステムのセットアップに入る。マウスの設定では一般的なマウス（PS/2）を選択し、3ボタンエミュレーションを有効にする。その次のネットワークの設定では、「いいえ」を選択する。PCMCIA Ethernetカードの設定は、インストール後にlinuxconfユーティリティで行うからだ。

タイムゾーンはデフォルトのJapanを選択し、起動するサービスの設定は特にこだわらなければデフォルトのままでOKだ。プリンタの設定はここで設定してもよいが、インストール後にコントロールパネルでも変更可能なのでとりあえず設定しなくともよい。rootのパスワード、ログイン認証の設定の後、ブートディスクの設定となる。Libretto ffのUSBフロッピーディスクはインストーラからは認識されないので、ここは「いいえ」を選択する。

LILOの設定では、注意が必要だ。Disk Druidはデフォルトで Linux用パーティションを論理パーティションに確保する。しかしながら、標準のマスターブートレコードは論理パーティションからブートできないので、この場合は必ずLILOをマスターブートレコードに書き込む必要がある。また、LASER5 Linux 6.0のLILOインストーラは、デフォルトの起動OSを今インストールしているLinuxに固定している。Windows98を起動したい場合はマシンブート時のLILO:プロンプトからdosとタイプする。インストール後にlinuxconfユーティリティを使用して、起動OSのデフォルトをWindows98に設定することも可能だ。

LILOの設定が終わると、いよいよX Window Systemの設定だ。Libretto ffは変則的な800×480ドットの画面を持つためか、インストーラの自動検出がことごとく失敗してしまう。ここでは、外部ディスプレイ表示のことを考えモニタの設定はカスタムで「高周波SVGA, 1024x768 垂直同期 70Hz」を選択し、垂直同期は「50-100」を選択した。スクリーンの設定では、ビデオメモリを「2mb」に、クロックチップの設定は「クロックチップ設定を行わない」を選択した。次の解像度の設定では、800×480ドットの設定がないので640×480ドットを選択するが、800×600ドットにパニングされてしまいフォントがギザギザでかなり汚い。しかも縦が600ドットになってしまったので、画面の480ドット以降が見えない。800×600ドットを選択すればパニングの問題は回避されたが、480ドット以降が見えないのは変わらない。このままではせっかくのLibretto ffの画面を生かせないので、インストール後に/etc/X11/XF86Configを編集して800×480に合わせることにする。

インストール後の設定

これでLASER5 Linux 6.0のインストールが完了したが、まだPCMCIA、X、サウンドの設定が完了していない。

PCMCIAの設定は、/etc/sysconfig/pcmciaを編集し、最初の2行を「PCMCIA=yes」「PCIC=i82365」と変更する（**画面2-2**）。これでPCMCIAのマネージャであるcardmgrがブート時に起動される。すぐに試したい場合は、

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

を実行すればよい。PCMCIA Ethernetカードを使用するには、linuxconfで「基本ホストの情報」を選択し、アダプタ1の項目に、設定モードとネットワークのパラメータを設定するだけで基本的にOKだ。このとき「有効化をoff」「ネットワークデバイスをeth0に設定」「カーネルモジュールは空欄」に設定することに気をつける。

LASER5 Linux 6.0のlinuxconfは日本語化されているので、コンソール上でlinuxconfを実行する場合、日本語表示のためにあらかじめkonを実行しておく必要がある。また、インストーラのbugのため、インストール直後はlinuxconfを終了させるときに「未来の日付になっている」というエラーが表示されるが無視してかまわない。

■Xの設定

/etc/X11/XF86Configを編集して、800×480でXサーバが起動されるようにしよう。まず、Monitorセクションに800×480のモードラインを追加する（**画面2-3**）。次に、SVGAサーバのDeviceセクションにOptionとClocksを追加する（**画面2-4**）。最後に、SVGAサーバのScreenセクションに800×480モードを追加し、Depth 16以外のDisplayサブセクションを削除する（**画**

```
PCMCIA=yes
PCIC=i82365
PCIC_OPTS=
CORE_OPTS=
```

画面2-2
PCMCIAの設定。/etc/sysconfig/pcmciaを編集する

```
209 # and video card can support for a given resolution is automatically
210 # used.
211
212 # 800x480
213 Modeline "800x480" 40 800 864 928 1088  480 481 484 509 +hsync +vsync
214
215 # 640x400 @ 70 Hz, 31.5 kHz hsync
216 Modeline "640x400"     25.175 640  664  760  800   400  409  411  450
```

画面2-3
Monitorセクションに800x480のモードラインを追加する

面2-5)。

編集が終わったら、Alt+F7でXの画面に戻り、Ctrl+Alt+BackspaceでXサーバを再起動させよう。Xの画面が800×480できちんと表示されれば成功だ。

サウンドの設定はsndconfigコマンドで行う(**画面2-6**)。

■そのほかの周辺装置は使えない?

Libretto ffにはまだSmartMediaスロット、内蔵モデム、USBなどのデバイスが装備されている。SmartMediaスロットはPCMCIAのスロット1として認識され、フラッシュメモリカードが挿入された状態と認識されるようだ。I/Oデータなどから発売されているフラッシュATAカードとして認識されるSmartMediaアダプタと違い、Linux上からはハードディスクとして認識されないので、そのままではデジタルカメラで撮った写真がファイルとしてアクセスできなかった。ただし、SmartMediaスロット自体はフラッシュメモリカードとして認識されているので、LinuxのPCMCIAドライバ次第でアクセスできる可能性はある。内蔵モデムはWinmodemのようで、Linux上からはモデムとしてアクセスできなかった。USB、赤外線は今回時間の都合で動作確認はしていない。

Linux上でサスペンドは動作したが、ハイバネーションが全く動作せずサスペンドになってしまう。ハイバネーションが動作しない理由について調べたところ、どうやらLibretto SS1000シリーズからハイバネーションはBIOSレベルではなくWindowsのOSレベルで行っているようで、Windows以外のOSではハイバネーションが動作しないようだ。最近のLibrettoの売りである高速なハイバネーションがWindowsのみでしか利用できないのは残念な話だが、サスペンドは動作するので電源オンですぐにLinux上で作業可能だ。

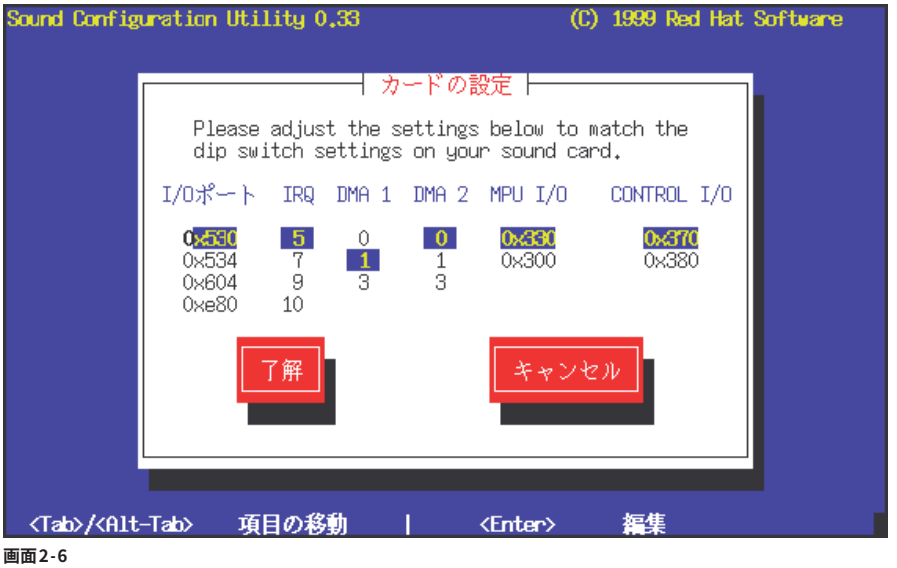
今回のインストールを通じて、Libretto ffにも問題なくLinuxがインストールできることが確認できたが、リモコン、カメラ、SmartMedia、内蔵モデムなどのLibretto ffの魅力的な機能はWindows98以外では利用できないのが現状だ。これらの機能も使いたいというユーザは、Windows98とLinuxの混在環境でTPOに合わせてどちらかのOSを使うという使用方法がベストだと感じた。

```
408 #    Option     "lcd_center"
409 #    Option     "no_stretch"
410    # Insert Clocks lines here if appropriate
411    Option "override_validate_mode"
412    Clocks 25.2 28.3 40.0
413 EndSection
```

画面2-4
SVGAサーバのDeviceセクションにOptionとClocksを追加する

```
420 # The Colour SVGA server
421
422 Section "Screen"
423     Driver      "svga"
424     # Use Device "Generic VGA" for Standard VGA 320x200x256
425     #Device      "Generic VGA"
426     Device      "My Video Card"
427     Monitor     "My Monitor"
428     Subsection "Display"
429         Depth       16
430         Modes       "800x480"
431         ViewPort    0 0
432     EndSubsection
433 EndSection
```

画面2-5
SVGAサーバのScreenセクションに800×480モードを追加し、Depth 16以外のDisplayサブセクションを削除する。



画面2-6
サウンドの設定はsndconfigコマンドで行う。

Section 3

モバイルLinuxの活用テクニック

ネットワーク設定は？　メーラの設定は？　データの管理は？
モバイルLinuxユーザーのための必須テクニックを一挙に公開!!

文：藤沢敏喜
Text : Toshiki Fujisawa

さて、ここまででノートPCでLinuxが使えるようになった。しかし、まだやっておかないといけないことが残っている。

ノートPCは、デスクトップPCの常識がそのままでは当てはまらない。それは、ノートPCが持ち運んで使うものだからだ。持ち運んで使うために考慮しておかないといけないことに、次のようなことがある。

・ネットワーク環境の切り替え

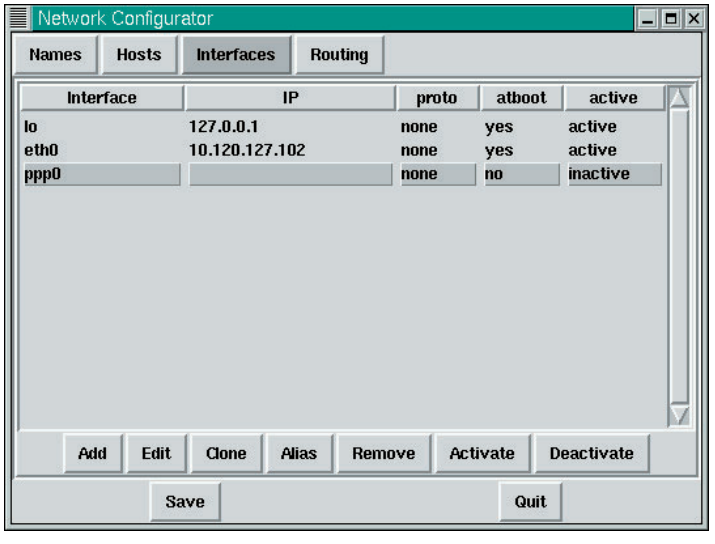
利用先（たとえば、会社と自宅）によって、IPアドレスをはじめとするネットワーク構成やネットワークデバイス（イーサネットやPPPなど）が変わってしまう。また、メールサーバやDNSなど各種サーバ設定なども変更しなければならない。これらの設定情報をなるべく簡単に切り替えられるようにして、使いやすくする必要がある。

・メールのオフライン送信

ネットワークに接続されていないときに書いたメールを一時的にためておき、ネットワークが接続されたときに、それらのメールが一気に送信されるようにしておく必要がある。

・メールの管理方法

出先でノートPCからメールを読んだり、会社でデスクトップPCからメールを読んだりしていると、メールがバラバラになってしまい、検索性が低下する。できることなら、メールは一括して管理しておきたい。



■画面3-1　ネットワーク設定コマンド「netcfg」
LASER5 Linux 6.0上で起動したnetcfg。インターフェイスを選択し、下にある［Activate］［Deactivate］ボタンをクリックすれば、インターフェイスの起動が行える。

・データの同期

デスクトップPCとノートPCにそれぞれあるデータが同一かつ最新の状態に保たれるよう、データを上手に同期させる必要がある。

・データのバックアップ

持ち歩くということは、ノートPCの故障や盗難などでデータが失われてしまう可能性が高いということを意味している。そういった事態が起こることをあらかじめ想定しておき、データのバックアップをとっておく必要がある。

ノートPCでLinuxを本格的に利用するためには、こういったノートPCならではの問題を解決しておかなければならない。そこで、このセクションではこれらの問題を解決し、快適なモバイルLinux生活をおくれるような、さまざまなテクニックを解説していこう。

なお、今回はテーマをシステム設定周りに限定して紹介するが、このほかにも運用上の工夫はたくさんある。

たとえば、電源管理である。バッテリメータの利用はもちろん、ヘビーユーザーなら、ノートPC購入時にバッテリを追加購入しておき、予備電源として持ち歩くことも考慮すべきだろう。また、ACアダプタを2つ購入して会社と自宅に置いておけば、多少ながらも重量を軽減できる。泥臭い方法だが、実際やってみると効果は大きい。

さらに、カバンなども重要なアイテムだ。移動が多いノートPCでは、カバンは命綱である。ノートPCそのものはたいした値段ではなくても、そこに入っているデータはとても高価なはずだ。

このように、ノートPCの利用には面倒なことも多いが、使いこなせればメリットも大きい。さあ、モバイルLinuxを始めよう。

ノートPCを持ち歩く場合、利用する環境に合わせて、いろいろな設定を変更する必要がある。たとえば、ある場所では外付けの英語キーボードと外部CRTを使うのなら、それに合わせてキーマップやXの画面サイズなどを変更しなければならない。

しかし、そういった設定変更のなかで、最も切実で、面倒になるのがネットワーク設定だろう。IPアドレスやDNSの指定はもとより、環境変数やWebブラウザなどのプロキシ設定を、その環境に合わせて変更しなくてはならない。

さらに、PHSや携帯電話、ISDN公衆電話を使い分けていたり、海外出張時に別のプロバイダを使いたいときも設定変更が必要である。

たとえば、VineLinux 1.1やLASER5 Linux 6.0のnetcfgコマンドを利用すれば、GUIによるマウスのクリックでインターフェイスの有効／無効を切り替えられるので、比較的簡単にイーサネット環境とダイヤルアップ環境を切り替えることができる（**画面3-1**）。

しかしこの方法では、会社ではDHCPだが、自宅では静的なIPアドレスを利用するといった、同一イーサネットインターフェイスでの切り替えができない。さらに、環境変数やプロキシ設定なども変更できないため、設定変更の手間をあまり軽減できるとはいえない。

だが、FreeBSDやLinuxといったPC-UNIX環境は、ほとんどすべての設定ファイルがテキストファイル形式となっているので、簡単なシェルスクリプトを書くことにより、すべてのネットワーク設定を一瞬で切り替えることが可能である。

何よりも、Linuxを使っているのなら、ここはやはりシェルスクリプトを用いるのが正道だろう（UNIXの強みはそこにあるだから）。

というわけで、ここではネットワーク環境変更のポイントについて解説する。

ネットワーク設定変更のポイント

移動に伴い、注意が必要なるネットワーク設定には次のものがある。

■ルーティングテーブルの削除

電源がオンになったまま違うネットワークに移動した場合や、レジュームから回復したときなどは、今まで利用していた環境のルーティングテーブルが残っている。したがって、このままでは適切にルーティングすることができない。これを削除するにはリブートするのが簡単だが、それではWindowsのようで美しくない。

リブートしないで設定変更するには、netstatコマンドに-rnオプションをつけて、削除すべきネットワークアドレスを見つけ出し、それをrouteコマンドで削除すればいいのだ。これは、たとえば**リスト3-1**に示すようなスクリプトで実行できる。なお、netstatやrouteコマンドはディストリビューションによってオプションなどが異なることもあるので、読者が使っている環境に応じて適当に書き換えてほしい。

■静的なIPアドレスの割り当て

静的にIPアドレスを割り当てている場合は、ifconfigコマンドやrouteコマンドを用いて再度割り当てを行わなくてはならない。頻度が少なければ手動でやってもいいが、頻度が高くなるよ

▼リスト3-1　ルーティングテーブル削除スクリプト

```
#!/bin/sh
#-----------------------------------------------------------------
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH DEL
#-----------------------------------------------------------------
# ルーティングテーブル削除のコマンドを環境変数DELへ設定
case `uname` in
FreeBSD)    DEL="route delete %s" ;;
Linux)      DEL="route del -net %s netmask %s" ;;
*)          { echo "error" ; exit 1; }
esac
#-----------------------------------------------------------------
# netstat -rn の出力からすべてのルーティングを削除する
netstat -rn | awk '
/^127\.0\.0\.0/{
        # ループバックアドレス(127.0.0.0)は削除しない
}
/^default/ || /^[0-9][./0-9]*[ ]/{
        sub( /\/[0-9]+$/ , ""      , $1 )
        if( $1 ~ /^[0-9]+$/                    ){ $1 = $1 ".0.0.0"; }
        if( $1 ~ /^[0-9]+[.][0-9]+$/           ){ $1 = $1 ".0.0";   }
        if( $1 ~ /^[0-9]+[.][0-9]+[.][0-9]+$/ ){ $1 = $1 ".0";     }
        printf ENVIRON["DEL"] "\n" , $1, $3
}' | sh
```

うなら、**リスト3-2**に示すようなスクリプトを作っておくと便利だし、入力間違いによるトラブルをなくすことができる。

なお、ほかの方法としては、自宅のネットワーク環境を会社と同じIPアドレス（プライベートアドレス）にする方法がある。こうすれば、切り替えそのものが必要なくなるので便利である。

また、自分が移動して使用する複数のIPアドレスを、IPエイリアスの機能を用いて、1つのイーサネットに割り振るという方法もある。しかし、ネットワーク環境の切り替えはIPアドレスだけではないので、やはりスクリプトで一気に変更してしまうのがよいだろう。

■DHCPによるIPアドレスの割り当て

移動先のネットワーク環境がDHCPでIPアドレスを割り当てている場合は、DHCPクライアントを動作させると、自動的にIPアドレスが割り当てられる。これなら、スクリプトすら利用する必要がなく、より簡単である。

たとえば、LASER5 Linux 6.0では、DHCPクライアントとしてpumpというコマンドが用意されている。イーサネットインターフェイス名がeth0だとすると、

```
# pump -i eth0 --status
```

を実行することにより、IPアドレスやデフォルトゲートウェイ、/etc/resolv.confなどを書き換えることができる。

また、WIDEプロジェクトで開発されたDHCPクライアント「dhcpc」を用いている場合、インターフェイス名がfxp0だとすると、

```
# dhcpc -r fxp0
```

とすればよい。

なお、DHCPクライアントはディストリビューションによってさまざまなものがあるので、DHCPクライアントを使う場合は、それぞれのマニュアルを参照してほしい。

■アプリケーションの設定

IPアドレス以外にも、Netscape Navigatorのプロキシ設定なども利用環境ごとに変更が必要になる。これは、ECMAScript（JavaScript）などでスクリプトを書く方法もあると思うが、筆者の場合はより単純に、

```
~/.netscape/preferences.js
```

というファイルを、移動した環境により、

```
~/.netscape/preferences.js-home
~/.netscape/preferences.js-work
```

などの実体を指すシンボリックリンクとなるよう、スクリプトを利用して切り替えている。

また、Mewでのメールサーバの切り

■リスト3-3　環境切り替えスクリプト「netsetup.sh」

```
#!/bin/sh
case $1 in
home)
        自宅での設定
        ;;
work)
        会社での設定
        ;;
PHS)
        PIAFS用設定
ISDN)
        ISDN公衆電話用設定
        ;;
esac
```

▼リスト3-2　ネットワーク再設定スクリプト「ifconfig.sh」

```
#!/bin/sh
#------------------------------------------------------------------
# 使用例:
#  % ifconfig.sh eth0 10.6.72.245 255.255.240.0 10.6.64.1 10.6.64.0
#
PATH=/bin:/sbin:/usr/bin:/usr/sbin
#------------------------------------------------------------------
#
DEVICE=$1       # DEVICE=fxp0(FreeBSD) or DEVICE=eth0(Linux)
IP=$2           # IP=10.6.72.245
NETMASK=$3      # NETMASK=255.255.240.0
GATEWAY=$4      # GATEWAY=10.6.64.1
NETWORK=$5      # NETWORK=10.6.64.0
#------------------------------------------------------------------
case `uname` in
FreeBSD)
        ifconfig $DEVICE $IP netmask $NETMASK
        route add default $GATEWAY
        ;;
Linux)
        ifconfig $DEVICE $IP netmask $NETMASK
        route add -net $NETWORK netmask $NETMASK $DEVICE
        route add default gw $GATEWAY
        ;;
*)
        { echo "error" ; exit 1; }
esac
#------------------------------------------------------------------
```

替えなども、~/.im/Configを上記のようなシンボリックリンクで切り替えるのが簡単だ。PPPの設定切り替えなども同じ手法が使えるだろう。

■全環境一発切り替え

上記のような各種のスクリプトを準備しておき、たとえば**リスト3-3**のようなスクリプト（netsetup.sh）で、環境に応じて、用意したスクリプトを呼び出すことにより、コマンド一発ですべての環境を切り替えることができる。

ここで、ネットワーク設定のスクリプトなどはroot権限で動作する必要があるが、いちいちsuコマンドを実行するのが面倒な場合は、一般ユーザーのままルート権限で実行できるように、**リスト3-4**をコンパイルしてnetsetupというプログラムを作成し、

```
# chown root.root netsetup
# chmod 4755 netsetup
```

としておくとよいだろう。

ただし、シェルスクリプトをルート権限（SUID）で呼び出す場合は、ファイルの書き込み権限など、セキュリティ面での注意が必要である。

最後に、このようなプログラムを、ウィンドウマネージャのメニューに入れておけば、マウスで選択するだけでネットワークの環境設定を行うことができるので、非常に便利であろう。

PC-UNIXのカスタマイズ能力

ここまでネットワーク設定を切り替える方法を紹介してきたが、これは一例に過ぎないので、あくまでも参考程度に考えてほしい。モバイルでの利用は各個人によって環境の違いが大きく、なかなか一般化することができないからだ。

筆者のようにすべてスクリプトによって切り替える方法もあれば、netcfgコマンドのようなGUIツールで変更する方法やすべて手作業で変更する方法もある。さらに、それらを併用することもできる。要は、ここまで述べてきたようなネットワーク切り替えに必要な要件を満たしつつ、自分が快適に使えてさえいればいいのだ。

このような柔軟性に富んだ運用ができるのもPC-UNIXが高いカスタマイズ能力を持っているからだ。トラブルやわからないことも、お仕着せじゃない「自由」を満喫しているのだと思って、楽しんでほしい。

▼リスト3-4　一般ユーザーにネットワーク設定を許可するCプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <ctype.h>

int
main(int argc, char **argv)
{
        char    *p;
        char    buf[256];
        int     c = 32;

        /* 引数の個数のチェック */
        if( argc != 2 ){
                fprintf( stderr, "error\n");
                return 1;
        }
        /* バッククォートなど不正な文字がないかを検査 */
        for( p = argv[1]; *p; ++p ){
                if( !isalnum(*p) ){
                        fprintf( stderr, "illegual arg\n");
                        return 1;
                }
                if( c-- == 0 ){
                        fprintf( stderr, "too long arg\n");
                        return 1;
                }
        }
        /* ルート権限を得る */
        if( seteuid(0) == -1 ){
                fprintf( stderr, "Can't seteuid(0)\n");
                return 1;
        }
        if( setuid(0) == -1 ){
                fprintf( stderr, "Can't setuid(0)\n");
                return 1;
        }
        /* 与えられた引数でシェルスクリプトを呼び出す */
        /* 呼び出すスクリプトの書き込み権限に注意!! */
        sprintf(buf, "/usr/local/bin/netsetup.sh %s", argv[1]);
        system(buf);
        return 0;
}
```

ノートPCを使ったモバイル生活を始めると、当然のことながらメールもノートPC上で処理したくなる。しかし、モバイル環境でメールを扱うのは、いろいろと難しい面が多い。具体的には、次のとおりである。

(1) デスクトップPCにPOPされたメールはノートPCでは読めない。逆に、ノートPCでPOPしてしまうと、デスクトップPCで読むことができなくなる。
(2) メールがデスクトップPCとノートPC上にバラバラになって置かれてしまう。
(3) オフライン送信など、モバイル環境特有のテクニックが必要。

ここでは、上記のような問題について考察し、定番のメーラ「Mew」(http://www.mew.org/)を対象に、いくつかの解決プランを提示してみよう。

POPサーバにメールを残す

モバイル生活を実際に始めると、メールサーバから自宅のデスクトップPCでPOPしてしまったメールを、出先で使っているノートPCで読みたいと思うことはよくある。

このような要求を簡単に実現するには、メールをサーバから削除せず残しておく設定にするという方法がある。

現在主流のPOP3プロトコルを用いているメーラのデフォルト設定では、メールサーバからメールを取り込んだ時点で、それを削除するようになっていることが多い。しかし、ほとんどのメーラではオプション指定で、メールをサーバ上に残しておくことができるようになっている。

たとえば、WindowsのAL-Mail32では、[オプション]－[受信]－[受信したメールをサーバに残す]をチェックして有効にすればよい。

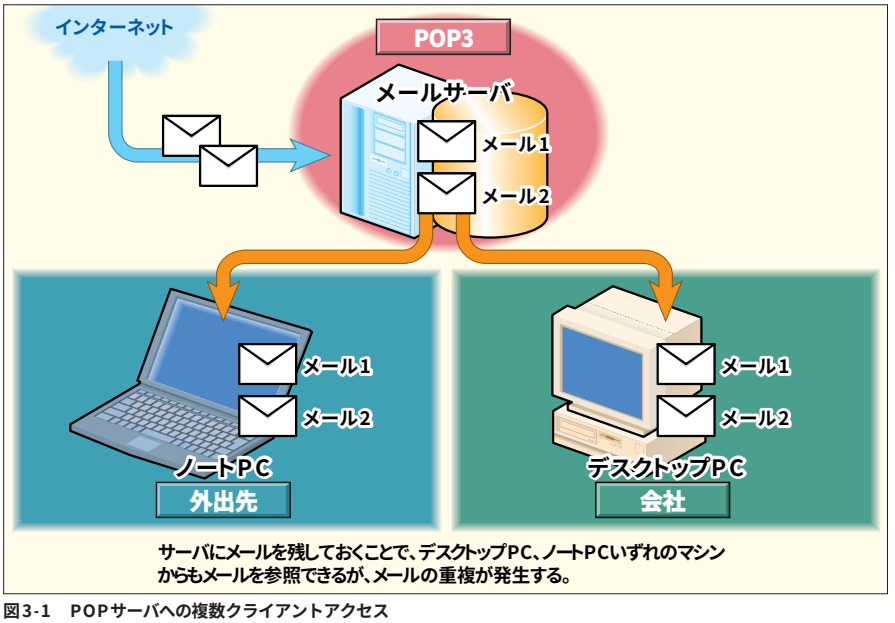


図3-1　POPサーバへの複数クライアントアクセス

また、Mewのメール受信時に利用されているimgetコマンドでは、ホームディレクトリにある、~/.im/Configというファイル中の、

```
#Keep=7 # preserve read messages
                        → on server
```

という行の左の“＃”を消しておくと「7日以内のメールは削除しない」というように設定できる。なお、ここでkeepに指定する数字の意味は、

-1: 永遠に削除しない。
0: 受信後ただちに削除する。
n: 受信後、n日経過してから削除する。

となっている。

この機能を使うには、サーバ上にあるメールの既読と未読を識別するための設定をしておく必要がある。読者が使っているPOPサーバが、メールにユニークなIDを割り当てるUILDコマンドをサポートしていれば、設定ファイルで“ProtoKeep=UIDL”を指定しておけばよいだろう。

このコマンドは、最近のPOPサーバならまず間違いなく実装しているが、念のためPOPサーバの110番ポートにtelnetして、UIDLコマンドを実際に実行してみることをお勧めする。

もし、利用しているメールサーバがこのコマンドをサポートしていない場合は、RFC1939やimパッケージ（im-100.tar.gz）に同梱されている「im-100/man/imget.jis」というマニュアルを参照して、適切なオプションを設定する必要がある。

なお、これらの指定はimgetコマンドのコマンドラインオプション（imget --helpでヘルプを表示）でも、

```
$ imget -k=7 -p=UIDL
```

とすれば、指定できる。

このように、サーバにメールを残す設定にしておけば、RFC1939の［8. 規模と運用］の項でも考察されているように、IMAP4の機能のひとつである「サーバに保存」という目的だけは簡易ながら実現することができる。したがって、どこからでもメールを読めるようにするという目的は、とりあえず達成できることになる。

しかし、メールスプールを5Mバイト程度で制限しているプロバイダは多いし、容量制限がないプロバイダでも、数週間たったメールは消してしまうなどの処置をしていることが多い。このため、送られてくるメール数が非常に多いユーザーは、すぐにメールスプールがあふれてしまう。

また、多数のメールを扱う場合には、複数のメールボックス（ディレクトリ）に振り分けて管理しないと破綻してしまうなど、問題点もあることは留意しておいてほしい。

IMAP4サーバを独自に立てる

最近話題になっているIMAP4（厳密にはIMAP4rev1）だが、これはクライアント側のローカルディスクにメールを保存・管理させるPOP3と異なり、サーバ上でメールを管理することをメインに考えられたプロトコルである。

このプロトコルでは新規メールのチェックや、多数のメールを管理しやすいように、サーバ上にメールボックスを作成したり、その削除や名前変更ができるようになっている。さらに、サーバ上での検索や未読管理など、さまざまな機能を実現することができる。

しかし、最近ではEmacs上で動作するWanderlust（http://www.gohome.org/wl/）など、IMAP4で使うことを重要視するクライアントも出現してきてはいるものの、まだまだ一般的とは言いがたいようである。

一方、メールサーバを管理する立場としても、莫大なディスク領域を管理しなければならないため、たいへんである。顧客のメールをなくすことは信用問題に関わるので、高価な記憶装置を導入する必要があるし、数千人を抱えるサーバでは検索などのためのCPUパワーも必要になってくる。

このような事情があるため、今のところIMAP4を採用しているプロバイダは多くないのが現状だ。しかし、企業や学校などで、どの端末からもメールを読めるようにしたり、会議室に持ち込んだノートPCでメールを読めるようにするという組織内だけでの利用を考えると、IMAP4という解決策も悪くないかもしれない。バックアップなど、サーバ管理に自信があるなら、IMAP4サーバを立ててみるのもよいだろう。大規模に実施せずに個人ベースでも、会社のPOPサーバから自分で立てたIMAP4サーバにメールをPOPしておけば、そのIMAP4サーバにアクセスすることで、IMAP4の機能を体感することができる。

なお、IMAP4関連の情報については、Linux magazine No.3の207ページからの「**IMAP4（前編 サーバーのインストール）**」で、Wanderlustを始めとするIMAP4クライアントに関する情報は、Linux magazine 10月号の213ページからの「**IMAP4（後編 クライアントの設定）**」に詳しく解説されているので、興味を持った方はそちらも参照してほしい。

なお、Linux magazine No.3の記事については、http://www.ascii.co.jp/linux/allascii/linuxmag/no_03.htmlからPDFファイルがダウンロード可能なので、記事の参照に利用してほしい。

メールの一元化

POP時にメールをサーバに残しながら、同時に.forwardなどを用いてプロバイダにメールを自動転送する設定にしている読者も多いだろう。ところがこのような場合、メールは2か所に保存

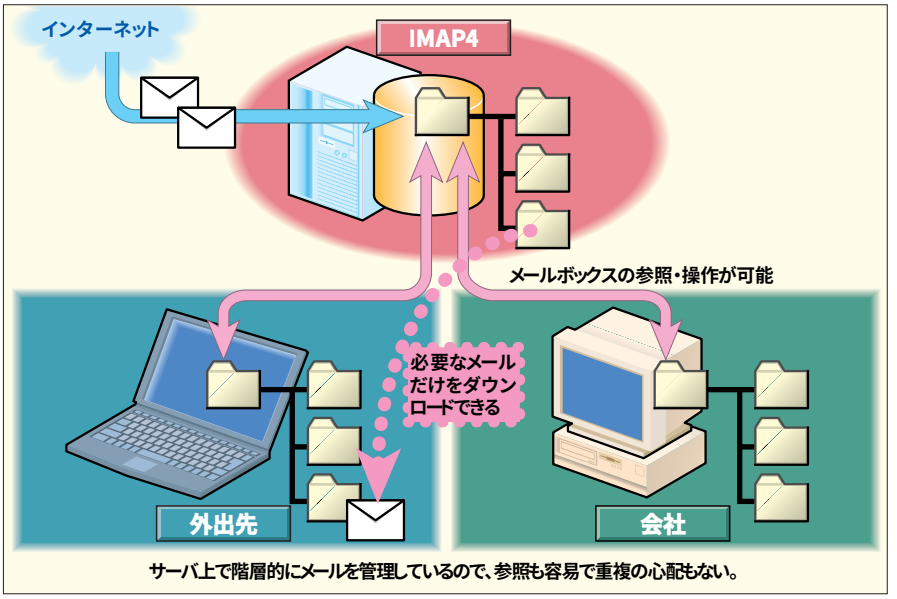


図3-2　IMAP4による複数クライアントアクセス

されることになるため、すでに読んだメールを何度も振り分けるといった無駄な作業をすることになる。

メールの流量が少ない場合は大きな問題ではないが、流量が多くなってくると管理しきれなくなってくるので、メールを一元管理することが必要になってくる。具体的には会社や学校宛てのメールをサーバに残さず、プロバイダにすべて自動転送しておいて、PHSなどを使ったモバイル環境から読めるようにすればよい。こうしておけば、すべてのメールがノートPCだけに保存されるようになるので、メールの一元管理が可能になる。

この方法はとても単純であり、あまり重要なメールを扱うのでなければ、すぐに実現できる簡単な方法である。ただし、流量が多かったり、重要なメールを扱うようなら、実施の前に次のような問題を解決しておく必要がある。

■バックアップ

仕事で利用しているメールは、それ自体が重要書類であり、顧客情報でもある。紛失は業務に支障をきたす。また、プライベートなメールであっても、大切なことには変わりない。

そのため、破壊や盗難に備えたバックアップが最重要の課題となる。これについては、次節でいろいろな方法を紹介しているので、参照してほしい。

■ファイアウォールの存在

次の問題は、どのようにして会社や学校などの環境から、プロバイダにあるメールを読むかということである。多くの環境はファイアウォールがあるため、直接プロバイダのメールを読むことができないはずである。

簡単なのは、PHSなどでPPP接続して読むことである。この場合、私用のメールを昼休みに読むような用途でも会社の資源をまったく使わないし、会社に盗聴されるような心配もないので、まったく問題はない。

しかし、会社宛てに来た業務用のメールを一元管理するためにプロバイダに転送している場合は、いちいちPHSで接続するというのは面倒だし、不経済である。

このような場合は、Linux magazineのProgrammingのコーナーで10月号から2回にわたって、ファイアウォールを越えてプロバイダにあるメールを読む、という方法を解説しているので、そちらを参照してほしい。

そのほかにも、ファイアウォールの形態によっては、imgetコマンドをtelnetプロキシ経由で使えるように改造したりするなどの方法もあるだろう。

■暗号化フォワード

会社宛てに来たメールを、暗号化しないでプロバイダへ送ることが、セキュリティの観点から問題になるケースもあるだろう。

このようなときは、自動的に暗号化を行ってからインターネットに送信し、ノートPCで受信してから自動的に複合化するという方法がある。

この方法についても、Linux magazine.No.2の137ページからの「**GPG ―自動暗号化システムをつくる―**」で、GPGによる自動暗号化と、MewやWindows上のAL-Mailなどを使った自動複合化の例を紹介しているので、興味がある方は参照してほしい（ダウンロード可能なPDFファイルが、http://www.ascii.co.jp/linux/allascii/linuxmag/no_02.htmlにある）。

メールの受信

次に、メールの受信について考えてみよう。前述のように、MewではPOPサーバからの受信にimgetコマンドを利用している（Mewでiコマンドを使うと、imgetコマンドが呼ばれ、メールが取り込まれる）。したがって、Mewにおける POPサーバの指定は、

```
~/.im/Config
```

で指定するようになっている。Mewは、ネットワーク環境を切り替えて使うことが考慮されており、このファイルに、

```
case PHS
Imget.Src=pop/apop:xxx@phs.ne.jp
case JOB
Imget.Src=pop/apop:yyy@job.co.jp
case HOBBY
Imget.Src=pop/pop:zzz@hobby.gr.jp
```

といった記述をしておけば、MewのSummaryモード時にCコマンドで、

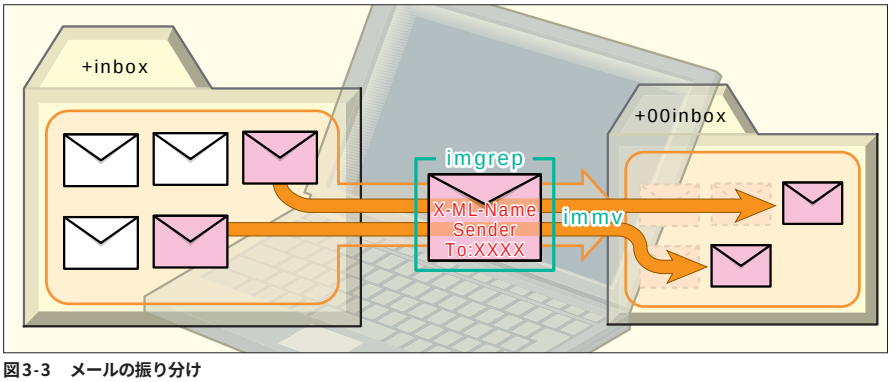


図3-3　メールの振り分け

POPサーバを切り替えることができる。これなら、シェルスクリプトなどの難しい知識がなくても環境を切り替えることができるので簡単だろう。

しかし、あちこちに移動するヘビーなモバイルユーザーの場合は、これだけではすまないこともある。前節の「ネットワーク環境の切り替え」でも述べたように、ノートPCの移動によって変更が必要なネットワーク設定は、POPサーバ以外にも数多くある。

したがって、設定ファイル自体をシンボリックリンクなどで書き換え、他のネットワーク設定といっしょに切り替えるようなシェルスクリプトを用意しておかないと、とても面倒な作業となってしまう。

メールの振り分け

モバイル環境でのメールの扱いというと、送受信の方法ばかりに目がいきがちだが、メールの読み方についても工夫がほしい。

つまり、処理時間に応じてメールを選択して読めるようにするのだ。具体的には、受信したメールを優先度や目的ごとに複数フォルダに振り分けておき、移動時のように処理時間が多くとれないときには、重要なメールだけを読めるようにしておけば、効率的であろう。

これは、imパッケージにあるimgrepコマンドとimmvコマンドを使えば実現することができる。

imgrepコマンドは、メールが保存してあるディレクトリから条件に合うものを検索（grep）し、条件にマッチしたメールの番号を表示するコマンドである。また、immvコマンドは指定したメールを、指定したディレクトリへ移動（mv）する。

たとえば**リスト3-5**のように、これらのコマンドをパイプで組み合わせると、+inboxにあるメールから、あとで読むメールを見つけだし、そのメールを+00inboxディレクトリへ移動することができる。

この例では、メールヘッダにある“X-ML-Name:”、“Sender:”、“To:”などの文字列から優先度の低いメーリングリスト宛てのメールと判断し、そのメールを移動することにしている。

ちなみに、xargsコマンドでは、標準入力で受け取ったリスト、つまりimgrepコマンドで見つけ出したメール番号を引数としてimmvコマンドを実行している（**図3-3**）。

なお、独自ドメインのレンタルサーバを借りていて、無制限にユーザーを作成できるような場合は、各メーリングリストを専門に受信するユーザーを作成しておくと都合がよいことが多い。こうしておけば、流量の多いメーリングリストなども選択的に受信できるので、（PHSの電波が届く）新幹線の停車時のような、わずかな時間にも重要なメールだけを取り込むことができるようになる。

オフラインでのメール作成

Mewが送信に使うimputコマンドの場合、~/.im/Config ファイルに、

```
JustQueuing=yes
```

と設定しておくと、メーラ上で送信を行っても、すぐにネットワークには送信されないで、「キュー」と呼ばれるディレクトリ（~/.im/queue/）に、1通ずつ（番号をファイル名として持つ）保存される。

ここに保存されたメールは、PPPで接続が確立している状態で、

```
$ imput -q
```

というコマンドを実行すれば、まとめ

▼リスト3-5　inbox-refile

```
#!/bin/sh
   imgrep -e "To=ylug@      ||
   X-ML-Name=janog          ||
   Sender=owner-twain       ||
   X-ML-Name=samba-jp"      \
|  xargs immv --dst=+00inbox
```

▼リスト3-6　自動メール送受信スクリプト

```
#!/bin/sh
ppp-on || { echo "ppp fail" ; exit 1; }
echo -n START: `date '+%Y-%m-%d %H:%M.%S'` >> /var/log/modem.mony
USER=fujisawa
MACHINE=10.11.12.13
while true ; do
        if ping -c 1 $MACHINE ; then
                break;
        fi
done
cd /home/$USER/.im/queue
if [ `ls -1 | grep '[0-9][0-9]*' | wc -l ` != 0 ]; then
        echo "===> imput -q"
        su - $USER -c "imput -q"
fi
#-------------------------------------------------------------
su - $USER <<EOF
imget .....
imget .....
imget .....
EOF
#-------------------------------------------------------------
ppp-off
echo -n END: `date '+%Y-%m-%d %H:%M.%S'` >> /var/log/modem.mony
```

て送信されるようになっている。

なお、このimputコマンドは、Mewから送信コマンド（Ctrl＋Cを2回押す）を実行して、キューに入れた時点でConfigファイルで指定されていたメールサーバに送るようになっているので注意が必要だ。

これは、最近はSPAMメールの防止のため、自分の顧客からしかSMTP接続を受けつけないプロバイダがほとんどであり、キューに入れる時点と実際に接続した時点で、Configファイルの SMTPサーバが違う場合は問題となってしまうからである。SMTPサーバを複数切り替えて使うようなパワーモバイルユーザーは気をつけてほしい。

なお、Configファイルの設定を気にしなくてもすむようにするには、送信に使われるimputコマンドが使用しているPop.pmファイルを改造し、SMTPサーバを自動的に選べるようにすることもできる。ここでは詳しく説明しないが、腕に自身のある人はトライしてみてもよいだろう。

自動送受信

外出先からPHSを使ってメールを読む場合は、さまざまな設定変更が必要になる。IPアドレスやDNSだけでなく、移動先によってアクセスポイントも切り替えなくてはならない。

そして、PPP接続を確立するコマンドを実行し、接続が確立したことを待ってから、imgetコマンドを実行する必要がある。この作業は簡単なように見えて、外出先で何度も行うのはかなり面倒な作業である。

しかしPC-UNIXなら、シェルスクリプトを書くことにより、この作業のほとんどを自動化できる。**リスト3-6**がその例である。

このスクリプトでは、最初に「ppp-on」スクリプトを実行して電話をかけ、接続が成功したら、スタート時間をログに記録し、メールサーバなどにpingコマンドを実行し、返事が帰ってくるのを待つ。キューに送信するメールがある場合は“imput -q”を実行して、オフラインで書いておいたメールを一気に送信している。

そして、imgetコマンドを実行してから、「ppp-off」を呼び、自動的に接続を切る。ログに終了時間を記録しておけば、最初に記録した時間との差を求めることで、課金情報を調べることもできるだろう。

Column

複数サーバからのメールの取り込み

複数のPOPサーバからメールを一気に取り込たい場合は、**リストc-1**のようなスクリプトを書けば可能である。このリストでは、読む優先度が高いメールを+inboxディレクトリへ保存している。

また、FreeBSDやLinuxのユーザーグループのメーリングリストのように流量が多すぎてすぐに読むことができないメールは、新幹線の移動時など暇なときに読めるよう、メールを、+00freebsdと+00linuxに振り分けている。

なお、ノートPC自身のroot宛てのログを読むためには、/etc/aliasesでroot宛てのメールを自分のメールアドレス宛てに送るように設定し、**リストc-2**の形式で指定するとよいだろう。

さらに、imgetはPOPサーバだけでなく、NNTPサーバなどからのメール読み込みもできるので、$HOME/.im-newsに、

```
fj.os.bsd.freebsd
fj.os.linux
```

というような記述をしておき、**リストc-3**のようなスクリプトで一気にノートPCに取り込めば、電車の中などでもMewを使ってNetNewsを読むことができる。

なお、Mewを立ち上げたままでimgetコマンドを実行すると、+inboxにメールが届いても、Mewのサマリーモードが更新されない。この場合は、sコマンドで“all”を指定すると、最新の状態が表示されるようになる。

▼リストc-1　inc-provider

```
#!/bin/sh
imget -q --src="pop/APOP:linux-mag@fujisawa.gr.jp"     --dst="+inbox"
imget -q --src="pop/APOP:freebsd-users@fujisawa.gr.jp" --dst="+00freebsd"
imget -q --src="pop/APOP:linux-users@fujisawa.gr.jp"   --dst="+00linux"
```

▼リストc-2　inc-local

```
imget -q --src="local:/var/mail/fujisawa"              --dst="+inbox"
```

▼リストc-3　inc-news

```
SRV=news.xxx.ne.jp
for GRP in `cat $HOME/.im-news`
do
        DIR=00`echo $GRP | sed -e 's,\.,/,g' `
        echo "===> $GRP"
        mkdir -p $HOME/Mail/$DIR
        imget -q --src="nntp:$GRP@$SRV" --dst="+$DIR"
done
```

ノートPCを使う場合のデータ同期

ノートPCとデスクトップマシンの両方を利用していると、それぞれのマシンにあるファイルの同期が問題になってくる。特に、同じファイルを両方のマシンで変更してしまった場合は、その整合性をとるのに非常に苦労することになる。

また、ノートPCは落として壊してしまうこともあるので、バックアップをいかに完璧にとるかということも重要である。

ここでは、まずrsyncによる簡単で手軽な同期方法を紹介し、そのあとでより高度なファイルの一元化について説明する。そしてバックアップ方法や、デスクトップからの使用方法についても紹介してみようと思う。

ファイルの同期

UNIXのシステムどうしでファイルの同期をとるには、rsyncコマンドが便利で高速である。このコマンドは、それぞれのマシンにあるファイルを数百バイトのブロックごとに32ビットのローリングチェックサムで計算し、そのチェックサム情報をネットワークを通して交換し、照合する。そして、差異があった部分だけをネットワークを通して送るという非常に凝ったアルゴリズムを用いている（ちなみに、このアルゴリズムの論文には、Linuxカーネルソースを同期させた結果が例として掲載されている）。

チェックサムが偶然に一致するファイルの同期の場合も考えると、まったく問題がないとはいえないが、通常のファイルではまず問題になることはないし、転送量が少なく高速であるなど、メリットは非常に大きい。

rsyncには非常に多くのオプションがあるが、deskという名前のデスクトップPCとnoteという名前の、ノートPCの同期を簡単に行うには、どちらかのマシンで、

```
# rsync -avub desk:/xx/ note:/xx/
# rsync -avub note:/xx/ dese:/xx/
```

という2行のコマンドを実行すれば、それぞれのマシンの/xxディレクトリの新しいファイルがそれぞれコピーされ、同期が行われる。なお、ここでは-bオプションをつけているので、古いファイルは「~」（チルダ）を後ろにつけたファイル名にリネームされる。

ファイルの一元管理

簡単にデータの同期をとるだけなら、rsyncを用いた上記のやり方でもよいのだが、削除したファイルが元に戻ってしまうといった問題がある。

また、日付情報だけで同期をとろうとすると、変更してほしくないものまで同期してしまい、古い内容に戻ってしまったりすることもあり得る。これを解決するには、はじめからデータを分散させないで、すべてのデータ（/home）をノートPCに置いて使うのが、一番単純かつ確実であり、面倒がない。

しかし、ノートPCのCPUは一般に非力であり、ノートPCの液晶はデスクトップマシンの高解像度CRTと比べるとかなり見づらい。また、キーボードやマウスもやはりノートPCについているものでは入力しづらく、ノートPCだけで作業を行うことは効率が悪い。

そこで、デスクトップマシンを使う場合はノートPC上にある「/home」をNFSマウントするとよいだろう。このようにすれば、データの同期を考えることなく、デスクトップマシンの優れた環境（表示・入力・演算）を使うことができるわけである。

しかし、これを実行するには次の点について十分考慮しておく必要がある。

■バックアップ

自分のすべてのデータが入っている「/home」をノートPC上に置くとなると、紛失や破壊に備え、定期的にバックアップする必要がある。しかし、手動でのバックアップは面倒なものであり、ついついバックアップをサボってしまいがちである。しかも、そういうときに限ってデータが壊れてしまうものである。したがって、「/home」を持ち歩くためには、自動的にバックアップを行うシステムにすることが絶対必要といってもよいだろう。

「/home」をサーバマシンにバックアップするにも、rsyncコマンドが便利である。1Gバイトの「/home」を10Mbpsのネットワーク上でバックアップするような場合でも、変更が少なければ数分で終わる。

たとえば、**リスト3-7**に示すような内容を/etc/crontabに書き、1時間ごとに**リスト3-8**のようなスクリプトを呼び出すとよいだろう。

この例では、スクリプトにバックアップすべきサーバ名を引数としてわたし、そのサーバがネットワーク上に存在すれば、ノートPCの液晶画面にrsyncの実行画面を表示させ、実行ログをrootへメールで送るように設定してある。 つまり、ノートが会社にあるときは、会社のサーバにデータがバッ

クアップされ、自宅のネットワークに接続しているときは、自宅のサーバにバックアップされることになる。

なお**リスト3-8**では、" --delete "を指定しているので、ノートPC上に存在しなくなったファイルは自動的に削除される。

デスクトップPCのX端末化

デスクトップPCよりもノートPCのCPUのほうが高速な場合は、すべての演算をノートPC上で行い、キーボードと表示だけをデスクトップマシンのものを利用するほうが快適だろう。

そこで、ノートPCの画面だけをデスクトップマシン側に表示させる方法も紹介しておこう。X Window Systemでは、ノートPC上で、

```
note%  DISPLAY=desk-pc.xxxx.co.jp:0.0
note% export DISPLAY
note% netscape
```

というように環境変数DISPLAYをセットすることにより、ノートPC上で動かしたnetscapeの入力と表示をデスクトップマシン（*desk-pc.xxxx.co.jp*）で行うことができるようになる。

ここで、デスクトップマシン上ではノートPCからの表示を許可するために、

```
desktop% xhost +note-pc.xxx.co.jp
```

を実行しておく必要があることに注意しよう。なお、セキュリティ上危険な設定になるが、

```
desktop% xhost +
```

とすると、すべてのホストからの表示を許可することができる。このxhostコマンドによる指定は、ホームディレクトリの ~/.xinitrc などに記述しておくとよいだろう。

また、xonコマンドという便利なものもある。xonは、rshを利用して環境変数DISPLAYやxhostの設定などを自動で設定してくれる。ただし、rshの実行許可が必要になるので、あらかじめリモートマシン（ここではノートPC）のホームディレクトリにある.rhostにローカルホスト名（ここではデスクトップPC）を記述しておかなければならない。そして、デスクトップPC上で、

```
desktop% xon note-pc.xxx.co.jp xcalc
```

というように使うと、ノートPC上でxcalcコマンドが実行され、その表示がデスクトップPC上に行われるというものである。

ちなみに、このxonを使うとウインドウマネージャをノートPCで動作させることもできる。たとえば、ウインドウマネージャとしてfvwmを使っている場合は、デスクトップ上で、

```
desktop% ps ax | grep fvwm
 5038  v0  S     0:00.13 fvwm
desktop% kill 5038
```

として、fvwmを終了させ、

```
desktop%␣xon␣note-pc.xxx.co.jp
fvwm95 &
```

とすることにより、ノートPC上のfvwm95を動作させることができる。この場合は、デスクトップPCは純粋に表示と入力のみを受け持つことになり、あたかもノートPCをそのまま使っているのと同じ操作感が得られるだろう。

▼リスト3-7　定期的にバックアップさせる/etc/crontab

```
0  * * * * root /root/bin/home-sync xxx.my-corp.co.jp
0  * * * * root /root/bin/home-sync xxx.my-home.gr.jp
```

▼リスト3-8　rsyncによるデータ同期

```
#!/bin/sh
host=$1
[ "$host" = "" ] && { echo ERROR ; exit 1; }

PATH=/usr/local/bin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin
LOCK=/tmp/.home-rsync-lock
export PATH LOCK

ping -c 1 $host > /dev/null || exit 0

xterm -display :0.0 -T rsync -n rsync -geometry 80x8-0+0 \
-e sh -c "
 {
   {
     echo ====== `date` =====
     [ -e  $LOCK ] && { echo $LOCK EXIST ; exit 1; }
     touch $LOCK
       rsync -aHv --delete /home/ $host:/00home/
     rm    $LOCK
   } 2>&1
 } | tee /dev/tty | mail -s 'home-rsync' root
 sleep 10
" &
```

番外編 またバカやっちゃいました

古いノートとLinux

文：のりぞう
Text：Norizoh

あなたも「Linuxにマシンパワーはいらない」と聞いたことがあるだろう。それは本当だろうか。確かに、486マシンでLinuxをバリバリに使っている人もいる。しかし、原型をとどめないほどに拡張できるデスクトップマシンとは違い、拡張性の低いノートPCは古くなるとあっけなくお払い箱になることが多い。そんなことでいいのか？ちっともエコじゃない。エコロジストでもないのにそういいたくなる（貧乏性なだけ？）。

Linuxは見捨てられたノートPCを復活させられるのか。消費文化はびこる世紀末の日本において、人類の未来のため、のりぞうが挑戦する（大げさ）。

●ターゲットにした386SXマシン

ターゲットマシンは、DOS/V華やかなりし頃の名機、IBM PS/55 note N23SXだ。学生時代、飯も食わずに貯金して、なんとか購入したこのマシンも、今では手のひらにのるマシンよりも非力になってしまった（生協特価で20万円もしたのに）。しかし、直線を基調とした秀逸なデザインは、後のThinkPadシリーズに受け継がれ、現在にいたっている（**写真1**）。

CPU	Intel 80386SX 16MHz
メモリ	6Mバイト
画面解像度	640×480ドット
画面表示色数	16階調モノクロ
ハードディスク	80Mバイト
フロッピーディスク	1台
CD-ROM	なし

表1　PS/55 note N23SXのスペック

マシンスペックは**表1**にまとめた。80386SXとは、インテル初のx86系32ビットCPUである80386の外部データバスを16ビットにした廉価版だ。このマシンは、標準で2Mバイトのメモリ、40Mバイトのハードディスクを搭載しているが、メモリをフル実装の6Mバイトに増設し、ハードディスクも大容量80Mバイト（！）のものに交換した。モノクロ画面だが、Windows 3.1が動作し、ソリティアをするのには十分だった。ポインティングデバイスは搭載していないので、手の中におさまるサイズの小型トラックボール「ミニ・ポインター」を接続して使う（**写真2**）。最近、秋葉原などで売られている「指マウス」の原型のようなものだ。もちろん、PS/2マウスを利用することも可能だ（っていうか普通そうする）。

DOSで動作するVGAベンチマークプログラム3DBENCH.EXEでは4.5をマークした。ちなみに、MATROX Mystiqueを搭載したPentium（P54C）166MHzのマシンでは32倍ほど速い142.8だった。

写真1　往年の名機 IBM PS/55 note N23SX

●ディストリビューションを選ぶ

さて、Linuxをインストールしようと思い、市販のディストリビューションを調べてガク然とした。要求されるマシンスペックがメチャ高いのだ。日本語 redhat Linux 5.2、LASER5 Linux 6.0ではi486以上のCPU、16Mバイト以上のメモリ（Xを使うなら32Mバイト以上）、TurboLinux 日本語版 4.xではPentium互換CPU、32Mバイト以上のメモリを必要とする（TurboLinuxは、egcsコンパイラでPentium用に最適化したバイナリを使っているようだ）。

だめだ、PS/55 note君は相手にしてもらえない。ダメモトで、これらのディストリビューションを試してみたが、インストーラさえまともに動かなかった。そもそも、このマシンにはCD-ROMドライブがない。昔のディストリビューションは、十数枚のインストールフロッピーを作り、そこからインストールができたのに……。調べてみると、老舗のディストリビューションであるSlackware 3.6J、Debian GNU/

写真2　本体と別売りの「ミニ・ポインター」（通称「目玉親父」）

写真3　ポケットイーサネットアダプタ D-Link DE-600（のりぞう秘蔵の一品だ）。


写真4　DE-600はパラレルポートに接続するのでPCMCIAスロットは不要だ。


写真5　Compaq Armada 4130T（ドック装着）

Linux 2.1（slink）は、今でもフロッピーインストールができることがわかった。

Slackwareは以前にもインストールしたことがあったので、挑戦してみたが、このマシンではインストーラを起動できなかった。多くのディストリビューションでは、RAMディスクにルートファイルシステムを作り、その上でインストーラを動作させる。ただですら少ないメモリ（6Mバイト）をRAMディスクに取られてしまうと、インストーラ自身を動かせるだけのメモリが確保できないようだ。boot:プロンプトから、ramdisk_sizeオプションで必要最低限の大きさのRAMディスクを作るようにしてもやはりダメだった。必要なドライバだけ組み込んだ小さなカーネルをほかのマシンでビルドし、インストールフロッピーのカーネルを置き換えればメモリを節約できる。ところが、それでもうまくいかない。深追いせずに（？）あきらめよう。

●Debian GNU/Linuxに決定！！

気を取り直してDebianに挑戦だ。フロッピーディスクを何枚も作るのはスマートではないので、フロッピーディスクで起動し、ネットワークからインストールする方法をとろう（実は面倒がイヤなだけ）。そのためにはネットワークインターフェイスが必要だ。普通のノートPCなら、PCMCIAのイーサネットカードを使うのだろうが、PS/55 note君にはPCMCIAスロットもないのだ。ここであきらめてはいけない。世の中には便利なものがある（あった？）。パラレルポートにつなぐイーサネットアダプタだ（**写真3、4**）。このD-Link DE-600は、古くからLinuxでサポートされていることで有名で、ほとんどのディストリビューションでインストールに利用できる。10BASE-2と10BASE-Tに対応しているが、転送速度はかなり遅い。

Debianは、NFS（Network File System）でマウントしたディレクトリからのインストールをサポートしているので、ほかのLinuxマシンのCD-ROMドライブに入れたDebian CD-ROMを、ノートPCからNFSマウントして利用できるのだ。

はたして、インストールは無事に完了した。必要最低限のベースシステムで32Mバイト、スワップパーティションに10Mバイトを使い、利用できるディスクスペースは33Mバイトほどだ。さらにXを入れるには大容量80Mバイトハードディスクでもちょっと厳しい。Xを使うにはCPUパワーも足りないだろう、なにせ、topコマンドを起動するだけで27%もCPU負荷がかかるのだ（Celeron 333MHzのマシンでは0.3%前後だ）。

CPU	Intel Pentium 133MHz
メモリ	32Mバイト
画面解像度	800×600ドット
画面表示色数	65535色
ハードディスク	4Gバイト
フロッピーディスク	1台
CD-ROM	ドックに装備

表2　Armada 4130Tのスペック

●やっぱ、Xは動いてほしいよね

PS/55 note君にLinuxを入れてできることは限りなく少ない。UNIXコマンドの勉強やtelnet端末にするには使えるかもしれないが、それでは寂しすぎる。せめてカラー画面でXが動けばと誰しもが思うだろう。

そこで、今度はカラーのノートを登場させよう。飲み代削って貯金して、一昨年のクリスマスに10万円で購入した中古のCompaq Armadaだ（**写真5**）。標準で16Mバイトのメモリも32Mバイトに増設済みで、別売りのCD-ROM付きドック（**写真6**）もおまけしちゃうというお買い得品だったが、今となっては少々見劣りするスペックだ


写真6　CD-ROMドライブはドックに装備している（ジョイスティックポートもあるが、誰が使うの？）。

(**表2**)。Windows 95でニュースリーダのWinVNを使い、ベッドでネットニュースを読むための専用機として重宝している。しかし、このままではかわいそうだ。Linuxで用途を広げてやろう。

このマシンには、1Gバイトのハードディスクが搭載されていたのだが、今回、(軟弱にも)LinuxとWindowsのデュアルブートにするため、中古の4Gバイトのものを購入した。これは日立製のDK227A-41というモデルで、税別1万5800円だった。換装してみると、体感できるほどディスクアクセスが速くなったのには驚いた(やったね)。

●さて、Linuxを動かそう

ドックのCD-ROMドライブはごく普通のATAPI CD-ROMドライブとして認識され、TurboLinux 日本語版 4.2ノンサポート版があっさりとインストールできた。Xもほぼ問題なく動作した。「ほぼ」というのは、Windows 95を使ったあとには、一度電源を切らないとXFree86がうまく動かないのだ。Windows 95のディスプレイドライバがナニかしているらしい。

GNOMEやKDEは動作が重いのでこのマシンには向かない。かといってfvwm2では味気なさ過ぎる。間を取って、ウィンドウマネージャにはWindowMakerをチョイスしてみた。

ネットワークインターフェイスは、Windows 95でも使っていた3ComのPCMCIAイーサネットカード3c589Cをそのまま使う(**写真7**)。これは、10BASE-2と10BASE-Tに対応。写真を注意深くご覧になった方は気づいたかもしれないが、コネクタの脱着部のツメが折れている。筆者の寝相の悪さがここに反映されているわけだ。

写真7　10BASE-Tに対応するPCMCIAイーサネットカード 3Com 3c589C。

●ベッドで使おうLinuxノート

ネットワークとXが動くようになったらこっちのものだ。Windows歴の長い方は「XってLinuxやUNIXにのっけてGUIを実現するだけじゃん」と思っているかもしれないが、Xは単なるGUIシェルではなく、クライアント/サーバモデルで動いている。これは何を意味するのか。そう、ほかのマシンで動いているプログラムを手もとのマシン上に表示できるのだ(そんなのあたりまえだという方、ごめんなさい)。実際の表示例が**画面1**だ。ほかのマシンで動作していることがわかるように、パフォーマンスメータのxosviewを使った。右下の1個だけがローカルマシンで動作しており、残りはそれぞれ別々のマシンで動いていることがわかるだろう。特に、上の2個はDual CPUのSMPマシンで動作している(CPU0、CPU1に注目)。

ここでは、わかりやすいようにxosviewを例として取り上げたが、Linux初心者に「このノート、Dual CPUなんだぜ」なんて言って驚かせるくらいしか役に立たない。実際には、パワフルなマシンで処理の重いプログラムを動かし、手もとのノートに表示させるのが実用的だろう。

Xでこのように使うためには次のように設定する。まず、手もとのマシンでstartxなどのコマンドを入力してXサーバを立ち上げる。次に、Xのプログラム(Xクライアント)を動作させるマシン(powerfull.hoge.netとする)を登録する。ktermなどで次のように入力しよう。

```
$ xhost +powerfull.hoge.net
```

登録がすんだら、手もとのマシンで動作しているXサーバで表示を行うよう設定する。具体的には、powerfull.hoge.netにtelnetログインして、環境変数DISPLAYをセットする。シェルがbashなら次のようにすればよい。

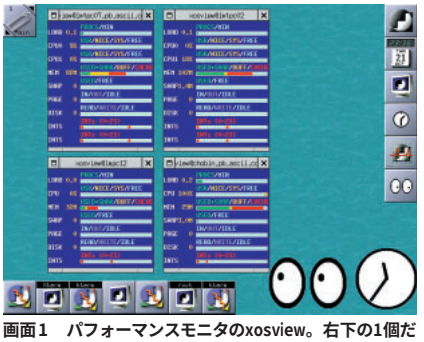
```
$ export DISPLAY=note.hoge.net:0
```

あとはpowerfull.hoge.netで好きなプログラムを実行するだけだ。

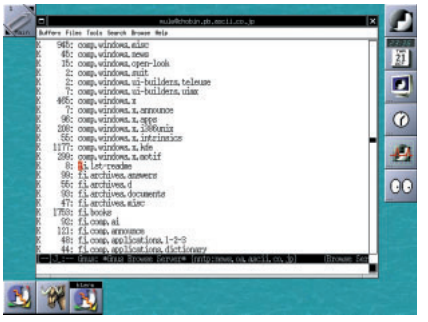
```
$ xclock &
```

これで手もとのマシンに時計が表示されれば成功だ。

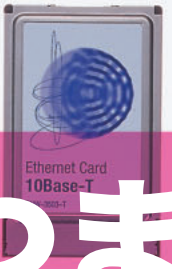
さて、ノートからほかのマシンも使えるし、gnusでネットニュースも読めるし(**画面2**)、これでベッドから出ないで生活できるぞ(できないって)。



画面1　パフォーマンスモニタのxosview。右下の1個だけがこのマシンで実行されている。

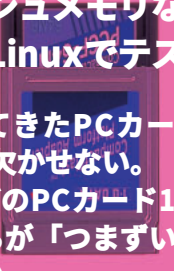


画面2　Muleを起動し、ネットニューズクライアントのgnusを実行した。ノートなら、ごろ寝しながら使うのに最適。



つまずかないPCカード選び

ネットワークやSCSI ,モデム、フラッシュメモリなどPCカード15製品をLinuxでテスト



古くからノートPCのハードウェアを拡張する手段として使われてきたPCカード。LinuxでノートPCを活用するなら、そのPCカードのサポートは決して欠かせない。

本記事では、ネットワークやSCSI、モデム、フラッシュメモリなどのPCカード15製品をLinuxで動作検証し、使いこなす方法を探ってみた。テスターたちが「つまずいた」ところを、読者諸氏にはうまく避けるか踏み越えて進んでいただきたい。

Linux＋pcmcia-csでPCカードを使うには

実装技術の進歩や製造コストの低下などにより、最近のノートPCには、モデムやEthernetはもとより、USBやIEEE1394、携帯電話のインターフェイスまでさまざまなハードウェアが内蔵されるようになってきた。それだけノートPCにオプションを追加してハードウェアを拡張する必要は薄まってきており、PCカードを利用する機会も減ってきているといえる。

とはいっても、それはWindowsなどメーカーサポートのあるOSでの話である。こうした内蔵デバイスはたいてい最新のハードウェアを利用しており、Linux用のドライバ開発が間に合わなくて結局役に立たない、ということも少なくない。その点PCカードなら、たいていのノートPCがそのソケットを装備しており、また数ある製品の中からLinuxで動作するカードを選ぶことができる。つまり、ノートPCをLinuxで使うなら、ハードウェアを拡張するのにPCカードの使いこなしは必要不可欠なのである。

そこで、ここではLinuxでさまざまなPCカードの動作をテストし、その使いこなしのテクニックを探っていく。なお動作検証には、**写真1、2**のノートPCを利用している。

pcmcia-csによるPCカードのサポート

LinuxでPCカードをサポートしているのは、David Hinds氏が作成しているpcmcia-csというソフトウェアである。pcmcia-csには、各種PCカードコントローラのドライバやPCカード自体のドライバ、設定ファイル、制御コマンドなど、PCカードを利用するのに必要なソフトウェアがひと揃い含まれている。配布元は**表1**の通りで、バイナリではなくソースコードで配布されている。もっとも、たいていのディストリビューションには最初からpcmcia-csが組み込まれており、インストール時に指定すれば、コンパイルすることなくその機能を利用できる。

ただし、ディストリビューションによってはパッケージでpcmcia-csのバイナリを配布している場合もある。

■pcmcia-csのバージョン

現在多くのディストリビューションに付属のpcmcia-csはVer.3.0.xである（LASER5 Linux 6.0はVer.3.0.13が組み込まれている）。一方、単体で配布されている最新バージョンはVer.3.1.3である。Ver.3.0からVer.3.1へのバージョンアップでは、Plug and Play BIOS（PnP BIOS）のファンクションを呼び出して、パラレル／シリアルポートなどマザーボード上のデバイスが使用中のシステムリソース[*1]をなるべく正確に判別するためのコードが追加された。もちろんバグフィックスも行われている。

現在、pcmcia-csはkernelに含まれず、独立した追加ソフトウェアとして扱われている。しかし、Linux 2.3.17よりPCカードをサポートするコードがkernelに組み込まれたので、次の安定バージョン（2.4）のkernelでは、PC

写真1
テストに使用した日本アイ・ビー・エムのThinkPad 600E 2645-55J。1998年11月に発表されたA4サイズのノートPCで、PentiumII-300MHzと13.3インチ液晶ディスプレイ、6.4Gバイトハードディスク、DVD-ROMドライブを装備するモデルだ。基本設計は現行のThinkPad 600Eと変わっていない。PCカードコントローラにはTI製PCI-1251を採用し、2ポートのPCカードソケットを装備する。

写真2
テストに使用したもう1台のノートPCである、ぷらっとホームのFLORA220CX for Linux PLAT'HOME Edition。日立製作所のB5サイズノートPCであるFLORA220CXに、ぷらっとホームが日本語redhat Linux 5.2をプレインストールしたモデルである（Windows 98とデュアルブート）。PCカードコントローラはTI製PCI-1211である。

カードの使い方が変わるかもしれない（現時点ではpcmcia-cs 3.0.xのコードをベースにkernelへの組み込みが行われているようなので、使い方はそう変わらないかもしれないが）。

なお本誌付録CD-ROMには、収録時点で最新のpcmcia-cs 3.1.1のほか、Ver.3.0.x系列で最新のVer.3.0.14も収録した（**表1**）。

■サポートされるハードウェア

現在、pcmcia-csでサポートされているPCカードコントローラは表2のとおりだ。これはVer.3.0.13以降でサポートされているものを記している。自分のノートPCに内蔵のPCカードコントローラがサポートされているかどうかは、コラム「Windows 98でPCカードの情報を得る」のようにベンダー名とデバイス名を調べ、この表とつきあわせればよい。

表2にはCardBus*2対応チップも含まれるが、現在CardBusのサポートは実験的とされており、動作するCardBusカードも少な目なので注意したい。

PCカードについては、pcmcia-csに含まれるSUPPORTED.CARDSというテキストファイルでサポート範囲を確かめられる。これには各ジャンルごとに、サポートされているPCカードのほか、実験的なサポートにとどまっているものも記されている。また、David Hinds氏が直接テストしていないドライバは、**表1**のpcmcia-csアーカイブ配布元に集められており、pcmcia-csとは別に配布されている（本誌付録CD-ROMの/PCCard/contribに最新のものを収録）。SUPPORTED.CARDSにないPCカードでも、こうしたドライバで動作する可能性はある。

表2やSUPPORTED.CARDSにないハードウェアでも動作する場合がある。対象のハードウェアが、サポートリストにあるハードウェアとの互換性を持っている場合だ。内蔵チップのベンダーや型番が異なっていても、ソフトウェアレベルでは互換性があることはそれほど珍しくはない。とはいえ、微妙な仕様の違いから、一見動作していても不具合が生じることもあるので注意が必要だ。

逆にSUPPORTED.CARDSに記されているPCカードでも、PCカードコントローラとの組み合わせや、内蔵チップのリビジョンの違いなどによって、正常に動作しない場合もある。SUPPORTED.CARDSの内容は、基本的に目安として考えたほうが無難だろう。

今回は、Ethernetやモデムなど通信デバイスとSCSIやIDEなどストレージデバイスの2種類に大別してテストしている。サポートされているPCカードの詳細については、それぞれ86ページと91ページからの記事を参照していただきたい。

本記事に関してのご注意

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

概要		URL
pcmcia-csのWebページ		http://pcmcia.sourceforge.org/
pcmcia-cs アーカイブ	一次配布元	ftp://sourceforge.org/pcmcia/
	付録CD-ROM	/PCCard/pcmcia-cs-3.x.x.tar.gz
PCMCIA-HOWTO	最新英語版	http://pcmcia.sourceforge.org/ftp/doc/PCMCIA-HOWTO.html
	付録CD-ROM	/PCCard/doc/PCMCIA-HOWTO.html
	日本語版	http://www.linux.or.jp/JF/JFdocs/PCMCIA-HOWTO.html*1
サポートされているカード一覧		http://pcmcia.sourceforge.org/ftp/SUPPORTED.CARDS

表1　アーカイブファイルやドキュメントのありか　＊1　JFのプライマリサイトでのURL。ここのトップページから身近なミラーサイトを選ぶのが望ましい

ベンダ名	コントローラ名
Cirrus Logic	PD6710、PD6720、PD6722、PD6729、PD6730、PD6732、PD6832
Databook	DB86082、DB86082A、DB86084、DB86084A、DB86072、DB86082B
Intel	i82365sl B、C、and DF steps、82092AA
O2Micro	OZ6729、OZ6730、OZ6812*1、OZ6832、OZ6833、OZ6836、OZ6860
Omega Micro	82C092G
SMSC（旧SMC）	SMC34C90
Texas Instruments（TI）	PCI1130、PCI1131、PCI1210、PCI1211*2、PCI1220、PCI1221、PCI1225、PCI1250A、PCI1251A、PCI1251B、PCI1420*2、PCI1450
Vadem	VG465、VG468、VG469
VIA Technologies	VT83C469
VLSI Technologies	82C146、VCF94365
東芝（Toshiba）	ToPIC95、ToPIC97*3
リコー（Ricoh）	RF5C296、RF5C396、RL5C465、RL5C466、RL5C475、RL5C476、RL5C478

表2　pcmcia-csでサポートされているPCカードコントローラの一覧　最近のノートPCは、TIやリコーのコントローラを採用することが多い（東芝はやはり東芝製コントローラを使っている）　＊1　pcmcia-cs 3.1.2にて追加　＊2　pcmcia-cs 3.1.0にて追加　＊3　実験的なサポートにとどまっている

Glossary

＊1　システムリソース
拡張バスに接続されるデバイスが使用する資源（リソース）のうち、IRQ、I/Oポート、DMAチャネル、メモリアドレスの4つをシステムリソースという。

＊2　CardBus
1995年に提唱された、より高速なPCカードの規格。PCIバスの仕様を取り入れており、33ビットデータ幅／33MHzのPCIバスと同程度の性能を発揮できる。またコントローラはISAバスではなくPCIバスに接続される。CardBusカードに対して、従来のカードは16bit PCカードと呼ばれる。

pcmcia-csの制御コマンド

pcmcia-csがインストールされていると、ドライバだけではなくPCカードの動作を制御したり状態を監視するコマンドも利用できる。コマンドラインではcardctlが便利で、**表3**のコマンドとPCカードソケット番号を指定して実行する。pcmcia-csのコンパイル時の設定によっては、**表3**のeject～schemeまで、設定を変更するコマンドはroot権限を持つユーザーにしか利用できないので注意したい。

X Window Systemの環境では、cardinfoというユーティリティでcardctlと同等の機能をGUIから利用できる（**画面3**）。ただし、これは標準ではインストールされず、ソースコードからコンパイルする必要がある。

またpcmcia-cs 3.1.0以降では、PnP BIOSからのシステムリソース情報を参照または更新するユーティリティ（lspnp、setpnp）が使えるようになった。これによりPnP BIOSが管理しているシステムリソース情報をユーザーの目で確認できるので、デバッグ時に役立つだろう。なお、lspnpとsetpnpを利用するには、コンパイル時の設定を変更する必要がある（84ページの**リスト3**を参照）。

制御コマンドとはちょっと違うが、pcmcia-csのサービス自体をコントロールするのは、LASER5 Linuxの場合

```
/etc/rc.d/init.d/pcmcia
```

というスクリプトである。これにstart／stop／restartというオプションを指定すると、それぞれ起動／停止／再起動が行われる。設定変更後の再起動などで使うので憶えておきたい。

pcmcia-csの設定ファイル

pcmcia-csに関わる設定ファイルは複数に分散配置されているので、慣れないとどこで何を設定していいかまごついてしまう。しかし、それは機能ごとに設定ファイルを分けているせいで、それ故にスクリプトによる自動設定や、設定内容の流用などが容易にできるというメリットもある。

■pcmcia起動オプション

表4のように、pcmciaのサービス起動に関係しているファイルのありかは、

Column

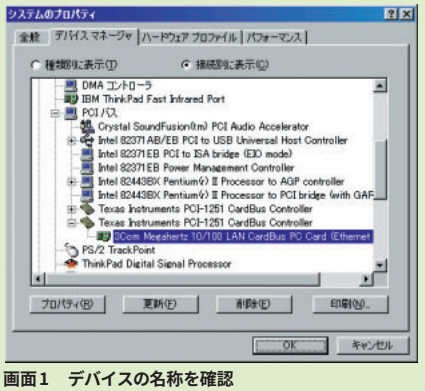
Windows 98でPCカードの情報を得る

Linuxに限らず、PCカードを支障なく使用するには、正しいドライバの選択・組み込みと、システムリソースの衝突防止が欠かせない。そのためには、PCカードやPCカードコントローラを事前に詳しく調べておいたほうがよい。Windows 98を使えばこうしたハードウェアの情報を得るのはそう難しくない。なお、Windows 95 OSR2.5より前のバージョンやWindows NTでは、CardBusがサポートされておらず役に立たないことが多い。できればWindows 98を用意したい。

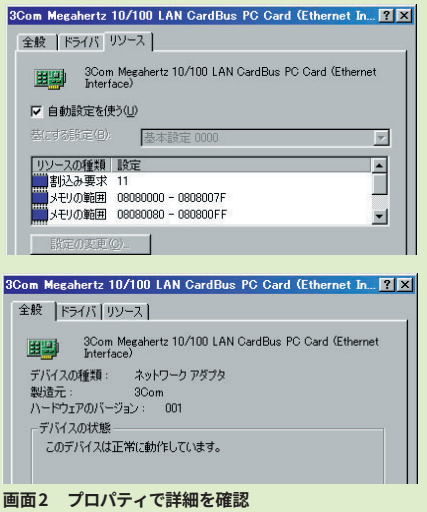
まずチェックしておきたいのは、Linuxで利用するマシンに搭載されたPCカードコントローラの正体である。これは**画面1**のようにデバイスマネージャで確認できる。PCカードについても同様だ。システムリソースについては、デバイスマネージャから該当デバイスのプロパティを開くと参照できる（**画面2上**）。念のために、ハードウェアのバージョンもチェックしておくとなおよい（**画面2下**）。このバージョンによっては、ドライバが動作しない可能性もあるからだ。

さらに詳細な情報を知りたければ、レジストリエディタでWindows 98が自動検出した情報を直接参照するという手がある。特にCardBusカードなら、\\HKEY_LOCAL_MACHINE\Enum\PCIというキーにベンダーIDやデバイスIDなどが保存されている。

PCカードの動作確認にもWindows 98は有効だ。およそPC/AT互換機向けの製品ならWindows 98での動作保証はあるので、まずWindows 98で正しく動作することを確認しておけば、Linuxで動作しなくてもハードウェアの故障を疑わずにすむ。実際、今回の動作検証中にメディアケーブルの断線でEthernetカードが通信できないというトラブルがあったが、Windows 98でのテストにも失敗したことで、ハードウェア的な問題と確信することができた。



画面1　デバイスの名称を確認

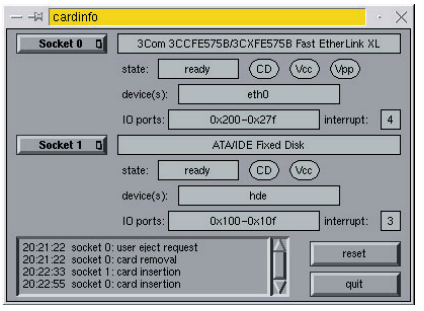


画面2　プロパティで詳細を確認

コマンド	内容
config	ソケットの状態を詳細に表示する
ident	挿入されたPCカードの情報を表示する
status	ソケットの状態を簡易に表示する
eject	ソフトウェア的にPCカードを取り外す
insert	ソフトウェア的にPCカードを挿入する
reset	ソケットをリセットする
resume	ソケットの電源を入れて機能を復活させる
suspend	ソケットの機能を停止して電源を止める
scheme	スキームのリスト表示や切り替えを行う*1

表3　cardctlのコマンド一覧

*1　詳細は86ページからの記事を参照



画面3　cardinfo
「select x」ボタンを押すと、PCカードのイジェクトやリセットなどの操作メニューが表れる。

ディストリビューションによって異なるので注意したい。実際に設定変更で書き換えるのは、オプション設定ファイルのほうである（**リスト1**）。pcmciaのサービスを起動しようとすると、PCカードコントローラのドライバ（i82365.o）とPCカードのメインドライバ（pcmcia_core.o）がロードされ、そしてcardmgrデーモンが実行される。各モジュールに対する設定は、オプション設定ファイルにて個別に行うわけだ。

さて実際に指定すべきオプションとしては、まずi82365に対する「poll_interval=100」が挙げられる。これを指定しないと、PCカードの挿入を検出するためだけにIRQが占有されてしまうのだ。最近の機能満載なノートPCではIRQがすぐ足りなくなるので、このオプションを指定してIRQの空きを増やしておきたい。同じくi82365に対して、確実に空いているIRQを「irq_list=10,11」といった具合に指定しておくとよいだろう。これで指定されたIRQだけがPCカードに割り当てられるので、間違ってほかのデバイスが使用中のIRQを割り当てずに済む。

■config、config.opts

これらのファイルは/etc/pcmciaに配置される。configはPCカードのデバイスドライバを定義しているファイルで、config.optsはPCカードおよびPCカードコントローラに割り当て可能なシステムリソースを指定するのに使われる。また、各PCカードのドライバに渡すオプションもconfig.optsに指定できる。

たとえばIRQ3とIRQ4は、たいてい内蔵のシリアルポートや赤外線ポートで使用中なので、config.opts内で、

```
exclude irq 3
exclude irq 4
```

と記述すれば、PCカードはこれらのIRQを使わない。もちろんこの設定はi82365の起動オプションで指定できる利用可能なIRQのリストと矛盾してはいけない。

PCカードとPC本体の間でシステムリソースの衝突が生じたら、config.optsの設定を変更して解決を図るわけだが、設定内容はPC本体のハードウェア構成やPCカードのドライバなどに深く依存する。そのため一般的な解決法というのはあまりない。まずは、デフォルトのconfig.optsやPCMCIA-HOWTOにあるヒントを参考に、設定を変更していくしかないだろう。

■デバイスクラスごとの設定ファイル

pcmcia-csでは各種のPCカードを8種類のデバイスクラスという単位に分類して管理している。このデバイスクラスごとに、PCカードが抜き差しされるたびに呼び出されて各種処理を行うスクリプトファイルと、オプションを指定するファイルの2つが用意されてい

ディストリビューション系列	スクリプトファイル	オプション設定ファイル
Red Hat Linux	/etc/rc.d/init.d/pcmcia	/etc/sysconfig/pcmcia
Debian GNU/Linux	/etc/init.d/pcmcia	/etc/pcmcia.conf
Slackware	/etc/rc.d/rc.pcmcia	/etc/rc.d/rc.pcmcia

表4　pcmciaのサービス起動に関わるファイルのありか
LASER5 Linux 6.0はRed Hat Linux 6.0と同じ

▼リスト1　pcmcia起動オプションファイルの例

```
PCMCIA=yes
PCIC=i82365
PCIC_OPTS="irq_list=10,11 poll_interval=100"
CORE_OPTS="pc_debug=1"
CARDMGR_OPTS="-v"
SCHEME=
```

- PCMCIA=yes：pcmciaサービスを起動するなら「yes」を指定
- PCIC=i82365：PCカードコントローラの種類を指定する。最近の製品なら間違いなく「i82365」
- PCIC_OPTS="irq_list=10,11 poll_interval=100"：PCカードコントローラのドライバに渡すオプションを設定する。詳細はmanで「i82365」を参照
- CORE_OPTS="pc_debug=1"：PCカードのメインドライバに渡すオプションを設定する。詳細はmanで「pcmcia_core」を参照
- CARDMGR_OPTS="-v"：cardmgrデーモンに渡すオプションを指定する。詳細はmanで「cardmgr」を参照
- SCHEME=：スキーム名を指定（86ページからの記事を参照）

▼リスト2　/var/run/stabの例

```
Socket 0: 3Com 3CCFE575B/3CXFE575B Fast EtherLink XL
0       network 3c575_cb 0      eth0
Socket 1: ATA/IDE Fixed Disk
1       ide     ide_cs0  0      hde      33       0
```

ソケット番号　デバイスクラス　ドライバ名　デバイス数　デバイス名　デバイス番号

▼リスト3　pcmcia-csのアップデート

```
# /etc/rc.d/init.d/pcmcia stop                                          ①
# cd /etc/pcmcia ; tar zvf ~/backup_pcmcia.tar.gz -c .                  ②
# rpm -qa | grep pcmcia                                                 ③
kernel-pcmcia-cs-2.2.5-rh60_L5_2
# rpm -e kernel-pcmcia-cs-2.2.5-rh60_L5_2
# mount -t iso9660 /dev/cdrom /mnt/cdrom
# cd /usr/src                                                           ④
# tar zxvf /mnt/cdrom/PCCard/XForms/linux-i386/elf/bxform-089-glibc2.1.tgz
# cd xforms ; make install
# cd /usr/src
# tar zxvf /mnt/cdrom/PCCard/pcmcia-cs-3.1.1.tar.gz                     ⑤
# cd pcmcia-cs-3.1.1 ; make config

    -------- Linux PCMCIA Configuration Script --------

Linux source directory [/usr/src/linux]:

Alternate target install directory []:
  Module install directory [/lib/modules/2.2.5-rh60_L5_2]:
Compiler flags for debugging []:
Build 'trusting' versions of card utilities (y/n) [n]: y              ⑥
Include 32-bit (CardBus) card support (y/n) [y]: y                    ⑦
Include PnP BIOS resource checking (y/n) [y]: y                       ⑧

How would you like to set kernel-specific options?
    1 - Read from the currently running kernel
    2 - Read from the Linux source tree
    3 - Set each option by hand (experts only!)
Enter option (1-3) [1]: 1                                             ⑨

Extracting kernel symbol versions...                                  ⑩
Kernel configuration options:
    Symmetric multiprocessing support is enabled.
    PCI BIOS support is enabled.
    Advanced Power Management (APM) support is Enabled.
    SCSI support is enabled.
    Networking support is enabled.
     Radio network interface support is enabled.
     Token Ring device support is enabled.
    Module version checking is enabled.
    /proc filesystem support is enabled.

X Windows include files found.                                        ⑪
/usr/lib/libforms.so and /usr/include/forms.h found.

# make all ; make install                                             ⑫
# cd ~/backup ; tar zvf backup_pcmcia.tar.gz -x network               ⑬
# cd /etc/pcmcia ; mv network network.org ; mv ~/backup/network .
```

る。scsiデバイスを例にとると、前者がscsi、後者がscsi.optsといった具合だ。これらも/etc/pcmciaに格納されている。

デバイスクラスごとの設定ファイルを用いれば、PCカード装着時にさまざまな処理を自動実行できる。たとえばCD-ROMドライブを接続したSCSIカードをソケットに挿入した際、もしCD-ROMメディアが装着されていたら自動的にマウントする、といった自動処理が可能だ。

当然だが設定方法はデバイスクラスごとに大きく異なる。詳細は86ページからの各記事を参照していただきたい。また設定上のヒントがPCMCIA-HOWTOの4章に数多く記載されている。

■stab

/var/run/にあるこのファイルには、各ソケットごとにデバイスの最新情報が**リスト2**のように記されている。cardctlコマンドで得られるのが、PCカードから直接得られるハードウェア情報であるのに対し、stabにはドライバ名やデバイスクラスなど、pcmcia-csにおけるデバイスの管理情報ともいうべきものが記録される。トラブルが生じたら、stabとcardctlコマンドの両方をチェックすればよい。

pcmcia-csのアップデート

pcmcia-csがバージョンアップされるペースは速く、特に1999年10月下旬のわずか10日間ですでに2回のバージョンアップがなされているくらいだ。もちろんPCカードのサポート範囲は広がり、バグの修正も進んでいるはずなので、トラブルが生じているなら最新のpcmcia-csをインストールしてみる手も

ある。またコンパイルという点では、pcmcia-csに含まれないドライバをインストールする場合にも再コンパイルが必要だ。手順を憶えておいて損はない。

ここでは付録CD-ROMに収録のpcmcia-cs 3.1.1をLASER5 Linux 6.0にインストールしてみる。以下、**リスト3**と照らし合わせながら読み進めていただきたい。また全作業はroot権限で行う。kernelのソースコードも展開しておく必要がある。

■インストール前の準備

もしpcmcia-csを利用中なら、pcmciaのサービスを停止し（①）、/etc/pcmciaの下をバックアップしておく（②）。特にLASER5 Linuxを含むRed Hat Linuxでは、/etc/pcmcia/networkというスクリプトファイルがpcmcia-cs標準と異なるため、消してしまうと後で支障をきたす場合がある（詳細は86ページからの記事を参照）。オプション設定ファイルをカスタマイズしている場合もあるので、バックアップは重要だ。

またRed Hat Linux系のディストリビューションをインストールすると、pcmcia-csはRPMで管理されているため、RPMでいったん削除してしまったほうが整合性がとれる（③）。単体で配布されているpcmcia-csをインストールしたなら、特にアンインストールする必要はない。

■XFormsのインストール

必要なアーカイブファイルはpcmcia-csだけではない。XFormsと呼ばれるX Window System用GUI Tool kitのライブラリファイルをインストールしておく必要がある。さもないとcardinfoというXで動くユーティリティがコンパイルされない。付録CD-ROMには、異なるプロセッサやglibcのバージョンごとに対応のライブラリを収録したので、READMEを参考にしながら適切なアーカイブファイルを選んでいただきたい。ここではLASER5 Linux 6.0に合わせてglibc 2.1対応のものをインストールしている（④）。

■pcmcia-csのインストール

pcmcia-csのアーカイブファイルを/usr/linuxに展開したら（⑤）、make configでコンパイル時の設定をする。ここで、root権限を持たないユーザーがPCカードの状態を変更できるようにするには、⑥で「y」と答える。またCardBusを使うなら⑦も「y」と答える。⑧はシステムリソースの確認にPnP BIOSを利用するオプションだが、ここでは「y」と答えた。まだ実装されて間もない機能なので、もし問題が生じたら、「n」としてリコンパイルすればよいだろう。

⑨ではkernelの設定を引き出す元を指定する。通常は1でよいだろう。⑩ではkernelの設定を確認できる。たとえばSCSIカードを使う予定なのに、SCSI supportがdisabledと表示されたら、kernelのリコンフィグもしなければならない。また、前述のXFormsライブラリが正しくインストールされていれば、⑪のようにその旨検出されるはずだ。

あとは実際にコンパイルしてモジュールを所定のディレクトリにインストールする（⑫）。最後に、事前にバックアップしておいた/etc/pcmcia/networkに差し替えれば（⑬）アップデート作業は完了である。

Column

kernelアップデートの必要性

現在、pcmcia-csはLinux kernelとは別個に配布されているが、だからといってソースコードがkernelから完全に独立しているわけではない。PCカードのドライバの一部は、kernelのソースコードと組み合わせてコンパイルされるからだ。実際、pcmcia-csのアーカイブファイルに含まれるバグリスト（BUGSファイル）には、kernelのバージョンを上げて初めて解決できる問題がいくつか報告されている。最新バージョンが常に理想かといえば決してそうとは限らないが、少なくともアップデートにより解消できるバグは確実に存在する。結局、pcmcia-csだけではなくkernelまでアップデートしなければならない場合はあるはずだ。

本誌付録CD-ROMには、収録時点で最新の安定版kernel 2.2.13を収録している（/kernel/linux-2.2.13.tar.bz2）。これでkernelをアップデートする場合、まず展開にはgzipやtar zxvfではなくbzip2が必要である。pcmcia-csとの関係で重要なのは、kernelをコンパイルする前の設定だ。/usr/src/linuxディレクトリにてmake menuconfigを実行するとさまざまな設定を変更できるが、PCカードで使う予定のハードウェアはすべてサポートするよう設定しなければならない。具体的には、本文で触れたSCSIのほかネットワークやIDE／ATAPI、各種ファイルシステムのドライバなどは有効にしておいたほうがよい。SCSIサポートは有効にしていても、SCSI CD-ROMのサポートは無効だった、といったミスをしないよう注意したい。組み込むべきかどうか分からないドライバなどは、とりあえずローダブルモジュールにしておくのが無難だろう。

kernelをコンパイルする実際の手順については、/usr/src/linux/READMEや本誌第2号Try＆Try（174ページ）を参照していただきたい。

通信デバイス系PCカードをテストする

ここからジャンルごとにPCカードをLinuxでテストし、その使いこなしの方法を探っていく。まずは通信デバイスとして、ネットワーク（Ethernet）カードとモデム／PHS、そしてネットワーク＋モデムのマルチファンクションカードの合計7製品をテストしてみる。

Ethernetカード

Linuxをインストールしたノートでまず必要となるPCカードは、おそらくEthernetカードだろう。インターネット接続はもちろん、他のデスクトップPCと接続してその周辺機器を利用できるなど、幅広く使えるからだ。実際pcmcia-csでサポートされている製品数も、Ethernetは段違いに多い。

ここでは16bit PCカードとCardBusカードそれぞれ2製品ずつ選んで、Linux上での動作確認を行った。テストした4製品のなかで、LASER5 Linux6.0付属のpcmcia-cs 3.0.13で動作確認されているカードのリスト（/usr/doc/kernel-pcmcia-cs-2.2.5/SUPPORTED.CARDSにある）に含まれているのは、コレガ FastEther PCC-TXと 3Com Megahertz 10/100 LAN CradBus PC Cardだけだ。ただ、プラネックスコミュニケーションズENW-3503-TはNS8390互換のコントローラを、コレガ FastEtherII CB-TXはIntel（旧DEC） 21143をコントローラに採用しており、これらはそれぞれpcnet_csドライバ、tulip_cbドライバでサポートされているので動作する可能性が高い。

以下ではまず、LASER5 Linux 6.0でのpcmcia Ethernetデバイスの設定方法を解説する。この方法はLASER5 Linux 6.0をはじめ、Red Hat Linuxをベースとしたディストリビューションでのみ有効である。

次に最新版のpcmcia-csでの設定方法を解説する。ノートPCでは複数のネットワーク接続先がある場合が珍しくない。pcmcia-csはスキーマという方法でこの問題に対処できる。Red Hat Linux系以外のディストリビューションを利用している場合は、このpcmcia-cs標準での設定方法と同様に設定できるので、こちらを参照していただきたい。

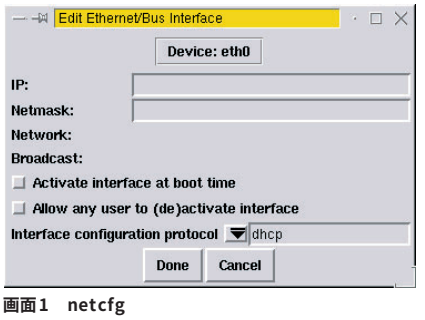
■Red Hat系でのネットワーク設定

ネットワークの設定をする前に、まずはpcmcia-csを適切に設定して、PCカードコントローラを正常に稼働させなければならない。80〜85ページの記事を参考にpcmcia-csの設定を済ませておく。

pcmcia-csが対応しているPCカードの場合、設定が間違っていなければスロットに挿入するだけで、2回連続して高いビープ音が鳴り、認識されるはずだ。対応しているカードなのに、1回目あるいは2回目に低いビープ音が聞こえる場合は、システムリソースの衝突などといったトラブルが発生しているはずだ。まずpcmcia-csの設定が正しいか確認する必要がある。

この段階ではまだethernetデバイス（eth0）の設定をしていないので、

```
# ifconfig eth0
```



画面1　netcfg
「Interfaces」→「Edit Ethernet/Bus Interface」を開いたところ。ここでIPアドレスを設定する。DHCPを用いるため、「Interface configuration protocol」をdhcpに変更している

Ethernetカード（16bit PCカード）

プラネックスコミュニケーションズ(株)

ENW-3503-T

3400円（直販）
http://www.planex.co.jp/

16bit PCカードであるENW-3503-Tは、10BASE-Tに対応したEthernetカードである。ENW-3502-Tの後継モデルにあたり、同一のドライバでどちらのモデルでも動作するよう考慮されている。LEDインジケータの見やすいメディアケーブルが付属する。

(株)コレガ

FastEther PCC-TX

6500円（直販価格：6000円）
http://www.corega.co.jp/

FastEther PCC-TXは、D-Link社のDL10019Cというコントローラを採用した10BASE-T／100BASE-TX両対応のEthernetカードである。100BASE-TXでの通信は半二重のみで、10BASE-Tでは全二重／半二重ともに可能である。16bit PCカードである。

と入力しても、eth0の情報は何も表示されないはずだ。Red Hat Linux系のLinuxでは、X Window System上で動作するnetcfgなどの設定ツールで、eth0デバイスに対してIPアドレスを割り当てることができる（TurboLinuxではturbonetcfgが使えるが、キャラクタベースである）。**画面1**はnetcfgのeth0デバイス設定タブを示している。この設定を行うと次のカード挿入からはコンソール上に、

```
Determining IP information for
  eth0... done.
```

と表示され、自動的にネットワークに接続されるようになる。

さて、今回対象としたPCカード4製品に対して、以下のテストを行った。

○pcmcia-csによる認識
○eth0の認識
○10BASE-T／100BASE-TXの認識
○サスペンド／レジューム

その結果、ENW-3503-Tと3Com Megahertz 10/100 LAN CardBus PC Cardはすべてのテストにパスした。コレガFastEther PCC-TXは10BASE-T対応ハブとの接続には成功したが、100BASE-TX対応のハブに対しては接続できなかった。またコレガFastEtherII CB-TXについては、/etc/pcmcia/configファイルにエントリを追加しても、最初のpcmcia-csのレベルで認識されなかった。

Ethernetカード（CardBusカード）

(株)コレガ

FastEtherII CB-TX

8100円（直販価格：7200円）
http://www.corega.co.jp/

FastEtherII CB-TXは10BASE-Tと100BASE-TXに両対応したEthernetカードである。32bitバスインターフェイスであるCardBusに対応しており、100BASE-TX／10BASE-Tのいずれでも全二重、半二重での通信が可能である。

スリーコム ジャパン(株)

3Com Megahertz 10/100 LAN CardBus PC カード

2万5800円
http://www.3com.co.jp/

3Com Megahertz 10/100 LAN CardBus PC カードは、10BASE-T／100BASE-TXに対応したCardBus対応のEthernetカードである。本カードで特徴的なのは、XJACK LANコネクタを採用しており、メディアケーブルの持ち運びが不要なことである。

■トラブルは解消できず…

FastEther PCC-TXについては、さまざまなオプションの指定を試しても100BASE-TX対応ハブを介して接続することはできなかった。ifportコマンドで10BASE-Tに強制したうえで10／100対応スイッチングハブに対して接続してみたが、それでも接続に失敗する。残念ながら10BASE-Tでないと正しく動作しないようだ。

一方のFastEtherII CB-TXは、前述のとおりIntel（旧DEC）21143をコントローラに採用している。ちょうど編集部にあった100BASE-TX対応CardBusカードのTDK LAK-CB100Xにも21143が使われていたので、同じテストをしたところ、こちらは手動設定なしで動作した。そこで原因を探ってみると、 TDKのカードではコントローラのバージョン番号は32だったのに対して、テストしたFastEtherII CB-TXではバージョン番号が65と大きく上がっていたことが判明した。21143はバージョンが異なると、同じドライバでも動作しないことがあるようだ。

この問題を解決するべく、Intel 21143用ドライバのソースコードであるtulip.cを最新の開発版に更新してみた。すると最初の認識には成功するものの、eth0デバイスとしてネットワークに接続することはできなかった。さらにドライバが更新されれば動作する可能性もあるので、ドライバのインストール手順を以下に記しておこう。

まず開発版のtulipドライバは、Linux and the DEC "Tulip" Chip（http://cesdis.gsfc.nasa.gov/linux/dri

▼リスト1　tulip_cb.oのコンパイル方法

```
# gcc -DCARDBUS -DMODULE -D__KERNEL__ -Wall -Wstrict-prototypes -O6 -c tulip.c -o
tulip_cb.o -I/usr/src/pcmcia-cs-3.0.13/include/

# mv /lib/modules/`uname -r`/pcmcia/tulip_cb.o tulip_cb.o.back

# cp tulip_cb.o /lib/modules/`uname -r`/pcmcia/tulip_cb.o

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

tulip.cのあるディレクトリで実行

このパスは環境に合わせて変更する必要あり

念のためpcmciaのサービスを再起動

vers/tulip.html）から入手できる。最新のリリース版はVer.0.91で、pcmcia-cs最新版にも含まれている。開発版の最新バージョンはVer.0.91gであった。いずれもPCIカード用とCardBusカード用を1つのソースコードtulip.cで管理している。

この開発版ドライバをダウンロードして適当な場所におき、**リスト1**のようにコマンドを実行し、tulipドライバを最新版と入れ替えればよい。

なおLASER5 Linux 6.0付属のtulip_cbドライバにはAPM対応に不具合が存在する。もしサスペンド／レジュームなどAPM関連のトラブルが生じているようなら、バージョン0.91以上のtulip.cに入れ替えてみるのもよいだろう。

■スキーマを利用する

スキーマ（scheme）とは、pcmcia-csが用意している各種設定を簡単に切り替えるための機能だ。特にネットワーク設定で威力を発揮する。その具体的な設定方法を紹介しよう。

Red Hat Linux系のOSはpcmcia-cs標準のスクリプトを変更しており、Red Hat Linuxのifupスクリプトを用いてネットワークを初期化できる。ネットワークの接続先が1つだけ、あるいはすべてDHCPでIPアドレスを取得できるなら何の問題もない。しかし固定IPとDHCPといった2つの接続先に対して設定を切り替えたい場合、この方式では難しい。

これのもっとも容易な解決法は、最新版のpcmcia-csに入れ替えて、pcmcia-cs標準の設定スクリプトを用いるという方法だ。この場合、一からネットワークの設定をやり直す必要があるので、以下にその方法を示す。pcmcia-csの入れ替えやアップデートの方法は、84〜85ページの記事を参照していただきたい。

自宅では固定IPを、また勤務先ではDHCPを利用して接続することを想定して変更した/etc/pcmcia/network.optsファイルを**リスト2**に示す。スキーマ間の切り替えはcardctlコマンドにスキーマ名を渡すことで実行される。**リスト2**では勤務先のための設定をデフォルトにしている。自宅でのネットワークにつないだ場合は、

```
# cardctl scheme home
```

を実行すると、PCカードを自動的に再認識し同時に設定を変更してくれる。同様に勤務先の設定に切り替えるには、

```
# cardctl scheme work
```

と入力するとよい。スキーマは/etc/lilo.confからも呼び出せるので、起動時に切り替えることも可能である。lilo.conf内ではappend命令をもちいて

```
append = "SCHEME=「スキーマ名」"
```

と指定する。lilo.confのついてはマニュアルを参照していただきたい。

■pcmcia-cs 3.1.1の誤認識問題

pcmcia-cs 3.1.1は、FastEther PCC-TXを「Allied Telesys AT-2800」と誤認識してしまい、本来pcnet_csドライバが必要であるところに、AT-2800 PCカードのためのtulip_cbドライバをロードしようとする。このエラーには、/etc/pcmcia/

▼リスト2　スキーマを利用した/etc/pcmcia/network.optsの例

```
case "$ADDRESS" in

home,*,*,*)
    #自宅向けの設定
    ;;
    IPADDR="192.168.100.3"
    NETMASK="255.255.255.0"
    NETWORK="192.168.100.0"
    BROADCAST="192.168.100.255"
    GATEWAY="192.168.100.1"
    DOMAIN="local.net"
    SEARCH="..."
    DNS_1="..."
    DNS_2="..."
    ;;
work,*,*,*|default,*,*,*)
    #勤務先での設定
    ;;
    DHCP="y"
    PUMP="y"
    ;;
esac
```

前から順番にスキーマ名、PCカードソケット番号、デバイス番号、MACアドレスを意味する。場合分けしないなら「*,*,*,*」と指定

192.168.100.0/24のネットワーク向け設定例。SEARCH、DNSフィールドには自宅での設定値を指定

勤務先向けのスキームを指定している。同じ設定をdefaultとしても用いる

DHCP利用時は、この2行をセットで記す

▼リスト3　/etc/pcmcia/configファイルの修正

```
        :
        :
card "Allied Telesyn AT-2800
10/100 Fast Ethernet"
#   manfid 0xc00f, 0x0000
  bind "tulip_cb"
        :
        :
```

この1行が誤認識の原因なので「#」でコメントアウト

configを一部修正することで簡易的ではあるが対応可能だ。修正個所は**リスト3**に示したとおりである。

このような誤認識でPCカードが正常に動作していない場合には、dump_cisコマンドを用いると、カード自身から情報を引き出して表示できるので、最低限の情報は得られる。Manufacture IDの衝突もこのコマンドで発見することができた。ただdump_cisはpcmcia-csに付属のツールで、Red Hat Linuxなどディストリビューションによっては標準装備されていないこともあるので注意されたい。

モデムカード／PHS通信端末

モデムカードやPHS／携帯通信カードなどのデバイスは、PCとのインターフェイスにRS-232C準拠のシリアル通信を用いることが多い。pcmcia-csでも、モデムやPHS通信カードなどはデバイスクラス「serial」と判断されることがほとんどだ。

ここではモデムカードのテストにアイ・オー・データ機器の56kbpsモデムPCML-560Eを、またPHS通信にはNTTドコモのパルディオ611Sを選んで動作確認を行ってみた。

■シリアルデバイスは必ず動く？

シリアルデバイスに分類されるPCカードのほとんどは、PC/AT互換機の標準シリアルデバイスであるNS16550Aと互換性を持たせた設計がなされている。そのため、pcmcia-csが正しく設定されており、必要なIRQとI/Oポートさえ与えられれば、ほとんどのシリアルデバイスは問題なく動作する。シリアルデバイスはIRQが割り当てられていなくても動作してしまう場合があるが、この場合、動作速度が遅くなるのですぐに気づくはずだ。システムリソースが正しく得られなかった場合は、80～85ページの記事を参考に、/etc/pcmcia/config.optsなどの設定ファイルの内容を確認する。

PCカード上のモデムをどのデバイスに割り当てるかは、/etc/pcmcia/serial.optsで設定する（**リスト4**）。ここでは/dev/ttyS0に設定している。もしマザーボード上のシリアルポートがすでに/dev/ttyS0を使っている場合は、どちらかの割り当てを、まだ使っていない/dev/ttyS?デバイスに替えなければならない。マザーボード上のシリアルポートについては、setserialコマンドで設定を変更できる。逆に、PCカード上のシリアルデバイスは、必要な設定すべてをserial.optsファイルで行なわなければならない。

モデムカードがシリアルデバイスとして認識されたか否かは、setserialコマンドで確認できる（**リスト5**）。UARTの項に16550Aと表示されていればpcmcia-cs側の設定は終了である。

モデムデバイスを利用するにはさら

▼リスト4　/etc/pcmcia/serial.optsの設定例

```
case "$ADDRESS" in
*,*,*)
    # Symbolic link to dialout device
    LINK="/dev/ttyS0"
    # Options for 'setserial'
    SERIAL_OPTS=""
    # Should we create an inittab entry for this port?
    #INITTAB="/sbin/mgetty"
    ;;
esac
```

割り当てるデバイス名（/dev/ttySx）をここに指定

setserialコマンドのオプションはここに指定

▼リスト5　シリアルデバイスの確認

```
# setserial -v /dev/ttyS0
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 3
```

デバイス名/dev/ttyS0は、16550A互換のシリアルデバイスで、I/Oポート0x3f8とIRQ3を使用していることがわかる

マルチファンクションカード

スリーコム ジャパン(株)
3Com Megahertz 10/100 LAN+56K グローバルモデム PCカード
5万8000円
http://www.3com.co.jp/

その名の通りEthernetカードとモデムを統合した16bit対応のマルチファンクションカードである。モデム部分はV.90に対応し、最大56kbpsで通信できる。Ethernet部分は100BASE-TXでは半二重のみ、10BASE-Tでは全二重／半二重に両対応している。

にPPPの設定が必要となるが、ここでは触れない。KDEなどはWindowsライクなPPP設定ツールを提供しており、設定が容易にできる。

今回テストした2機種はともにPPPでの通信に容易に成功した。

■ハードウェア名を表示するには

今回テストに用いたパルディオ611Sは、/etc/pcmcia/configファイルにエントリが登録されていないため、ツールによっては「Serial or Modem」と無味乾燥なハードウェア名が表示される。これでもなんら動作に支障はないのだが、複数のシリアルデバイスを用いる場合など、ハードウェア名を区別したい場合は、configファイルにリスト6のようにエントリを追加することで、次の接続から/var/run/stabファイルに"NTT DoCoMo PALDIO 611S PC CARD"と記録されるようになる。また、cardinfoの表示にも反映されるようになる。**リスト6**中にあるversionの値などは、dump_cisコマンドの出力から読み取ることができるので、試してみるとよいだろう。

マルチファンクションカード

昨今のサブノートPCの躍進で、PCカードソケットがTypeII×1本のみというノートPCもずいぶんと多くなった。このようなノートPCで、Ethernetとモデムを同時に利用するにはマルチファンクションカードという選択肢がある。今回は3Com Megahertz 10/100 LAN+56K グローバルモデム PCカードにて、マルチファンクションでの動作を検証した。

■確実に動作するマルチファンクションカードを選ぶ

マルチファンクションカードは一般にEthernetとモデムで同じIRQを用いる。IRQの共用をサポートした最近のカーネルとpcmcia-csの最新版ならば、マルチファンクションカードを利用可能である。**リスト7**は3Com Megahertz 10/100 LAN+56K グローバルモデム PCカードを挿入した状態での/var/run/stabファイルを示している。

この状態でモデムをPPP接続で利用しながらEthernetポートもアクセスしてみたが、特にエラーもなく動作した（こうした使い方では、当然ながらルーティングを適切に設定する必要がある）。

マルチファンクションカードで難しいのは、ノートPC本体との接続において、ハードウェア的な問題を引き起こす場合があることだ。このような場合はドライバサポートのあるWindows 98を用いたとしても不安定であることには変わりない。マルチファンクションカードを選ぶ場合はLinuxで動作するかはもちろん、ノートPCとの互換性の問題も事前に調べる必要がある。

またCardBus対応のマルチファンクションカードでは、モデムがWinModemと呼ばれる専用ドライバが必要なタイプであることが多い。そのため、Linuxからモデムが利用できず、単なるEthernetカードになってしまうこともあるので注意したい。

▼リスト6　修正した/etc/pcmcia/configのModemセクション

```
          :
          :
#
# Modems and other serial devices
#
          :
          :
card "Socket Communications Dual RS-232"
  manfid 0x0104, 0x0006
  bind "serial_cs"

card "NTT DoCoMo PALDIO 611S PC CARD"
  version "NTT DoCoMo","PALDIO 611S PC CARD"
  bind "serial_cs"
```

この3行を追加した

▼リスト7　/var/run/stabの表示例

```
# cat /var/run/stab

Socket 0: 3Com/Megahertz 3CCFEM556 Ethernet/Modem
0         network   3c574_cs  0         eth0
0         serial    serial_cs 0         ttyS0     4         64
Socket 1: empty
```

ソケット0にてEthernetとserialそれぞれの機能が同時に有効になっていることがわかる

ストレージ系のPCカードをテストする

ここでは、ストレージデバイスとノートPCをつなぐSCSIカードやIDEカードといったインターフェイスデバイスのほか、フラッシュメモリやマイクロドライブのように、ストレージデバイスそのものを、Linuxでどの程度利用できるのか探っていく。

SCSIカード

さまざまなストレージデバイスを接続するには、やはりSCSIカードが最も便利だろう。pcmcia-csでもEthernetカードほどではないが、多くのSCSIカードをサポートしている。ドライバ自体は、アダプテックSlimSCSI 1460-Hとその互換製品で使えるaha152x_cs と、QLogic製コントローラ用のqlogic_cs、今はなきFuture Domain製コントローラ用のdomain_cs、そして唯一のCardBus対応ドライバであるアダプテックAPA-1480シリーズ用のapa1480_cb、という4種類だけが、pcmcia-csに同梱されている。一見少なく感じるが、実際にはaha152x_csまたはqlogic_csで動作するSCSIカードが意外に多い。つまりSCSIカードの製品数は多いけれど、使用されているSCSIコントローラの種類は少ないのが現状のようだ。

テスト対象に選んだ製品は、アダプテックジャパンのSlimSCSI 1460-HとAPA-1480-H、そしてメルコのIFC-SCD2である。

■IFC-SCD2のドライバを作成する

前述のようにアダプテックの2製品は、pcmcia-csに同梱のドライバでサポートされている。一方、メルコのIFC-SCD2が内蔵するSCSIコントローラは、ワークビットが開発したNinjaSCSI-3というチップで、ドライバはpcmcia-csに含まれていない。従って単にpcmcia-csがインストールされただけの状態では、IFC-SCD2をソケットに挿入しても認識されず、ブッという低い音が1回鳴るだけだ。

しかしワークビットはWebページでLinuxやFreeBSDのNinjaSCSI-3用ドライバを公開している。これは2.0.x以前のkernelに対応したものだが、それをもとに横田 裕思氏がkernel 2.1／2.2で動作するよう改良したバージョンが、同氏のWebページで公開されている（このドライバは付録CD-ROMにも収録している）。ここではそのドライバを使って、IFC-SCD2をpcmcia-csに認識させてみる。

まずは**リスト1**のように、付録CD-ROMに収録されているドライバのソースコードなどを、pcmcia-csのディレクトリに展開する。ここでもし、pcmcia-csのソースディレクトリが/usr/src/linux以外にあるなら、**リスト2**のようにmodules/nin_cs.c中のインクルードファイルへのパスを変更しておく。そしてpcmcia-csを通常の手順（84〜85ページ参照）で再コンパイルし、モジュールのインストールを行

SCSIカード

アダプテックジャパン(株)

SlimSCSI 1460-H

オープンプライス

http://www.adaptec.co.jp/

SlimSCSI 1460-Hは16bit PCカードのSCSI-2ホストアダプタである。SCSIコントローラにはAIC-6350を採用し、データ転送はPIO方式である。SCSI側の転送レートは最大10Mバイト/秒。添付のSCSIケーブルは、SCSI-2標準の高密度コネクタ付き。

(株)メルコ

IFC-SCD2

8800円

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

IFC-SCD2は16bit PCカードのSCSI-2ホストアダプタである。SCSIコントローラにはワークビットのNinjaSCSI-3を採用し、転送レートは最大10Mバイト/秒。Windows上でパーティション設定やフォーマットを実行するユーティリティが付属する。

アダプテックジャパン(株)

APA-1480-H/J97 EZ kit

2万9800円

http://www.adaptec.co.jp/

APA-1480は、SCSIコントローラにAIC-7860を採用したUltra SCSIカードだ。CardBus対応でバスマスタDMA転送もサポートする。添付ケーブルのコネクタは、SCSI-2標準の高密度タイプ。Windows用のEZ-SCSI デラックス版というソフトウェアが添付される。

う。これでNinjaSCSI-3のドライバモジュールであるnin_cs.oが作られインストールされる。

次にIFC-SCD2を認識させるために、/etc/pcmcia/configを修正する。いったんIFC-SCD2をPCカードソケットに挿入すると、ログにPCカードの情報を記録される。それを**リスト3**のように表示し、それをもとに新しいエントリをconfigに追加すればよい。

■サスペンド／レジュームは使えず

3枚のSCSIカードすべてのドライバが揃ったところで、以下のようなテストを行った。

○pcmcia-csによる認識
○ハードディスクの読み書き
○CD-ROMドライブからの読み出し
○メディア交換検出のチェック
○サスペンド／レジューム

概要	URL
ワークビットのWebページ	http://www.workbit.co.jp/workbit/products/nscsi-3.html
横田氏のWebページ	http://www.netlab.is.tsukuba.ac.jp/~yokota/izumi/ninja/
付録CD-ROM	/PCCard/NinjaSCSI3

表1　NinjaSCSI-3に関する情報のありか

▼リスト1　NinjaSCSI-3ドライバのインストール

```
# mount -t iso9660 /dev/cdrom /mnt/cdrom
# cd /usr/src/pcmcia-cs-3.0.13/          ← このパスは環境に合わせて変更

# tar zvxf /mnt/cdrom/PCCard/NinjaSCSI3/NinjaSCSI3-0.2.1.tar.gz
# vi modules/nin_cs.c                    ← リスト2に従ってnin_cs.cを変更

# make config                            ← pcmcia-csのmake
          :
```

▼リスト2　nin_cs.cの変更箇所

```
■48〜49行目
#include "drivers/scsi/scsi.h"          →      #include <../drivers/scsi/scsi.h>
#include "drivers/scsi/hosts.h"         →      #include <../drivers/scsi/hosts.h>
■53行目
#include "drivers/scsi/scsi_ioctl.h"  →  #include <../drivers/scsi/scsi_ioctl.h>
■99行目
#include "drivers/scsi/sd.h"   →      #include <../drivers/scsi/sd.h>
```

▼リスト3　/etc/pcmcia/configへのエントリ追加

```
■IFC-SCD2の内部名称を引き出す

                    :          ← ここでIFC-SCD2をPCカードソケットに挿入
# tail /var/log/messages       ← Debianなら/var/log/daemon.log
                    :
Oct 24  4:32:59 localhost cardmgr[274]: unsupported card in socket 1
Oct 24  4:32:59 localhost cardmgr[274]:   product info: "WBT", "NinjaSCSI-3", "R1.0"

#

■/etc/pcmcia/configに以下を追加する
device "nin_cs"
  class "scsi" module "nin_cs"

card "Workbit NinjaSCSI-3"
  version "WBT", "NinjaSCSI-3", "*"
  bind "nin_cs"
```

電源の入ったハードディスクをつないだSCSIカードをソケットに挿入すると、自動的にハードディスクが検出され、/dev/sdaといったデバイス名が割り当てられる。CD-ROMドライブだったら/dev/sr0あるいは/dev/scd0といった具合だ。テスト時には、こうして割り当てられたデバイス名を使ってハードディスクやCD-ROMをマウントした。

結果は、アダプテックの2製品はサスペンド／レジューム以外はまったく問題なく実行できた。IFC-SCD2はpcmcia-csによる認識とハードディスクの読み書きについては問題なかったが、CD-ROMドライブからの読み出しでトラブルが生じた。アップデートしていないLASER5 Linux 6.0の環境では、ISO9660フォーマットのCD-ROMをマウントしようとしても、エラーが生じてマウントできないのだ。またkernelとpcmcia-csをアップデートしたLASER5 Linux 6.0では、PCカード挿入後にCD-ROMドライブを検出した直後にマシンがハングアップしてしまう。ハードディスクの読み書きは正常だったことから、CD-ROMドライブのようなリムーバブルドライブに起因するトラブルのようだ。NinjaSCSI-3ドライバのアーカイブに含まれるドキュメントにも、メディア交換の検出がうまく働かないという問題が記されている。

どのSCSIカードでもまったく動作しなかったのがサスペンド／レジュームである。“ cardctl suspend ”を実行すると予期しないIRQが生じた、などというエラーメッセージが表示されるほか、その直後にマシンがハングアップしてしまうこともしばしばだった。SCSIカードをノートPCに装着したままサスペンドする場合は、その前に“ cardctl eject ”を実行してPCカードを停めるべきだろう（レジューム時に

は自動的に起動される)。

■scsi.optsの役割

テスト中にはマウントの作業を手動で行っていたが、ふだんの環境では、SCSIデバイスをつなげたSCSIカードがPCカードソケットに装着されたら、ドライブのマウントも決められたとおりに自動実行されるほうが便利だろう。それにはscsi.optsをカスタマイズすればよい。**リスト4**は、2つのパーティションからなるハードディスクを、所定の方式でマウントする例だ。同様な役割を果たす設定ファイルとして/etc/fstabが存在するが、pcmcia-csは独自にその機能を実現している。

フラッシュメモリ／マイクロドライブ

現在、デジタルスチルカメラとLinux PCの間で画像データをやり取りするには、PC側でコンパクトフラッシュまたはスマートメディアにアクセスできるよう、PCカードソケットとアダプタを用意するのが一般的だろう。こうした用途を想定してテストに選んだのが、アイ・オー・データ機器のPCCF-64M（64Mバイトのコンパクトフラッシュ）とPCFDC IV-ADP（スマートメディアのアダプタ）である。後者は別途購入した16Mバイト3.3Vスマートメディアと組み合わせてテストした。このほかメルコが販売するIBM製マイクロドライブRMD-CA340Mもテスト製品に加えてみた。

この3製品に共通なのは、いずれもPCとのインターフェイスにIDEを採用していることだ。そのため、実際にテストしてみると、PCカードドライバはどの製品でもide_cs.oが使用された。pcmcia-csにはメモリデバイス専用のドライバが何種類か用意されているが、実際に現在流通しているPCカード互換のメモリカードの多くはIDEインターフェイスを採用しているため、ide_cs.oばかりが使われるのが現状だ。

■動作に問題点はなし

さてLASER5 Linux 6.0（アップデートなし）でのテスト結果だが、まず3製品とも手動設定なしでPCカードが認識され、IDEドライブとして/dev/hdeが割り当てられた（hda～hddは内蔵ドライブが使用)。またメディアへのアクセスも問題は生じなかった。

サスペンドのテストでは、3製品とも「unexpected interrupts」というエラーメッセージが表示される。この症状はkernelとpcmcia-csをアップデートしたLASER5 Linux 6.0でも変わらなかった。どうやらドライバ（ide_cs.o）がサスペンドに完全対応していないようだ。SCSIと同様、サスペンド前に“cardctl eject”と実行して停止させたほうが無難だろう。

なおide_cs.oはホットスワップに対応していないので、必ずcardctl／cardinfoでPCカードを停止（ソフトウェア的に取り外す）してからPCカードそのものを取り外す必要がある。

■シリアルナンバで自動設定

pcmcia-cs付属のide_infoコマンドを使うと、**リスト5**のように各ドライブのシリアル番号が表示される。シリアル

フラッシュメモリ／マイクロドライブ

(株)アイ・オー・データ機器
PCCF-64M
3万1000円
http://www.iodata.co.jp/

PCCF-64Mは、容量64Mbytesのコンパクトフラッシュカードである。PCカードソケットに装着するためのアダプタが標準添付されており、フラッシュATAカードとして使用することも可能だ。より大容量のカードもラインアップされている。

(株)アイ・オー・データ機器
PCFDC IV-ADP
8000円
http://www.iodata.co.jp/

PCFDC IV-ADPは、スマートメディアをPCカードソケットに装着するためのアダプタだ（写真のスマートメディアは製品に含まれない)。スマートメディアをフラッシュATAカードとしてPCから利用できる。スマートメディアの3.3V／5V仕様の両方に対応。

(株)メルコ
RMD-CA340M
5万8000円
http://www.melcoinc.co.jp/

RMD-CA340Mは、IBMの開発したマイクロドライブと呼ばれる小型ハードディスクで、ディスク直径500円玉大程度の大きさに340Mバイトを記録できる。PCには専用PCカードアダプタを介して接続する。デジタルカメラでも対応機種が存在する。

番号には製品ごとにユニークな値が割り当てられているので、これさえ判別できればドライブを特定できるし、/etc/pcmcia/ide.optsに渡されるデバイスアドレスには、このシリアル番号が含まれる。ところが**リスト5**を見直すと、PCFDC IV-ADPだけはシリアル番号がない。後述のポータブルCDROMドライブも同様だ。どういう法則かは不明だが、同種のIDEドライブでもシリアル番号が読み出せないデバイスが存在する。すべてのIDEドライブを特定するのは無理のようだ。

ポータブルCD-ROMドライブ

ここでは、IDE／ATAPIインターフェイスカードを採用したポータブルCD-ROMドライブとして、メルコのCDN-D24VAとCDN-24EXという2機種をテスト対象に選んでみた。これらは同じCD-ROMドライブを使っているものの、インターフェイスカードは16bit PCカード(CDN-D24VA)とCard Busカード（CDN-24EX）と異なる。

■CardBusのIDEは利用できない

まずCDN-D24VAは、アップデートしていないLASER5 Linux 6.0でも、手動設定なしで認識された。ドライバはide_cs.oが使用され、メディアのマウント／アンマウントも問題なく実行できた。メディアの入れ替えも検出している。サスペンド／レジュームではやはりエラーが表示されるものの、実用上の問題は見つからなかった。なお、kernelとpcmcia-csをアップデートしたLASER5 Linux 6.0でもテストしたが、サスペンド／レジューム時のエラーに変化はなかった。

一方、CDN-24EXはPCカード自体が認識されず、まったく利用できなかった。pcmcia-csでは現時点でCardBus対応のIDEドライバが開発されていないため、割り当てるべきドライバが存在しない。Linuxで利用するなら、16bit PCカードのIDEインターフェイスをセットにしたCD-ROMドライブか、サポートされているSCSIカードとSCSI CD-ROMドライブのどちらかを選ぶしかない。

ポータブルCD-ROMドライブ

(株)メルコ

CDN-D24VA

2万3500円

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

CDN-D24VAは、ATAPI接続カード（16bit PCカード）添付の最大24倍速ポータブルCD-ROMドライブだ。ソニーVAIOシリーズの純正ドライブと同等機能を持っており、VAIO505シリーズやC1シリーズでは起動フロッピーなしでOSを再セットアップできる。

(株)メルコ

CDN-D24EX

3万1000円

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

CDN-D24EXは、最大24倍速のATAPI CD-ROMドライブである。CardBusに対応したATAPI接続カードが標準添付されており、CDN-D24VAより高速なデータ転送が可能だ。しかしその分、価格が多少高めになっている。コントローラはワークビット製。

▼リスト4　scsi.optsの例

```
case "$ADDRESS" in
*,sd,*,0,1,0)
    PARTS="1 2"                          ← パーティションの数
    ;;
*,sd,*,0,1,0,1)
    DO_FSTAB="y" ; DO_FSCK="n" ; DO_MOUNT="y"   ← マウント時のオプション
    FSTYPE="vfat"                        ← フォーマットの種類
    OPTS=""                              ← PCカードドライバへのオプション
    MOUNTPT="/home"                      ← マウントポイント
    ;;
*,sd,*,0,1,0,2)
    DO_FSTAB="y" ; DO_FSCK="y" ; DO_MOUNT="y"
    FSTYPE="ext2"
    OPTS=""
    MOUNTPT="/prj"
    ;;
esac
```

左から順番にスキーマ名、デバイスのタイプ、ソケット番号、SCSIチャンネル、SCSI ID番号、論理ユニット番号、パーティション番号

▼リスト5　IDEデバイスのシリアル番号

■PCCF-64M

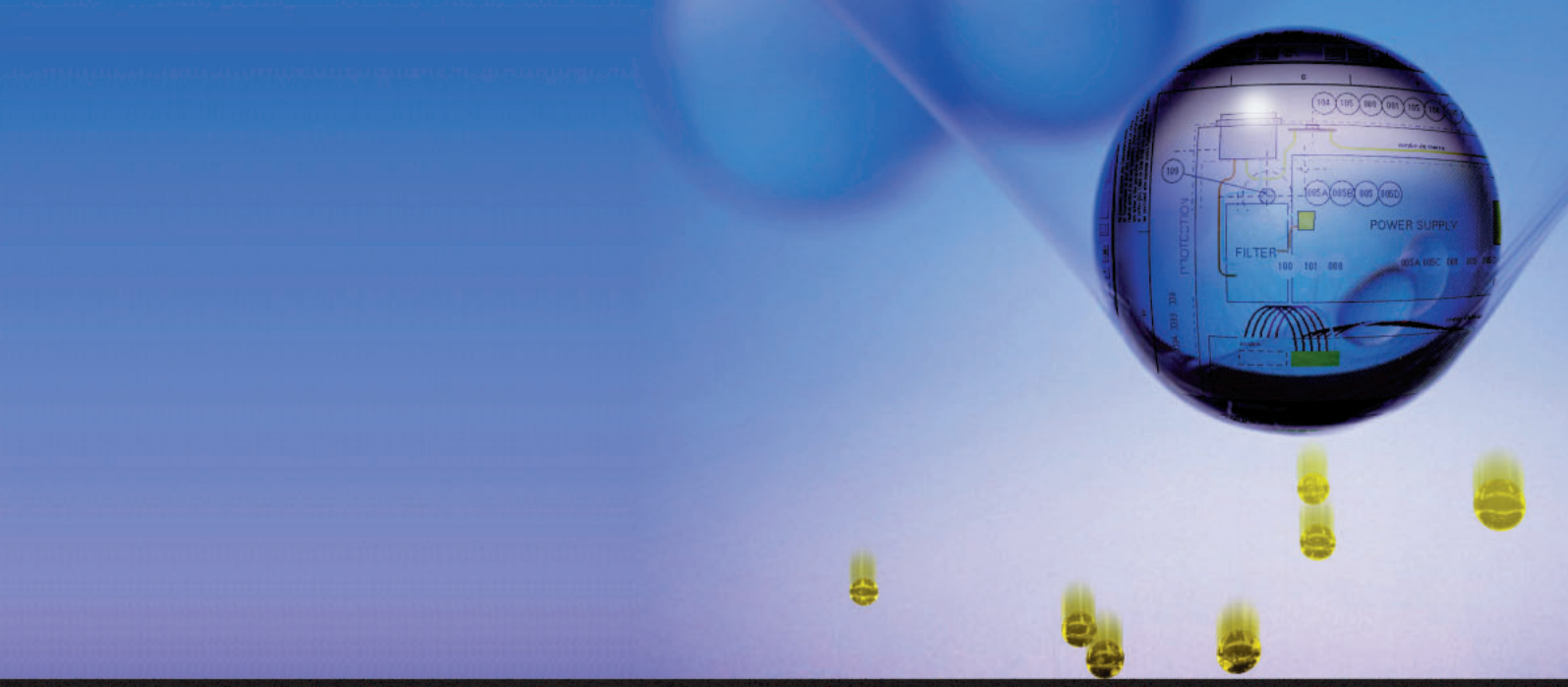
```
MODEL="SunDisk SDCFB-64"
FW_REV="vcb 1.34SunDisk SDCFB-64"
SERIAL_NO="MT312423310"
```

■PCFDCIV-ADP（スマートメディアを含む）

```
MODEL=""
FW_REV="KS522"
SERIAL_NO=""                             ← シリアル番号がない
```

■RMD-CA340M

```
MODEL="IBM-DMDM-10340"
FW_REV="MD2IC501IBM-DMDM-10340"
SERIAL_NO="XHA12631"
```



Software Review

開発ツール!? Applixware 4.4.2

先月号に引き続き、『Applixware 日本語版 4.4.2』を紹介する。すでに先月号の94ページでワードプロセッサ「Applix Words」、表計算ソフト「Applix Spreadsheets」の紹介を行っているが、Applixwareはオフィスアプリケーションスイートとしてだけでなく、開発ツールとしての側面も持っている。今回は、マクロ言語「ELF」や開発環境「Applix Builder」の機能を紹介し、開発ツールとしてのApplixwareを紹介する。

開発ツールとしてのApplixware

前号でも紹介したとおり、「TurboLinux PRO 日本語版 4.2」にバンドルされている「Applixware 日本語版 4.4.2」（以下、Applixware）は、Microsoft Office（以下、MS-Office）に迫る機能を持つオフィススイートである。

しかし、Applixwareの実力はそれだけではない。Applixwareは、各機能をコンポーネント化することによって、アプリケーションの柔軟なカスタマイズが可能となっている。さらに、カスタムアプリケーションを作成するためのツールも搭載するなど、開発環境としての側面も持っているのだ。

■コンポーネント化された各機能

「機能のコンポーネント化」ということについて、具体例を挙げてもう少し説明しよう。

Applixwareを使ってみるとすぐに気づくが、似たようなインターフェイスを持ったアプリケーションが多い。

たとえば、プレゼンテーションツール「Applix Presents」（**画面1**）とグラフィックツール「Applix Graphics」（**画面2**）を見比べてほしい。

ツールバー（Applixwareではエクスプレスラインと呼ぶ）の構成などが若干異なるが、一見した感じではどちらがどちらかわからないくらい似ている。同様に、ワードプロセッサ「Applix Words」とHTMLエディタ「Applix HTML Author」やマクロエディタなどもかなり似ている。

これは、アプリケーションごとに機能を実装するのではなく、いくつかのアプリケーションで共通するような機能（たとえば、作表機能やスペルチェック機能など）を、部品（コンポーネント）として作成し、それらを必要に応じて「コンテナ」と呼ばれる土台に呼び出すことで、1つのアプリケーションに見せかけているからである。

そういった意味では、Applixwareの各アプリケーションは、「コンポーネントの集合体」ともいえよう。ちなみに、MS-Officeも同様にコンポーネント化された構造を持っている。

■柔軟なマクロ言語と開発環境

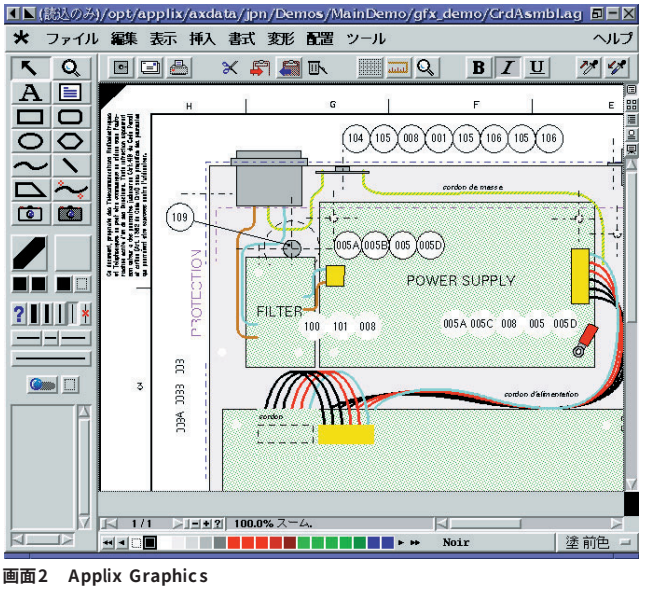
これらの各コンポーネントを関連づける「のり」のような役割を果たしているのが、これから紹介するマクロ言語「ELF」である。Applixwareは、このELFを利用してコンポーネントの呼び出しや操作を行っている。

ELFは、マクロの作成、編集、デバッグなどが可能なツール「マクロエディタ」を利用して、アプリケーション用のマクロを作成することができる。さらに、開発ツール「Applix Builder」を利用すれば、アプリケーションの枠を取り払い、各コンポーネントを組み合わせたカスタムアプリケーションも比較的容易に作成できる。

今回は、ELFを利用する、「マクロエディタ」と「Applix Builder」という2つのツールを紹介する。いずれも、ビジネス利用を意識した本格的なツールである。

オフィススイートとしてだけでなく、開発環境としてのApplixwareの実力をご覧いただきたい。

画面1　Applix Presents



画面2　Applix Graphics

マクロ言語「ELF」

ELFという言語がApplixwareにおいて重要な働きをすることは、すでに前節で述べたとおりである。では、このELFとはどんな言語なのか？　また、どのようにして利用するのか？

ここでは、まずELFという言語の概要を紹介をしてから、Applixwareアプリケーションのマクロ言語としての利用方法を解説する。

ELFとは？

ELF（Applix Extension Language Facility）は、Applixwareに搭載されているコンパイラ型のプログラミング言語である。Applixwareのアプリケーションを操作するマクロ言語としての利用はもとより、各アプリケーションから一部の機能だけを呼び出して、カスタムアプリケーションを作成することも可能である。その機能は強力かつ柔軟で、MS-Officeに搭載されているVBA（Visual Basic for Applications）に勝るとも劣らない機能を持っているといえよう。

このELFをApplixwareのマクロ言語として利用する際、「.am」という拡張子を持つテキストファイルとして保存される（書体指定を行っていても、タグが埋め込まれるだけで、データそのものはテキストファイルであることに変わりはない）。

ELFマクロの編集ツールとして、後述する「マクロエディタ」が用意されているが、マクロファイルそのものはテキストファイルであるため、grepやsedのようなLinuxに搭載されているフィルタ系ツールやviやemacsといったエディタなどと組み合わせて利用することが可能である。

EFLマクロは、Applixwareによって自動的にコンパイルされて、メモリ内にインストールされる（「マクロエディタ」のメニューからコンパイルした状態で保存しておくこともできる。この際のファイルの拡張子はeloとなる）。

なお、作成されたマクロファイルを/*usrname*/axhome/macros/に保存すると、そのユーザー専用のマクロとなる。/*install_dir*/axlocal/にマクロを保存しておくと、Applixwareを利用しているすべてのユーザーがここにあるマクロを利用できる。

ELFマクロを作成するには次の2つの方法がある。

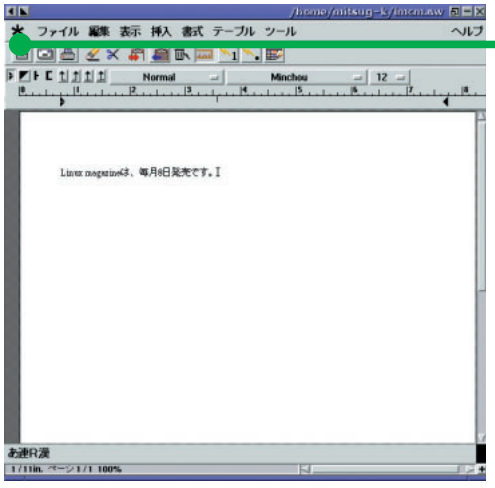
キーストロークの記録

ELFマクロを利用するのに、最も手軽な方法がキーボード上の動作を記録させる「キーストロークの記録」（Windowsなどでは、キーボードマクロともいう）だろう。これなら、ELFの文法を覚える必要がない。

たとえば、あるサンプル文章（**画面3**）を用意して、次のような手順でマクロの登録を行う。

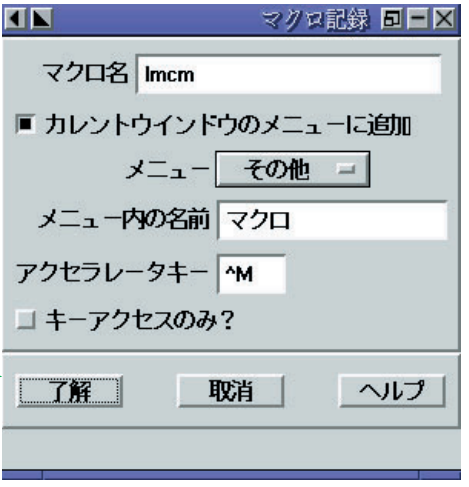
1.メニューの左端にある「＊」メニューから［マクロ記録］を選択し（Shift＋F8でもよい）、マクロの記録を開始する（**画面4**）。

2.カーソルキーとシフトキーなどを利用して、キーボードから文字列を選択し、いろいろな書式選択をする（ここでは文字列を選択し、数行下にコピーしたのち、フォントを24ポイントにしてから、ある文字列だけを赤に変更してい



画面3　Applix Wordsによるサンプル文書

画面4［マクロ記録］メニュー



画面5　マクロの保存

る)。このとき、書式選択にはマウスを使っても構わないが、文字列の選択にはマウスを使わず、キーボードから選択しておかないと、マクロが正確に動作しないので注意してほしい。

3.作業が終わったら、再度「＊」メニューから［マクロ記録］を選択し、マクロの記録を終了させる。

4.マクロの記録が終了すると、**画面5**のようなダイアログボックスが開くので、マクロ名をつけて保存しておく（ここでは、“ lmcm ”という名前をつけた)。このとき、［メニュー］プルダウンメニューから［その他］を選択しておくと、記録したマクロが［その他］メニュー（現在のウィンドウにないメニューを選んだ場合は、作成される）の下に登録される。

これで、マクロの登録は終了である。マクロを記録させるために作った文字列は削除して（カーソル位置も含めて、最初の状態に戻して)、［その他］メニューにある［マクロ］を選択すると、自動的に文書が変更される（**画面6**)。

マクロエディタ

「キーストロークの記録」によるマクロ作成は、手軽で簡単だ。しかし、だんだん使っていくと、ある一部分の挙動を変更したいだけなのに、いちいちすべての動作をキーボードで再現しないといけないのが面倒になってくるはずだ。

また、「キーストロークの記録」で作成されたマクロは、あまり効率のよいものではない。というのは、キーボードの移動によって指定するため、カーソルが右に何回、左に何回という単純な位置指定しかできない。

本来なら、段落単位やページ単位といった、文章の論理構造に合わせた指定をして、汎用性のあるマクロにするのが理想的である。こういったマクロを作成するためにも、ある程度慣れてきたら、少しずつELFの文法を覚えて、マクロエディタを活用することをお勧めする。

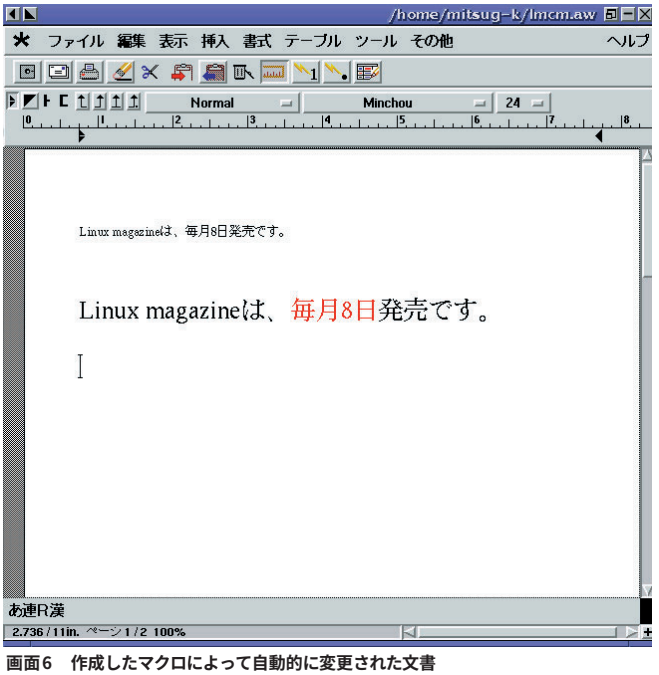
また、作り方がわからなくなったときも、「キーストロークの記録」を使って作成したいマクロと同様の働きをするマクロを作成しておいて、それをマクロエディタでながめてみると勉強にもなるだろう。/opt/applix/axdata/elfディレクトリには、サンプルプログラムが収録されているので、それらを参照してみるのもよいだろう。

■マクロエディタによる編集

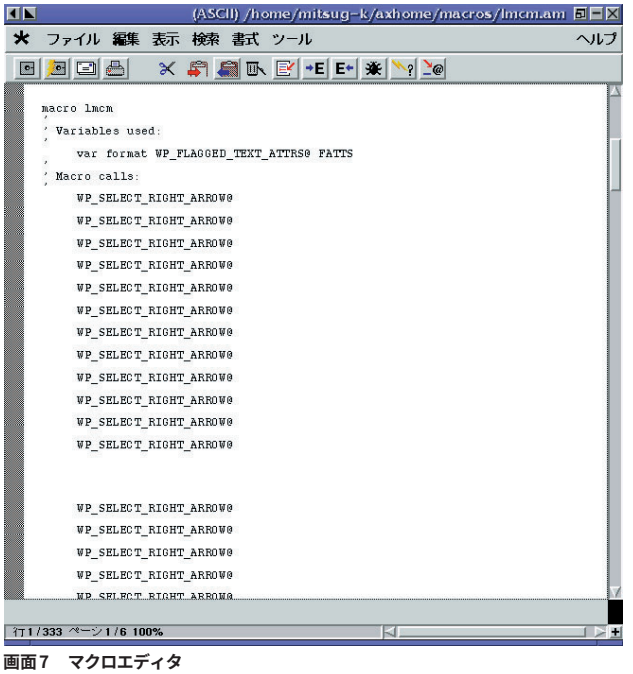
マクロエディタは、［＊］メニューから［マクロエディタ］を選択して起動する。ファイルの読み込みは、起動したマクロエディタの［ファイル］メニューから［オープン］を選択し、ダイアログボックスからファイルをクリックする。

では、さきほど「キーストロークの記録」で作成したマクロ「lmcm」をマクロエディタで開いてみよう（**画面7**)。

“ WP_SELECT_RIGHT_ARROW@ ”という文字列がずっと連続しているの



画面6　作成したマクロによって自動的に変更された文書



画面7　マクロエディタ

が見える。この文字列の意味を調べるときは、カーソルを調べたい文字列に合わせておいて、エクスプレスラインの右から2番目のボタン（稲妻に？マーク）をクリックすれば、説明や使用法が書かれたヘルプが起動する（**画面8**）。

これによると、これはどうやら選択範囲を1文字分右に移動させていることがわかる。なお、末尾が「@」で終わっているのは、Applixwareがあらかじめ用意している組み込みマクロライブラリのことで、約4000近くのマクロがあらかじめ用意されている。また、接頭辞「WP」は、Wordsマクロであることを表している。

また、覚えづらい組み込みマクロ名は、途中まで入力してから、エクスプレスラインの右端のボタン（矢印に@マーク）を押せば、その文字列を含んだ組み込みマクロの一覧が表示されるので、そこから選べばスペルミスも防ぐことができる（**画面9**）。

■デバッガ

マクロエディタには、デバッガも用意されており、ブレイクポイントの設定や変数値の表示など、基本的なデバッグ作業が行える（**画面10**）。

ウィンドウは、同時に16まで開くことが可能で並列作業の際は便利だろう。

■ダイアログボックスエディタ

ダイアログボックスエディタを利用すれば、グラフィカルなユーザーインターフェイスを作成することも可能である。なお、このダイアログボックスエディタは、あくまでもダイアログボックスをレイアウトするためのエディタで、イベントに対するステートメントは、作成したダイアログボックスを呼び出すマクロ内に記述しておかなくてはならない。

ダイアログボックスエディタを起動するには、マクロエディタの［ツール］メニューから［ダイアログボックスの作成］を選択する（**画面11**）。

■ビットマップエディタ

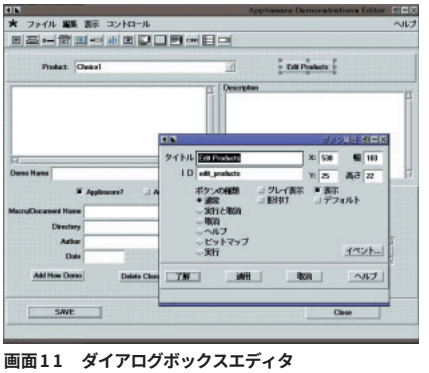
ビットマップエディタは、ダイアログボックスのプッシュボタンや装飾などに使用できる、ビットマップイメージを作成、変更が可能なツールである。［ツール］メニューの［ビットマップ作成］で起動する。

充実のオンラインヘルプ

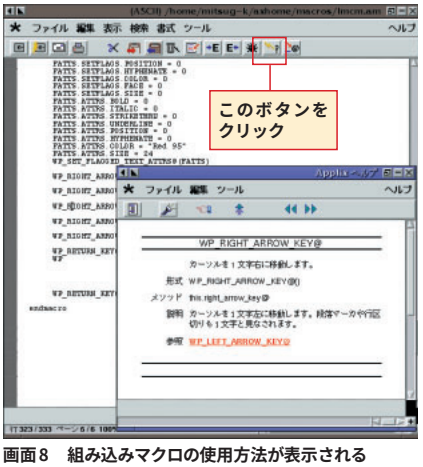
このようなマクロを書く際、ドキュメントは必須となる。Applixwareには、マニュアルとなる冊子は同梱されていないが、代わりにオンラインヘルプがついている。これは、章ごとにジャンプ可能なナビゲートウィンドウと、ヘルプそのものを表示するウィンドウの2つが表示される。これまでのLinuxアプリケーションの中では、かなり充実している部類に入る。

また、他のドキュメントや項目などへの参照もオンラインヘルプらしくリンクされているので、参照が容易だろう。さらに、PSファイルなどへエクスポートすることができるので、プリントアウトして読むこともできる。

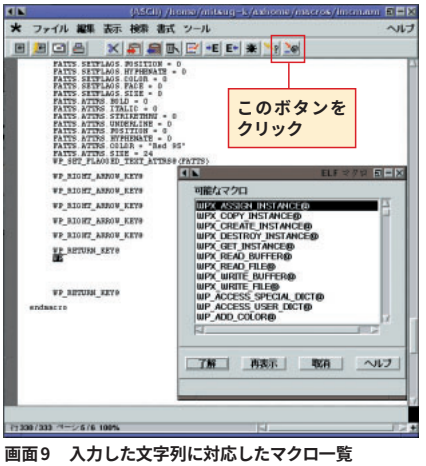
ただ残念なのは、組み込みマクロの説明や関連マクロの説明が少ないことである。MS-Officeのオンラインヘルプには、関連項目へのリンクはもとより、その関数を用いたサンプルまで数多く載っていて使いやすい。このようなヘルプがないと作成時に不便なので、今後より一層の充実に期待したい。



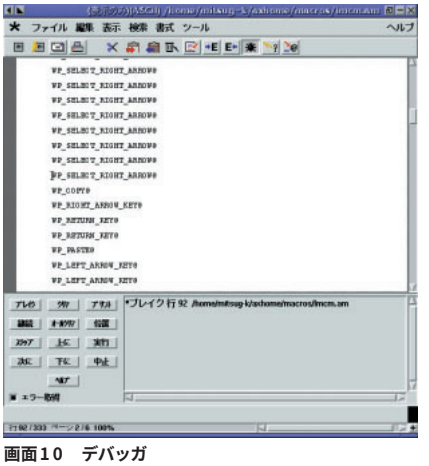
画面11　ダイアログボックスエディタ



画面8　組み込みマクロの使用方法が表示される



画面9　入力した文字列に対応したマクロ一覧



画面10　デバッガ

開発ツール「Applix Builder」

前節では、ELFをアプリケーションのマクロ言語として利用した例を紹介した。さらに、ここではELFの応用事例として、開発ツール「Applix Builder」を用いたカスタムアプリケーションの作成について紹介する。

Applix Builderとは？

Applix Builder（以下、Builder）は、ELFをプログラミングメソッドに用いたオブジェクト指向のGUI開発ツールで、イベントドリブンなアプリケーションの作成が可能である。主な開発用途としては、次のようなものが考えられるだろう。

- 複数のデータベースを検索するデータベースアプリケーション
- リアルタイムアプリケーション
- C++ライブラリを統合した強力なアプリケーション
- 複数のアプリケーションで共有される再使用可能なアプリケーション
- Applixwareアプリケーションを統合したカスタムアプリケーション

Applix Builderの機能

Builderは、［＊］メニューから［Builder］を選択して起動する（**画面12**）。主に、［ブラウザ］、［ソース］、［デザイナ］、［コネクタ］、［デバッガ］という5つのツールからなる。これらのツールのそれぞれの機能や役割は次のとおりである。

■ブラウザ

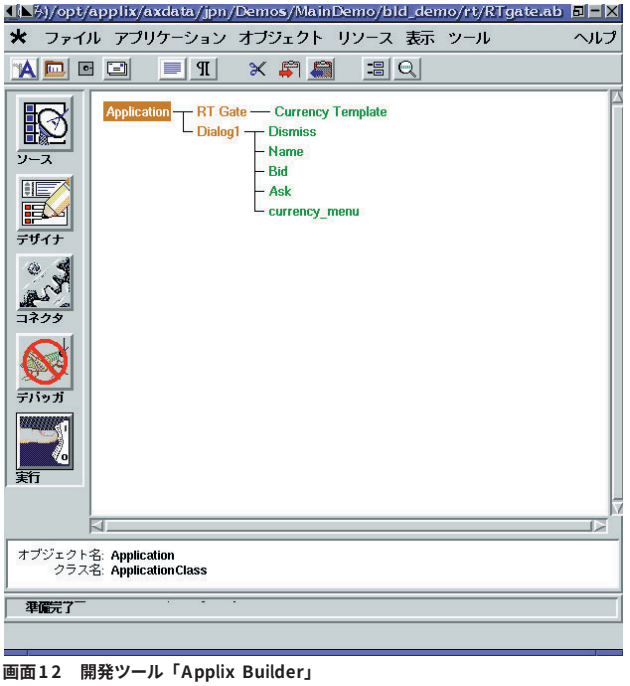
起動したBuilderの中心にあるウィンドウが「ブラウザ」で、アプリケーション内のオブジェクト構造を表示したり、Builderのすべての機能にアクセスすることができる。また、オブジェクトの追加や削除、メソッドの変更、基本クラスの表示を行うこともできる。

■クラスブラウザ

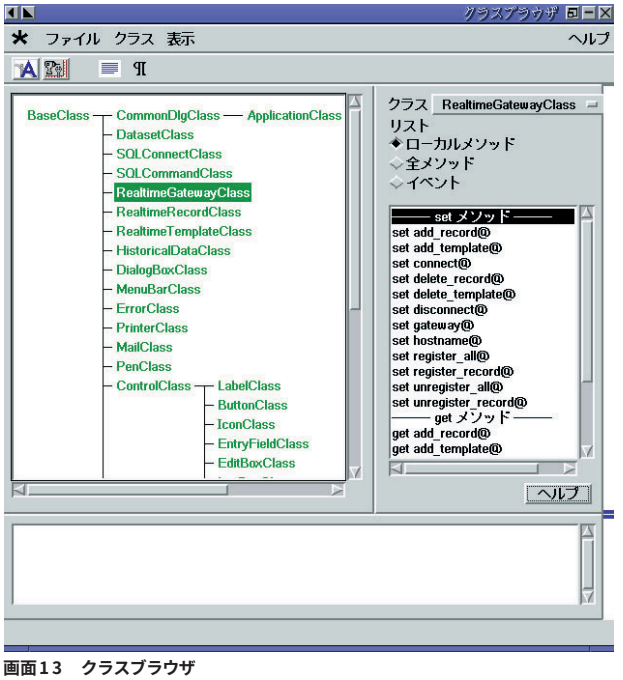
Builderはオブジェクト指向の開発ツールであり、基本クラスやユーザーが定義したクラスから属性を継承することで、新たなオブジェクトを作成することができる。そのクラス構成を参照するのに便利なのが、「クラスブラウザ」である（**画面13**）。使い方は簡単で、階層表示されているクラスから目的のものをクリックすると、右側の［リスト］エリアに使用可能なメソッドやイベントが表示される。

■ソースツール

ソースツールは、データベースアプリケーションなど、データ中心のBuilderアプリケーションの作成に利用される。データソースを作成し、OracleやSybaseといったデータベースにアクセスするクライアントアプリケーションを作成できる。また、継続的



画面12　開発ツール「Applix Builder」



画面13　クラスブラウザ

に変化するデータの取得・分析・配布を行うためのリアルタイムデータソースも作成することができる。

■デザイナツール

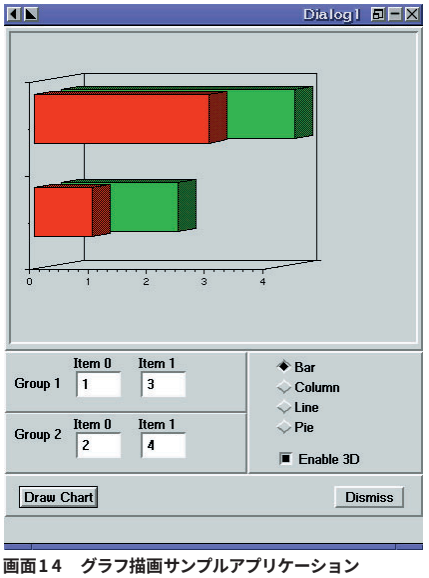
デザイナツールは、データセットやマクロへのインターフェイスとなるダイアログボックスを作成するためのツールである。

■コネクタツール

コネクタツールは、ダイアログボックスの属性や動作を変更したり、ダイアログボックスオブジェクトをデータソースに接続することができるツールである。

Applix Builder プログラミングの実際

上記のツールによって、Builderは非常に柔軟性が高く、さまざまなビジネスアプリケーションを作成することができる。手ごろなサンプル集が、/opt/applix/axdata/jpn/Demos/MainDemo/bld_demoディレクトリ内に収められているので、開発時の参考になるだろう。



画面14　グラフ描画サンプルアプリケーション

このサンプル集から、2つをピックアップして紹介しよう。

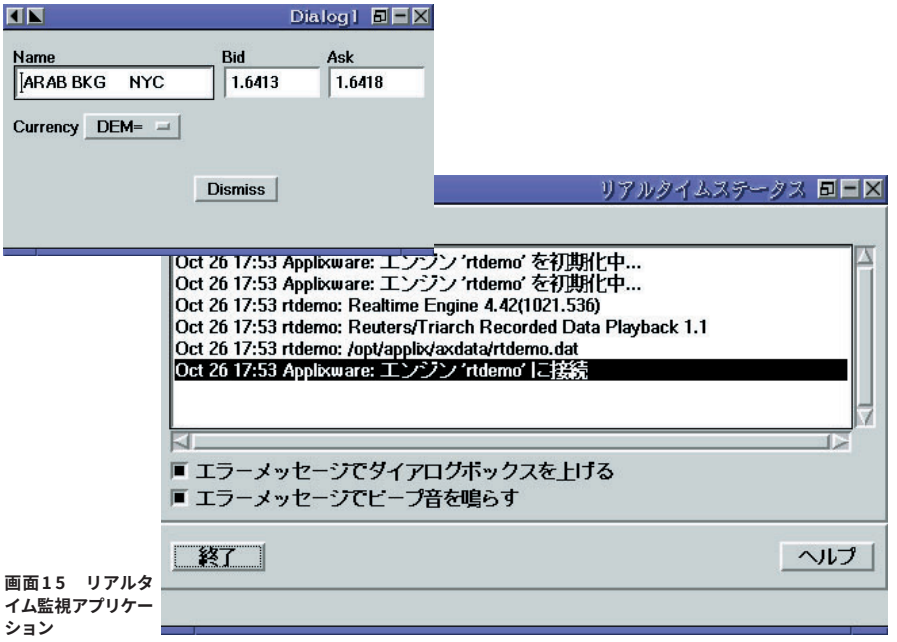
■グラフ表示アプリケーション

まずは、サンプルディレクトリ内のchartディレクトリ以下にある、ダイアログボックスからグラフ描画コンポーネントを呼び出すサンプルプログラム（Chart3.ab）である（**画面14**）。画面下のボックスの数字を変更すると、グラフも変更されるようになっている。

また、2Dと3Dの切り替え表示も可能である。あるコンポーネントだけを呼び出した、簡易的なカスタムアプリケーションといえよう。

■リアルタイム監視アプリケーション

より高度なアプリケーションのサンプルが、サンプルディレクトリ内のrtディレクトリ以下にある、リアルタイムにデータベースの更新を監視するサンプルプログラム（RTgate.ab）である（**画面15**）。これは、サンプルといいながら、かなり本格的なアプリケーションで、このサンプルを参考にすることで、実用的なアプリケーションの開発も可能だろう。



画面15　リアルタイム監視アプリケーション

より大規模な開発に向けて

Builderは、さらに大規模なアプリケーションの開発のためのインターフェイスも装備している。CORBAサポートがそれである。

CORBA（Common Object Request Broker Architecture）は、ネットワーク上にあるオブジェクトへのアクセスや利用を可能にする、分散オブジェクトテクノロジーの標準規格である。ApplixwareのCORBAサポートは、ネットワークを利用した大規模アプリケーションの開発が可能であることを意味する。多くのデベロッパーにはまだまだ必要がない機能かもしれないが、スケーラビリティへの保証は導入への動機づけになるはずである。

ここまで、開発ツールとしてのApplixwareを簡単に紹介してきた。これを機に、Applixwareを利用したアプリケーションプログラミングのきっかけとなれば幸いである。

実験：フィルタツールが使える？

今まで書いていなかったが、実はApplixwareにはもう1つ大きな特徴がある。それは、Applixwareで作成したデータファイルは、拡張子が.awや.asとなってはいるものの、Applixware独自のバイナリ形式ではなく、テキスト形式で保存されているということだ。lessなどでファイルをのぞいてみるとわかるが、書体指定などの部分は、HTMLのようにタグ（のようなもの）を入れておき、Applixwareがそれを解釈することでリッチテキストとして表示している。なお、画像データなどは符号化される。

では、このテキストファイル形式でデータを持つことがどんな意味を持つのか？　それは、それほどうれしいことなのか？

答えは、とってもうれしいである。テキストファイルということは、grep、sed、AWKといったUNIXの超強力フィルタツールが利用できるということだ。これをうれしいといわずとしてなんといおう。

これなら、Wordsで作った文書をsedでガリガリ置換したり、Spreadsheetsで作ったデータをAWKでガリガリ列を入れ替えたりできるはずだ。

では、さっそく実験ということで、/opt/applix/axdata/jpn/Demos/MainDemo/wp_demoにあるArticle.aw（**画面16**）の文章中の「bears」を「LION」に変換してみた（なお、わかりやすいよう「bears」の文字列のフォントを大きめにして、赤にしておいた）。

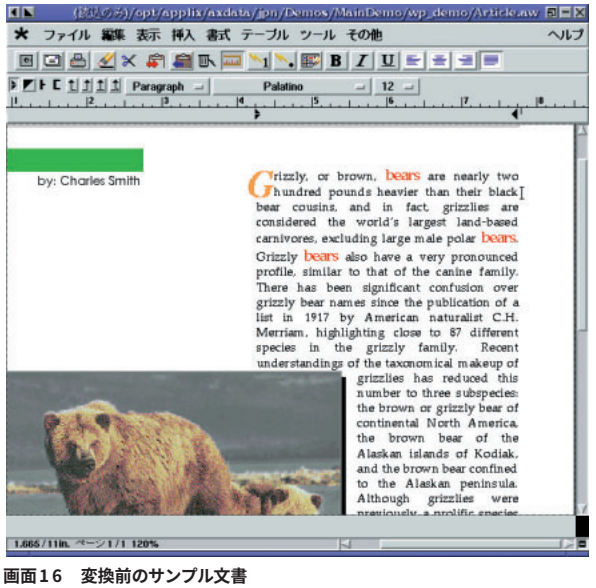
```
# sed -e 's/bears/LION/g' Article.aw
> Article2.aw
```

このArticle2.awを表示してみよう（**画面17**）。うまくいっている。シメシメだ。この程度ならWordsの置換コマンドを使っても大差ないが、さまざまなコマンドと組み合わせれば、全文書の文字列を一度に置換したり、ある文字列だけを装飾するようなタグの埋め込みも簡単だ。これで、Windows風アプリケーションとUNIXフィルタツールの夢の融合が実現できた。

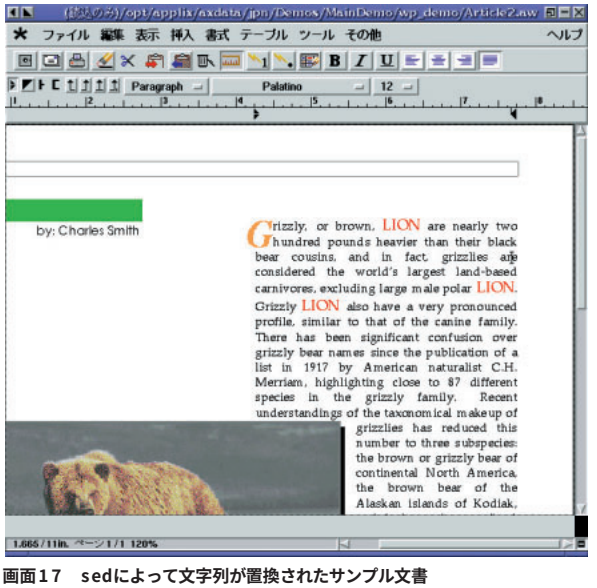
…と、思っていたら、そうは甘くなかった。次にチャレンジしようとした、SpreadsheetsデータのAWKによる整形は、データ構造がCSVと異なりすぎて、挫折してしまった。セパレータを変更すれば、できないことはないだろうが面倒すぎる。これなら、おとなしくSpreadsheetsを使ったほうがよい。

しかも、さらに最も重大な問題が見つかった。それは、「日本語が符号化されている」こと‼　これでは手も足も出ない。使用している符号化アルゴリズムがわかれば、一度日本語に戻してフィルタをかましてから、また符号化して保存するという方法がないではないが、それは現実的ではないだろうと判断した。なんといっても、軽快さがウリのフィルタツールが、これでは意味ないからだ。

かくして、今回は夢破れたわけだが、ぜひ日本語もフィルタツールで操作できるようにしてほしいところだ。マクロを覚えるより簡単だし、なによりもLinuxというプラットフォームに合っていると思う。



画面16　変換前のサンプル文書



画面17　sedによって文字列が置換されたサンプル文書

Applixware体験版の使い方

今月号の付属CD-ROMには、「Applixware 日本語版 4.4.2（体験版）」を収録した。この体験版には、**表1**のような制限事項があるので、あらかじめご了承いただきたい。

■インストール手順

この体験版のインストールは、次のような手順で行う。

（rootユーザーでログインし、konやktermなど日本語表示可能なターミナル上で）

```
# mount /mnt/cdrom
# cd /mnt/cdrom
# cd Applixware/setup/applix/
# ./install
```

このあと、対話式のメニューが起動するので、指示にしたがって、インストールを進めればよい。

なお、途中で［お客様番号］の入力を求められるが、ここには何も入力せずに、リターンキーを押せば次のステップに進むことができる。

■実行

Applixwareを実行する方法は、次のとおりである。

（インストールディレクトリがデフォルトの/opt/applix/の場合）

```
# /opt/applix/applix
```

しばらくすると、［Applixwareメインメニュー］（**画面18**）が起動するので、メニューにあるボタンや左端の［＊］をクリックして表示されるメニュー（**画面19**）から、Applixwareのいろいろなアプリケーションを試用することができる。

なお、一般ユーザーからsuコマンドによって、root権限を取得してインストールを行った場合、インストールが終了してからすぐにApplixwareを起動しようとすると、「環境変数DISPLAYを適切に設定してほしい」という旨のエラーが出ることがある。このときは、一度exitコマンドで一般ユーザーに戻ってから、再度左記の起動コマンドを実行すると、Applixwareが起動するはずである。

■注意事項

なお、Applixware 日本語版 4.4.2（製品版）は、「TurboLinux PRO 日本語版 4.2」にバンドルという形態のみで販売されており、TurboLinux 日本語版以外のディストリビューションでの動作はサポートされていない。そのため、この体験版も他のディストリビューションでは動作しない可能性があることをあらかじめご了承いただきたい。

制限事項	
Words	最初の2ページのみ編集、表示可能
Spreadsheets	20行30列まで編集、表示可能
Presents/Graphic	最初の4スライド／ページまで編集、表示可能
使用期限	2000年4月5日まで

表1　Applixware 日本語版 4.4.2（体験版）の制限事項

画面18　Applixwareメインメニュー

■他のディストリビューションの場合

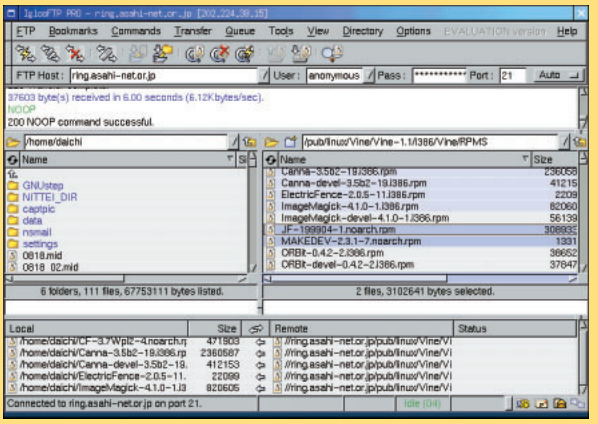
Vine Linux 1.1CRやredhat Linux 5.2など、TurboLinux以外のディストリビューションを使用している場合、そのままではApplixwareの体験版をインストールすることができない（ライブラリ関連のエラーが出る）。libnsl-2.0.7.soをプリロードするという回避方法もあるが、これは、ライブラリ関連の操作が必要であるうえ、他のアプリケーションへの影響も考えられるので、ここでは紹介しない。なお、LASER5 Linux 6.0では、インストール途中でsegmentation faultが出て、インストールできなかった。

編集部としては、TurboLinux 4.x上での試用を強く勧めるが、どうしても他のディストリビューションで試用してみたい場合は、Vine Linuxのオフィシャルサイト（http://vine.flatout.org/）などを参考に、各自の責任で試してほしい。

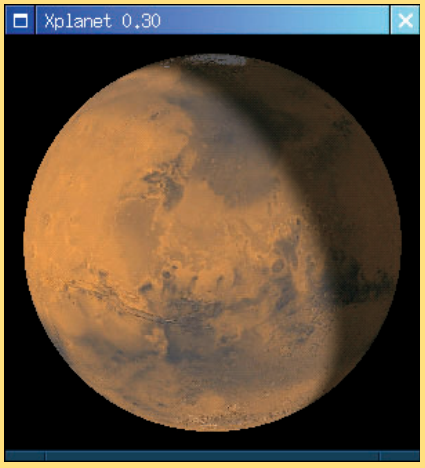
画面19　Applixwareの共通メニュー

Free Application Showcase

文：出井 一
Text : Hajime Dei



IglooFTP PRO　P.106



XPlanet　P.116

glclock　P.117

※紹介したソフトは、すべて付録のCD-ROMに収録されています。

CPU、ビデオ、ディスクのベンチマークを計測

HDBENCH clone

バージョン：0.14.0　　種別：GPL

http://www.enjoy.ne.jp/~gm/program/ihdbench.html

HDBENCH cloneは、有名なWindows用国産ベンチマークソフト「HDBENCH」にそっくりのインターフェイスを持つ、UNIX系OS用の国産ベンチマークソフトだ。CPU、ビデオ、ディスクのベンチマークを計測する。現バージョンではグラフ表示など一部実装されていない機能があるものの、基本的な使い方はHDBENCHと同じで、簡単に各種のベンチマークを計測できる。なお、実行にはGTK+1.2以降が必要だ。

●ビルドから起動まで

HDBENCH cloneはファイル一式をtar＋gzipしたtarボールのほか、RPM形式やDEB形式のバイナリパッケージも配布されている。RedHat系LinuxやDebian GNU/Linuxではこれらのバイナリパッケージを利用したほうが簡単だ。ソースの場合も、「./configure」→「make」→「make install」と一般的な手順でビルドできる。

「hdbench&」として起動すると、いくつかのボタンが並んだウィンドウが開く（**画面1**）。なお、ウィンドウ中の日本語が文字化けする場合は、~/.gtkrcで日本語を含むフォントセットの設定を行う必要がある。

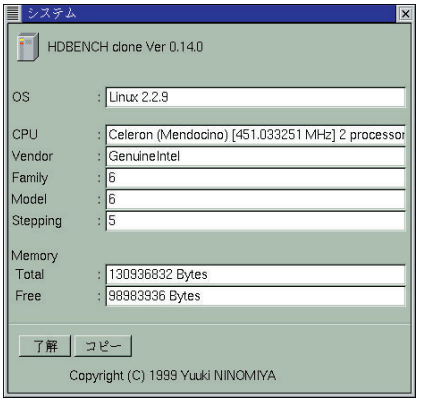
●[ALL]ボタンで計測開始

HDBENCH cloneでは、大きく分けて3種類（CPU、ビデオ、ディスク）のベンチマークを計測できる。これらは、それぞれ独立して計測することもできるし（[CPU]、[VIDEO]、[DISK]ボタン）、まとめて順番に計測することも可能だ（[ALL]ボタン）。

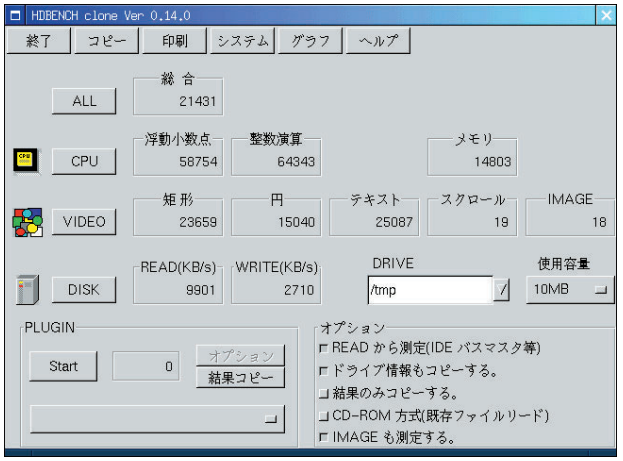
CPUに関しては、1秒間あたりの浮動小数点演算および整数演算の計算回数、メモリ転送速度を計測する。ビデオでは、1秒間あたりの矩形、円、テキストの描画回数（**画面2**）、3秒間のスクロールでの描画回数、5秒間のイメージの描画回数を計測する。ただし、イメージは、65536色表示画面でしか計測できない。また、ディスクについては、1秒間の書き込み・読み込み速度を計測する。計測時に使用するドライブ（マウント後のディレクトリ）とファイルサイズ（1Mバイト～2Gバイト）はユーザーが自由に変更できる。

計測結果は、[コピー]ボタンでクリップボードにコピー、[印刷]ボタンで印刷できる。また、[システム]ボタンを押すと、CPUの種類やメモリ容量などの情報が別ウィンドウに表示される（**画面3**）。

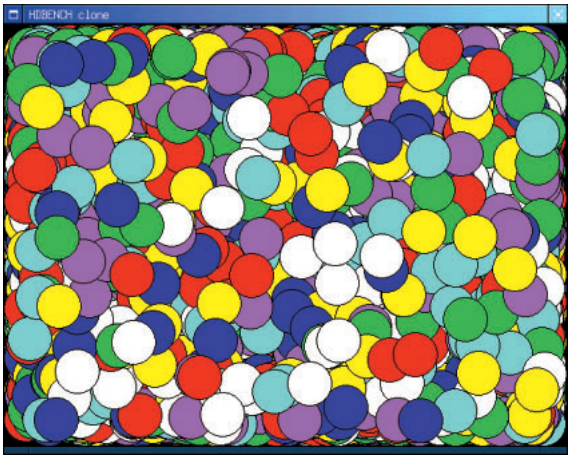
なお、他のマシンの計測結果とグラフを使って比較する機能は、現時点では実装されていない。自分のマシンの相対的な性能を知りたい人は、作者のWebページに、メールで送られてきた計測結果をまとめたページが用意されているので、それらを参照するとよいだろう。



画面3
CPUの種類やメモリ容量などの情報を表示することも可能だ。



画面1
ボタンを押すだけで簡単にベンチマークを計測できる



画面2
1秒間あたりの円の描画回数を計測中。

2パネル構成の高機能なFTPクライアント

IglooFTP PRO

バージョン：0.9.6　　種別：シェアウェア（29.95USドル）

http://www.littleigloo.org/iglooftp.php3

IglooFTP PROは、ローカルとリモートのファイル一覧が同時に表示されるFTPクライアントソフトだ。ツールバーや右クリックメニューによる操作、FTPサイトのブックマークによる管理、すべての設定をGUIで変更できる設定ダイアログなど、とても使いやすいソフトに仕上がっている。試用期間30日のシェアウェアだが、機能制限はいっさいない。フリー版のIglooFTPも配布されている。

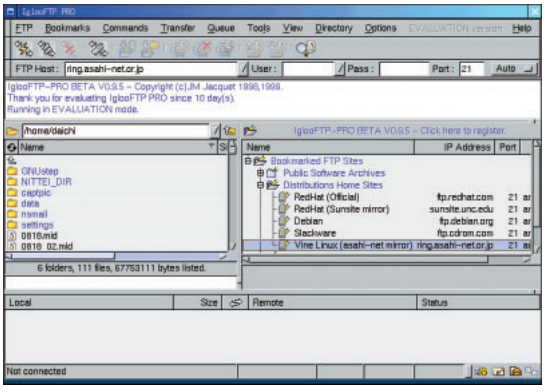
●インストール

IglooFTP PROはシェアウェアのためソースファイルは用意されておらず、RPM形式およびtarボールのバイナリパッケージのみ配布されている。RedHat系Linuxを使っている人は、RPMパッケージからのインストールがお勧めだ。tarボールのほうも、ファイルを展開後に（rootになって）「./Install」とするだけで、各種ファイルのインストールを行ってくれる。

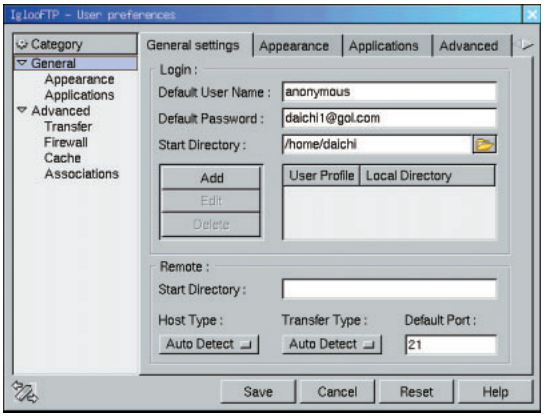
ソースが提供されないと、日本語の表示や入力への対応が気になるところ。実際には、GTK+のlocale設定が正しく行われており、そのままの状態でファイル一覧や内蔵ビューア、エディタで日本語を扱える。

●起動と初期設定

ktermなどのコマンドラインで「IglooFTP-PRO&」として起動すると、複数のパネルで構成されるウィンドウが開く（**画面1**）。左側のパネルにはローカルのディレクトリのファイル一覧、右側のパネルにはFTPサイトのブックマークがツリー表示される。ブックマークには、RedHatやDebianなどのディストリビューション、GTK+やGnomeプロジェクトのFTPサイトなどが、フォルダで分類されている。



画面1
接続前は、右のパネルにブックマークがツリー表示される。



画面2
この設定ダイアログですべての設定を変更可能だ。

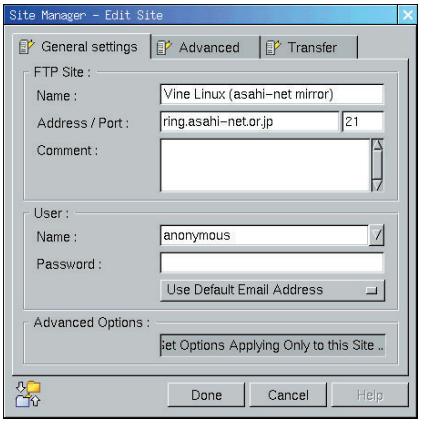
まずは初期設定を行おう。ツールバー左から4番目の[Set Preferences]ボタンを押すか、[Options]-[Preferences]を選択すると、ユーザー設定ダイアログが開く（**画面2**）。

設定ダイアログはカテゴリ別にページ分けされおり、IglooFTP PROのすべての設定をこのダイアログで変更できる。とりあえず、[General settings]ページで、デフォルトのユーザー名（anonymous）とパスワード（メールアドレス）、ローカル・リモートの初期ディレクトリを設定しよう。

●ブックマークの登録

続いて、頻繁に利用するFTPサイトのブックマークを登録する。右側のパネルのツリーからFTPサイトを選択し、右クリックメニューから[Add Site]をクリックする。サイトマネージャダイアログ（**画面3**）が表示されたら、FTPサイト名やアドレス、アカウントなどを設定する。

パスワードについては、「Save



画面3
FTPサイトの情報はブックマークで管理される。

Password（設定して保存）」「User Default Email Address（デフォルトメールアドレスを使用）」「Ask for Password on Connect（接続時に入力）」の中から選択できる。誰でもアクセスできるアノニマスFTPサーバの場合は、「デフォルトメールアドレスを使用」にしておけばいい。

なお、一番下にある[Set Options Applying...]を押すと、ダイアログに[Advanced][Transfer]ページが追加され、接続時の初期ディレクトリなどをサイトごとに設定できる。

●接続と基本的な操作

ブックマークを登録したFTPサイトについては、ブックマークをダブルクリックするだけで接続される。一方、一時的に訪れるFTPサイトの場合は、ホスト名やアカウントなどの情報をツールバー下のエリアに直接入力すればいい。

いずれにせよ、FTPサイトへの接続が成功すると、ウェルカムメッセージが表示されるとともに、ウィンドウ右側のパネルの表示が、接続先のFTPサイト（リモート）のファイル一覧に切り替わる（**画面4**）。

サブディレクトリに移動するには、ディレクトリ名をダブルクリックする。また、左右のパネルの左上に、ひとつ上のディレクトリに移動するボタンが用意されている。

ファイルの選択は、単にマウスでクリックするだけでいい。このほか、Shift-クリックやドラッグによる範囲選択や、Ctrl-クリックによる複数選択にも対応している。

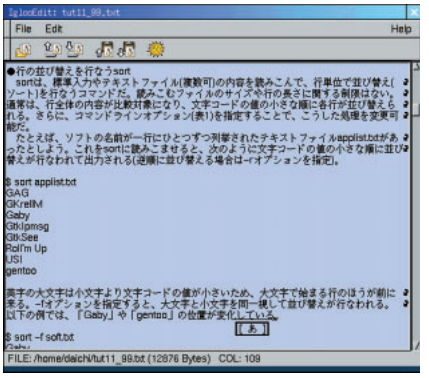
ローカル／リモートどちらのファイルに対しても、右クリックメニューを利用して、選択したファイルの閲覧（**画面5**）や編集、削除、属性変更などが可能だ。IglooFTP PROに付属するエディタ「IglooEdit」（**画面6**）は、基本的な機能しか持たないものの、日本語の表示や入力にも対応しており、ちょっとしたファイルの編集には十分だ。機能に満足できなければ、XEmacsやviなどの使いなれた外部エディタを利用する設定も可能だ。

●2種類のファイル転送方法

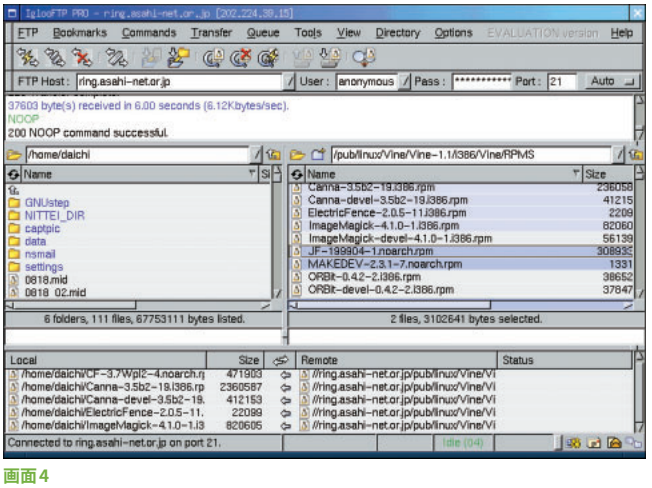
FTPクライアントの要といえるファイル転送に関しては、選択したファイルを即座に転送する方法と、ファイルの情報をいったんキューに溜めておき、あとでまとめて一括転送する方法が用意されている。

即座に転送する場合は、ファイルを選択した状態で、ツールバーの[Upload Selection]や[Download Selection]ボタンを押すか、右クリックメニューの[Upload]や[Download]を選択すればいい。

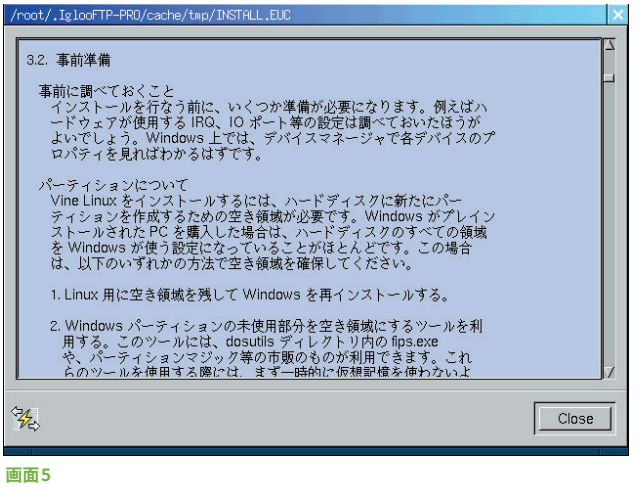
一括転送する場合は、ツールバーの[Add to Queue]ボタンや、右クリックメニューの[Queue]を選択してキューに登録する。登録したファイルはウィンドウ下のエリアに表示され、順番の変更や削除などが可能だ。[Transfer Queue]ボタンを押すと、登録されたファイルが一括転送される。また、キューの内容をいったんファイルに保存し、余裕のあるときにロードして転送を行うという使い方も可能だ。



画面6
日本語の表示や入力に対応しているエディタ「IglooEdit」。



画面4
接続中はローカル・リモートのファイル一覧を左右に表示する。



画面5
ファイル閲覧やウェルカムメッセージの表示に使われるビューア。

画像閲覧に便利なファイルマネージャ

Endeavour

バージョン：1.05　　種別：GPL

http://fox.mit.edu/xsw/edv.htm

Endeavourは、黒い背景がクールなファイルマネージャ。外見に似合わず、ファイルの削除などを一律に禁止するマスターライトプロテクトや、削除ファイルを復活できるゴミ箱を装備するなど親切な設計だ。また、イメージビューアやブラウザを内蔵しており、画像の閲覧にも向いている（imlibが必要）。日本語には対応していないが、外部ソフトを起動できるため日本語を含むファイルの表示や編集は問題なく行える。

●ビルドとインストール

Endeavourは、ファイル一式をtar＋gzipしたtarボールで配布されている。ビルド手順は簡単で、ファイルを展開後、endeavourディレクトリに移動してから、「make」→「make install」とすればいい。

ただし、Makefile内のライブラリ検索パスとインストール先の記述が、それぞれ「/usr/X11/lib」（20行目）と「/usr/X11/bin」（82行目）となっている。Linuxのディストリビューションでは、通常は「X11」ではなく「X11R6」というディレクトリ名が使われるので、このままでは正常にビルドとインストールを行えない。両者の記述を「X11R6」に修正するか、あらかじめ「ln -s /usr/X11R6 /usr/X11」としてリンクを張っておこう。

●起動と初期設定

「endeavour&」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ブラウザウィンドウが開く（**画面1**）。同じ作者グループ（Wold Pack）によるゲーム「XShipWars」（http://fox.mit.edu/xsw/）のウィンドウ部品（ウィジェット）が使われているため、サイバーな香りが漂う。

表示するディレクトリを切り替えるには、ツリー上のディレクトリをクリックするか、[Location:]エリアに直接ディレクトリ名を入力すればいい。複数のウィンドウを開いて、異なるディレクトリの内容を同時に参照することも可能だ。

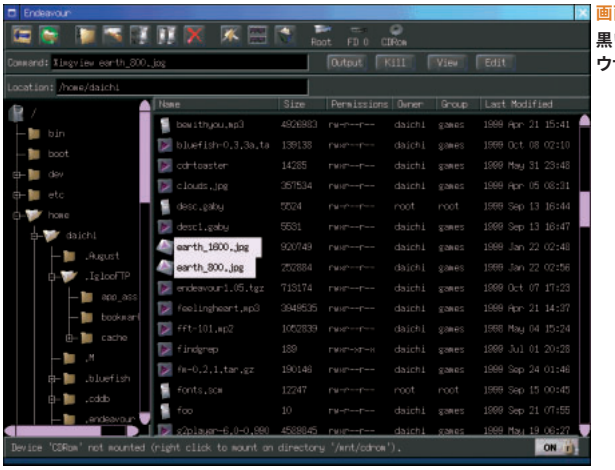
最初の起動時に、ホームディレクトリに設定ファイル（.endeavour/endeavourrc）が作成される。初期設定ではテキストの閲覧、編集にnxterm上のpicoが使われるので、これをkterm上のlessやMuleで置き換えてみよう。

使用するテキストエディタをMuleに変えるには、177～178行目の「TextEditCommand = …」という部分を、「TextEditCommand = mule -fg "#ffffff" -bg "#000000" "%t"」とすればいい（%tは実行時にファイルのフルパス名に置換される）。また、テキスト（拡張子txtなど）の内容をkterm上のlessで表示するには、336行目の「nxterm」を「kterm」に、337行目の「pico」を「less」に変更すればいい。

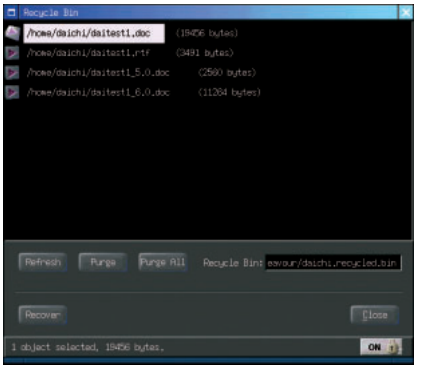
なお、設定ファイルの内容はEndeavourの起動時に読み込まれるので、変更した設定を有効にするには、いったんEndeavourを終了して再起動する必要がある。

●ファイルの選択と操作

ファイルの選択はクリックで行う。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。選択したファイルに対しては、ツールバーのボタンや右クリックメニューにより移動やコピー、削除（リサイクル）を行える。このほか、ドラッグ&ドロップにも対応しており、選択したファイルをドラッグし、フォルダツリーや別のEndeavourのウィンドウ



画面1
黒い背景がクールなEndeavourのブラウザウィンドウ。



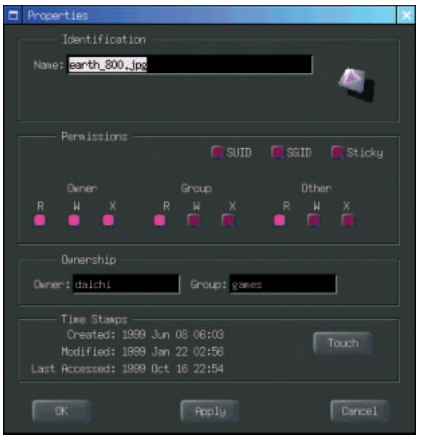
画面2
削除（リサイクル）したファイルはゴミ箱に保管されている。

上でドロップすることでファイルを移動できる。

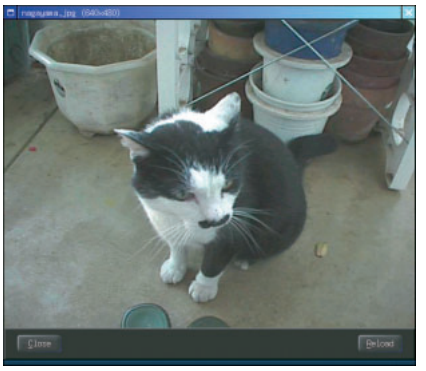
ただし、右下のマスターライトプロテクトボタンの表示が「ON」の状態では、ファイルの移動、コピー、削除といった操作は一律に禁止される。ファイル操作の前にボタンを押して「OFF」に切り替え、操作後は「ON」に戻しておけば、操作ミスの多くを避けられるだろう。

なお、削除（リサイクル）したファイルは、ゴミ箱の中に保存されている。ツールバーのゴミ箱ボタンを押すと、その内容が一覧表示され、削除したファイルの復活や一部/完全消去などの操作が可能だ（**画面2**）。

また、右クリックメニューの[Properties]を選択すると、選択したファイルに関する情報が表示され、ファイル名、属性、所有者、アクセス日時などを変更できる（**画面3**）。



画面3
ファイル名や属性などを確認・変更できるプロパティ画面。



画面4
JPEG/GIF/TIFF/BMP形式などに対応しているイメージビューア。

●ダブルクリック時の動作

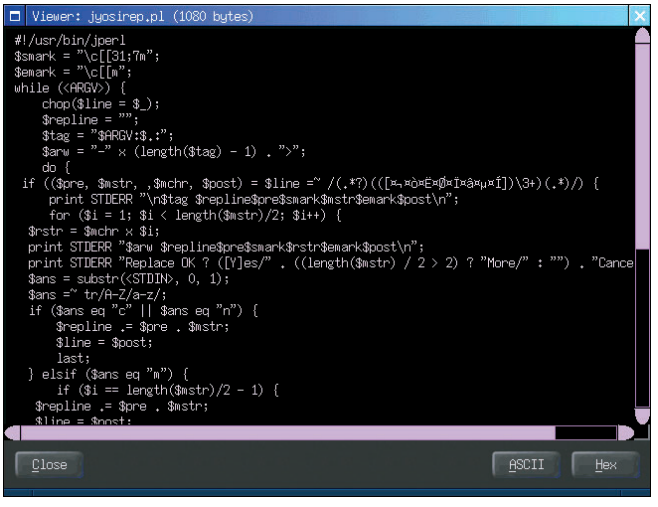
Endeavourでは、ファイルを選択した時点で、ウィンドウの[Command:]エリアに拡張子ごとに異なるコマンドラインが表示され、ファイルをダブルクリックするとその内容が実行される（内容を書き換えることも可能）。

ファイルの拡張子と実行するコマンドラインの関連付けは、設定ファイルで行う。テキスト、オーディオ、HTMLファイルなどについては、最初から設定済みだ。たとえば、HTMLファイルの場合はNetscapeが起動する。オーディオ関連で使われるコマンド（wavplay/mxaudio/drvmidi）は一般的ではないので注意されたい。

設定ファイルで拡張子が関連付けられていないファイルのうち、画像ファイルについては、幅広い画像形式に対応したイメージビューア（%imgview）で表示される（**画面4**）。

その他のファイルの場合は、テキスト表示と16進ダンプ表示を切り替え可能な汎用ビューア（%view）が使われる（**画面5**）。どちらもEndeavourの内部コマンドだ。

内部コマンドには、このほかにも、サムネイル（縮小画像）を一覧表示するイメージブラウザ（%ib）があり、Endeavourのコマンドラインに「%ib」を入力することで起動できる（**画面6**）。左側にサムネイル一覧、右側に選択したファイルの（サムネイルより大きな）画像が表示され、ダブルクリックでイメージビューアが起動して本来のサイズの画像を表示する。



画面5
テキスト表示と16進ダンプ表示を切り替え可能な汎用ビューア。

画面6
サムネイルを一覧表示するイメージブラウザは「%ib」で起動。

さまざまなデジカメの画像を扱える画像ソフト

gPhoto

バージョン：0.4.0　　種別：GPL

http://www.gphoto.org/

gPhotoは、主にデジタルカメラの画像を扱うことを目的に作られた画像ソフトだ。対応しているデジカメは、現時点で90機種近くにも及ぶ。ケーブル接続されたデジカメに対し、ハードディスク上のディレクトリを扱う感覚で、サムネイル（縮小画像）や画像の取得、削除などを行える。取得した画像を好きな形式で保存したり、プリンタで印刷することも可能。なお、実行にはGTK+とimlibが必要だ。

●ビルドとインストール

gPhotoはファイル一式をtar＋gzipしたtarボールとRPM形式の両方で配布されている。RedHat系Linuxでは、RPMのバイナリパッケージを利用するのが最も簡単だが、ファイルオープンや保存時のダイアログで使われている日本語が文字化けする。

文字化けが気になる人は、tarボールを利用してソースを修正しよう。ファイルを展開後、src/main.cの72行目、

```
gtk_init(&argc,&argv);
```

の前に、

```
gtk_set_locale();
```

を1行追加する。ビルド手順は、「./configure」→「make」→「make install」という一般的なものだ。

●起動と初期設定

ktermなどのコマンドラインで「gphoto&」として起動すると、ロゴを含んだウィンドウが開く（**画面1**）。初めて起動したときには、デジカメの機種とシリアルポートを選択して保存する必要がある。

[Configure]-[Select Port - Camera Model]で設定ダイアログが開く（**画面2**）。[Camera Model]のリストから自分の使っているデジカメを選択し、[Port]の中からデジカメ用のケーブルを接続しているシリアルポートを選択しよう。最後に[Save]ボタンを押して設定ファイルに保存する。

続いて、パソコンとデジカメをケーブルで接続し、デジカメの電源を入れた状態で、[Camera]-[Camera Summary]を選択する。カメラの状態（現在のモードなど）が表示されれば、正常に接続されている（**画面3**）。「Error Opening Camera」と表示される場合は、再度設定を確認してみよう。

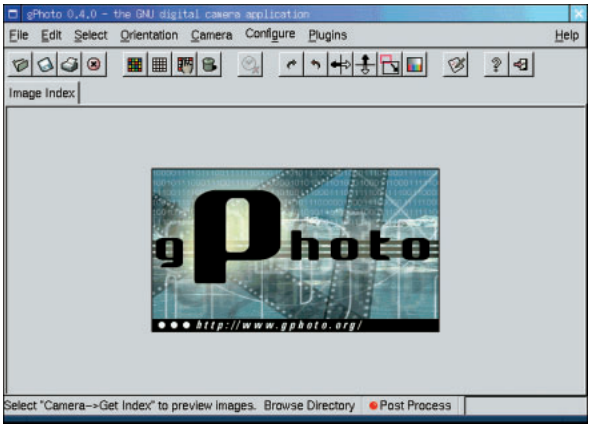
もし、rootではカメラに接続できるのに、一般ユーザーではできないという場合、シリアルポートへの一般ユーザーのアクセスが許可されていないことが原因だ。これは、rootになった状態で「chmod 666 /dev/ttyS1」（ttyS1はシリアルポート2のデバイス名）などとすることで解消できる。

セキュリティ保護のため一般ユーザーのアクセスを許可できないという場合、gPhotoを利用するユーザーのグループを新規に作成し、シリアルポートの所有グループにするという方法もある。詳細については付属のFAQを参照されたい。

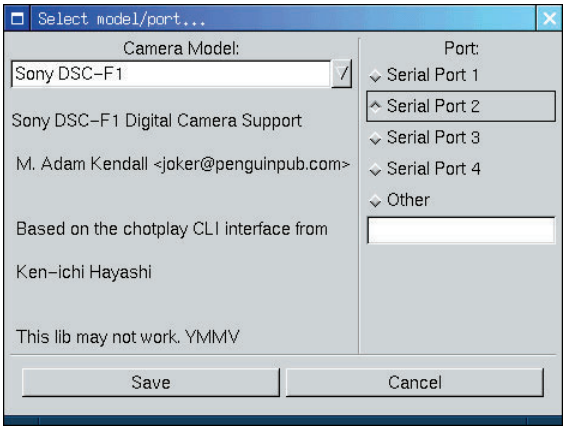
●デジカメの画像を扱う

デジカメで撮影した画像をgPhotoで取得するには、最初にサムネイル（縮小画像）の一覧を取得してインデックスとして使用し、取得したい画像を選択するという手順をとる。

サムネイル一覧は、ツールバー左から5番目の[Get Thumbnail Index]ボタン（またはCtrl-Iキー）で取得できる。しばらく待っていると、撮影画像



画面1
起動後のウィンドウ。まずはカメラの設定を行う。



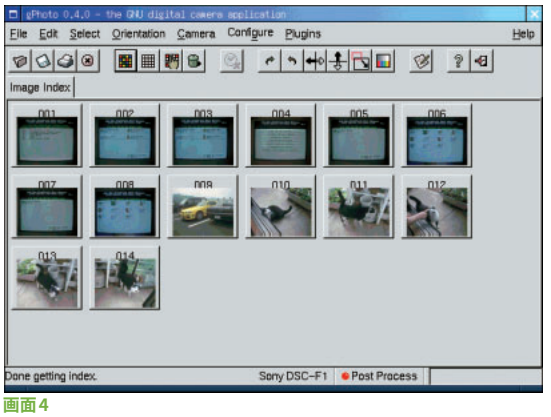
画面2
デジカメの種類とケーブルを接続するポートを設定する。

のサムネイルがインデックス内部に表示されるはずだ（**画面4**）。取得したい画像のインデックスがあらかじめわかっている場合は、[Get Empty Index]ボタン（またはCtrl-Eキー）で、サムネイルなしのインデックスを使うといい。サムネイルの読み込みが省略される分、素早く作業を行える。

これらのサムネイルを見ながら（あるいはインデックス番号を頼りに）、フルサイズ画像を取得したいものをクリックで選択する（複数可）。選択されたインデックスは背景色が赤く変化する。続いて、ツールバーの[Get Selected Images]ボタンを押すと、選択したインデックスのフルサイズ画像がデジカメから取得され、独立したページに表示される（**画面5**）。なお、取得したい画像が1つだけなら、いちいち選択やボタン操作を行う必要はなく、インデックスをダブルクリックするだけで取得できる。

それぞれの画像に対しては、ツールバーのボタンを使って、左右・上下反転や90度回転、サイズ変更、カラーバランス調整（**画面6**）といった画像処理が可能だ。画像をディスク上に保存するには、ツールバーの[Save Opend Image（s）]ボタンを押して、保存するファイル名を指定する。保存時の画像形式はJPEG／GIF／TIFF／PNG／BMPなど幅広く対応しており、ファイルの拡張子により自動的に判別される。

画面3
カメラサマリーで正常に接続されているか確認しよう。



画面4
デジカメのサムネイル一覧が[Image Index]ページに表示される。

●ディスク上の画像も扱える

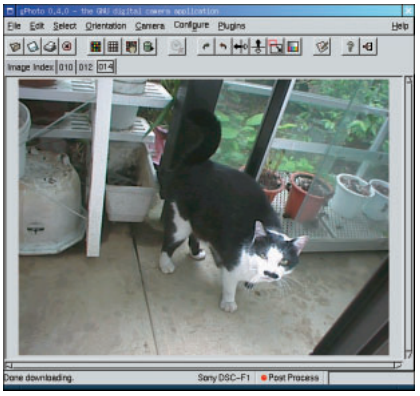
gPhotoは、ディレクトリのサムネイル一覧から選択した画像を表示するという、一般的な画像管理ソフトとして使うことも可能だ。

[Open]-[Directory]（またはCtrl-Oキー）でディレクトリ選択ダイアログを開いてディレクトリを指定すると、そのディレクトリに含まれる画像のサムネイルが一覧表示される。フルサイズ画像の取得はデジカメの場合と同じで、インデックスを選択した後で[Get Selected Images]ボタンを押すか、インデックスを直接ダブルクリックすればいい。

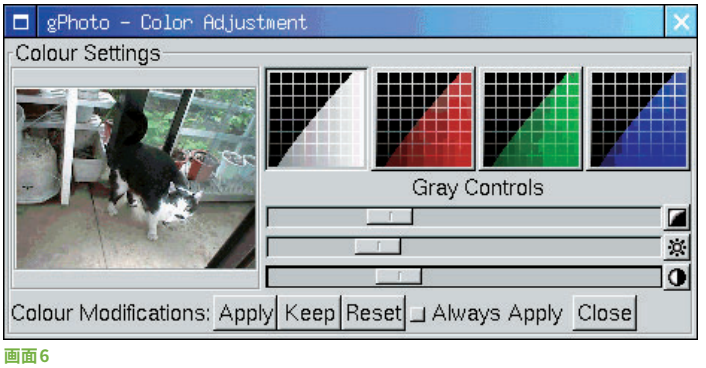
ただし、ディレクトリ内のファイルをすべて画像ファイルとみなして読み込もうとするため、画像以外のファイルがあるとエラーメッセージが表示され、インデックスの表示に時間がかかるので注意されたい。

なお、ディスクに対するインデックス表示は、仮想デジカメドライバ（Browse Directory）を利用しているので、設定ダイアログ（**画面2**）の[Camera Model]の設定が一時的に変更されてしまう。gPhotoを終了しないままデジカメの画像を扱う際には、[Camera Model]を設定し直す必要がある。変更は設定ファイルには保存されないので、再起動する場合には再設定は不要だ。

デジカメを持っていなかったり、主にディスク上の画像を扱いたい場合には、逆に[Camera Model]を「Browse Directory」に設定してセーブしておくとよいだろう。



画面5
フルサイズの画像はインデックス番号のページに表示される。



画面6
画像のカラーバランスを変更することも可能だ。

CDリッパなどと連動するCDプレーヤ

Grip/GCD

バージョン：2.7　　種別：GPL

http://www.nostatic.org/grip/

Gripは、CDリッパやMP3エンコーダと組み合わせて、MP3作成のフロントエンドとして使えるCDプレーヤソフトだ。CDリッパなどのフロントエンド部分を持たないGCDも用意されている。いずれもCDDBに対応しており、インターネット上のCDDBサーバから音楽CDのタイトルや曲目情報を取得可能。画面表示の各部分は必要に応じて表示をオン／オフできる。実行にはGTK+1.2以降が必要だ。

●ビルドとインストール

Grip/GCDはファイル一式をtar＋gzipしたtarボールのほか、RPM形式やDEB形式のパッケージでも配布されている。なお、RPM形式のバイナリパッケージはRedHat6.x用（要glibc2.1）なので、glibc2.0採用のディストリビューションではソースパッケージをリビルドするとよいだろう。

曲目の表示などに日本語を使いたいときは、tarボールを展開してソースを書き換える必要がある。grip.cの5093行目、

```
gtk_init(&argc,&argv);
```

の前に、

```
gtk_set_locale();
```

を1行追加してビルドすればいい。また、Makefileの7行目「PREFIX=/usr」を「PREFIX=/usr/local」に変更しておこう。

ビルド自体は、「make」→「make install」とするだけなので難しくはない。ただし、内蔵用のCDリッパ（音楽CDからWAVEファイルを作成するソフト）として組み込まれるcdparanoiaのライブラリ部分をあらかじめビルドしておく必要がある。MP3作成時には、コマンドラインベースのMP3エンコーダ（bladeencなど）とID3タグツール（mp3info）が別途必要だ。

●CDプレーヤとして使う

「grip&」としてGripを起動すると、上部に曲名などが並んだトラックリスト、下部に操作用のボタンが並んだウィンドウが開く（**画面1**）。トラックリストのエリアは、上部のタブによりリップや設定変更、バージョン表示などに切り替えられる。

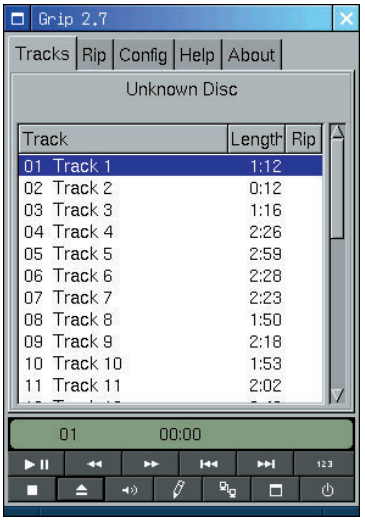
再生やスキップなどのボタンは説明するまでもないだろう。このほか、タイトルや曲名を入力するディスクエディタ、ボリュームコントロール、プレイモードコントロールなどが用意されており、それぞれボタンで表示をオン／オフできる（**画面2**）。

一方、「gcd&」で起動するGCD（画面3）は、GripからCDリッパやMP3エンコーダのフロントエンド部分を取り去ったもの。それ以外のGripの機能はすべて備わっている。ファイルサイズがGripの約半分と小さいため、純粋にCDプレーヤとして使うだけならこちらのほうがお勧めだ。

なお、rootでは音楽CDを再生できるのに、一般ユーザーではできないのは、CD-ROMドライブへの一般ユーザーのアクセスが許可されていないことが原因だ。これは、rootになった状態で「chmod 666 /dev/hdc」（/dev/hdcはCD-ROMドライブのデバイス名）とすることで解消できる。

●CDDBとディスクエディタ

Grip/GCDは、インターネット上の



画面1
トラックリストとボタンのみ表示される起動直後のGrip。

画面2
ディスクエディタやボリュームコントロールなどを表示する。

画面3
MP3のフロントエンド部分を取り除いたGCD。

CDデータベース、CDDB（http://www.cddb.com/）での検索に対応している。音楽CDをCD-ROMドライブに挿入すると、ローカルなデータベースからタイトルや曲目の情報を検索し、見つからない場合は自動的にインターネット上のCDDBサーバに接続して検索する。

CDDB関連機能は、起動時に-l（小文字のエル）オプションを付ければ無効になる。また、ボタンを押した場合にだけCDDBサーバに接続するような設定（後述）も可能だ。

なお、CDDBで使える文字セットはISO-8859-1のみなので、日本語の曲名はローマ字表記で送られてくる（画面4）。このため、日本語のタイトルや曲目を使いたい場合は、自分で設定する必要がある。もちろん、該当するCDの情報が見つからない場合も同様だ。

ペンのボタンを押してディスクエディタを表示し、タイトル・アーティスト・ジャンル・発表年・曲目（トラック）を設定しよう。ディスクエディタ右下のフォルダボタンを押すと、ローカルなデータベースに保存される。CDDBサーバに情報を送信することも可能だ（ISO-8859-1のみ）。

●設定変更はGUIで行う

ウィンドウ上部の[Config]をクリックすると設定ページに切り替わる。このページ内にはさらに複数のページが用意され、CD演奏時の設定をはじめ、外部ソフト、CDDB、プロキシなどの設定がジャンル別に分類されている。たとえば、CDDBサーバへの自動接続を無効にするには、[CDDB]ページの[Perform CDDB lookup automatically]のチェックを外せばいい。

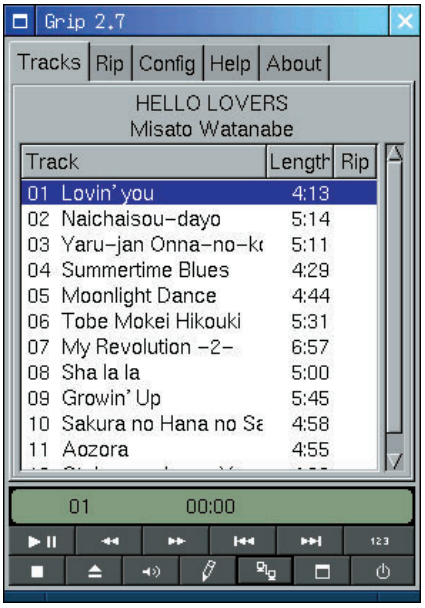
CDリッパやMP3エンコーダ、ID3タグツールについては、使用するコマンドとコマンドラインを設定する。内蔵CDリッパやコマンドラインベースの外部ソフトを利用可能だ。cdparanoiaやbladeencといった定番ソフトについては、コマンドラインが最初から設定されている。

その他のソフトでは、ユーザーがコマンドラインを設定する必要がある。たとえば、国産の高速MP3エンコーダ「午後のこ～だ」（http://www.kurims.kyoto-u.ac.jp/~shigeo/）の場合、実行ファイルを「gogo」、コマンドラインを「%f -b %b」にすればいい（**画面5**）。%fと%bは、実行時にファイル名とビットレートに置換される。

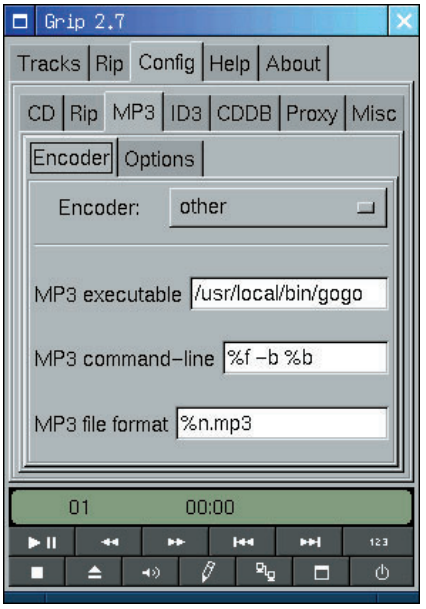
●CDリッパやMP3プレーヤと連動

MP3作成のフロントエンドとして使うには、まずトラックリストからリップしたいものを右クリックで選ぶ（複数可）。選択したトラックにはチェックマークが表示される。続いて、[Rip]ページに切り替えて、[Rip+Encode]ボタンを押すと、自動的にWAVEファイルの作成（リップ）とMP3エンコード作業を行ってくれる（**画面6**）。

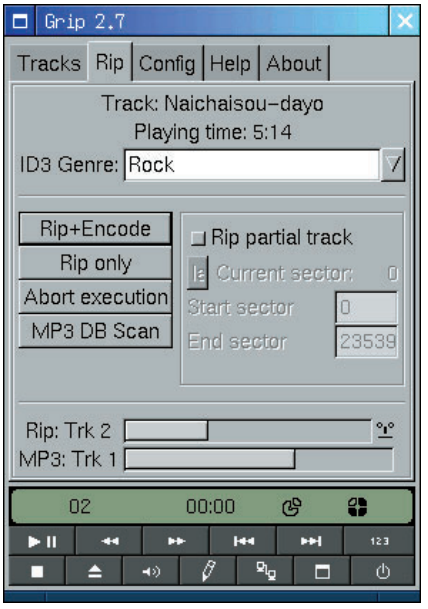
なお、WAVEファイルやMP3ファイルは、ホームディレクトリの「mp3/アーティスト名/タイトル名」というサブディレクトリに作成され、ファイルのベース名（拡張子を除いた部分）にはトラック名が使われる。日本語を使用する場合には注意されたい。



画面4
CDDBから曲目情報などを取得。日本語の曲目はローマ字表記だ。



画面5
設定ページ内の[MP3]ページで「午後のこ～だ」を指定。



画面6
ボタンをひとつ押すだけで自動的にMP3ファイルの作成が可能。

高機能なファイル検索をGUIで利用する

gtkfind

バージョン：1.1　　種別：GPL

http://www.oz.net/~mattg/download.html

gtkfindは、ファイル名、日時、属性、サイズなど、さまざまな条件によるファイル検索を行うソフトだ。コマンドラインで複雑なオプションを指定しなければならないfindコマンドに比べ、gtkfindでは検索条件をGUIで設定できるので、初心者でも複雑な検索を簡単に行える。検索結果はウィンドウに表示するだけでなく、標準出力に出力することも可能だ。実行にはGTK+が必要となる。

●ビルドと起動方法

gtkfindはファイル一式をtar＋gzipしたtarボールで配布されている。ビルド手順は、「./configure」→「make」→「make install」という一般的なもの。なお、gtkfind.cの56行目、

```
gtk_init(&argc, &argv);
```

の前に、

```
gtk_set_locale();
```

を1行追加してビルドすると、GTK+の設定ファイル（/etc/gtk/gtkrc.jaなど）で設定した日本語を含むフォントセットが使われるようになる。

gtkfindを単独で使用する場合は、ktermなどのコマンドラインで「gtkfind&」として起動する。一方、gtkfindをfindコマンドの代わりとして使う場合には、

```
$ gtkfind -vanish | sort
```

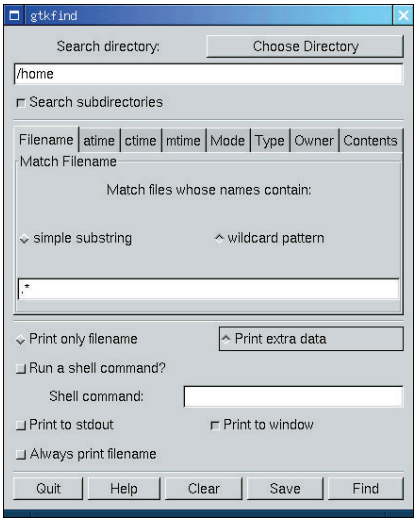
のように、「|」（パイプ）を使って他のコマンドと接続し、（後で説明するように）検索結果を標準出力へ出力するよう設定すればいい。なお、「-vanish」は、検索後にgtkfindを自動的に終了するオプションだ。

●検索開始ディレクトリの設定

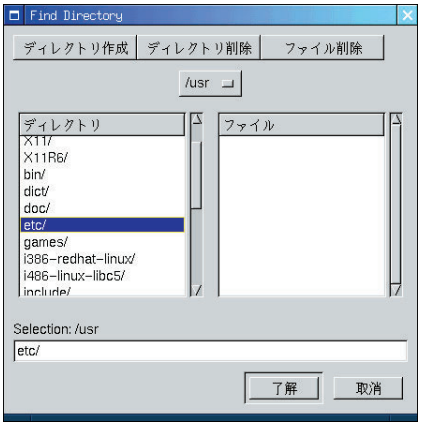
gtkfindのウィンドウには、上部に検索ディレクトリ、中央部にさまざまな検索条件を設定する「カード」、下部には検索後の処理の選択や検索開始用ボタンなどが並んでいる（**画面1**）。どこから設定しても構わないが、以下では上から下に向かって順番に項目を設定していこう。

まずは、[Search directory]に、検索開始ディレクトリを1つだけ入力する。[Choose Directory]ボタンで、ファイル選択ダイアログから選択することも可能だ（**画面2**）。

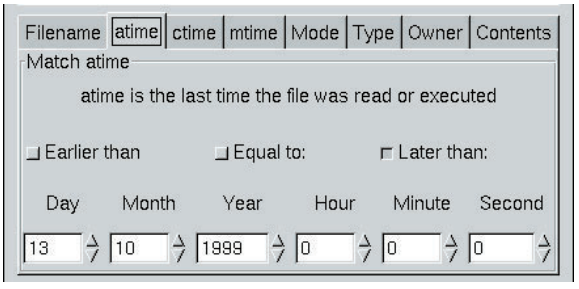
なお、通常はサブディレクトリのファイルも検索対象だが、[Search subdirectories]のチェックを外すと、指定したディレクトリのファイルだけが検索対象となる。



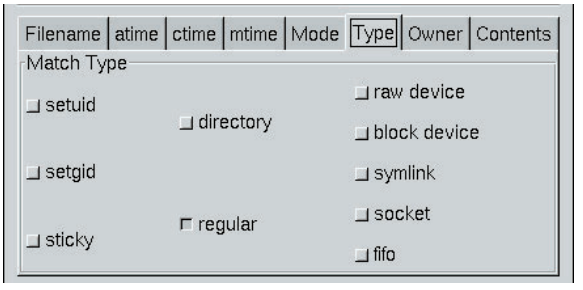
画面1
さまざまな検索条件をGUIで指定できる。



画面2
検索開始ディレクトリはダイアログからも選択可能。



画面3
[atime]カードでは、最終アクセス日時に関する検索条件を設定。



画面4
[Type]カードでは、ファイルの種類に関する検索条件を設定する。

●検索条件の設定

続いて、中央部の「カード」を切り替えて検索条件を設定する。gtkfindには、8種類の検索条件が用意されている（**表1**）。

複数のカードで条件を設定すると、それらをすべて満たすファイルだけが検索される。たとえば、「昨日から現時点までにアクセスされたドットファイル」を検索するには、[Filename]カードで[wildcard pattern]をチェックしてファイル名に「.＊」を設定し、[atime]カードで[Later than]をチェックして昨日の0時0分を設定、さらに[Type]カードで[regular]をチェックすればいい。

なお、ウィンドウ下の[Clear]ボタンを押すと、これまでの設定がすべて破棄され、あらためて検索条件を一から設定できる。

●検索結果に対する処理の設定

最後に、検索条件にあてはまるファイルに対してどのような処理を行うかを設定し、[Find]ボタンを押して検索を実行する。

ファイル一覧を表示するだけではなく、外部コマンドの実行や、検索結果を標準出力に出力してパイプで他のコマンドに渡すなど、さまざまな使い方が可能だ。

初期設定では、検索されたファイル名が独立したウィンドウに表示される（**画面5**）。[Save]ボタンでファイルに保存することも可能だ。[Print extra data]をチェックすると、ファイル名のほか、ファイルサイズや属性など詳しい情報がlsコマンドの形式で表示されるようになる（**画面6**）。

検索したファイルに対してコマンドを実行する場合は、[Run a shell command?]をチェックして、コマンドを含んだコマンドラインの内容を[Shell command]に設定する。コマンドライン中では、検索ファイル名を「\0」で参照可能だ。たとえば、「rm \0」とすれば、検索したファイルをrmコマンドで削除できる。

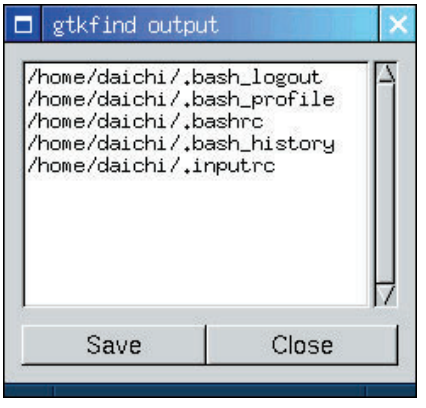
もし、ファイル名の一部だけを参照する必要があるなら、[Filename]カードでファイル名パターンを指定する際に、その部分をあらかじめカッコで囲んでおかなければならない。カッコで囲んだ文字列は、左から順に「\1」「\2」…で参照可能だ。

たとえば、「検索したファイルの拡張子jpgをjpegに変更する」という場合は、まずファイル名パターンに「(＊).jpg」を設定する。これにより拡張子を除く部分を「\1」で参照できるので、コマンドラインには「mv \0 \1.jpeg」と設定すればいい。

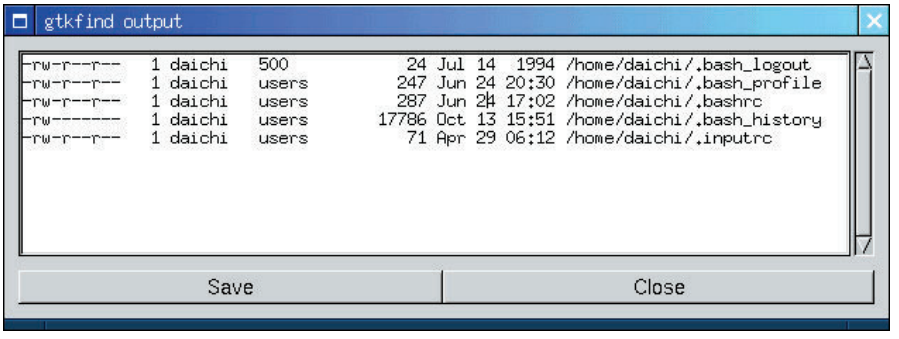
このほか、「ビルドと起動方法」で説明したように、findコマンドの代わりに他のコマンドとパイプで接続して使う方法もある。この場合、検索結果が標準出力に出力されるように、[Print to stdout]をチェックする必要がある。通常は、[Print to window]のチェックを外して、結果のウィンドウ表示を抑止しておくとよいだろう。両方チェックしておき、ウィンドウ表示を確認に使うことも可能だ。

カード名	条件
Filename	ファイル名に含まれる文字列
atime	ファイルを最後にアクセスした日時（画面3）
ctime	最後にファイルのステータスを変更した日時
mtime	ファイルを最後に変更した日時
Mode	ファイルの属性
Type	ファイルの種類（画面4）
Owner	ファイルの所有者
Contents	ファイルの内容に含まれる文字列

表1 検索条件



画面5
初期設定では、検索されたファイルが別ウィンドウに表示される。



画面6
検索ファイルに関する詳しい情報を表示することも可能だ。

ルートウィンドウにリアルな地球を表示

XPlanet

バージョン：0.30　　種別：GPL

http://www.alumni.caltech.edu/~hari/xplanet/

XPlanetは、Xのルートウィンドウ（背景）に地球の画像を表示するソフト。各種のマップ画像を利用して、平面や球面に投影された地球（またはその他の惑星・衛星など）を夜の影や雲も含めてリアルに表示する。一定時間ごとにXPlanetを呼び出して再描画を行うコマンドも同梱されている。また、通常のウィンドウに表示した場合には、リアルタイムに画像を回転できる（Mesaライブラリが必要）。

●ビルドとマップ画像の入手

XPlanetはファイル一式をtar＋gzipしたtarボールとRPM形式の両方で配布されている。RedHat系LinuxではRPM形式のバイナリパッケージを使うといいだろう。実行にはOpenGL互換ライブラリのMesaが必要だ。Mesaを利用できない場合、tarボールを展開し、ソースやMakefileの一部を修正する必要がある。ビルド自体は、「./configure」→「make」→「make install」と簡単なものだ。

マップ画像は、「XGLOBE & XPLANET MAPS」（http://www.radcyberzine.com/xglobe/）や「Ssystem」（http://plasma.gsfc.nasa.gov/Ssystem/）といったWebページから別途取得する。前者には数種類の地球の画像、後者には雲や月、火星の画像がある。

ルートウィンドウと同じか少し大きい程度の画像を使うと、作成される画像の質がよくなる。ただし、大きな画像を扱うと大量にメモリを消費するので、Gimpなどを使って適当に縮小してから使うとよいだろう。

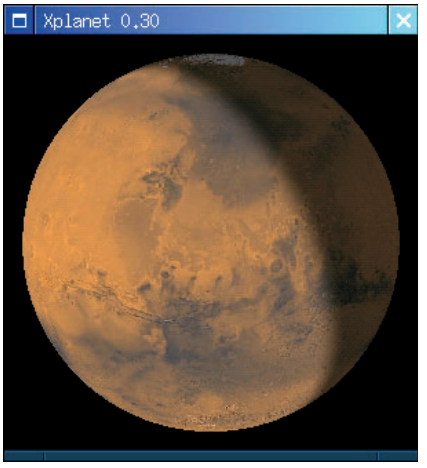
●オプションで画像などを指定

Xのルートウィンドウ（背景）に地球の画像を表示するには、「xplanet --image earth_800.jpg --or」のように、--imageオプションで地球のマップ画像のファイル名を指定する。しばらくすると、ルートウィンドウに丸い地球が表示される（**画面1**）。

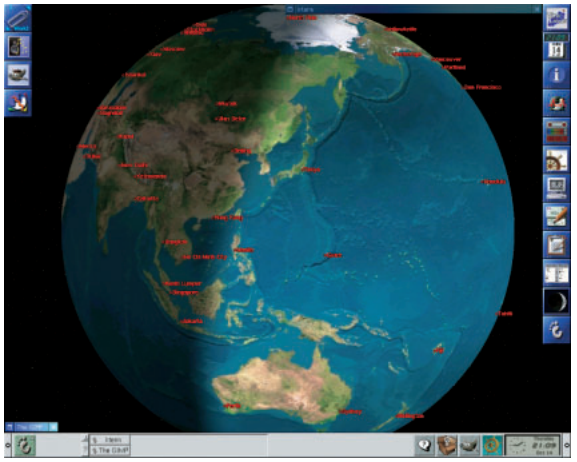
--orオプションは正射投影図法による（つまり円形の）表示を意味しており、これを省略するとメルカトル図法による平面表示となる（**画面2**）。このほか、表示の中心となる緯度・経度、地名の表示など、すべてコマンドラインオプションで指定する。

一定時間ごとに表示を更新するには、xplanetbgコマンドを利用する。オプションはxplanetと同じだ。初期設定では5分ごとにxplanetを呼び出す（-waitオプションで変更可能）。

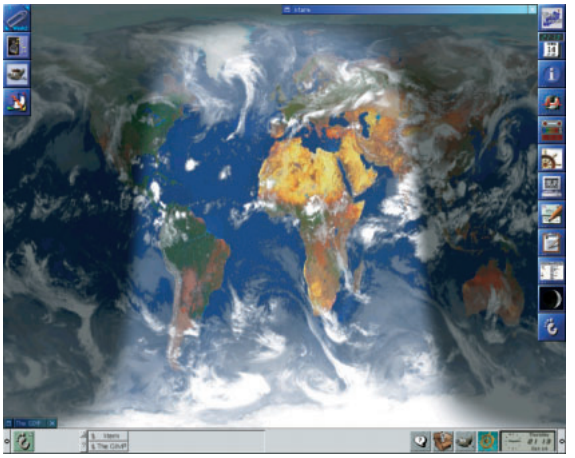
また、xplanetを--wや--aオプション付きで起動すると、ルートウィンドウではなく通常のウィンドウに画像が表示される。特に--aオプションの場合は、画像がリアルタイムに回転し、いくつかのキーを使って回転速度や方向などを変更できる（**画面3**）。



画面3
火星を通常のウィンドウに表示。画像を回転させることも可能だ。



画面1
地球の画像を正射投影図法で表示。地名も表示してみた。



画面2
メルカトル図法による表示の例。雲の画像も重ねられる。

OpenGLを利用したリアルな3D時計

glclock

バージョン：5.0　　種別：フリー

http://www.daionet.gr.jp/~masa/glclock/index.html

glclockは、OpenGL（あるいは互換ライブラリのMesa）を利用して、リアルな3D時計を表示するソフトだ。美しい時計を手元に置いて、さまざまな角度から眺める楽しみを味わえる。オプションの指定により、ソフトウェアによる描画でも快適に表示されるシンプルなものから、3Dアクセラレータが必須のゴージャスなものまでバリエーションに富んだ時計の外観を楽しめる。

●ビルドとインストール

glclockはファイル一式をtar＋gzipしたtarボールで配布されている。一般的なLinuxのディストリビューションでは、ビルドする前にmakefileの10、11行目の「/usr/local/X11R6/…」という部分を、「/usr/X11R6/…」と修正する必要があるだろう。

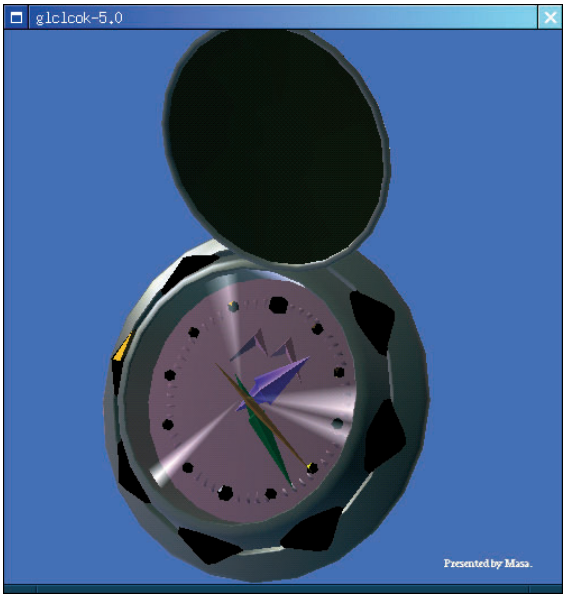
ビルド手順は簡単で、ファイルを展開後、「make」とするだけでいい。インストールは手動で行う。実行ファイル（glclock）とシェルスクリプト（＊clock）を、「cp ＊clock /usr/local/bin」などとして、/usr/local/binなどにコピーする。また、テクスチャ用の画像ファイル（＊.ppm）は、適当なディレクトリ（/usr/local/share/glclockなど）にコピーし、環境変数GLCLOCK_IMAGE_PATHにそのディレクトリを設定しておく。

●時計の向きはマウスで変更可能

「glclock&」として起動すると、フタ付きの円形の時計が表示される（**画面1**）。起動時オプションで時計の形や質感を変えられるが、いちいち指定するのが面倒なら同梱のシェルスクリプトを利用しよう。「crystalclock」（**画面2**）や「f_mwclock」（**画面3**）などのバリエーションを手軽に楽しめる。なお、「f_」で始まるシェルスクリプトは、外形を六角形にして処理を軽くしたもの。CPUの遅いマシンではこれらを使うとよいだろう。

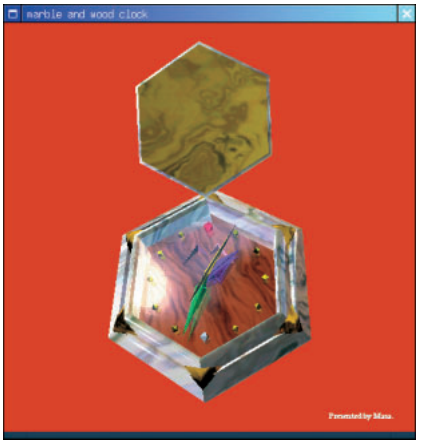
ボタンを押したままマウスを動かすことで、時計の移動・回転や、フタの開閉が可能だ。また、fキーでフレームレート（秒間あたりの表示フレーム数）の表示のオン／オフ、cキーでスクリーンショットの作成、m／n／aキーで各種特殊効果のオン／オフといったキー操作も可能だ。さまざまな設定で時計を表示し、フレームレートを計測するベンチマーク用スクリプト（benchclock）も用意されている。

このほか、スクリーンセーバ的な使い方も可能だ。-SAオプションを指定して起動すると、画面全体にウィンドウが表示され、その中を時計が回転しつつ移動する。時計のサイズや回転速度、視野角の設定は、このモード専用のオプション（-sizeなど）によって変更可能だ。



画面1
基本はこの時計。マウスで回転・移動やフタの開閉が可能だ。

画面2
半透明のcrystalclock。3Dアクセラレータなしではつらい。



画面3
大理石&木目調のf_mwclock。外形が六角形なので処理は軽い。

2つのファイルの違いをカラー表示

gtkdiff

バージョン：1.0.0　　**種別：GPL**

http://www.ainet.or.jp/~inoue/software/gtkdiff/index.html

gtkdiffは、2つのファイルの内容を並べて表示し、両者の相違点をカラーで表示するソフトだ。プログラムのソースや各種設定ファイルなど、内容の一部を変更することの多いファイルに対して使うと、どの部分が変更されたのか素早く把握できる。2つのディレクトリを比較して、追加、変更されたファイルを一覧表示することも可能だ。なお、動作にはGTK+1.2以降とGNOMEが必要。

●ビルドと起動方法

gtkdiffはファイル一式をtar＋gzipしたtarボールとRPM形式の両方で配布されている。RedHat系Linuxを使っている人はRPM形式のバイナリパッケージを使うといい。ソースからのビルド手順も、「./configure」→「make」→「make install」という一般的なものだ。

ktermなどのコマンドラインから、

```
$ gtkdiff ファイル1 ファイル2 &
```

と比較する2つのファイルを指定して起動すると、gtkdiffのウィンドウが開いて、両者の内容が左右の領域に表示される（**画面1**）。あるいは、ファイル名を指定せずに起動し、[ファイル]－[開く]で2つのファイルを順番に選択してもいい。

●相違点を色付き表示

内容の異なる行の背景色はオレンジ色と空色で表示されるので、一目で違いを判別できる。また、右側には、各ファイル内での相違点とその対応が水色・赤色のバーとそれを結ぶ線で示されている。

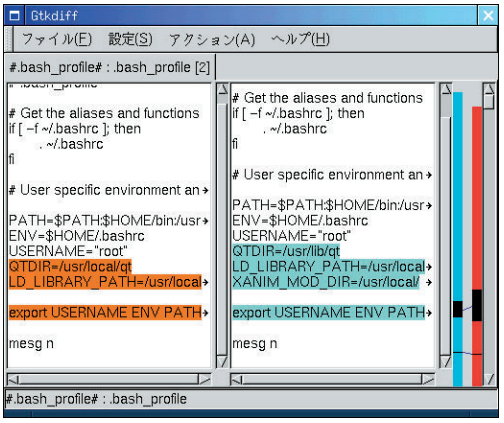
画面が狭い環境では、[設定]－[画面表示]で[1画面表示]に切り替えるといいだろう。こちらは、2つのファイルの内容をひとつにまとめ、相違点だけ背景色を変えて上下に並べて表示してくれる（**画面2**）。

[アクション]以下の項目を選択するか、Ctrl-N／Pキーを押すと、前後の相違点に即座に移動できる。なお、[設定]－[行の折り返し]をチェックした状態では、これらの機能は動作しないため、通常は行を折り返さない設定で使うことをお勧めする。

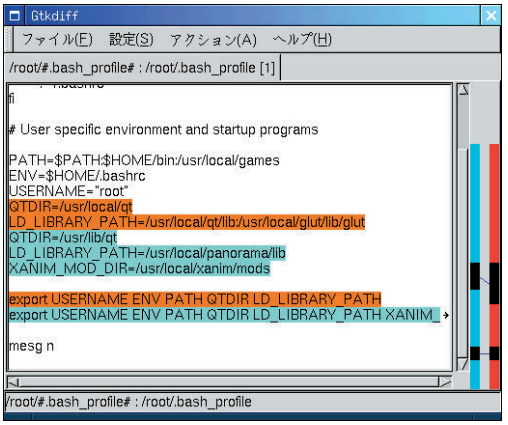
●ディレクトリの比較も可能

2つのディレクトリの内容を比較するには、コマンドラインでそれらを指定するか、[ファイル]-[開く]で[Compare directories]をチェックしてからディレクトリを選択する。

すると、片方にしか含まれないファイルや、同名だが内容の異なるファイルのリストが表示され、よく似た2つのディレクトリの内容の違いを簡単に把握できる（**画面3**）。

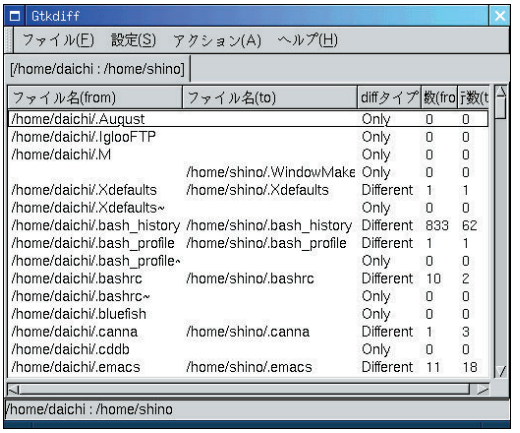


画面1
色分け表示されているため、2つのファイルの差は一目瞭然だ。



画面2
1画面表示に設定すれば、狭い画面でも使いやすい。

画面3
2つのディレクトリに含まれているファイルの比較も容易に行える。



DOS形式のフロッピーを簡単に扱える

mfm

バージョン：1.0　　種別：GPL

http://www.core-coutainville.org/mfm/

mfmは、MS-DOSやWindowsのFAT形式のフロッピーを、Linux上で簡単に扱えるファイル管理ソフトだ。いちいちフロッピーをマウントすることなく、フロッピー上のファイル一覧を参照し、ファイルのコピーや削除を行える。ただし、日本語のファイル名には対応していない。なお、内部でmtools（mdir/mcopy/mdelなど）を呼び出しているので、実行にはmtoolsが必要だ（ほとんどのディストリビューションに最初から含まれている）。

●ビルドとインストール

mfmはファイル一式をtar＋gzipしたtarボールとRPM形式の両方で配布されている。面倒を避けたい人はtarボールからのインストールがお勧め。手順は簡単で、ファイルを展開後、「make」→「make install」とするだけでいい。実行ファイルは/usr/local/binに格納される。

RPMはちょっと面倒だ。まず、バイナリのRPMがSuSE 6.1用なので、Vine Linuxなどではそのままインストールしても実行できない。そこで、ソースのRPMを利用して、バイナリのRPMをリビルドする必要がある。その際、実行ファイルのインストール先が、/usr/binや/usr/X11R6/binではなく、/usr/X11/binになっていることに注意。あらかじめ、

```
# ln -s /usr/X11R6 /usr/X11
```

としてリンクを張っておけば、問題なくリビルドとインストールを行えるはずだ。

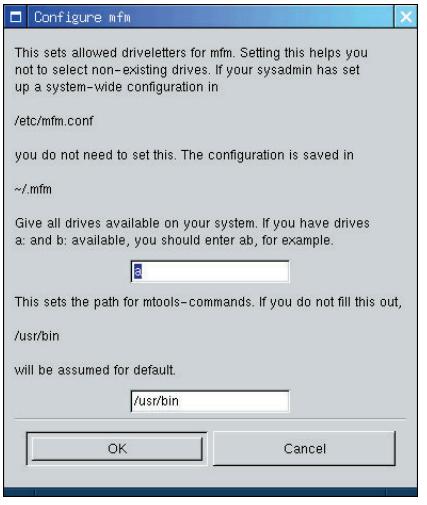
●フロッピーの内容を一覧表示

「mfm&」として起動すると、左右にファイルリスト表示ペインを持つウィンドウが開く（**画面1**）。最初はどちらもカレントディレクトリの内容が表示されている。ディレクトリを切り替えるには、リスト中のディレクトリをダブルクリックすればいい。

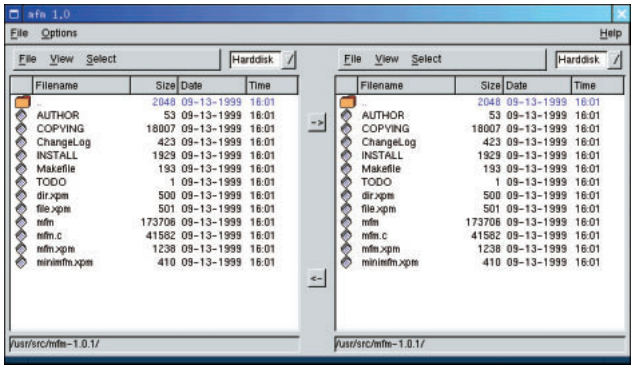
MS-DOSやWindowsのフロッピーをFDDに挿入した後で、各ペインの領域の左上にあるリストボックスを「Hardisk」から「a:」に切り替えると、フロッピーの内容が表示される。VFATに対応しているので、長いファイル名もそのまま表示可能だ（**画面2**）。

コピーや削除の際は、対象となるファイルをクリックで選択する。Shift-クリックによる範囲指定や、Ctrl-クリックによる複数選択に対応しているほか、［Select］－［All］で全ファイルを選択できる。選択後、中央部の［←］／［→］ボタンを押すと、それらのファイルがまとめてコピーされる。また、［File］－［Delete］により、選択したファイルを削除も可能だ。

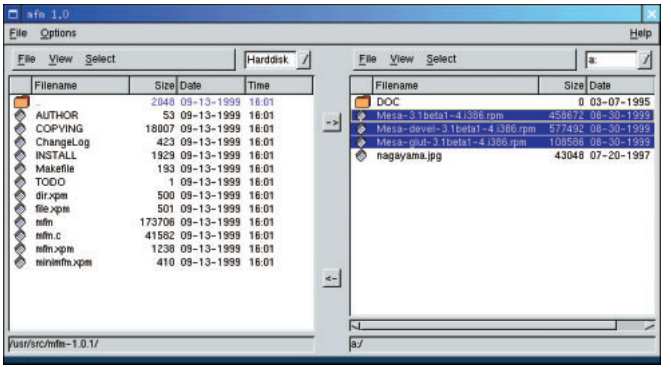
なお、FDDが複数あったり、mtoolsが/usr/bin以外に置かれている場合は、［Options］－［Configure mfm］で設定ダイアログを開いて、設定を変更する必要がある（**画面3**）。



画面3
複数のFDDのドライブ名やmtoolsのディレクトリはここで設定。



画面1
2つのファイル表示領域を持ち、ファイルを簡単にコピーできる。



画面2
VFATに対応しているので、長いファイル名も正しく表示される。

Linuxマシンを600台導入した京都産業大学

マシンの空き時間を利用して「仮想並列コンピュータ」を実現

京都産業大学（京都市北区）が21世紀型キャンパスの中核として新設した情報教育施設「10号館」が、1999年4月より稼働を始めた。ギガビットEthernetを使用したキャンパスネットワークシステムの導入をはじめ、CALL（Computer Assisted Language Learning）の採用や国際化教育用衛星受信機器の設置など、さまざまな先端教育環境を備えたこの施設の最大の特徴は「Linux/NT教室」の設置にある。
情報教育の中核システムにLinuxを採用するというのは国内でも初の試み。そこで、その実状を取材するため、同学の計算機センターを訪ねた。

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

京都の繁華街、河原町を鴨川沿いに北へ。鞍馬へと通じる洛北の落ち着いた環境の中に京都産業大学のキャンパスは広がっている。経済、経営、法、外国語、理、工の6学部に約1万3000名の学生を擁するこの総合大学において、新たな情報教育施設「10号館」が稼働を始めたのは今春のことである。

「情報化ゼネラリスト育成」を目標に掲げるこの施設には、全学部のカリキュラムにおいて共同利用される603台のPCが設置されている。そして、そのすべてのマシンにLinuxとWindows NTのデュアルブート環境を導入しているのが、この「10号館」の大きな特徴だ。しかも、この600台超のLinuxマシンが仮想並列コンピュータとして稼働するという。もちろん、これだけの規模の仮想並列環境は世界にも例を見ない。

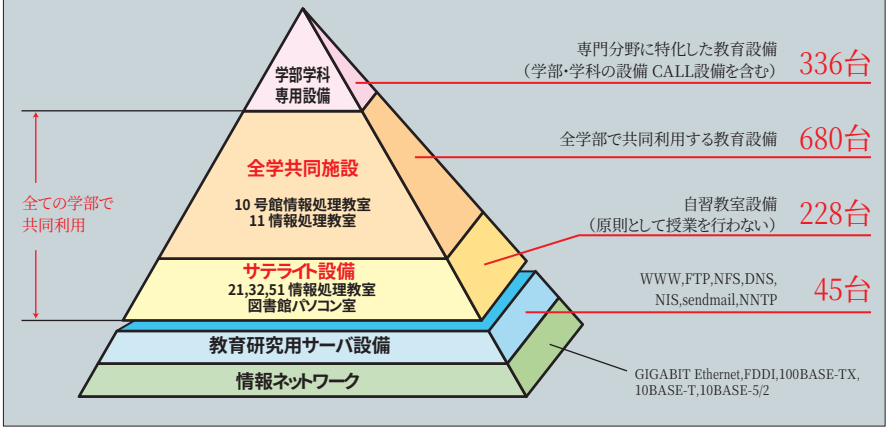
この先端情報教育施設の全容を知るべくキャンパスを訪ね、その計画段階からプロジェクトに参画し、現在もその運用に携わっておられる、同大学計算機センター教育研究システム課長、坪内伸夫氏にお話をうかがった。

全学部共同利用を目的とした情報教育システムの構築

京都産業大学において、情報教育の中核となる情報教育棟の建設計画が持ち上がったのは、今から4年前のこと。「その計画当初より、全学部で共同利用が可能な情報教育システムの構築を目指した」と坪内氏は当時を振り返りながら語った。

大学内に情報教室が設置される場合、その運用は各学部の管理下に置かれるのが通例であり、その際、システムはそれぞれのカリキュラムに対応するため、ハードウェア、ソフトウェア双方において学部に特化した構成となってしまう。新校舎の設計図からは、スペース的に500台程度のコンピュータが導入できると想定されたが、それを6つの学部に分割してしまえば、それぞれの学部から見ればわずかに1～2教室が増えるに過ぎず、講義の受講生人数によっては全員の座席を確保することができない状況も起こりうる。仮に、隣の別学部の大きな教室が同時間帯に空いていたとしても、そこに導入され

■京都産業大学情報教育設備の構成



たハードやソフトが学部ごとに特化したものであったならば、せっかく空いている設備を利用することは困難となるわけだ。

そのため、学内の導入検討委員会において、「新設する情報教室を学部に分割せず、すべての学部で共用できるスペースとして、学部色に特化した設備を導入することなく、全マシンを無個性なクライアントとして設備を有効に活用する」という意見が採択されることとなった。

しかし、ここで問題となったのが、理工系学部のために導入が必須とされたUNIXマシンの存在である。当時、学内ではすでに商用UNIXワークステーションが利用されていたのだが、これだけ大規模なシステムに同じものを導入するとなると、膨大なコストが発生してしまう。かといって、理工系学部専用のシステムを設けてしまっては、「全学部共同利用」という目的と矛盾する。

PC-UNIXを採用し、Windows NTとのデュアルブート環境を構築するというプランは、こうした状況を解決する選択肢として必然的に生まれてきたともいえる。

TurboLinuxの採用

しかし、「PC-UNIXの採用」というプランは、最初からすんなり受け入れられたわけではない。はじめは多くの関係者からの反対意見もあったという。なにせ、600台を超す大規模な教育システムにフリーOSを採用しようというのだから、これも当然といえよう。OSとPC/AT互換機に対する信頼性の問題、デュアルブート環境の安定性への不信感など、反対の理由はさまざまであり、関係者の大半は商用UNIXの採用を支持していた。

そこで、坪内氏をはじめとするPC-UNIX支持派は、OS評価用に準備していた3台のマシンを試用機として関係者に開放した。LinuxとFreeBSDを導入したこの自作機の試用後の評価はすこぶる好評で、最新PCの性能に対する驚きの声も多かったという。ベンチマークテストでは、学内の研究室で稼動している商用UNIXマシン（導入1998年7月、価格450万円）を上回る数値をこの評価機がたたき出し、その風向きは一気に逆転した。学内でPC-UNIXを使用しているヘビーユーザー教員からの説得などもあり、最終的に全マシンへのPC-UNIXとWindows NTのデュアルブート採用が決定したのである。

次の課題はOSの選定。Solaris for x86、Linux、FreeBSDなど、数あるPC-UNIXの中から、どれを採用するのか？　動作の検証やさまざまな情報収集が行われ、その結果、学内で利用されているUNIX版のMathematicaが動作するのがLinuxであることが判明。これが決め手となり、ここにLinuxの採用が決定した。もちろん、昨今のLinuxが持つ勢いも大きな判断基準になったのはいうまでもない。

Linuxのディストリビューションのうち何を選択するかという問題については、日本語の商用ソフトウェアが多く移植されるだろうという判断からTurboLinuxが選ばれた。Red Hat Linuxの採用もかなり検討されたそうだが、Red Hat Linuxの強みであるRPM（RedHat Package Manager）がTurboLinuxでも稼働することから、最終的には惜しくも選に洩れる結果となったという。

最後に残されたのがハードの選定である。もちろん、このハードウェアにはLinuxの動作を完全保証するパーツのみで構成されたシステムが必須となる。そのため、計算機センターではLinuxの動作が確認されたパーツのサポートリストを作成。それに沿ったハ

写真1
本年4月に「21世紀のインテリジェント校舎」をテーマに誕生した京都産業大学10号館。学内のネットワーク拠点にふさわしい機能を備えている。

写真2
受講生用クライアントマシンIBM Netfinity 3000。Wake On LAN 機能による遠隔起動は仮想並列コンピュータとしての利用に威力を発揮する。

ードウェア仕様書を作成し、なおかつ、コンピュータの仮想並列利用に必要な遠隔起動機能を備えたマシンの提案をメーカーサイドに委託した。最終的には、Linux/NT双方で電源管理が可能なIBM社のNetfinity3000が採用されたのである。

Linux/NT教室に見るさまざまなアイデア

PCとLinuxの採用により、機材導入に伴うコストを大幅に削減しながら、10号館のLinux/NT教室は晴れて今春より稼働を始めた。しかし、600台を超えるコンピュータの保守管理を行うとなると、そこに発生する運用経費も無視できないものとなってしまう。マシンの保守管理にかかる経費のほとんどは人件費である。そこで、この教室のシステムには、極力少ない人員で保守管理が行えるような工夫も盛り込まれている。

OS、特にNTのメンテナンスにおいては、多くの場合コンソールによる直接操作が必要となってしまう。そこで、この施設ではIBM社の提供するLCCM（LAN Client Control Manager）を利用することでクライアントマシンの遠隔操作を可能にした。これにより、LinuxとNTのインストールイメージを600台にほぼ完全に遠隔自動配布することができ、保守担当者はクライアントマシンの前へ赴くことなく、別室からコマンドひとつでメンテナンスが行えるわけだ。

現在、京都産業大学では、Linux/NT教室に設置されている600台超のクライアントマシンの運用管理をわずか1名で行っているというからすごい。深夜、無人の教室でコンピュータが突如起動し、無言のまませっせとその中身を書き換えていく。これが夏の夜なら、ちょっとぞっとするような光景ではあるが……。

さらに、教室風景をご覧いただくとおわかりの通り、ここに設置されているモニタはすべて液晶ディスプレイである。機材の選定を行ったのは1998年であり、液晶ディスプレイはブラウン管式ディスプレイに比べかなり高価ではあったが、新設される教室に配置する什器類との関係も考慮し、思い切って採用に踏み切ったという。モニタなどの機材は約3年で入れ替えとなるが、机などの設備はさらに長い期間使用する。そのため、省スペースに有効な液晶ディスプレイを採用し、机の奥行きを10cm短く設計することで、通路幅の広い快適な利用環境を実現しているのだ。

液晶ディスプレイの採用は、このほかにも大きな効果をもたらしている。液晶は、その消費電力の少なさに加え、ブラウン管式ディスプレイに比べ発熱量が非常に少ない。そのため、エアコン負荷などが大幅に軽減される。結果として、液晶ディスプレイの採用はランニングコスト削減に大きく貢献した。

そして、こうした数々の工夫の中でも特筆すべきなのが、ネットワークに接続された各教室内のLinuxマシンを有機的に結合し、マシンの空き時間や夜間の時間帯を利用してスーパーコンピュータ並のパフォーマンスを引き出す「仮想並列コンピュータ」としての利用である。

Linux環境による仮想並列コンピュータ

仮想並列コンピュータは、並列アプリケーションプログラムの実行環境であるPVM（Parallel Virtual Machine）などにより実現される。今春、京都産業大学に設置されたLinuxマシンは603台。これだけの数のマシンをネットワークを通じて有機的に結合するというのは果たして可能なのか？　そしてそのパフォーマンスの程は？

この興味ある実験について、京都産業大学工学部長、黒住祥祐教授に詳細をうかがうことができた。

黒住教授のご専門は、コンピュータグラフィックス論や画像シミュレーシ

写真3
Linux/NTのデュアルブート環境。左にLinux、右にNTの起動画面が表示されている。センターにあるモニタは講師の画面を表示するためのもの。

写真4
採用されているモニタは全て液晶ディスプレイ。そのため机の奥行きも狭く、省スペースにも貢献。実にうらやましい学習環境だ。

写真5
Linux/NT教室に設置された全てのマシンの稼働状況を一目で確認できる。別の建物からの遠隔操作も可能だ。

ョン論など。マルチメディア系のヘビーな画像処理がその研究対象とあって、この仮想並列コンピュータの実験を行う京都産業大学PVMグループでも中心メンバーとして活躍されている。

黒住教授のお話によると、仮想並列で結ばれたマシンは、バーチャルリアリティの構築に必要とされる3次元物体のレンダリング処理などにおいて大きな効果を発揮するという。画像処理の場合、そのパフォーマンスはマシンの接続台数に比例して向上していくというから驚きだ。

Linux/NT教室に設置されているクライアントマシンNetfinity3000のハードウェア環境は、CPUがPentium II 400MHz、実装メモリ128Mバイト。仮にそれぞれのマシンがOSに使用する部分を除いたメモリの空き領域が90Mバイトと想定すると、600台のマシンで利用できるメモリ量はなんと54Gバイトにも及ぶ。この環境に400MHzの石が600個装着されているわけだから、スーパーコンピュータ並の性能を実現するというのもうなずける。それぞれのLinuxマシンは、お互いに連携を取りながら仕事するため、その接続には高速なネットワーク環境が必須となる。クライアントマシンNetfinity3000は、100Base-TXのEthernetカードを内蔵しており、100Mビット／秒の転送速度でデータを転送できるようになっている。また、キャンパスネットワークシステムにはギガビットEthernetを導入している。

ここでもう一度思い出していただきたいのが、このパラレル・バーチャル・マシンを構成する個々のパーツが、Linux/NT教室に設置されたPCであり、学生が利用していない空きマシンを使ってこれらの実験がなされているという事実だ。大規模な投資をすることなく、教室内に配置した既存のマシンを有効に活用することで、スーパーコンピュータに匹敵する高度な演算処理を実現している。なんとも実用的で、有意義なコンピュータ利用であることか。現状、200台までの仮想並列処理の安定動作が確認されており、今後さらに台数を増やした安定処理の実現を目指し、研究が進められているという。

黒住教授のお話によると、このような仮想並列コンピュータの構築は特に複雑なハードウェアを必要とするわけではなく、前述のPVMを導入することで比較的容易に実現できるという。PVMに関する情報はWeb上にも公開されているので、興味のある方はそちらを覗いてみてはいかがだろう。（PVM公式サイトhttp://www.epm.ornl.gov/pvm/）

一般家庭でも利用されているクラスのPCとフリーOSであるLinuxの組み合わせによるパラレル・バーチャル・マシン。今後、対応ソフトウェアが充実してくれば、こうしたコンピュータ利用法がさらに身近なものになってくるのかもしれない。

とはいえ、このような芸当は、Linuxをはじめとするunix系OSだからこそなせる技。現状のWake On Lan環境下では、異なる仮想並列処理を実行するには少々骨が折れるそうだ。黒住教授は、「今後はさまざまなクラスタOS環境における並列化処理についても研究してゆきたい」と語ってくださった。

京都産業大学が最新の設備を備え、「21世紀のインテリジェント校舎」を自称する10号館。その最大の特徴は、利用者の側からの発想を具現化していく創意工夫の姿勢と、関係者の方々の熱意にあった。そして、この京都産業大学の挑戦には、Open Sourceを育んできた多くのコミュニティも注目している。

これを機に、Linuxを取り巻く環境がさらに前進し、こうした有意義な試みが全国の教育機関に波及してゆくことに期待したい。

写真6
Linux/NT教室に設置されたクライアントマシンは、実験室からでも仮想並列コンピュータとしてコントロールできる（写真は黒住教授）。

写真7
広い教室内にクライアントマシンが整然と並べられている。マシンは机の下に置かれている。

京都産業大学
〒603-8555 京都市北区上賀茂本山
http://www.kyoto-su.ac.jp/

隠喩としてのコンピュータ

――GPLと自由、そして啓蒙

文：豊福 剛
Text : Tsuyoshi Toyofuku

リチャード・ストールマンが主張する「自由」というのは、実はとても単純なことなのかもしれない。しかし、単純なことが必ずしも理解しやすいとは限らない。これは、GPL（GNU General Public License）の条項を読むときにも感じることだ。

GPLにおいて、フリーソフトウェアは、「使用、コピー、変更、配布が自由にできるソフトウェア」として定義されている。このことはよく理解できる。また、GPLが適用されたソフトウェアの修正版にもGPLが適用される「Copyleft」という方法論の効果も理解できる。それでも、GPLを読んだだけでは、GPLがなぜ考案されたのかという根本的な疑問に対する答えを得ることは難しい。

本当の意味でこの答えを得るためには、やはりストールマンがどのような人間で、どのような経験を通して、 GPLの背景にある思想を形成していったのかという過程を、歴史的な流れをふまえたうえで検証していく必要があるように思う。

コミュニティ崩壊の経験と自由を守る覚悟

ストールマンはいったいどんな人間なのだろう、という関心から、スティーブン・レビーの『ハッカーズ』（“ HACKERS－ Heroes of the Computer Revolution ”、Steven Levy著、工学社刊）を、これまで何度となく読み返してきた。この本のエピローグには、ストールマンがGNUプロジェクトを始めるに至った歴史的な経緯が書かれている。ここに書かれているストールマンの話を読んで、涙を流したといった知人がいた。ぼくもこれを最初に読んだときは、やはり涙した。何年かおきに読み返してきたのだが、そのたびに心の深いところを揺さぶる力がある、不思議な本だ。

ストールマンが自由に固執するのは、おそらく彼が決定的な世界喪失を経験してしまったからだ。彼にとっての「喪失された世界」は、具体的にはMIT（Massachusetts Institute of Technology；マサチューセッツ工科大学）のAIラボ（人工知能研究所）のことを指している。

当時の、MITのAIラボではハッカー倫理が当たり前の文化としてその場に溶けこんでいた。しかし、そこに「官僚主義、セキュリティ、仲間と共有し合うことを拒否するような障害物」が迫ってきたのだ。また、AIラボのハッカーが作ったベンチャー企業、LMI社とシンボリックス社の対立によって、AIラボのハッカーたちは分裂してしまった。ストールマンが「自分の妻が死んでしまった」と語るAIラボ・コミュニ

ティの崩壊によって、「ひとつの文化が、跡かたもなく、消し去られてしまった」のだ。

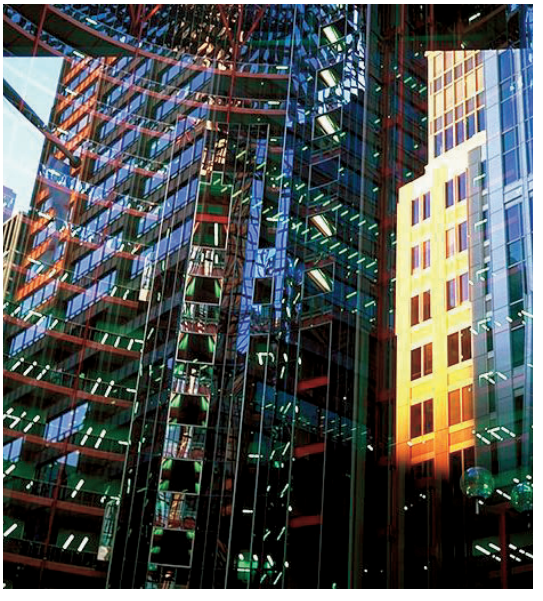
ただし、AIラボ・コミュニティが崩壊する渦中にあって、ストールマンはただ嘆き悲しんでいただけではない。パスワード・システムを強制する官僚主義に対しては、空白パスワード運動で抵抗し、シンボリックス社のLISPに対してリバース・エンジニアリングで報復する。それらの行動が怨恨と憎悪によるものだとしても、ストールマンの徹底した孤独な抵抗に、シンボリックス社のビル・ゴスパーさえも心情的には賛嘆してしまう。

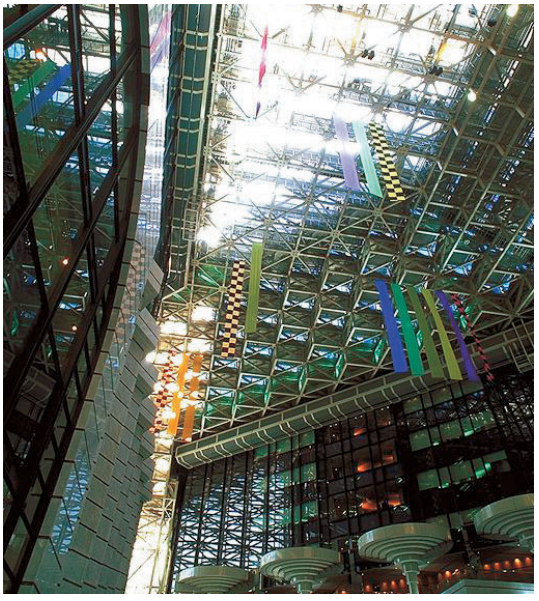
シンボリックス社への報復を'83年で区切りをつけたストールマンは、いったんMITを去る。ストールマンは、その胸中を「ここがぼくの居場所だっていえるところは、もう世界のどこにもなくなってしまった。ある意味じゃ、死んだほうがましだって気がするよ」と語っている。そして、その絶望の縁で彼が選んだもうひとつの道がGNUプロジェクトだった。その目的は、フリーソフトウェアによるOSを作ることによって、失われたハッカーズ・コミュニティを再生することだ。

自由を手放したとき、AIラボ・コミュニティは崩壊した。コミュニティを守るためには、自由を守らなければならない。だから、GNUプロジェクトが開始されてから15年が経過する中で、ストールマンは一貫して自由にこだわり続けてきたのだ。「自由がこれからも手元にあると思ってはならない。当然の権利だと思って軽視してはならない。 自由を手放したくなければ、あなたはそれを守る覚悟を固めなければならない」（オライリー・ジャパン刊『オープンソースソフトウェア』の5章「GNUシステムとフリーソフト運動」）という言葉を理解するには、コミュニティ崩壊の記憶が背景にあることを忘れてはならないだろう。

GNUの本質は社会運動にある

自由が「足かせをはめられていない」状態だとしても、それだけでは曖昧な概念でしかない。自由が語られるとき、そこでは競争や効率主義が語られることもある。自由を理解するには、それが「いかなる足かせ」からの自由であるかを見極めることが重要なのだ。ストールマンにとっての自由は、「自由のための自由」ではなく、「コミュニティのための自由」である。ストールマンが主張する自由にわかりにくさがあるとすれば、つきつめれば、自由を前提とするコミュニティがわかりにくいからだろう。





それでは、足かせのないコミュニティとは、どのようなものなのだろうか。ストールマンにとって、AIラボは「建設的アナーキズム」を具体化したものだった。このアナーキズムとは「徹底した利己主義に基づく競争社会を標榜するという意味ではない。アメリカ社会は、すでに利己的な競争社会だ。その規則（注：競争という規則）が社会を利己的なものにし続けているんだ。ぼくたちハッカーは、建設的な協力への関心が規則に取って代わるような社会にしたいと思う」（『ハッカーズ』）という考え方にほかならない。アナーキズム（無政府主義）というと、一般に暴力的なイメージでとらえられがちだが、暴力のための暴力ではなく、むしろ自主的な協力によってコミュニティを成立させる原理なのだ。そうしたアナーキズムが、理想としてではなく、現実に実践できることの実例がAIラボだった。

建設的な協力への関心、あるいは隣人に手を貸すこと、他人を助けることで成り立つ社会。このような社会は、AIラボという例外的な空間でのみ実現できたのであり、それが、「ハッカーのハッカーによるハッカーのためのコミュニティ」であったとしても、コミュニティの外部にいる者にとっては、あくまで別世界にすぎないのだろうか？　コミュニティが、あくまでハッカーに限定された専門家集団なのであれば、そこで主張される自由も専門家集団のための自由にすぎないだろう。ストールマンの思想の核心部分は、そのようなコミュニティが、ハッカーだけに限定されたものではなく、ソフトウェアのユーザーにも開かれていなければならない、と考えている点にあるように思える。

ストールマンは、「ソフトウェアを使って何ができるか」ではなく、ソフトウェアのユーザーがどんな社会に暮らすかが、一番重要なことだと考えている。ソフトウェアメーカーにはソフトウェアを排他的に所有する当然の権利があると、ユーザーが認めることは、ユーザーがソフトウェアメーカーに自分たちを支配する権限を与えることにつながる。そして、ユーザーはソフトウェアメーカーの主張を黙っておとなしく受け入れるしかなくなる。そのような社会に抵抗するためにGPLは考案されたのだ。そして、GPLにはハッカーだけでなく、ユーザーにとっても重要な概念が提起されている。

GPLにおける無保証の論理

GPLが提起する無保証の概念は、ソースコードの公開と密接に関連している。このことについて、土屋 俊は「情報は誰

のものか―知識への権利」（『現代哲学の冒険（10）交換と所有』、岩波書店刊）で、専門的知識とその社会的責任という文脈で、次のような興味深い考察を展開している。

――「保証するということは、製品の本質的仕組みについて、いわば専門家のみが責任をもち、カバーしていくという立場である。ところが、ソフトウェアは、判断を支援するだけではなく、それ自身が判断をくだす。もちろん、そのとき、機械に対してもまた、ソフトウェアの作製者に対しても責任を負わせることができない。さらに、ソフトウェアには完全だということが不可能である。その働きは、ソースコードによって規定されている内的性質だけでなく、機械と状況に依存する。ソースコードを提供してあるということは、少なくとも責任の範囲について事前に自分の側を限定していることになる。しかし、まさにそのことによって、機械と状況という偶然に依存する原因による使用者の側での被害についての責任を免れることになるであろう。」

――「このことは、専門家の知識という問題に関して、さらに重要な洞察を導く。この議論を一般化するならば、専門家は、その知識を広く人々に知られるようにすることによって責任を免れることが可能となる。つまり、自分達だけが知っているという状況は、処世訓的には危険な状態なのである。」

ソフトウェアは、プログラマーという専門家の専門的知識によって作られる。しかし、完全にバグのないソフトウェアが現実的に不可能であるだけでなく、ソフトウェアの効果が機械と状況に依存するものである以上、プログラマーはソフトウェアに対する全面的な責任を持ちえない。それにも関わらず、ソースコードが公開されない製品ソフトウェアにおいては、ユーザーは何らかの被害を被ったとしても、製造者の主張に服従するしかない。ユーザーは未成年状態にあり、製造者という他人の指導がなければ、自分自身の悟性を使用しえないからだ。

カントによれば、啓蒙とは「あえて賢くあれ。自分自身の悟性を使用する勇気をもて」であった。そして自由とは、悟性を使用する自由のことである。ただし、ソフトウェアのユーザーが未成年状態から抜け出て、その悟性を使用するためには、リテラシーが必要とされるだろう。専門家がその知識を公開することによって責任を免れるという論理は、同時に、ユーザーの側に対しても、公開された知識を理解できるだけのリテラシーを要請しているからだ。

Profile

とよふく つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

オープンソース
・
アクティビズム

オープンソースはアナーキズム？

文：安田幸弘
Text : Yukihiro Yasuda

本当は暢気なアナーキスト

アナーキズムというと、なにやら過激で暴力的なイメージがある。だけど、もともとアナーキズムってのは「オレは国家権力なんかに縛られるのはイヤだね」という、どっちかというとボヘミアンっぽい、暢気な考え方でもある。

もっとも、アナーキズムという言葉はそれなりに今でも使われているけれど、どっちかというとセンスの悪いお兄ちゃんがアホなことやって「あー、オレってアナーキー」と自己陶酔してるようなケースが多い。勘弁してほしいもんだ。

それはともかく、確かに政治的な「イズム」の話になると、「国家が決めた法律なんて」だとか「税金なんて払わない」ということになって、法律を守らせて税金を取りたい国家との間で摩擦が起きる。こうなると少々過激なにおいがしてくる。でも、これも要するに、国家だの政府だの権威だのというややこしいものがあって、暴力的に（？）権力を振り回すから摩擦が生じるわけで、もし、そんな強制がなかったとすれば、アナーキストたちは過激になることもなく、自分たちのコミュニティでのんびり平和に暮らしていけるのだ。

ちなみにアナーキズムは、無政府主義なんて訳されるけれど、アナーキストたちに言わせれば、これは誤訳だという。むしろ、分散主義とか自律主義ぐらいに訳すほうがいいんだそうだ。つまり、中央の権力やら規制やら法律やらではなく、それぞれのコミュニティが持つ自律的な秩序を調和させていくことで、政府という存在がなくても回っていく仕組みを作ろうとする人たちがアナーキストなのだ。

現実の社会を考えると、アナーキーな社会を実現するのはちょっと無理がありそうな気もする。しかし、こうしたアナーキズムの発想は、ほかでもない、最新流行のインターネットの設計思想にかなり近い。

もちろん、インターネットにもネットワーク管理組織があって、メタな秩序が管理されているし、そのコンテンツは国家による著作権法で縛られるわけだけど、とにかく一度ネットワークの割り当てさえもらってしまえば、技術的にはもうアナーキーである。勝手にプロトコルを作って勝手にサービスを提供してもかまわない。もっとも、勝手にやっても相手にされないだけで、まあ、それがアナーキズムってものなんだろう。

GNU＝マルクス、ソフト企業＝国家社会主義

インターネットだけじゃない。ぼくはオープンソースも、一種のアナーキズムなんじゃないかと思う。

有名なカテドラルとバザールの話じゃないけれど、ソフトの開発をなんとかイズムのやり方にたとえてみると、オープンソースがアナーキズムだとすれば、GNUはマルクスレーニン主義、巨大なソフトウェア

企業は、さしずめ国家社会主義ってところかもしれない。

オープンソースがアナーキーなところは、さまざまなソフトウェアの開発集団がコミュニティを作り、勝手にやっているようで、それでいてコミュニティ間で絶妙な調和が取れているというところ。さまざまなオープンソースソフトウェアを組み合わせて、無数のLinuxのディストリビューションが配布されているのも、オープンソースのアナーキーな秩序感覚の結果なんじゃないだろうか。もっとも、それが可能だったのも、カテドラルとバザールの違いはあるとしても、GNUのソフトウェア共産主義めいた共有の思想やGPLやコピーレフトの理論があってこそだったわけだ。現在のオープンソースとGNUは、アナーキズムとマルクス主義ほどの対立があるわけじゃない。だいたい、エンドユーザーにしてみれば、使っているソフトがオープンソースかGNUかなんて、ライセンスのファイルをじっくり読まなきゃわからないというのもまた事実。まあ、昔から、アナーキストと共産主義者たちは、お互いに「なんだ、あいつら」とか何とか、悪口を言ってケンカをしながらも、結構、仲良くやってきたんだしね。

21世紀のコンセプトに？

歴史的に見れば、アナーキズムは決して成功しなかったし、社会運動の主流にもなれなかった。そもそも、主流になって政権を取る、なんて発想がないんだから当然かもしれない。

でも、最近のインターネットやオープンソースを見ていると、アナーキーなシステムってのも、結構うまく動くんじゃないかという気がする。国家権力によるガチガチの統制の中で暮らしていると、規制はあるのが常識のように感じられるが、規制緩和が叫ばれて、本当に規制がなくなってみると、あれはいったい何だったんだろうという種類の規制が少なくなかったことに気づく。規制撤廃で猛威をふるい始めた多国籍企業もアナルコキャピタリズム（アナーキー資本主義）なんて呼ばれることもあるようだけど、それはまた別の話。とにかく、インターネットが世界の国境を取り払い、情報は国を超えて自由に行き来するようになり、オープンソースのような活動が現実の社会にも無視できないだけの実績を積み上げているわけだ。

今後、ソフトウェアの分野以外でも、インターネットをベースとした草の根的な活動が誕生すれば、21世紀の世界のコンセプトは、結構、アナーキズムということになるのかもしれない。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ドクターShiotaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shiota

10・5　Intel 64ビットCPUの商品名がItaniumに決定
10・6　PhilipsがCEから撤退
10・4〜5　AMD Athron 700MHz/SledgeHammerを発表
9・28　820チップセットの出荷を延期
9・24　ソニーメモリースティックウォークマン発表
9・14　HandSpring Visor発表
9・14　Palmが3comから分離へ

さて、Linux関連では、Torvaldsの発言が、秘密企業であるTransmetaに絡んで大きく取り上げられたのが、今月は大きな話題か。

トランスメタがComdexで発表か？

Linuxの開発者であるLinus Torvaldsは、自身が勤める会社TransmetaがComdexで発表を行うことをほのめかした。このTransmeta、業界のウワサによると、インテル互換チップを開発しているらしい。しかも、同社からの特許申請などによると、ハードウェア自体は、x86命令セットを直接実行するものではなく、ソフトウェアにより、命令コードを変換して実行するもので、これにより、インテルのさまざまな特許を回避できるらしい。しかし、このTransmetaは、徹底した秘密主義の会社で、いまのところ、特許申請などやむを得ないものを除いてなにも公開していない。同社のWebサーバには、「This web page is not here yet! 」と表示されるのみ（でもページが表示されるんだからなんか矛盾してない？　なんかI Can't Speak Englishみたい……）。

Solarisソースコード公開

そういえば、米国で開かれているInternet Worldでは、Torvaldsは、SunのSolarisソースコード公開を多少は評価したのだとか。

「多少は」というのは、このライセンスが例のSCSL（Sun's Community Source Licensing）による公開で、完全にオープンというわけでもないからだ。これはいわば「見せてあげるけど、金儲けするならお金ちょうだい」的なもので、ソースを見るためにはSUNとの契約が必要。なので、Torvalsはこのライセンスプログラム自体には納得していないのだとか。

そういえば、SUNはSolarisでLinuxバイナリを実行可能にしたり、無料にしたり、ソースコード公開するなど、ここのところ、なんかSolaris自体の生き残りに一生懸命って感じ。たしかにLinuxのほうが人気あるもんねぇ。

マイクロソフトの対Linux文書

このSUNといつも仲の悪いマイクロソフトのほうだけど、Windows NTとLinuxを比較した文書「linux Myths」を公開した。これは、業界で言われている「LinuxはNTよりいい」とか「NTより安定している」なんてことに反論したもの。また「Linuxはフリー」ってことにも、「フリーのOSとは、Total Cost of Ownershipが低いことは意味しない」って反論しているのだけど、日常的なサポートでNTよりLinuxのほうが大変ってことはないと思うし（悪くても同等だと思う）、何をするにも高価なソフトを購入しなければならないNTよりは、Linuxのほうがランニングコストとしてみれば、ぜんぜん安いような気がするのだが。きっと、あちこちで反論が出ると思うよ。

それで、マイクロソフトが今年賭けているWindows2000だが、一部に、年内の出荷を危ぶむ声がある。RC3の配布開始が11月になることなどから、こういうウワサが出てきたわけだけど、「完璧になるまで出荷しない」なんていって、一方では、ソフトウェアに「完璧はない」ってのがコンピュータ界の常識。たかだか数行のプログラムならいざしらず、Windows2000ほど大きいソフトが完璧になるなんてことはありえないと思うのだが（それに完璧にな

るなら、サービスパックなんか出ないはずだ)。

RC3の配布が11月だとすると、どう考えても配布後フィードバックがかかるためには2〜3週間は必要、それから最終版を作成して、CD-ROMのプレスやパッケージ作成の時間を考えると年内出荷は不可能というのが、業界の見方。　そういえば、前にWindowsのオープンソースなんて話があったけど、その後の話がまったくありませんが、あれはどうなっちゃったんでしょう？

CobaltがIPO

さて、RedHatが株式公開で大儲けって話に続いて、Linux関係ではどこが続くかという話もありましたが、どうやら、ThinサーバーマシンのCobalt Networksが米国証券取引委員会(SEC)に株式の新規公開(IPO)を申請したのだとか。このIPOについては、Linuxを扱うハードメーカーVA LinuxもIPOするのではと言われていたのだが、申請としてはCobaltのほうが先になったようである。

そのCobaltのサーバマシン(QubeとかRAQ)をゲートウェイが販売するそうである。ゲートウェイは、かつてサーバマシンを中心に開発していたASTを買収して、いちおうサーバマシンをラインアップしているが、いまだにサーバメーカーとしては認識されていない(うちの女房なんかウシの会社といっている)。もちろん、インテルアーキテクチャでLinuxサーバという方向もありえるのだろうが、いまのPCサーバって、どちらかというとフルタワー以上で、ごつい作りが主流。簡単なサーバというと、デスクトップマシンを流用することが多いが、そうなると、メーカーとしてはハードウェア価格が安すぎて、あまりいい商売ではない。そこで、Qubeみたいに小型で簡単なサーバマシンをラインアップに加えることで、デスクトップマシンの流用も防げるし、サポートも簡単になるというもの。これは、ちょっと流行りそうな気配がする。来年は、ThinサーバがPCメーカー各社から出荷されるかも。

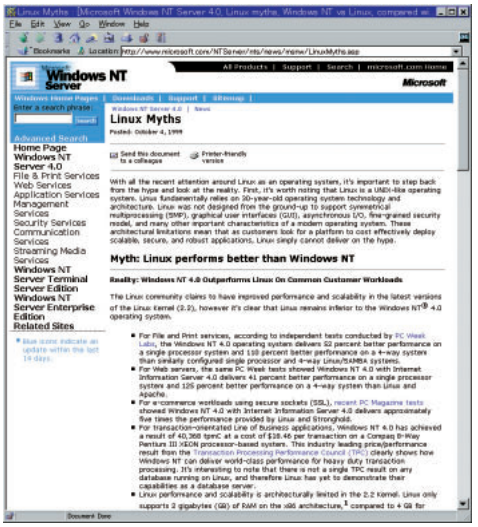
IA-64はItanium

Cobaltというと、一定年齢以上の方は、鉄腕アトムの兄ちゃんしか思い浮かばないかも。このCobalt、金属の1種だが地の妖精であるKoboldなどが語源らしい。

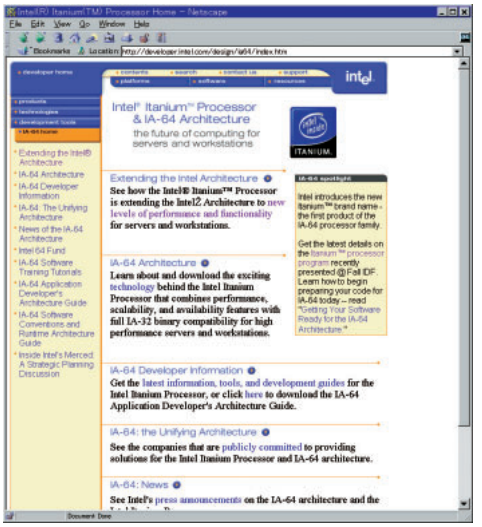
同じようになんか金属っぽい名前がItanium。インテルが次世代64ビットCPUであるMercedにつけた商品名である。アイテニアムと発音するそうだ。

インテルは、Pentium以来、それまでの番号による命名(386とか486ってやつ)をやめて合成語を作り、商標登録することにしている。というのは互換チップ対策で、数字の組み合わせは商標として登録することができないことによる。Pentiumのときもそうだったが、だいたい「-nium」と金属元素のような名前の付け方。業界では、半導体のことを「石」というので、まあ、近い表現ではある。

このItaniumだが、来年には製品が登場する。IBMやSequent、SCOは、Monterey/64(Project Montereyによる64ビット版UNIX。AIXやSCOを統合したものとなるという)をこのItaniumで動作させることに成功したらしい。インテル自身、先のIDFでLinuxを動作デモを行っているし、IBMもLinuxをIA-64に移植するTrillian Consortiumへの参加を表明している。IA-64上でもLinux優勢は代わりがないような感じ(ただし、OSの64ビット化という問題が先に待っているわけだが)。今月はこのへんで。



マイクロソフトの対Linux文書「linux Myths」
http://www.microsoft.com/NTServer/nts/news/msnw/LinuxMyths.asp



インテルItaniumホームページ
http://developer.intel.com/design/ia64/index.htm

正しきタコへの道

初級Linuxer養成講座

第3回　Linuxの中身を覗く

Linuxをインストールしたけど、何に使ってよいのかどうしても思いつかなかったら、まずは「Linuxそのものを理解する」ことに目を向けてみてもよいだろう。機械の詳しいことはわからなくても、分解してみることに楽しみを見いだせる人なら、Linuxもとりあえず分解してみると、当面の知識欲を満たすことができる。Linuxの内部を覗く方法はいろいろあるが、理屈は抜きに、まずは「生きたまま」のLinuxを覗くことから始めてみることをお勧めする。

文：竹田善太郎
Text : Zentaro Takeda

Linuxを使い始めた動機は人それぞれだろうが、「マルチタスク／マルチユーザーの言葉に惹かれて……」、という人も少なくないだろう。ちょっと古いコンピュータの世界を知っている人なら、マルチユーザーシステム（あるいは「タイムシェアリングシステム」でもよい）と聞くと、**冷房がガンガン**に利いた専用のマシンルームに、轟音をたてながら回転するディスクパックやテープデバイスに囲まれて鎮座する、**メインフレームコンピュータ**を想像するだろう。現在のパソコンは、そんな10年ほど前の「メインフレーム」をしのぐほどの機能と性能を持っていて、その上で動く「Linux」も、当時のメインフレーム上のマルチユーザー、マルチタスクOSなどに比べて、はるかに多くの機能と性能を持つOSなのである。

そして、メインフレーム全盛の時代から、多くのコンピュータ少年のみならず、技術者にとってもあこがれの的だったのがUNIXであり、その機能と性能はそのままに、われわれ一般ユーザーの手の届くところに降りてきたのが「Linux」なのだ。数十人ものユーザーが同時にログインして使うような高機能のOSを、たった1人で1台を（いや、それ以上の台数でも）独占して使えるのは、とても幸せなことであるはずなのだが、実際にLinuxを使っていてそれを実感しているユーザーはどれだけいるだろうか？

もちろん、X Window Systemをばりばりに使いこなして、画面上にアプリケーションやアクセサリのウィンドウを**これでもか**といったくらいに開きまくって、なおかつcronを使ってバックグラウンドで怪しげなジョブを動かしまくれば、マルチタスクの恩恵を受けている、ということになるのだろうが、昨日今日にLinuxを使い始めたユーザーには、そこまでやる元気も根気もないことだろう。

また、Windows（これも現在では立派なマルチタスクOSなのだが）に慣れたユーザーとしては、1つのマシンで複数のアプリケーションが動いたからといって、**何の感動もない**といった向きもほとんどだろう。

しかし、コマンドラインを中心にLinuxを使いこなせるようになるには、どうしても**プロセス**というものについての「概念」というか、「センス」のようなものを身につけなければならない。単純に、

```
$ cat hogehoge.txt | grep "foobar"
| less
```

といったパイプラインを実行する場合でも、「プロセス」がどんなものなのかについて、（専門的な詳細はともかく）イメージをつかんでいないと、とんでもない落とし穴があったりするのだ。

難しい話はともかく、**Linuxって本当にマルチタスクなのか**とか、**キーボードに触れてもいないのにディスクにアクセスすることがあって気持ち悪い**などと感じる読者がいたら、いろいろなコマンドを覚える前にちょっと寄り道をして、Linuxの中身を覗いてみることをおすすめする。Linuxの「プロセス」について、ちょっとでもイメージをつかめれば、コマンドを覚えていく上でも役に立つことは保証する。

何が動いているのか

細かい話は抜きにして、Linuxのコンソールやターミナルウィンドウの中で、次のコマンドを実行してもらいた

い。

```
$ ps axu
```

すると、**画面1**のような、難しげな英数字の並んだ表のようなものが表示されるはずだ。X Window Systemを使っている場合は、かなり長い表になるので、

```
$ ps axu | less
```

とすれば、表をスクロールして眺めることができる。

種をあかせば、psコマンドとは「process status」、すなわちプロセスの状態を表示するためのコマンドである。「ps」の後の「axu」は、Linuxのシステム上で現在動いている（あるいは待機している）すべてのプロセスを、ユーザー名付きで表示せよ、という意味のpsコマンドのオプションである。

この一覧の（先頭行の「見出し」部分を除く）1行1行が、それぞれ1つのプロセスを示している。10行表示されれば10個のプロセスが動いていることを意味するのだ。一覧のいちばん右側の列を見ると、見慣れたもの、見なれないものなど、さまざまなコマンドの名前があるのに気がつくだろう。表のどこかには、「ps axu」とか、「less」とかの項目も見つかるはずだ。これは、まさに今実行しているpsコマンドやlessコマンドのプロセスである。

この一覧を眺めれば、たった1つ（あるいは2つ）のコマンドを実行しているだけなのに、ずいぶん多くのプロセスが動いていることが理解できるだろう。左から2番目の列にある数字は、「プロセスID」、あるいは「プロセス番号」と呼ばれるもので、システムが起動してから何番目に生まれたプロセスなのかを示している。

これらのプロセスのほとんどは、「デーモン」と呼ばれる常駐プログラムや、X Window Systemを起動してる場合なら、ウィンドウシステムのさまざまな裏方作業を行っているプログラムなのである。ユーザーがコマンドラインからコマンドを実行して、それを処理するために動き出すプロセスは、実はLinuxシステムの中で見ればごく限られた数にしかならないのである。

たった1人のユーザーが使っているだけでも、実はこれだけの数のプロセスが**もぞもぞと**動いているので、これだけでも**Linuxはマルチタスクだ**ということを実感できるだろう。

Linuxを使っていて手持ち無沙汰に感じたら、よく意味は分からなくてもpsコマンドを実行してみる、というくせをつけるのは悪いことではない。ゆくゆく、Linuxを自在に使いこなせるようになったとき、psコマンドの結果を見れば、**いつもと様子が違う**ことに気がついて、マシンのトラブルの前兆をとらえたり、クラッカーの侵入（個人で使っているマシンには、そのようなことはまれだろうが）を察知できるようになるのだ。

だれが何をやってるのか

psコマンドに似ているものの、目的がちょっと違うのがW**コマンド**である。「w」1文字の覚えやすい（したがって忘れやすい）コマンド名だが、これは、「マシンに誰がログインしていて、どんなコマンドを実行しているか」を表示するコマンドだ。wコマンドを実行すると、**画面2**のような一覧が出力される。左端にユーザーのログイン名、いろいろな数値が並び、右端には現在そのユーザーが実行しているコマンドが表示される。X Window Systemを使っている場合なら、1人のユーザーしかログインしていないときでも、同じユーザー名の行が複数表示されるかもしれない。これは、同じユーザーが複数回同時にログインしていると見なされる場合があるからだ。

ちょっと脱線するが、筆者がUNIXを使い始めた十数年前は、UNIXを自由に使えるユーザーはまだそれほど多

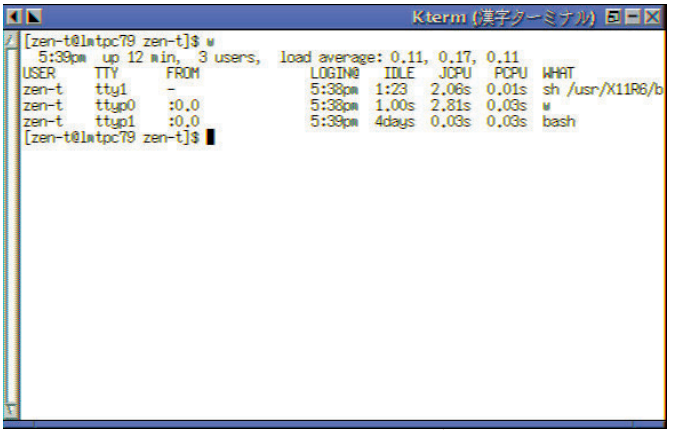
画面1　psコマンドの実行例
自分しかログインしていない寂しいはずのマシンでも、常時これだけの「プロセス」が動いている。これを見て、孤独感をいやせるようになったら、はじめて一人前の「Linuxer」と言えるのだ（？）。

くなく、初心者向けの参考書や相談相手も少ない時代だった。そんな中で、ある日「UNIXを使え」と命令されてログインIDをもらったのだが、右も左もわからない間は、ログインはしてみても次に何をすればよいか、まったくわからなかった。UNIXの世界は（今ではそんなことはないだろうが）ちょっと職人肌のユーザーが多くて、**習うより慣れよ**とか、**人の技を盗め**とか、**××の設定方法は一子相伝**などという考え方が平気でまかりとおっていたのだ。

そんなときに役に立ったのが、psコマンドとwコマンドだった。多くのユーザーがログインしているホストで、psコマンドやwコマンドを実行すると、だれがどんなコマンドをどんなオプションで使っているかが、一目瞭然になってしまうのだ。それを文字通り**盗み見て**新しいコマンドやオプションの使い方を覚えたのである。

もちろん、これは正統なUNIXの学習法とはいえないし、人の作業を覗き見するのは誉められた行為ではないのだが、実際、なかなか効率のよい方法だった。ただし、どんな場面でも使えるとは限らず、システム管理者の方針によってはwコマンドが使えないようになっている場合もある。また、もちろん1人で専用に使っているLinuxマシンでは、自分以外のユーザーはいないので、お手本にすることはできないだろう。

さて、話を元に戻すと、wコマンドの出力には、実行中のコマンド以外にも興味深い情報が含まれている。出力の先頭行には、現在の時刻、システムが起動してから現在までに経過した日数と時間、ログインしているユーザーの数、そして、システムの「負荷」が示されている。システムの「負荷」とは、同時にどれだけのプロセスが動いているのかを、一定時間ごとに平均した値で、通常3つの数字の組で表される。wの出力の場合、左から過去1分間の平均値、5分間の平均値、15分間の平均値となっている。これらの数値が大きくなればなるほど、システムが**忙しい**状態にあることを示す。個人で1台のマシンを使っている場合は、この数値が1を越えることはほとんどないのだが、X Window Systemでたくさんのアプリケーションを開いたり、プログラムのコンパイルなどの複雑な処理をさせた場合などには、数値はぐんと大きくなる。一般に、この数値が3を越えると「過負荷」の状態となり、ディスクアクセスが激しく起こって、マシンの速度が低下したように感じることだろう。



画面2　wコマンドの実行例
先ほどのpsコマンドよりも、かなり寂しい画面になるが、システムを起動してからどれだけの日数が経ったのかを見るのはまた楽しい。この日数が長ければ長いほど、「安定して動いている」ことの証になる。とはいえ、必要もないのにマシンの電源を入れっぱなしにするのは、電気の無駄遣いになるので戒めたい。

Linuxを生体解剖する

psやwコマンドよりもっとおもしろいのは**topコマンド**だ。コンソール画面で、

```
$ top
```

と入力してみよう。**画面3**のような一覧表が表示され、約5秒おきにその内容が変化するのがわかるだろう。一見すれば、この表が何を意味しているのかはだいたいわかるだろう。つまり、現在システム中で動いているプロセスを、**リアルタイム**で表示しているのだ。実際には、システム中のすべてのプロセスではなくて、CPUのパワーをもっとも多く消費しているプロセスから順に、画面に入るだけのプロセスを表示するしているのだが、その時点でシステムがどのような処理をしているかが一目でわかる、とても便利なコマンドなのだ。いわば、Linuxを**生きたまま**解剖して、臓器が蠢（うごめ）いているのを観察しているようなものだ。

X Window Systemの環境でtopコマンドを動かすと、とてもおもしろい。マウスを動かしたり、ウィンドウのメニューボタンを操作するだけでも、プロセス一覧に変化が起こって、どんなプロセスがどんな操作を担当しているかが、なんとなくわかってくる。Linuxの難しい知識がなくても、コンピュータが本当に**動いている**様子を眺めているだけでもおもしろいものだ。

topコマンドでは、個々のプロセス情報以外にも、システムの根幹に関わる有用な情報が数多く表示されている。画面に沿って、順に説明してみよう。ちょっとばかり専門用語も出てくるが、

細かいところは気にしないで、自分のマシンでtopコマンドを実行しながら読み解いてもらいたい。

まず、いちばん上の行には、システムの稼働時間や平均負荷の数字がある。これは、wコマンドの結果で表示されるものと同じである。ただし、5秒ごとに最新の数値に書き換えられる。

次の行には、システム上に存在するプロセスの総数に続いて、休止状態（「さぼっている」状態、sleeping）のプロセスの数、実行中（「働いている」状態、running）のプロセスの数、実行を終了したはずなのになんらかの理由でメモリ上に残っている（「ゾンビ状態」と呼ばれている）プロセスの数（zombie）、実行を一時的に中断しているプロセスの数（stopped）が表示されている。**ゾンビ状態**などという言葉が出てきて、**冗談なんじゃない？**と思う読者もいるかもしれないが、UNIXの世界ではまぎれもない**専門用語**なのである。詳しい話は難しくなるのでやめるが、要は、**死んだ（殺した）はずなのに、そこらをウロウロしているじゃまなプロセス**ということである。

ちなみに、UNIXやLinuxのパワーユーザーが、プロセスを**殺す**という言葉を使うのをよく耳にするかと思うが、プロセスを手動で停止させるためのコマンドは、文字通り**kill**コマンドである。

さらに次の行は、CPUの能力を、どのような仕事がどの割合で使っているかを示す数値だ。ここも専門的な話を始めると長くなるのだが、乱暴に言ってしまうと、「user」はアプリケーションのコードを処理するための時間、「system」はLinuxの本体（カーネルやデバイスドライバ）のコードを処理するための時間と考えてもらえば間違いない。

次の「nice」はアプリケーションのプロセスの内で、特別に高い優先権を与えられたものを実行するのに費やされた時間である。

最後の「idle」は文字通り、何もしていない時間のことである。ふつう、ひとりで1台のLinuxマシンを使っているような場合だと、このidle時間はほとんどの間、99%以上を示すことになるだろう。まあ、もったいない話ではあるが。

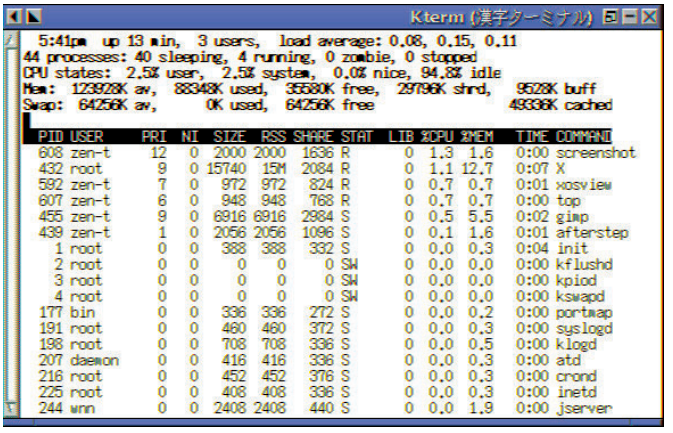
4行目と5行目には、メモリの使用状況が表示されている。4行目は「物理メモリ」すなわちマザーボードに刺さっているRAMの利用状況を示し、4行目は「スワップメモリ」を示す。スワップメモリとは、物理メモリがいっぱいになってしまったときに、ハードディスクをメモリの代わりに使おうとするものである。Linuxをインストールしたときに「スワップパーティション」というものを作ったのを覚えているだろう。

「av」は総容量（available）、「used」はアプリケーションによって使用されている量、「free」は空いている容量、「shrd」は複数のアプリケーションが共同で利用しているメモリ領域の量、「buff」は各種デバイスの入出力用バッファとして使われている量、「cached」はディスクキャッシュとして使われている量をそれぞれ示している。細かい数値にはあまり意味はないのだが、物理メモリの空き（free）が少なくなって、スワップメモリの使用量（used）が多くなればなるほど、システムは「重い」状態になっているといえる。

プロセス一覧の部分にもいろいろな数値があるが、特に興味を引くのは「%CPU」、「%MEM」、「TIME」の3つだろう。

%CPUはそのプロセスがCPUを使っている時間の割合、%MEMはそのプロセスが使っているメモリ領域の割合を示している。これらの数値が異常に大きいプロセスは、重要な処理を行っているか、そうでなければ**暴走している**プロセスと見なすことができる。

「TIME」は、そのプロセスがCPUを使用した実時間の合計を「分：秒」の形式で表示している。ふつう、プロセスというものは仕事を終えたらさっさと消えてなくなるものなので、TIMEの値も数秒程度までしか増えないものだが、たまに長生きして地道に仕事を続けているプロセスがあると、この数値が数百分以上になることもある。そんなプロセスを見つけて感慨に耽るのも一興だろう。



画面3　topコマンドの実行画面
プロセスが動いている様子を、まさに生きた状態で見られるのがtopコマンドだ。コマンドの実行に合わせて、リストの中身が書き換えられてゆくのを見るのは、理屈抜きに楽しい。ちなみに、topコマンドを終了するには「q」キーを押せばよい。このほか、topコマンドにはいろいろな機能があるが、「h」キーを押せば簡単なヘルプ画面を表示させることもできる。

日刊アスキーLinux on Linux magazine

日刊アスキー Linuxの裏舞台
～LinuxWorld Expo/Tokyo'99総力取材～

Linuxの総合情報サイト「日刊アスキー Linux(http://www.linux24.com/)の出張店舗というべき当コーナー。今月は、東京ファッションタウンで大々的に行われたLinuxWorld Expo/Tokyo'99の話題と、日刊アスキー Linuxで調査したディストリビューション人気についてご報告！

LinuxWorld Expo/Tokyo'99に編集部総出で取材に

去る9月29日から30日、東京ファッションタウンにて、ビジネス系のLinuxイベント「LinuxWorld Expo/Tokyo'99(以下LWET99)」が開催された。イベントの総合的な情報は本誌のニュースコーナーでも扱っているので、参照いただきたい。

日刊アスキー Linuxではこのイベントに対し、総力取材を行った。延べ2日間、技術担当の加賀を含めた編集部員4人すべてが総出で取材にあたったのだ。

朝から晩まで、編集部の人間がそれぞれワークショップや講演会場に入場したばかりでなく、ASCII network PRO誌で取材に出かける面々にまで「あとでお話聞かせてね」とお願いし、われわれの行けないワークショップに出席してもらった。

スタッフ全員が会場の東京ファッションタウンで記事を書く。

結果、24本のレポート記事と、5本の事前告知記事をアップロードすることができた。そして、これらの記事は、「LinuxWorld Expo/Tokyo'99スペシャル」と題して、日刊アスキー Linux内の特集という形で公開。会場に行けなかった人、行ったけれども時間が重なって興味のある講演が聞けなかった人には重宝していただいたのではないかと思う。

■Larry M.Augstin氏の講演に思うこと

LWET99は、ビジネス系のイベントだっただけに、企業の発表やワークショップ、講演が目立った。その中でも異色だったのが、「オープンソースとビジネス」といった話題では、VA LiNUX Systems社長兼CEO Larry M.Augstin氏の講演「ビジネスへのオープンソース開発テクニックの利用」である。すでにLinuxは「語り」の時代を過ぎている、といった風潮で、全体的に「ビジネスが云々」という講演は興味を持たれなくなってきているようだが、Augstin氏の場合は特別だったようで、ほかの講演に比べて人の入りもよかった。

ちなみに、VA LiNUX Systemsは Linux用のサーバマシンやクラスタシステムなどを製作、販売するメーカーだ。Augstin氏が学生時代に友達にLinuxマシンを作ってあげていたのが発端だったそうだ。

氏の講演は、なぜオープンソースソフトウェアが勝利したのかについて、開発者が圧倒的に多く、問題点への対策が早い、ソフトウェア入手のためのコストが低い、ソースコードが公開されているため、1社独占が起こりにくく、ユーザーに選択肢が与えられる点などを挙げた。面白かったのは、Microsoftが成功した背景を説明した話だ。すなわち、PC/ATというオープンな環境が、MS-DOSやWindowsの普及を促したのであり、Microsoftもオープン化によって成長した企業である、というわけだ。

全体で語られた内容を総合すると、以下の通り。

1. **オープンソースは1企業と比較すれば、人的リソースが豊富**
2. **オープンソースは、ユーザーに公平で豊富な選択肢を与える**
3. **オープンソースはソフトウェアに生き延びる道を与える（つまり、Microsoftが倒れたらWindowsは終わりだが、オープンソースはそんなことはない）**
4. **世の中はオープンに向かって進んでいる**

というところだろうか。会場の反応はというと、オープンソースをビジネスに採り入れるという話自体にとまどう人も多かったようで、質疑応答の時間でも、「オープンソースのコミュニティとビジネスはあまり合わないのでは？」といった質問も投げかけられていた。それに対し、Augstin氏は、「オープン

企業名	ワークショップなどで紹介された製品	サポートしているディストリビューション
コンパックコンピュータ	「コンパックLinuxコンピテンシセンター」を開設し、コンパックLinux-redyモデルの、Linux動作確認や、Linux事業におけるパートナーにテストラボ開放を行う。同社のPCサーバ「ProLiant」、「ProSignia」と、Alphaサーバ「AlphaServer」をLinux対応にし、「Linux-readyモデル」とする。コンパック製品をベースにLinux事業を展開する販売パートナーの支援体制を整える。Linuxのインストールサービス、導入後のヘルプデスク業務を行う。直販ルートにおけるLinux-readyモデルのアナウンス	TurboLinux Server 1.0日本語版、日本語redhat Linux 5.2、日本語redhat Linux 6.0 ServerEdition
日本アイ・ビー・エム	サポート&サービスとして、「Linuxサポートセンター」を設置。同社の「Windows NTサポートセンター」と協力し、共存環境のテストも行う。ソフトウェアサポートとして、「ロータス ドミノ」のLinux対応を行っているほか、「DB2」のLinux版リリース、「インターネット翻訳の王様」、「ホームページビルダー」のLinux対応版をリリース。ハードウェアサポートとして、「Netfinity」、「IntelliStation」、「ThinkPad」の動作確認を行い、Webにて情報を公開する>ディストリビュータとの協業として、独自ディストリビューションを出さず、あくまでもディストリビュータと緊密な関係を維持する。オープンソースコミュニティへの貢献として、RAIDドライバをGPLに準拠した形で公開。関連団体への貢献	世界的にはRed Hat、TurboLinux、Caldera、SuSE。国内ではLASER5 Linuxも含む
富士通（グループ全体として）	サポートでは、企画コンサルティングから、教育、運用段階におけるまでのサポートメニューを用意。小型のLinuxサーバ「CyberCOM PathNavigator」の発売。Fortran、C、C++をひとまとめにした開発環境「Fortran&C Package」、「C++ Package」の販売、サーバ管理システム「SystemWalker」の販売、ウィルス対策ソフト「InterScanVirusWall」販売、グループウェア「YellowTail」販売。Oracleのサポート	日本語redhat Linux、TurboLinux
サン・マイクロシステムズ	UltraSPARCのLinux対応(UltraPenguin、Debian GNU/Linux。Vine Linuxも対応予定) Linux用JDK「BlackDown」の開発。Star Officeの開発	Debian GNU/Linux、Vine Linux（予定）
NEC／NECソフトウェア	日本オラクルと提携し、「Oracle8i Workgroupe Server for Linux Release8.1.5」をインストールした「NEC Express5800」シリーズを出荷するとともに、上記製品を使用したコンサルティングと設計構築サービス「Oracle8iデータベースサーバソリューション」を、NECソフトウェアが展開する	TurboLinux、Red Hat Linux
日立製作所	ソリューションメニューの充実。OSインストール代行サービス。教育講座	日本語redhat Linux、TurboLinux
インプライズ	リレーショナルデータベース「InterBase」、Java開発環境「Jbuilder」	Red Hat Linux5.2、日本語redhat Linux5.2
日本ユニシス	Uniscriptの開発環境「TIPPLER」	Linux版は11月中旬予定
ジャストシステム	ATOK12 SE、一太郎Ark for Javaなど	
日本オラクル	Oracle8i Workgroupe Server for Linux Release8.1.5リリース、NECとの提携(NECの欄参照)、Oracleの資格「ORACLE MASTER Linux+」設置、アクアリウムコンピュータのマイクロLinuxサーバ「blue grass」製品発表	Red Hat Linux、TurboLinux
ロータス	Domino R5 for Linux	Red Hat Linux、Caldera OpenLinuxなど

表 各企業のLinuxへの取り組み

ソースビジネスを従来のビジネスにあてはめてはうまくいかないと思う」と答えている。

Linuxはビジネスとして浸透したが、オープンソースとなると話は少し違ってくる。今回LWET99に参加している企業でも、オープンソースを前面に出している企業は少ない（企業はまったくコミュニティに対して貢献しようとしていない、というわけではないが）。

「Linuxでビジネス」は成り立つが、「オープンソースでビジネス」となると、少なくとも現段階ではそれを行う企業やプロジェクトは限られてくる。現在「オープンソース企業」として認識されているRed HatやVA LiNUX Systemsも、Linuxブームになる前からLinuxを扱って脚光を浴びた、言ってみれば「Linuxネイティブ」な企業なわけで、ビジネスにおけるオープンソースの有効性を証明する材料として参考になるかどうかは微妙だろう。

各ソフトベンダーともLinuxの利点としてオープンソースを挙げつつも、自社の主力ソフトウェアはオープンソースではない。そしてまた、そうしたプロプライエタリ（ノンフリー）なソフトウェアのLinux参入がLinux市場を加速している状況というのは、今後どのような展開をみせるのだろう。

なお、「ビジネスとオープンソース」という主題では、日刊アスキー Linuxで樋口貴章氏に、コラム「Going For The One」を執筆していただいているので、ご覧いただきたい。また、木村誠氏のWebページ(http://www.

bekkoame.ne.jp/~kmakoto/)もお勧めする。木村氏は、経営情報学の専門家で、最近では、根来龍之氏とともに「ネットビジネスの経営戦略:知識交換とバリューチューン」（根来龍之・木村誠著　日科技連出版社　2600円　ISBN4-8171-6079-9）といった本も執筆されている。また、Web上では「オープンソーシングと知識コミュニティ論」といった論文や、「ネットビジネスのための経営戦略」といった論文を掲載している。そして、もちろん山形浩生氏のWebページ(http://www.post1.com/home/hiyori13/)は当然見るべき。また、日経コンピュータの1999年10月号でも、「オープンソースの幕が開く」という特集が組まれている。この日経コンピュータの特集は、かなり「オープンソースイケイケ」的な論調なのだが、日経独特の豊富な取材に裏打ちされた記事は読み応えがある。

■数々の製品発表

さて、LWET99では、大手メーカーの参加、発表も目立った。インプライズ、NEC、コンパックコンピュータ、サン・マイクロシステムズ、ジャストシステム、日本アイ・ビー・エム、日本オラクル、日本ユニシス、日立製作所、富士通などなど、挙げていったらきりがないが、顔ぶれだけを見れば、大規模なコンピュータ系のイベントと見まがうばかりだ。ここで、各社の戦略や発表した内容を横並びで見てみよう（表1）。

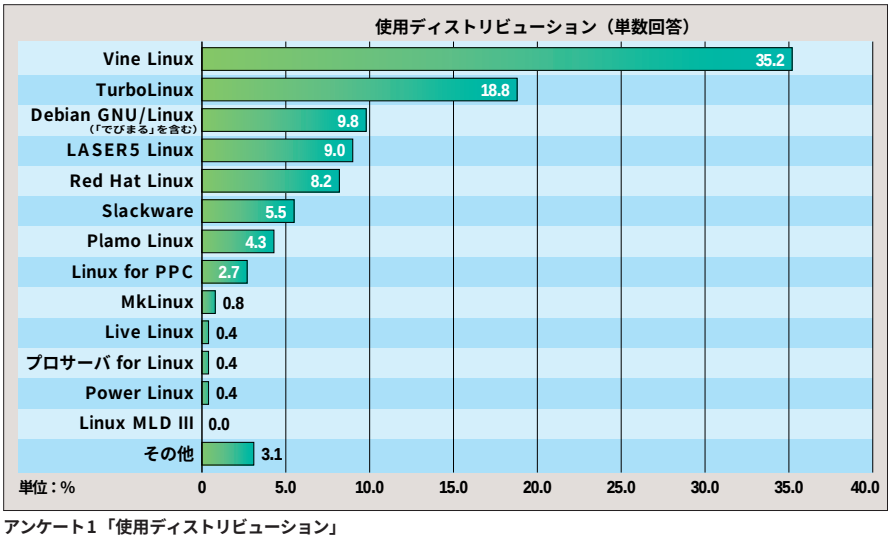
この表を見ていただければ、サーバ系からデスクトップまで、ひと通りのものが出そろった、という感を持つことができるだろう。メーカーがLinux対応を表明していた段階から、それらが次々とリリースされて、Windows NTにおけるシステムをそのままリプレイスできそうな段階になってきている。いままでは、Webサーバやメールサーバなどの、いわゆる「インターネット系サーバ」として使われていたLinuxが、分散環境構築のためのリーズナブルな選択肢として存在できるようになったといえるだろう。LWET99には、たしかに一時のような熱気はなかったのかもしれないが、その分、ワークショップに代表されるように、冷静に製品情報を仕入れよう、という雰囲気が色濃いイベントだった。

■みんなどんなディストリビューションを使っているの?

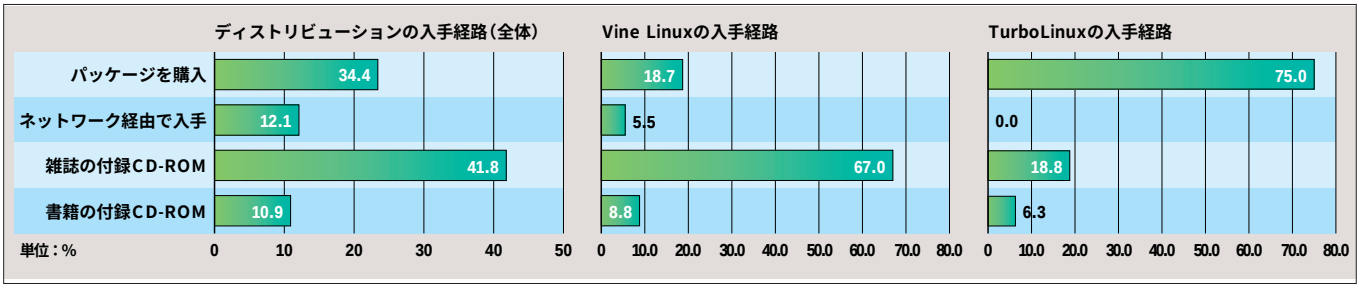
日刊アスキー Linuxでは、10日間隔でアンケートを開催している。テーマは毎回まったく別のものだが、10月6日～10月17日にかけては「Linuxのディストリビューションについて」である（アンケートに答えていただいた皆様、ありがとうございました）。

■Vine Linux強し!

さっそくだが、**アンケート1「使用ディストリビューション」**について見てみよう。1位がVine Linux、2位がTurboLinux、3位がDebian GNU/Linuxとなっている。Vine Linuxが圧倒的だ。筆者が個人的にもっていた印象としては、パッケージ販



アンケート1「使用ディストリビューション」



アンケート2「ディストリビューションの入手経路」

売の歴史が長いTurboLinuxかLASER5 Linuxかな、と思っていたのだが、Vine Linux強し、である。Vine Linuxは7月に日刊アスキーが調査した際にも1位だった。つまり、日刊アスキー上では、2度目の1位獲得なのだ。

もっとも、同様の調査を行っているLinux magazine編集部にたずねてみると、9月8日発売のLinux magazineアンケートはがきの集計結果は、Red Hat、TurboLinux、Vine Linuxがほぼ同数で並んでいるという。また、新しいバージョンが出るたびに順位が変わったりするという。このように、調査するメディアの性格によって数字が大きく変わるということには注意していただきたい。

また、Linux初心者にはあまりなじみのないDebian GNU/Linuxが3位というのは意外だ（日刊アスキーの植山は、「真打登場だ」とひとこと述べた）。メディアの露出度も低く、TurboLinux、Vine Linux、LASER5 Linuxは、パッケージ、書籍ともに目に付くのだが、Debian GNU/Linuxの場合、こうしたコマーシャル効果がそれほど期待できるとは思えない。

これは、当サイト読者層を見ると納得がいく。日刊アスキーにアクセスするのは、プログラマー、エンジニア、コンピュータ関連の専門・技術職、そして学生のみなさんが多い。結果、Debian GNU/Linuxもごく普通の選択肢として入ってきているのであろう（日刊アスキーとしては、初心者も大歓迎なのだが）。Debian GNU/Linuxを使う理由も、なかなか面白い。ほかのディストリビューションは、日本語環境や、メーリングリストが活発だから、といった点をあげていたのだが、Debian GNU/Linuxの場合は「フリーの精神をもっとも忠実に表わしているから」といったものや、「GNUの正統派だから」といった精神的な理由も混ざっていた。もっとも、大半の理由はやはり「パッケージ管理が強力だから」というものだったのだが。

■ディストリビューションによって分かれる入手経路

次の質問は、**アンケート2「ディストリビューションの入手経路」**だ。全体を見てみると、パッケージ購入と雑誌の付録というのが大半だが、ディストリビューションごととなると、くっきりと違いが現われてきた。Vine Linuxは雑誌付録が圧倒的で、TurboLinuxは逆にパッケージが多い。

やはり、雑誌のCD-ROM連動記事が目立ったVine Linuxと、ATOK12 SEなどをバンドルし、パッケージが華やかだったTurboLinuxの性格の違いだろうか。

■ディストリビューションの選択理由

国内のディストリビューション人気を左右するのは、日本語環境のようだ。「ディストリビューションを選択した理由は」という設問も設けたが、全体として「日本語環境が充実していると思ったから」そのディストリビューションを選択した、と答えた人が54.7%と半分以上あった。次がなんと「購入した書籍に付いていたから」。成り行きということなのだろうか。3位は「使っている人が多いと思ったから」というもので、納得できる理由である。「使っている人が近くにいたから」という回答と合わせると21%を超え、「ほかの人が使っているから」という意味では第2位になる。特に初心者の場合は、周りもしくはネットワークに教えてくれる人が大勢いるということは、非常に心強いだろう。

そして、ディストリビューション1位と2位、Vine LinuxとTurboLinuxはどうだろうか。両ディストリビューションのユーザーの選択理由を見てみると、それぞれ88%と60%と、どちらも突出しているのが、日本語環境の充実だ。やはり、何はなくともまず日本語が使えないと、メールを見たりWebをブラウズしたりといった面で困る。自分でなんとかするほど知識があれば話は別だが、最初にインストールするディストリビューションの日本語環境が貧弱であれば、もうお手上げである。

Vine LinuxとTurboLinuxの選択理由を自由回答から見てみよう。

Vine Linuxの選択理由は、信頼性や日本語環境の充実といった面が挙げられていた。また、開発者の実績を鑑みて、といったものや、Project Vineを応援しているから、と開発チームに対する信頼度も、選択理由となっているようだ。

パッケージ比率の多いTurboLinuxの場合、4.0にバンドルされているATOK12 SEなどのソフトウェアが貢献したといえそうだ。回答者からの選択理由としては、「昔から使っているから」、といったものや「使いたいアプリケーションが入っているから」といった答えもある。

全体の結果として、日本語環境の充実といった面以外では、機能に惹かれたり、バンドルソフトがほしかったり、心情的な理由だったりと、それぞれのユーザーがそれぞれの理由で好き勝手にディストリビューションを選択しているという、Linuxの「選択の自由」といった楽しさを反映したアンケートになったといえるだろう。

（日刊アスキー吉川）

Hardware Special

あなたのサーバの電源対策は万全ですか？

Linuxで使う小型低価格UPS

文：編集部
Text：Linux magazine
Photo : Shuichi Mito (Dee)

日本の電力事情は、諸外国に比べると安定していると言われている。大都市圏では電源電圧が大きく変動することや、停電することはほとんどない。

しかし、停電しなくとも電力会社の送電線切り替えにともなう瞬断が起こることもあるし、コピーやエアコンなど大きな負荷がかかる機器を使用したことで、一時的な電圧低下や、時にはブレーカが落ちることもあるだろう。そういうときに大事なサーバが止まってしまわないように、UPS（無停電電源装置）を導入しておきたい。

UPSはバッテリを内蔵し、停電時に数分から数十分間PCに安定した電源を供給する。また、UPS管理ソフトを使うことで停電を検出して、自動的にサーバをシャットダウンすることができる。今回はLinuxに対応するUPS管理ソフトと小型低価格なUPS装置を紹介する。

UPSの基本方式と特徴

UPSの基本的な動作は、AC（交流）電源からUPSのバッテリに充電しておき、停電や電源異常が起こった時に、バッテリからインバータを使ってDC－AC変換（直流から交流に）し、AC電源を供給するというものである。そしてインバータの動作状況や構造の違いによって4つの方式に分けられる。

■常時商用給電方式

平常時は入力のAC電源をそのまま供給し、停電時にバッテリからの出力に切り替えて電力を供給する。インバータは停電時にだけ動作するようになっているため、余計な電力を消費しない。停電時の切り替えの際にはインバータ起動のため最大10m秒ほどの瞬断がある。比較的安価で、停電時の電源

出力が矩形波である製品が多い。

■ラインインタラクティブ方式

常時商用給電方式を改良したもので、電圧補正回路（AVR）を内蔵し、トランスのタップを切り替えることにより、入力電圧に変動（80V～120Vぐらいまで）があっても正常な電圧を出力できる。平常時にもインバータ回路が動作しており、停電時の切り替えが短時間（4m秒程度）で行える。停電時の電源出力はノイズフィルターで歪みの少ない正弦波出力を得ている製品が多い。

■常時インバータ方式

常にAC入力を直流に変換し、それをインバータでACに再変換した出力を供給するため、入力からの影響を受けずに安定した電圧と波形を出力できる。停電時にも切り替えシーケンスがないので出力がとぎれたりしない。AC－DC－AC変換時にロスが発生するので効率が悪いということと、高価になるといったデメリットがある。

■パワーマルチプロセッシング方式

電圧補正回路を、トランスではなく電子回路で構成しているため、切り替え時間なしで安定した電圧を出力できる。停電時の切り替えの時にも出力が瞬断することがなく、常時インバータ方式に比べてロスが少ないというメリットがある。サンケン電気が開発した新方式で12月より対応製品が発売される。

UPS対応ソフトウェアが必要な理由

WindowsでもLinuxでもそうだが、終了作業を行わないで電源を切ってしまうとハードディスクの内容が壊れることがある。これはディスクへ書き込むデータをメモリ上にバッファリングしているので、実際にデータが書かれないうちに電源が落ちると、ハードディスク内のデータの整合性が取れなくなってしまうからだ。

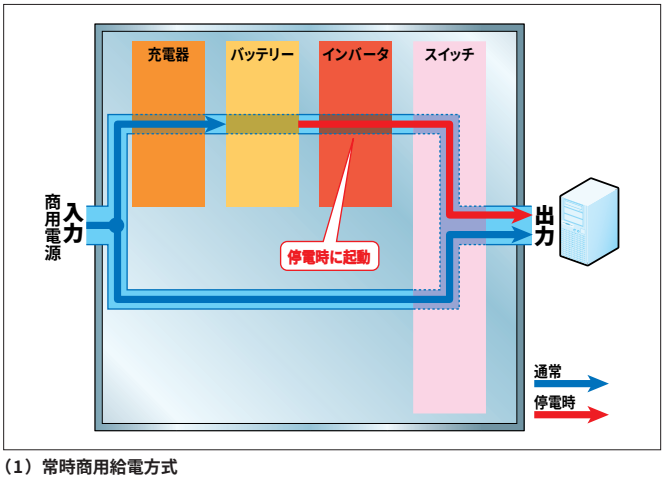
UPSは、電源障害時に内蔵のバッテリでAC電源を供給するが、バッテリの負荷容量いっぱいに機器が接続されていた場合、だいたい5分程度しか供給を続けられない。すなわち停電が発生したら、そのことを検知して速やかにコンピュータをシャットダウンさせるようにしないといけないわけだ。停電が発生したことは、UPSが一番よく知っているので、UPSはシリアルケーブルで停電をPCに通報し、PCが自動的にシャットダウンを行うという仕組みだ。

低価格なUPSではシリアルポートに、ON／OFFのシグナルだけで状態を示すのに対し、高機能なUPSではデータ通信を行うことで、UPS内部の情報をメッセージとして受け取ることができる。この機能を生かして、バッテリの充電状態やAC電源の入出力電圧と電流などをPCからモニタすることができるようになっている機種もある。

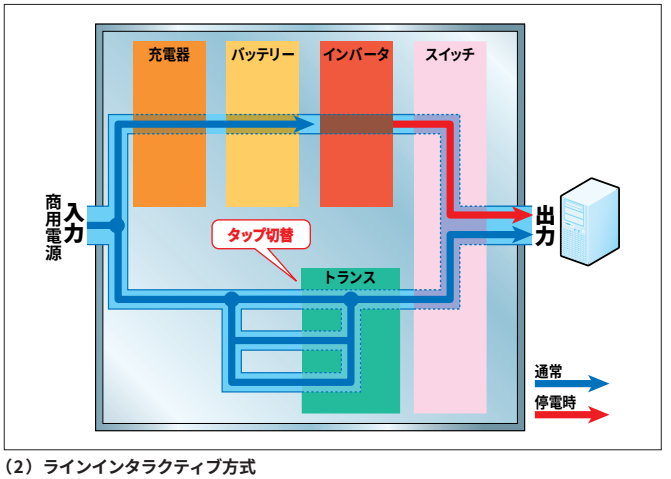
また、複数のUPSをコントロールする場合には、UPSごとに別のシリアルポートを用意するのは不便なので、LANを経由してUPSとメッセージ交換をすると便利である。上位機種の中には、そのためのオプションボードを内蔵できるUPSもあるので、複数のUPSを必要とするなら機種を選ぶときに注

運転方式の比較	通常時出力電圧精度	停電切り替え時の瞬断	効率	大きさ	重量	価格
(1) 常時商用給電方式	△	△	◎	◎	○	◎
(2) ラインインタラクティブ方式	○	○	○	△	△	○
(3) 常時インバータ方式	◎	◎	△	○	△	△
(4) パワーマルチプロセッシング方式	◎	◎	○	◎	○	○

表1 運転方式の比較



(1) 常時商用給電方式



(2) ラインインタラクティブ方式

意しておきたい点だ。

それではLinuxに対応しているUPS用ソフトウェアを紹介しよう。

Linuxに対応しているUPS管理ソフト

■apcupsd

apcupsdは、APCのすべてのUPSに対応しているUPS管理プログラムで、フリーソフトウェア（GPL2）として公開されているのでWebサイト（**画面1**）からダウンロードできる。また、LASER5 Linux 6.0やTurboLinux日本語版4.0、同PRO日本語版4.2の製品版には、RPM形式のファイルも収録されている。

apcupsdは、電源障害時のシャットダウン、シャットダウンまでの時間設定、ログ記録機能などがある。また、Smart-UPSのように内部のEEPROMにパラメータを設定できる機種に対しては、各種パラメータを/etc/apcupsd.confファイルに書いておくことで、トランス切替電圧、電源ノイズに対する感度、低バッテリ警告時間、再起動待機時間などを設定することが可能である。

なお、apcupsdはAPCが開発したものではないため、このソフトウェアの使用法についてAPCからのサポートは受けられないことに注意していただきたい。

■PowerChute plus for Linux

APCのSmart-UPSに対応しているUPS管理プログラムで、従来、WindowsやOS/2、NetWare、UNIX用に用意されていたPowerChute plusだが、新たにLinuxに対応した。APC社が開発し、11月末より同社のWebサイト（http://www.apc.co.jp/）から無償でダウンロードできる予定だ（**画面2**は開発中のもの）。

PowerChute plus for Linuxは、電源障害時のシャットダウン、ステータス監視、ログ記録機能、スケジュール運転、ネットワーク間でのUPS監視、UPSで発生するイベントに対するアクション設定、管理者やユーザーへの通知（メール）などの機能を持っている。

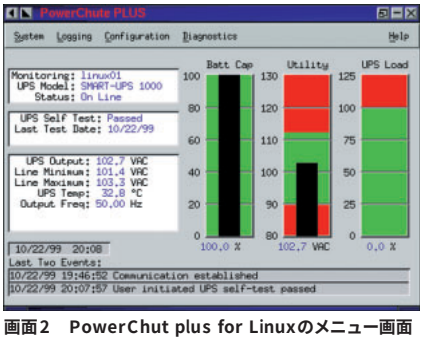
■FREQSHIP for Linux

三菱電機のFREQUPS-A、FREQUPS-Mシリーズに対応しているUPS管理プログラムで、フロッピーディスクにRPMパッケージが収められている。通信ケーブルが付属して、価格は1万1800円。

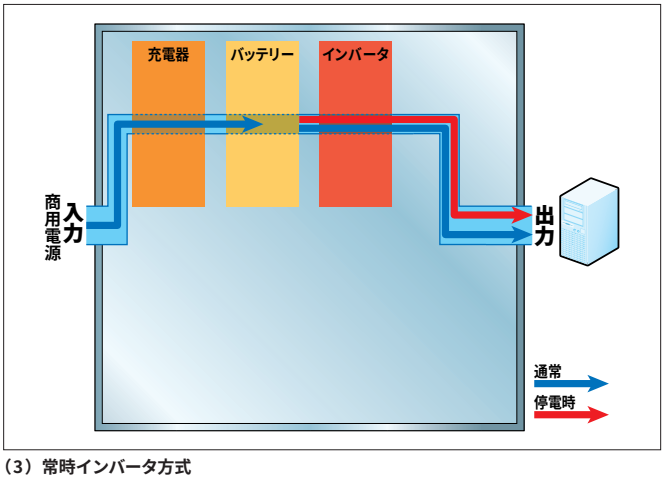
FREQSHIP for Linuxは、UPSの電源状態や負荷状態の監視、UPS環境の電圧や電流などの計測記録（データロギング）、スケジュール運転によるシャットダウンや起動、UPSのイベント発生時のシャットダウンやロギング、コマンドの実行を行うことができる。

FREQSHIP for Linuxの操作は、すべてテキスト画面で行うことができる

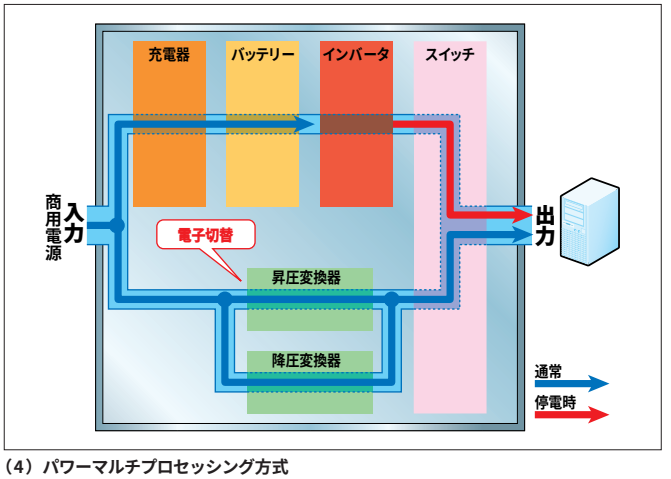
画面1　APC UPS Daemon Support Center
http://www.brisse.dk/site/apcupsd/



画面2　PowerChut plus for Linuxのメニュー画面



（3）常時インバータ方式



（4）パワーマルチプロセッシング方式

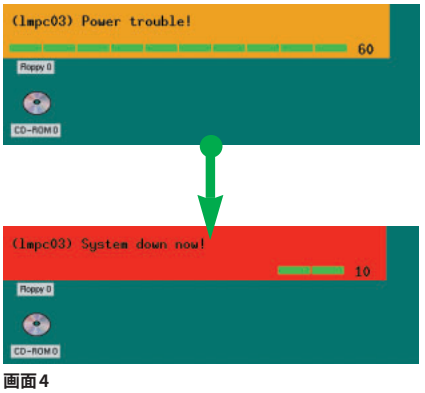
画面3　FREQSHIP for Linuxのモニタ表示画面

ため、ほかのマシンからtelnetでサーバにログインしてコントロールすることが簡単に行える（**画面3**）。

■FREQSHIP-mini

三菱電機のFREQUPS全シリーズに対応しているUPS管理プログラムで、オープンソースで公開している（http://www.nagoya.melco.co.jp/FREQUPS/down/download.html）。Linuxのほとんどのディストリビューションに対応しており、Cobaltマシン用のpkg形式ファイルも用意されている。

FREQSHIP-miniは、電源障害時のシャットダウン、シャットダウンの遅延時間の設定、UPS停止遅延時間の設定、システムのON／OFFスケジュールの設定（1回のみ）といった機能がある。FREQUPS-FシリーズとPシリーズでは、シャットダウンとシャットダウンの遅延時間の設定機能だけが可能となっている。



画面4
電源障害時には、X Window System画面の左上にアラートが表示される。指定時間が経過すると自動的にシャットダウンする

なお、Linux版のFREQSHIP-miniの設定は、コマンドラインで操作するようになっていて、メニュー形式の表示／設定とUPSの状態表示は行えない。

■パワーバイザv3 for Linux

パワーバイザv3 for Linuxは、GSEEのBIROS-mini GNXシリーズに対応しているUPS管理プログラムで、ディストリビューションごとに別のtarファイルが用意されている。通信ケーブルが付属し、インストール後3カ月間の電話、FAXによるサポート料金を含めた価格は1万7800円となっている。

パワーバイザv3 for Linuxは、停電時のオートシャットダウン、スケジュール運転によるUPSの起動と停止、OSのシャットダウン、複数台のUPSの一括管理（LAN対応）、電源障害時のSNMPトラップや電子メール送信機能などの機能がある。

パワーバイザv3 for Linuxは、X Window SystemのGUIで設定変更が行え、xupsというX用の監視プログラムを使用することで、電源障害を検出するとX Window System上に緊急メッセージを表示できる（**画面4**）。また、ほかのマシンからWebブラウザを使用して状態の監視（**画面5**）や簡易設定（**画面6**）を行うことも可能。

なお、パワーバイザv3の開発はアルファテックが行っている。

■パワーバイザS

パワーバイザSは、GSEEのGE3115シリーズにバンドルされているUPS管理プログラムで、Windows 95/98、Linux、FreeBSDに対応している。

パワーバイザv3のサブセットのような形で構成されており、複数台のUPSの管理とWebブラウザ経由で操作が行えないなどの点が違っている。インストールのしかたや操作方法、画面表示などは、パワーバイザv3とほぼ同様である。

■POWER MONITOR for Linux

POWER MONITOR for Linuxは、アルファテックが開発したUPS管理プログラムで、パワーバイザSとプログラム自体は同じもののようである。フリーソフトとして公開されており、同社のWebサイト（http://www.alfatech.co.jp/network/download/pmlinux.html）からダウンロードできる。

画面5
WebブラウザでUPSの計測値情報を表示可能。一定時間ごとに画面がリフレッシュされる

画面6
WebブラウザでUPSの簡単な動作条件を設定できる

APC Smart-UPSシリーズ、BK350／500

APC JAPANから発売されているAPC Smart-UPSは、ラインインタラクティブ方式のUPSで、400VA、700VA、1kVA、1.4kVA、2.2kVA、3kVAの6種類の出力容量を持つモデルが用意されている。価格は4万9800円～35万8000円。

どの機種を選べばよいのか？　まずバックアップする機器の消費電力と消費電流を調べる。UPSに接続するすべての機器を合計して、消費電力も消費電流もUPSの容量を超えないようにする必要がある。普通のミドルタワーPCやデスクトップPCの最大消費電力は、200～300Wなので、それを目安に計算すればいいだろう。UPSの能力より少な目の負荷で済ませば、長い時間バックアップできる。

Smart-UPSは入力電圧が81V～90Vになった場合にはSmartBoostという回路で電圧を12%上昇させ、110V～124Vになった場合にはSmartTrimという回路で電圧を12%減少させるという電圧調整機能を持っている（AP400Jを除く）。バッテリ使用時には、正弦波でAC100V±5%と安定した電源を供給する。

AP400J以外のモデルでは、前面のパネルを簡単に外すことができ、システム稼働中であってもユーザーがバッテリの交換を手早く行える。背面のスロットにオプションカードを挿すことで、さまざまな機能を追加可能である。オプションカードには、Webブラウザでの管理が行えるLAN対応カードや、2ポートのシリアルインターフェイスカード、温度・湿度などを監視するモジュールなどが用意されている。

管理ソフトには、PowerChute Plusがオプションで用意されていて、Windows 95/98/NT、NetWare、UNIX、OS/2、SGI、VMS、Macintoshに対応する。Linuxには、11月よりSmart-UPSに対応するPowerChute Plus for Linuxが同社のWebページから無償ダウンロードできるようになる。

■BK350／500

BK350／500は、小型、低価格なUPSである。BK350Jは350VA/210W、BK500JSは500VA/300Wの出力容量で、常時商用給電方式を採用している。

価格は、BK350が2万800円、電源管理ソフトPowerChute plus for Windows NT/95/98をバンドルしたBK500JSが2万7800円となっている。

株式会社APC JAPAN
TEL 03-5434-2021
http://www.apc.co.jp/

Smart-UPSシリーズ

型番	BK350J	BK500JS
運転方式	常時商用給電方式	
容量	350VA/210W	500VA/300W
入力電圧範囲	AC 100V	
バックアップ時出力	AC 100V、矩形波出力	
外形寸法（W×H×D、mm）	89×150×356	
重量	7kg	8.2kg
対応ソフトウェア	オプション(4000円)	PowerChute plus for Windows NT/95/98バンドル
対応OS		Windows NT/95/98
価格	2万800円	2万7800円

BK 350／500の主な仕様

型番	AP400J	SU700J	SU1000J	SU1400J
運転方式	ラインインタラクティブ方式			
容量	400VA/250W	700VA/450W	1000VA/670W	1400VA/950W
入力電圧範囲	81V～124V			
バックアップ時出力	AC 100V±5%、正弦波出力			
外形寸法（W×H×D、mm）	302×57×394	137×158×358	170×216×439	170×216×439
重量	7.7kg	13.1kg	20kg	24.1kg
対応ソフトウェア	PowerChute Plus			
対応OS	Winodws 95/98/NT、NetWare、Macintosh、OS/2、UNIX、HP-UX、VMS、SGI、Linux			
価格	4万9800円	8万5800円	12万2800円	15万2800円

Smart-UPSシリーズの主な仕様

三菱電機 FREQUPS-Aシリーズ、FREQUPS-Fシリーズ

三菱電機から発売されているFREQUPS-Aシリーズは、ラインインタラクティブ方式のUPSで、700VA、1kVA、1.4kVA、2.2kVAの3種類の出力容量を持つモデルが用意されている。価格は7万9800円～27万8000円。

AVR機能による電圧補正回路で、81V～124Vの入力に対応し、バッテリ使用時には、正弦波でAC100V±5%と安定した電源を出力する。

UPSの出力を2系統に分けて給電でき、時間差ON／OFFや、片方の出力する停止時間をコントロールすることで、停電時にまず周辺装置へのバックアップを止め、残ったバッテリでサーバマシンをシャットダウンするといった使い方ができる。

管理ソフトには、FREQSHIP for Windows、FREQSHIP for UNIX、FREQSHIP for Linux、FREQSHIP for CQ2がオプションで用意されていて、Windows 95/98/NT、HP-UX、Solaris、Linux、Cobalt Qube2に対応する。

また、Linux／FreeBSDに対応したFREQSHIP-miniをオープンソースとして公開していて、同社のWebサイトから無償でダウンロードできる。FREQSHIP-miniはOSのシャットダウン、1回だけ設定可能なスケジュール機能を持っている。

■FREQUPS-Fシリーズ

FREQUPS-Fシリーズは、厚さ45mmとスタイルを重視した薄型のケースを採用し、縦置きも可能となったコンパクトUPSである。出力容量が350VA/210W、500VA/300Wの2モデルで、常時商用給電方式を採用している。価格はそれぞれ、2万9800円、3万9800円。通信ケーブルも付属している。

管理ソフトは付属しないが、同社のホームページからダウンロードできるFREQSHIP-miniを使うことで、Linux、FreeBSD、Windows95/98に対応する。また、Windows NT、NetWareを使用している場合には、OSの標準機能を利用した自動シャットダウンが可能である。

FREQUPS-Fシリーズ

三菱電機株式会社
TEL 03-3459-5693
http://www.nagoya.melco.co.jp/FREQUPS/

FREQUPS-Aシリーズ

型番	FW-F10-0.3K	FW-F10-0.5K
運転方式	常時商用給電方式	
容量	350VA/210W	500VA/300W
入力電圧範囲	AC 100V±10%	
バックアップ時出力	100V±10%、矩形波出力	
外形寸法（W×H×D、mm）	45×280×275	
重量	4kg	
対応ソフトウェア（オプション）	FREQSHIP-mini	
対応OS	Windows 95/98/NT、NetWare、Linux、FreeBSD、Cobalt	
価格	2万9800円	3万9800円

FREQUPS-Fシリーズの主な仕様

型番	FW-A10-0.7K-01	FW-A10-1.0K	FW-A10-1.4K
運転方式	ラインインタラクティブ方式		
容量	700VA/490W	1000VA/700W	1400VA/1000W
入力電圧範囲	81V～124V		
バックアップ時出力	AC 100V±5%、正弦波出力		
外形寸法（W×H×D、mm）	140×160×360	170×220×440	
重量	17kg	21kg	26kg
対応ソフトウェア（オプション）	FREQSHIP、FREQSHIP for Linux、FREQSHIP-mini		
対応OS	Windows 95/98/NT、NetWare、HP-UX、Solaris、Linux、FreeBSD、Cobalt		
価格	7万9800円	11万8000円	14万8000円

FREQUPS-Aシリーズの主な仕様

GSEE GE3115シリーズ、BIROS-mini GNXシリーズ

GSEEから10月1日より発売されているGE3115シリーズは、パソコン用の小型UPSで、300VA、420VA、650VAの3種類の出力容量を持つモデルが用意されている。価格は2万4900円～4万4900円。

常時商用給電方式を採用し、無騒音、省エネタイプであり、300jと420jは幅87mm、650jは幅119mmとコンパクトなサイズになっている。ユーザーによるバッテリ交換が可能で、バッテリ寿命を警告する機能をもっている。

管理ソフトには、Windows 95/98、Linux、FreeBSDに対応するパワーバイザSをバンドルし、通信ケーブルも添付されている。停電時の自動シャットダウンとスクリプト実行、イベントログ機能、停電・電力障害時のメール送信、SNMPトラップ送信などの機能がある。

パワーバイザSは、X Window System上で作業中に停電があった場合、画面の左上にアラームを表示する。

■BIROS-mini GNXシリーズ

BIROS-mini GNXシリーズは、ラインインタラクティブ方式のUPSで、700VA、1000VAの2種類の出力容量を持つモデルがあり、価格はそれぞれ8万5000円、12万2000円となっている。

電圧補正回路で、81V～124Vの入力に対応し、バッテリ使用時には、正弦波でAC100V±5%と安定した電源を出力する。

UPS管理ソフトには、オプションのパワーバイザv3で、Windows NT、UNIX、FreeBSD、Linuxに対応し、停電時のオートシャットダウン、スケジュール運転によるUPSの起動と停止、OSのシャットダウン、複数台のUPSの一括管理（LAN対応）、電源障害時のSNMPトラップや電子メール送信機能などの機能がある。

Linuxに対応した電源管理ソフトパワーバイザv3 for Linuxと通信ケーブル、インストール後3カ月の電話、FAXによるサポート料金を含んだサポートパッケージが1万7800円で用意されている。

BIROS-mini GNXシリーズ

ジーエス・イーイー株式会社
TEL 03-3502-6554
http://www.gsee.co.jp/

GE3115シリーズ

型番	BM700 GNX	BM1000 GNX
運転方式	ラインインタラクティブ方式	
容量	700VA/460W	1000VA/660W
入力電圧範囲	81V～124V	
バックアップ時出力	AC 100V±5%、正弦波出力	
外形寸法（W×H×D、mm）	145×160×370	164×175×390
重量	約14kg	約19kg
対応ソフトウェア	パワーバイザーv3 for Linux（オプション）	
対応OS	Linux*	
価格	8万5000円	12万2000円

＊オプションのパワーバイザーv3で、Windows NT、UNIX、FreeBSDに対応

BIROS-mini GNXシリーズの主な仕様

型番	GE3115-300j	GE3115-420j	GE3115-650j
運転方式	常時商用給電方式		
容量	300VA/180W	420VA/252W	650VA/400W
入力電圧範囲	85V～117V（DIPスイッチ切替可能）		
バックアップ時出力	AC 100V±5%、矩形波出力		
外形寸法（W×H×D、mm）	87×152×323		119×165×356
重量	5.2kg		7.5kg
対応ソフトウェア	パワーバイザーS標準装備（通信ケーブル添付）		
対応OS	Windows 95/98、Linux、FreeBSD、（Windows NT Workstation予定）		
価格	2万4900円	2万9900円	4万4900円

GE3115シリーズの主な仕様

サンケン電気 FULLBACK SMUシリーズ、SCU501

世界初の給電方式「パワーマルチプロセッシング方式」を採用したPCサーバ、UNIXサーバ向けの小型小容量UPS、FULLBACK SMUシリーズが、サンケン電気から10月20日に発表された。出力容量が750VA、1kVA、1.5kVA、2kVA、3kVAの5種類で、それぞれ出力電圧精度が±2%のハイグレードモデルと、±10%のエコノミーモデルの2タイプが用意されており、12月上旬から出荷される。価格は7万6000円～47万1000円。

FULLBACK SMUシリーズは、パワーマルチプロセッシング方式により93%という高効率を実現している（常時インバータ方式では83%前後）。電圧制御に半導体素子を使用しているため、80V～144Vというワイドレンジの入力が可能となっており、入力電圧の変化に対し瞬時に応答し、正弦波で安定したAC出力を得ている。

本体背面の出力タップを複数（2または3系統）に分割し給電することも可能で、停電時にクライアントマシンを先に、サーバマシンを後でシャットダウンするといった使い方が考えられる。

ハイグレードモデルは、ネットワークに対応しており、複数のUPSの一括管理や、複数UPSの出力を並列に接続して電源を二重化する冗長運転も可能になっている。

管理ソフトには用途に合わせて、FULLBACK Manager、FullBack NetAgent、LanSafeIII／FailSafeIIIがオプションで用意されていて、Windows 95/98/NT、NetWare、UNIX、Linux、OS/2、Windows2000に対応する。

■FULLBACK SCU501

FULLBACK SCU501は、幅70mm、3.9kgという小型コンパクトなUPSである。500VA／300Wの出力容量で常時商用給電方式を採用している。

従来の約2倍の寿命を持つバッテリを搭載しているので、バッテリ交換は5～6年に1度で十分である。

価格は、SCU501が2万9800円、電源管理ソフトLanSafeIII／FailSafeIIIをバンドルしたSCU501-S1が3万4000円となっている。

サンケン電気株式会社
TEL 03-3986-6150
http://www.sanken-ele.co.jp/

FULLBACK SMUシリーズ

型番	SCU501	SCU501-S1
運転方式	常時商用給電方式	
容量	500VA/300W	
入力電圧範囲	AC 100V±10%	
バックアップ時出力	AC 100V±10%、矩形波出力	
外形寸法（W×H×D、mm）	70×164×248	
重量	3.9kg	
対応ソフトウェア	オプション	LanSafeIII／FailSafeIIIバンドル
対応OS		Windows 95/98/NT、NetWare、OS/2
価格	2万9800円	3万4000円

FULLBACK SCU501シリーズの主な仕様

型番（エコノミーモデル）	SMU-E751	SMU-E102	SMU-E152	SMU-E202
型番（ハイグレードモデル）	SMU-H751	SMU-H102	SMU-H152	SMU-H202
運転方式	パワーマルチプロセッシング給電方式			
容量	750VA/525W	1000VA/700W	1500VA/1050W	2000VA/1400W
入力電圧範囲	80V～144V			
バックアップ時出力	AC 100V、正弦波出力			
外形寸法（W×H×D、mm）	120×295×380		150×330×380	190×450×450
重量	14kg	16.5kg	22kg	35kg
対応ソフトウェア（オプション）	FULLBACK Manager、FullBackNetAgent、LanSafeIII/FailSafeIII			
対応OS	Windows 95/98/NT、NetWare、UNIX、Linux、OS/2			
価格（エコノミーモデル）	7万6000円	10万9000円	13万8000円	21万9000円
価格（ハイグレードモデル）	10万8000円	12万9000円	19万6000円	31万1000円

FULLBACK SMUシリーズの主な仕様

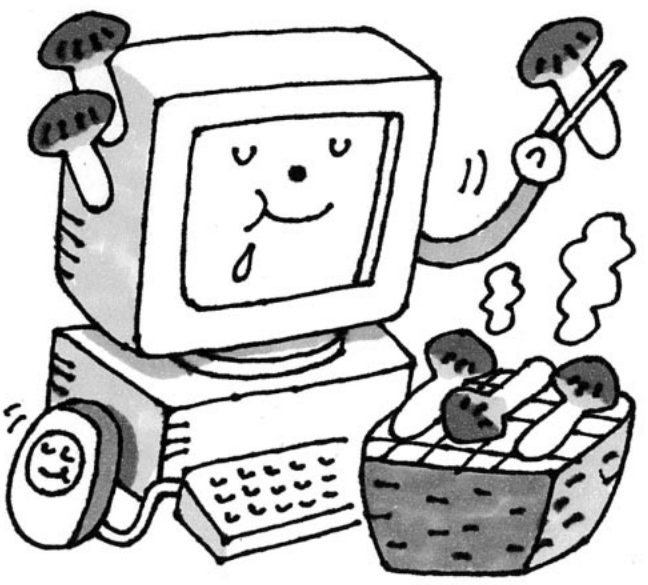
※ エコノミーモデルは出力電圧精度が±10%、ハイグレードモデルは出力電圧精度が±2%

viバージンに贈る超入門講座

viはじめました

第4回　exコマンドでもっと便利に

今回で連載4回目。よく使うコマンドは一通り覚えたことになります。これまででも十分実用的な編集ができるはずですが、今回はもっと複雑なことをこなせるようになりましょう。いままであまり触れてこなかったexコマンドを使うと、viに行番号を表示させたり、行の移動や置換ができるようになります。



文：佐々木太良
Text : Taroh Sasaki

illustration ; Manami Kato

●はじめに

早いものでこの連載もすでに4回目を迎えることになりました。ちょっと飛ばし過ぎのペースかな、とも思うのですが、自分で試しながらあれこれやっている方にはスローペース過ぎるでしょう。今回はexコマンドを中心に説明していきますが、そろそろvi以外のUNIX系OSの各コマンドとの連携プレーなども出てきます。自分でもどんどん面白い使い方を見つけてください。

●exコマンド

exコマンドについて簡単におさらいしておきます。一般にviと呼ばれるエディタは、exと呼ばれるエディタと一心同体です。一心同体というのは、実体は同一なのに、起動されたときの名前によって動作が変わってくる、という意味です。同一人物なのに歌手のときはユーミンで、作詞家であるときは呉田軽穂であるようなもんですね。ってよくわからんか。

そんなわけで、vi←→exへの移行は簡単にできますし、viの中にもexを一時利用する方法があるわけです。viモードの中からは[:]を押すと1行だけexコマンドを実行してviモードに戻り、[Q]でそのままexになっちゃうのでしたね（exモードからはexコマンド『vi』でviモードに戻れます）。

さてここまではよろしいでしょうか。以下の説明では、いちいち『viモードから[:]を押して～を入力し、最後に[Enter]（リターン）』という説明はしません。『～を入力』の部分しか書きません。ちなみにこれもおさらいになりますが、exコマンドの最後の[Enter]は、[ESC]と操作してもそのまま動くのでした。

●setコマンド

setコマンドはすでに一部紹介しました。このコマンドは規則があって、

set 動作（または省略形）
set no動作（または省略形）
set 変数（または省略形）＝値

というタイプに分けられます。たとえばすでに出てきた『常に行番号を表示するモードにする』というのは、『set number』ですが、『number』は省略形『nu』なので『set nu』とも書けます。行番号の表示をやめたいときは『set nonumber』になりますが、したがって『set nonu』とも書けるわけです。ではsetパラメータを全部表示させてみましょう。

```
:set all
```

こんな感じで出てきたと思います（**図1**）。最下行にある通り、通常の編集に戻りたければ何かキーを押す（[:]で次のexコマンドが実行できる）ことになってますから慌てて編集部に電話しないように。また、setしまくってどこを変更したかわからなくなる場合があります。こんなときは『set all』ではなく『set』としてみましょう。viのデフォルトで定められている動作から変更したところだけが表示されます。

これらの表示は実はすごく親切で、もし表示されている動作をオンオフし

たり、パラメータをいじったりしたいときは上にある表示をそのまま『set 表示の通り』という書式で書けばよいわけです。イコールが必要かどうか、ダブルクオートで囲まなければならないかどうか、などは自信がなければ『set all』してみればわかるわけです。なお、viにはいろいろなバージョンがありますので、皆さんのviで必ずしもこうなるとは限りません。では、大半のviにあっていじると役に立つ（あるいは人に自慢できる）パラメータを見てみましょう。なお、[no]は機能をオフにするときに付け、カッコ内は省略形です。

●オン・オフ動作をするもの

・[no]number （nu）

前記にあるように行番号表示モードです。行番号を付けてC言語のソースを編集すると、怖い先輩に殴られます（笑）。さらに最近は、

・[no]ruler （ru）

なんておせっかいなものまであるんですね……（笑）。

・[no]autowrite （aw）

これは復習になりますが、ファイルがダーティなときに編集終了／次のファイルを編集しようとした／一時停止をするとき、などに自動セーブしてくれるモードです。使い方と使う人の性格によっては危険です。

・[no]autoindent （ai）

行頭をTABで字下げ（インデント）している場合、改行時に次の行頭も同じ深さまで下げてくれる機能です。やはりC言語のソースを書く人はほとんどオンにしているようです。後述のtabstopの設定も参照してください。

なおこれをオンにしていて、次の行のインデントを浅くしたい場合は、[BS]でTAB文字が1つ消せます（[^D]で行頭まで全部のTAB文字が消せます）。

・[no]showmatch （sm）

これをオンにしていると、『カッコ閉じ』関係を打ったときに、『カッコ開き』文字にカーソルを一瞬だけ移動して示してくれます（タイピングの邪魔はしません）。これもプログラマー好みかな?

・[no]wrapscan （ws）

ファイルの最後まで検索したとき、次に先頭に戻ってくれます。

・[no]readonly （ro）

そそっかしい人の友です。その名の

```
:set all
--- Options ---
noautoindent          noignorecase            scrolljump=1          textwidth=0
noautowrite           noincsearch             scrolloff=0         notildeop
  background=light    noinfercase           nosecure                timeout
  backspace=0         noinsertmode            selectmode=           timeoutlen=1000
nobackup                isprint=@,161-255     shell=/bin/bash     notitle
  backupext=~           joinspaces            shellcmdflag=-c       titlelen=85
nobinary                keywordprg=man        shellquote=           titlestring=
nocindent               keymodel=             shelltype=0         nottimeout
  cinoptions=           laststatus=1          shellxquote=          ttimeoutlen=-1
  cmdheight=1         nolazyredraw          noshiftround            ttybuiltin
  columns=80          nolinebreak             shiftwidth=8        nottyfast
nocompatible            lines=24              shortmess=            ttyscroll=999
  complete=.,b        nolisp                noshortname             ttytype=vt100
noconfirm             nolist                  showbreak=            ttymouse=
  cpoptions=aABceFs     listchars=eol:$     noshowcmd               undolevels=1000
  dictionary=           magic               noshowfulltag           updatecount=200
nodigraph               makeprg=make        noshowmatch             updatetime=4000
noedcompatible          matchtime=5           showmode              verbose=0
  endofline             maxfuncdepth=100      sidescroll=0          viminfo=
  equalalways           maxmapdepth=1000    nosmartcase           novisualbell
  equalprg=             maxmem=5120         nosmartindent           warn
-- More --
```

<図1>

通り、ファイルを書き込み禁止にします。元々書き込む権限がないファイルはこのモードでオープンされています（が、『set noro』しても書き込めるようになるわけではありません）。

●変数になっているもの

・tabstop（ts）＝数字

編集ファイル中にTAB文字があると、空白として表示されますが、TABの次の文字が開始される位置（タブストップ）が画面上で何文字おきにあるかを設定します。デフォルトは8になっているviが多いのですが、C言語ハッカーは4を好むといわれています（一部には5という宗教も存在します）。またGNUのソースなどでは1になっている（スペースとTABの区別があまりありませんね）ものも多く見かけます。

・[no]wrapmargin（wm）＝数字

これは文章を書く人向け。数字で指定した右マージンを残して自動的にラップ（一定の長さで改行する）してくれます。なお、英語の場合は必ずスペースまたはTABがラップの合図となります（単語中では切れない）が、日本語化されている場合、日本語文字列の途中で自動ラップするかどうかはviのバージョンによって異なるようです。

このほか、仔細にみてみるとまあ実にいろいろありますね……パラグラフの切れ目（[{][}]で移動できる）に文字列が設定できるのは、nroff （文書整形ツール、UNIX系のmanを書くために便利）の段落区切りがコマンド文字列で指定されるからです。また、バックアップファイルの作成設定やその拡張子、各種表示や応答の細かさ、一時ファイルの作成先（前回登場）、[ESC]キーの反応速度の変更（カーソルキーを使えるようにするかどうかに関係する場合があります）なんてものまであります……。

●外部コマンドを起動する（exコマンド／viコマンド）

ファイルの編集中に今月のカレンダーが見たくなったらどうするでしょうか。「別のウィンドウがあるし」「xcalendarがいつもルートウィンドウにあるもん」…退場。exコマンドから、

```
!cal
```

としてみましょう。先頭に[:]を打ってexモードに入るのを忘れないでください。というのも、[!]というviコマンドがあるのです。しかもそれは、外部コマンドを起動するだけではなく、その範囲を外部コマンドに渡し、結果で指

```
the end of the millenium
 you and i will be together.
it's not the end of the century.
~
~
```

↓

```
the end of the millenium
   December 1999
Su Mo Tu We Th Fr Sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
it's not the end of the century.
~
~
```

※ ■はカーソルの位置です。

＜図2＞

Column

viの操作説明の難しさ

この連載でもたびたび触れてきましたが、viにはいくつかのバージョンがあり、細かい部分の操作（しかもviが改良されてきた経緯で追加されたコマンドなのでかなり便利だったりする）が違っています。読者の皆さんも「私のviと違う！」っていうことがあったかもしれません。申し訳ありません（私は悪くないけど）。

ところが今回触れたsetコマンドを次回『.exrc』ファイルの設定と組み合わせれば、さらに同じバージョンのviでも全く挙動が異なって見えたりすることがあります。たとえば本文中では、『set autowrite』などは重要なので、セットした場合、していない場合の違いをあげていますが、それ以外はあまり詳しく説明していません。ところがset一発で全然違うエディタに見えてしまいます。パラグラフの区切り目の設定のほか、[w]系コマンド（第3回）でカーソルのジャンプのしかたの挙動を変更したり、改行したときに行頭のマージン幅、行番号が付くか付かないか、などがこれにあたります。この連載に限らず、説明と実際に行った結果が異なっていたら、viのバージョン（種類）が違うのか、どこかでカスタマイズできるのか（あるいは説明している本の間違いか）、うやむやにせずに解決することを心掛けてください。viウィザー　ドへの早道だと思います。

これは何のツールでもそうなのですが、結局は自分で便利な使い方を探していくのが『はっかー道』です。ある程度ツールを使い込んで「これは便利だよ」という技を後輩に教えてあげようとすると、そのツールのデフォルトの使い方からはだいぶかけ離れてしまったり（本やマニュアルとは挙動が違っていたり）、かといって便利な機能を知らずにデフォルトのまま使っていると不幸、ということもあり得ます。

さらにカスタマイズしまくると、今度はカスタマイズしていないデフォルトのものを体が受け付けなくなったりします。そう、私は他人のmuleとシェルと歯ブラシは使いたくありません……。

定した範囲を置き換えてしまうのです。範囲の指定は移動コマンドを組み合わせるか（先月号参照）、または[!][!]でその1行の置き換え（数字[!][!]で数字行分の置き換え）です。[d]や[y]などと同様ですね。たとえば**図2**上のような文章があったときに、

```
[!][!]cal 12 1999[Enter]
```

としてみると、**図2**下のようになります([Enter]は[ESC]でも可)。
※注意：今度は最初の[!]はviコマンドですので、[:]は打ってはいけません。exコマンドで同じ動作をさせるときは、[:][.][!]cal 12 1999[Enter]になります。なかなか便利ではありませんか。これはたとえば、

```
[:][1][,][$][!]sort
```

として現在編集中のテキストをソートするというような使い方を想定した機能なのです。

●置換・削除・移動

このあたりは、実は正規表現を知っていないとかなり厳しいです。と脅しておきますが、実は限られた仕事しかしないならば、限られたコマンドだけ覚えておけばこれまた便利でしょう。詳しい正規表現の使い方はマニュアルを見るほか、sed、awk（gawk）、Perlなどのツールの使い方のチュートリアル本にもよく解説されていますので、そちらを参照したほうがよいでしょう。手抜きですか。でも逆に正規表現をマスターすることで、これらのツールを全部、使いこなせるようになるともいえます。

さて正規表現はsed…といいましたが、実はここで説明するある範囲の置換・削除・コピー・移動は、まさにsedそのものです。というかsedがedというエディタのパイプ版であり、exモードの元祖はedであることを考えると、チンパンジーとマントヒヒくらいは似ているでしょう（わからん）。

- **置換……範囲s/パターン1/パターン2/スイッチ**
- **削除……範囲d**
- **移動……範囲m行番号**

このほかにもたくさんのexコマンドがあります。しかも『s』『d』『m』というのは実は短縮形なのです（フルスペルのほうが覚えやすいですが、実際にタイプするときは皆『s』などを使っているようです）。特にこの3つがよく使うものなので（しかも置換と削除はそのままでsedの動作を決める命令になり得る）、これだけは死んでも覚えましょう。

○範囲

『行番号,行番号』の形だと思ってもらえば結構です。単純に『1,10』とすると、1行目から10行目までをナニします。番号の代わりに『$』を書くと、最後の行の行番号を書いたのと同じことです。さらに『/正規表現/』（『?正規表現?』）と書くと、現在いる行から直後の（直前の）正規表現にマッチした行を指定したのと同じことになります。さらにさらに、『1,$』に限っては『%』とだけ書けばよいというお約束もあります。

ちょっと分かりにくいので例を挙げてみましょう。

3,5d	**3行目から5行目まで削除**
6,10m$	**6行目から10行目までを最終行（の後）に移動**
%s/^/0/	**全部の行の頭を0に置換（行頭に0を挿入する）**

Column

正規表現ってナニ？

viが愛され続けている理由の38%（当社比）は正規表現が使えるからです。Windows上などの他のエディタなどでは、『初心者のため』を思い遣ってかどうか知りませんが、固定文字列の置換（最初の例で触れたような）しか使えないものがあります。また、正規表現『もどき』であっても正規表現と等価に機能しないものがあります。正規表現は一見面倒くさくできていますが、この表現にマッチする文字列の集合は数学的に規定できるというのが優れている点なのです。
『非決定性有限オートマトンで解析できる文字列』と書くとなんのこっちゃと思うでしょうが、かなり複雑な規則を持った多数の文字の並びを一括して表現できるということが、数学的に証明もされている、ということですね。計算機科学のプロには欠かせない常識となっています（まあviを学習する皆さんの全部が全部、計算機屋さんとは思いませんが）。
なおかつUNIX系のOSの標準ツールで用いられている正規表現は、理論的にどうこうというだけではなく、プラクティカル（実用的）な観点から見てシンプルに書いて使える、という点が優れています（こんな呪文みたいなののどこがシンプルじゃ、といわれそうですが）。
興味のある人は正規表現を覚えましょうと書きましたが、固定文字列の置換や『もどき』検索・置換のためだけならばわざわざ操作が面倒くさい（といわれる）viを使う理由はないのです。この意味で『viを使う人は正規表現を覚えなさい』と書きたいくらいです。日本国憲法に宗教の自由が保証されていなければ『国民は正規表現を覚えるべきだ』と書いているところです（うそ）。

これを見ると、まあ削除『d』と移動『m』については理解してもらえますね（ダメなら納得いくまで自分でviをいじってください）。

○置換の『パターン』や『スイッチ』

置換『s』については何か変なモノが出てきました。これはなんでしょう。『パターン』の中に書かれるものは正規表現と呼ばれるものです。文字『[].*+\』は特殊な振る舞いをするのですが、逆にいえばこれ以外の文字については、単なる置換だと思ってもらえば結構です。

```
%s/vi/ex/
```

＝すべての行に含まれる最初の『vi』を『ex』に置換する

では、**図3**上のテキストファイルについて置換を実行してみましょう。

あれ、2行目にはviが2カ所あるのに最初しか置換されていませんね。行内のパターンをすべて置換するためには、

Column

正規表現の種類

UNIX系OSの標準ツールであるed、sed、grep（egrep、fgrep）、vi／ex、Perlなど、またC言語のライブラリ関数であるregcomp()、reg()で扱える正規表現は微妙に違っています。

基本的に正規表現で表わせる文字列の範囲は変わらないのですが、どの文字を特殊文字とするか、あるいはプラクティカルな観点から見ていかに書きやすいか（同じ表現が短く書けるか）によってこのような違いが出てきています。

ここでは一般的な正規表現に従って書いていますが、viの実装によっては異なっている場合があるようです。また、いくつかの最近のviでは、正規表現のバージョンを切り替えることができたりするようです。

いずれにしても重要で広範囲にわたる置換を行う前には、簡単な例を1行作ってその行だけ置換してテストしてみる、などの心掛けが必要なようです（これはsedなどを使うときにも言えることで、筆者も複雑な正規表現を使う前にはいつもテストファイルを作ってテストしています）。

さて、単純置換については分かってもらえたと思いますが、正規表現を用いた置換については、以下に少々プラクティカルな例を載せるにとどめます。興味のある人はマニュアルページ、あるいは書籍などで意味を調べてみるとよいでしょう。

%s/^[<TAB><スペース>]*//	行頭の空白を削除
%s/\.$/!/	行末の『.』を『!』に置換 ※ピリオドは『\.』となっていることに注意
%s/.*/mv & &.bak/	各行に書かれた『a』『hoge』などを、 『mv a a.bak』『mv hoge hoge.bak』に置換 ※『&』は『/左辺/右辺/』のときの左辺を示す
%s/\(.*\)\.c$/mv & \1.bak/	各行に書かれた『a.c』『hoge.c』などを、 『mv a.c a.bak』『mv hoge.c hoge.bak』に置換 ※『\1』は『\(...\)』で囲まれた範囲の1番目を示す 同様に『\2』『\3』…も可能

表1

```
the last line of vi is used for you to
give commands to vi, and for vi to give
information to you.
~
~
(中略)
:%s/vi/ex/
```

↓

```
the last line of ex is used for you to
give commands to ex, and for vi to give
information to you.
~
...
```

図3

```
the last line of vi is used for you to
give commands to vi, and for vi to give
information to you.
~
~
(中略)
:%s/vi/ex/g
```

↓

```
the last line of ex is used for you to
give commands to ex, and for ex to give
information to you.
~
~
```

図4

```
the last line of ex/vi is used for you to
give commands to ex/vi, and for ex/vi to
give information to you.
~
~
(中略)
:%s+ex/vi+the editor+g
```

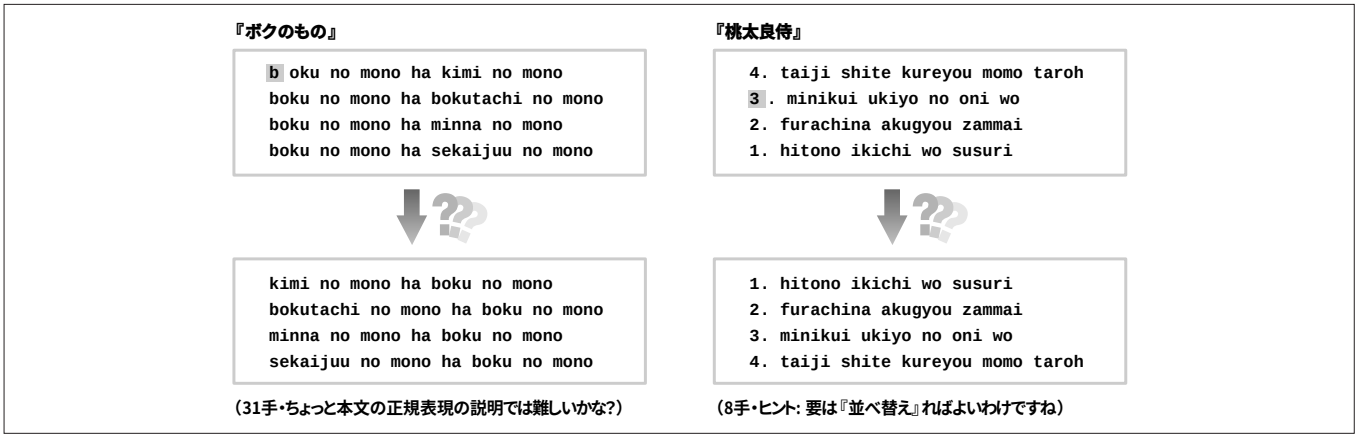
↓

```
the last line of the editor is used for you to
give commands to the editor, and for the editor to
give information to you.
~
~
```

図5

●詰めvi 第2回　（出題・たりゃー佐々木6段）

今回はコマンド・外部コマンドの導入ということで、かなり難しめの4級・3級（当社比）問題となっています



スイッチ『g』を付けます**（図4）**。

実は『s』の後の3つの『/』は、3つが同じであれば何でもよいので、置換される文字に『/』が含まれているときは別の文字を使います。

鋭い方ならすでに気付いているかと思いますが、viの検索コマンド[/][?]でも正規表現を使った検索ができるのです。したがってviの検索コマンド[/]および『d』『m』『s』各コマンドの『範囲』を示す『/…/』とお揃いでおしゃれをするために、『s』コマンドの『パターン』を囲む3つの文字も『/』を使うことが多い、というだけでしょう。

図5の例は『/』が含まれた文字列を置換するために、『+』を使った例です。

●viとシェルスクリプト

コラムの表の最後の例で出てきた置換の例は、実はファイル名のリストからシェルスクリプトを作っているのです。シェルスクリプトというと構える人が多いのですが、要はシェルに手入力で1つ1つ『mv a a.bak』……としてファイル名を変更しているのを、一挙に実行している（実行するようなプログラムを書いている）だけです。

私はよく、おもむろにviを起動して、

```
[!][!]ls
```

（ファイル名のないviのバッファにカレントディレクトリのファイルリストが出現する）

```
[:]%s/.*/mv & &.bak/
```

（上記の例の通り。全てのファイル名に『.bak』をつけた名前にmvするスクリプトが出現する）

```
[:][%][!]sh
```

（シェルに食わせる）

```
[:]q!
```

（編集放棄、終了）

などとしています。[!]コマンドも今回解説したので、どうなるか考えつつどうでもよいディレクトリでやってみてください。

ところでWindows95以来、管理者でもCLI（command-line interface）シェルよりGUIシェルを好む人が増えてきました。確かに日常の仕事ではGUIシェルのほうが手っ取り早い仕事もあるかもしれません（まあそれとて、bash・tcshのコンプリーション機能を併用すれば、どう考えてもGUIシェルのほうがトロいです）。しかし、上記の作業を行うディレクトリにファイルが1000個あったら、これはもはやGUIシェルでは『不可能な仕事』と言い切っても構わないかもしれません。ところがCLIシェルとviを組み合わせると、ここまで10秒ちょっとしかかかりません（当社比）。

古風なものが残っているのは、伝統や懐古趣味などとは違う、ということがおわかりいただけたでしょうか。ただし、こういうパワーの強い『兵器』を使うときはくれぐれも慎重にしてくださいね。

●グローバルコマンド

グローバルコマンド『g』では行単位の操作ができる、ということだけを覚えておいてください。ここでは改行のみの空行を削除する例を示します。

```
%g/^$/d……空行の削除
```

●次はカスタマイズに挑戦

次回はviのカスタマイズを中心に説明していく予定です。いよいよviの（そしてはっかー道）の深みにはまりつつある今日この頃、当コーナーだけでは説明不足かとも思いますが、ご意見・ご要望などがありましたらtaroh@taroh.orgまでお寄せください。では、happy viing!!

詰めviの正解　問題1　[:]%s/\(.＊\)␣ha␣\(.＊\)/\2␣ha␣\1/<Enter>　**問題2**　[:][%][!]sort<Enter>

賢く使うUNIX

これであなたもスマートなUNIX使い！

――指定した時刻にコマンドを実行する――

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、指定した時刻や一定時間後にコマンドを実行するための仕組みについて取り上げ、特に指定時刻に決められたコマンドを実行するcronシステムを中心に紹介する。

Illustration : Manami Kato

文：大池 浩一
Text : Kouichi Ooike

今月のお題

毎朝8時前にモデムやダイヤルアップルータの回線を切断する

パソコンを使っていると、いますぐにではなく、指定した時刻になったり、一定時間経過した後で処理を行いたい場合がある。たとえば、「今カップラーメンにお湯を入れたのできっかり3分後にメッセージを表示」とか、「10分後の会議に出るので忘れないように音声で知らせる」とか、「深夜のテレホーダイ時間に指定したファイルをダウンロードする」といった具合だ。さらに、「週に一度不要なファイルを消去する」、「毎日深夜に高速ファイル名検索のためのデータベースを更新する」など、指定時刻に定期的に行ってほしい作業もある。

LinuxなどのUNIX系OSでは、sleepやat、crontabといったコマンドとそれをサポートするデーモンにより、指定時刻や一定時間後の処理を比較的簡単に行える。こうしたコマンドは、システム管理者(root)だけでなく、一般ユーザーでも利用可能だ。実行するコマンドにはコンソールベースのものが多く、シェルのスクリプト機能が強力なので、トータルで見ても使い勝手がいい。特に、crontabで設定を行うcronシステムは、「毎時0分」「毎時0分と30分」「毎日7時58分」「毎週金曜日の0時45分」「毎月1日の4時30分」といったように、繰り返し実行する時刻や日付を柔軟に設定できる優れモノ。Linuxユーザーとしては、ぜひとも使い方を身につけておきたい。

Sleep、at、crontabの使い方をマスターしよう

ここでは、処理の簡単なコマンドから順に、指定した時間だけ処理を停止するsleepから始め、続いて指定時刻になるまで（あるいは一定時間）待機した後で1回だけ処理を行うat、最後に指定時刻に繰り返し処理を行うcrontabの使い方について説明する。

待機後に実行するコマンドやスクリプトの指定方法は異なるものの、機能の一部は重複しているので、それぞれの特徴を生かして使い分けるとよい。たとえば、sleepの場合は、秒単位の経過時間を指定でき、端末画面に文字列を出力できる。このため、1分以内の経過時間の指定が必要な処理や、秒単位で正確な経過時間の指定が必要な処理、あるいは端末画面へ文字列を出力する簡単な処理などに使用するとよいだろう。

一方、「明日の5時」とか「今から3時間後」といったように、長い時間待機した後で1回限りの処理をする場合にはatが向いている。「毎時0分と30分」とか「午前9時から午後5時まで1時間おき」、「毎週金曜日の0時45分」のように繰り返し行う必要のある処理の場合はcrontabで決まりだ。

●一定時間処理を停止するsleep

最初に紹介するsleepは、指定した時間だけ処理を停止するという単純なコマンドだ。「sleep 時間」という書式で使用し、時間には数値と単位を表す文字（s、m、h、d）の組み合わせを指定する。

たとえば、10秒間だけ処理を停止するには、数値（10）と秒単位を表す文字（s）を組み合わせた「10s」をsleepの引数に指定して、

```
$ sleep 10s
```

とすればいい。実行後、10秒たってからシェルのプロンプト（ここでは「$ 」）が表示される。分単位（m）や時間単位（h）、日単位（d）の場合も同様だ。なお、単位の文字を省略すると秒単位とみなされるので、

```
$ sleep 10
```

としても上の例と同じ結果が得られる。

これを、コマンドを順次実行するシェルの機能と組み合わせると、一定時間後にコマンドを実行する仕組みを実現できる。たとえば、10秒後に「時間です」という文字列を表示させるには、

```
$ sleep 10; echo "時間です"
```

とすればいい。左から順にコマンドを実行するシェルのコマンドセパレータ「;」により、まず「sleep 10」が実行されて10秒間処理が停止し、その後bashの組み込みコマンドechoにより「時間です」という文字列が標準出力（ここでは端末画面）に出力される。さらにほかのコマンドを続けて実行することも可能だ。

もっとも、このままではsleepとechoがフォアグラウンドジョブとして実行されるので、すべての処理が終わるまで次のコマンドを入力できない。10秒程度ならともかく、たとえばカップラーメンにお湯を入れてから3分間、画面を見つめて過ごすのは苦痛だし、わざわざパソコンを使う意味がない。

sleepにより処理が停止している間も、別のコマンドを実行できるようにするには、一連の処理をバックグラウンドジョブとして実行する必要がある。具体的には、コマンドライン全体をカッコ「(」「)」で囲み、末尾に「&」を付けて、

```
$ ( sleep 3m; echo "時間です" )&
```

とする。すぐにシェルのプロンプトが表示され、次のコマンドを入力可能だ。一方、カッコ内のsleepとechoは別のプロセス（サブシェル）で実行され、3分たつと端末画面に「時間です」と表示される。

なお、コマンドを囲むカッコは、sleepとechoをまとめてバックグラウンドで実行するためのもの。もし、カッコなしで「&」だけ付けると、echoだけがバックグラウンドで動作することになり、sleepはフォアグラウンドで実行されるため、3分間コマンドの入力を受け付けなくなってしまうのだ（Ctrl-Cキーで中断できる）。

この例では、コマンドの順次実行やバックグラウンドジョブといったシェルの機能をsleepとともに利用して、一定時間後の処理を実現している。これに対し、後で説明するatやcrontabの場合は、それぞれatdとcrondという「デーモン」（コラム参照）が用意されており、バックグラウンドで動作するデーモンが指定時刻にコマンドを実行する処理を行ってくれる。

また、複数のジョブの登録、待機中のジョブの一覧表示、不要なジョブの削除など、sleepでは処理が煩雑になってしまう部分についても、専用のコマンドやオプションが用意されており、本格的な運用が可能だ。

●一度きりの処理に向いているat

atは、指定時刻にジョブを1回だけ実行するコマンドで、「at 時刻」という書式で使用する。実行するジョブのコマンドは、標準入力（通常はキーボード）から入力するか、フ

Column

『悪魔』ではないデーモン

UNIX系OSでは、システム起動時に起動され、バックグラウンドで動作するサーバプロセスのことを「デーモン」（daemon）と呼ぶ。これはキリスト教的な「悪魔」（demon）のイメージではなく、ギリシャ神話における「ダイモン」（daimon：神と人の間に位する超自然的存在）、あるいは（人・土地などについている）「守護神」という意味で使われている。今回の記事で取り上げているatdやcrondのほか、inetd、smbd、lpd、syslogdなど数多くのデーモンが人知れずマシンの中で動作している。ためしに、コマンドラインで「ps ax」として表示されるプロセス一覧を眺めてみよう。端末から起動した覚えがなく、コマンド名の末尾がdで終わっているコマンドがあれば、それはほとんどの場合デーモンだ。

ァイルから読み込む。類似したコマンドにbatchがあり、指定時刻のシステム負荷が低い場合に限ってジョブを実行するという点だけが異なる。

それでは、午後4時に「会議の時間です」と書かれたウィンドウを表示してみよう。まずは、指定したメッセージをウィンドウに表示するTcl/Tkスクリプト「mymesg」（リスト1）をホームディレクトリに作成し、

```
$ chmod +x mymesg
```

として実行属性を与える。ためしに、コマンドラインで、

```
$ ~/mymesg 会議の時間です
```

とすると、「会議の時間です」というメッセージと[終了]ボタンで構成されるウィンドウが、ベルの音とともに表示されるはずだ(**画面1**)。

時刻の指定は、24時間制の「16:00」「1600」「16」、12時間制の「4:00pm」「0400pm」「4pm」、さらには午後4時がお茶の時間であることに由来する「teatime」など、さまざまな表現を利用できる。ここでは短くてわかりやすい「4pm」を使うことにしよう。

```
$ at 4pm
```

と入力してEnterキーを押すと、次の行に「at>」というプロンプトが表示される。

このプロンプトでは、ジョブで実行したいコマンドを1行に1つずつ入力する。複数のコマンドを入力可能だ。この例では「~/mymesg 会議の時間です」と入力してEnterキーを押せばいい。次の行の先頭でCtrl-Dキーを押すと入力終了となり、端末画面には、

```
$ at 4pm
at> ~/mymesg 会議の時間です
at> <EOT>
```

と表示される。

「at>」プロンプトではbashの編集・履歴機能が使えないので、同じコマンドラインを何度も指定する場合は、ファイルから読み込むようにするといい。コマンドを記述したファイルを用意し、-fオプションに続けてファイル名を指定する（実行属性をつける必要はない）。

上の例では、ホームディレクトリのkaigiというファイルに「~/mymesg 会議の時間です」と記述して、

```
$ at -f ~/kaigi 4pm
```

とする（「at>」プロンプトは表示されない）。別の会議があるなら、時刻だけ変えて再度atを実行しよう。

こうして設定したジョブは、at専用のキューに保存され、デーモンであるatdによって1分ごとに設定時刻と現在時刻の比較が行われる。そして、設定時刻が現在時刻と等しいジョブがあれば、設定したコマンドが実行される。たとえば、上の例なら午後4時ちょうどになるとウィンドウが開いて「会議の時間です」と表示されるわけだ。なお、atを実行した時点で午後4時をまわっていた場合は、翌日の午後4時ちょうどにウィンドウが開く。

▼リスト1 ウィンドウを開いてメッセージを表示する「mymesg」

```
1 #!/bin/sh
2 # 下の1行はシェルが実行 \
3 exec wish "$0" "$@" -display :0.0
4 # これ以降はTcl/Tkが実行
5 label .l1 -text "$argv"
6 button .b1 -text "終了" -command exit
7 pack .l1 .b1 -fill x
8 bell
```

画面1　時間になるとこのようなウィンドウが開く

●atのキューの制御を行う

このほか、atに関連したコマンドやオプションがいくつか用意されているので、重要なものについて説明しよう。まず、atqというコマンドでは、現時点でatのキューに登録されているジョブの一覧を確認できる。

```
$ atq
11        1999-11-20 16:00 a
12        2000-12-24 18:30 a
14        1999-11-20 23:00 b
```

atqの出力は「ジョブ番号 動作時刻 名称」という書式だ。ジョブ番号は、ジョブの内容表示やキューから削除する場合に使用する。また、名称は1文字の英字（大文字または小文字）で、通常はatで設定したジョブは「a」、batchで設定したジョブ合は「b」になる。

それぞれのジョブのコマンドを知りたい場合は、atの-cオプションに続けてジョブ番号を指定する。

```
$ atq -c 11
#!/bin/sh
(略)
~/mymesg 会議の時間です
```

実行時の環境変数などの設定部分が含まれるため、出力内容は入力したものよりずっと長くなる。実際のコマンドは末尾に書かれている。

もし、コマンドの打ち間違いなどに気づいたり、ジョブの実行自体が不要になった場合には、atrmというコマンドを使ってそのジョブをキューから削除する。以下のように、削除するジョブ番号(複数可)を指定すればいい。

```
$ atrm 11 14
```

ところで、atにおける時刻の指定方法はとても柔軟で、日付も含めて指定したり、「今から10分後」といった相対的な指定も可能だ。以下に例を示そう。

00:00 01/01/00	2000年1月1日の午前0時
8pm Dec 24	次の12月24日午後8時
10:30 today	今日の午前10時30分
4am + 3 days	3日後の午前4時
now + 1 hour	今から1時間後

最後の例の「now + 時間」という書式を利用すると、atで定期的な繰り返し処理を行える。具体的には、atで読み込むファイルの最後の行に、atでそのファイルを再度読み込むように記述すればいい。もっとも、このような一定時間ごとに繰り返しされる処理に対しては、次に説明するcrontabを使ったcronシステムのほうが向いている。atは、コマンドを1回だけ実行する場合に限って使ったほうがよいだろう。

図1　atは一度きりの処理に向いている

●crontabを利用したcronシステム

一度実行したらそれでおしまいではなく、何度も繰り返して実行する必要のある処理については、crontabというコマンドを利用したcronシステムが最適だ。最初から繰り返し処理用にデザインされているため、ジョブを実行する日時を柔軟に設定できる。たとえば、「毎時0分」「毎時0分と30分」「毎時0分から30分まで5分おき」「毎日7時58分」「午前9時から午後17時まで1時間おき」「毎月1日の4時30分」「毎年3月6日の18時36分」「毎週金曜日の0時45分」といった具合だ。

cronシステムでは、デーモンであるcrondがジョブのリストを監視しており、指定された時刻になるとそのジョブを自動的に実行する。atと大きく異なるのは、ジョブのリストの登録をファイル（crontab）に対して行うという点だ。なお、ファイル名とコマンド名が同じで紛らわしいので、以下では「crontabファイル」「crontabコマンド」と記述することにする。

さて、UNIXやUNIX互換OSで使われているcronシステムの実装は次の2種類の方式に大別できる。

(1) 集中管理方式

ひとつのcrontabファイル（/etc/crontabや/usr/lib/crontab）で全ユーザーのジョブを管理する。このため、ジョブの書式には実行ユーザー名のフィールドが含まれる。ジョブの登録や削除の際には、crontabファイルを直接書き換える。ただし、書き込み権限を持つのはrootだけなので、一般ユーザーはrootになって設定するか、rootに設定を依頼する必要がある。

(2) 個別管理方式

ユーザーごとに個別のcrontabファイル（/var/spool/cron/ユーザー名）が用意され、個々のユーザーごとに管理される。このため、ジョブの書式には実行ユーザー名のフィールドは含まれない。ジョブの登録や一覧表示、削除などの操作にはcrontabコマンドを使用する。一般ユーザーも自由にジョブを登録可能だ。

それでは、Linuxは（1）と（2）のどちらの方式を採用しているのだろうか。正解は「どちらも」だ。Linuxのディストリビューションで使われている「Vixie cron」では、集中管理と個別管理を同時に行っている。実際には、それぞれの場面で便利なほうを使えばいい。一般ユーザーにとっては、crontabコマンドでジョブを登録できる（2）の個別管理方式を利用しよう。一方、rootが行うべきシステム管理的な作業については、（1）の集中管理方式で/etc/crontabに書いたほうが見通しがいい。以下では、使用する機会が多いと思われる（2）について詳しく説明し、（1）については軽く触れるにとどめる。

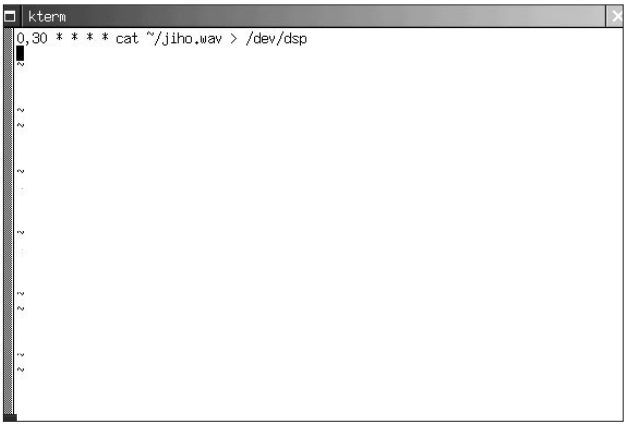
ユーザーごとのcrontabファイルにジョブを登録・削除するには、crontabコマンドを-eオプション付きで起動する。すると、テキストエディタ（初期設定はvi）が起動して登録済みのジョブの一覧が表示される（**画面2**）。あとは、新たなジョブを追加するなり、必要なくなったジョブを削除するなりすればいい（ジョブの書式については後述）。変更後のcrontabファイルを保存すると、crondがそれを検知して新たな設定が有効になる。

なお、viの操作に慣れていない人は、本誌記事「viはじめました」を読んで精進するのもいいが、手っ取り早い解決法は環境変数EDITOR（またはVISUAL）に使い慣れたエディタの実行ファイル名を設定することだ。たとえばmuleを使いたいなら、

```
$ export EDITOR=mule
```

とすればいい。常にmuleを使いたいならば、.bash_profileにも上の内容を追加しよう。

このほか、crontabコマンドの-lオプションでcrontabファイルの内容が標準出力（通常は端末画面）に表示され、-rオプションでcrontabファイルを削除できる。また、-uオプションに続けて自分以外のユーザー名を指定すると、そのユーザーのcrontabファイルを編集・表示・削除できるようになる（rootのみ可）。



画面2　Viでcrontabファイルを編集する

●crontabファイルの構造

crontabファイルには、1行にひとつずつジョブが登録されており、各行は6つのフィールドが空白区切りで並んでいる。最初の5つは日時の設定（分・時・日・月・曜日）で、最後の1つが実行するコマンドだ。複数のコマンドを実行したいときは、それらをシェルスクリプトにまとめておき、crontabファイルではスクリプト名を指定する。

日時の各フィールドには、具体的な数値や名称を設定するほか、「任意」を意味するワイルドカード「＊」も使用できる。たとえば、「毎時0分」は、分のフィールドを0、それ以外はワイルドカードとして「0＊＊＊＊」と書ける。同様に、「毎日7時58分」は「58 7＊＊＊」、「毎月1日の4時30分」は「30 4 1＊＊」、「毎週金曜日の0時45分」は「45 0＊＊Fri」といった具合だ。曜日を数値で指定することもできるが、英語の名称をそのまま使ったほうがわかりやすいだろう。

これらのフィールドには、複数の値を「,」区切って並べられる。たとえば、「毎時0分と30分」なら「0,30＊＊＊＊」と書けばいい。連続した整数値は「-」で範囲指定できるので、「9時から17時まで1時間おき」は「0 9-17＊＊＊」となる。一定間隔の値の場合は、範囲指定や「*」の後に「/増分値」を指定する。「毎時0分から5分おき」なら「0-59/5＊＊＊＊」と書けばいい。

それでは、「毎時0分と30分に時報（~/jiho.wav）を鳴らす」という処理を考えてみよう。WAVEファイルを再生するには、デバイス/dev/dspに対してファイルの内容をcatで出力する。よって、crontabファイルには、

```
0,30 * * * *     cat ~/jiho.wav > /dev/dsp
```

という1行を追加すればいい。なお、GNOMEのサウンドデーモン「EsounD」(esd)が動作中なら、EsounD用プレーヤを利用して「esdplay ~/kaigi.wav」とする。

crontabファイルには、こうしたジョブ情報だけでなく、コマンド検索パス（PATH＝…）やメールの送り先（MAILTO＝…）などの設定を記述することも可能だ。デフォルトでは、PATHには「/usr/bin:/bin」、MAILTOには本人のユーザー名が設定されている。コマンドが標準出力や標準エラー出力をファイルにリダイレクトせずに書き込んだ場合、その内容がメールとなって送られてくる。

最後に、集中管理方式のcrontabファイル（/etc/crontab）について簡単に説明しよう。集中crontabの書式は、コマンドの前に実行ユーザー名を指定するフィールドが加わって全7フィールドになる。たとえば、毎時0分にnobodyユーザーが/usr/local/lib/sounds/jiho.wavを再生するには、rootになって/etc/crontabをエディタで編集し、

```
0 * * * * nobody cat/usr/local/lib/sounds/jiho.wav
> /dev/dsp
```

を追加する。なお、Vixie cronでは、/etc/crontabをcrondが1分ごとに監視しており、更新があると即座に反映されるので、crondを再起動する必要はない。

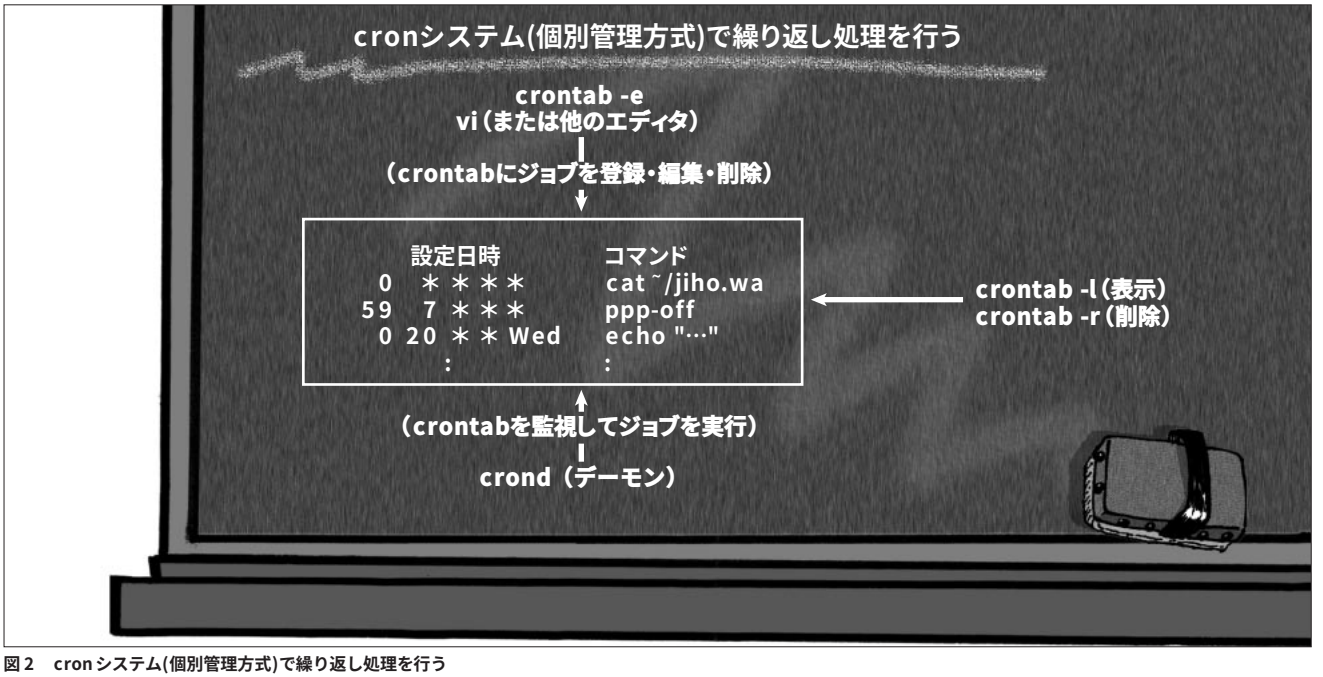


図2　cronシステム(個別管理方式)で繰り返し処理を行う

今月のお題

☑ 毎朝8時前にモデムやダイヤルアップルータの回線を切断する

後半は毎回ひとつのテーマに絞り、コマンドを組み合わせて実現する方法を説明する。今回のお題は、**モデムによるPPP接続やダイヤルアップルータによる接続環境で、テレホーダイの終了時（午前8時）の直前に回線を自動切断する**というもの。どちらの場合もcronシステムを利用して毎日8時前にジョブを起動し、PPP接続の場合は切断スクリプトを実行、LAN接続の場合はダイヤルアップルータへのtelnet接続と回線切断用コマンドを送信する。なお、ダイヤルアップルータ関連についてはvine-users ML #02460以降のスレッドを参考にさせていただいた。

●モデムの場合

以下では、PPP接続のための各種設定はすでに行われており、一般ユーザーがダイヤルやPPPサーバへの接続、切断といった処理を行える状態になっているという前提で話を進める。こうした状況では、一般ユーザーでも接続や切断のためのシェルスクリプトを使えるはずだ。

たとえば、Linuxのディストリビューションで広く採用されているpppdでは、接続用のppp-on、切断用のppp-offというシェルスクリプトが用意されており、/usr/sbinに置かれている場合が多い。cronシステムを利用して回線切断を行う場合にも、このppp-offを利用すればいい。crontab -eでエディタを起動し、以下の1行を追加する。

```
58 7 * * *   /usr/sbin/ppp-off
```

時刻を8時少し前（7時58分）に設定するのは、万が一にも課金されないための安全策だ。また、一般ユーザーの場合、ppp-offが置かれた/usr/sbinは環境変数PATHに含まれていないはずなので、crontabファイルではフルパスでコマンドを指定する。

エディタを終了すると自動的にcrondが再起動され、以後毎日7時58分になると自動的に回線が切断されるようになる。なお、ppp-offは、現在回線が接続中かどうかを自分で判断し、接続中の場合だけ切断処理を行うため、接続していない状態で実行しても問題ない。

一方、国産のLinuxディストリビューション（Vineなど）に採用されているPPxPを利用している場合は、ppxpというコマンドを使って回線の接続や切断処理を行う。通常はppxpのコンソール画面でコマンド（たとえば切断は「disconnect」）を入力するが、オプション-Cに続けてコマンドを記述すると、コンソール画面を経由せずに直接コマンドを実行できる。

cronシステムを利用して回線切断を行う場合にも、この-Cオプションを利用すればいい。crontab -eでエディタを起動し、以下の1行を追加する。

```
58 7 * * *   /usr/bin/ppxp CONFIGFILE -C disconnect
```

時刻の設定についてはpppdの場合と同じだ。「CONFIGFILE」には、PPxPのクイックダイヤルアップ機能などを利用して作成した設定ファイルのファイル名を指定する。

●ダイヤルアップルータの場合

LANでマシンと接続されたダイヤルアップルータを利用してインターネットにアクセスする場合は、モデムの場合よりも面倒だ。というのも、たいていの場合、ダイヤルアップルータが提供する管理用のWebページをブラウザでアクセスし、フォームに設定を入力したり、ボタンを押したりすることで設定変更や回線の手動接続・切断を行うからだ。GUIを使ったこの手の処理が自動化しにくいことは簡単に理解できるだろう。

もっとも、手段がないわけではない。ダイヤルアップルータの機種の多くは、telnet接続してコマンドを実行することによっても設定を変更できるからだ。強力なマクロ言語を備えたtelnetクライアント（たとえばWindows用の「TeraTerm Pro」など）を使えば、

- **ダイヤルアップルータにtelnet接続**
- **「login:」の表示を待ってユーザー名を送信**
- **「Password:」の表示を待ってパスワードを送信**
- **プロンプトの表示を待って回線切断用コマンドを送信**
- **プロンプトの表示を待ってログアウトコマンドを送信**

という一連の処理をマクロを書くことで自動化できる（実際にそのようなマクロも作られている）。

しかし、Linuxのディストリビューションで広く採用されているtelnetクライアントには、残念ながらこのようなマクロ言語は装備されていない。それでは、ダイヤルアップルータの場合は、cronを使った回線の自動切断をあきらめるしかないのだろうか。

●Expectでtelnet接続を自動化する

実は、telnetやftpなどのような対話的なソフトの動作を自動化するための「Expect」(http://expect.nist.gov/)というスクリプト言語がある。スクリプト言語Tclの機能を拡張したもので、対話的ソフトの入出力を横取りして、指定した文字列が送られてくるのを待ち、キーボードから入力しているかのように文字列を出力できる。ftpクライアントなども含め、たいていのLinuxのディストリビューションでは、最初からインストールされているはずだ。「which expect」で確認してみよう。

そこで、代表的なダイヤルアップルータのひとつである「MN128-SOHOシリーズ」にtelnet接続し、回線切断を行うExpectスクリプト「soho-off」（**リスト2**）を作成してみた。ホームディレクトリにコピーし、chmodで実行可能属性をつけておこう。

実行する前に、環境に合わせて修正する点がいくつかある。まず、ダイヤルアップルータのローカルIPアドレスが「192.168.0.1」（デフォルト値）ではない場合は、6行目の該当部分を修正する。また、ダイヤルアップルータのパスワードを設定していないなら、10～11行目の先頭に「#」を挿入してコメントにしておく。逆に、パスワードを設定している場合は11行目の「XXXXXXXX」を実際のパスワードに書き換え、所有者以外が読み書き・実行できないように「chmod 700 soho-off」として、パスワードがばれないようにしておくとよいだろう。

こうした修正が終わったら、コマンドラインで、

```
$ ~/soho-off
```

としてみよう。telnet接続、ログイン、回線切断コマンドの送信と処理が進んで、最後にプロンプトに戻るはずだ（メッセージの文字化けはシフトJISで出力されるためなので気にしなくていい）。

正常に動作するなら、テレホーダイ終了時に自動切断されるようにcrontabファイルに記述する。crontab -eでエディタを起動し、

```
58 7 * * *   ~/soho-off > /dev/null
```

という1行を追加する。これで、毎日7時58分になると、ダイヤルアップルータの回線を自動的に切断するようになる。なお、RedHat 5.2やそれに基づいたディストリビューション（Vine 1.1など）では、cronシステムからExpectを利用する際に、コマンドの出力をリダイレクトしないと正常に動かないので注意が必要だ。

ところで、MN128-SOHOシリーズでは、telnet接続で使われるユーザー名は「user」、接続後に表示されるプロンプトは「MN128-SOHO% 」、回線切断用コマンドは「disconnect all」となっている。これらの文字列は、ダイヤルアップルータの機種やメーカーごとに異なっているため、このスクリプトはMN128-SOHOシリーズ以外では動作しない。その他の機種では、まず手動でダイヤルアップルータにtelnet接続し、実際に使われているユーザー名やプロンプト、回線切断用コマンドを確認したうえで、soho-offの該当部分を書き換える必要がある。

誌面の都合により詳細な内容の説明はできないが、最後に簡単に触れることにしよう。1～3行目は、同一ファイルにシェルスクリプトとExpectスクリプトを混在させるための仕掛けだ。telnetの起動とダイヤルアップルータへの接続は6行目のspawnコマンドで行われる。接続後の処理は8行目以降で、文字列を待つexpectコマンドと、文字列を送るsendコマンドを交互に使って、ユーザー名や回線切断用コマンドを送出している

▼リスト2 MN128-SOHOにtelnetで接続して回線を切断する「soho-off」

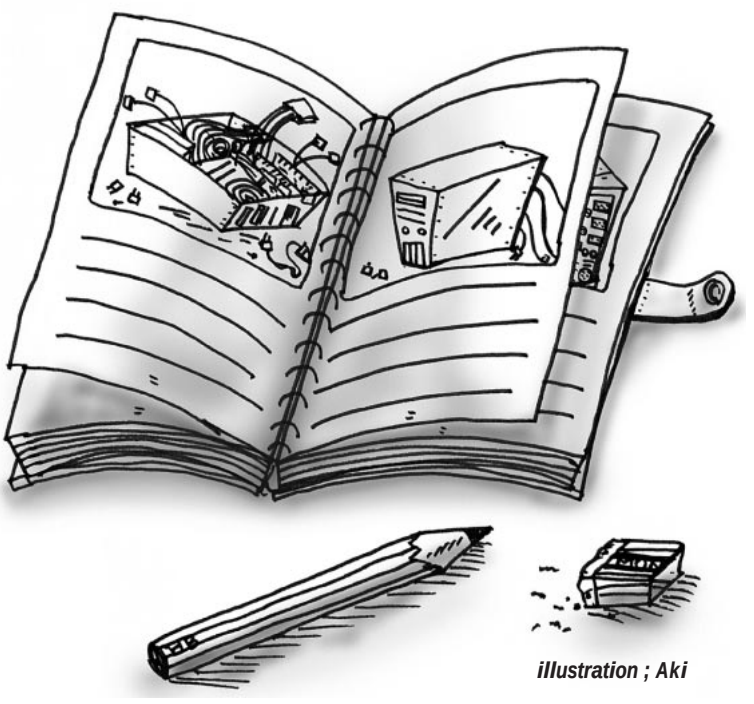
```
 1 #!/bin/sh
 2 # 下の1行はシェルが実行 \
 3 exec expect -f "$0" "$@"
 4 # これ以降はExpectが実行
 5 set timeout -1
 6 spawn telnet 192.168.0.1
 7 match_max 100000
 8 expect -ex "login: "
 9 send "user\r"
10 expect -ex "assword:"
11 send -- "XXXXXXXX\r" # 実際のパスワードを記述
12 expect -ex "MN128-SOHO% "
13 send "disconnect all\r"
14 expect -ex "MN128-SOHO% "
15 send "exit\r"
16 expect eof
```

※実際には行番号は付かない。

Linux日記

第3回　仮想記憶にかかわるコマンド

仮想記憶の仕組みに引き続き、今月はLinuxで仮想記憶を実現するデーモン、メモリの状況などを表示するコマンドを解説します。あなたのLinuxマシンではメモリは足りているでしょうか？



文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

前回は、仮想記憶システムの基本的な構造を解説した。今回は、仮想記憶に関連するLinuxのコマンド、仮想記憶のためのデーモンなどについて見ていこう（連載3回目で、やっとLinuxにたどり着いた）。だが、その前に前回書ききれなかったことについて説明しておこう。物理メモリ量とスワップパーティションのバランスの話の続きだ。

■スラッシング

仮想記憶といえども、無限のメモリが使えるわけではない。まず思いつく限界は、スワップ領域を使い果たすということである。どんどんページアウトを行い、スワップ領域を使い切ってしまったら、それ以上のメモリを割り当てることはできない。プロセスはメモリ割り当てに失敗し、エラー終了してしまう。この場合、とりあえずの対処として、スワップ領域を拡張すればいい。そうすれば、さらにメモリ空間を使うことができるようになる。

しかし実際には、（極端にスワップ領域が少ないといった場合を除いて）スワップ領域を使い切る前に、事実上の限界に至ることが多い。通常、オペレーティングシステム上では、いくつかのプロセスが時分割で動作している。ユーザーが動かしているプログラム、X Window Systemなどの環境、各種デーモンなどである。そして、これらをすべてまかなうだけの実メモリ容量がない場合は、使用頻度に応じてページアウトが発生する。実行コンテキストが切り換わり、今までスリープしていたプロセスが再開したときに、もし必要なメモリがページアウトされていれば、それをページインして処理を続行することになる。このページインにともない、どこかのメモリがページアウトされるだろう。このとき、ページアウトされるのが本当に使用頻度の低いメモリならよいのだが、実メモリがあっぷあっぷの状態だと、そこそこの使用頻度のメモリがページアウトされてしまうことがある。

たとえばアプリケーションからデーモンの実行に切り換わり、デーモン側でページインが行われるとしよう。実メモリ容量が切迫していると、ついさっきまで動いていたアプリケーションのメモリがページアウトされてしまう可能性があるのだ。やがて実行はデーモンからアプリケーションに再び切り換わる。すると同じことが起こる。アプリケーション側がページインするために、さっきまで動いていたデーモンのメモリがページアウトされてしまうのだ。これがさらにひどくなることもある。大量にメモリ領域を消費するアプリケーションやサーバなどを少ない実メモリで動かすと、自分自身の動作のために、自身のメモリの一部をページアウトするはめになる。

仮想記憶の初期のうたい文句に、実メモリ以上の大きさのプログラムやデータを処理できるというものがあった。メモリが非常に高価だった時代には、このようにプログラムの一部をページングしながらでも動作できるというのは、大きな魅力であった。だが、ペー

ジングにはディスクI/Oがともなうので、頻繁に行われるとプログラムの実行速度が一気に低下してしまう。これがさらにひどくなると、ページングのためのディスクアクセスがほとんど止まらなくなり、システムはプログラムの実行よりもページング処理に多くの時間を費やすようになる。このような状態をスラッシングという。この状態に陥ると、ユーザーに対する応答時間も長くなり（ユーザープロセスへの切り換えにより、またスラッシングがひどくなるからだ）、ロードアベレージも高まる（当然、実行待ちのジョブが増える）。しかもシステムは、忙しく動いているにもかかわらず、有意義な処理はほとんどしていないのである。

このような状態に陥ったら、選択肢は2つしかない。使用メモリ量を減らすか、実メモリを増やすかである。注意してほしいのは、この状態でスワップ領域を増やしても、事態は何も解決しないということだ。不足しているのは実メモリで、スワップ領域ではない。悪いことは言わない。今はメモリが安いのだから、メモリを増設しよう。

■ページングとセグメンテーション

仮想記憶の実現には、ページングとセグメンテーションという2種類の方法がある。前回から解説している仕組みはすべてページングである。

ページングが数Kバイトの固定長ページを単位として外部記憶とやり取りするのに対して、セグメンテーションは、可変長のブロック（セグメント）を単位として外部記憶とのやり取りを行う。たとえばプロセスのコード、データ、さらにはそれらの特定のブロックというようにモジュールごとにセグメントを割り当て、そのセグメントを単位として、ディスクとのやり取りや破棄の処理を行う。メモリの割り当てや新しいプログラムの実行にともなって、実メモリが不足したときに、すでにメモリ中にある別のプロセスの（あるいは自分自身の）セグメントをディスクに書き出したり破棄したりして、メモリを解放するのである**（図1）**。

ページングとセグメンテーションを比較した場合、一般にページングのほうが、データやコードの使用頻度をより細かく管理できる。

ページングをサポートしているシステムは、セグメンテーションもサポートできるので、古いバージョンのシステムとの互換性のために、セグメンテーションをサポートしているページングベースのシステムもある。

ページングを実現するためには、ページ単位でアドレスのマッピングとページフォルトを検出できるMMUと、命令再開をサポートしたCPUが必要である。セグメンテーションの場合も同じ機能が必要になる。実行時に、実メモリに存在しないセグメントのアクセスを検出しする必要があるからだ。

MMUなど、高度なハードウェアサポートを必要としない簡易的なシステムもある。これはセグメンテーションをさらに単純化したようなものだ。プロセス全体、あるいはプロセスのコード、データ程度の単位でセグメンテーションを行い、実行時のアクセス違反検出を行わない。その代わりに、そのプロセスの実行コンテキストで必要とされるセグメントが、必ず実メモリ中にあることを保証するのである。コンテキスト切り換えが発生し、スワップアウトされているプロセスを実行するときには、そのプロセスを再開する前にすべてのセグメントをスワップインする。プロセスが物理アドレスしか使わないのであれば、MMUなどは不要になる。ただし、スワップインされたときに前と同じアドレスにロードされる保証はないので、実行時の動的なリロケートをサポートしていなければならない。x86アーキテクチャのセグメントベースのアドレスは、このような用途に向いたものだ。

■Linuxの仮想記憶

さて、Linuxの仮想記憶について見てみよう。ps axを実行して表示されるプロセス一覧のはじめのほうに、kswapdというプロセスが表示される。これがページングを行うためのデーモンである。UNIXの場合、デーモンとユーザープロセスは、ユーザーから見れば、制御端末の有無、システムの初期化スクリプトから起動される、rootなどの特殊な権限で動作しているといった違いがあるが、カーネルから見る

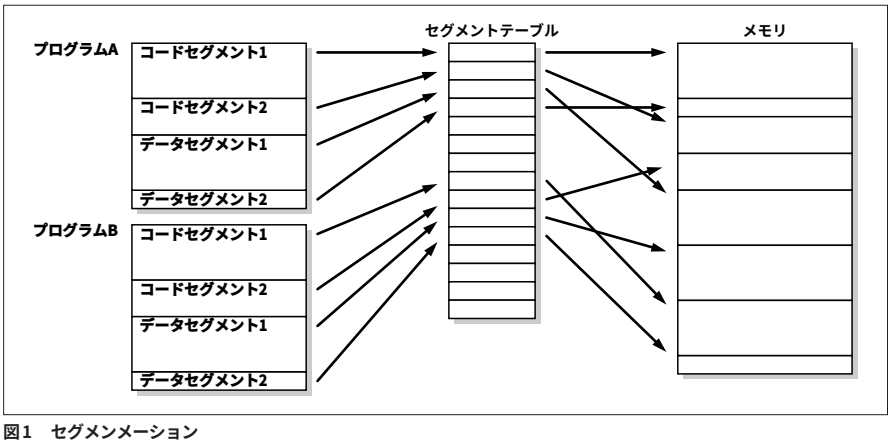


図1　セグメンメーション

限り、特に違いはない。デーモンは、たまたま制御端末がなく、rootなどの特権権限で動いているユーザープロセスに過ぎない。これに対して、ページングのデーモンは、完全にオペレーティングシステム中の機能であり、ユーザープロセスではない。システムのプロセステーブルに登録されているのでpsで表示されるが、ほかのユーザープロセスとはまったく別種のものである。また、このプロセスのための実行ファイルも存在しない。コードはカーネル中に含まれている。

ページング用のデーモンが特殊な扱いを必要とするのは、いくつかの理由がある。まず、このデーモンは絶対にページアウトされてはならない。これがページアウトされたら、自身のページインも含めて、以後のページングが行えなくなってしまうからだ。また、ほかのプロセスよりも高いプライオリティが割り当てられている。もしプライオリティが低いと、ページング要求の際に、よりプライオリティの高い別のプロセスが実行を再開し、いつまでもページング処理が始まらなくなってしまう（実際にはいつかは始まるのだが、かなり後になってしまう。その頃には、さらにページングのリクエストがたまっていることだろう）。また、ページング処理に際して、MMUなど、多くのカーネルが管理するリソースにアクセスすることになるが、これらの操作はシステムコールを介して公開されているものではないし、その性質上、公開すべきものでもない。そのため、ページング処理は、カーネルの独自のコンテキスト（カーネルスレッド）で実行するのが好ましいのである。

psでこの種のデーモンを見ると、コマンド名がカッコで括られている。これはスワップアウトされていることを意味するが、実際にスワップアウトされているわけではなく、単に常駐メモリ量（RSS）が0であるため、スワップアウトと見なされているのだ。要するにこれらのプロセスは、プロセステーブルに登録されているのでプロセスとして表示されるが、通常のユーザープロセスが使うような方法ではメモリリソースを使っていないのだ。

■仮想記憶に関連するコマンド

仮想記憶によるページアウト、ページイン処理は、すべて自動的に行われ、メッセージなどが表示されるわけでもないので、それが行われているという気配はほとんどわからない。逆にわかるようであれば、つまり、実行プロセスの切り換えのたびにディスクアクセスがあり、レスポンスが悪いといった状況は、スラッシングの兆しがあるということだ。

UNIXには、仮想記憶の状態を表示するコマンドがいくつか用意されているので、それを使ってシステムの状態をある程度知ることができる。「メモリが不足しているような気がするのだが、実際はどんな状況なのか」といったことを具体的に調べられるのだ。

ここで、この種のチェックに使える代表的なコマンドを紹介しよう。freeとvmstatとpsである。

■freeコマンド

freeコマンドは、現在の実メモリとスワップ領域の使用状況を報告する**（画面1）**。これを見れば、物理メモリが切迫しているかどうか、どの程度ページアウトされているかといったことがわかる。各フィールドの意味はほとんど明らかだろう。totalは合計、usedは現在使用されている容量、freeは未使用容量、sharedは共有メモリ量、buffersはI/Oバッファとして使用されている容量、cachedはディスクキャッ

Column

スワップ

仮想記憶では、スワップ領域、スワップイン、スワップアウトなど、「スワップ」という言葉がよく使われる。厳密な意味では、スワップアウト、スワップインというのは、あるプロセスがすべてディスクに書き出されている、あるいはそれがすべてメモリに読み込まれることを指す。つまり、ページングによるプロセスの部分的なディスクへの書き出しは、スワップアウトではない。ページアウトである。そして一般的にページングを使ったシステムでは、プロセスすべてをスワップ領域に書き出すことはなく、管理情報など、一部は必ず実メモリ上に残されている。もっとも、常駐部以外がすべてページアウトされてしまえば、実質的にスワップアウトされたことになる。

現在のページングベースのシステムの多くでは、「スワップアウト」が行われることはほとんどないのだが、用語としてのスワップは数多く残っている。これはcoreやttyなどと同じで、ほとんど歴史的経緯によるものだ。

●コラムの蛇足

coreというのは、半導体メモリが主流になる以前に使われていたメモリ素子である。小さなフェライトのリング（コア）を銅線で2次元状に編み、各コアの磁化の向きにより1と0を表していた。core dumpというのは、このコアの内容をダンプしたファイル、転じて、プロセスの実行イメージ（正確にはデータやスタックなど）のスナップショットである。

ttyはTeletype（商品名、転じてシリアル接続の印字式端末装置の通称名）の略語である。紙テープリーダ、紙テープパンチャ、電動タイプライタを組み合わせた、印字式のシリアル接続端末装置である。

シュとして使用されている容量（Kバイト）である。Swapのusedの数字が大きく、Memのfreeとcachedが小さければ、実メモリをほぼ使い切り、一部がページアウトされているということを示している。ディスクキャッシュ容量は、空きメモリ量に応じて自動的に増減するので、これによってメモリ容量が圧迫されるということはないが、ディスクキャッシュが減るとI/Oパフォーマンスが多少低下する。

freeコマンドは、おおざっぱな概要は教えてくれるが、気合いを入れて調べるには少々情報が足りないだろう。

■vmstatコマンド

vmstat（Virtual Memory Status）はその名のとおり、仮想記憶の状態を示す。psが個々のプロセスの情報を示すのに対して、vmstatはシステム全体に関する情報を示す。vmstatをオプションを指定せずに実行すると、**画面2**のような出力が得られる。

またオプションにより、指定時間間隔で表示を繰り返すことができる。たとえばvmstat 3と指定すると、3秒間隔で繰り返し表示する。繰り返し表示することにより、システムの状態を継続的に観察することができる。システムの動作状況は時々刻々変化するので、実際の状況を調べるには、さまざまな状況において、ある程度の時間観察を続ける必要がある。

表示について説明しよう。procsフィールドは、さまざまな状態にあるプロセスの数である。rは実行可能（Runnable）プロセスの数だ。実行可能とは、実行中であるか、あるいは実行のためにCPUが空くのを待っているという状態である。bは、何らかの理由でブロックされているプロセスの数である。通常は、ページングも含めて、ディスクI/O待ちで止まっているプロセスの数となる。wはスワップアウトされている実行可能プロセスの数だ。

rの数字が大きいということは、実行可能でありながら、システムが忙しいために待たされているプロセスが多いということだ。uptimeで表示されるロードアベレージも高くなり、システムのCPUパワーに対して負荷が重いということになる。CPUの忙しさは、cpuのフィールドで定量的に見ることができる。usはユーザーCPUタイム、syはシステムCPUタイム、idはアイドルタイムの割合である。idがほとんど0で、usとsyの合計が100パーセントに近ければ、CPUはほとんど休みなく動作していることになる。**画面2**の実行結果では、システムはほとんど何もしていないということがわかる。

一方、procsのbの数が多いのは、ディスクI/O処理待ちのプロセスが多く、ディスク負荷が大きいことを意味する。また、このような状態は、io、systemフィールドにも現れてくる。ioのbiとboは、それぞれ秒あたりのブロックデバイスへの書き込み、読み込み回数で、systemのinとcsは、それぞれ秒当たりの割り込みとコンテキスト切り換えの回数である。ディスクI/Oが多ければ、bi、bo、inの数字が大きくなる。

一般に、rやbの数字が大きいということは、アクティブなプロセス（何らかの処理を行っているプロセス）が多いということで、システムが忙しく働いているということを示している。rが多くてbが少ないのは、ディスクI/Oをともなわないで CPU集約的な処理を行うプロセスが多数ある場合だ。rが少なくてbが多いのは、I/O集約型の処理が多いということになる。

その他のフィールドについて説明しよう。memoryの内容はfreeと同じだ。swpdは使用されているスワップ領域の容量である。swapのsiとsoは、秒あたりのスワップ領域との転送量（Kバイト）だ。ページングが行われていなければ、これらの数字は0になる。使用頻度が低いデーモンなどが一部ページングされている程度なら、一時的にここの数字が増加し、また少なくなるだろう。このフィールドが、継続的なページング処理を示していれば、実メモリが相当切迫している。コンテキスト切り換えに伴い、頻繁にページングが行われているということになるからだ。

メモリが切迫し、スラッシング状態になると、vmstatの表示はどうなるだろうか。実際にはやっていないが、おおよその見当は付く。まず、スワップ領域アクセスの多発により、si、soの数値が大きくなるだろう。実メモリは切迫しているので、free、cacheの数字は小さくなる。そして、ページアウトが多くなっているので、swpdが大きく

```
             total       used       free     shared    buffers     cached
Mem:        160088     156264       3824      30908      11876     125664
-/+ buffers/cache:      18724     141364
Swap:        72256        224      72032
```

画面1　freeコマンドの出力

```
procs                  memory    swap    io     system      cpu
 r b w   swpd  free  buff  cache  si  so  bi  bo   in  cs  us sy id
 0 0 0   224  4444  3876 134284   0   0   4   2  101  13   0  0 99
```

画面2　vmstatの出力

なる。また、ページング待ちなどでブロックされているプロセスが増え、bの数字が上がる。かなりのCPUサイクルがページング処理に回されるので、rで示される実行待ちの実行可能プロセスの数も増えるだろう。ユーザープロセスの実行は、そのかなりの部分がユーザーモードで行われるので、（あくまでも一般論だが）通常はusとsyは同程度か、usが大きくなるが、スラッシング状態だとページング処理（これはシステムモードで行われる）が増えるため、syの割合が大幅に増えるはずだ。

ここで具体的な数字を挙げることはできないが、判断はさほど難しいものではない。ポイントは2つだ。健全な状態をあらかじめ知っておくこと、そして、ある程度時間をかけて観察することだ。システムの起動直後、まだデーモンもユーザーのアプリケーションも起動しただけで、サービスをほとんど始めていないという、システムがもっとも遊んでいる状況におけるvmstatの結果を見ておこう。CPUやメモリ、ディスクに対する負荷が最小のときに、どの程度になるかを調べておくのだ。また、システムが実際に稼働し、デーモンやユーザープログラムが適度に動いており、なおかつそれらが快適な速度で稼働している、つまり、過負荷でもなく、メモリも不足していないという状況で、vmstatの数字がどの程度になるかを知っておくことも大切だ。こういった正常時のリファレンスとなる数字（そして操作感）を知っていれば、異常事態の発生と、それがどの程度悲惨なのかといったことを判断できる。

もう1つのポイントは、異常と思われる事態が、定常的に発生しているのか、たまたま発生しただけなのかといったことの判断である。たとえば、重い処理をしているときに、たまたまデイリーで動く重いバッチが起動したとか、サーバ上で重い処理がたまたまぶつかったといったことなのか、普通に使っているだけなのに、1日に何回もサーバが異常に遅くなることがあるといったことなのか？ また、たまに起こる程度なら構わないのか、それとも困るのか？ こういったことも考慮にいれないと、本当に正しい判断は行えない。

■psコマンド

名前が無愛想に短いことからわかるように、昔からあるコマンドだ。これは基本的に実行中の各プロセスの情報を示す。psのふだんの使い方といえば、プロセスのPID（プロセスID）を調べて、killコマンドでシグナルを送るといったものだろうが、psはPID以外にも、数多くの情報を提供してくれる。今回の主題である仮想記憶についても、多くの情報を提供してくれる。

まず、もっとも基本的な情報から見ていこう。psにオプションを付けずに実行すると、**画面3**のように、自分のプロセスに関する情報が得られる。

自分のプロセスといっても、現在使っている端末からのセッションに限られるわけではなく、複数のセッション

Column

スラッシングの経験

今はメモリが安く、標準で搭載されている容量もかなりあるので、普通の使い方をしている限り、そうそうスラッシングに至ることはない。486マシンでBSD/OSを使っていた頃は、さほどメモリもなかったので、結構ページングのお世話になった。16Mバイトの容量で動かしていたときは、起動直後は実メモリだけで間に合っているのだが、Xを使い始めると、動いていないデーモンがページアウトされた。32Mバイトにすると、Xサーバを起動してもページングはほとんど起こらないが、画像を多用するような処理を行うと、ページングが起こるようになった。個人で使うワークステーションでは、32Mバイトが実用上の最低メモリ容量だろう。

現在筆者が使っているシステムは、192Mバイトのメモリを搭載しているので、かなりの負荷をかけても、そうは簡単にスラッシングには至らないだろう。極端に負荷の重いサーバとか、特殊なアプリケーションを実行している環境でもない限り、そう簡単にスラッシングに至るものではない。よくある原因は、プログラムのバグによる異常なメモリ割り当てとか、操作ミスやスクリプトのミスによるプロセスの異常増殖などだ。

```
  PID TTY STAT TIME COMMAND
 1201  p1 S    0:00 -bash
 1688  p1 R    0:00 ps
 1521  p2 S    0:00 -bash
 1532  p2 S    0:00 sh /usr/bin/man ps
 1533  p2 S    0:00 man.exe ps
 1534  p2 S    0:00 sh -c /usr/bin/gunzip -c /var/catman/cat1/ps.1.gz | /usr/bin
 1536  p2 S    0:00 /usr/bin/less -is
```

画面3　オプションのないpsの出力

Column

制御端末

UNIXのプロセスは、その親プロセスから標準入力、標準出力、標準エラー出力を継承する。これは各種ttyデバイスに対する入出力用ハンドルの継承という形式で行われる。コンソール経由、シリアル回線経由、ネットワーク経由など、シェルセッションを開く方法はさまざまであり、そして使用するttyデバイスも、コンソールだったり、実在するシリアルI/Fであったり、ネットワーク用のデーモンが割り当てた擬似端末であったりする。ログインシェルは、それを生成した親プロセスからttyデバイスを継承し、そしてそのシェルから起動されたユーザープロセスも、シェルからttyデバイスを継承する。これが各プロセスの制御端末である。キーボード、ディスプレイ（あるいはI/Oリダイレクト）を使ってユーザーと対話するプロセスは、制御端末に対してI/Oを行い、その結果がユーザーとの対話になる。

一方、デーモンなどは、ttyを使った入出力を行わず、特定のデバイスと直接やり取りを行う。このようなプロセスは制御端末を必要としない。psでこれらのプロセスを見ると、TTYフィールドが？になっているが、これらは制御端末を持たないプロセスだ。

```
  PID TTY STAT TIME COMMAND
    1  ?  S    0:02 init [3]
    2  ?  SW   0:00 (kflushd)
    3  ?  SW<  0:00 (kswapd)
    4  ?  SW   0:00 (md_thread)
    5  ?  SW   0:00 (md_thread)
   38  ?  S    0:00
/sbin/kerneld
  222  ?  S    0:01 syslogd
  231  ?  S    0:00 klogd
  253  ?  S    0:00 crond
  265  ?  S    0:00 inetd
  276  ?  S    0:00 named
  287  ?  S    0:00 lpd     •
     •
     •
27541  p0 S    0:00 -bash
32476  p0 R    0:00 ps ax
```

画面4　ps axの出力

上のすべてのプロセスが表示される。この例では、psを実行したttyp1とttyp2でセッションを開いており、ttyp1ではpsを動かし、ttyp2ではpsのmanを見ていることがわかる（manの後に続くいくつかのプロセスは、manから起動されたプロセスだ）。このモードで表示されるのは、プロセスID（PID）、制御端末（TTY）、状態（STAT）、使用CPUタイムの分：秒（TIME）、コマンド名（COMMAND）である。特定のプロセスをキルするためにPIDを調べたいといった状況で使うことになるだろう。

プロセスの状態については、ちょっと説明が必要だろう。Rは実行可能（実行中を含む）である。S、D、Tは停止中を示す。Sはスリープで、キー入力やイベント待ち、プログラムによるスリープ状態など、シグナルによる割り込みを受理できる停止状態である。Dはディスク I/O待ちなど、割り込みを受理しない停止状態である。Tは、サスペンドなどで止まっている停止状態を示す。これらの状態に、W、N、不等号などが組み合わされることもある。Wはスワップアウトされた状態、Nはniceやreniceでプライオリティを変更されている状態、不等号もプライオリティが変更されている状態である。

ほかのユーザーのプロセスについても知りたい場合は、オプションaを指定する。またデーモンなど、制御端末を持たないプロセスについても知りたい場合は、xオプションを指定する。つまり、ps axと指定すれば、デーモンや他人のプロセスも含めて、全プロセスが表示されるわけだ**（画面4）**。

psでもう少し情報を調べてみよう。より多くの情報を表示するためのオプションがいくつかあるので、これを個々に紹介しよう。これらのオプションとa、xを組み合わせることで、自分のプロセスだけでなく、他人のプロセスやデーモンについても調べられる。

```
FLAGS   UID   PID  PPID PRI  NI   SIZE   RSS WCHAN      STA TTY TIME COMMAND
  100     0     1     0   0   0    792   416  12df7d     S   ?   0:02 init [3]
   40     0     2     1   0   0      0     0  127b1f     SW  ?   0:00 (kflushd)
   40     0     3     1 -12 -12      0     0  120691     SW< ?   0:00 (kswapd)
   40     0     4     1   0   0      0     0  16d17f     SW  ?   0:00 (md_threa
   40     0     5     1   0   0      0     0  16d17f     SW  ?   0:00 (md_threa
  140     0    38     1   0   0    752   364  1350d9     S   ?   0:00 /sbin/ker
100140    1   203     1   0   0    776   340  12df7d     S   ?   0:00 portmap
100040    1   208     1   0   0   1152   720  12df7d     S   ?   0:00 /usr/sbin
  140     0   222     1   0   0    812   436  12df7d     S   ?   0:02 syslogd
100140    0   231     1   0   0    792   400  1143e2     S   ?   0:00 klogd
```

画面5　ps lの出力の一部

・lオプション

プロセスのフラグ、親のPID（PPID）、実行の優先順位、使用メモリ量などが表示される**（画面5）**。ここで重要なのは、SIZEとRSSである。SIZEは、使用している仮想メモリ量、つまり、使っている実メモリとページアウトされた領域の合計である。RSSは実メモリ中のプログラムのサイズである。

・uオプション

uオプションは、各プロセスのメモリ、CPUの利用状況を表示する**（画面6）**。%CPU、%MEMは、各プロセスがCPUサイクル、実メモリ容量の何パーセントを使っているかを示す。これを見ることで、過度にメモリやCPUサイクルを消費しているプロセスを見つけることができる。

また、システムが異常に重いのに、過度にCPUサイクルやメモリを消費しているプロセスがないといった場合、スラッシングが起こっていたり、あるいはマシンパワーが絶対的に不足している（あるいは1台のマシンに負荷のかけすぎ）可能性がある。

・mオプション

このオプションを使えば、プロセスの使う実メモリについて、さらに詳しい情報が得られる**（画面7）**。MAJFLTは、実際にディスクアクセスを要したページフォルトの回数、MINFLTはディスクアクセスを要さなかったページフォルトの回数である。このページインには、コードやデータの最初のロードも含まれる。RSSは、プロセスの実メモリに常駐しているサイズ、TRSとDRSは常駐しているテキスト（コード）とデータのサイズである。

Column

System VのpsとBSDのps

今回の記事のpsの出力例は、すべてredhat 5.2 日本語版のものである。Red Hat系Linuxでは、5.2と6.xでpsのバージョンが変わっている。5.2のものはバージョン1、6.xのものはバージョン2で、バージョン2では、出力の形式やオプションなどが変更されている。また、バージョン2で、環境変数PS_PERSONALITYにoldを指定すると、バージョン1に近い形式になる。

psやvmstatなど、システムの状態を報告する各種コマンドは、その処理の性質上、システム（カーネル）の構造に大きく依存する。psは大きく分けて、System V系とBSD系の2種類があるが、これらはオプションがまったく違っている。今回解説したpsはLinux上で動作するものであるが、オプション体系は基本的にBSDのものである（かなり拡張されているが）。

またこの種のコマンドは、カーネル内部の各種変数やテーブルの内容を集計／表示するので、カーネルの微妙なマイナーチェンジの影響も受けることがある。Linuxの場合は、/procディレクトリの下にある各種ファイルを読み込むことで、システムの各種状態情報を簡単に得られるようになっているので、カーネル側の/procファイルサポートが互換性を意識していれば、仕様変更に強いものにできる。

編注：Linux kernel 2.2を採用したTurboLinux 日本語版 4.x、LASER5 Linux 6.0でもpsバージョン2を採用しています。

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
masa      6648  0.0  0.5  1248   808  p0 S    07:54   0:00 -bash
masa      6660  0.0  0.5  1248   808  p1 S    07:54   0:00 -bash
masa      7566  0.0  0.4  1212   652  p1 S    08:28   0:00 sh /usr/bin/man ps
masa      7567  0.0  0.2   844   448  p1 S    08:28   0:00 man.exe ps
masa      7568  0.0  0.4  1212   648  p1 S    08:28   0:00 sh -c /usr/bin/gunzip
masa      7569  0.0  0.2  1104   372  p1 S    08:28   0:00 /usr/bin/gunzip -c /v
masa      7570  0.0  0.3  1092   572  p1 S    08:28   0:00 /usr/bin/less -is
masa      7573  0.0  0.3   868   496  p0 R    08:30   0:00 ps u
```

画面6　ps uの出力

```
  PID TTY MAJFLT MINFLT   TRS   DRS  SIZE  SWAP   RSS  SHRD   LIB  DT COMMAND
 7569  p1     83     31    40   332   372     0   372   256     0  29 /usr/bin/g
 7567  p1    101     37    32   416   448     0   448   324     0  31 man.exe ps
 7572  p0    123     26    16   480   496     0   496   408     0  22 ps m
 7570  p1    140     23    84   488   572     0   572   452     0  30 /usr/bin/l
 7568  p1    167     47   184   468   652     0   652   540     0  27 sh -c /usr
 7566  p1    169     41   188   468   656     0   656   544     0  27 sh /usr/bi
 6648  p0    433    186   292   516   808     0   808   652     0  39 -bash
 6660  p1    422    168   292   516   808     0   808   652     0  39 -bash
```

画面7　ps mの出力

```
  PID TTY STAT TIME  PAGEIN TSIZ DSIZ  RSS   LIM %MEM COMMAND
 7569  p1 S    0:00     83   39 1064  372    xx  0.2 /usr/bin/gunzip -c /var/ca
 7567  p1 S    0:00    101   28  815  448    xx  0.2 man.exe ps
 7574  p0 R    0:00    123   10  857  496    xx  0.3 ps v
 7570  p1 S    0:00    140   77 1014  572    xx  0.3 /usr/bin/less -is
 7568  p1 S    0:00    167  322  889  648    xx  0.4 sh -c /usr/bin/gunzip -c /
 7566  p1 S    0:00    169  322  889  652    xx  0.4 sh /usr/bin/man ps
 6648  p0 S    0:00    433  322  925  808    xx  0.5 -bash
 6660  p1 S    0:00    422  322  925  808    xx  0.5 -bash
```

画面8　ps vの出力

・vオプション

このオプションは、各プロセスの仮想記憶関連の情報を表示する**（画面8）**。PAGEINはMAJFLTと同じで、ページフォルトの回数である。TSIZとDSIZは、それぞれテキストセグメント（コード部）、データセグメントのサイズである。

vmstatがシステム全体のメモリについて情報を提供するのに対して、psは個々のプロセスの情報を提供する。たとえば、サーバが提供するサービスが、メモリ不足により満足なパフォーマンスが得られないといった場合、一部のサービスを別のサーバに移行するといった方法で、メモリを有効に利用できるかもしれない。psの結果をうまく分析し、運がよければ、既存のメモリ量で問題なく動かせるデーモンの組み合わせを見つけることができるだろう。

■topコマンド

topコマンドは、ps、vmstat、free、uptime、killコマンドを組み合わせたようなもので、システムの稼働状況を継続的にモニターし、必要に応じてプロセスをキルすることができる。サーバなどのコンソールで表示しておくと便利といったコマンドだろう。起動すると、**画面9**のような画面が表示される。

最初の行はuptimeの出力と同じで、システムが起動してからの時間とユーザー数、過去1分、5分、15分のロードアベレージを表示する。次の行は、さまざまな状態にあるプロセスの数、その次はvmstatのCPUパラメータと同様の情報、続く2行がメモリ情報だ。avは利用可能量で、実メモリ容量とスワップ領域のサイズを示している。ほかのものはvmstatと同じだ。

その下はps axlとps axuを組み合わせたようなもので、各プロセスの状態を、CPU使用量やメモリ使用量などでソートして表示することができる。そして、キー操作でプロセスを選択して、シグナルを送ることもできる。

```
 10:17am  up 2 days,  7:09,  3 users,  load average: 0.00, 0.00, 0.00
61 processes: 58 sleeping, 2 running, 0 zombie, 1 stopped
CPU states:  1.1% user,  1.9% system,  0.0% nice, 97.1% idle
Mem:  160088K av, 156496K used,   3592K free,  31456K shrd,   9316K buff
Swap:  72256K av,    224K used,  72032K free                128496K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
30897 root      11   0   728  728   556 R       0  6.2  0.4   0:00 top
30895 root       3   0   356  356   320 S       0  0.7  0.2   0:00 script
    1 root       0   0   416  416   340 S       0  0.0  0.2   0:02 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root     -12 -12     0    0     0 SW<     0  0.0  0.0   0:00 kswapd
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
  444 news       0   0   748  748   584 S       0  0.0  0.4   4:31 innwatch
  436 root       4   0   812  812   656 S       0  0.0  0.5   0:00 bash
  437 root       0   0   304  304   252 S       0  0.0  0.1   0:00 mingetty
   38 root       0   0   364  364   308 S       0  0.0  0.2   0:00 kerneld
  203 bin        0   0   340  340   268 S       0  0.0  0.2   0:00 portmap
  208 bin        0   0   732  720   420 S       0  0.0  0.4   0:00 cannaserver
```

画面9　topの画面出力

Linuxの日本語環境

文：富樫 秀昭
Text：Hideaki Togashi

X-TrueType Server

X-TrueType ServerがXFree86のpre-4.0 snapshotに入ったことは前回紹介した通りだが、X-TrueType Serverそのものについては、以前に簡単に紹介しただけだったので、X Window Systemの論理フォント名のしくみとあわせて、今回もう少し詳しく紹介することにしよう。

X Window Systemのフォント

Xで使われるフォントは、抽象的な論理フォント名（XLFD：X Logical Font Description）によって管理されている。Xのアプリケーションがフォントを取り扱う場合は、すべてこのXLFDでフォントを指定し、Xサーバもしくはフォントサーバが動いている場合はそのフォントサーバが、指定されたXLFDを解釈してフォントファイルを読み、該当するフォントのメトリック情報やグリフをアプリケーションに提供したり、Xサーバが指定されたフォントを表示するなどするのである。

抽象的なXLFDを使うとはいえ、結局はそのXLFDに対応するフォントファイルが使われるのだということは、注意すべき点だろう。以下、xfontselを使って、具定的に見ていこう。

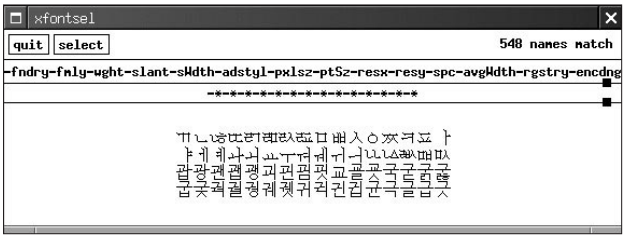
●Xの論理フォント名（XLFD）

XLFDを理解するには、xfontselを起動して実際にいろいろな指定を試してみるのが、一番の近道だ（**画面1**）。上から3行目の表示が、

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

となっているが、これは「−」で区切られたXLFDの各構成要素の指定に任意の文字列が使われていることを表している。このように、XLFDにはファイル名のように、任意の文字列に「*」、任意の1文字に「?」のワイルドカードを使うことができる。xfontselの最上段右端に指定されたフォント名にマッチしたフォントの数が表示される。

xfontselで各構成要素を指定するには、上から2行目の各構成要素を略字で表示した部分を左クリックする。そうすると、現在選択可能な指定の一覧が表示されるようになっている。左クリックしたままマウスを上下して、希望す



画面1　xfontselを起動したところ

画面2　xfontselでfmlyにtimesを選択

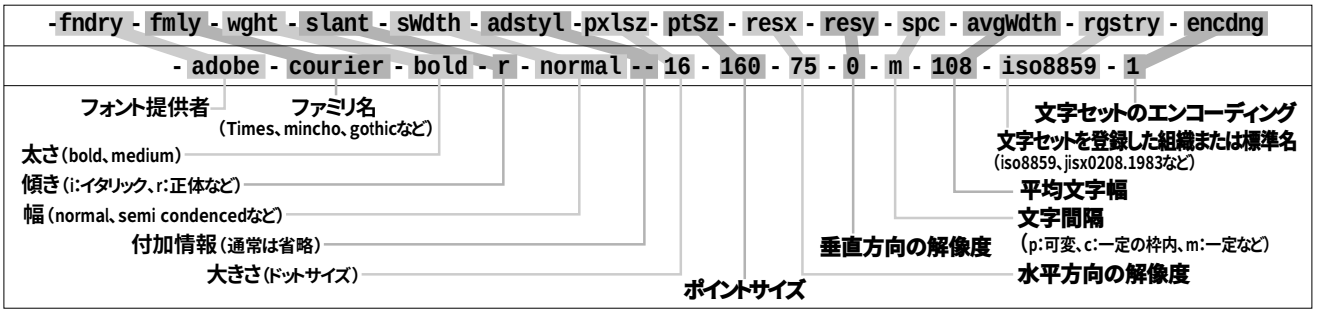


図1　XLFD：X Window Systemの論理フォント名

る指定でボタンを離すと、その指定が有効になる。各構成要素の意味は**図1**の通りである。各構成要素の詳細については、あとで必要に応じて触れることにしよう。まずはxfontselを使って、XLFDに慣れることが先決だ。

たとえば、左から2番目のfmlyで「times」を選ぶと、上から3行目の表示の左から2番目の表示も、

```
-*-times-*-*-*-*-*-*-*-*-*-*-*-*
```

という具合に「times」となり、下のサンプルの文字列も英数字とアクセント付きの英字となる（**画面2**）。これはtimesファミリで利用できる文字コードがiso8859だけなので、それに合わせたサンプル文字列が使われているのだ（環境によっては別の文字コード、たとえばkoi8なども選択できる場合がある。本稿では日本語版redhat Linux 5.2での例を示す）。この状態でほかの構成要素を選択すると、構成要素によっては濃い文字と薄い文字で表示されるものがあるが、薄い文字で表示されるものは現在の指定で選択できないものを表している。この環境で右から2番目のrgstryを左クリックしてみると、「iso8859」だけが濃い文字になっていて、ほかの文字コードは選択できないことを示している。

ほかの構成要素も左クリックしてみよう。slantをクリックすると、「i」と「r」だけが濃い文字になっている。「i：イタリック（斜体）」と「r：正体」のフォントが用意されているのだ。現在サンプル文字列はイタリックになっているが、このslantで「r：正体」を選ぶと、3行目も、

```
-*-times-*-r-*-*-*-*-*-*-*-*-*-*
```

に変化し、サンプル文字列も正体になる（**画面3**）。

今度は「wght」（weight：見た目の太さ）を左クリックしてみよう。「medium」と「bold」が選択可能になる。現在表示されているサンプル文字列は、boldだ。「medium」を選択してみよう（**画面4**）。3行目はこうなる。

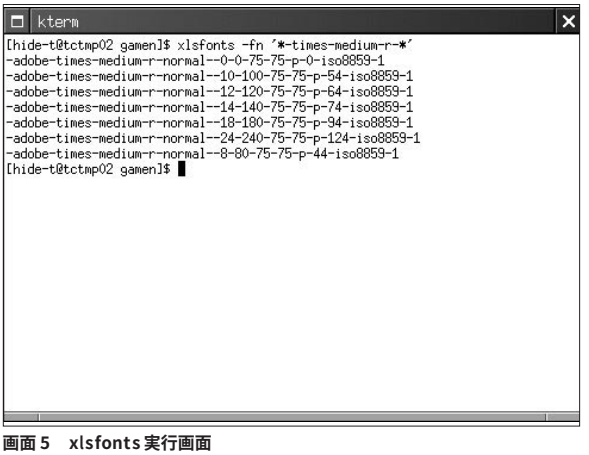
```
-*-times-medium-r-*-*-*-*-*-*-*-*-*-*
```

このXLFDにマッチするフォントのリストを見るには、xlsfontsを使ってたとえば次のように実行する（**画面5**）。

```
$ xlsfonts -fn '*-times-medium-r-*'
```

画面3　xfontselでslantにrを選択

画面4　xfontselでwghtにmediumを選択



画面5　xlsfonts実行画面

▼リスト1　xlsfontsで表示されるリスト

```
-adobe-times-medium-r-normal--0-0-75-75-p-0-iso8859-1
-adobe-times-medium-r-normal--10-100-75-75-p-54-iso8859-1
-adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
-adobe-times-medium-r-normal--14-140-75-75-p-74-iso8859-1
-adobe-times-medium-r-normal--18-180-75-75-p-94-iso8859-1
-adobe-times-medium-r-normal--24-240-75-75-p-124-iso8859-1
-adobe-times-medium-r-normal--8-80-75-75-p-44-iso8859-1
```

上のxfontselの3行目に表示された文字列をそのまま指定してもよいが、「＊」は任意の文字列の省略を表すので、「＊-＊-＊-＊-＊-＊-＊-＊-＊-＊-＊-＊-＊」全体を1つの「＊」で表現できるのだ。こう実行すると、上の環境ではリスト1のフォントが表示される。引数なしでxlsfontsを実行すると、全てのフォントが表示される。

このリストのうち実際に使われるフォントは、その指定で最初に見つかったフォントであるが、xlsfontsで表示される最初のフォントとは限らない。実際に使われるフォントを確認するには、次のようにxfdを実行する（**画面6**）。

```
$ xfd -fn '*-times-medium-r-*'
```

実際に使われるフォントが、xlsfontsで表示される一覧の最初のフォントとは限らないことに注意して欲しい。

●Xのフォント・インストール

通常フォントをリソースなどで指定するには、ここまでの知識で十分だが、X-TrueType Serverでフォントを指定するには、具体的なXLFDの指定方法についてもう少し詳しい知識が必要となる。そのために必要な一般的知識を、ここでまとめておこう。

Xのフォントは通常、「/usr/X11R6/lib/X11/fonts/」以下のサブディレクトリにインストールされている。現在フォントがインストールされているディレクトリを見るには、

```
$ xset q
```

と実行する。xsetは、Xの各種設定を表示・変更するためのツールだ。新たにフォントディレクトリを追加するには、

```
$ xset fp+ ディレクトリ名
$ xset fp rehash
```

と実行する。

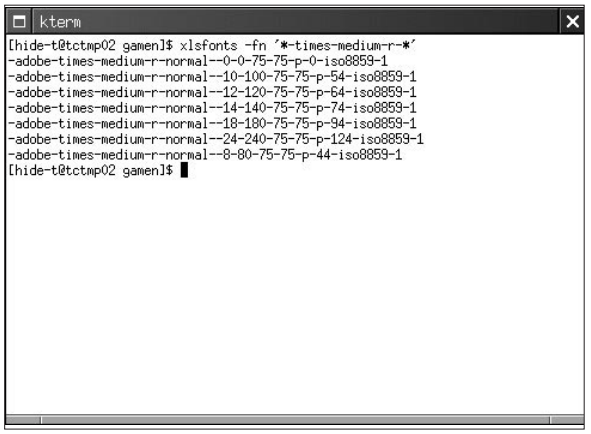
各フォントディレクトリにフォントを置けばよいのだが、各フォントがどういうXLFDに対応するかを指示する必要がある。その指示を行うのが、「fonts.dir」というファイルだ。これは各フォントディレクトリに置いておく必要がある。

fonts.dirは通常、各フォントディレクトリでmkfontdirというコマンドを実行すると自動的に作ってくれるのだが、X-TrueType Serverで細かくフォントを指定する際には、自分でfonts.dirにXLFDを記述する必要がある。

fonts.dirの形式は、**リスト2**のようなものだ。1行目に記述されているフォントの数を記述し、2行目以降に各フォントを、

```
フォントファイル名 XLFD
```

の形式で記述していく。たとえば、**リスト3**のようになる。XLFDには、もちろんワイルドカードは使えない。XLFDは、**図1**にある通りだが、以下fonts.dirにXLFDを記述する際に必要となるポイントをいくつか解説しておこう。カギ括弧内は、xfontselで表示される各構成要素の表記だ。



画面6　xfd実行画面

・FOUNDRY　[fndry]

フォント供給者名である。自由に設定できるが、通常はフォントを提供した会社名を使う。

・FAMILY　[fmly]

フォントの種類を表す。これも自由に設定できる。フォ

▼リスト2　fonts.dirの形式

```
フォント数
フォントファイル名1 論理フォント名1
フォントファイル名2 論理フォント名2
```

▼リスト3　通常のfonts.dirの例

```
198
UTI___24.pcf.gz -adobe-utopia-regular-i-normal--25-240-75-75-p-133-iso8859-1
UTI___14.pcf.gz -adobe-utopia-regular-i-normal--15-140-75-75-p-79-iso8859-1
UTI___12.pcf.gz -adobe-utopia-regular-i-normal--12-120-75-75-p-67-iso8859-1
UTI___10.pcf.gz -adobe-utopia-regular-i-normal--10-100-75-75-p-55-iso8859-1
courO24.pcf.gz -adobe-courier-medium-o-normal--24-240-75-75-m-150-iso8859-1
courO18.pcf.gz -adobe-courier-medium-o-normal--18-180-75-75-m-110-iso8859-1
courO14.pcf.gz -adobe-courier-medium-o-normal--14-140-75-75-m-90-iso8859-1
courO12.pcf.gz -adobe-courier-medium-o-normal--12-120-75-75-m-70-iso8859-1
          ・
          ・
```

ントを提供した会社がそのフォントの名前として使っている名前を使うのが普通だ。

・WEIGHT［wght］

文字の見た目の太さを表している。理想的にはこのフィールドが同じフォントがどれも同じ太さになるべきなのだろうが、実際には各社で開発されたフォントがどれも同じ太さになることはないのが現状であり、アプリケーションでも「medium」か「bold」しか使っていなかったりする場合があるので、それ以外を指定すると全く使われないフォントになったりする。割り切って、

```
medium ―― 通常の太さ
bold   ―― 太字
```

に大別して指定する。

・SLANT［slant］

文字の見た目の傾きを表す。通常、

```
r ―― 正体
i ―― イタリック
```

を指定する。逆イタリックの「ri」も指定できる。「イタリック」は本来フォント名で、このフィールドでも斜体を意味する「o」を使うことができるが、XLFDでは斜体に「i」を使うのが通例となっている。

・WIDTH［sWdth］

フォントの幅を表す。通常は「normal」を指定する。

・ADDITIONAL STYLE［adsty l］

付加情報。通常は省略される。

・SPACING［spc］

フォントのスペーシング（文字グリフと文字枠との間のアキ）の取り方を指示する。以下の指定ができる。

```
p ―― プロポーショナル
m ―― モノスペース
c ―― 定スペース
```

X-TrueType Serverの際には注意が必要となる。この点については、後述しよう。

・CHARSET_REGISTRY-CHARSET_ENCODING［rgstry］［encdng］

合わせてCHARSETと呼ばれることもある。xfontselでrgstryとencdngと表示される部分である。ここでキャラクタセットを指定する。キャラクタセットとしてはたとえば、

```
iso8859-1
jisx0201.1976-0
jisx0201.1983-0
unicode-0
```

などがある。

X-TrueType Server

X-TrueType Server（以下「X-TT」と略）は、TrueTypeフォントをX Window Systemの標準的なXLFDでスケーラブルに利用するためのXFree86のパッチである。X-TTそのものはパッチの形で配布されているが、このパッチの当てられたバイナリがRPM形式で配布

されており、現在の日本語Linuxではこのパッチを当てたXFree86が標準でインストールされるようになっているのが普通なので、導入に苦労するということはほとんどないだろう。TrueTypeフォントのラスタライズには、TrueTypeフォントのラスタライザライブラリFreeTypeを使っている。FreeTypeは、TrueTypeフォントのヒント情報も使っているので、小さなフォントでもかなりきれいに展開される。当初はVFlibを使って開発が行われたようだが、FreeTypeの出現によってそれを使うように方針転換されたとのことである。基本部分は渡邊剛氏によって作成され、現在のバージョン（1.3）は塩崎拓也氏によって管理されている。X-TTプロジェクトのWebサイトはhttp://x-tt.dsl.gr.jp/index-ja.htmlにあり、メーリングリストの案内、詳しい解説、いろいろな楽しい話などもあるので、ぜひご覧いただきたい。

ここでは、現在の日本語Linuxを普通にインストールしたところ、つまりX-TTの導入が一通り終わったところから、話をはじめよう。つまり、フォント環境を整えるところからである。ディストリビューションをインストールしたままの環境が十分とは限らないし、安価なTrueTypeフォントを購入して追加したいなどの場合にも、参考になるだろう。

環境としては、レーザーファイブから発売されている日本語redhat Linux 5.2を利用している。これには、X-TT 1.2を使ったXFree86 3.3.3.1が使われており、DynaLab JapanのTrueTypeフォントも導入済みとなっている。

なお、市販のTrueTypeフォントを利用する場合は、付属の使用許諾書などをよく読んで、その使用条件にしたがって利用してほしい。

●TrueTypeフォントのインストール

新たにTrueTypeフォントを自分で追加する場合は、フォントパスの通ったディレクトリ（/usr/X11R6/lib/X11/fonts/truetype/など）にフォントを追加し、fonts.dirを書く必要がある。

通常はmkfontdirを使えばこれを自動で書いてくれるのだが、TrueTypeフォントには利用できない。代わりに、http://www.io.com/~kazushi/xtt/でKazushi Marukawa氏がTrueType用のmkfontdirとして機能する「mkfontdir.pl」というPerlのスクリプトを提供してくれているので、これを利用するとよいだろう。

これをプロトタイプとして、必要に応じて記述を変更、追加して、

```
$ xset fp rehash
```

を実行する。

X-TTでTrueTypeフォントのXLFDを記述する際の注意をここでまとめておこう。X-TTのTrueTypeフォントのXLFDのプロトタイプは、**リスト4**のようになる。スケーラブルフォントなので、数値項目はすべて「0」にするのだ。ちなみに、ビットマップフォントでスケーラブルなフォントを利用する場合は、RESOLUTION_X、RESOLUTION_Yはもとの値のままにする。

注意が必要なのは、SPACINGのフィールドだ。プロポーショナルの「p」を指定すると、フォントに含まれる文字すべてのサイズを見にいくことになる。「m」を指定してもフォントの境界を全フォントを操作して決定するので、「p」ほどではないが処理に時間がかかることになる。256文字の英字フォントならば何の問題もないのだが、日本語フォントの場合は非常に文字数が多いので、極めて遅くなってしまう。このためプロポーショナルを指定する場合は、このあと説明するTTCapの指定を併用したほうがよいだろう。「c」を指定した場合は、すべてのメトリックにフォントファイルのヘッダに書かれた情報を使うので、こういった問題は起こらない。

●TTCap

TTCapは、fonts.dirの文法を拡張して、X-TTにオプションを与えて、TrueTypeフォントの指定をさらに詳細に行うことができるようにしたものである。

一般的な書式は、**リスト5**のようになる。通常の

▼リスト4　X-TTのTrueTypeフォントのXLFDのプロトタイプ

```
-fndry-fmly-wght-slant-sWdth--0-0-0-0-spc-0-rgstry-encdng
```

▼リスト5　TTCapの書式

```
TTCapオプション1:TTCapオプション2:フォントファイル名 論理フォント名
```

属性	説明
fn=整数	TrueTypeコレクション(*.ttc)ファイルのface番号を指定
ai=実数	文字の傾きを指定
fs=[pmc]	フォントメトリックの計算法を指定
bw=実数	フォントのバウンディングボックスの幅の倍率を指定
sw=実数	フォントの文字幅の倍率を指定
ds=[yn]	ダブルストライク効果の指定
vl=[yn]	メトリック計算方法を指定
eo=文字列	コード変換モジュールに渡すオプションを指定
hi=[yn]	ヒント情報を使うかどうかの指定
cr=範囲	フォントのコードの範囲を制限
eb=[yn]	埋込みビットマップを使う指定
bs=実数	埋込みビットマップの幅の倍率を指定

表1　X-TTのTTCapオプション一覧

fonts.dirの記述の行頭に、「:」で区切ってTTCapオプションを記述する。TTCapオプションは、さらに「:」で区切って複数指定することができる。TTCapの一覧は、**表1**の通りである。大文字小文字の区別はない。以下、主なものに簡単に解説を加えておこう。

・fn=整数（Face Number）

「.ttc」という拡張子を持つTrueTypeフォント（TrueTypeコレクションフォント）には、複数のfaceが含まれている。たとえば、Windowsで使われているmsgothic.ttcには「MS ゴシック」と「MS Pゴシック」と「MS UI Gothic」という具合だ。その複数含まれるfaceの何番目を使うかを指定するのが、このオプションだ。

・ai=実数（Auto Italic）

「ai=0.4」という具合に、文字の傾きを指定する。

・ds=[yn]（Double Strike）

以前は「ab（Automatic Bolding）」という名前のオプションだったが、バージョン1.3から変更されている。「ab」も使えるが「ds」が勧められている。「y」を指定すると、文字が太くなる。

・vl=[yn]（Very Lazy metrics calculation method）

XLFDのSPACINGでp、mを指定した場合に、ヘッダ情報をもとにして「非常に怠惰な方法」でメトリック計算を行う。メトリック情報は厳密なものではなくなるが、特に日本語のように文字数の多いフォントの場合は、表示がとても速くなるというメリットがある。Xのフォントは表示を前提としたもので、それほど厳密さが要求されるものではないので、このオプションを指定しても問題になることはほとんどないだろう。むしろ積極的に活用したいオプションだ。

・fs=[pmc]（Force Spacing）

XLFDのSPACINGフィールドの指定の代わりに、ここで指定した計算方法でフォントメトリックを計算する。

・sw=実数（Scale Width）

フォントの文字幅の倍率を指定する。実際のフォントの文字幅に、ここで与えられた数値が掛けられる。

・bw=実数（Bounding box Width）

固定幅のフォントでのみ有効。フォントのバウンディングボックスの幅の倍率を指定する。swも指定された場合は、swの計算が行われたあとで、このオプションが適用される。

●アプリケーション用のフォント指定

では、このTTCapを実際に使って設定を行ってみよう。

日本語版redhat Linux 5.2には、DynaLabのフォントが5つ入っているが、どれも違う書体なので、ここでは「DF華康明朝体W3」の「mimp3.ttc」のbold体として、「DF華康ゴシック体W5」の「gotp5.ttc」を使うことにしよう。

```
mincho      ―― mimp3.ttc
mincho bold ―― gotp5.ttc
```

「mincho」がFAMILYである。本来は違う書体なので同じFAMILYにすべきではないだろうが、日本では明朝のboldとしてゴシックを使うことはよく行われている。

利用するアプリケーションとしては、端末で動作するアプリケーションと、Netscapeを考えることにする。端末な

▼リスト6　1バイト系文字セット固定幅 mincho 正体

```
fn=1:mimp3.ttc -dynalab-mincho-medium-r-normal--0-0-0-0-c-0-iso8859-1
fn=1:mimp3.ttc -dynalab-mincho-medium-r-normal--0-0-0-0-c-0-jisx0201.1976-0
fn=1:ab=y:gotp5.ttc -dynalab-mincho-bold-r-normal--0-0-0-0-c-0-iso8859-1
fn=1:ab=y:gotp5.ttc -dynalab-mincho-bold-r-normal--0-0-0-0-c-0-jisx0201.1976-0
```

いしEmacsなどのような元来端末で利用することを前提としたアプリケーションでは固定幅のフォントが要求される。Netscapeでは固定幅のフォントとプロポーショナルなフォントが使われるが、NetscapeがSPACINGの「p」を理解するにもかかわらず、フォントをロードする際には「*」を要求するために同じ名前で「m」「c」があると、そちらが先にマッチしてしまうという仕様になっているため、プロポーショナルフォントのフォント名を変えておかなければならない。Netscapeでフォント名として認識するのは、

```
FAMILY(FOUNDRY)
```

の形式である。FOUNDRYはここでは「dynalab」を指定する。また、Netscapeで利用するプロポーショナルフォントのFAMILYは、「pmincho」とすることにしよう。Netscapeでの文字の変形には、TTCapを使った記述で対応する。よって、ここで作るのは、

固定幅 mincho **正体**
固定幅 mincho bold **正体**
固定幅 mincho italic
固定幅 mincho bold italic
プロポーショナル pmincho **正体**
プロポーショナル pmincho bold **正体**
プロポーショナル pmincho italic
プロポーショナル pmincho bold italic

の8種類の書体である。それぞれの書体について、1バイト系の文字セットiso8859-1、 jisx0201.1976-0と2バイト系の文字セットjisx0208.1983-0が必要になる。

・固定幅のフォント

まず、固定幅の1バイト系文字セットを作る。**リスト6**は正体のmediumとboldだ。XLFDから見ていこう。min3.ttcにもgotp5.ttcにも同じREGISTRYとFAMILYの「-dynalab-mincho-」を使っている。正体なのでSLANTは「r」、固定幅なのでSPACINGに「c」を使っている。1バイト系文字セットなので、CHARSETは「iso8859-1」と「jisx0201.1976-0」の2つを用意する。

TTCapの指定を見てみよう。min3.ttcもgotp5.ttcも、2番目のfaceが固定ピッチのフォントなので「fn=1」でそれを使うように指定している。gotp5.ttcには「ab=y」が指定してあるが、なくてもかまわない。boldが画面で伝わりにくいことを考慮して、指定してある。日本語redhat Linux 5.2に収録されているX-TTのバージョンは1.2で「ds」に変更される前のバージョンなので、「ab」を使っている。

リスト7のように同じセットのイタリックも用意する。XLFDは、イタリックなのでSLANTを「i」にしているところが違うだけである。TTCapの「ai=0.3」の指定で、イタリックに変形している。

2バイト系文字セットの「jisx0208.1983-0」について、上と同じセットを作ったのが**リスト8**である。

・プロポーショナルフォント

続いてNetscape用にプロポーショナルフォントを用意する（**リスト9**）。XLFDは、FAMILYを「pmincho」にし、プロポーショナルフォントなのでSPACINGを「p」にしてある。TTCapは、min3.ttc、 gotp5.ttcのプロポーショナルフォントのfaceを使うので、「fn=1」は必要ない。SPACINGに「p」を指定してあるためにロードが非常に遅くなってしまうのを避けるために、フォントメトリックの計算に簡略化した方法を使うことを指定する「vl=y」を指定しておく。後の部分は、固定幅のフォントと同じである。

▼リスト7　1バイト系文字セット固定幅 mincho italic

```
fn=1:ai=0.3:mimp3.ttc -dynalab-mincho-medium-i-normal--0-0-0-0-c-0-iso8859-1
fn=1:ai=0.3:mimp3.ttc -dynalab-mincho-medium-i-normal--0-0-0-0-c-0-jisx0201.1976-0
fn=1:ai=0.3:ab=y:gotp5.ttc -dynalab-mincho-bold-i-normal--0-0-0-0-c-0-iso8859-1
fn=1:ai=0.3:ab=y:gotp5.ttc -dynalab-mincho-bold-i-normal--0-0-0-0-c-0-jisx0201.1976-0
```

▼リスト8　2バイト系文字セット固定幅 mincho

```
fn=1:mimp3.ttc -dynalab-mincho-medium-r-normal--0-0-0-0-c-0-jisx0208.1983-0
fn=1:ab=y:gotp5.ttc -dynalab-mincho-bold-r-normal--0-0-0-0-c-0-jisx0208.1983-0
fn=1:ai=0.3:ab=y:gotp5.ttc -dynalab-mincho-bold-i-normal--0-0-0-0-c-0-jisx0208.1983-0
fn=1:ai=0.3:mimp3.ttc -dynalab-mincho-medium-i-normal--0-0-0-0-c-0-jisx0208.1983-0
```

▼リスト9　プロポーショナル pmincho

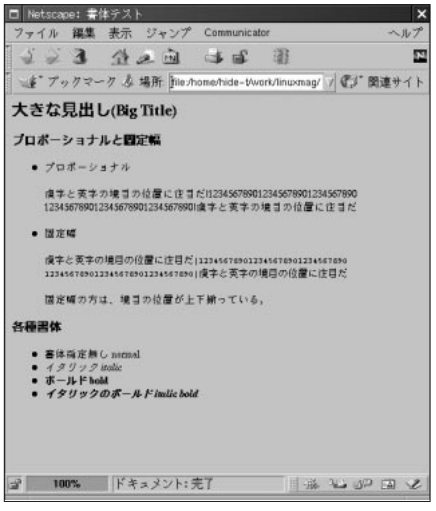
```
vl=y:mimp3.ttc -dynalab-pmincho-medium-r-normal--0-0-0-0-p-0-iso8859-1
vl=y:mimp3.ttc -dynalab-pmincho-medium-r-normal--0-0-0-0-p-0-jisx0201.1976-0
vl=y:ab=y:gotp5.ttc -dynalab-pmincho-bold-r-normal--0-0-0-0-p-0-iso8859-1
vl=y:ab=y:gotp5.ttc -dynalab-pmincho-bold-r-normal--0-0-0-0-p-0-jisx0201.1976-0
vl=y:ai=0.3:mimp3.ttc -dynalab-pmincho-medium-i-normal--0-0-0-0-p-0-iso8859-1
vl=y:ai=0.3:mimp3.ttc -dynalab-pmincho-medium-i-normal--0-0-0-0-p-0-jisx0201.1976-0
vl=y:ai=0.3:ab=y:gotp5.ttc -dynalab-pmincho-bold-i-normal--0-0-0-0-p-0-iso8859-1
vl=y:ai=0.3:ab=y:gotp5.ttc -dynalab-pmincho-bold-i-normal--0-0-0-0-p-0-jisx0201.1976-0
vl=y:mimp3.ttc -dynalab-pmincho-medium-r-normal--0-0-0-0-p-0-jisx0208.1983-0
vl=y:ab=y:gotp5.ttc -dynalab-pmincho-bold-r-normal--0-0-0-0-p-0-jisx0208.1983-0
vl=y:ai=0.3:mimp3.ttc -dynalab-pmincho-medium-i-normal--0-0-0-0-p-0-jisx0208.1983-0
vl=y:ai=0.3:ab=y:gotp5.ttc -dynalab-pmincho-bold-i-normal--0-0-0-0-p-0-jisx0208.1983-0
```

以上の設定が終わったら、追加したfonts.dirの1行目のフォント数を適切な値に直してから、元のユーザーに戻って、

```
$ xset fp rehash
```

と実行して、フォントを読み直そう。

実際にNetscapeで表示してみる。日本語redhat Linux 5.2でインストールした時のままの状態ではNetscapeで日本語の太字、斜体が表示できなかったのだが、**画面7**のようにきちんと表示できている。使っているhtmlソースは**リスト10**のものだ。Netscapeでフォントの指定を行うには、［編集］－［設定］で表示されるメニューの［表示］の中の［フォント］の項目で、プロポーショナルフォントに「pmincho」、等幅フォントに「mincho」を指定する。そして、その下の「スケーリング」をチェックしておこう。このチェックを忘れると、「<h1>…</h1>」などで日本語が拡大されなかった。以上の指定が終わったら、Netscapeを起動し直すと、新しいフォントの指定が有効になっている。**画面7**のように等幅フォントが使われているところで、上下の境目の位置が合っていない場合は、固定ピッチのフォントの全角フォントの幅と半角フォントの幅が2対1になっていない可能性があるので、そのような場合はTTCapのbwオプションで補正するとよいだろう。2対1になっていないと、端末で利用した時に表示が崩れることがある。



画面7　Netscapeでリスト10を表示

▼リスト10　Netscapeフォントテストソース

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<title>書体テスト</title>
</head>
<body>
<h1>大きな見出し(Big Title)</h1>
<h3>プロポーショナルと固定幅</h3>
<ul>
<li>プロポーショナル
<p>
漢字と英字の境目の位置に注目だ
|1234567890123456789012345678901234567890<br>
1234567890123456789012345678901234567890|漢字と英字の境目の
位置に注目だ<br>
</p>
<li>固定幅
<pre>
漢字と英字の境目の位置に注目だ
|1234567890123456789012345678901234567890
1234567890123456789012345678901234567890|漢字と英字の境目の
位置に注目だ
</pre>
固定幅の方は、境目の位置が上下揃っている。
</ul>
<h3>各種書体</h3>
<ul>
<li>書体指定無し normal
<li><i>イタリック italic</i>
<li><b>ボールド bold</b>
<li><i><b>イタリックのボールド italic bold</b></i>
</ul>
</body>
</html>
```

Webサーバ構築術(第4回)

Apacheに関する個別の設定が今回のテーマである。Apacheが動きだせば、管理者はさまざまな問い合わせを受けることになる。おそらく、問い合わせの中でもっとも多い内容は、画面に表示したくないバイナリをダウンロードさせる方法、ユーザー認証による会員制ページの作成、IPアドレスによるアクセス制限だ。

MIMEタイプ、ユーザー認証の設定

文:中島昌彦
Text : Masahiko Nakajima

たとえば、MP3ファイルをWebサイトで聞けるようにする場合を考えよう。Apache側でMP3のMIME設定を設定していない場合、IEではクリックするとMedia Playerが立ち上がる。

しかし、Netscape Communicatorでは、**画面1**のようにバイナリがそのまま表示される。Netscapeでファイル情報を見てみると、**画面2**のようにMIMEタイプがtext/plainとなっているのがわかる。

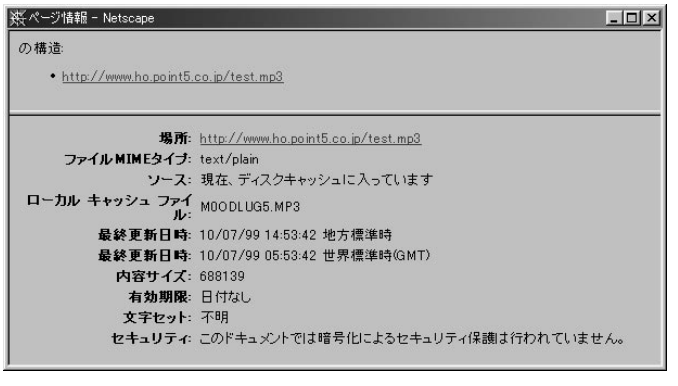
本来であれば、クリックしたらMedia Playerが起動して、MP3ファイルの再生をしてほしいわけで、ダウンロードしてほしいわけではない。ましてや、画面にバイナリを表示してほしいわけでもない。

この問題を解決するには、Apache側で、ファイルのMIMEタイプを設定する。これさえしておけば、サーバは正しくMIMEタイプを渡すため、ユーザー側のMIMEタイプ設定が正しければ、MP3ファイルが読み出されるわけだ。

こうしたMIMEタイプの設定をするファイルが、mime.typesである。

Red Hat LinuxのApache 1.3.6では、 /etc/mime.typesにファイルがあり、

画面1
MIMEタイプ非定義時に、mp3ファイルをNetscape Communicatorでアクセスした。ブラウザに渡されるMIMEタイプはtext/plainのため、Netscapeはバイナリを画面に表示する。



画面2
画面1のファイル情報。text/plainとして渡されている。

ここに情報を加えていく。

MIMEタイプは、

メディアタイプ/メディアサブタイプ　拡張子

の形式で割り付けていく。拡張子が複数ある場合はスペースで区切り羅列する。また、メディアタイプに関しては、RFC 1521に従う。

いずれにしても、Apache 1.3.6のmime.typesにはMP3のMIMEタイプ設定がデフォルトで加わっていない。このため、拡張子がmp3のファイルは、/etc/httpd/conf/srm.confで設定されているDefaultType設定のMIMEタイプで送出される。この場合、text/plainのMIMEタイプ設定が使われる。

このため、MIMEタイプに忠実なNetscapeは、MIMEヘッダに合わせてブラウザのナビゲーションウィンドウ内にバイナリが表示されるわけだ。

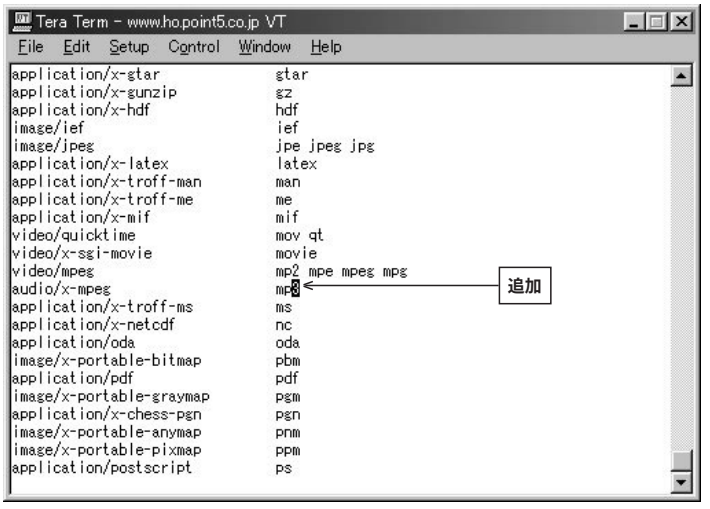
mime.typesにMIMEタイプを追加する

そこで、mime.typesにMIMEタイプを追加する。MP3であれば、メディアタイプがaudio、メディアサブタイプにx-mpegを指定する。そして、拡張子として、mp3を指定すれば動作する。

```
audio/x-mpeg mp3
```

という一行を、/etc/mime.typesに追加すればいい。場所はどこでもかまわないが、デフォルトの/etc/mime.typesは、拡張子順にソートされている。加えるならば、video/mpeg mp2 mpe mpeg mpgと application/x-troff-ms msとの間に入れておくと、あとで検索がしやすくなる（**画面3**）。

この追加をしたあとで、



画面3
mp3のMIMEタイプを追加した。RedHatであれば、/etc/mime.typesを修正し、/etc/rc.d/init.d/httpd restartでApacheを再起動すると、MIMEタイプの設定を読み込む。

```
# /etc/rc.d/initd/httpd restart
```

でApacheを再起動すると、/etc/mime.typesの設定が組み込まれる。この変更で、WindowsのNetscape Communicatorでも、MediaPlayerが立ち上がってMP3を再生するようになるはずだ。

前回説明したインデックス表示対応策として、/etc/httpd/conf/srm.confで、インデックス表示時のアイコン対策を施しておこう。タイプに合わせて、

```
AddIconByType （音声ファイル,/icons/sound2.gif） audio/*
```

という行を加えておけば、メディアタイプがaudioのものをすべて/icons/sound2.gifのアイコンで表示する。さらに、AddDescriptionとして、

```
AddDescription “MP3音声ファイル” .mp3
```

も加えておけば、インデックス表示時の対策も取れる。

なお、MIME設定時はmp3だけに限らず、WindowsやUNIXの圧縮ファイルも一緒に登録をしておくといい。どのMIMEタイプを割り当てるかで、それぞれ設定が変わってくるが、圧縮ファイルはローカルディスクに保存すべきものとすれば、

```
application/octet-stream lha
application/octet-stream zip
application/octet-stream sit
application/octet-stream sea
```

のような設定をしておくといい。

もっとも手間のかからないグローバルな設定

こうしたサーバ側の設定は、すべてのドキュメントに対して有効となるグローバルな設定だ。よく使うMIMEタイプに関しては、こうして登録してしまう方法がもっともベストだが、ユーザー側のMIMEタイプ設定の依頼に対して、即座に対応できるほどの余裕がないときもあるし、グローバルに設定するほど重要でないMIMEタイプの設定もある。その反面、ユーザー側にとってみれば、MIMEタイプの設定を望んでいる。

こうした場合、管理側のアプローチとしては2つの方法がある。ひとつは、DefaultTypeをtext/plainでないもの

にしてしまうこと。もうひとつは、.htaccessファイルで指定することである。たとえば、

```
DefaultType application/octet-stream
```

に設定し、Apacheをリスタートすると、未登録のMIMEタイプはすべてダウンロード対象となり、ナビゲーションウィンドウにバイナリが表示されるという最悪のケースが防げる。ユーザーにとっても、ファイルとして保存できるならば、それほど文句は出ないはずだ。

ただし、MIME未登録ファイルはすべて保存するというのにも問題がある。この場合、HTML登録者に対して、.htaccessで各自必要な設定を加えてもらうという方法がある。

.htaccessで指定するMIMEタイプの設定

.htaccessは、アクセスコントロール情報ファイルで、Apacheの設定変更をそのディレクトリ以下に対して有効にするものだ。グローバルに定義されているApacheの設定内容に対して、部分的に設定を変えるときに利用する。

ファイル名自体はデフォルトでは.htaccessとなっているが、必ずしもこの名前であるとは限らない。/etc/httpd/conf/srm.conf中で定義しているAccessFileNameで定義した名前が、アクセスコントロール情報ファイルとして利用される。

たとえばMP3のMIMEタイプを有効にしたいという問い合わせに対して、該当するディレクトリ上に

```
AddType application/x-mpeg mp3
```

と書いた.htaccessというファイルを置いておくように指示をすれば、管理者側に一切負担がかかることなくMIMEタイプ設定をコントロールできるようになる。むしろ、MIME設定に関しては、HTML登録者にある程度コントロールできる余地を残しておいたほうが、繁雑な管理から解放されるはずだ。

ただし、各ディレクトリの.htaccessを有効にするためには、/etc/httpd/conf/access.confの中の設定が必要になる。access.conf中に、AllowOverrideというコマンドがある（**表1**）。もしここがNoneであれば、.htaccessが有効とはならない。有効にするためには、Allを設定することになるが、AllにするとMIMEタイプ以外にも、ユーザー認証やOption指定もHTML登録者が行えるようになってしまう。このため、勝手にユーザー認証ページを作られたり、CGIを動かされかねない。確実にドキュメントタイプのコントロールだけを許したいならば、

```
AllowOverride FileInfo
```

というように、ファイル情報の.htaccessだけを有効にする。この設定をすれば、ドキュントタイプのコントロールだけを.htaccessでHTML登録者が管理できるようになる。ただし、ドキュメントタイプ以外のコマンドを.htaccessに記述したときは、Internal Server Errorが返ってくる。Apacheは不要のコマンドを無視するわけではなく、Internal Server Errorを返すため、エラーの原因がCGIのエラーなのか判別しにくい。こんなときは、/var/log/httpd/error_logを見て、

```
configured -- resuming normal operations
```

というエラーが記載されていれば、.htaccessでのエラーと判別する。万一エラーログが見られない状態であれば、動作内容からエラー元を特定する。.htaccessで不要なコマンドを指定した場合には、CGIや特定ファイルに限らず、そのディレクトリ以下のすべてのファイルにアクセスしようとすると、必ずInternal Server Errorが生じる。このエラーは独特なので、.htaccessで生じることさえ分かっていれば、比較的早期に解決できる症状だ。

None	ユーザーの.htaccessを認めない
All	ユーザーの.htaccess指定を優先利用する
AuthConfig	ユーザー認証のコントロールコマンドのみ記述を許す
FileInfo	ドキュメントタイプ関係のコマンドの記述のみ許す
Indexes	ディレクトリインデックス関係のコマンドのみ記述を許す
Limit	ユーザーのアクセスコントロール関係のコマンドのみ記述を許す
Options	Optionsやディレクトリ設定関係のコマンドのみ記述を許す

表1 AllowOverrideで使えるオプション

Apacheでユーザー認証をかける

Webサーバを運用していると、ユーザー認証ページの構築に関する問い合わせを受けることも多い。

この場合、何かしらのゲートウェイを動かしたり、一部の利用者のみ閲覧できる会員制ページを作りたいという目的であったりする。掲示板荒らしを避けるために、掲示板自体のCGIをユーザー認証の下に置くという使い方もよくある。

ユーザー認証をする方法にもいくつかあり、/etc/passwdを参照したり、

別のデータベースをアクセスする方法もある。が、もっとも手軽にできる方法は、Apacheのユーザー認証を使う方法だ。特に、SSLを使わずにユーザー名とパスワードを入力するとなると、ネットワーク上に平文のユーザー名とパスワードが流れる。セキュリティを考えると、/etc/passwdとは別のユーザー名とパスワードを利用すべきであるし、ユーザー情報を貯えているデータベースとも別の名前を使ったほうが得策といえる。

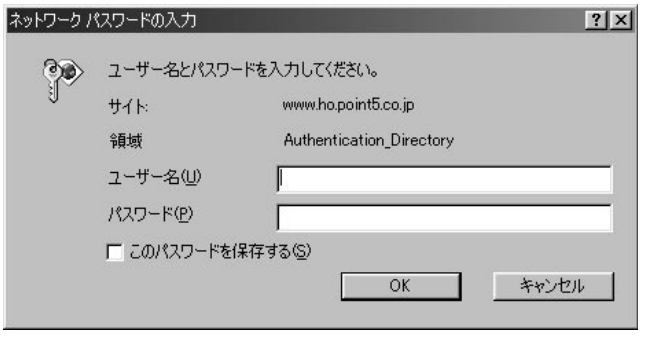
また、Apacheのユーザー認証を管理者が管理しようとすると、毎日のようにユーザー、パスワードの設定を持ち込まれる。繁雑なユーザー管理から解放されるには、ユーザー管理は利用者に任せるべきだ。そうでなければ、毎回ユーザー登録やパスワードの設定変更の依頼に振り回されることになる。

サーバ側では.htaccessを有効にする

Apacheのユーザー認証をユーザーに解放するためには、.htaccessを有効にする。このために、/etc/httpd/access.conf中にある、AllowOverrideを修正する。ここをAllowOverride Allか、AllowOverride AuthConfig Limitにする。Allを指定すれば、ユーザー認証とLimitに関するコントロール以外に、ドキュメントタイプのコントロールと、Option指定のコントロールもユーザーができるようになる。Auth Config Limitというように、ユーザー認証命令群とアクセスコントロールだけを指定すれば、それ以外のコマンドが.htaccessにあったときには、Internal Errorを戻す。スマートではないが、それなりのセキュリティは守れる。なお、さきほどのMIMEタイプの設定と共にユーザー認証も生かすならば、

▼リスト1 .htaccessのユーザー認証基本形

```
AuthType Basic
AuthName Authentication_Directory
AuthUserFile /home/httpd/html/.htpasswd
Require valid-user
```



画面4
AuthNameはブラウザに渡され、ユーザー認証ウィンドウで表示される。領域の意味は、一度認証が通ったエリアを判別するために、ブラウザは領域名で判断する。同じ領域名であれば、認証が通ったものとする。

```
AllowOverride FileInfo AuthConfig Limit
```

と指定すると、Option設定以外をすべてHTML登録者がコントロールできるという設定になる。

こうして、/etc/httpd/access.confの設定を変更し、

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

でApacheを再起動すると、ドキュメントディレクトリ内の.htaccessで、HTML登録者によるユーザー認証の設定ができる。

.htaccessによるユーザー認証

まず、ユーザー認証をかけたいドキュメントディレクトリに、次の内容のファイルを.htaccessという名前で保存する（**リスト1**）。

これが基本型になる。順に説明していくと、AuthType Basicは、これ以外の書式がない。Apacheのドキュメントによれば、現状ではBasicのみサポートである。

AuthNameが認証の名前部分だ。この部分は、ブラウザに渡され、ユーザー認証のウィンドウ内の領域として表示される（**画面4**）。注意する部分は、文字列内にスペースが含まれないことだ。もしスペースを含んだAuthNameを定義するなら、

```
AuthName “Authentication Directory”
```

というように、ダブルクォーテーションで名前部分を区切る。名前部分は日本語でも通らないことはないが、ブラウザ側がそこまで漢字コード変換に対応していない。EUCコードでは、IEもNetscapeのどちらも正しく表示されないため、7ビットの文字にしておくほうが無難だ。

AuthUserFileは、パスワードファイルの場所をフルパスで指定する。Webブラウザからダイレクトに見られないように、.htpasswdのように“.”（ドット）で始まるファイルにしておくほうがいい。

そして、最後のRequire valid-userで、パスワードの通ったユーザーのみ受け入れるという意味合いだ。

ただし、この場合では、すべてのアクセスに対してユーザー認証をかけている。掲示板のような仕組みでは、自由にだれでも見られるものの、書き込みだけ制限をかけたいというときがある。つまり、GETには認証をかけずに、POSTにだけ認証をかけたいという仕組みだ。

そこで、<Limit>…</Limit>で囲まれた範囲内で、どういう動作に対して認証をして、どういうユーザーに対して解放するかを指定する。

```
<Limit POST>
require valid-user
</Limit>
```

とすれば、POSTに関する操作のみユーザー認証の対象となる。逆に、閲覧はできないものの、書き込みだけは自由にできるという仕組み作りたいときには、

```
<Limit GET>
require valid-user
</Limit>
```

とする。

.htpasswdにパスワードファイルを作る

Linuxのパッケージでは、/usr/bin/htpasswdというものがある。これを使い、ユーザー認証用のファイルを作る。使い方は、

/usr/bin/htpasswd パスワードファイル名 ユーザー名

となるが、最初にファイルも作りそこにユーザー名を登録するときは、-cオプションを使う。

```
# /usr/bin/htpasswd -c /home/httpd/html/.htpasswd hogehoge
New password:          ←パスワードを入力
Re-type new password:  ←もう一度同じパスワードを入力
Adding password for user hogehoge
```

画面5 .htpasswdの新規作成

たとえば、.htaccessというファイルを新規に作成し、同時にhogehogeというユーザーを登録するときには、**画面5**のような作業を行う。New passwordとRe-type New passwordには、登録したいパスワード名を入力する。これで、ユーザー登録は終わりだ。

あとは、実際にそのディレクトリにアクセスすると、ユーザー認証がかかっている。そして、その認証を抜けない限り、コンテンツは表示されない。もし、Internal Server Errorが返ってくるようならば、.htaccessの設定がまちがっている。また、登録したユーザー名とパスワードを使っているのに、認証が抜けられないときは、たいていAuthUserFileで指定しているパスワードファイルの場所がまちがっている。そのあたりを確認したい。

ところで、htpasswdを使って吐き出されたファイルは、暗号化されている。しかし、htpasswdのメリットはそれだけではない。二重登録の抑制もでき、すでに登録をしてあるユーザーに対しては、パスワードをアップデートするという仕組みを持っている。

内部からのアクセスはパスワード認証なしに

Apacheの持つユーザー認証は実に便利な仕組みだが、使っていると不便を感じることもある。たとえば、Apacheをファイアウォール上に設置した場合、インターネット側からのアクセスに対しては認証をかけても、イントラネット側からのアクセスに関しては非認証で通したいといった要望だ。ほかにも、ハウジングやホスティング先のWebサーバに対して、自分の組織のIPアドレスからは非認証で、それ以外のIPアドレスには認証をかけるという仕組みだ。

こうした仕組みを実現すれば、社内や学校からアクセスするときに、認証ページであってもユーザー認証プロセスを通さずにすむ。そして、インターネット側からアクセスしたときでも、正しいパスワードを知ってさえいれば、目的のページにアクセスできる。

よく社内限定ページとして、IPアドレスだけでアクセスできるかできないかのチェックをしているページがあるが、その仕組みでは、社内や固定したIPアドレスを持っていないと目的のページにアクセスできない。出張時や外出時にISP経由でアクセスしようとすると、この方法では越えられないページがどうしても存在してしまう。

特定のドメインやIPアドレスからのアクセスは非認証、それ以外はパスワード認証という仕組みにするためには、.htaccessを**リスト2**のように書き換える。

変更のあった部分は、Requireよりうしろの行だ。まず、Satisfy anyで、いずれかのディレクトリアクセス条件を通ったものを通過させるという指示を出す。この場合では、パスワードによるユーザー認証か、IPアドレス、もしくはドメイン名のどちらかの条件が整っているユーザーのみアクセスできる。

Order deny,allowで、排除するか受け入れるかの指定をする。そして、Deny from allで、非認証ユーザーを

排除する。一方で、Allow from 192.168.1. ho.point5.co.jpの記述により、192.168.1.＊のIPアドレスと、逆引きをした結果が、ho.point5.co.jpからのアクセスであれば非認証で通す。

このIPアドレスやドメイン名部分を各自のものに変更すれば、Allow fromで指定したところからのアクセスが非認証となる。

ちなみに、Satisfy anyの部分をSatisfy Allに変更すると、IPアドレス、もしくはドメインと、パスワード認証のすべてが通過しないとアクセスができない。より重要なコンテンツが置かれている場所には、こうした二重のセキュリティを施す方法もある。

グループによるアクセス制限

Apacheのパスワード認証をひとつにまとめ、必要なユーザーのみユーザー認証をかけるという方式もある。どうしても管理者がApacheのユーザー認証を管理しなければいけない場合、こうした方法をとることもある。ただし、パスワードファイルが1つになっているため、1つのファイルを持っていかれると、すべてのパスワードが露出する危険もある。このように、パスワードファイル1つで、各コンテンツのアクセスを管理する場合、allow valid-userの部分を、

```
allow ユーザー名 ユーザー名
```

というように指定していく。ただし、何人もユーザー名を指定するようなときには、この方法では管理が繁雑になる。そこで、グループによるアクセス制限をかけることになる。

リスト3のような形式で、.htaccessを記述する。ユーザー認証時の形式に、AuthGroupFileが加わり、Requireでgroupオプションを追加して、Group名を指定しただけである。

▼リスト2 内部IPアドレスのみ非認証の書式（.htaccess）

```
AuthType Basic
AuthName “Authentication Directory”
AuthUserFile /home/httpd/html/.htpasswd
Require valid-user
Satisfy any
Order deny,allow
Deny from all
Allow from 192.168.1. ho.point5.co.jp
```

▼リスト3 グループによるアクセス制限（.htaccess）

```
AuthType Basic
AuthName “Authentication Directory”
AuthUserFile /home/httpd/html/.htpasswd
AuthGroupFile /home/httpd/html/.htgroup
Require group グループ名
```

一方のグループファイルである.htgroupは、

```
group名: ユーザー名 ユーザー名
```

という形式のファイルになる。たとえば、

```
ADMIN: user01 user02 user03
```

という.htgroupファイルを作れば、あらかじめ.htpasswdに登録してあるuser01、user02、user03をADMINグループとするという意味だ。このとき、.htaccess内のRequire groupを、

```
Require group ADMIN
```

としておけば、user01、user02、user03は認証が通ればそのコンテンツにアクセスができる。あくまでもApacheにとってみれば、ユーザー単位に認証を通過させたあと、グループ単位でアクセス制御をかける。UNIX系OSのグループパーミッション同様に、個人のユーザー認証をしたあとに、グループ単位にアクセスを許す処理であって、グループ単位での認証処理をする仕組みではない。

もしグループ単位で同じパスワードと同じユーザー名を使った認証をしたいのであれば、こんなややこしい仕組みを使う必要はない。セキュリティ面ではさらに大きな問題となるが、最初からユーザー名としてグループ名とパスワードを登録し、グループごとに登録したユーザー名とパスワードを共有してもらうほうが楽だ。

むしろ、こうした使い方をするよりも、部署単位で切り分けられたサブドメインやIPアドレスを使い、IPアドレスやサブドメインによるアクセス制御をしたほうが安心だ。ただし、イントラネットでしっかりとしたネットワーク設計がされている必要があるが。それさえできていれば、

```
Allow from 192.168.1. 192.168.2.
```

というように、サブドメイン単位で認証をかけたほうが、よほど安心である。

PostgreSQLを極める

最近のソフトウェアは、デフォルト設定のままでも使えるようになっているものが多くなっていますが、機能を十分に使いきるにはやはりカスタマイズが必要です。もちろん、PostgreSQLも例外ではありません。今回は、PostgreSQLカスタマイズの第一歩として「ユーザー定義関数」を紹介します。

第2回　ユーザー定義関数

文：片岡裕生
Text : Hiroki kataoka

前回は、PostgreSQLが持つ機能をいろいろ紹介しました。PostgreSQLがフリーソフトウェアでありながら、商用のRDBMSにひけを取らないくらい高機能なRDBMSであることがわかってもらえたものと思います。

しかし、このようなPostgreSQLが持つ機能を使いこなすためには、いくつかの基本的な事柄を習得しておかなくてはなりません。

今回は、こういった事柄の中で、最も基本的かつ重要な「ユーザー定義関数」について解説します。

ユーザー定義関数とは

PostgreSQLには、いくつかの関数が最初から搭載されています。たとえば、文字列中のアルファベットを大文字や小文字に変換するUPPER関数やLOWER関数、文字列の一部を取り出すSUBSTR関数などです。こういった、すでに搭載されている関数だけでもある程度のことはできますが、PostgreSQLを利用していると、「こんな関数があったらなぁ」と感じることもあります。

たとえば、テーブル内で男女の区別を“0/1”や“M/F”などのコードで表現していて、それらを“男性/女性”などのわかりやすい文字列に変換したい場合などです。もちろん、SQL文の中に直接そのような変換を行う命令を記述することもできますが、同様な変換が何度も必要になるようなら、あらかじめ関数にしておいたほうが便利でしょう。

また、別途マスターテーブルを持っていて、それを利用してコードから名称への変換を行う場合にも、関数になっていたほうがSQL文の記述が単純になり、便利な場合があります。

このような状況に利用できる機能として、PostgreSQLでは「ユーザー定義関数」というものが用意されています。これは、その名のとおりユーザーが新しい関数を作成できるという機能のことです。

ユーザー定義関数の動作を記述する言語にはいくつかの種類があり、その言語ごとに下記のようなユーザー定義関数があります。

- **SQL関数**
- **PL/Tcl関数**
- **PL/pgSQL関数**
- **C関数**

SQL関数は、ここに列挙した中では最も簡単に利用できるのですが、定義できる動作も簡単なことだけに限られてしまいます。したがって、より複雑な動作を定義したい場合には、PL/pgSQL関数を利用します。そして、最も柔軟な関数を定義できるのがC関数となるのですが、欠点もないわけではありません。というのも、C関数内で独自にテーブルにアクセスするためには、「SPI」と呼ばれるインターフェイスを介さなければならず、SQL関数やPL/

pgSQL関数に比べると容易とはいえません。また、C言語を利用するわけですから、関数はCコンパイラを利用して作成されるわけで、できあがった関数の実体（共有オブジェクト）は動作環境に依存する場合がほとんどです。したがって、C関数を利用する際は、その柔軟性と引き換えに、こういったデメリットがあることをあらかじめ認識しておく必要があるでしょう。

PL/Tcl関数とは、その名のとおりTcl言語を利用した記述言語で、比較的古くからサポートされていたのですが、PL/pgSQLが利用できる現在ではあまり使われないと思います。ですから、今回はPL/Tclの解説は行いません。

それでは、それぞれのユーザー定義関数について解説していきましょう。

SQL関数

関数の動作の定義にSQL言語を利用したユーザー定義関数です。前述したとおり、SQL関数でできることは少ないのですが、最も簡単に利用できるため、比較的出番の多いユーザー定義関数です。

SQL関数を作成するSQL文の形式は次のとおりです。

```
CREATE FUNCTION 《関数名》(《引数のデータ型リスト》)
RETURNS [SETOF] 《返り値のデータ型》
AS '《SQL文》'
LANGUAGE 'sql';
```

▼リスト1　SQL関数“sex_label”

```
CREATE FUNCTION sex_label(INTEGER) RETURNS
TEXT
AS '
        SELECT
        CASE
                WHEN $1 = 0 THEN ''男性´''::TEXT
                WHEN $1 = 1 THEN ''女性´''::TEXT
                ELSE ''不明''::TEXT
        END
'
LANGUAGE 'sql';
```

```
ascii2=> SELECT sex_label(1);
sex_label
---------
女性
(1 row)
```

画面1　SQL関数“sex_label”の実行

《関数名》には新しく作成する関数の名前を指定します。《引数のデータ型リスト》には、関数が受け取る引数のデータ型を“,”（カンマ）で区切って指定します。たとえば、整数（INTEGER型）1個と文字列（TEXT型）1個を受け取る関数の場合には、《引数のデータ型リスト》に“INTEGER, TEXT”と指定します。引数として既存のテーブルやビューと同じ構造の行データを受け取る場合には、データ型としてテーブル名やビュー名を指定できます。

《返り値のデータ型》には、関数が返す値のデータ型を指定します。《引数のデータ型リスト》と同様にテーブル名やビュー名も指定できます。ここで、SETOFキーワードを記述すると、複数の行を返すことができます。

《SQL文》には、関数の動作をSQL文で指定します。このSQL文内で関数の引数にアクセスするには、1番目の引数に対しては“$1”を、2番目の引数に対しては“$2”を、というように“$”文字の後ろに引数の番号を付加した形式で指定します。また、“;”（セミコロン）で区切ることにより、《SQL文》内に複数のSQL文を記述することもできます。なお、《SQL文》内で最後のSELECT文の結果が、関数の返り値になります。

CREATE FUNCTION文の中では《SQL文》全体を“'”（シングルクォーテーション）で囲んで文字列として指定するため、《SQL文》の中の“'”を記述するときは、すべて“''”（シングルクォーテーションが2個）で記述する必要がありますので、注意してください。

リスト1に、“0/1”の整数（INTEGER型）で表現されている男女の区別を“男性/女性”の文字列（TEXT型）に変換するSQL関数“sex_label”の例を示します。

画面1に、psqlコマンドのプロンプトからこの関数をテストしている様子を示します。

PL/pgSQL関数

PL/pgSQL関数は、柔軟でかつ比較的簡単に利用できるユーザー定義関数です。PL/pgSQL関数の動作の定義にはPL/pgSQL言語を利用しますので、まずこの言語について簡単に解説します。

PL/pgSQL言語によるプログラムは、1個の「ブロック」と呼ばれる処理単位で構成されています。ブロックの形式は次のとおりです。

```
[<<《ラベル》>>]  ※ “[”と“]”で囲んである部分は省略可能
[《宣言部》]
《処理部》
```

《ラベル》には、ブロックにつける名称を指定できます。《宣言部》では、《処理部》で使用する変数の宣言を行います。宣言した変数は、このブロック内でのみ有効です。《処理部》には具体的な処理を記述します。子ブロックを含めることもできます。

宣言部の形式は次のとおりです。

```
DECLARE
        《変数宣言》
        《変数宣言》
            :
```

《変数宣言》には複数の形式があるので、それらを196ページの**表1**にまとめておきます。処理部の形式は次のとおりです。

```
BEGINE
        《処理》
        《処理》
            :
END;
```

《処理》には具体的な手続きを記述します。ここでは通常のSQL文のほかに、条件文や繰り返し文なども記述できます。処理部に記述できるものを196ページの**表2**から**表4**にまとめておきます。

PL/pgSQL言語におけるコメント文の書き方には、SQLでは一般的な“--”（ハイフンが2個）形式のほかに、C言語と同じ“/* ～ */”形式も利用できます。

なお、1個のPL/pgSQL関数として作成できるプログラムのサイズには制限がありますので（だいたい7～8Kバイトくらいまで）、大きなプログラムを作成する場合には複数の関数に分割します。

さっそく、サンプルのPL/pgSQL関数を紹介したいところですが、実はPL/pgSQL関数を利用するためには事前の準備が必要なのです。というのも、PostgreSQLのデフォルトの状態ではPL/pgSQL関数が利用できない設定になっているのです。まずは、データベースに対してPL/pgSQL言語の登録作業を行わなければなりません。それにはPL/pgSQL言語を利用するデータベース上で**リスト2**に示したSQL文を実行します。なお、この登録作業はデータベースに対して一度だけ実行します。

なお、**リスト2**のSQL文中にある“/usr/local/pgsql/lib”の部分は、実際にお使いの環境に合わせて変更してください。具体的にはPostgreSQLのライブラリディレクトリを指定します。お使いのPostgreSQL環境がPostgreSQL標準のディレクトリレイアウト（/usr/local/pgsql/～）でインストールされているのであれば、上記のSQL文を変更する必要はありません。

PL/pgSQL関数を作成するSQL文の形式は次のとおりです。

```
CREATE FUNCTION 《関数名》(《引数のデータ型リスト》)
RETURNS 《返り値のデータ型》
AS '《PL/pgSQL文》'
LANGUAGE 'plpgsql';
```

上記はSQL関数の作成の場合とほとんど同じ内容ですが、SETOFキーワードが利用できない点と、最後のLANGUAGEの指定が“plpgsql”である点が異なります。そのほかは、SQL関数を作成する場合と基本的に同じです。

それでは、サンプルのPL/pgSQL関数を**リスト3**に示します。これは、住所から都道府県の部分のみを取り出すPL/pgSQL関数です。住所文字列の先頭から“都・道・府・県”のいずれかの文字を検索し、見つかったところまでを都道府県文字列としています。ただし、“京都府”の場合だけは、2文字目に“都”の文字が入っているので例外処理をしています。

なお、付属CD-ROMには住所から都道府県以外の部分を取り出すaddress_city関数も収録してあります。紙面の都合で解説はしませんが、そちらも参考にしてください。

このようにちょっと凝った処理も比較的簡単に記述できます。この例ではたまたまPL/pgSQL関数内でテーブルにアクセスしていませんが、PL/pgSQL言語ではそのような関数の記述も容易に行えます。

画面2に、psqlコマンドのプロンプトからこの関数をテストしている様子を示します。

▼リスト2　PL/pgSQL言語の登録

```
CREATE FUNCTION plpgsql_call_handler()
RETURNS OPAQUE
AS '/usr/local/pgsql/lib/plpgsql.so'
LANGUAGE 'c';          ← この部分は自分の環境に合わせる

CREATE TRUSTED PROCEDURAL LANGUAGE 'plpgsql'
HANDLER plpgsql_call_handler
LANCOMPILER 'PL/pgSQL';
```

C関数

C関数とは、PostgreSQLで利用するユーザー定義関数をC言語で記述し、Cコンパイラなどで共有オブジェクトファイル形式にコンパイルして利用するものです。C関数は、PL/pgSQL関数では効率の悪いような複雑な処理を記述するのに向いています。たとえば、全角文字を半角文字に変換するような細かな文字列操作などはPL/pgSQL関数よりC関数のほうが向いていることを示す好例といえるでしょう。

それではさっそく、文字列中の全角数字や全角ハイフン“－”を半角文字に変換するC関数“decimal_zh”の例（C言語ソースファイル）を**リスト4**に示します。

このリスト内で、最初にインクルードしているpostgres.hというヘッダファイルはPostgreSQLが提供しているファイルのひとつで、PostgreSQLの各種データ型の定義やPostgreSQLが提供する各種APIの関数宣言などが含まれています。ですから、ユーザー定義関数でC言語を利用する場合には、ソースファイルの最初のほうで必ずpostgres.hファイルをインクルードするようにします。

このソースファイルをコンパイルするには、**リスト5**のような手順で共有オブジェクトを作成します。

最後に**リスト6**のSQL文を実行してデータベースに登録すれば、C関数“decimal_zh”が利用できるようになります。

画面3に、psqlコマンドのプロンプトからこの関数をテストしている様子を示します。

このようにC言語を利用してユーザー定義関数を作成する場合でも、PostgreSQL本体を再コンパイルするといった面倒な作業は必要ありません。これも、PostgreSQLの拡張性の高さの賜物といえるでしょう。

ユーザー定義関数の利用

ここまでで、ユーザー定義関数についての大まかなイメージはつかんでいただけたかと思います。そこで、ここでは先ほどサンプルとして作成したユーザー定義関数を利用

```
ascii2=> SELECT address_pref('東京都小金井市中町4');
address_pref
------------
'東京都
(1 row)
```

画面2　PL/pgSQL関数“address_pref”の実行

▼リスト3　PL/pgSQL関数“address_pref”

```
CREATE FUNCTION address_pref(TEXT)
RETURNS TEXT
AS '
DECLARE
    addr ALIAS FOR $1;
    pref TEXT := NULL;
    c TEXT;
BEGIN
    -- 住所から“都・道・府・県”文字を探す。
    -- 探すのは最初の4文字で十分。
    FOR i IN 1 .. 4 LOOP
        c := SUBSTRING(addr FROM i FOR 1);
        IF c = ''都'' OR c = ''道'' OR
           c = ''府'' OR c = ''県'' THEN
            -- 見つかったなら、その文字までを都道府県とする。
            pref := SUBSTRING(addr FOR i);
            -- 京都府には例外処理が必要。
            IF pref = ''京都'' THEN
                pref := ''京都府'';
            END IF;
            EXIT;
        END IF;
    END LOOP;
    RETURN pref;
END;
'
LANGUAGE 'plpgsql';
```

▼リスト4　C関数“decimal_zh.c”

```
#include "postgres.h"

text *decimal_zh(text *zenkaku)
{
    text *hankaku;      /* 出力データ（TEXT型）領域 */
    int pz, ph;         /* 処理カウンタ */
    /* 出力データ（TEXT型）領域の確保 */
    hankaku = (text *)palloc(VARHDRSZ + VARSIZE(zenkaku));
    /* 処理カウンタの初期化 */
    pz = ph = 0;
    /* 入力文字列のスキャンを開始 */
    while (pz < VARSIZE(zenkaku))
    {
        int cw;                 /* 処理中の文字のバイト数 */
        unsigned char c0, c1;   /* 処理中の文字データ */
        cw = 1;
        /* 1バイト目のデータを取得 */
        c0 = ((char *)VARDATA(zenkaku))[pz++];
        if (pz < VARSIZE(zenkaku) && c0 & 0x80)
        {
            /* 処理中の文字は2バイト文字（漢字）だ！ */
            cw = 2;
            /* 2バイト目のデータを取得 */
            c1 = ((char *)VARDATA(zenkaku))[pz++];
            if (c0 == 0xa3 && c1 >= 0xb0 && c1 <= 0xb9)
            {
                /* 処理中の文字は全角数字だ！ 半角へ変換 */
                c0 = c1 - 0x80;
                cw = 1;
            }
            else if (c0 == 0xa1 && c1 == 0xdd)
            {
                /* 全角ハイフン“－”だ！ 半角へ変換 */
                c0 = '-';
                cw = 1;
            }
        }
        /* 処理結果を出力データ（TEXT型）領域へ格納 */
        ((char *)VARDATA(hankaku))[ph++] = c0;
        if (cw == 2)
            ((char *)VARDATA(hankaku))[ph++] = c1;
    }
    /* 出力データ（TEXT型）のデータサイズを設定 */
    VARSIZE(hankaku) = ph;
    /* 変換後のデータを返して関数終了 */
    return hankaku;
}
```

▼リスト5　C関数“decimal_zh”のコンパイル

```
gcc -I/usr/local/pgsql/include -fpic -c decimal_zh.c -o decimal_zh.o
ld -Bdynamic -shared -soname decimal_zh.so -o decimal_zh.so decimal_zh.o -lc
```

して、前回にも紹介した「顧客名簿」テーブルを操作してみようと思います。

あらかじめ断っておきますが、これから扱う顧客テーブル（今月号の付属CD-ROMに収録）は、実は前回（先月号の付属CD-ROMに収録）のものとは多少異なります。というのは、今回解説したユーザー定義関数をテストしてみるために、住所項目が「addr」1つだけになっていたり、「sex」カラムが追加されていたりします。

それではさっそく、先ほどのユーザー定義関数を利用していろいろやってみましょう。

最初に**表5**のような構造の顧客テーブル（テーブル名“customer”）があり、**画面4**のようなデータが格納されていたとします。これを見ていただければわかるとおり、住所や郵便番号に「あえて」全角数字や半角数字を取り混ぜてあります。

まず、最初に作成したSQL関数の“sex_label”を試してみましょう。この関数は整数の“0”や“1”を“男性”や“女性”に変換する関数でした。顧客テーブルの性別カラム（カラム名“sex”）を、この関数を利用して性別をわかりやすい漢字で表示させてみます**（画面5）**。

次に、PL/pgSQL関数の“address_pref”を試してみましょう。これは住所から都道府県を取り出す関数ですから、顧客テーブルの住所カラム（カラム名“addr”）を利用して試してみます**（画面6）**。

うまい具合に都道府県の部分だけが取り出されています。これを利用すれば、都道府県別に統計をまとめる際も安心です。

最後に、全角数字などを半角文字に変換するC関数“decimal_zh”を使ってみます。

前述のとおり、顧客テーブルの住所カラムや郵便番号カ

▼リスト6　C関数“decimal_zh”の登録

```
CREATE FUNCTION decimal_zh(TEXT)
RETURNS TEXT
AS '/コンパイルしてできあがった/decimal_zh.so'
LANGUAGE 'c';
```

```
ascii2=> SELECT decimal_zh('１２３４５');
decimal_zh
----------
     12345
(1 row)
```

画面3　C関数“decimal_zh”の実行

カラム名	データ型	説明
ccode	integer	顧客コード
name1	text	姓
name2	text	名
addr	text	住所
zip	text	郵便番号
sex	integer	性別（0=男性、1=女性）
email	text	電子メールアドレス

表5　顧客テーブル（テーブル名“customer”）

```
ascii2=> SELECT * FROM customer;
ccode|name1|name2 |addr                        |zip              |sex|email
-----+-----+------+----------------------------+-----------------+---+------------
    1|和栗 |真由美 |東京都杉並区高円寺北１－１５ |１６６－０００２  |  1|mayu@co.jp
    2|松林 |啓介   |東京都世田谷区代田５－１０   |１５５－００３３  |  0|keisuke@co.jp
    3|渡辺 |順子   |東京都八王子市旭町3-2        |192-0083         |  1|jun@co.jp
    4|斎藤 |純子   |東京都中野区東中野５－３     |１６４－０００３  |  1|junko@co.jp
    5|須田 |美加子 |埼玉県大宮市東町4-9          |330-0841         |  1|mk@co.jp
    :
(29 rows)
```

画面4　顧客テーブル“customer”の表示

```
ascii2=> SELECT sex, sex_label(sex) FROM customer;
sex|sex_label
---+---------
  1|女性
  0|男性
  1|女性
  1|女性
  1|女性
    :
(29 rows)
```

画面5　SQL関数の“sex_label”による性別表示の変更

ラムには、全角数字や半角数字は入り乱れて登録されています。このままでは住所から顧客を検索する際に、全角と半角の違いが原因で目的のデータがなかなか見つからないという事態も容易に想像できます。さっそく“decimal_zh”を使った住所の半角変換を試してみましょう**（画面7）**。

それでは最後に、今回作成したすべての関数を利用して、顧客テーブルの内容をわかりやすく表示させてみます。**画面8**がその実行結果です（付属CD-ROMに含まれているaddress_city関数も利用しています）。オリジナルデータを表示させた**画面4**と比べれば、違いが一目瞭然のはずです。

おわりに

今回はPostgreSQLの重要な機能のひとつ、ユーザー定義関数について解説しました。これはPostgreSQLを使いこなすうえで非常に重要な機能です。というのも、PostgreSQLが提供する高度な機能の中にはユーザー定義関数を利用しなければならないものがあるのです。今後解説していく予定のトリガやユーザー定義データ型などがその例です。いずれにしても、ユーザー定義関数はなかなか面白い機能ですので、気軽に使ってみてください。

なお、例によって今月の付属CD-ROMには、今回の解説で出てきたすべてのデータやソースコードが含まれています。また、それらを簡単にセットアップして実行するためのSQLファイルも用意してあります。詳しくは、付属CD-ROM内のREADME.txtを参照してください。

次回は、「トリガ」を中心に解説する予定です。

```
ascii2=> SELECT addr, address_pref(addr) FROM customer;
addr                        |address_pref
----------------------------+---------------
東京都杉並区高円寺北１－１５    |東京都
東京都世田谷区代田５－１０      |東京都
東京都八王子市旭町3-2          |東京都
東京都中野区東中野５－３        |東京都
埼玉県大宮市東町4-9            |埼玉県
    :
(29 rows)
```

画面6　PL/pgSQL関数“address_pref”による都道府県名の抽出

```
ascii2=> SELECT addr, decimal_zh(addr) FROM customer;
addr                        |decimal_zh
----------------------------+------------------------------
東京都杉並区高円寺北１－１５    |東京都杉並区高円寺北1-15
東京都世田谷区代田５－１０      |東京都世田谷区代田5-10
東京都八王子市旭町3-2          |東京都八王子市旭町3-2
東京都中野区東中野５－３        |東京都中野区東中野5-3
埼玉県大宮市東町4-9            |埼玉県大宮市東町4-9
    :
(29 rows)
```

画面7　C関数“decimal_zh”による全角→半角変換

```
ascii2=> SELECT ccode, name1, name2, address_pref(decimal_zh(addr)), address_city(decimal_zh(addr)),
decimal_zh(zip), sex_label(sex), email FROM customer;
ccode|name1|name2     |address_pref  |address_city        |decimal_zh   |sex_label |email
-----+-----+----------+--------------+--------------------+-------------+----------+--------------
    1|和栗 |真由美     |東京都         |杉並区高円寺北1-15   |166-0002     |女性      |mayu@co.jp
    2|松林 |啓介       |東京都         |世田谷区代田5-10     |155-0033     |男性      |keisuke@co.jp
    3|渡辺 |順子       |東京都         |八王子市旭町3-2      |192-0083     |女性      |jun@co.jp
    4|斎藤 |純子       |東京都         |中野区東中野5-3      |164-0003     |女性      |junko@co.jp
    5|須田 |美加子     |埼玉県         |大宮市東町4-9        |330-0841     |女性      |mk@co.jp
    :
(29 rows)
```

画面8　ユーザー定義関数によって整形されたサンプルデータ

《変数名》 [CONSTANT] 《データ型》 [NOT NULL] [:= 《初期値》];	**通常の変数宣言** 《データ型》には、通常のデータ型のほかに、既存のカラムや変数のデータ型を引用するための“テーブル名.カラム名% TYPE”表記も指定できる
《変数名》 《複合データ型》;	**行データが格納できる変数の宣言** 《複合データ型》には、指定したテーブルと同じ構造の行データが格納できる“テーブル名%ROWTYPE”表記や任意の行データが格納できる“RECORD”表記などを指定する
《変数名》 ALIAS FOR 《関数の引数》;	**関数引数に対する別名割り当て** $1や$2などの関数の引数にわかりやすい別名を与える。なお、関数の引数が行データの場合には必ず別名を与える必要がある
RENAME 《旧変数名》 TO 《新変数名》;	**変数の名称変更**

表1 PL/pgSQLの宣言文

IF 《条件式》 THEN
《TRUE処理部》
[ELSE
《FALSE処理部》]
END IF;
	条件処理

[<<《ラベル》>>] LOOP 《処理部》 END LOOP;	**無条件の繰り返し** 無条件に《処理部》を繰り返す。繰り返し処理を終了するには、RETURN文かEXIT文を利用する。《ラベル》には、繰り返し処理の名称を指定する
[<<《ラベル》>>] WHILE 《条件式》 LOOP 《処理部》 END LOOP;	**条件つき繰り返し** 《条件式》がTRUEを返す限り《処理部》を繰り返す。《ラベル》には、繰り返し処理の名称を指定する
[<<《ラベル》>>] FOR 《変数名》 IN [REVERSE] 《開始値》 .. 《終了値》 LOOP 《処理部》 END LOOP;	**整数値の繰り返し** 《開始値》から《終了値》までの整数を順に《変数名》に代入しながら《処理部》を繰り返す。《開始値》より《終了値》のほうが小さい場合にはREVERSEキーワードの指定が必要となる。《ラベル》には、繰り返し処理の名称を指定する。なお、《変数名》は自動的にINTEGER型として宣言され、繰り返し処理が終了した時点で自動的に消滅する
[<<《ラベル》>>] FOR 《変数名》 IN 《SELECT文》 LOOP 《処理部》 END LOOP;	**SELECT結果の繰り返し** 《SELECT文》を実行して、結果の各行を順に《変数名》に代入しながら《処理部》を繰り返す。《変数名》には行データが格納できる変数を指定する（自動的な宣言は行われない）。《ラベル》には、繰り返し処理の名称を指定する
EXIT [《ラベル》] [WHEN 《条件式》];	**ブロックもしくは繰り返し処理の終了** 《ラベル》には、終了したいブロックもしくは繰り返し処理の名称を指定する。《ラベル》を指定しなかった場合には、最も内側の繰り返し処理が対象となる。必要に応じて終了条件が指定できる。《条件式》を指定しなかった場合は無条件に終了する

表2 PL/pgSQLの制御文

RAISE 《レベル》 '《メッセージ》' [, 《式》] …;	**メッセージの表示、または処理の中断** 《レベル》には“NOTICE”（単にメッセージを表示）、“DEBUG”（デバッグ用メッセージの表示）、“EXCEPTION”（メッセージを表示してトランザクションを中断）のいずれかのキーワードが指定できる。《メッセージ》中に“%”文字が含まれている場合、“%”文字が現れる順に《式》の値と置き換えられる。“%%”文字は《式》ではなく“%”に置き換えられる。なお、《メッセージ》は文字列定数でなければならない
RETURN 《式》;	**関数の終了** 《式》には関数の返り値を指定する

表3 PL/pgSQLの命令文

《変数名》 := 《式》;	**式の代入** 《式》の演算結果を《変数名》に代入する
SELECT 《式リスト》 INTO 《変数名リスト》 [FROM …];	**式の代入（SELECT文形式）** 《式リスト》中の“,”（カンマ）で区切られた各式の演算結果を、《変数名リスト》中の“,”（カンマ）で区切られた各変数に代入する。FROM句などを記述することにより、テーブルにアクセスすることができる。なお、変数への代入が行われた場合、FOUNDシステム変数がTRUEにセットされ、代入が行われなかった場合にはFALSEにセットされる
PERFORM 《式リスト》 [FROM …];	**式の実行（演算）** 《式リスト》中の“,”（カンマ）で区切られた各式を演算する。このPERFORM文では式の実行結果を変数へ代入することはできない。FROM句などを記述することにより、テーブルにアクセスすることができる
INSERT文、UPDATE文、DELETE文、NOTIFY文など	**テーブルの操作や通知など**

表4 PL/pgSQLのデータ処理文

Rubyで行こう

突然ですが、Perl使ってます？　使いやすいですか？　読みやすいですか？　いいえ、と答えたあなたは幸福です。あなたには、Rubyがあります。Perlに挫折したあなたも、Perlに失望したあなたも、今回から始まるこのRuby講座で、Ruby時代の到来を実感してください。やっぱ、国産がいいっすよ。

第1回　Rubyとの遭遇

文：赤松智也
Text : Tomoya Akamatsu

近頃「Ruby」なるものの噂を耳にします。あの赤い宝石のことではなくて、ソフトウェア、それもプログラミング言語の名前だそうです。ルビーか、そういえばパールってのもあったな…

というわけで、この連載ではその噂（本当にあちこちで噂になってるのかどうかは不明）の元、オブジェクト指向スクリプト言語Rubyについて解説します。Rubyについてはすでにあちこちの雑誌で名前を見かけますし（http://www.ruby-lang.org/ja/press.html 参照）、作者自身による雑誌記事（「DDJ日本版」1997年10月号、「C MAGAZINE」1999年8月号、「TransTECH」1999年8月号）もありますが、本連載ではそれらとは違った切り口で解説しようと思っています。

今回は第1回なので、ありきたりではありますが、まず「Rubyとはなにか？」という点を紹介します。さらに、最近このRubyについて扱った最初の書籍『オブジェクト指向スクリプト言語Ruby』が出版されましたので、この本の著者のひとりでRuby作者でもある、まつもとゆきひろさんにインタビューした様子も紹介します。

Rubyとは?

Rubyというのは、まつもとゆきひろさんによって開発されたオブジェクト指向のスクリプト言語です。現在の最新の安定版は1.4.2です。開発版の1.5.0というものもあるようです。

Rubyのホームページの言葉によれば「手軽なオブジェクト指向プログラミングを実現するための種々の機能を持つオブジェクト指向スクリプト言語」で、「Perlのような手軽さで楽しくオブジェクト指向しようという言語」です。

要するに、設計者は初めから「オブジェクト指向言語」を意識して、Rubyを設計したようです。同じスクリプト言語の範疇に含まれるPerlなど他の言語は、まずスクリプト言語ありきで、オブジェクト指向機能はあとから追加されていますから、それらの言語とRubyとはちょっと印象が異なるはずです。

Rubyの仕様を見ていて、もうひとつ感じるのは「手軽だが、本格的」という点です。世の中にはたくさんのプログラミング言語があります。その中の多くは、いわゆる「簡易言語」で「手軽だが限定した分野でしか使えない」とか、「いまひとつ自由度が少ない」などの制約を感じさせます。ところが、Rubyはスクリプト言語としての「手軽さ」を犠牲にせずに、本格的なプログラミング言語の匂いを感じさせる仕様になっています。このあたりに作者の「本気」が感じられます。

REXXというプログラミング言語の作者である、IBMのMicheal Cowlishaw氏の発言に「言語の作者とユーザーではユーザーが圧倒的に多いのだから、トータルのコストから考えると苦労は言語作者が負うべきだ」というものがあります。Rubyの作者がこの言葉を知っているかどう

かはわかりませんが、Rubyの親切さというか機能の豊富さはこの言葉を思い起こさせます。そのほか、できることは徹底的にやるというような、よい意味のパラノイア的な雰囲気をあちこちで感じさせてくれます。

真・Rubyとは？

ま、言語仕様に基づく心理分析のようなことはこれくらいにして、実際のRubyの紹介に入りましょう。

オブジェクト指向スクリプト言語「Ruby」の特徴には、以下のようなものがあります。

・フリーソフトウェア

Rubyは、ソースコードが公開されたフリーソフトウェアです。ライセンスは、「GPL」もしくは「より制限の緩いRuby独自のもの」のいずれかを選択できます。したがって、利用には一切費用がかかりません。

・インタプリタによる手軽さ

Rubyは、インタプリタ型の言語ですから、コンパイル／リンクのような手間はありません。プログラムを書いたら、すぐに実行できます。そのため、修正／実行の間隔が短くなり、気持ちよくプログラミングできます。

・型宣言が不要

Rubyの変数や式には、型宣言がありません。コンパイル時の型チェックはありませんが、実行時に検出されます。このことにより、プログラムが簡潔で柔軟になる傾向があります。なお、型がないのは式や変数で、オブジェクトにはそれぞれの型がしっかりあります。これを「動的型」といいます。Rubyの変数にはどのような型の値も代入することができます。

・変数の種別が名前で決まる

Perlでも変数の種別（スカラー、配列、ハッシュ）ごとに変数名が違いますが、Rubyの場合には変数のスコープ（ローカル変数、グローバル変数、インスタンス変数）によって名前が変わります。Rubyでは、すべての値が参照（Perlにおけるリファレンス）ですから、Perlの意味での変数の種別の区別は存在しません。

・オブジェクト指向による統一感

Rubyは、最初からオブジェクト指向言語として設計され、組み込みの機能もクラスライブラリとして整備されています。言語全体がひとつの思想に基づいて設計されているので、仕様に統一感があります。

・書きやすく、読みやすい文法

作者によると、Rubyの文法は「プログラムが書きやすく、かつ比較的読みやすいプログラムになりやすい」と考えているようです。たしかに、Perlなどと比較してプログラムがわかりやすい気がします。

Column

ダレでもわかるオブジェクト指向

Rubyはオブジェクト指向言語だからうれしい、といわれても、オブジェクト指向のことを知らなければ、どんなにうれしいのかピンと来ませんよね。そこで、「オブジェクト指向のうれしさ」を簡単に紹介しましょう。

オブジェクト指向というのは、簡単にいえば「オブジェクトを中心にした考え方」です（まんまですが…）。「オブジェクト」というのは、普通のプログラミングにおけるデータと大差ないものですが、オブジェクトの場合は「自分のできることを知っている」という点が異なります。各オブジェクトは、ユーザーからの要求にしたがい、自分の内部構造に合わせた処理を、自分で選択して実行します。

たとえば、Rubyの場合、文字列も配列も長さを求めるためには、まったく同じ「size」というメソッドを使います。

```
v = [1,2,3]  # 配列
v.size       # => 3
v = "123"    # 文字列
v.size       # => 3
```

つまり、“v”という変数に入っているものが配列であろうと、文字列であろうと、内部構造をまったく気にせずに「長さを求める」という共通の処理を行うことができます。

旧来の方法（手続き指向）ではデータに手続きを適用するために、データと手続きの組み合わせが適切になされていないと、変な結果が出てしまいます。Perlでの例を見てみましょう。

```
$v = "123";   # 文字列
length($v);   # => 3
@v = (1,2,3); # 配列
length(@v);   # => 1???
$#v;      # => 2(配列の最後のインデックス)
$v = [1,2,3]; # 配列へのリファレンス
length($v);   # => 16???
```

もちろん、Perlでこのような値が返るの

・ガベージコレクション

Rubyには「ガベージコレクション」という機能があって、使われなくなったオブジェクトは自動的に回収されます。この機能により、プログラマーがどのオブジェクトはいつまで有効かということを管理する必要がなくなるうえ、間違って解放してしまったり、必要ないものを解放し忘れたりするようなエラーがなくなります。

・スレッド機能

Rubyにはスレッド機能があります。最近のOSでは普通になってきたスレッド機能ですが、なかなか実際に使う機会は少ないものです。Rubyのスレッド機能はユーザーレベルスレッドであるため、どのOSでも（DOSでも）動きます。どこでも同じ挙動ですから、安心して手軽にスレッドを使うことができます。

・マルチプラットフォーム

Rubyの開発は、主にLinuxで行われているそうです。しかし、FreeBSDをはじめとする各種UNIX系OS、またWindows NT/95/98などでも動作するということです。しかも、同じスクリプトが、どのプラットフォームでもほぼ同じように動作します。

こういった多くの特徴を持つRubyですが、実際使ってみた印象としては「Perlを一度バラバラにしてから、使いやすくなるよう統一感を持たせて再構成した」という感じでしょうか？　小さめのPerlスクリプトをいくつかRubyに移植してみましたが、「Rubyらしさ」にこだわらなければ、比較的すんなり移植できました。つまり、Perlインタプリタが持っている機能はひと通り持っているようです。

ということは、Perlが最適とされている多くの分野（テキスト処理、システム管理、CGI、ネットワークプログラミングなど）にRubyが向いていて、かつ多くの場合、Perlよりもわかりやすくプログラミングできるといえるでしょう。

となると、実行速度が気になるところです。単純なタスクの場合、Rubyプログラムの実行速度は、同等の働きをするPerlプログラムよりも数割程度遅いようです。ただ、数割の差なら、わかりやすさをとるか、速度をとるかでそれほど悩む必要はなさそうです。最近はマシンも高速になっていますし。また、Rubyではオブジェクト指向機能（特にメソッド呼び出し）はそれなりに最適化を行っているらしく、オブジェクトを多用したプログラムではRubyのほうが勝つこともあるようです。

Rubyの欠点といえば、アプリケーションとライブラリの蓄積がまだまだ少ないことでしょう。最近までは、ドキュメントなどの情報の少なさも欠点といわれていましたが、これは改善されてきているようです。

Rubyプログラミングの実際

では、このRubyを何に使うのがいいのでしょうか？　この答えは、やはり「Perlの代替として」ではないでしょ

にはそれぞれそれなりの理由があるのですが、見てわかるように総じてバラバラな印象を受け、「長さ(要素数)を求める」という概念的に同じものに対して、それぞれ違った方法を選択する必要があります。

しかし、前述のような知的（?）なデータを取り扱うことにより、人間にとって理解しやすいプログラムを作りやすくするのが、オブジェクト指向の特長です。

さらに、オブジェクト指向では、プログラミング組み込みのデータ構造に制限されないで、自分でオブジェクトを自由に定義できること、また、データ構造の定義にあたって、既存のデータ構造から似た部分を借りてくることができる（継承）という便利な機能もあります。

そして、おそらく最大の利点は、データを単なるデータとしてだけでなく、自分の処理を自分で選択する能動的なものとしてとらえることによって、人間がプログラムのイメージをつかみやすくなるという点でしょう。

Rubyは、言語全体がオブジェクト指向という一貫したポリシーに基づいて設計されていますから、全体が統一された印象を受けます。さらに結果として、プログラムのわかりやすさにも貢献しているのです。

オブジェクト指向は、すでに常識となりつつある考え方ですが、なかなか奥が深いものです。詳しい解説はここでは書ききれませんので、参考文献を参照してください。

■参考文献

『改訂新版 オブジェクト指向プログラミング』
石塚圭樹 著
ISBN4-7561-0276-X
アスキー出版局刊
本体価格3107円

『オブジェクト指向入門』
Bertrand Meyer著
酒匂寛・酒匂順子共訳　二木 厚吉 監訳
ISBN4-7561-0050-3
アスキー出版局刊
本体価格4835円

うか。前述のとおり、RubyはPerlの組み込み機能とほとんど同等の機能を提供していますから、Perlのできることはほとんど同じようにできます。しかも、同じ仕事がより読みやすく、わかりやすく書けるとなると、かなり魅力的に感じられます。

真のPerl使いは否定しますが、私のようなハンパ者にはPerlのプログラムがだんだん暗号のように見えてきます。自分で書いたものでさえこうですから、Perlで書かれた他人のプログラムなど、読める自信はまったくありません。一方、Rubyのプログラムはなんとなく読みやすい気がします。記号の密度が低いせいかもしれませんし、私がすでにRubyに慣れてしまったせいかもしれませんが…。

では、Perlの代替となる証明と、Rubyの特徴の紹介という意味を兼ねて、いくつかのプログラムをRubyとPerlの両方で紹介します。Perlに精通してなくても、「ちょっとは知っている」という人にもわかるようにするつもりです。この2つの言語の違いを堪能していただければ幸いです。

■ハローワールド

まずは、定番「Hello World」です（**リスト1**）。あれ？　ほとんど違いがないですね。こういう超カンタンなレベルではそれほど違いは出ません。ただ、Rubyは改行が文の区切りになるので、セミコロンが不要（Perlはブロックの末尾なら省略可）ということを指摘しておきます。

■現在時刻

今度は、現在時刻を表示してみましょう（**リスト2**）。出てくる個々の単語は似通っていますが、順番が違います。これは、Rubyがオブジェクト指向の「メッセージセンド」という形式を使っているからです。リスト中の

```
print Time.now.strftime("%a %b %e %H:%M:%S %Y\n")
```

は「Timeというクラスオブジェクトに、nowというメッセージを送って、その結果として現在時刻を表すオブジェクトを得て、そのオブジェクトにstrftimeというメッセージをフォーマットを引数として送って、フォーマットされた時刻の文字列を得て、それをprintする」という意味です。print以外は、語順が日本語における処理説明の順序どおりになっており、すっきりしているのがわかります。

この表記の違いに、オブジェクトをベースにしたRubyと関数（手続き）をベースとしたPerlの差が表れています。

■ネットワーク

Rubyではネットワークプログラミングも簡単です。たとえば、ローカルホストから時刻を取り出すプログラムです（**リスト3**）。なお、ローカルホストが、時刻を返すdaytimeサービスを提供していない場合には、“ localhost ”の部分を適当なホスト名に変えてみてください。

Rubyの場合は、ライブラリ利用の宣言であるrequire行を除くと、実質1行ですんでいることに注目してください。Rubyでは、ネットワーク機能が「TCPSocket」というクラスにまとめられています。

それに比べて、その下にあるPerlプログラムは、同じ働きをしているにもかかわらず、かなり冗長な感じです（**Perlその1**）。Perlでのソケットプログラミングは、Cのスタイルをそのまま継承しているため、かなり複雑に見えます。

ただし、Perlにはオブジェクト指向機能を使って、もう少し簡単に記述する方法もありますから、公平を期すため、こちらも紹介しておきましょう（**Perlその2**）。こちらは最初のもの（Cスタイル）に比べると、かなりすっきりし

▼リスト1　ハローワールド

■Ruby
```
print "hello world"
```

■Perl
```
print "hello world";
```

▼リスト2　現在時刻の表示

■Ruby
```
print Time.now.strftime("%a %b %e %H:%M:%S %Y\n")
```

■Perl
```
use POSIX qw(strftime);
print strftime("%a %b %e %H:%M:%S %Y\n", localtime(time()));
```

ています。Rubyとどちらがよいかは意見が分かれるところでしょうが、Rubyのスタイルのほうが簡潔で、しかも読みやすいように思えるので、私はRubyのほうが好みです。

■例外

リスト4に示したのは、それぞれファイルをオープンするプログラムです。

Perlのほうにある「or die ～」というのは「オープンに失敗したら終了する」という有名なイディオムですが、Rubyにはそれに相当するものが見当たりません。

Rubyには「例外」という機能があり、プログラマーが明示的に失敗を検出しなくても自動的にPerlの「or die ～」相当のことを行ってくれます。Javaの例外をご存じの方は似たようなものだと考えてください。この機能により、プログラムが簡単になるだけでなく、失敗の検出を忘れて突き進んでしまうようなミスを自動的に回避できます。

■ディレクトリ操作

リスト5に示したのはあるディレクトリにある、.（ドット）で始まるファイルを取り出すプログラムです。

今までのプログラムよりも違いが大きいですね。Ruby版ではDirクラスのopenメソッドによってディレクトリオブジェクトを得て、そのオブジェクトのcollectメソッドを呼び出しています。collectメソッドは、各要素に対して後

▼リスト3　ネットワークプログラミング

■Ruby
```
require 'socket'
print TCPSocket.open("localhost", "daytime").read
```

■Perlその1
```
use Socket;

$proto = getprotobyname('tcp');
socket(Sock, PF_INET, SOCK_STREAM, $proto);
$port = getservbyname('daytime', 'tcp');
$sin = sockaddr_in($port, inet_aton("localhost"));
connect(Sock,$sin) || die "connect: $!";
print <Sock>;
```

■Perlその2
```
use IO::Socket;

$sock = IO::Socket::INET->new(PeerAddr => 'localhost:daytime');
print <$sock>;
```

▼リスト4　例外

■Ruby
```
f = open("/path/to/file")
```

■Perl
```
open(F, "/path/to/file") or die "open: $!"
```

▼リスト5　ディレクトリ操作

■Ruby
```
dots = Dir.open(some_dir).collect{|f|
  /^\./ && File.exist?("#{some_dir}/#{f}")
}
```

■Perl
```
opendir(DIR, $some_dir) || die "can't opendir $some_dir: $!";
@dots = grep { /^\./ && -f "$some_dir/$_" } readdir(DIR);
closedir DIR;
```

ろに付いている { } （ブロックと呼びます）の内部を実行して、それが「真」であるものを集めます。このような、要素に対する繰り返し処理を行う手続きを「イテレータ」と呼びます。Rubyではイテレータをユーザーが任意に定義できるという特徴があります。

Ruby版ではclosedirに相当する処理を行っていませんが、ガベージコレクションが後始末をきちんとしてくれます。

Rubyの入手方法

今回は簡単な紹介ということで、Rubyの詳細には触れずに、Perlといくつかの機能について比較してみました。いかがだったでしょうか？　少なくともここで紹介した範囲内では、手軽で簡潔だといわれているPerlと比較しても負けないどころか、さらに簡潔に表現できていることを実感していただけたのではないかと思います。

Rubyを学ぶには使ってみるのが一番です。Perlのように初めからパッケージに組み込まれているLinuxディストリビューションはまだまだ少ないのですが、JRPMプロジェクト（http://jrpm.linux.or.jp/）に.rmpファイルが登録されているので、RPM系のディストリビューションをお使いの方は、これをダウンロードして使ってください（TurboLinux 4.0およびTurboLinux Pro 4.2には収録されているが、バージョンが1.1c9で古い）。

また、Debian GNU/Linuxの最新バージョンでは本家に登録されていますので、dselectまたはapt-getで「ruby」というパッケージを選択するだけでOKです。

最新版を利用したい方は、下記の手順でソースコードからインストールしてください。

```
$ tar xvzf ruby-1.4.2.tar.gz
$ cd ruby-1.4.2
$ ./configure
$ make
 (rootになって)
# make install
```

今月号のCD-ROM Disk2にrpmファイルとソースコードを収録してありますのでどうぞ試してみてください。

今後の予定

次回以降から、具体的なプログラムを交えながら、Rubyをどう活かすか、あるいはRubyでいかに楽するかを中心に解説していきたいと思います。Perlを引き合いに出すことは多くなりそうですが、Perlの知識がなくてもわかるような記事にしたいと思っています。

情報と文献

来月以降の準備のために、Rubyの参考となる情報は以下のWebサイトで得ることができます。

■Rubyホームページ（工事中）

http://www.ruby-lang.org/ja/

■旧Rubyホームページ　netlab.co.jpは、まつもとさんの勤務先

http://www.netlab.co.jp/ruby/jp/

■Unofficial Ruby Home Page　メーリングリストの過去のトピックが整理されているので必見です。

http://ruby.freak.ne.jp/

■Linkavailable: Ruby　Ruby関連のページが更新時間順に整理されています。

http:www.jin.gr.jp/~nahi/link-Ruby.html

■Ruby関連書籍

『オブジェクト指向スクリプト言語Ruby』
まつもとゆきひろ・石塚圭樹 著
ISBN4-7561-3254-5

■Rubyオフィシャルサイト

Ruby本発売記念直撃インタビュー

長らく「出る、出る」と噂されていた、作者執筆によるRuby本『オブジェクト指向スクリプト言語Ruby』がとうとう出版されます（原稿執筆時点では未発売）。

そこで、Ruby本発売を記念して、著者のひとりでRubyの設計・開発者でもある、まつもとゆきひろさんにインタビューを行いました。なお、このインタビューは対話風に再構成したもので、文責は赤松にあります。

赤松（以下A）：とうとう出ましたね。
まつもとさん（以下M）：とうとう出ました。過去の文章などを振り返ると1997年の秋ごろから「出るぞ、出るぞ」といい続けていました。もう2年も遅れてますね。一時はあいさつ代わりに「Ruby本はいつ出るんですか」と（笑）。あんまり出ないんで、ほとんど狼少年のような状態だったのですが、とうとう本当になりました。本業が忙しかったとか、思ったより分量が多くなったとかいろいろ言い訳はあるのですが、とにかく長らくお待たせして本当に申し訳ありませんでした。

A: で、そのとうとう出たRuby本はどんな本なんですか？
M: 内容はこんな感じです（「目次」参照）。文法の解説から、プログラミング全般、果てはRubyインタプリタの内部構造まで盛りだくさんです。当初は、A5版400ページ程度を考えていたんですが、いろいろ書いてるうちに結局B5版600ページ弱にまでふくらんでしまいました。最初の本ということでどうしても欲張ってしまったようです。これも遅くなった原因のひとつなんですが…。

この本を一言で表現するなら、「何度も参照する本」というよりは「読みながら学ぶ本」ではないかと思います。ただし、付録Aのミニリファレンスは参照するにはなかなか便利だと思います。「ミニ」といいながらも、結構分量があるんですが。

■Ruby本「目次」

1章 Rubyとは
2章 Ruby入門
3章 Rubyプログラミング入門
4章 Rubyオブジェクト指向プログラミング
5章 Rubyオブジェクト指向設計
6章 Rubyオブジェクト指向プログラミング - 実践編
7章 応用Rubyプログラミング
8章 Inside Ruby
9章 Rubyを強化する
10章 Rubyを取り巻く世界
付録A Rubyミニリファレンス
付録B プログラムリスト
付録C Ruby用語集

A: この本の対象はどのような人でしょう。
M: 少なくとも何らかのプログラミングの経験があったほうが読みやすいでしょうね。わりと親切に書いたつもりですが、プログラミングの入門書には向かないかもしれません。ただし、プログラミングの経験がある人にとっては、Rubyの情報源としても、もう一歩先のプログラミングを学ぶためにも役に立つのではないでしょうか。また、本書中で何度もPerlのことに触れていますが、Perlに対する知識は特に必要ではありません。

A: 内容の特徴というか、見どころはどこですか？
M: やはり、一番見どころになるところは、3章から7章までのRubyプログラミングを解説している部分だと思います。このあたりはRubyの本というよりもRubyを素材にしたプログラミングの本という印象もあります。

5章では他の書籍とは違った切り口から「プログラマーによるプログラマーのためのオブジェクト指向設計」を解説しています。プログラマーである私が、何となく直感でクラスやメソッドを設計しているやり方を文章化しました。このやり方が絶対だというつもりはありませんが、こういう文章はあまり見ませんし、参考にはなると思います。

また、6章ではいわゆる「ちゃんとした」オブジェクト指向設計の手法についても解説してます。この解説は、オブジェクト指向設計の本家、日本ラショナルの石塚圭樹さんによるものですから、内容的には申し分のないものになっています。

ほかにも2章、8章、9章、付録Aなどは今までリファレンスマニュアルなどでもきちんと文書化されていなかった内容にまで踏み込んでいるので、それなりに価値があると思います。

付録Cは、Rubyホームページの「今日のひとこと」で好評だった「Ruby用語集」の総集編です。Perlのラクダ本のマネじゃないかという声も聞こえてきそうです

が、まったくもってそのとおりです（笑）。ただし、内容は十分楽しんでいただけるものになったのではないかと思います。あ、でも、これは「読みもの」ですから用語集としての実用性はあまり期待しないでくださいね。

A: 表紙はどうなったんですか？
M: この記事が載るころにはとっくに書店に出回っているでしょうが、「普通」の表紙です。宝石が表紙になってます。一部の人には残念かもしれませんが、動物の表紙じゃありません。オライリーの本じゃないんで…。まあ、本の愛称は普通表紙のデザインで決まるんで、これで名実ともに「Ruby本」になったかなと。ただ、各ページの上のほうに小さな羊とハチドリのイラストが入ってます。これは編集の方が気を利かせて入れてくださいました。

A: どうして羊とハチドリなんですか？
M: 羊は、うちの奥さんと母が羊が好きで、羊グッズを集めてたりするんです。私も好きですけど。で、まつもと家のマスコットということで「羊」を選びました。ハチドリはRuby throated hammingbirdというのがブラジルにいるんだそうで、そこからの連想です。以前、メーリングリストで表紙について話題になったときには、鳩、兎、カバ、今井美樹、小泉今日子なんかが候補として挙がりました。

A: 鳩とかはともかく今井美樹ですか？
M: 『Ruby』ってアルバム出してますしね。オライリーテイストの今井美樹ってのは見てみたかった気はしますが、ま、冗談ですから。

A: Rubyの開発に関してですが、フリーのソフトウェアを開発する原動力ってのは何なんですか？
M: そうですねえ。やっぱりプログラミングは楽しいからじゃないかと思います。ソフトウェアの開発は「無から有を造り出す究極の創造活動」ではないかと思っています。それとフリーソフトウェアにはずいぶんお世話になっていますから、私がRubyを開発するのはある意味一種の恩返しではないかとも考えています。最近、「オープンソース」とかいう名前でフリーソフトウェアが注目されているので、かなりやりやすくなりましたね。

A: ムカつくこととか、困ることとかありますか？
M: 「お客さん」は困りますね。「動かんぞ、どーしてくれる」とか詰め寄られちゃうと、しみじみ「お金もらってるわけじゃないんだけどなぁ」とか思います。あと、「Windowsでインストールできないんですけど」ってメールをもらうのも困りますね。「私にもできません」とか答えちゃいそうになります。最近、Windows用インストーラのinstall shield版が出たので、そういうのは減りましたけど。

A: 最後に読者のみなさんにひとこと。
M: ここ数年「書籍はないのか」といわれ続けましたが、とうとう出すことができました。完璧な本とはいえないかもしれませんが、それなりに役に立つと思います。どうぞよろしく。

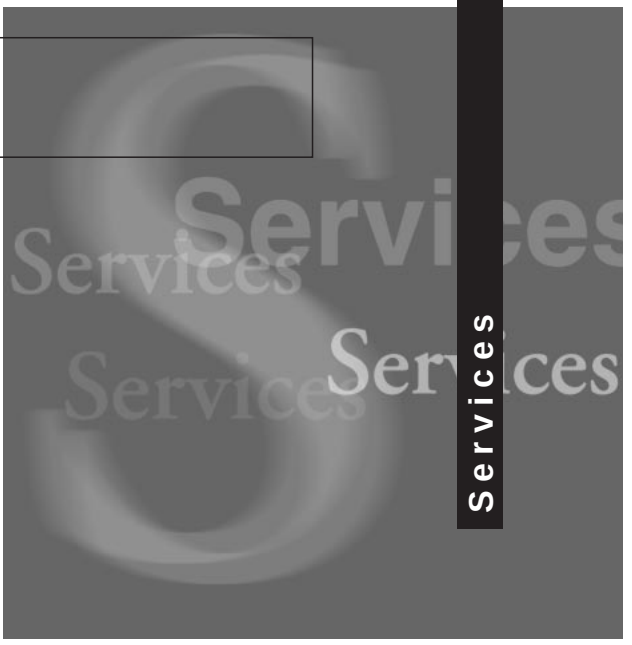
A: ありがとうございました。

今後もRubyに関する書籍の出版が予定されているそうです。書籍をはじめとする情報の充実により、Rubyのより一層の発展に期待したいところです。

ついに姿を現したRubyバイブル。表紙は「普通」だが、内容のほうはRubyの全貌がぎっしりと解説されており、かなり「尋常ではない」仕上がり。なお、本誌付属CD-ROMに第2章を丸ごとPDFファイルにしたものが収録されています。

『オブジェクト指向スクリプト言語Ruby』
まつもとゆきひろ・石塚圭樹 著
ISBN4-7561-3254-5
アスキー出版局刊
本体価格4000円

Sambaと闘う（第2回）



前回は、Linuxサーバを構築し、とりあえずSambaを起動するまでを紹介した。が、それだけではWindowsクライアントからは全くアクセスできない。今回はクライアントから共有リソースを使えるようにするための、「ユーザー登録」に取り組むことにしよう。

ユーザーの登録

文：梅原系
Text : Kei Umehara

さて、前回の作業の最後では、Windowsクライアントからのアクセスにチャレンジし、見事失敗した。実は、これで当たり前なのだ。というのも、前回の作業ではSambaをインストールしただけで、それを利用するユーザーの登録をまったく行っていないからだ。前回の最後では画面写真を2つ（NTとWin 9x）紹介したが、ネットワーク構成によって症状（エラーメッセージ）は千差万別なので、あまり気にしないで今回の作業に取りかかることにしよう。

もちろん、ユーザーの登録以外の原因、たとえばTCP/IPの設定やsmb.confの指定ミスによってSambaがアクセスできない可能性もある。しかし、まずは今回紹介する作業を終えてからでないと、何が問題なのかを判断するのも面倒というものだ。

ユーザー認証：Sambaの鬼門

Linux（UNIX系OS）とWindowsでは、システムの育ちがまったく違う。しかもWindowsの場合、OSのバージョンなどによって同じ機能でも異なる仕組みを採用していることが珍しくない。そのため、Sambaのような両者の橋渡しをするプログラムでは、2つのOSファミリー（のそれぞれのバージョン）をサポートするために、実に複雑な機能を用意しなくてはならない。

さらに話を複雑にしているのが、同じバージョンのOSでも、サービスパック（SP）を適用するとOSの挙動が変化する場合があることだ。

たとえばNT 4.0の場合、SP（サービスパック）3以前だと平文（＝暗号化しない）パスワードも有効だったのに対して、SP3以降では暗号化パスワードがデフォルトになり、平文パスワードを使うにはレジストリの書き換えが必要になるといった具合だ。

■想定環境

そんなわけで、今回の解説では動作環境について、以下のように限定することにした。

- **ワークグループによるネットワーク（ドメインは使用しない）**
- **複数のサブネットは想定しない**
- **Sambaマシンは1台**
- **使用するクライアントOSは次のいずれか**
  NT4.0+SP5（SP3以降を想定）
  Windows 98（Windows 95でも問題はないはずだが、筆者はチェックしていない）

要は、「非常に単純で、しかもあまり古いOSが含まれていない」ネットワークということだ。それ以外の環境については、いずれ解説したい。

■設定の方針

Sambaは非常にフレキシビリティの高いソフトウェアで、Sambaの仕組みと設定方法に通暁し、クライアントマシンを含めたマシン設定に奔走すれば、Windows 3.x（含むWindows for Workgroup）やLAN Managerクライアントなどといったマイクロソフト系OSの大部分サポートできる。しかし、よほど暇と能力を持て余している読者以外なら、そんなことに時間を費やしたくないと思うのが普通だろう（一生懸命環境設定しても、ソフトウェアがバージョンアップしたらまた大変な作業が待っているんだし）。

というわけで、Sambaをラクに使うためのポイントは次の2つだ。

・できる限り単純なネットワーク環境にする。

たとえば、クライアントOSの種類やSPを統一する。不必要にサブネットに分けない。…といったことだ。Windows系のOSはバージョンごと、SPごとにかなり挙動が違うので、それらを統一しておかないと、Samba側の設定が破綻をきたしやすくなる。

・クライアントの設定は、可能な限り変更しない。

ネットワークのセットアップでは、クライアントの設定変更を伴うと作業の手間が大幅に増える。システム管理者が、エンドユーザーのマシンをいじるのもなにかと面倒だ。マシンが2～3台位しかない小規模なLAN環境ならともかく、普通の環境では「クライアント環境はできる限りOSのデフォルトで使う」というポリシーを貫くことが大切になる。

Sambaのユーザー認証

くどいようだが、Sambaのユーザー認証は非常に複雑なので、そのすべてを理解するのにはかなりの勉強が必要になる。ここではその概要を簡単に紹介するにとどめておこう。

ユーザー認証（というか、アクセス権の制御）のツボは3つある。それは、「どのような単位で管理するか」、「パスワードは暗号化されているか」、そして「誰が認証を行うか」だ。

■どのような単位で管理するか

Windowsネットワークでは、OSによって2種類の管理単位が存在する。

・NT：ユーザー単位での管理

NTはUnixなどと同じように、ユーザー単位での認証が行われる。ファイルなどへのアクセスでは、そのユーザーにアクセス権があるかどうかチェックされる。

・Windows 9x：リソース単位での管理

Windows 9xでは、ユーザー認証という概念が非常に弱く、ネットワークでもリソース単位でアクセス権を管理している。つまり、それぞれのリソースに設定されたパスワードを知っているかどうかで、そのリソースにアクセスできるかどうかが決まる。

この2つの管理方式は、認証とアクセス権の制御だけが異なっており、実際のファイルアクセスなどでは同じSMBプロトコルが利用されている（細部はいろいろ違っているようだが）。また、Windows 9xにはログオン時に入力したユーザー名とパスワードを使って、NTのユーザー単位の認証も受けられるようになっている。

■パスワードは暗号化されているか

パスワードの扱いでは、認証時、ネットワークを流れる際に暗号化されているかどうかがポイントになる。

・平文パスワード

初期のWindowsネットワークでは、パスワードは暗号化されておらず、パスワードが平文のままネットワーク上で転送される。この方式はUNIXシステムとの相性はいいのだが、ネットワーク、パケットモニタなどを使うことで、簡単に第三者が他人のパスワードを盗むことができるという欠点がある。NT 4.0ではSP3以降、Windows 95ではOSR2から、Windows 98では最初のリリースから、デフォルトでは平文パスワードが使えなくなっている。

・暗号化パスワード

第三者によるパスワードの窃盗を防ぐために、現在のWindows（NTと9xの両方）で使われているのが暗号化パスワードと呼ばれる機能だ。この方式は、パスワードそのものではなく、「サーバから渡されたデータを、パスワードを使って暗号化して送り返す」という方法（チャレンジ&レスポンス方式と呼ばれる）で、ネットワーク上を流れるデータを盗聴（？）するだけでは、パスワードを推測するのは不可能に近い。

現在リリースされているNT（4.0＋SP5またはWindows 98）のどちらも、デフォルトでは暗号化パスワード方式しか使えないようになっている。「クライアントの設定は極力変更しない」という原則からすれば、これから立ち上げるSambaサーバでは暗号化パスワードを使ったほうがいいだろう。

■誰が認証を行うか

クライアントから渡されたパスワードは、設定に応じてどこで認証されるかが決まる。

・OSの認証システムを使う（図1）

平文パスワードの場合、Sambaが稼働しているOSのユーザー認証を使うことができる。パスワードの管理が一元化されるので、管理者としても面倒はない。ただし、この方式は暗号化パスワードを利用する場合には利用できない。またクライアント側のWindowsでは、レジストリの書き換えが必要になる。

・Samba自身の認証システムを使う（図2）

暗号化パスワードを使う場合、Linuxの認証システムは直接使えない。そのためSambaに、NTなどと同等の認証システムが用意されている。ワークグループ構成のLANでは、もっとも現実的な方法だろう。

・NTサーバのドメインコントローラの認証システムを使う（図3）

この方式では、Samba自身はパスワードを管理せず、クライアントの認証をドメインコントローラ（NTサーバ）に依頼する。すでにNTドメインでLANを構築している環境にSambaを追加する場合、もっともラクなアプローチだ。ただし、この場合にはSambaマシンにも、NTサーバへのアクセスライセンス（CAL）が必要になるので注意してほしい。

Windowsクライアントにおいて暗号化パスワードが標準になりつつある現在、Sambaのコンフィギュレーションもそれに対応させたほうがいい。ということで、NTドメイン環境ではドメインコントローラによる認証を、ワークグループ環境ではSambaによる認証を使うことになるだろう。

ここでは、これまでに説明した「ユーザー単位での認証」、「暗号化パスワードの利用」を前提に、「Samba自身による認証」を例に紹介しよう。

作業の実際

さて、方針さえ決まれば、ユーザー登録の作業自体はそれほど難しいものではない。sambaの設定とはちょっと異なるが、ここではLinuxへのユーザー登録も含めて、ユーザーの登録作業をひと通り行ってみることにしよう。

■adduser：Linuxへのユーザー登録

Sambaを使ってLinux（UNIX）上のファイルをアクセ

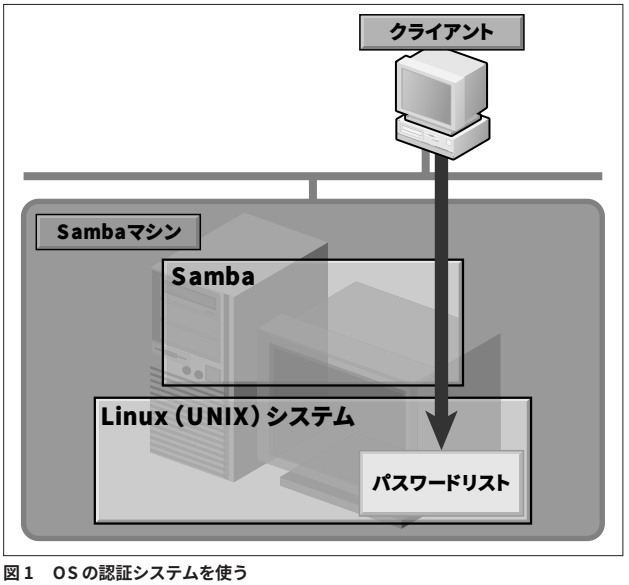


図1　OSの認証システムを使う
クライアントから渡された（平文）パスワードは、Linuxのパスワード認証機構によってチェックされる。

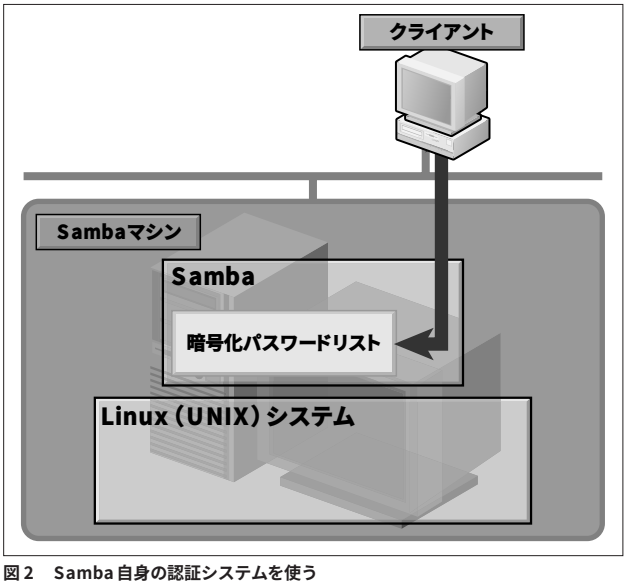


図2　Samba自身の認証システムを使う
クライアントから渡された（暗号化）パスワードは、Sambaが管理するパスワードを使って、Samba自身によって認証される。

スする場合には、認証されたユーザーの権限によって、個々のファイル（など）へのアクセスが制御される（そうする必要がある）。そのためSambaマシンには、Windowsクライアントのユーザーに対応するユーザーが登録されていなければならない。もし、Linuxにユーザーを登録していない場合には、まずはユーザーアカウントの作成から始めよう。

新しいユーザーの登録には、adduser（またはuseradd）コマンドを使う。/etc/passwdファイルを直接編集してもよいが、ミスを犯しやすいし、作業に余計なステップが要求されるので、ここは素直にadduserを使うことにしよう。rootとしてログイン（もちろん、suでrootになってもいい）して、以下のように実行しよう。

```
/usr/sbin/adduser -g group -p password username
```

usernameには新しいユーザー名を、groupにはグループ名を指定する。-gオプションを使ってグループ名を指定しないと、新しいユーザー名と同名のグループ名が新規に作成され、そのグループによって登録されることになる。パスワードを指定しないと、とりあえずそのユーザーのパスワードは空になる。その場合は、あとでpasswdコマンドを使って設定すればいい。

ユーザー名には、原則としてWindowsクライアントのユーザー名と同じものを使用する。ただし、Linuxでは、空白を含む名前や、漢字の名前はトラブルの元になるので、半角英数字によるログイン名を付けるようにしよう。

ここでは、新規ユーザーumeharaを、グループusers（このグループは、通常のLinuxだとデフォルトで作成されている）で作成してみよう。

```
# /usr/sbin/adduser -g users -p my_passwd umehara
```

adduserは新規ユーザーを/etc/passwdに登録するとともに、ユーザーのホームディレクトリの作成、Xなどの初期化ファイルのコピーなどを自動的に行う。

adduser以外にも、Xなどで動作するGUIベースのユーザー管理ツールが使えることもあるので、それらを使ってもいいだろう。

■smbpasswd：Sambaへのユーザー登録

Linuxへのユーザー登録が終わったら、次はSambaへの登録だ。これには、smbpasswdを使う。

```
smbpasswd -a username
```

smbpasswdは、Sambaで管理しているパスワードを変更するためのコマンドだが、-aオプションを指定して実行すると新規にユーザーを登録できるようになっている。コマンドを実行し、パスワード（Windowsクライアントのパスワードと同じもの）を入力すれば、登録完了だ。

ここで注意を1つ。smbpasswdで入力したパスワードは/etc/smbpasswd（smb.confの指定によって、ファイル名は変化する。ディストリビューションによってもディレクトリが異なることがある）に格納されるが、これは「符号化されたパスワード」であって、「暗号化されたパスワード」ではない。つまり、

Sambaのソースを読みこなせる人間にとっては、smbpasswdは平文で書かれたメモと同じ

なのだ。

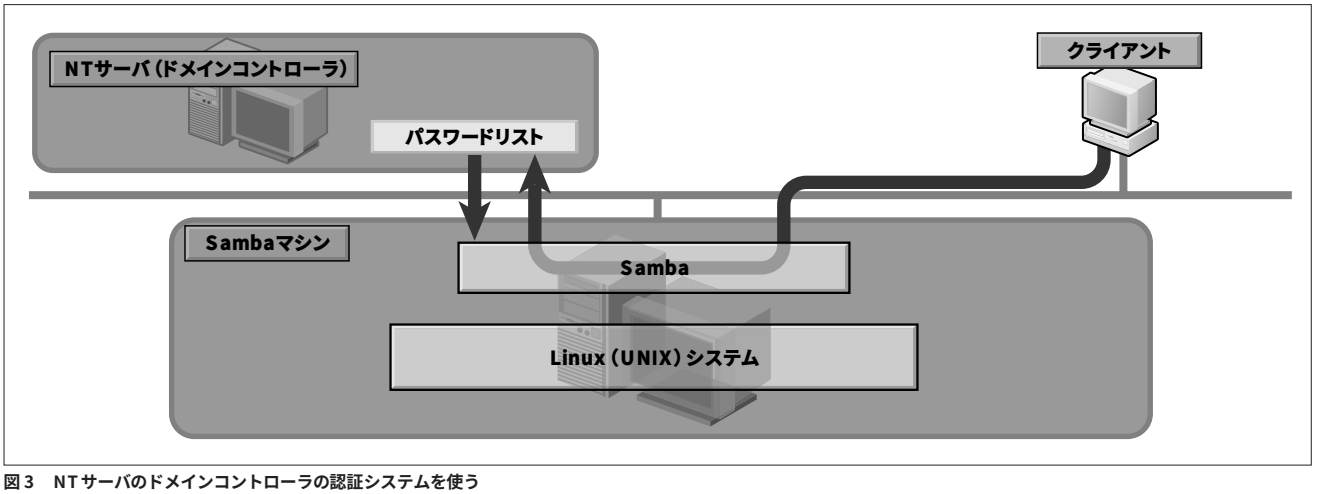


図3　NTサーバのドメインコントローラの認証システムを使う
クライアントから渡されたパスワードはドメインサーバによって認証され、その結果がSambaに返される。

内容を少しでも読まれないように、smbpasswdは/etc/smbpasswdファイルを、ルート以外のユーザーには一切アクセスできないようにしているが、ちょっとした不注意で、Sambaユーザー全員のパスワードが盗まれる可能性があるということに、十分留意してほしい。

以上の2ステップで、Sambaへのユーザー登録は終わりだ。改めて、WindowsクライアントからSambaをアクセスしてみてほしい。他の設定が間違っていなければ、ネットワークコンピュータの1つとしてSambaサーバが見えるようになっているはずだ。

■mksmbpasswd.sh：Linuxユーザーの一括登録

上の例ではsmbpasswdコマンドを使って1ユーザーを登録したが、何十、何百ものユーザーがいる環境では、これを何回も実行するのは面倒だ。そのような場合には、mksmbpasswd.shというシェルスクリプトを利用するのが便利だ。ルートユーザーとしてログインして、次のようにすれば、Linuxに登録されているユーザー全員を、一括して/etc/smbpasswdに登録することができる。ディストリビューションやsmb.confの設定によってsmbpasswdファイルの格納ディレクトリは変化するので、その点には注意してほしい。

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

2行目のchmodコマンドは、ルート以外のユーザーが/etc/smbpasswdにアクセスできなくするためのおまじないだ。本当は、ファイルを格納するディレクトリ自体をルート以外からアクセスできないようにしたいのだが、多くのディストリビューションでは/etcディレクトリに格納してしまっているのでしかたない。

このコマンドでは、passwdファイルに登録されているすべてのユーザーをSambaユーザーとして登録してしまう。daemonやbinといった、疑似ユーザーまで登録されてしまうのだ。そこでmksmbpasswd.shの実行後に、viなどのエディタを使って、実ユーザー以外の行を削除しておこう。

このようにして作成したsmbpasswdファイルには、まだ個々のユーザーのパスワードは設定されておらず、Sambaでもアクセスを拒否する。ただしエントリさえ用意されていれば、個々のパスワードはユーザー本人がsmbpasswdコマンドを使って設定できるので、その旨をエンドユーザーに周知徹底すればいい。

■smbusers：ユーザー名の変換

ネットワークによっては、Samba（Windowsネットワーク）のユーザー名と、Linuxのログイン名が一致しない（させられない）場合もあるだろう。そのような場合には、/etc/smbusersというファイル（smb.confの指定によって、ファイル名は変化する）によって、ユーザー名のマッピング（変換）が行える。これにより、たとえば、Windows環境で“Umehara”というログオン名のユーザーからのアクセスを、Linuxの“kei-u”というユーザーからのものと見なす（ファイルのオーナーやアクセス制御に“kei-u”を適用する）ということが可能になるのだ。

Red Hatに含まれているSambaをインストールすると、次のような内容のsmbusersファイルが（デフォルトで）作成されている。

```
# Unix_name = SMB_name1 SMB_name2 ...
root = administrator admin
nobody = guest pcguest smbguest
```

指定方法は、最初にLinuxのユーザー名、‘=’の後ろにはSMB名（Windowsクライアントからの名前）のリストを指定するだけだ。WindowsクライアントとLinuxとで、ログイン名を共通化している場合には、このファイルによる指定は一切必要ない。

次回に向けて

ここまでの作業で、ようやくSambaが公開するリソース（前回紹介したsmb.confを使っているのなら、ユーザーのホームディレクトリと/tmp、プリンタ）をWindowsクライアントからアクセスできるようになった。ファイルのアクセスや書き込みがうまくいくか、テストしてみてほしい。問題がないようなら、ベンチマークなどで性能を測ったり、ストレステストにチャレンジしたりするのもいいだろう。

次回は、新しいリソースを公開する方法の紹介か、アクセスできない場合のトラブルシューティングに取り組む予定だ。

ニューテクノロジー From SGI

オープンソースに提供されるSGIのテクノロジー

Knowledge

第2回　OpenGLリアルタイム 3Dグラフィックス環境への誘い

文：**鈴木大輔**
Text : Daisuke Suzuki

日本SGI株式会社オペレーティングシステム担当
日本Linux協会理事
ds-suzuki@nsg.sgi.com
daisuke@linux.or.jp

先月から始まった連載だが、1カ月たった今も残念ながらXFSは公開されていない。しかし、高性能なファイルシステムに対する要求は日に日に多くなっていることも事実である。

ここへ最近になってext3ファイルシステムという新しいファイルシステムが登場した。これは以前から計画されていたジャーナリング機能を盛り込んだファイルシステムである。長い間待たれていたext3ファイルシステムではあるが、実際公開されたものはext2ファイルシステムをそのまま使い、ジャーナリングの機能だけを拡張したものであった。ジャーナルファイルはext2上の1ファイルとして書き込まれ、ext3ファイルシステムとして利用した場合には仮想的にジャーナルファイルシステムとなるものである。XFSの持っているようなスケーラビリティやハイパフォーマンスな特徴は一切持っていないため、どちらかというとext2の機能強化というべきものだろう。

今回は、Linuxにおけるグラフィックス環境についてSGIの取り組みを紹介しよう。

Linuxとグラフィックス

現在Linuxは、サーバシステムとしてはその安定性と頑強さが評価され、インターネット、イントラネットのシステムを中心に使われるようになってきた。

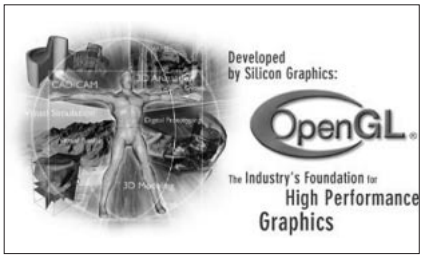
しかしながら、デスクトップ用途として見てみると、グラフィックスアプリケーションは、まだ少ないという状況にある。たとえば2Dグラフィックスのアプリケーションはいくつか存在するが、3Dグラフィックスのアプリケーションについては、ここ数カ月でようやく出始めた程度で、まだまだ非常に数も種類も少ない。この原因はなんだろうか？

まず考えられるのは、3Dグラフィックスのハードウェアサポートがないことだろう。さらに、業界標準のグラフィックスAPIであるOpenGLやさらに上位APIなどへの正式サポートがないことも原因だろう。この2つが要因となり、アプリケーションベンダーによるLinuxへのグラフィックスソフトウェアの移植が遅れ、結果としてLinuxにおける3Dグラフィックス環境が発展しないのではないだろうか。

この問題をユーザー、アプリケーションベンダー、ハードウェアベンダーの3つの視点から見てみると次のようになる。

・アプリケーションベンダー

3DグラフィックスアプリケーションをLinuxに移植したいが、Linuxでハードウェアアクセラレーションが可能なマシンがない。また、グラフィックスAPIもサポートされていない。



・ハードウェアベンダー

アプリケーションベンダーからの要求はあるが、ユーザーが少ないためサポート対象になっていない。また、ハードウェアアクセラレーションドライバ作成のための枠組みもない。

・ユーザー

3Dグラフィックスアプリケーションはほしいがアプリケーションがないため、グラフィックスはLinuxではあきらめかけている。

このように三者がデッドロックのような状態になってしまっているのが、現在の状況だと思われる。この状況に対してSGIとして何ができるか、何をすればよいのかを考えた答えが、今回紹介するOpenGL/GLXとPerformerなのである。

Linuxの3D グラフィックスの現状

現在Linuxでグラフィックス、特にVR（バーチャルリアリティ）、ビジュアルシュミレーション、CAD、エンターテイメントなどの分野では非常に普及率が低い。これは先ほど述べたようにデバイスへの対応、メーカーのサポートがないことが原因であろう。

商用OSを見てみると、ほとんどのOSがOpenGLのハードウェアアクセラレーションに対応しているが、Linuxはどうだろうか？　Xi Graphics社やMetro Link社が商用Xサーバとともに OpenGLの実装を提供しているが、今もなおソフトウェアのみによる実装で、ハードウェアアクセラレーションはまだ使用できない（次期バージョンとしてアクセラレーションサポートも入っているが、まだβバージョンのようである）。フリーソフトウェアとしてMesaというOpenGL互換のライブラリがあるが、ハードウェアアクセラレーションは、ごく一部のグラフィックスチップしかサポートされていない。

アクセラレーションのあるチップもハードウェアベンダー独自のライブラリを使用しており、X Window Systemに統合されているわけではない。もちろん、メーカーによるサポートというものもないのが現状である。

```
Mesa:          http://www.mesa3d.org/
Xi Graphics:   http://www.xig.com/
Metro Link:    http://www.metrolink
               .com/
```

SGIとLinux グラフィックス

このような状況でSGIがどのような貢献を行って来たかを紹介しよう。もともとSGIではIRIS GLというグラフィックライブラリを使用していた。これはSGI独自の仕様でSGIのハードウェアでのみ使えるものであったが、これをオープンアーキテクチャにしたのがOpenGLである。OpenGLはそれまでの独自な仕様をやめ、オープンな仕様策定機関を設けオープンなAPIとして提供してきた（**画面1**）。この結果、現在では3DグラフィックスAPIとしてOpenGLは業界標準のものとなった。

しかし、オープンソース・コミュニティにおいてOpenGLをきちんとサポートできるようにするためには、基本的な部分からしっかりした実装をしなければならない。また、ハードウェアベンダーがこの実装を容易に提供できるようにしなければならない。

これに対してSGIは1999年2月にGLXのソースコードをオープンソースコミュニティに提供した。GLXはあまり馴染みがないかも知れないが、X Window SystemのOpenGL eXtentionで、OpenGLとX Window Systemの橋渡しをする部分であり、Xを利用したどのようなOpenGLの実装でも必要となるものである。

現在SGIはこのGLXをXFree86 4.0へ盛り込むためにPrecision Insight社とともに技術協力を行っている。これによってLinuxをはじめとするXFree86を利用しているOSでOpenGLが利用できるようになるはずである。このGLXのソース公開は主にXFree86のハードウェアドライバを開発している人向けであり、このGLXの仕組みを利用することで、近い将来にXFree86でハードウェアアクセラレーションのかかった3Dグラフィックスが使えるようになるだろう。

Precision Insight社はこのGLXとともに動作するDRI（Direct Rendering Infrastructure）という仕組みの実装を進めており、SGIも技術アドバイザーとして協力をしている。このDRIは3Dのデータをグラフィックスハードウェアに直接送り込むための高速な経路を提供する仕組みである（**図1**）。

このDRIが利用できるようになると、ハードウェアベンダーが自社製品用の高速ハードウェアアクセラレーションドライバを容易に開発できるようになる。さらにOpenGLを利用するアプリ

画面1　OpenGLのWebページ
http://www.opengl.org/

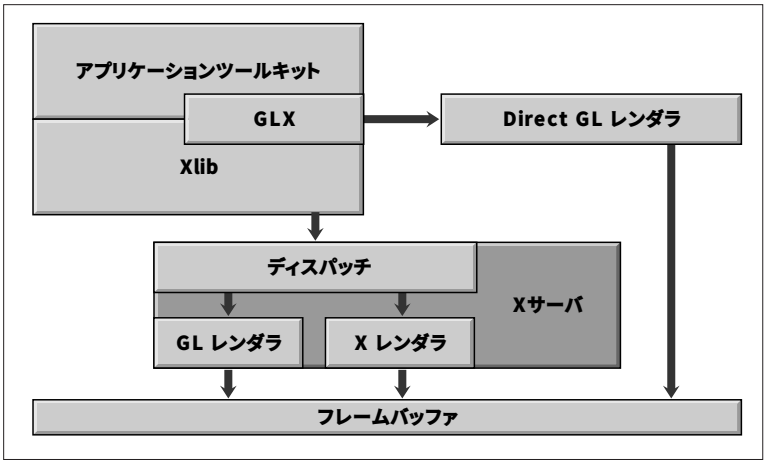
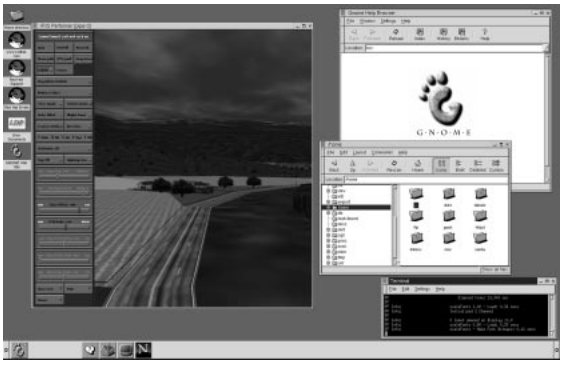


図1 OpenGLとGLX、DRIの仕組み



画面2　Linuxで動作しているMongoose

“毒蛇に対する戦闘能力で知られるマングースは、その「スピード」、「明敏さ」、絶妙の「タイミング」、そして「分厚い毛皮」のために、いつも勝利を手にすることができる”（Mongoose 開発チームスローガン）

ケーションはハードウェアの能力を最大限に利用することができるようになる。

現在XFree86 4.0のスナップショットであるXFree86 3.9.16ではDRIを使ったサンプル実装が提供されており、4.0リリース時には多くのサンプルドライバが提供される予定である。

また、各種ドライバ開発者によるDRIの評価も積極的に行われている。これらのDRIのソースコードを見ることで、GLXとMesaの統合の技術を知ることができ、ほかのOpenGLの実装がLinuxに移植される場合の重要な情報元となるだろう。SGIが長年培って来たOpenGL実装のノウハウを効率良く利用できることは非常に好ましいことではないだろうか。

こうしてSGIはOpenGLとハードウェアアクセラレーションといった基本的な部分への貢献をしたが、この上でさらにVRといった3Dグラフィックスアプリケーションが Linuxで普及するためにSGIは何が提供できるだろうか。その答えのひとつがMongoose（IRIS Performer）なのである（**画面2**）。

バーチャルリアリティ（VR）業界において最も優れた開発ツールキットとして知られているIRIS Performerの Linux版（開発コード名: Mongoose）が1999年9月20日にβ公開された。すでに米国SGIのWebサイトからIRIS Performer 2.3 Beta 1.0が評価用として入手できるようになっている（**画面3**）。

Mongooseは高速なインタラクティブレンダリング環境を少ないコストで実現し、VRシステムのスケーラビリティを高める目的で開発された。また、同時にLinux上での初のハイレベルVR開発環境として、業界における注目度は非常に高い。

Performer

IRIS Performerは、リアルタイム・インタラクティブ・グラフィックスアプリケーションを開発するための高速3Dレンダリングツールキットであり、この種のツールキットの先駆者的な役割を果たしてきた。

Performerを使うことで以下のような複雑なアプリケーションの開発が劇的に単純化できる。

- ビジュアルシュミレーション
- シュミレーションベース設計
- バーチャルリアリティ
- インタラクティブエンターテイメント
- 放送用ビデオ
- CAD
- 建築ウォークスルー

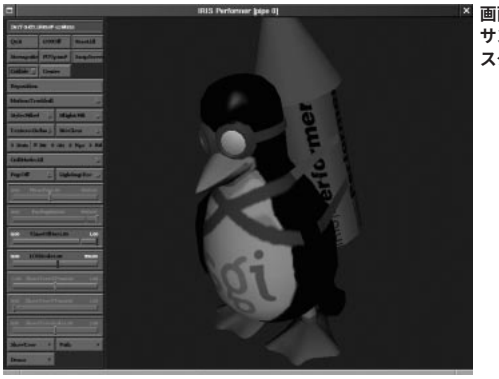
Linux版の登場によりこのようなアプリケーションのLinuxプラットフォームへの移植が容易になるだろう。また、同時に、SGI製品でしか動かなかったものがLinuxでも動くようになり、VRシステムのスケーラビリティやポータビリティがさらに広がっていくと考えられる。

ここでMongooseの特徴について簡単にまとめておこう。Mongooseは上で述べたようにIRIS PerformerのLinuxへの移植であるが、すべての機能が移植済みというわけではない。

画面3　PerformerのWebページ
http://www.sgi.com/software/performer/

表1 Mongoose（Linux版Performer）の特徴		
Performer for IRIX完全互換	IRIS Performer 2.2.x API互換	アプリケーションの移植作業は必要最小限となる
	IRIX版Performerと共通のソースコード	機能拡張、バグ修正、変更などがIRIX版とLinux版で完全に同期している
Beta 1.0で移植済みの機能	ライブラリ	libpf、libpr、libpfutil、libpfui、libpfuiD、libpfdu
	ファイルローダの大部分	
	サンプルプログラムとperfly	
今後サポートする予定の機能	マルチプロセッシング	ForkベースのCull、Draw、Insect、Dbase、X入力イベント
現在実現していない機能	マルチパイプ	グラフィックスのアクセラレーションを複数のチップで並行して行う機能
	マルチウインドウ	
	IRIX独自機能	
	InfiniteReality独自機能	Onyx2 IR独自の機能

画面4
サンプルデータによる
スクリーンショット



画面5
サンプルデータによる
スクリーンショット

Beta 1.0である現時点では**表1**のような特徴を持っている。

Mongooseの動作環境

IRIS Performer 2.3 Beta 1.0 for Linuxは、Intel Pentium IIベースのPC上でRed Hat Linux 6.0で開発、テストされている。Beta 1.0で提供されているライブラリやサンプルプログラムを使用するには、Red Hat Linux 6.0標準のパッケージ以外に以下のものが必要である。

- `libMesaGL.so.3`、`libMesaGLU.so.3`
  `Mesa-3.1beta1`以上
- `libXm`
  `Lesstif-current-1999.01.28`以降

このように現時点ではMesaを使用しているため、ハードウェアアクセラレーションを使うにはMesaが対応しているグラフィックスカードを使う必要がある。詳しくはMesaのWebサイトを参照してほしいが、たとえば、3DfxのチップやnVIDIAのチップを使うことでハードウェアアクセラレーションがかかった高速なVR環境を体験できるだろう。

Performerにはサンプルプログラムとしてperflyを使ったデモがあるため、これを体験してみるといいだろう。デモデータは以下のところにある。

```
/usr/share/Performer/data/
```

このデータをperflyに渡して体験してみてほしい。たとえばOnyxのデモとして有名なPerformer Townなら、

```
$ perfly -W800,600 town_ogl.perfly
```

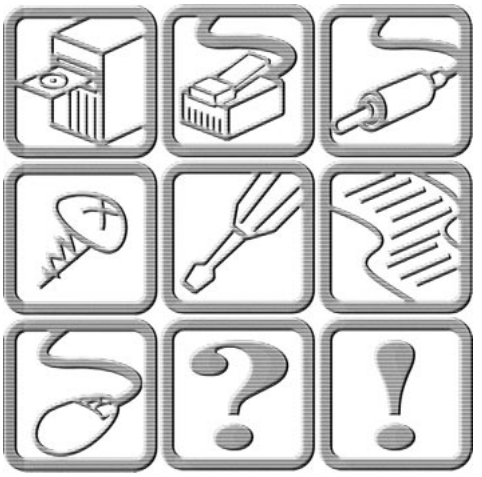
とすることで、ある街を自動車で走り回るデモが体験できる。なお、この-Wというオプションはウインドウの大きさの指定である。その他、ウインドウ左に出ているメニューによってテクスチャを張ったりフォグをかけたりすることができるので、試してみてほしい。

またいくつかのサンプルデータを読み込ませることで、**画面4**、**画面5**のようなデモも見ることができる。

Linuxではまだまだ使われにくいグラフィックスではあるが、ここ最近いくつかのアプリケーションも登場している。CADソフトウェアやモデリングソフトウェア、レンダリングソフトウェアなどが登場し始めた。

これで、OpenGL、GLX、Performerが正式に登場すれば、リアルタイム3Dグラフィックスアプリケーションの多くがLinuxで動作するようになる日も近いだろう。SGIは今後もグラフィックスを推進していく予定であり、Linuxのグラフィックスでのイニシアティブをとっていくつもりである。Linuxとグラフィックスは、これからますます目が離せないのではないだろうか。

Try & Try [trái]



Linuxシステムのステータス情報

文：政久忠由
Text : Tadayoshi Masahisa

最近ガーデニングに凝っている。といっても、うちの小さなベランダに箱庭を作ったわけではない。一日のうちで一番多く眺めているもの、そうPCのデスクトップを庭に見立ててガーデニングを行っているのである。

突然、庭を造ろうと思ったのは、直接的には、最近読んだ小説の神秘的で秘密めいた庭園の話が心に残っていたのと、美しい庭が登場するCMやTV番組に触発されたからだ（間接的な理由のほうが影響が大きいのだけれども、ちょっとそれらはうまく言葉にできない）。

幽玄、余情、無常…、さまざまな言葉でも表現される庭は、造る人の心を表し、見る人の心を映し出すものとして知られている。僕がまだ幼いころふれていた実家の庭には、父親の無節操さを反映してか、鯉の泳ぐ池（その後、水漏れのため空池となる）に松やら柊やら椿やら、名前のわからない（僕には）さまざまな木々が所狭しと植えられ、まるでジャングルのようだった。また庭には木々のほかに、どこでもらって来たのかわからないデカイ石や踏み石が乱雑に配置されていた。そのころの僕は、花咲く木々にあつまる蜂をはじめとする虫がきらいで木々の植えられている部分の庭はあまり好きではなかったが、石が配置されている部分は格好の遊び場で面白かった。結構な月日、この庭を見ていない。今見ると果たしてどのような印象を受けるのだろう。

本来であれば、自然にふれ、仕事の疲れを癒し、心を洗い、そしてもの思いにふける場所として庭を造るべきなのだが、そこはそれ、常識にとらわれることもないだろうと思って、デスクトップでの庭造りに挑戦してみた次第である。

デスクトップガーデンは今のところ未完成なのだが、対称性の美しい英国式庭園と神秘的な空中庭園が目下のところのテーマだ。『禅』を意識した日本庭園はちょっと難しいので後回し（宇宙空間をイメージさせる庭っていいよなぁ）。現段階では、色やアイコン、壁紙など、手っ取り早くできるところをスキャンしたデータなどを使って変更している。全体の構図はさまざまな庭園の設計図などを参考に思案中だ。スクリーンセーバもスケルトン部分はある程度完成したし、各部品リソースも着々と出来上がっている。だが各部品をどのように構成するかが、難しく大きな問題として残っている。

未完成の段階なので、空間が発する力のようなものは今ひとつ足りない気がするが、きっと煌きのある素敵なものになる予感がする。だって僕が作っているのだから…。

完成したら、ページの隅っこで紹介しようと思う（それとも、ありきたりだけど、広く募集して自分自慢デスクトップ紹介のページをでっち上げるかな？）。

前回の補足

本題に入る前に、まずは前回の補足。カーネル2.4では、バッファキャッシュメモリの未保存のデータをディスクに書き出すためのbdflushとして知られるデーモン（プログ

ラムは通常/sbin/update）がカーネルスレッドとして実行されることを報告した。そこで、今まで実行していた/sbin/updateはどないすんねんという話をした。

従来/sbin/updateは1回起動するだけで、実際にバッファの書き出し処理を行うルーチンとしてのbdflushと、そのbdflushの機能を定期的に呼び起こすupdateという2つのデーモンプログラムとして機能していた。

前回、試用したカーネル2.3.18では、カーネルスレッドとして実行されるbdflushだけが実装されていて、その中で“interruptible_sleep_on_timeout(&bdflush_wait, 5＊HZ);”としてインターバル（5＊HZは5秒ってこと）が指定されているのでこれだけで済ますのかなと思っていたのだが、カーネル2.3.23のプレリリースを見ると、updateルーチンが追加され、カーネルスレッドとして実行されるようになっている。これは間違ってKILLしてしまうことを防ぐ目的もあるようだ。また、従来の設定を変更しなくても済むようにユーザースペースで実行された/sbin/updateは、最初にバッファがフラッシュされるタイミングで終了させるようにもなっている。つまりユーザーは、これらに関しては何も気にしなくてもよいってことだ。たとえば、僕の環境のようにinittabにある“ud::once:/sbin/update”はそのままにしておいても問題ないのだ。まぁ、無効にしたけりゃ、してもよいのだけれども（そのほうがすっきりするし）。ちなみに筆者の環境のカーネル2.3.23で実行されるカーネルスレッドは次のようになっている。

```
2 ?        SW     0:00 [kapmd]
3 ?        SW     0:00 [kswapd]
4 ?        SW     0:00 [kflushd]
5 ?        SW     0:00 [kupdate]
```

ところで、古くからUNIX系OSに慣れ親しんでいるユーザーは知っていると思うけど、どうしてページアウトやバッファキャッシュの操作をユーザースペースのデーモンで行っているのかというは、最近のOS（Windowsとか）しか知らないユーザーにとっては、素朴な疑問のひとつだと思う。その答えをごく簡単に説明すると、それらのOSのカーネルは機能を提供するわけだが、基本的にその各機能は自立的に実行されるわけではなく、何かしらに呼ばれてはじめて実行されるものだからだ。つまり、カーネルにバッファキャッシュをフラッシュする機能があったとしても、その機能は何かしらに呼ばれないと実行されないのである。では、その呼び出すものは何かというと、スケジュールの対象であるプロセスだ（ハードウェア割り込みなどで実行される機能もあるけど、それらを説明しはじめると難しくなるのでここでは省略）。したがって、バッファキャッシュをフラッシュするにも、ユーザースペースのデーモンやカーネルスレッドなどから該当する機能を呼び出す必要があるのだ。なお、この説明では大体のイメージしかつかめないと思うので、ちゃんと知りたい人は、きちんとした書籍などを読んで頂戴。

Linuxでは、従来ユーザースペースデーモンとして実行されていた機能の多くがカーネルスレッドとして実装されてきている。上記の例では、パワーマネージメント、ページアウト、バッファキャッシュを操作する機能がカーネルスレッドとして実行されているのがわかると思う。これ以外にも、PCカードサービスやNFSの機能などもカーネルスレッドになっている。ユーザースペースデーモンだったものをカーネルスレッドにすることには動作スピード面などでメリットも大きいが、反面、システムが保護されないという致命的なデメリットもあるので、一概にすべてカーネルスレッドにしちゃえってわけにはいかない。常識的には、ソースコードの信頼性を確保しやすいシンプルな機能のみがカーネルスレッドとしても問題ないと判断できるわけで、それ以外は各人の要求いかんということになる。そんなわけで、両方のバージョンが用意されていて、ユーザーが選択できるようになっているといいかもしれない。カーネルスレッドとして実行するSQLサーバサービスって、やっぱり、まずい？

procファイルシステム

インターネットサーバ、特にキャッシュサーバとして利用しようとした場合、オープンしたい（すべき）ファイルの数がそのシステムのファイルディスクリプタ数を超えちゃって困ったなんてことは、よく聞いた話だ。これはハードコーディング、つまりソースコードの段階であらかじめ値が設定されているために生じる問題で、解決するには設定を変更してカーネルをリコンパイルする必要があった。

システムは、さまざまなハードウェア環境やその使用目的に適応できるように、パラメータでもって各機能の挙動を決定するようになっており、また円滑な動作や消費メモリを低く抑えるためにリソース（先ほどのファイルディスクリプタ数やプロセスの数など）の上限なども設定している。

従来、システムの扱うリソースの上限や動作を決定する

各種パラメータは、カーネルの構築時にスタティックに設定されることが多かったが、最近の傾向としてはシステムの初期化時にハードウェアをチェックして、それに合わせて調整したり、動作中、動的に自動または手動で変更したりできるようになっている。たとえばLinuxでは、搭載物理メモリサイズに合わせてバッファキャッシュやページキャッシュの設定を調整したり、先ほどのファイルディスクリプタ数の上限も動作中に変更したりできるようになっている。そのため、古臭いUNIXシステムでは、カーネルのパラメータをちょっと調整したり、修正したりするには、リブートやカーネルのリコンパイルが必要になることが多かったが、Linuxをはじめとする多くのシステムでは、その必要性がかなり軽減されるようになっている。

では、自動的に調整された設定は完璧か、というとそうとは限らない。それなりに問題ないデフォルト値が設定されているだけだ。そんなもの動作中に動的に判断して調整すればよいではないか、という話もあるが、その判断にCPUリソースを消費しすぎては元も子もないし、かといってシンプルな解析方法では、やはりそれなりの結果でしかない。ただ一般的なユーザーにとっては、それなりであるデフォルトのバランスのとれた設定で何の問題もないし、逆に中途半端に調整すると全体のバランスが崩れてしまうだけなので、むしろ触らないほうがいいかもしれない。と、まぁちょっと注意した上で、Linuxシステムに実装されているカーネルやプロセスのステータスやカーネルのデフォルトパラメータ設定を変更できる/procファイルシステムについて見ていくことにしよう。

カーネルパラメータを調整する

Linuxカーネルでカーネルパラメータを変更するには、ソースコードを直接修正してリコンパイルする以外に、ブート時にカーネルに引数として渡す方法と、/procファイルシステムを利用する方法がある。

■ブートパラメータ

ブート時にカーネルにパラメータを渡す方法は、主にハードウェア関連のパラメータを変更するために利用される。また、“init=”として最初に実行するプログラム（ランレベルではない）を指定したり、“root=”としてマウントするルートデバイスを指定したりできるようになっている。ランレベルは単に数字などを指定するだけだ。どのようなものが指定できるかは、/usr/src/linux/Documentation/kernel-parameters.txtを参照してほしい。でもこのファイルには、各ハードウェア依存の設定は記述されていないので、それらに関しては、それぞれのハードウェアのドキュメントを見つけるか、対応するソースコードを覗かなきゃなんない。結構面倒だが、指定する必要があるのは問題が生じた場合だけなので、ここではそんな機能があると記憶しておくことにしよう。

どんな風に指定するかというと、LILOの場合は“boot:”が表示された時に次のように入力すればよい。以下の例ではルートデバイスに/dev/hdb1、ランレベル1を指定している。

```
LILO boot: ラベル名（例：Linux） 1 root=/dev/hdb1
```

/procファイルシステム

では本題の/procファイルシステムを見てみることにしよう。Linuxに実装された/procファイルシステムは、システムとプロセスのステータスについての情報を提供するもので、そのいくつかは変更できるようになっている。このファイルシステムが有効になっていれば、/proc以下に各機能に応じたツリーが構成されるようになっている。実際に有効にするには、カーネルで/procファイルシステムをサポートしていて、/etc/fstabで次のように記述されている必要がある。

```
none /proc proc defaults 0 0
```

/procファイルシステムが有効になっていれば、mountコマンドを実行すると次のようにマウントされているのがわかる。

```
none on /proc type proc (rw)
```

/procファイルシステムの場合、ファイルシステムといっても実体がそこにあるというか、ディスク上のスペースを消費しているわけではない。/procファイルシステムで提供される情報は、アクセスされた時にカーネルによってその場で生成されるようになっている。ファイルに見立てられている各機能は、ディレクトリでカテゴリ分けされ、cat、more、lessなどでステータスを参照することができる（バージョンの古いlessでは読めないかも）。

/procファイルシステムは、カーネルの構成で有効、無効の選択ができるようになっている。有効にすることで、

67Kバイト程度消費カーネルメモリが増えるようだが、得られる有益な情報に比べたら大したことはないので、通常は有効になっているはずだし、特に理由がない限り有効にすべきである。

さて、実際に/procディレクトリを見ていくことにしよう。操作は、普通のファイルシステムと同様に、cd、lsコマンドで行い、前述のようにcatやmoreでファイルを参照することでステータスを取得することができる。また変更可能なものは、echoコマンドなどで修正できるようになっている。

/procファイルシステムでは、システムのすべてのステータス情報を提供しているといっても過言ではないのだが、これらは大きくプロセス関係とハードウェア、システム情報、そしてシステム（カーネル）パラメータに分けることができる（**リスト1**）。

プロセス関係

各プロセスのステータスは、プロセスIDに対応したディレクトリとして見ることができる。また各プロセスIDディレクトリ以下には、**リスト2**のようなファイルがある（プロセスIDが1のinitプロセスの例）。

プロセス関係に関しては、それぞれ説明するまでもなく、そのファイル名からどのようなステータスが取得できるかはわかると思うし、通常変更することもないので、ここでは詳細は割愛する。実はプロセスのステータスを取得するコマンドとして有名なpsやtopなどのコマンドも間接もしくは直接的にこの/procで情報を取得し、そのデータを加

▼リスト2　プロセスIDが1のinitプロセスのステータス例

```
-r--r--r--    1 root  root       0 Oct 21 17:47 cmdline
lrwx------    1 root  root       0 Oct 21 17:47 cwd -> /
-r--------    1 root  root       0 Oct 21 17:47 environ
lrwx------    1 root  root       0 Oct 21 17:47 exe -> /sbin/init
dr-x------    2 root  root       0 Oct 21 17:47 fd
pr--r--r--    1 root  root       0 Oct 21 17:47 maps
-rw-------    1 root  root       0 Oct 21 17:47 mem
lrwx------    1 root  root       0 Oct 21 17:47 root -> /
-r--r--r--    1 root  root       0 Oct 21 17:47 stat
-r--r--r--    1 root  root       0 Oct 21 17:47 statm
-r--r--r--    1 root  root       0 Oct 21 17:47 status

./fd:
lrwx------    1 root  root       64 Oct 21 17:47 0 -> socket:[191]
lrwx------    1 root  root       64 Oct 21 17:47 10 -> /dev/initctl
```

psコマンドでよく見るプロセスのステータスのほか、環境変数や使用しているファイルディスクリプタなども見ることができる。またプロセスのメモリイメージに直接アクセスすることも不可能ではない。

▼リスト1　ある時点の/procの内容

```
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 1
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 2
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 215
dr-xr-xr-x    3 bin      root            0 Oct 21 17:41 265
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 2729
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 3
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 312
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 323
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 337
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 348
dr-xr-xr-x    3 daemon   daemon          0 Oct 21 17:41 362
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 376
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 390
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 4
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 418
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 432
dr-xr-xr-x    3 nobody   nobody          0 Oct 21 17:41 446
dr-xr-xr-x    3 nobody   nobody          0 Oct 21 17:41 447
dr-xr-xr-x    3 bin      bin             0 Oct 21 17:41 448
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 469
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 470
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 471
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 472
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 473
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 476
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 5
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 501
dr-xr-xr-x    3 root     tadayo-m        0 Oct 21 17:41 502
dr-xr-xr-x    3 tadayo-m tadayo-m        0 Oct 21 17:41 503
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 517
dr-xr-xr-x    3 root     tadayo-m        0 Oct 21 17:41 522
dr-xr-xr-x    3 tadayo-m tadayo-m        0 Oct 21 17:41 523
-r--r--r--    1 root     root            0 Oct 21 17:41 apm
dr-xr-xr-x    3 root     root            0 Oct 21 17:41 bus
-r--r--r--    1 root     root            0 Oct 21 17:41 cmdline
-r--r--r--    1 root     root            0 Oct 21 17:41 cpuinfo
-r--r--r--    1 root     root            0 Oct 21 17:41 devices
-r--r--r--    1 root     root            0 Oct 21 17:41 dma
dr-xr-xr-x    2 root     root            0 Oct 21 17:41 driver
-r--r--r--    1 root     root            0 Oct 21 17:41 filesystems
dr-xr-xr-x    2 root     root            0 Oct 21 17:41 fs
dr-xr-xr-x    4 root     root            0 Oct 21 17:41 ide
-r--r--r--    1 root     root            0 Oct 21 17:41 interrupts
-r--r--r--    1 root     root            0 Oct 21 17:41 iomem
-r--r--r--    1 root     root            0 Oct 21 17:41 ioports
-rw-r--r--    1 root     root            0 Oct 21 17:41 isapnp
-r--------    1 root     root    134221824 Oct 21 17:41 kcore
-r--------    1 root     root            0 Oct 21 16:56 kmsg
-r--r--r--    1 root     root            0 Oct 21 17:41 ksyms
-r--r--r--    1 root     root            0 Oct 21 17:41 loadavg
-r--r--r--    1 root     root            0 Oct 21 17:41 locks
-r--r--r--    1 root     root            0 Oct 21 17:41 meminfo
-r--r--r--    1 root     root            0 Oct 21 17:41 misc
-r--r--r--    1 root     root            0 Oct 21 17:41 modules
-r--r--r--    1 root     root            0 Oct 21 17:41 mounts
-rw-r--r--    1 root     root          269 Oct 21 17:41 mtrr
dr-xr-xr-x    2 root     root            0 Oct 21 17:41 net
-r--r--r--    1 root     root            0 Oct 21 17:41 partitions
-r--r--r--    1 root     root            0 Oct 21 17:41 pci
dr-xr-xr-x    2 root     root            0 Oct 21 17:41 scsi
lrwxrwxrwx    1 root     root           64 Oct 21 17:41 self -> 2729
-r--r--r--    1 root     root            0 Oct 21 17:41 slabinfo
-r--r--r--    1 root     root            0 Oct 21 17:41 stat
-r--r--r--    1 root     root            0 Oct 21 17:41 swaps
dr-xr-xr-x    9 root     root            0 Oct 21 17:41 sys
dr-xr-xr-x    2 root     root            0 Oct 21 17:41 sysvipc
dr-xr-xr-x    4 root     root            0 Oct 21 17:41 tty
-r--r--r--    1 root     root            0 Oct 21 17:41 uptime
-r--r--r--    1 root     root            0 Oct 21 17:41 version
```

数字のディレクトリが各プロセス、それ以外はシステムの各ステータスを表している。

工して表示しているので、プロセス情報はそれらを使ったほうがわかりやすいと思う。もしそれらの情報でもの足りなければここを直接参照するとよいだろう。

ハードウェア、システム情報

/procにあるディレクトリとファイルの中で、数字のディレクトリはプロセスステータスであることは説明した。残りのディレクトリとファイルは、ハードウェアとシステム情報となっている。これらのほとんどはステータスを見ることができるだけなので、興味がある項目をどんどんcatしてみてほしい。

マウントしているものではなく、ハードウェア的に接続されているディスクのパーティション情報や割り込み、PCIバスなど情報をはじめ、ハードディスクの情報も細かく見ることができるので、トラブルシューティングの際にはぜひとも利用したい（**リスト3**）。

▼リスト3　PCIバスに接続されているデバイスの状況とハードディスクドライブの詳細情報

```
# cat /proc/pci
PCI devices found:
  Bus  0, device   0, function  0:
    Host bridge: VIA Technologies VT 82C585 Apollo VP1/VPX (rev 35).
      Master Capable.  Latency=32.
  Bus  0, device   7, function  0:
    ISA bridge: VIA Technologies VT 82C586 Apollo ISA (rev 37).
  Bus  0, device   7, function  1:
    IDE interface: VIA Technologies VT 82C586 Apollo IDE (rev 6).
      Master Capable.  Latency=32.
      I/O at 0x6000 [0x600f].
  Bus  0, device   8, function  0:
    VGA compatible controller: Cirrus Logic Laguna 3D (rev 1).
      Master Capable.  Latency=160.
      Non-prefetchable 32 bit memory at 0xe2000000 [0xe2007fff].
      Non-prefetchable 32 bit memory at 0xe0000000 [0xe1ffffff].
  Bus  0, device   9, function  0:
    Ethernet controller: PCI device 1106:3043 (VIA Technologies) (rev 6).
      IRQ 11.
      Master Capable.  Latency=64.  Min Gnt=118.Max Lat=152.
      I/O at 0x6200 [0x627f].
      Non-prefetchable 32 bit memory at 0xe2008000 [0xe200807f].

# cat /proc/ide/hda/settings
name                    value           min             max             mode
----                    -----           ---             ---             ----
bios_cyl                1232            0               65535           rw
bios_head               255             0               255             rw
bios_sect               63              0               63              rw
breada_readahead        4               0               127             rw
bswap                   0               0               1               r
file_readahead          124             0               2097151         rw
io_32bit                1               0               3               rw
keepsettings            1               0               1               rw
max_kb_per_request      64              1               127             rw
multcount               8               0               8               rw
nice1                   1               0               1               rw
nowerr                  0               0               1               rw
pio_mode                write-only      0               255             w
slow                    0               0               1               rw
unmaskirq               0               0               1               rw
using_dma               1               0               1               rw
```

```
# cat partitions
major minor  #blocks  name

   3     0    9903600 hda
   3     1    4112608 hda1
   3     2    1534207 hda2
   3     3    4152802 hda3
   3     4      96390 hda4
   3    64    9873360 hdb
   3    65    9871911 hdb1
```

▼リスト4　/proc/slabinfoの例

```
# cat /proc/slabinfo
slabinfo - version: 1.0
kmem_cache              32      42
tcp_tw_bucket           0       0
tcp_bind_bucket         13      127
tcp_open_request        0       63
ip_fib_hash             10      127
ip_dst_cache            8       25
arp_cache               3       31
skbuff_head_cache       18      42
sock                    38      44
filp                    366     378
inode_cache             181183  182424
signal_queue            0       0
kiobuf                  0       0
buffer_head             4513    4536
mm_struct               27      50
vm_area_struct          763     819
dentry_cache            139293  145762
files_cache             26      36
uid_cache               4       127
size-131072             0       0
size-65536              0       0
size-32768              0       0
size-16384              0       0
size-8192               4       4
size-4096               1       2
size-2048               65      68
size-1024               22      24
size-512                21      24
size-256                13      28
size-128                333     350
size-64                 288     420
size-32                 7549    11781
slab_cache              28      63
```

Linuxが採用するメモリ割り当て機構のステータスを確認できる。

```
# cat interrupts
           CPU0
  0:    7221808          XT-PIC  timer
  1:          4          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
 11:      45182          XT-PIC  eth0
 12:          2          XT-PIC  PS/2 Mouse
 13:          1          XT-PIC  fpu
 14:      49526          XT-PIC  ide0
 15:          2          XT-PIC  ide1
NMI:          0
ERR:          0
```

また/proc/slabinfoでは、Linuxが採用しているメモリの効率的な利用のためのSlab Allocatorとして知られるオブジェクトキャッシングカーネルメモリ割り当て機能の状態を見ることができる（**リスト4**）。Slab Allocatorは、ハートウェア寄りのメモリ管理機能であるページングシステムと実際のメモリ要求との間に存在し、各オブジェクトに対してメモリ割り当てを行う作業を担当している。これは、かなり面白いのでまた今度詳細をお伝えすることにしよう。

システム（カーネル）パラメータ

/proc/sysには、変更可能なシステム（カーネル）パラメータが集められている。主な項目としては、ファイルシステム、ネットワーク機能、メモリ管理（キャッシュ）がある。ここではシステムのパラメータを変更でき、それによりシステムのチューニングが可能なわけだが、変更によっては、システムがクラッシュする危険性があるので十分に注意してほしい。なお値の設定には、echoコマンドが利用でき、変更は即座にシステムに反映される。また設定できるのは、各ファイルのパーミッションを見ればわかると思うが、スーパーユーザーのみである。あと、ここで設定したものは、どこかに保存されるわけではないので、システムをリブートしてしまえば、その設定は消えてデフォルトの状態に戻ることになる。そこで、十分に安定していることを確認した最適な設定を見つけた場合は、/etc/init.d/rc.localなどの初期化ファイルに記述するとよいだろう。あっそうそう、ちまちまcatしていくのも面倒なので、“ for a in ＊ ;do echo $a ;cat $a;done ”てな感じで参照していくと便利だ。

実際に参照してみるとわかるが、複数のフィールドで構成されているパラメータもあり、単にここを参照しただけでは何がなんだかわからないと思う。そこで、/usr/src/linux/Documentation/filesystems/proc.txtはもちろんのこと、該当するソースファイルも必要に応じて参照するように心がけたほうがよい。

ここでは設定可能なパラメータのさわりを紹介することにしよう。

■/proc/sys/fs（ファイルシステム）

/proc/sys/fsで設定可能なのは、キャッシュするディスククォータエントリとファイル、スーパーブロックそれぞれのハンドル（ディスクリプタ）の最大値だ。

```
# for a in *max* ;do echo $a ;cat $a;done
dquot-max
0
file-max
4096
super-max
256
```

筆者の環境では、ディスククォータは無効にしているので値は0となっているが、ファイルハンドルとスーパーブロックハンドルの最大値はそれぞれ、4096と256となっている。

ファイルハンドルの最大値は、冒頭でも述べたようにシステムで同時にオープン可能なファイルの上限である。4096だとProxyキャッシュサーバ用途でもとりあえず問題ないのではあるが、キャッシュサイズによっては8192程度にしてもよいかも知れない。しかし、ここにはひとつ落とし穴がある。それはシステムの上限とは別にプロセス単位でも上限が設定されているということだ。通常プロセス単位での上限は、ulimit -nで確認できる。ulimitで指定する設定は、プロセスの親から子へと継承されているので、明示的に変更しない限りinitプロセス以下、すべてのプロセスの設定は同じということになる。initによって生み出されたデーモンプロセスもその例外ではない。通常ファイルハンドルのリミットは1024になっているはずだ。というわけで、このような場合には、デーモンプロセスを起動する前に実行される初期化スクリプト、もしくはそのデーモンプロセス起動用のスクリプトの冒頭に“ ulimit -n 4096 ”などと指定する必要がある。もちろんソースコードを直接変更してリコンパイルしてもよいのだけれども。

なおスーパーブロックハンドルの最大値は、マウント可

能なファイルシステムの数の上限である。

■/proc/sys/kernel

/proc/sys/kernelには僕の心をくすぐるようなものが見いだせなかったので省略。

■/proc/sys/net

/proc/sys/netは、さらにcore、unix、802、ethernet、ipv4、bridgeなどの機能ごとのディレクトリに分かれており、ネットワーク関連の設定項目が集約されている（**リスト5**）。ここには、有益な設定項目があまりにも多くて、紹介しきれないので機会をあらためることにする。

■/proc/sys/vm

/proc/sys/vmでは、システムの仮想記憶機構の調整とバッファキャッシュなどの調整を行えるようになっている。一応次のようなものが用意されているのだが、すべてが有効な値であるとは限らない。ドキュメント上では有効なように記述されていても、ソースコードを見てみると実は参照されていない場合もあるので、結局のところソースコードを見るしかなかったりする。

```
# for a in * ;do echo $a;cat $a;done
bdflush
40    500   64    256   500   3000  500   1884  2
buffermem
2     10    60
freepages
256   512   768
kswapd
512   32    32
overcommit_memory
0
page-cluster
4
pagecache
2     15    75
pagetable_cache
25    50
```

それぞれ説明するのは面倒なので、ここではこれらの中でもっとも調整のしがいがありそうなbdflushを調整してみることにしよう。

bdflushは、9つのフィールドで構成されている。単にcatしただけでは、それぞれのフィールドが何のパラメータかわからないので、実際にソースコードを参照することにしよう。bdflushのパラメータの定義は、/usr/src/linux/fs/buffer.cにある（**リスト6**）。

これらを参照すると、各フィールドの意味やパラメータの有効範囲が一目瞭然だ。ソースコード内では、インター

▼リスト5　/proc/sys/netの主な項目

```
dr-xr-xr-x   2 root     root            0 Oct 27 13:45 802/
dr-xr-xr-x   2 root     root            0 Oct 27 13:45 core/
dr-xr-xr-x   2 root     root            0 Oct 27 13:45 ethernet/
dr-xr-xr-x   5 root     root            0 Oct 27 13:45 ipv4/
dr-xr-xr-x   2 root     root            0 Oct 27 13:45 token-ring/
dr-xr-xr-x   2 root     root            0 Oct 27 13:45 unix/

./core:
-rw-r--r--   1 root     root            0 Oct 22 01:49 message_burst
-rw-r--r--   1 root     root            0 Oct 22 01:49 message_cost
-rw-r--r--   1 root     root            0 Oct 22 01:49 netdev_max_backlog
-rw-r--r--   1 root     root            0 Oct 22 01:49 optmem_max
-rw-r--r--   1 root     root            0 Oct 22 01:49 rmem_default
-rw-r--r--   1 root     root            0 Oct 22 01:49 rmem_max
-rw-r--r--   1 root     root            0 Oct 22 01:49 wmem_default
-rw-r--r--   1 root     root            0 Oct 22 01:49 wmem_max

./ipv4:
dr-xr-xr-x   6 root     root            0 Oct 22 01:49 conf
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_destunreach_rate
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_echo_ignore_all
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_echo_ignore_broadcasts
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_echoreply_rate
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_ignore_bogus_error_responses
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_paramprob_rate
-rw-r--r--   1 root     root            0 Oct 22 01:49 icmp_timeexceed_rate
-rw-r--r--   1 root     root            0 Oct 22 01:49 ip_autoconfig
-rw-r--r--   1 root     root            0 Oct 22 01:49 ip_default_ttl
-rw-r--r--   1 root     root            0 Oct 22 01:49 ip_dynaddr
-rw-r--r--   1 root     root            0 Oct 22 01:49 ip_forward
-rw-r--r--   1 root     root            0 Oct 22 01:49 ip_local_port_range
-rw-r--r--   1 root     root            0 Oct 22 01:49 ip_no_pmtu_disc
-rw-r--r--   1 root     root            0 Oct 22 01:49 ipfrag_high_thresh
-rw-r--r--   1 root     root            0 Oct 22 01:49 ipfrag_low_thresh
-rw-r--r--   1 root     root            0 Oct 22 01:49 ipfrag_time
dr-xr-xr-x   5 root     root            0 Oct 22 01:49 neigh
dr-xr-xr-x   2 root     root            0 Oct 22 01:49 route
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_fin_timeout
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_keepalive_intvl
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_keepalive_probes
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_keepalive_time
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_max_syn_backlog
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_retrans_collapse
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_retries1
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_retries2
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_rfc1337
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_sack
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_stdurg
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_syn_retries
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_timestamps
-rw-r--r--   1 root     root            0 Oct 22 01:49 tcp_window_scaling
```

core、unix、802、ethernet、ipv4、bridgeなどの機能ごとのステータスを確認できる。またファイルのパーミッションが書き込み可能になっているものは変更することができる。最近はシステムの搭載メモリが多くなっていることもあって、デフォルト設定でメモリは消費するけどパフォーマンスもよいものとなっている。

バルなどの時間を設定するところで数値にHZを乗算して、秒換算としている。つまり5＊HZは5秒ということだ。/proc/sys/vm/bdflushでは、5＊HZは500として表示されている。HZは、jiffiesと呼ばれるシステムで用いられる時間の単位、普通PCは10ミリ秒（Alphaシステムは5ミリ秒だったかな）をもとにしている。でも/proc/sys/vm/bdflushにechoコマンドで設定する時は、5＊HZといった表記は解釈されないので、500などの数値で指定する必要がある。まぁ、指定したい秒数に100をかけた値を指定すると覚えておけばよい。

さて、どのような設定が最適なのだろう。システムの性能やそのバランスに関係しているので、一概にこの設定がよいとはいいがたいのだけれども、ひとつ覚えておきたいことがある。それはbdflushの設定の調整は、局所的なパフォーマンスの向上に貢献するということだ。つまり、バッファキャッシュをディスクに書き出すタイミングを遅らせ、後回しにすることで、あるタスクの見かけ上の処理時間を短くすることができる。ディスク性能はよくなったとはいえ、メインメモリと比べると数十倍から数百倍遅いので、その実処理を後回しにすることは、多くの場合、効果的なのだ。しかし、後回しにするといっても限界がある（あまり長い間変更を反映させないとシステム上のリスクが大きくなる）ので、処理時間の長くかかるタスクでは、遅延書き込みによる効果は薄くなる（まぁ、読み取り時の効果があるので調整の意味がないなわけではないのだけど）。ただ、未保存バッファの書き出しサイズに影響する2番目のフィールドを大きくし、ディスクアクセスの回数を減らす（バーストライトっぽくする）ことによる効果はあるかもしれない。しかし、これを極端にやってしまうと、ディスクアクセスの効率化はできるが、逆に長時間の割り込みのできない処理が発生するため、タスクの円滑な動作が損なわれ、システム挙動のスムーズさが欠けることにもなってしまうので、注意が必要だ。

とりあえず、変更は動的に簡単に行えるので、いろいろ値をかえてチェックしてみてほしい。一応、僕のテスト環境では、全体的に2倍にしたような次の設定でしばらく運用している。

```
# echo "80 1500 128 512 1000 6000 500 1884 2"  >
/proc/sys/vm/bdflush
```

効果は、と言いたいところなのだけど、別の問題で僕のシステム（2.3.22と2.3.23）が不安定（ページスワップのあたりが変で、kswapdがゾンビになりやがった）なので、結果はまた今度お伝えすることにしよう（安定したシステムでね）。

最後にCD-ROMなどのリムーバブルメディアをマウントした場合は、/proc/sys/devディレクトリにcdrom/infoが現れるようになっている。マウントした際に見てみよう。

```
# cat /proc/sys/dev/cdrom/info
CD-ROM information, Id: cdrom.c 3.05 1999/09/23
drive name:             hdc
drive speed:            32
（以下略）
```

▼リスト6　bdflushのパラメータの定義

```
union bdflush_param {
        struct {
                int nfract;  /* Percentage of buffer cache dirty to activate bdflush */
                int ndirty;  /* Maximum number of dirty blocks to write out per wake-cycle */
                int nrefill; /* Number of clean buffers to try to obtain each time we call refill */
                int nref_dirt; /* Dirty buffer threshold for activating bdflush when trying to refill buffers. */
                int interval; /* jiffies delay between kupdate flushes */
                int age_buffer;  /* Time for normal buffer to age before we flush it */
                int age_super;  /* Time for superblock to age before we flush it */
                int dummy2;    /* unused */
                int dummy3;    /* unused */
                } b_un;
        unsigned int data[N_PARAM];
} bdf_prm = {{40, 500, 64, 256, 5*HZ, 30*HZ, 5*HZ, 1884, 2}};

/* These are the min and max parameter values that we will allow to be assigned */
int bdflush_min[N_PARAM] = {  0,  10,    5,   25,  0,   1*HZ,   1*HZ, 1, 1};
int bdflush_max[N_PARAM] = {100,50000, 20000, 20000,600*HZ, 6000*HZ, 6000*HZ, 2047, 5};
```

Linux Today

訳：日下部圭子

「フリーハードウェア」について

Text : Richard Stallman

■ソフトとハードの違い

多くの人々がGNUプロジェクトに対し、フリーソフトウェアだけでなくフリーハードウェアデザインに手を広げたいのではないかと質問したり、その仕事をしてみたいと言ったりする。フリーなチップをデザインするプロジェクトを提案した人さえいた。

この問題をはっきりと理解するために、「フリーソフトウェア」とは自由にかかわるものであって、価格の問題ではないということを思い出してほしい。おおまかに言えば、ユーザーはそのソフトウェアを自由に複製したり、変更を加えたりできるということだ。だから、それと同じ概念をハードウェアに適用すると、「フリーハードウェア」というのは、ユーザーが自由に複製したり変更を加えたりできるハードウェアを意味することになる。また「フリーハードウェアデザイン」というのは、ユーザーが自由に変更を加えたり、ハードウェアにしたりできるデザインという意味になる。

フリーソフトウェアはしばしば無料で入手できるが、これは複製を作るのにコストがかからないことが多いためだ。このために「フリー」を「ただ」と混同する風潮がある。ハードウェアに関しては、「フリー」と「ただ」との差はより明確である。ネットワークを介してハードウェアをダウンロードすることは不可能だし、ハードウェアの自動複製機も存在しない。このため、ハードウェアやデザインがフリーであっても、ハードウェアの複製を新しく作るにはコストがかかるものと考える必要がある。部品にお金がかかるだろうし、基板を作ったり線やチップをハンダづけしたりしてくれるのは、本当に仲のよい友人ぐらいなものだろう。

ソフトウェアの複製の自由は重要な権利だ。それを行うことは簡単で、コンピュータのユーザーなら誰でもできることだ。ハードウェアの複製の自由はそれほどには重要ではない。というのは、ハードウェアの複製は困難だからだ。

しかし、多くのハードウェア愛好者は、フリーハードウェアデザインの開発に興味をもっている。これは彼らがハードウェアを設計したり、カスタマイズしたりするのが好きだからだ。ハードウェアの設計やカスタマイズをして働きたいならば、フリーハードウェアデザインの仕事はすばらしいものだろう。GNUのボランティアコーディネイターは、そのような人に対して同じ興味をもつ人々を紹介することができる。もしこの目的のための組織が作られることあれば、GNUプロジェクトは興味のある人々に対し、その組織を紹介するだろう。

■インターフェイス仕様はフリーに

GNU GPLやその他の種類のコピーレフトを、ハードウェアデザインに使う可能性について質問されることがよくある。

プログラム可能な論理デバイスや、マイクロコードマシンのためのプログラムのようなファームウェアはソフトウェアであって、他のすべてのソフトウェアと同様にコピーレフトできる。しかし、回路そのものについては事情がもっと複雑だ。

回路はコピーレフトできない、というのはそれがコピーライトできないからだ。HDL（ハードウェア定義言語）で書かれた定義はコピーレフトできるが、コピーレフトは単にその定義の表現を扱うだけで、その回路そのものを扱うものではない。同様に、回路の図面や実際の配置はコピーレフトできるが、それは単にその図面や配置だけに関するものであって、回路そのもののコピーレフトではない。つまり、同じ回路構成を別の見かけで書いたり、同じ回路を作る別のHDL定義を書いたりすることがだれにでも合法的に行えるということだ。したがって、回路に適用した場合のコピーレフトの効力は限定される。しかしそれでもなお、HDL定義やプリント配線回路の配置をコピーレフトすることは何らかの利点があるだろう。

また、この目的に特許を使うこともおそらく不可能だろう。特許はコピーライトのようには機能しないし、取得にたいへんコストがかかる。

ハードウェアデバイスの内部設計がフリーであろうとなかろうと、とにかく重要なことはそのインターフェイス仕様がフリーであるべきだということだ。ハードウェアの制御方法が分からなければ、それを動かすフリーソフトウェアが作れない(ハードウェアを販売しながら、顧客にはその使い方を教えないというのは、私には恥知らずのふるまいとしか思えない)。しかしそれはまた別の話だ。

プロフィール

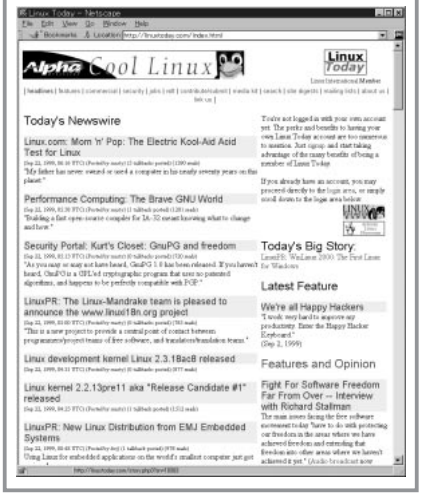
Richard Stallmanは Free Software Foundation(FSF)の創始者であり、GNU General Public License (GPL)の著者であり、gccやEmacsといった重要なソフトウェアのオリジナルの開発者である。

米国LinuxToday提携

http://linuxtoday.com/

毎月、米国の人気Linuxサイト「Linux Today」に掲載された記事の要約をお届けします。記事の全文は日刊アスキーLinuxで読むことができます。

http://www.linux24.com/



Linuxをうまく売り込む方法

Text : Tom Adelstein

●Linuxをビジネスにするための5つの戦術

Linuxを使って商売を繁栄させたければ、それに役立ついくつかの習慣に馴染む必要がある。オープンソースソフトウェアは、コンサルタント業にとっても今よりずっと儲かるものである。ここにいくつかの役に立つ戦術を挙げよう。

1. 会社の目標に対して売り込むこと

会社には毎年、達成目標というものがある。「彼らの目標達成を手助けする方法はないか」を考えよう。そして相手に会う約束を取りつけて、その会社の方向性について話し合うことだ。たとえば「Web戦略として現在どんなことをしていますか？」というようなことを質問すると、彼らはそれについて、とめどなく話し続ける。そのようなミーティング中のある時点で、経営者は私がどのような手助けができるかを、話す機会を与えてくれる。うまくいくと思わせることができれば、その時点で仕事を得たことになる。

2. ソリューションの規模を自分のマーケットに合わせること

仕事を始めたばかりなのであれば、快適にこなせる仕事だけを探すことだ。私は、こなせそうにない規模の仕事を紹介されたときには、他の簡単な解決策を検討し、提案するという方法をとる。

仕事の規模を合わせるということは、予算を切り詰めた小さな仕事を探せという意味ではない。予算の厳しい顧客はあなたの時間を食いつぶし、あなたのサービスに見合う額のお金を払いたがらない。プライドをもち、率先してよい仕事を追い求めることだ。

3. 適所に特化し、その範囲で最良のものになること

Linuxを使えば、ITの管理者が日々出会う多くの問題を解決することができる。まず、そうした問題のひとつについて、うまく解決できるようになること。そして自分を必要とする人々を探すこと。Windows NTのプリントサーバを使うならば、LinuxとSambaを使えばライセンス料を節約できる。それでITの管理者は、あなたのおかげで節約した費用を別のことに使うことができる。

4. 理屈や言い訳、正当化をやめよ

とにかく行動することだ。「Linuxはまだ使えないといわれたらどうしよう」などという臆病な考えには耳を貸さないようにする。そんな言葉には何の意味もない。

商売においては、ひとつの「Yes」にいくつかの「No」がついてまわる。ある人が「No」と言ったら、それはあなたが「Yes」に一歩近づいたというだけのことだ。あなたはこの相手の抱えている問題への、最良の解決策を持っているのだ。彼がそれを望まないなら、それを望むほかの人を探せばよい。このやり方でいけば、あなたは楽しく自分の仕事することができるだろう。

5. 取引をまとめる方法を学ぶこと

会社の方向性について話し始めたとき、すでに商売のプロセスは始まっている。要求を知ることができたら、解決策を話すことができる。次の瞬間には、取りまとめのプロセスが始まっている。

そうなると、顧客は答えるとなると「Yes」か「No」か「わからない」の3つの選択肢しかない。「Yes」と言ったら、その提供を始めるための打ち合わせをしなければならない。「No」といったら、製品についてか、価格についてなのかをたずねればよい。「No」という答は、「わからない」という意味で言っている場合もある。この場合、決定を下すために知る必要があるものを提供すればよい。

最後に、取りまとめを行う際にやってしまいがちな最大の失敗は、しゃべり続けてしまうことだ。だれかが「Yes」といったら、考え直される前に開始時期のスケジューリングを始めることにしよう。

以上5つの戦術で、商売には多少活気が与えられるはずだ。我々は自分たちの持っている道具を使い、仕事で成功したい。我々はオープンソフトウェアソフトウェア社会で起きることに対し、プライドを持つことができるはずだ。お金を儲け、自分の仕事を楽しめる人々は、そうすると言っているからこそ、そうできるのだ。あなたはLinuxを売ることができる。とにかく、あなたはもうすでに1つ売っているのだ。誰に？ あなた自身にだ。さあがんばろう！

http://www.linuxtoday.com/stories/10506.html

プロフィール
Tom AdelsteinはBynari Systemsの財務担当役員兼IT担当役員。Bynari SystemsはLinuxをベースにしたサポートとサービスを提供している。

Books

PC-UNIXサーバのためのクラッカー撃退計画

まえだひさこ 著
翔泳社
B5変形判／276ページ／CD-ROM付き　　本体価格 2800円

Linuxマシンをインターネットに接続しているあなた、クラッキング対策はしているだろうか？　何の対策も講じていない、あるいは、shadowパスワード、SATAN、OTP、sshというキーワードを聞いたことがないなら、すぐにこの書籍を購入すべきだ。さもないと、世界中からroot失格の烙印を押される日も遠くないだろう。

クラッキングへの対策は日頃の情報収集と、まめなメンテナンスが重要だが、最初はどこから手をつけていいのかがわかりにくい。本書は、システム管理者として知っておくべきセキュリティの知識を解説し、数々のフリーソフトウェアを使ったセキュリティチェックの方法と防御のしかたをわかりやすく解説している。付属のCD-ROMには、Red Hat Linux 6.0英語版と、本文中で紹介しているフリーソフトウェアを収録している。

自分の身は自分で守る、ハードボイルドなrootを目指そう。

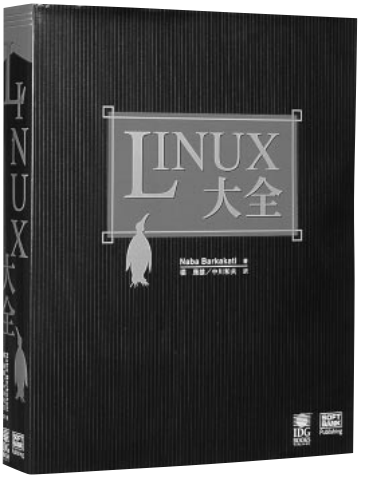
Linux大全

Naba Barkakati 著橘 康雄、中川和夫 訳
ソフトバンク　パブリッシング
B5変形判／728ページ／CD-ROM付き　　本体価格 4800円

『Red Hat LINUX Secret, 2nd Edition』の翻訳書である本書は、インストールからプログラミングまで、極めて多岐にわたる内容を網羅し、解説も詳細だ。Part1がインストールとカーネルの再構築、Part2がLinuxの持つ機能、Part3では各種ハードウェアの解説と設定、最後のPart4ではネットワークやプログラミングなど特定用途でのLinux活用法についてそれぞれ取り上げている。Linux関連書籍の中でも、この情報量は随一だろう。

原書では英語版のRed Hat Linux 5.2をターゲットとしているが、本書にはRed Hat Linux 5.2をベースとして日本語対応を行ったVine Linux 1.1および、Red Hat Linux 6.0英語版を収録したCD-ROMが付属する。

初心者には少々荷が重いが、中級者の方が必要な情報を調べる事典として使うのに適した書籍といえよう。



PHP徹底攻略

堀田倫英／石井達夫／広川類 著
ソフトバンク　パブリッシング
B5変型判／336ページ／CD-ROM付き　　本体価格 3200円

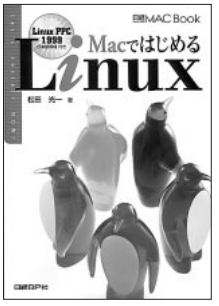
PHPとは、HTML埋め込み型のサーバサイドスクリプト言語であり、C言語やPerlのような文法でプログラムを記述できる。WebサーバソフトのApacheのモジュールとして動作するため、CGIプログラムのように別プロセスで起動されない。そのため無駄なメモリを消費せず、処理も高速である。各種データベースへのインターフェイスが用意されており、HTMLにPHPスクリプトを埋め込むために、プログラムの管理がしやすくなっている。

PHPは、ApacheやPostgreSQLと同様にフリーソフトウェアとして公開されていて、多くのユーザーがいる。本書はPHPを日本語で解説した初めての書籍で、PostgreSQLデータベースソフトとの連携の例や、関数のリファレンスに多くのページを割いており、プログラミングの知識がある読者を対象としている。Webサーバ上でCGIプログラムを使っていて、負荷が重くて困っているならば、PHPに置き換えてみるとよいだろう。

特選 星降る夜のパソコン情話 ～Linux狂騒曲～

中村 正三郎 著
ビレッジセンター出版局
四六判／408ページ
本体価格 1400円

MacではじめるLinux



松田 光一 著
日経BP
B5判／256ページ／CD-ROM付き
本体価格 3200円

UNIXプログラミングツール

Eric Foster-Johnson 著　たかのゆきまさ ~ 監訳
翔泳社
B5変型判／290ページ／CD-ROM付き
本体価格 2800円

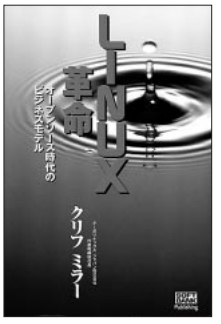
伽藍とバザール

Eric Steven Raymond 著　山形 浩生 訳・解説
光芒社
A5判／256ページ
本体価格 1800円

UNIXコマンド ポケットリファレンス ビギナー編

IDEA・C 著
技術評論社
四六判／346ページ
本体価格 1880円

LINUX革命



クリフ ミラー 著
ソフトバンク　パブリッシング
四六判／216ページ
本体価格 1400円

最近売れてるLinux書籍は？

ここ数年、コンピュータ関連の認定資格試験が流行しています。MicrosoftのMCP／MCSEやMOUS、LotusのCLSやCLP、CiscoのCCNAやCCNP、OracleのORACLE MASTERなど各社独自の認定資格試験が数多くあり、不況という世相を反映してか、受講者数も増加しています。そして、2000年初めには、ついにというべきか、やっというべきか、Linuxの資格試験が登場するようです。

このLinuxの資格試験で、現在のところわかっているものに次の2つがあります。まず、「SAIR LINUX and GNU Certification」（http://www.linuxcertification.org/）。これに対応して、John Wiley & Sonsから1999年12月に『Installation & Configuration』、2000年3月に『System Administration』といった書籍が出版される予定です。ほかに、『Networking』『Security, Ethics & Administration』があり、まず、初級者向けのLevel 1として出版されるこれらの以外にも、レベル別に全16分冊程度のラインナップになるようです。もう1つは、「Linux Professional Institute Certification（LPIC）」（http://www.lpi.org/）。こちらは、Macmillan Computer Publishingが書籍を出版し、Level 3まで出す予定のようですが、詳しいタイトルなどはまだ不明です。国内でも、こうしたLinux認定試験への対応に向けて、いろいろ動きが水面下であるという噂です。

現在のLinux人気を見ていると、このような資格試験はこれからさらにいろいろな種類のものが出てくるのは間違いないところでしょう。もしかしたら、2000年ごろにはMCSEなどにLinux資格試験が取って代わっているのかもしれません。そうなることを信じて、今のうちからLinux magazineで勉強を始めておけば、将来は高給取りになれるかもしれません（保証はできませんが…）。

最後に最近話題のLinux関連書籍を1冊紹介しておきましょう。中村正三郎氏の著書第5弾でLinuxを題材にした『特選 星降る夜のパソコン情話～Linux狂騒曲～』（中村正三郎著、ビレッジセンター出版局刊、ISBN4-89436-131-0、本体1400円）が、発売後好調な滑り出しを見せています。全編正三郎ワールドが繰り広げられていて、肩のこらない読み物として楽しめます。（書泉ブックドーム 古田島）

読者の声

俺にもいわせろ！

だんだん冬が近づいて、人恋しくなる季節になりました。Linuxだけでなく、家族やお友達と遊ぶのも忘れずに。

まだまだこれから

今年の初めにPC/AT機を自作したのがLinuxに触れるキッカケでした。自作機をWindowsに占有させるのが面白くなく、TurboLinux 3.0から始めて、今ではFreeBSDまでインストールしてしまい、我が自作機には、現在5つのOSがインストールされています。50歳からの挑戦です。

（東京都　飯沼良文さん）

何事も挑戦ですね。私の両親も、還暦を過ぎてパソコンを使うようになりました。飯沼さんとはレベルが違って、メールを1通書くのに2時間ほどかかっていますが……。はたして、Linuxまでたどり着けるのかしら。

残業……なんですか？

終電を逃してしまった某日、会社のマシンにこっそりTurboLinuxを入れて悦に入ってました。意味もなく残業する日が増えそうです。

（東京都　中森尚子さん）

ますます終電を逃すようになってしまうかもしれませんよ。ところで、それって残業と言うのでしょうか？

Linuxまでの道程

純正IBM機→DOS/V→Windows 3.0+DDD→OS/2の次がUNIX（Linux）ですが、Windows 9xと違い「簡単には働かない」ところが気に入っています。（働き出すと、なかなかコケませんが）。OS/2は、日本語版の「Officeもの」がありませんので実用的とはいえませんでしたが、Linuxは期待できそうだと思っています。

（大阪府　久保 寿さん）

自宅ではOS/2をメインに使ってます。Linuxもそうですが、放っておいても落ちずに動くところが気に入っています。Linux、OS/2両方のユーザーとしては、StarOfficeの日本語版が待ち遠しい今日この頃です。

めざせ、コマンド使い

X Window Systemが立ち上がらず（ビデオカードがおかしい）ずっと黒地に白のDOS様式の画面……。だが、Xが使用できるようになってもほとんどコマンドでやるオレって……

（三重県　中子敦雄さん）

トラブルに遭うとたくさんのコマンドを覚えますよね。バキバキとコマンドを叩き込んでいる熟練ユーザーをみると、感心すると同時に、今までどれほどハマってきたのか聞いてみたくなります。でも、コマンドを使いこなしている人ってカッコいいですよね。

叱咤激励

このようなコーナーでよく言わせてもらうのは、「その雑誌でなければ入手できないような情報があるべきである」ということだ。大部分の雑誌は五十歩百歩である。

（埼玉県　岡本之良さん）

より役立ち、面白く、独自のカラーが出せるよう、努力をいたします。今後ともよろしくお願いいたします。

忍び寄るWinmodemの恐怖

Linuxを使いはじめて1年ほどですが、Windowsとくらべてハードの設定などが難しくて、悩みまくりです。特にモデムが外付けのものはなんとかなるのですが内蔵型は、ぜんぜんウンともスンともいいません。なにかヒントを!!

（神奈川県　矢島俊彦さん）

今月の特集で取り上げているように、ノートPCを中心に、Windows専用の内蔵モデムが多くなってきたようです。このようなモデムをLinuxで使えるようにしようというプロジェクトもありますが、実用になるのはもう少し先のようです。

求む！　Beep音の専門家

キーボード入力ミスした時に鳴る、ビープ音を消すのはどうすればよいでしょうか？

（福岡県　井上陽介さん）

X Window System上でなら、ターミナルから「xset b off」と入力すればビープ音を消すことができます。ターミナルは、terminfoをいじるのだと思いますが、どなたかご存知ありませんか？　え、おまえはどうしてるのかって？　スピーカーコネクタを抜いています(^^ゞ

禁断のLinux

Linux、まったくやっかいな物にはまってしまいました！

（大阪府　中西規雄さん）

いくらでもハマれるのが、Linuxの良いところでもあり、悪いところでもあるのかもしれません。厄介なことにはなりませんので、Linux magazineにもぜひハマってください。

キラーアプリはコレだっ！

LASER5 Linux Free版をインストールしてみました。KDEも初めて使ってみましたが、必要なツールはすべてそろっているようで、あとはポストペットさえあればLinuxだけでいけるのですが。

（東京都　関 のり子さん）

関さんからのお便りで、編集部に衝撃が走りました。Linuxが爆発的に普及するきっかけとなるアプリケーションはナニか。部内でも話し合ったことはあるのですが、結論が出ませんでした。そうか、そうだったのか……。編集部員は、口をそろえて言ったのです「うん、たしかに欲しい」。ソニーコミュニケーションネットワークさん、Linux版をお願いします。

Linux 親子鷹

DOS時代から、頭を悩ませている親父よりも早くLinuxをものにしたいものです。

（鳥取県　榎本貴志さん）

親子でLinux、いいですね。だんだんそういう家族も増えてくるのでしょう。お父さんに負けないよう、私たちもお手伝いします。

日々是精進

私は貴社刊行の「Linux&BSD」や「256本」でUNIXの世界にハマリました。そのアスキーさんからLinuxマガジンや姉妹誌（？）のBSDマガジンが発行されるようになったことは、大変うれしく思っております。「スーパーアスキー」が休刊となり、しばらく寂しい思いをしておりましたが、これからは毎月8日が楽しみになりそうです。

（栃木県　永井裕一さん）

創刊号から読んでいるが、内容的にも面白いものが増えてきたと思う。他誌の内容も向上していると感じられつつある現在、負けないくらいがんばってほしい。

（東京都　林 昌幸さん）

ありがとうございます。BSD magazineの編集部も同じフロアにあり、切磋琢磨しながら誌面づくりに励んでおります。BSD magazineともども、これからもよろしくお願いいたします。

viに目覚める

月刊化を期にUN?X＊から浮気してみることにしました。「viはじめました」がおもしろいので、本気になるかも……。Linuxをはじめてから1年足らずですが、ずっとMuleを使っていました。Linuxをインストールしてはじめてviを開いたときは、何じゃこれは〜という感じで、文字を入力するどころか開いたviを閉じることもままならなかったので、以来viには触っていませんでした。「viはじめました」を先生におそるおそるviをいじっていると、それがけっこう面白いと思えてくるこの頃です。アルファベットの1文字コマンドを組み合わせてxpとかxPとかやると、MuleのC-tと同じじゃないか！などという大発見に感激したりするわけです。ああ、早くviでシェルスクリプトをバリバリ書いて、Linuxを賢く使うようになりたい!!

（奈良県　奥田芳史さん）

白状します。私もvi苦手なんです。暗い過去があるもので。触りもしないで、あまりに嫌うので、デスクからはボロクソに言われるは、先輩からはvi以外使えない環境で仕事をさせるしかないと言われるはで、本当にそんな環境に叩き込まれたらどうしようかと毎日おびえて暮らしています（私も「viはじめました」で勉強して、viマスターになってやるという野望を抱いているのは内緒です）。

えええっ？ (^^;;;;

NeXTの特集をして下さい（すごく無理なお願い）

（山口県　藤井弘一さん）

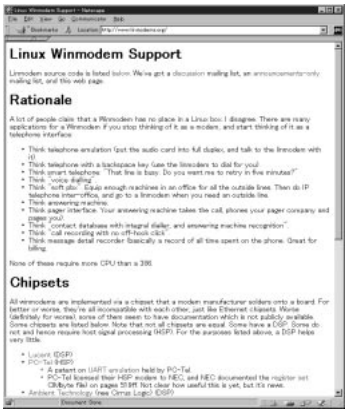
う〜ん。ちょっと難しいと思います。

Internet Resources
～ちょっと気になるWebサイト紹介～

Development

Linux Winmodem Support

http://www.linmodems.org/



内蔵モデムの主流であるWinmodemは、本来モデムのチップが行う仕事を、本体のCPUがソフトウェアで実現している。これによりモデムの製造コストは抑えられるが、仕様が公開されておらず、ドライバの作成が非常に難しい。そのためWindows以外のOSでは動作せず、Linux関連のMLでもたびたび「Linuxがモデムを認識しない」と話題にのぼる。このプロジェクトではWinmodemをLinuxで使用可能にすることを目的としている。

Mailing List

Red Hat Linux メーリングリスト

http://www21.cds.ne.jp/~net/redhat/

日本でのRed Hat系ディストリビューションのユーザー全般を対象としたメーリングリストの紹介ページ。ここではMLへの入会、掲示板の閲覧と投稿、メールマガジンの申し込みなどができる。MLの過去ログをたどってみると、Linuxの初心者にも敷居が高くないことがうかがえる。「Linuxを始めたいけど難しそう」という人や、「始めたはいいけどわからないことだらけで」という人は、まずここで質問してみてはどうだろうか。

Knowledge for Public

EDGE: To Arrive At The EDGE of The World's Knowledge

http://www.edge.org/

身近なものからそうでないものまで、多くの人が不思議に思うような科学分野を、オーソリティーへのインタビューやコラムで、わかりやすく紹介している。コラムには「カッコウはコンピュータに卵を産む」でおなじみのCliff Stollや、SunのScott McNealyなどが登場し、インタビューでは「子どもは中途半端がキライ」という題で、親が子に及ぼす影響と子供の人格形成について心理学者が語るなど、興味深い話が満載だ。

Information

Tom's Hardware Guide（日本語版）

http://tom.g-micro.co.jp/

ハードウェア情報専門サイトとして定評のある「Tom's Hardware Guide」の日本語版。本家の記事の翻訳そのままというより、日本のPCユーザー全般に向けたニュースが目立つ。OSの最新情報、ハイスペックCPU情報など、多くのPCユーザーが待ち望んでいるでいる情報がてんこ盛りだ。「動きの早い分野の情報収集はオンラインで」というのを地でゆく典型的なサイトらしく、更新頻度も高いので、こまめにチェックしてほしい。

Game on Linux

The Linux Game Tome

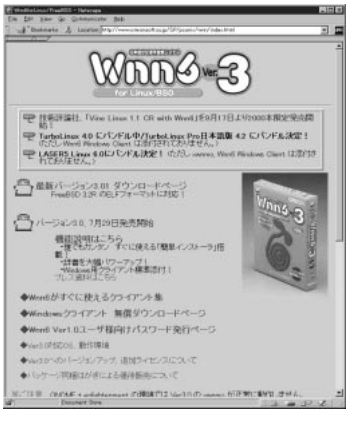
http://happypenguin.org/news/

タイトルの通り「Linuxでゲームしよー！」というサイト。近頃のLinuxは最初から当然のようにXが立ち上がるので、カラフルなゲームが簡単に楽しめる。ここで紹介されているXGalagaなんて説明不要のあのゲームだ。1つ1つのゲームを複数人でレビューし、星なんかつけちゃっている。WindowsからLinuxに移ったけど結局はゲームをしている自分にハタと気づいてみたりする。もちろんコンソールものもあるぞ。

Japanize

Wnn6 Ver3.0 for Linux/FreeBSD

http://www.omronsoft.co.jp/SP/pcunix/wnn/index.html



FreeWnnと共にUNIX上での日本語入力システムとして定番なのがWnn6だ。ここはその発売元であるオムロンソフトウェアのオフィシャルサイト。日本語化されたLinuxディストリビューションとして定評のあるVine Linuxだが、Wnn6をバンドルした限定版、Vine Linux 1.1CRWが発売されMLなどでも大きな話題となった。また、先頃行われたマイナーバージョンアップにともない、追加モジュールが同サイトから無償でダウンロードできる。