Linuxを革新するビジュアル開発ツール「Borland Kylix」日本語版発売

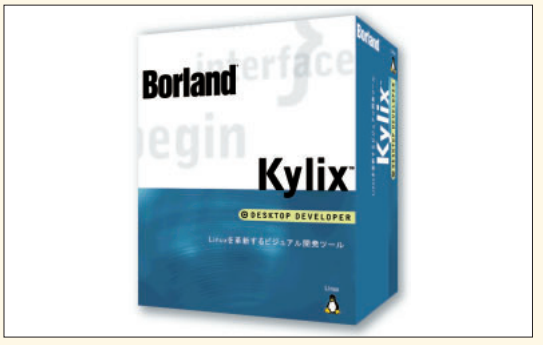
ボーランドは、Linux初の本格的ビジュアル開発環境「Borland Kylix」日本語版を、5月18日より発売する。価格は、アプリケーション開発環境「Kylix Desktop Developer」が12万円、サーバ開発環境「Kylix Server Developer」が24万円。なお、9月20日までの期間は、Kylix発売記念キャンペーンとしてボーランド製品を初めて使うユーザーにDesktop Developerを6万8000円、Delphiユーザーには3万2000円～3万8000円の優待価格で提供する。

Kylixは、Windows用の「Borland Delphi」をLinux上に移植したもので、開発言語に「Object Pascal」を採用している。ビジュアル開発とコーディングとの間の完全な双方向性を実現した2Way-Tool、ソースレベルでのデバッグ環境、Linux専用のネイティブコードコンパイラなどの特徴を備えている。

Kylixは、CLX（Component Library for Cross Platform）によってWindowsとLinux間で高い移植性を実現した。Apache用モジュール開発のためのNetCLXコンポーネントを利用して、Webサーバアプリケーションをコンポーネントベースでビジュアルに開発できる。また、Oracle8iやIBM DB2などのRDBMSへの接続が可能な「dbExpress」が提供される。

これらを用いることで、Webやデスクトップ、データベースに対応した高速なLinuxアプリケーションを簡単に開発し、短期間で運用・配布することが可能になる。

なお、無償ダウンロード版「Open Edition」の提供も予定されている。



発売日	2001年5月18日
発売	ボーランド株式会社
TEL	03-5350-9380
価格	12万円～
URL	http://www.borland.co.jp/

News Headline from 日刊アスキーLinux

www.Linux24.com

ジャスト、英Tao、リネオ、組み込み型LinuxベースのJava環境で協業

2001年4月16日

ジャストシステム、英Tao Group社、リネオは16日、組み込み型Linuxをベースとした Java環境で協業すると発表した。これにより、日本市場向けに情報家電やPDA向けのリファレンスプラットフォームを提供する。

同プラットフォームは、リネオの組み込み型Linux「Embedix」に、Taoの組み込み型マルチメディアJava環境「intent」を搭載し、ジャストシステムの組み込み用日本語変換システム「ATOK Pocket」を統合したもの。StrongARMプロセッサのリファレンスボードを使用して実現している。

ジャストシステム
(http://www.justsystem.co.jp/)
リネオ（http://www.lineo.co.jp/)
Tao Group（http://www.tao-group.com/)
アスキーintentホームページ
(http://www.ascii.co.jp/tao/)

組み込み用Linuxに開発環境、開発ボードまでバンドル

2001年4月12日

米Lineoと米Metrowerksは、San Franciscoで行われたEmbedded System Conference（4月9日～13日）にて、「CodeWarrior for Embedix Development System for the PowerQUICC II MPC8260」の出荷を発表した。

この製品は、MetrowerksのCodeWarrior for Embedixに、あらかじめ動作環境を設定したMotorolaの通信用マイクロプロセッサPowerQUICC II MPC8260（以下MPC8260）と、Lineoの組み込み用Linux開発環境であるEmbedix SDK、開発プロセスの自動化を支援するTarget Wizardといった、組み込みLinux用ソフトウェア開発に必要なハードウェアとソフトウェアがすべてバンドルされたもの。WindowsまたはLinuxをホストとして開発を行う。

Lineoは、組み込み用Linuxディストリビューションの Embedixで有名。また、一方のMetrowerksは、マルチプラットホームの統合開発環境であるCodeWarriorを開発／販売している。開発ボード上に乗るMPUのMPC8260は、インターネットのインフラやテレコミュニケーション、ネットワーク市場で用いられており、組み込み型PowerPCコアであるPC603eとコミュニケーションプロセッサモジュール、システムインターフェイスユニットが1つのチップに集約された製品で、1998年に発表されている。

両社によると、この製品ではあらかじめフラッシュメモリなどが設定されているため、デベロッパーは開発環境の設定に関わる手間を省くことが可能になり、ソフトウェアの開発に専念することができるので、開発サイクルの効率化とコスト低減を実現できるという。

リネオ（http://www.lineo.co.jp/)
メトロワークス（http://www.metrowerks.co.jp/)
Embedded System conference
(http://www.esconline.com/)

Linux Conference 2001を9月に開催 論文を大募集!

2001年4月11日

日本Linux協会は、9月26日から28日までの3日間、明治記念館にて「Linux Conference 2001」を開催すると発表した。同イベントは、開発者や技術者を中心としたコミュニティと、Linuxビジネスのキーマンが集合する、今年で5回目となるデベロッパーカンファレンスだ。

今回は、すべてのオープンソースソフトウェアを対象とし、多くの研究開発者、技術者、企業による先端テクノロジーを集結したいということで、論文やプレゼンテーションの公募を幅広く行う。優秀な論文には、“Linux Conference Award”が贈られ、商業誌への掲載、賞金および副賞などが用意される。

カンファレンスは、以下の4つのテーマのトラックで開催され、各トラックごとに論文を募集する。

- プラットフォーム
- エンベデッド
- エンタープライズ
- デスクトップ

募集する論文は、フルペーパー（カンファレンス発表原稿）またはアブストラクトの2種類。6月18日までにlc-submit@linux.or.jpへ送付とのことだ。詳しくは「Linux Conference 2001 論文募集」ページをご参照いただきたい。

日本Linux協会（http://jla.linux.or.jp/)
「Linux Conference 2001」開催告知ページ
(http://lc.linux.or.jp/lc2001/)

HDEと日本ベリサイン、セキュリティソリューション分野で提携

2001年4月4日

ホライズン・デジタル・エンタープライズと日本ベリサインは、セキュリティソリューション分野で提携したと発表した。「HDE Linux Controller」と「ベリサイン・グローバル・サーバID」を組み合わせ、eコマース事業者向けに提供する。

HDE Linux Controllerに、ベリサイン・グローバル・サーバIDを取得する際に必要となるCSR（Certificate Signing Request：証明書署名要求）の発行・申請機能を付加する。これにより、サーバの運用管理からセキュアなWebサーバの立ち上げまでをWeb上で行うことが可能となる。

両社は今後、eコマース事業者を顧客に持つiDCを対象に、HDEのiDC向けサーバ管理システム「HDE Management

Suite for iDC」と組み合わせたサービスを展開していくという。

ホライズン・デジタル・エンタープライズ（http://www.hde.co.jp/）

日本ベリサイン（http://www.verisign.co.jp/）

北陸朝日放送のECサイト「金沢屋.com」、Turbolinux Serverで構築

2001年4月3日

ターボリナックス ジャパンは、北陸朝日放送のECサイト「金沢屋.com」のWebサーバとDBサーバにTurbolinux Serverが採用されたと発表した。4月25日より稼動する予定。システム構築はカルテックが担当した。

OSにTurbolinux Server日本語版6.0を搭載したIBMのNetfinity 4500R上で、Webアプリケーション＆DBサーバにロータス ドミノR5、ドミノ専用のEC構築ツールsmartEC for Domino R1.1を使う。決済機能はNICOSゲートウェイサーバを利用する。

今回、Turbolinuxが採用された理由としては、コンテンツのメンテナンスやECサイト事務局の運営を行う制作プロダクションに、Linuxでのサイト運用実績があったことなどが挙げられるという。

カルテックは、CTIやグループウェア構築を得意とするソフトウェアハウス。昨今はe-ビジネス市場全般へと事業を拡大している。金沢屋.comは石川県内の特産品やさまざまな情報をインターネット上で紹介、販売するサイト。

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

カルテック（http://www.quartech.co.jp/）

金沢屋.com（http://www.kanazawa-ya.com/）

LinuxとWindowsの両方に感染するウイルス「Winux」発見される

2001年3月30日

米国のアンチウイルスソフトベンダーCentral Commandは、LinuxとWindowsの両方に感染するウイルス「Winux」（別名Lindose）の存在を報告した。同社によると、Winuxはメモリに常駐はせず、電子メールやLAN経由で、Windows PE（Portable Executable）ファイルやLinux ELFファイルに感染する。詳しくは以下のとおり。

Windowsの場合、PEファイルの中から、「.reloc」形式のものを選んで上書きする。また、API機能を使って他のファイルにも感染する。

Linuxの場合、ELFファイルのエントリーポイントにある命令を上書きし、それをファイルの最後に格納する。感染したファイルが実行されるとき、オリジナルのコードに戻る前に、再び感染を広げる。

Winuxには次のテキストが含まれている。

“[Win32/Linux.Winux] multi-platform virus by Benny/29A”
“'This GNU program is covered by GPL.”

感染の拡大が危惧されるが、ウイルス専門家によると、Winuxは電子メールアドレスから自動的に伝染しないため、大きな被害を及ぼす可能性は小さいとのことである。Central Commandは、すでに同社のウイルス対策プログラム「AVX Professional」に対応したWinux除去プログラムを開発しており、同社のサイトで公開している。

米Central Command（http://www.avx.com/）

米Borland、InterBase6.0英語版を出荷開始

2001年3月27日

米Borlandは、Linux、Windows、Solaris上で動作するRDBMS「Borland InterBase 6.0」を3月13日より出荷開始した。主な機能は以下のとおり。

- 標準インターフェイスはANSI SQL、ODBC、JDBC
- ボーランドの他のツールとの緊密な統合
- シングルユーザーから数百ユーザーまでサポート
- Linux、 Windows 2000、 Windows NT、Solaris、Novell、HP-UXなどでの互換性
- 容易なインストール、管理者不要な運用、省リソース性

SourceForgeのみからソースコードが入手可能な「Open Edition」と、ローカルデータベース用の「Desktop Edition」、JDBC準拠ドライバ“InterClient”の配布ライセンスやデータベースへの接続、レプリケーション、ドキュメンテーション、サードパーティツールを同梱した「Server Edition」の3種類が用意されている。

米Borland（http://www.borland.com/）

東京アプリケーションシステムの社内情報管理システム、Turbolinux Serverで構築

2001年3月27日

ターボリナックス ジャパンは、国内5拠点でSI事業を展開する東京アプリケーションシステム（以下TASC）の新しい社内情報管理システムが、Windows NTおよびNetWareベースのシステムから、Turbolinux ServerをOSとするシステムに移行したと発表した。同社の社内組織が、各拠点中心から事業部制に変更したのを機に、分散していた情報の一元管理と、各拠点からアクセスできる情報サーバ構築の必要性が生じたという。

新しいシステムは、各種ドキュメントを保存／共有するファイルサーバ、ロータスドミノR5をベースに自社開発した、社内業務や品質管理を行う業務サーバと、ネオジャパンのiOffice2000 for Turbolinuxによるグループウェア／メールサーバで構成され、いずれもTurbolinux Serverを搭載したNECのExpress5800上で稼動している。

TASCでは、現在Windows 2000 Serverベースで構築している他の業務システムについても、Apache、PostgreSQL、PHPなどによるTurbolinuxベースのシステムに移行する予定。ターボリナックスジャパンのビジネスパートナーでもある同社は、今回の社内システム構築の経験／ノウハウを活用して、TurbolinuxベースのSI事業ならびにサポート／教育を積極的に展開する計画だという。

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

東京アプリケーションシステム（http://www.tasc.co.jp/）

Hardware　発売日　2001年5月上旬

高い安全性をコンセプトにしたデュアルサーバ Dual Concept

URL　http://www.duaxes.com/

発売	デュアキシズ株式会社
TEL	03-3523-6933
価格	260万円〜

デュアキシズは、独自監視システムを使用したサーバ二重化により耐障害性、信頼性を高めたアプライアンスサーバ「Dual Concept」を5月上旬に発売する。

価格は、二重化の方法により異なり、ホットスタンバイ型が260万円、リアルスタンバイ型は380万円。ホットスタンバイ型は切り替え時にデータのリンクが一度切れるが、リアルスタンバイ型はリンクが切れないためクライアント側に影響が出ない。

4Uラックマウントサイズの筐体にサーバを2台格納、同社独自のフェイルセーフ機能により、短時間で異常の検出と切り替えを行い、処理を継続する。

プラットフォームには独自改造Linuxを採用し、サーバの設定は付属のセットアップユーティリティを使用してGUI画面で行う。1台分の導入価格で二重化が可能なため2台のサーバを導入するより安価になっている。Pentium III 750MHz×2、メモリ128Mバイト（最大512Mバイト）×2、20Gバイトハードディスク×2、24倍速CD-ROM×2、1.44Mバイトフロッピードライブ×2を搭載している。

Hardware　発売日　2001年4月2日

オールインワン・ファイアウォールサーバ ディジタルスター

URL　http://www.digitalstar.ne.jp/

発売	株式会社アイアイティーヴィー
TEL	0120-00-8611
価格	オープンプライス

アイアイティーヴィーはオールインワン・ファイアウォールサーバ「ディジタルスター」を4月2日より発売した。

Celeron 466MHz、メモリ 64Mバイト、20Gバイトハードディスクを搭載。100BASE-Tポートを3個装備し、LAN、WAN、DMZにそれぞれ接続する。筐体のサイズは、340（W）×300（D）×84（H）mm。

DNSサーバ、Webサーバ、メールサーバ、FTPサーバ、DHCPサーバ、ファイアウォール機能を持つ。サーバの基本設定と管理はクライアントのブラウザ上からGUIで操作できる。ポータルサイトにアクセスしてバージョンアップを含めたオンラインサポートを受けられる。管理ツール、サーバアプリケーションやクライアント用ツールなどのダウンロードが可能（一部有料）。オンラインサポートを受けるには同社への申し込みが必要。

Hardware　発売日　2001年4月6日

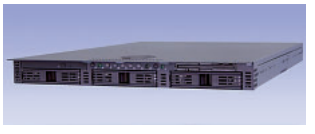
Miracle Linuxインストールサーバ Vintageラックマウントサーバ

URL　http://www.jcsn.co.jp/

発売	株式会社日本コンピューティングシステム
TEL	03-3821-3200
価格	130万円〜

日本コンピューティングシステムはRDBMSのOracle8i Workgroup ServerのCPU無制限アクセスライセンスとMiracle Linuxをプリインストールしたサーバマシン「Vintageラックマウントサーバ」を5月31日までの期間限定で販売する。

シングルCPUとデュアルCPUの2機種4モデル。Pentium III 1GHz、HDDを2台まで搭載可能なのがSystem1と System2で、 Pentium III 866MHz〜1GHz、HDDを3台まで搭載可能なSystem3とSystem4が用意されている。System2とSystem4がデュアルCPU。いずれも1Uラックマウントサイズで、メモリ128Mバイト（最大4Gバイト）18.4GバイトUltra160 SCSIハードディスク（最大36.7Gバイト）、24倍速CD-ROMを搭載している。シングルCPUのマシンが130万円、デュアルCPUのマシンが220万円。



Hardware　発売日　2001年4月1日

LinuxベースOS搭載PDA AgendaVR3デベロッパーエディション

URL　http://www.attic-jp.com/

発売	株式会社コシダテック
TEL	03-3435-7071
価格	オープンプライス

コシダテックは米Agenda Computing Inc.が開発・販売している、LinuxベースOS搭載PDAの開発者向けバージョン「AgendaVR3デベロッパーエディション」を4月1日より発売した。

32ビット66MHzのMIPSプロセッサとメモリ8Mバイトと16Mバイトのフラッシュメモリを搭載。複数のアプリケーションを同時使用できるマルチタスクを実現した。160×240ピクセル16階調グレースケールの液晶画面で手書き認識入力方式を採用している。

今回はエンジニアや開発者向けの限定発売で、開発途中のソフトウェアや未リリースのソフトウェアを含む。コシダテックでは、国内開発者用にメーリングリストを立ち上げ、Webでの日本語による情報提供を行い、開発者コミュニティを支援する。

Hardware

発売日　2001年4月3日

30万円を切る低価格NAS製品 PowerVault 705N

URL　http://www.dell.com/jp/

発売	デルコンピュータ株式会社
TEL	044-556-6190
価格	29万8000円～

デルコンピュータはストレージ製品「PowerVault」シリーズのNAS（Network Attached Storage）新製品「PowerVault 705N」を4月3日より発売した。

高さ1Uサイズ（約4.4cm）で、160Gバイトモデル（IDE 40Gバイト×4）と240Gバイトモデル（IDE 60Gバイト×4）の2種類。LANに接続して初期設定を行うだけでファイルサーバなどを介さずに同一LAN上でデータの共有が可能になる。

データの信頼性を高めるRAID 5、大容量ストレージを実現するRAID 0、ミラーリング機能のRAID 1をサポート（工場出荷時設定はRAID 5）。

異機種混在環境下でのデータ共有をサポート。OSはWindows 95／98／2000／NT、Sun OS、Solaris、SCO UNIX、Macintosh System 7.5.5／7.6、Mac OS 8.X／9.X／X、Red Hat Linuxに、ネットワークプロトコルは、TCP/IP、IPX、NetBEUI、AppleTalkに、ファイル共有プロトコルはNFS（UNIX）、CIFS/SMB（Windows）、AFP（Apple）、HTTP1.0に対応している。

Hardware

発売日　2001年4月

Linuxマイクロ・アプライアンスサーバ FutureNet MA-360

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	5万円（サンプル価格）～

センチュリー・システムズは、Linuxマイクロ・アプライアンスサーバ「FutureNet MA-360」を4月よりOEM向けとして発売した。

FutureNet MA-360はIBM製PowerPC 405GP 200MHz（最大266MHz）、メモリ64Mバイト（最大128Mバイト）、イーサネット100BASE-T×4ポートHUB、5.5GバイトIDEハードディスク（最大36Gバイト）を搭載。PowerPC 405GPに移植したカーネル2.4.0を使用したLinux OSを使用。筐体のサイズは、250（W）×180（D）×53（H）mm。ほかにISDN内蔵タイプ（TA＋DSU内蔵）がある。

ASPが提供するアプリケーション／データ保管用サーバとして、あるいは、特定のソフトウェア専用サーバ、遠隔からLANを監視するサーバなどの利用を見込んでいる。価格は、サンプル価格で、OEM用に販売する価格はそれよりも安くなる。

Software

発売日　2001年4月19日

ブロードバンド・メディア・ストリーミング製品 Kasenna MediaBase

URL　http://www.sgi.co.jp/

発売	日本SGI株式会社
TEL	0120-161-086
価格	184万8000円～

日本SGIは、Linux OSサーバ用のストリーミング・ソリューションである「Kasenna MediaBase 4.0.1」を4月19日より発売した。

価格には、100ストリームのライセンスが含まれている。製品はブロードバンドにも対応したビデオコンテンツの配信を可能にしている（転送レート8kbps～19.4Mbpsに対応）。

対応しているストリーミングメディアフォーマットは、MPEG-1、MPEG-2、MPEG-4、MP3、QuickTime、RealVideo、RealAudio（G2サーバ使用）、Windowsメディア（Windows NTのWindowsメディアサーバを使用）。対応するOSは、Red Hat Linux 7.0以上。

システム管理はブラウザ（Netscape、Internet Explorer）から行う。

Software

発売日　2001年4月23日

iモード／EZweb／J-SKYなどからWebにアクセス @MESSAGE-Mobile

URL　http://www.3rsoft.com/jp/

発売	スリーアールソフト株式会社
TEL	03-5330-8851
価格	100万1500円～

スリーアールソフトは、同社のメールサーバ製品、@MESSAGE-MailStudioをベースに、携帯端末に対応するメールサーバソフト「@MESSAGE-Mobile」を4月23日より発売した。

価格は1ユーザーから10000ユーザーまでは、初期費用100万円と1ユーザー当たり1500円、それ以上のユーザーに対しては初期費用500万円と1ユーザー当たり980円で提供している。

@MESSAGE-MobileはLinux、Solarisなどの OS上で動作し、PDAや携帯電話からのアクセスを可能にするため、小型携帯端末の異なる言語（iモード：cHTML、EZweb：HDML、WML、J-SKY：MHTML）を1つのシステムでサポートしている。これによって、携帯端末を用いたメールの送受信だけでなく、サーバ側のアドレス帳、スケジューラ、掲示板への検索や書き込みも可能になった。

Software　発売日　2001年7月予定

Webインターフェイスを使ったカード型データベース サイボウズDBメーカー

URL　http://cybozu.co.jp/

発売	サイボウズ株式会社
TEL	03-5805-9035
価格	未定

サイボウズはWebブラウザで操作するサーバ・クライアント型のカード型データベース「サイボウズDBメーカー」を7月に発売する。

「サイボウズDBメーカー」はグループウェアの「サイボウズOffice」のデータベースエンジンを利用して、複数のユーザーがカード型データベースを使って情報共有を行うためのツールで、設定や操作の簡単さが特徴。1つのデータベースにつき10万件程度までのデータを扱える。対応OSはWindows NT／2000、Linux、FreeBSD（以上Intel系CPUのみ）、Solaris（SPARCのみ）で、IIS、Apache、サイボウズWebサーバー2などのWebサーバが稼働している必要がある。

価格体系は通常のソフトウェアと違い、データベース数×利用期間で課金されるシステム。サポート、バージョンアップ費用などもこれに含まれる。クライアント数は無制限。ベータバージョンが同社Web上で7月31日まで公開されている。

Software　発売日　2001年5月7日

1時間に100万通のメールを配信 HDE Lightning Messenger 1.0

URL　http://www.hde.co.jp/

発売	株式会社ホライズン・デジタル・エンタープライズ
TEL	03-5738-5410
価格	1500万円〜

ホライズン・デジタル・エンタープライズは、クラスタリング構成により1時間に最大100万通のカスタマイズメール（個々に内容が異なるメール）を配信できるメールサーバ「HDE Lightning Messenger 1.0」を発売した。

Lightning MessengerはオラクルやPostgreSQLなどのデータベースから指定した条件にあてはまるデータを取り込み、1通ごとに異なるメールを作成、指定した日時に自動送信する。また、エラーで戻ってきたメールも自動処理される。データ抽出条件、文面作成などのオペレーションはWebベースの簡単なインターフェイスで行える。

対応OSはRed Hat Linux 6.2J、Turbolinux Server日本語版6.1、Solaris（対応予定）で、最小ハードウェア構成は6台から。6台のうち管理やデータベースとのインターフェイスに1台、ロードバランサに1台、メール送信に3台、エラー処理に1台が割り振られる。

Software　発売日　2001年4月20日

Webメールやiモードにも対応する多機能メールサーバシステム CyberMail

URL　http://www.cybersolutions.co.jp/

発売	サイバーソリューションズ株式会社
TEL	03-5542-2710
価格	50万円〜

サイバーソリューションズは、Linux、FreeBSDで稼動するサーバに対応した、メールサーバシステムCyberMailを4月20日より発売した。

価格は100アカウントで50万円から。ただし、10アカウントまではフリーライセンス。

1サーバで20万アカウントを管理できる。1サーバでもマルチドメイン管理ができ、ドメインごとにサーバを立てる必要がない。20万アカウントを超える場合には、マルチサーバ機能により分散構築が可能。Webブラウザからユーザーアカウントの登録／システム管理が行える。Webメールの画面（HTML）のカスタマイズも可能。

稼働環境は、PentiumIII（450MHz以上推奨）、メモリ256Mバイト（512Mバイト以上推奨）。対応しているWebブラウザは、Microsoft Internet Explorer 4.0以上、Netscape Communicator 4.0以上。

Software　発売日　2001年4月10日

組織内のインターネットアクセス状況を分析するソフト Mind Scope Ver 1.0

URL　http://www.networld.co.jp/

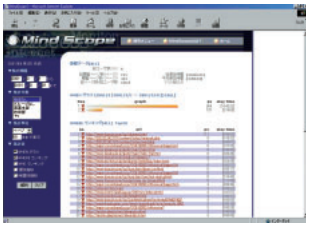
発売	株式会社ネットワールド
TEL	03-5210-5081
価格	77万円

ネットワールドは、組織内のインターネットアクセス状況を個別に収集・分析できる新製品「Mind Scope Ver 1.0」を4月10日より発売した。

本製品は、社内など組織内でのインターネット使用状況を、所属別、個人別、年月日別などでアクセス時間、アクセス先などの情報を収集・分析して情報の有効活用をサポートする。データはCSV形式で出力できる。

CDにはOS（Turbolinux FTP版）が含まれており、OSとMind Scopeの両方を同時にインストールできる方法を採用している。

価格は、ユーザー数とは無関係。1年間無償のオフサイト・サポートとバージョンアップの権利が付いている。次年度からのサポートは有償で12万円。動作環境は、CPUにPentium相当以上、メモリ128Mバイト以上、ハードディスク20Gバイト以上が必要。



Software　　発売日　2001年4月25日

Java／XMLアプリケーションサーバ Lutris Enhydra 3.5 日本語版

URL　http://www.necsoft.co.jp/soft/enhydra/

発売　NECソフト株式会社
TEL　03-5569-3399
価格　19万8000円〜

NECソフトは、米Lutrisのアプリケーションサーバ製品の日本語版、「Lutris Enhydra 3.5 日本語版」を4月25日より発売した。

Lutris Enhydraは、マルチOS対応のJava／XMLアプリケーションサーバ。3.5版では、iモードなどの携帯端末向けに強化されている。

価格は「開発キット」が19万8000円、「運用ライセンス」が24万8000円。5月末までは、キャンペーン価格で「開発キット」が15万8400円、「運用ライセンス」が19万8400円。開発キットは、開発者単位で、運用ライセンスはCPU単位の価格となる。

Lutris Enhydraは、Java2に完全対応。データベースInstantDB 3.25が付属している。同梱されているサードパーティ製品のうち、Forte for Java release 2.0、Community EditionおよびPixo インターネットマイクロブラウザは日本語化されている。WebサーバはApache、iPlanet、Netscape、Microsoft IISに対応している。

Software　　発売日　2001年5月1日

Webシステムに組み込み可能なPDF帳票出力システム WebReportCraft

URL　http://www.10art-ni.co.jp/

発売　株式会社テンアートニ
TEL　03-5298-2924
価格　60万円〜

テンアートニは、既存のWebシステムに組み込んでPDF形式での帳票出力を可能にする「WebReportCraft」を5月1日より発売した。

製品は、データベースクエリと帳票設計を担うReportDesigner、PDF出力エンジンのReportEngine、レポートサービス機能、運用サービスや管理を担うReportServiceの3コンポーネントで構成される。

ReportDesignerが帳票レイアウト定義をXMLで出力し、ReportEngineがそのXMLファイルを読み込みPDFファイルを出力する。

ReportEngineとReportServiceはJavaで動作し、ReportDesignerはクライエントのWindowsマシン上で動作する（Windows 2000、Windows NTに対応）。

ReportServiceを使うには、同社フレームワーク製品のWebWorkBench Enterprise版およびBorland VisiBroker 4.x for JavaまたはBorland AppServer 4.5が必要。レポートサービス機能を提供するCORBAオブジェクトとして分散環境を構築できる。

Software　　発売日　2001年5月29日

Linuxに対応したBtoBサーバソフト Asteria for RosettaNet

URL　http://www.infoteria.com/

発売　インフォテリア株式会社
TEL　03-5718-1260
価格　2400万円〜

インフォテリアは情報機器や電子部品業界の標準ビジネスプロトコルであるRosettaNetに対応したBtoBサーバソフト「Asteria for RosettaNet」を5月29日より発売する。

Asteriaはプラットフォームや固有のアプリケーションに依存せず、テキストデータを柔軟に扱えるXML技術を基盤にしており、同様にXML対応しているRosettaNetにはビジネスプラグインで対応する。

Asteria for RosettaNetはサーバソフトウェアとしてのアステリアにRosettaNetを組み込んだ製品。社内外の取引の特殊事情をAsteria独自の抽象化したAPIを使って従来の手続きにできるだけ変更を加えずにBtoB取引を自動化できるようにしている。Asteriaは扱うXMLデータをRDBに接続するためのiConnectorというミドルウェアを同梱しており、これによってRDBを中核とするシステムの構築も可能。すでにSolarisに対応した製品が発売中で、今回Linux環境に対応した。

WebSphere 2001 Tokyo-Mission to WebSphere- -WebSphereスペシャリストと話そう-

http://ibm.com/jp/websphere/2001/

次世代eビジネスを支える、IBMのWebSphereソフトウェア・プラットフォームの最新情報についてのセッションと、デベロッパー仲間が交流するコンファレンス「WebSphere2001」が開催される。

今回のテーマはMission to WebSphere。J2EE、EJB、Webサービスなど計42セッションを公開。フレームワーク、コンポーネントデザインを学んでWebSphere V4の最新情報を得るチャンス。IBMのWebSphereスペシャリストとの交流を目的としたWebSphere night、問題解決の糸口を与えてくれるかもしれない、レスキュー2001などが予定されている。日時と場所については下記のとおり。詳細はWebサイトにて。インターネットからも申し込める。

日時：5月30日（水）〜6月1日（金）
場所：ヒルトン東京ベイ（舞浜）
主催：日本IBM株式会社
登録料：1万8000円（3日間）

日刊アスキー Linux Business Report

エンドユーザー市場の可能性

http://www.linux24.com/

最近、Linux関連企業が発信するニュースリリースは、サーバ製品やIT関連サービス、組み込み機器関連といったビジネス市場向けのものが中心で、「Linuxビジネスはサーバ関連がメイン」「Linuxビジネスはハイエンドかアプライアンスに向かっている」という雰囲気もある。エンドユーザー向けビジネスは、基本的に「数」が勝負となる。しかしLinuxユーザーの数は、Windowsユーザーと比べればはるかに少ない。各ディストリビューターの、エンドユーザー向けビジネスの舵取りはやさしいものではないだろう。

そんな中、最近エンドユーザー向けサポートを開始した「Omoikane GNU/Linux」のオモイカネ株式会社に話を聞いた。オモイカネは、一番効率が良いと思われるSI的なビジネスから、エンドユーザー向けビジネスにも範囲を拡大するという、逆行的とも思える行動をとっている。その意図とは何なのか、Linuxのエンドユーザー向け市場にどのような分析を行ったうえで参入を果たしたのか、同社代表取締役・大熊但由氏に話を聞いた。

エンドユーザーパッケージ市場の規模は変化していない

――まず、現在のエンドユーザーパッケージ市場について、どのようにお考えでしょうか？

大熊氏：Linuxのパッケージ市場は、Linxuビジネスが始まって以来市場規模が変化していないと思います。つまり、「ハコ売り」は変化がないということです。ハコ売り事業を続けてきたディストリビューターさんも、マーケティングに手を尽くしてもあまり変化がない、という状況が見えてきているのではないかと思います。ほかのディストリビューターさんは、わりと大きな市場を想定して事業を進めて来たのではないかと思いますが、ハコ売りの市場というのは、実は固定された範囲でしかありません。また、大きな流れとして、ネットバブルが崩壊する前、大手のディストリビューターさんはある程度資本を獲得できました。オモイカネは後発ですので、そのペースには乗っていません。我々は、開発スタッフも限られていて、価格もオンライン直販だけということもありますが、他のディストリビューターさんと比べると軽量なパッケージで価格を安く設定しているわけです。ハコ売りとは少しランクが違います。そこにつけ込んでみようかと思いました。実際、そんなにビジネスにならないのではないか、というところを、逆に狙っていこうかなと思っているのです。1～3万円のパッケージのニーズというのは市場が固定化していますが、1000円、2000円といった、音楽CDの感覚で配布されて、中身も市販ソフトがそんなに入っていない、手軽に使える、といったところで広がる可能性は、まだまだあります。

コンビニエンスストア店頭売りと同じ感覚で

――コンビニエンスストア店頭で販売されている、低価格なPCゲームパッケージのようなイメージでしょうか？

大熊氏：そうです。もちろん将来的には、Windowsの代替も狙えるわけですし。KDEやGNOMEなど、デスクトッ

基本的にサポートはWeb上でのみ行う。フォームに入力をして質問すると、回答内容が登録されたメールアドレスに返ってくる仕組みだ。

プの標準が確立して、開発ツールも普及すれば、アプリケーションの移植も進むのではないでしょうか。オモイカネという会社は、もともと受託開発がメインですので、Windowsのアプリケーションベンダーに対して「Linux版を出しましょう」と積極的に持ちかけていきたいと思っています。

――では、いままでの「Linuxエンドユーザー市場」というものとはちょっと別の所を狙っていらっしゃる？

大熊氏：今回の場合はそうですね。そういう意味では、市場調査をしつつ、ある程度利益を得ていく、という考え方で進んでいます。したがって、強力に「狙って」やっているわけではありません。「1000円で100万本売る」という状況でもないですから。もっとも、Debian GNU/Linux系でエンドユーザーサポートを行っているところがないというのも理由のひとつです。ビジネスをしている方からも、「ソフトウェアの評価が高くてもサポートがないと……」というお話を聞くこともありましたので。さらに、我々が「OS屋さん」という振る舞いをしても、今日のディストリビューター乱立の状況では、目立つのは難しいでしょうから、ほかとは違う方向を目指したわけです。

――「OS屋さん」とおっしゃいましたが、どういった定義でしょうか。OSの市場力を軸にしてビジネスを展開していくという形ですか？

大熊氏：そうですね。「OS屋さん」として振る舞ったほうが、周囲も付き合いやすい。ハコを買ってくればそれで使えるわけですし、今までの習慣を変えなくてもいいので。

企業にオープンソース戦略をアドバイス

――そこでオモイカネさんは違う方向でやっていこうと？

大熊氏：はい。我々は当初から「技術サポートのみの会社である」ということです。だから普通のサポートだけでなく、もっと高いレベルのサポートも行います。たとえば、我々はクライアントがオープンソースを自社のシステムに取り込みたい、使いたいというときに、間違いがないようにアドバイスすることも可能です。よくある間違いとしては、「オープンソースが好きだ」といって自社の大切なリソースをフリーにしてしまうケースですね。これは会社にとっては損失なわけです。そこで、フリーにすべきか、してはいけないかのアドバイスを行います。オープンソースにする、しないというのは、微妙な判断だと思います。すでに市場流通を押さえている企業、市場を持っている企業が、その価値を高めたいから、リソースをオープンソースにしてつぎ込む、というのは当然の選択だと思いますが、中小が同じことをして何かを得るかというと、そうでもないと思いますね。我々は基本的には受託の会社なのですが、ほかの企業もオープンソースと付き合うノウハウというのはある程度求めていらっしゃるので、そういうところで付加価値を出してビジネスをしています。

――今のお言葉は誤解を招くかもしれませんね。「オモイカネの人たちはオープンソースにしないといい出す」と言われたりして。

大熊氏：そうですね（笑）。でも、結局エンジニアの作業がタダになるわけではないし、教育費も莫大なわけです。ビジネスの立場から言うと、会社が成り立たなくなったら意味がない。会社のリソースは会社のリソースだと。健康的にというか、健全に経営できたうえでオープンソースと付き合っていけば、それは必ず貢献できるはずなので、「無理なく着実に」というところでしょうか。そうはいってもうちの会社は基本的に全部オープンソースになっています。みなさんにお配りしたパンフレット（プレスミーティングの際の配付資料）も、GIMPやTeXで作っています。名刺もTeXなんですよ。

（聞き手：吉川大郎）

Column

オモイカネのエンドユーザー対応

オモイカネのエンドユーザー向けサポートのメニューには以下の4つがある。

①インストールサポート
②クライアントサポート：Linuxをクライアントとして使用する設定方法や情報
③センドバックインストールサポート：マシンを送付してもらい、Linuxをインストールして返送
④サーバサポート：サーバとして運用していくためのサポート

ユーザーは「ポイント」を購入し、上記メニューの難易度に応じてポイントを消費する。難しい種類のサポートだと、1インシデントの場合でも、高いポイントが必要となる。こうしたポイント制を採用することで、より柔軟な価格設定が可能になる。

「オモイカネでは、ソースコードレベルのサポートをやっていて、たとえばFreeBSDと連携したシステムで不具合が出た場合、FreeBSDのソースを調べたり、ということまで行っています。こうしたソースレベルのサポートからエンドユーザー向けのサポートまでを1つのサポート体系にまとめたかったのが、ポイント制を導入した理由です」（大熊氏）。

Distribution

新着ディストリビューション

Vine Linux 2.1.5

日本で高い人気を誇るVineに、最強のパッケージ管理ツールaptがDebianからポーティングされた。古いサーバアプリケーションがクラックの対象となるインターネットの世界では、こまめなアップデートがクラックから身を守る秘訣である。これからは、Vineユーザーも1日1回「apt-get update && apt-get upgrade」といこう。

HOLON Linux 2.0 Server

コマンド操作は不要。初心者にやさしいがウリのHOLONサーバでは、独自拡張されたWebminを使って、WebブラウザからLinuxを管理可能だ。QuickTimeデータなどをストリーミング配信するアプリケーションを収録するなど、ホロンらしい視点がいっぱいの作りになっている。

Omoikane GNU/Linux 1.2

デスクトップ環境で真に必要なもの。それは、Internet ExploreにひけをとらないWebブラウザと、ユーザーフレンドリーなメールクライントではないだろうか。Omoikane 1.2では、Webページを高速に表示できるようチューニングされたMozillaや、GUIメーラSylpheedを採用してデスクトップ環境を充実させている。Debian互換なOmoikaneは、もちろんaptを使ったシステムのアップデートが可能だ。

Vine Linux 2.1.5

日本語ディストリビューションとして高い人気を誇るVine Linux（http://www.vinelinux.org/）の最新バージョン2.1.5（以下、Vine 2.1.5）が3月24日にリリースされた。

Vine 2.1.5の基本システムは、glibc（2.1.3）とXFree86（3.3.6）のバージョンはVine 2.1と変わらず、カーネルはバージョン2.2.17から2.2.18にアップデートされた。

また、アップデートパッケージが取り込まれたり、PostfixやProFTPDを始めとするサーバ用アプリケーションが新しいバージョンに更新されたほか、XEmacsが削除されてEmacsのみの収録となった。

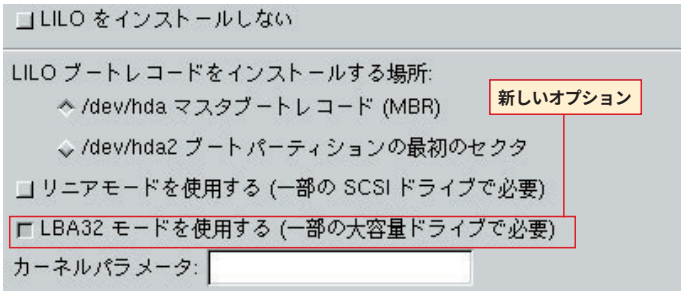
Vine 2.1.5はFTPや雑誌などでの配布のみで、製品版は販売されない。

8Gバイトの壁を超える

ブートローダLILOは、Vine 2.1でも大容量ディスクを扱うためのLBAモードをサポートしていたが、インストール中にLBAモードを設定できなかった。

このため、ハードディスクの先頭から8Gバイトを超える領域にVineをインストールする場合、一度レスキュー用のフロッピーディスクでVineを起動してからLILOを設定する必要があった。

しかし、Vine 2.1.5ではインストール中にLBAモードを選択できるように改良されたので（**画面1**）、LBAモードに対応したマザーボードを使えば、初回起動時から、LILOを使って8Gバイトを超えた領域にあるVineを起動できるようになった。



画面1　LBAモードの設定に対応したインストーラ［LBA32モードを使用する（一部の大容量ドライブで必要）］をチェックしておけば、8Gバイトを超える領域にあるVineもLILOで起動できる。

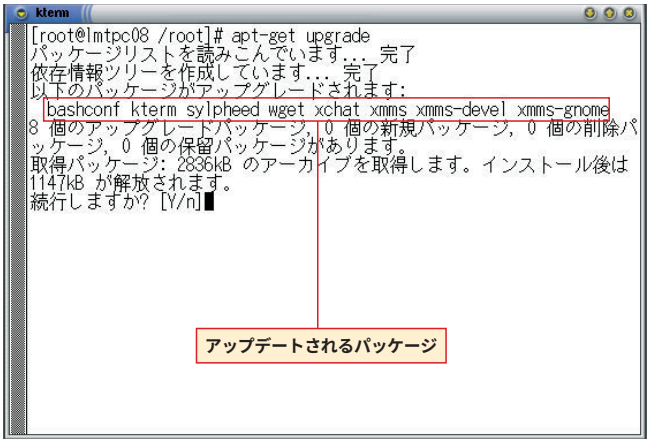
aptを使った簡単パッケージ管理

パッケージのインストールなどを行うrpmコマンドは、パッケージ間の依存関係をチェックするが、依存関係の解消までは面倒をみてくれない。

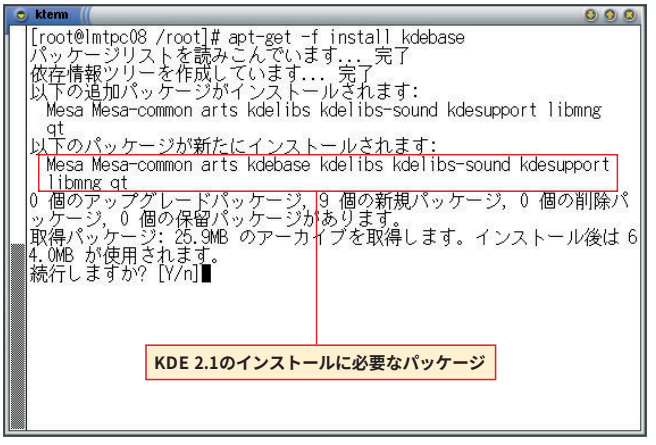
このため、rpmコマンドでパッケージをインストールする場合、依存関係の問題が起きると、初心者はお手上げになるケースが多かった。

Vine 2.1.5には、この問題を解決するaptという高機能パッケージ管理ツールが、新たにDebianから取り込まれた。

このaptを使えば、コマンド一発でシステムをアップデートしたり（**画面2**）、依存関係を考えることなくパッケージをインストールできるので（**画面3**）、依存関係の問題で悩むケースが減るだろう。ただ、aptを生かすも殺すもVineに用意されるパッケージの充実度次第なので、今後Vineユーザーからの積極的なパッケージのコントリビュートが期待されるところだ。



画面2　aptを使ったシステムのアップデート



画面3　aptを使ったVinePlusのKDE 2.1のインストール

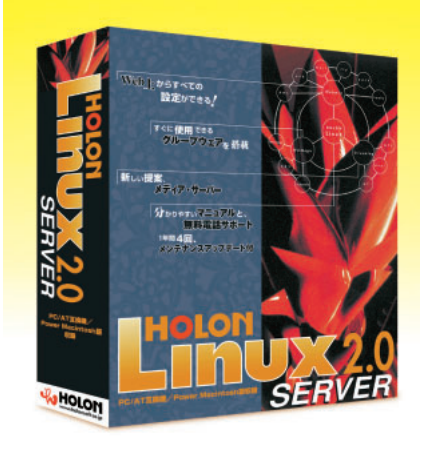
HOLON Linux 2.0 Server

初心者にやさしいつくりが特徴の、HOLON Linuxを開発するホロンが、サーバ版ディストリビューションHOLON Linux 2.0 Server（以下、HOLONサーバ）を5月11日に発売する。

HOLONサーバの基本システムには、カーネル2.2.18、glibc 2.1.3、XFree86 3.3.6が採用されている。デスクトップ用のHOLON Linuxと同じく、Linuxサーバ管理の初心者向けに工夫されているほか、Quick Time形式などのデータを配信するアプリケーションが収録されているのが特徴だ。

また、Debianから高機能パッケージ管理ツールaptが移植されているので、このツールを使って、常にシステムを最新状態に保つことが可能である。

HOLONサーバには、Intel版とPowerPC版の2種類のインストールCD-ROMのほかに、インストールや運用のためのマニュアルが3冊収録され、ユーザーは、90日間件数無制限のサポートを受けられる。



製品名　HOLON Linux 2.0 Server
価格　3万4800円
問い合わせ先　株式会社ホロン
03-5282-5101
http://www.holonlinux.com/

コマンド不要のサーバ管理

HOLONサーバには、ホロンが独自にカスタマイズした、オープンソースのLinux管理ツールWebmin（**画面1**）が付属する。

HOLON版Webminを使えば、ユーザーはコマンド操作することなく、新規ユーザーの作成やデーモンプログラムの起動と停止といった基本的なシステム管理、サーバデータのCD-Rへのバックアップのほか、SambaやNetatalkを使ったファイルサーバや、Majordomoを使ったメーリングリストなど、各種サーバアプリケーションの設定をWebブラウザから行える。

ホロン版Webminは、標準でSSLに対応しているので、リモートマシンからWebminを使ってLinuxを操作するときも、通信内容が暗号化されるので安全だ。

また、HOLONサーバに付属するグループウェアWebMagicを使えば、容易にスケジュール管理やアドレス帳などを共有できるので、部内やSOHOのサーバとして使う際に重宝するだろう。

画面1　HOLON版Webmin
ホロンによって拡張されたオープンソースの管理ツールWebminを使えば、コマンド操作することなくWebブラウザからLinuxを管理できる。

ストリーミングサーバ

HOLONサーバには、RealPlayer、QuickTime、MP3といったデータ形式に対応したストリーミングサーバアプリケーションが収録されている。

ユーザーは、作成した動画データなどを特定のディレクトリへ保存しておくことで、難しい設定なしにクライアントマシンからQuickTimeなどを使って、手軽に動画を楽しめる（**画面2**）。

動画配信をどんな用途に使うかはユーザー次第だが、デジタルビデオで録画した動画を、家族や友人で共有するといった使い方も、高速ネットワークが普及しつつある現状では考えられる活用方法だろう。

このように、サーバ管理から極力コマンド操作を取り除いたり、ストリーミングサーバといった新しいソリューションに取り込む姿勢は、ホロンならではと言ってよいだろう。

画面2　QuickTimeで再生中のサンプルムービー
標準で収録されているストリーミングサーバプログラムを使って、HOLONサーバから配信される動画をWindows版QuickTimeで再生している様子。

Omoikane GNU/Linux 1.2

Debian GNU/Linux（以下、Debian）をベースにするOmoikane GNU/Linux（以下、OGL）の最新バージョン1.2が、オモイカネ(http://www.omoikane.co.jp/)から4月3日にリリースされた。

OGL 1.2は基本システムに、カーネル2.2.18、glibc2.1.3、XFree86 3.3.6を採用している。

OGLは、基本的に安定版のDebianをベースにしているが、日本語対応を強化したGNOME 1.2、Xの設定ツール、デーモン起動のツールなどを収録して独自拡張しているほか、GRUB 0.5.96、modutils 2.4.2など積極的に新しいバージョンのパッケージを採用しているのが特徴だ。

また、OGLのインストーラは、ハードウェアの自動検出に対応しており、テキストベースながら非常にわかりやすい作りになっている。

OGLはDebianとの互換性が高く、aptを使ってDebianレポジトリから、システムをアップデート可能である。

ユーザーサポート開始

OGL 1.2は4種類のラインナップで構成される（**表1**）。

デスクトップ用のWorkStationとWorkStation++の違いは、収録されるパッケージ数にある。WorkStation++は、WorkStationにGNOME用のアプリケーションや、スクリプト言語Rubyなど多くのアプリケーションを追加している。

Serverは、Apache、Samba、OpenLDAPなどのサーバアプリケーションと、最低限のX Window Systemを収録するバージョンで、MTAに標準でqmailを採用するという特徴をもつ。

X-termはCD-ROMからブートして使うハードディスク不要のX端末用のバージョンだ。

Server、X-term、WorkStationの3つはFTPや雑誌で配布可能だが、バイナリとソースが2枚のCD-ROMに分かれるWorkStation++は、バイナリパッケージのみが配布されることを防ぐため、入手方法はWebでの通信販売のみとなる。

また、今回からポイント制によるユーザーサポートも開始された。サポートポイントを購入したユーザーは、インストール、ソフトウェアやサーバの設定、オモイカネにマシンを送ってインストール作業を依頼する、センドバックインストールといったサポートを、オモイカネのWebページにログインして利用できる。

チューニングされたMozilla

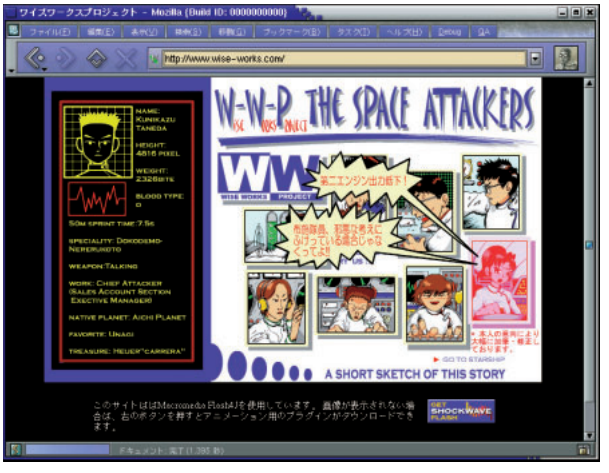
OGL 1.2には、オモイカネによりチューニングされたMozillaが標準のWebブラウザに採用されている。

一般的に、MozillaはWebページの表示速度が遅いと言われているが、OGL版のMozillaは、コンパイルオプションを調整することにより、表示能力が向上している。

また、Macromedia Flash用のプラグインが組み込み済みなので、Flashを使って作成されたWebページを問題なく表示できる（**画面1**）。

このほかにも、標準でSSLに対応しており、Netscapeに比べてバグの少ないMozillaの採用によって、快適なWebブラウズ環境を堪能できるだろう。

Sylpheed、Mew、Wanderlust（WorkStation++のみの収録）といったポピュラーなメーラも収録されているので、日本語デスクトップ環境を手っ取り早く構築したいDebianユーザーや、Debian初心者に特にお勧めできるディストリビューションだ。



画面1　デフォルトのWebブラウザMozilla 0.8.1
Flashを使って作成されたWebページを問題なく表示できる。オモイカネ独自のチューニングにより、Webページの表示速度も高速だ。

パッケージ名	概要
WorkStation	Webブラウズなど一般的なデスクトップ用
WorkStation++	WorkStationに多くのアプリケーションを追加（FTP配布されない）
Server	qmailやOpenLDAPなどを採用するサーバ用のパッケージ
X-term	CD-ROMから起動して使うX端末用パッケージ

表1　OGL 1.2のラインナップ
通信販売で購入する場合、WorkStation、Server、X-termの3つはCD-ROMが1枚で1000円、CD-ROMが2枚のWorkStation++は1600円だ。

Distribution ▶▶▶

カーネル2.4搭載のRed Hat Linux 7.1 5月11日発売

複数のパーティションを仮想的な1つのボリュームとして扱えるLVM、2Gバイトを超えるファイルサイズが扱えるext2ファイルシステム、USBサポート、同時実行可能なプロセス数の増加、マルチプロセッサ対応の効率化など、数々の新機能を携えたカーネル2.4搭載のRed Hat Linux 7.1は、すでに4月にFTP版が公開されているが、商用ソフトなどを加えた製品版がレッドハットより5月11日に発売される。

Red Hat Linux 7.1はXFree86 4.0.3を採用、BIND、Apache、プリントサーバなどにコンフィグレーションツールを用意、高速なWebサーバであるTUXを標準収録、新しいRPMやバージョンアップの情報を自動的に通知するシステムをサポートしている。

製品には「Official Red Hat Linux 7.1 Deluxe」（1万4800円）と同「Professional」（3万4800円）の2種類があり、両者の違いはサポート期間や商用アプリケーションの種類などとなっている。

各国語をサポートしたGNOMEもアップデートされている。

レッドハット株式会社（http://www.jp.redhat.com/）

Hard Hat Linux2.0リリース

MontaVista Softwareはカーネル2.4ベースの組み込み用Linuxの新バージョン、Hard Hat Linux2.0と開発ツールを発表した。カーネル2.4をサポートした組み込み用Linuxでは最初のリリースとなる。

Hard Hat Linux2.0は開発用プラットフォームとしてRed Hat Linux、Yellow Dog Linux、Solaris、Linux Mandrake、Turbolinux、Hard Hat Linux自身、Windows上のVMWareなどをサポート、対象アーキテクチャはx86、PowerPC、StrongARM、MIPS、日立SHとなっている。

Hard Hat Linux2.0にはC、C++をはじめとするコンパイラ、デバッガ、クロスコンパイラ、パフォーマンスモニタなどが含まれ、開発ツールには統合化開発環境Code Crusader、ソースコードツール、Linuxトレースツールなどが用意されている。

Hard Hat Linux2.0は開発用のProfessional Editionと評価用のJourneyman Editionがあり、後者はWebサイトやCDで無償提供される。

モンタビスタソフトウエアジャパン株式会社
（http://www.montavista.co.jp/）

各Linuxディストリビューター、ボーランドKylixとの協力関係を築く

本格的なビジュアル開発環境としてボーランドのKylixが注目を集めているが、各Linuxディストリビューターも早速協力関係を結んでいる。まず、ボーランドがKylixの動作検証済みディストリビューションを示す「Kylix Ready」プログラムを開始。すでに、ターボリナックス ジャパン、日本SCO（Open Linux）、ホロン、ミラクル・リナックス、メディアラボ（Linux MLD）、レーザーファイブ、レッドハットなどが同プログラムに参加している。

ミラクル・リナックスは「Miracle Linux with Oracle8i/Kylix」を発表。Miracle LinuxとOracle8i、Kylixの3つを同梱したパッケージで5月末出荷予定。価格はオープン。

また、テンアートニもKylixとRed Hat Linuxをカップリングした「Do Meister」を発表。価格は7万4800円で5月下旬発売を予定している。

ボーランド株式会社（http://www.borland.co.jp/）

SuSE Linux日本語版上陸

カーネル開発者を多数かかえ、高い技術力をほこるドイツ最大手のLinuxディストリビューターSuSE Inc.が、日本語版SuSE Linux 7.1を4月21日にリリースした。

SuSE Linuxは、その時点で最新バージョンのパッケージを採用する、エッジのきいたディストリビューションで、日本語対応版のSuSEでは、カーネル2.2.19（2.2.18）、KDE 2.1.1（2.0.1）、GNOME 1.4.0（1.2.4）を採用して、オリジナル版よりも新しいパッケージが収録されている（カッコ内のバージョンはオリジナル版のもの）。KDEなどは日本語対応済みで、インストール時に2.4系のカーネルも選択可能だ。

現在CD-ROM作成用のイメージファイルは、オリジナル版と日本語対応版で別のブランチに分かれているが、次期バージョンSuSE Linux 7.2では、これらのバージョンは統合される予定だ。

ダウンロードサイト（ftp://ftp.suse.com/pub/suse/i386/japanese-7.1/）
SuSE Inc.（http://www.suse.com/）

Products

コンパクトでありながら使いやすいキーボード

Happy Hacking Keyboard Lite 2

コンパクトでありながら、使いやすいキーボードとして定評のあるHappy Hacking Keyboardに新ラインナップが追加された。今回のモデルでは、従来の英語配列に加え、日本語配列モデルと、USBモデルやカラーバリエーションが追加され、より魅力的なものとなった。

製品名	Happy Hacking Keybard Lite 2
価格	オープン価格（PFUダイレクトでの通信販売価格：7800円）
問い合わせ先	株式会社PFUサイバービジネス営業課 TEL 0120-144-541 http://www.pfu.co.jp/hhkeyboard/

Happy Hacking Keyboard Lite 2（以下、Lite2）が3月21日にPFUから発売された（提供は4月2日より）。これによりHappy Hacking Keyboardシリーズは Happy Hacking Keyboard（以下、HHK）、Happy Hacking Keyboard Lite（以下、Lite）、Lite 2の3種類となる。

写真1　英語（UNIX）配列のLite 2
右下にカーソルキーが追加された。ほかのキートップに比べてかなり小さい。ファンクションキーはFnキーと1〜^キーとの組み合わせで打鍵する。

今回追加されたLite 2は全部で4ラインナップだ。

- **PD-KB200W/P　英語配列、PS/2インターフェイス、白**
- **PD-KB200B/P 英語配列、PS/2インターフェイス、黒**
- **PD-KB200W/U 英語配列、USBインターフェイス、白**
- **PD-KB210W/P 日本語配列、PS/2インターフェイス、白**

ラインナップを見るとわかるように、従来の英語（UNIX）配列に加え、日本語配列が用意された。今まで英語配列だったために購入をためらっていたユーザーには嬉しいラインナップだ。

また、英語配列のPS/2タイプでは黒色モデルも用意された。ただし、USBモデルや日本語配列モデルでは黒色モデルは用意されていない。このように、ラインナップは少し複雑なので購入する際には注意が必要だ。

なお、Lite 2ではノート用のキーボードのようにキートップ、ストロー

クともに小さいものだが独立カーソルキーが配置されるようになった。

Lite 2で新たに追加されたUSBタイプは、Linuxでの動作が気になるところだ。実際に編集部で入手したUSBタイプのLite 2を使って試したところ問題なく使えるようだ。ただし、USBタイプにはキーボードの後ろにUSB端子が2つ用意されており、ここにUSBマウスなどをカスケード接続することができるようになっているが、ディストリビューションによっては、インストール時にカスケード接続したUSBマウスを認識できないという問題があった。

キーボードへのこだわり

HHKシリーズは東京大学名誉教授の和田英一氏とPFU研究所の共同研究によって開発されたキーボードだ。HHKシリーズの設計思想は、徹底的なキーボードへのこだわりにある。どのマシンでも同じキーボードを利用したいという要望から生まれたもので、Sunのワークステーションのキーボードと同じ配列となっている。特に、CtrlキーがAキーの左隣り（通常のAT用キーボードのCapsキーの位置）にあるため、EmacsなどCtrlキーを多用するユーザーにとって使いやすい。

HHKシリーズのもうひとつの魅力はそのコンパクトさだ。必要最低限のキーだけに絞り込み、A4判の約半分のサイズでありながら、フルキーと同じサイズのキートップを持つ。ただし、Lite 2は従来のHHKシリーズに比べ、追加されたカーソルキーの分だけ10.5mmほど奥行きが長くなっている。このためサイズは294mm(W)×120.5mm(D)×38.6mm(H)となった。

なお、ファンクションキーなど、ふだんあまり使わないキーはFnキーを併用して打鍵することで、ホームポジションから指を離すことなく操作できるようになっている。

使い心地

ホームポジションから指を離すことなく使えるLite 2は非常に使いやすいキーボードだ。タッチタイピングユーザーであれば、容易に想像できるだろう。

しかし、実際に2週間ほど使ってみたところ、右Ctrlキーと、Deleteキーがないのが意外と不便であった（DelキーはFnキーとBackSpaceキーの組み合わせ）。

また、キータッチ感が少し気になった。キータッチは個人的な趣向の

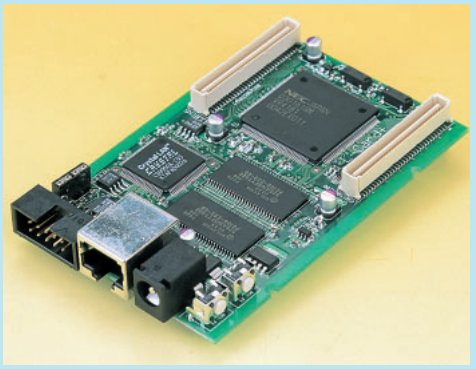
写真2　USBタイプのUSBコネクタ
USBマウスなどをカスケード接続することが可能なUSBハブが用意されている。USBタイプのみでPS/2タイプには用意されていない。

部分が大きいが、クリック感が得られないためストロークの限界までキーボードを押してしまう。これは、HHKがメンブレンスイッチ（フィルム状接点）方式を採用しているためだろう。この方式は、キーボードのスイッチ機構がゴムキャップ（ラバーキャップ、クリックラバーともいう）で構成されており、キーを打鍵することによってゴムが押されスイッチが入る構造のもので、比較的安価なキーボードやノートPCで採用されている方式だ。メカニカルスイッチ方式のキーボードに比べ、クリック感を得にくい。確かにメカニカルスイッチ方式はコストが割高になることは否めないが、HHKを購入するユーザーはキータッチ感も重視すると思われるので、多少高価になってもメカニカルスイッチ方式のHHKのラインナップが望まれるところだ。

写真3　日本語配列（PS/2タイプ）
全角／半角キーや変換キーなどが配列されている。DeleteキーはFnキーとの組み合わせで打鍵する。DeleteキーとBackSpaceキーを切り替えることはできない。Linuxではあまり問題にならないが、Windowsでは少し不便だ。

写真4　英語配列（USBタイプ）
英語版ではDeleteキーとBackSpaceキーを、◇キーはAltキーとWindowsキーをそれぞれディップスイッチで切り替えることができる。日本語配列の68キーにくらべ3つ少ない65キーとなっている。

MIPS CPUを搭載した名刺サイズの超小型組み込み用Linuxサーバ



L-Card+

組み込み機器のOSとして、ライセンス不要で開発ツールも揃っているLinuxが注目されている。低消費電力の組み込み用64ビットCPUを搭載し、10BASE-Tイーサネットを備えている組み込み用の超小型Linuxカードを紹介しよう。

製品名	L-Card+
価格	3万8600円
問い合わせ先	レーザーファイブ株式会社 TEL 03-5818-6626 http://www.laser5.co.jp/

レーザーファイブから、64ビットCPUを搭載した組み込み用コンピュータ「L-Card+」が発売されている。レーザーファイブは、LASER5 Linuxのディストリビューターだが、組み込み用Linuxソリューション事業も手がけており、PCIカードサイズの「L-Board」やルータ製品なども今後発売する予定になっている。

L-Card+は、名刺サイズ（60mm×91mm）というコンパクトな基板の両面に、VR4181-66（66MHz動作）、16MバイトSDRAM、2Mバイトフラッシュ ROM、シリアルポート、10BASE-T、コンパクトフラッシュインターフェイス、小型LED×8個、電源用レギュレータを搭載している（**写真1**）。

VR4181は、MIPSアークテクチャを採用した64ビットRISC CPUで、32個の64ビットの内部レジスタ、高速積和演算器、メモリ管理ユニットを搭載し、MIPS IIIとMIPS16の命令セットをサポートするVR4110コアをベースに、各種周辺機器用インターフェイスを内蔵している。FPU（浮動小数点ユニット）は搭載していない。

64ピンの小型拡張コネクタが2個装着されていて、液晶表示信号、タッチパネル信号、ROM拡張、I/O、音声入出力などが出力されている（**写真2**）。

フラッシュROMには簡易モニタとブートプログラムが書き込まれていて、コンパクトフラッシュを利用したファイルシステム上のLinuxを起動する。

製品には、本体のほかに、ACアダプタ（DC +5V／2.3A）、シリアル変換ケーブル（DIP 10ピンコネクタ←→D-Sub 9ピンコネクタ）、開発ツールやドキュメントが入ったCD-ROMが添付されている。

また、デモキット「L-Card+DEMO」が用意されていて、上記に加えて、Linuxインストール済みの64Mバイトコンパクトフラッシュカード、RS-232Cクロスケーブル、カテゴリ5クロスLANケーブルが添付される。

リアルタイムクロックはCPUに内蔵しているが、バッテリーバックアップが付いていないため、電源が供給されないとリセットされてしまう。

製品の保証期間は1年間。サポートは、カーネルとコマンドに関してFAXまたはメールで3カ月間受けられる。それ以外の内容や3カ月以降は有償となっている。

組み込みLinuxの実際

今回はデモキットを試用した。まず、L-Card+にコンパクトフラッシュを装着し、シリアル変換ケーブルとRS-232Cクロスケーブルを使ってWindowsマシンに接続した。基板がむき出しなので、ショートさせないように金属製のものをそばに置かないように注意が必要だ。

写真1　コンパクトフラッシュを装着したL-Card+
組み込み用途のためむき出しのボードで提供される。500円硬貨と比べても、L-Card+の小ささがよくわかる。

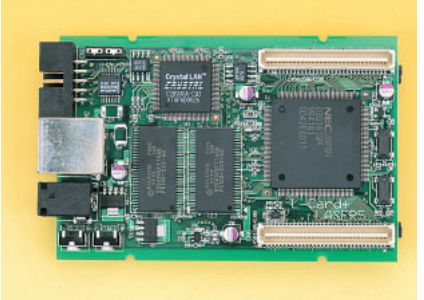


写真2　ボードのCPU実装面
VR4181 CPUとSDRAM×2、LANコントローラなどを搭載。CPUの上下にあるのが小型の64ピン拡張コネクタ。

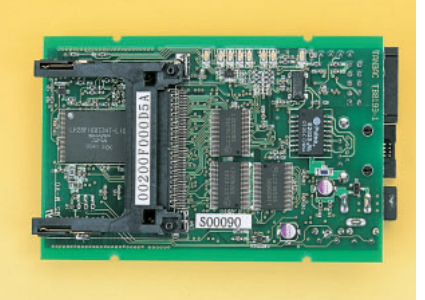


写真3　ボードのコンパクトフラッシュ実装面
コンパクトフラッシュ用コネクタの下にフラッシュROMを搭載。上のほうに表面実装タイプのLEDが8個並んでいる。

通信ソフトにTera Term Proを利用し、シリアルポートを115200bpsにセットして、L-Card＋にACアダプタを繋ぐと、「waiting...」と「＊」が表示される。数秒後にLinuxが起動し始める（**画面1**）。

見慣れたLinuxのメッセージが続いて、「(none) login: 」というログインメッセージが表示される。rootでログインすると、パスワードは設定されていないため、すぐにプロンプト「#」が表示される。以後は、lsやcatなどのLinuxのコマンドが普通に使用できる。

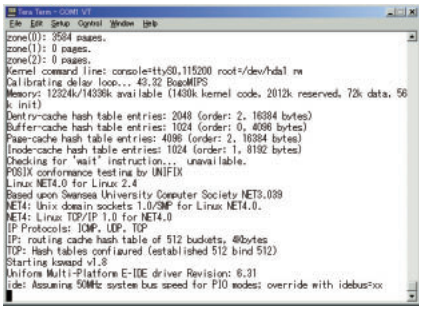
次に、イーサネットで接続してみる。ネットワークアドレスは、192.168.128.104に初期設定されているが、/sbin/etherupファイルを編集することで変更できる。シリアル経由で接続したターミナルから適切なIPアドレスに変更して、L-Card＋を再起動する。LANケーブルを接続し、Tera Term Proからtelnetでログインする。

画面2を見てわかるように、カーネル2.4.0-test9ベースのLinuxで動作している。わずか64Mバイトのコンパクトフラッシュに入れるために、必要なコマンドだけがインストールされている。また、シェルはbashではなくtcshが使われていた。

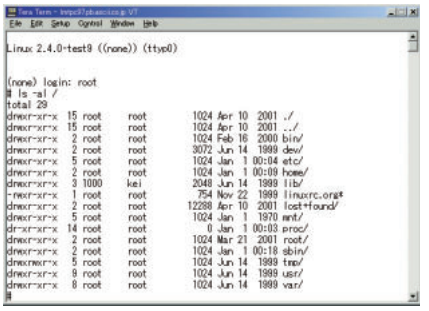
クロス開発環境の構築

L-Card＋は、MIPS CPUが使われているために、x86用のバイナリプログラムをそのまま実行することはできない。そのためソースからMIPS用のバイナリを作成する必要がある。

Linuxプログラムの開発にはGCC開発環境を使用するが、ハードディスクを装着していないL-Card＋上にGCCをインストールすることはコンパクトフラッシュの容量の点から難しい。そこでクロス開発環境をPC上に構築する。



画面1　Tera Termでシリアルポートに繋いでコンソールを表示
Linuxの起動メッセージを表示中。約1.5Mバイトと非常にコンパクトなカーネルを使用している。



画面2　イーサネットからtelnetでログインした
購入した当初はrootにパスワードが設定されていない。コンパクトフラッシュ上のファイルシステムを利用する。

添付されるCD-ROMには、L-Card＋に搭載されているカーネルのソースと、コンパイル済みのクロス開発環境ファイルが収録されているので、これを利用するのが一番楽だろう。手順はドキュメント（CD-ROM内のHTMLファイル）に書かれている。

CD-ROM内のCross-VR.tar.bz2を、/usr/localディレクトリに展開し、PATHに/usr/local/Cross-VR/binを追加し、includeファイルのシンボリックリンクを張り直せば、クロス開発環境はできあがる。

gccコマンドの代わりに、mipsel-linux-gccコマンドを使って、C言語のソースをコンパイルできる。

ユーザーが作成したプログラムをL-Card＋に転送するには、イーサネットで接続されているならば、L-Card＋からLinuxマシンにftpコマンドで接続して取ってくればよい。

ところで、L-Card＋のディレクトリを探していると、X Window Systemのプログラムがいくつかインストールされていた。L-Card＋のI/Oポートに液晶ディスプレイを装着して使うことを想定しているのかもしれないが、LANでLinuxマシンに接続しているのならば、「-display」オプションを指定することで、LinuxマシンのXサーバ上に表示することが可能だ。

L-Card＋上で動作させるアプリケーションは、Xを使ったGUIインターフェイスを利用するとよいかもしれない。実際にLinux搭載PDA「AgendaVR3」は、L-Card＋と同じCPUが使われている。

L-Card＋はバッテリーでの動作を考慮した電源レギュレータが装着されている。持ち運べるシステムへの組み込み用途としても応用が可能だろう。

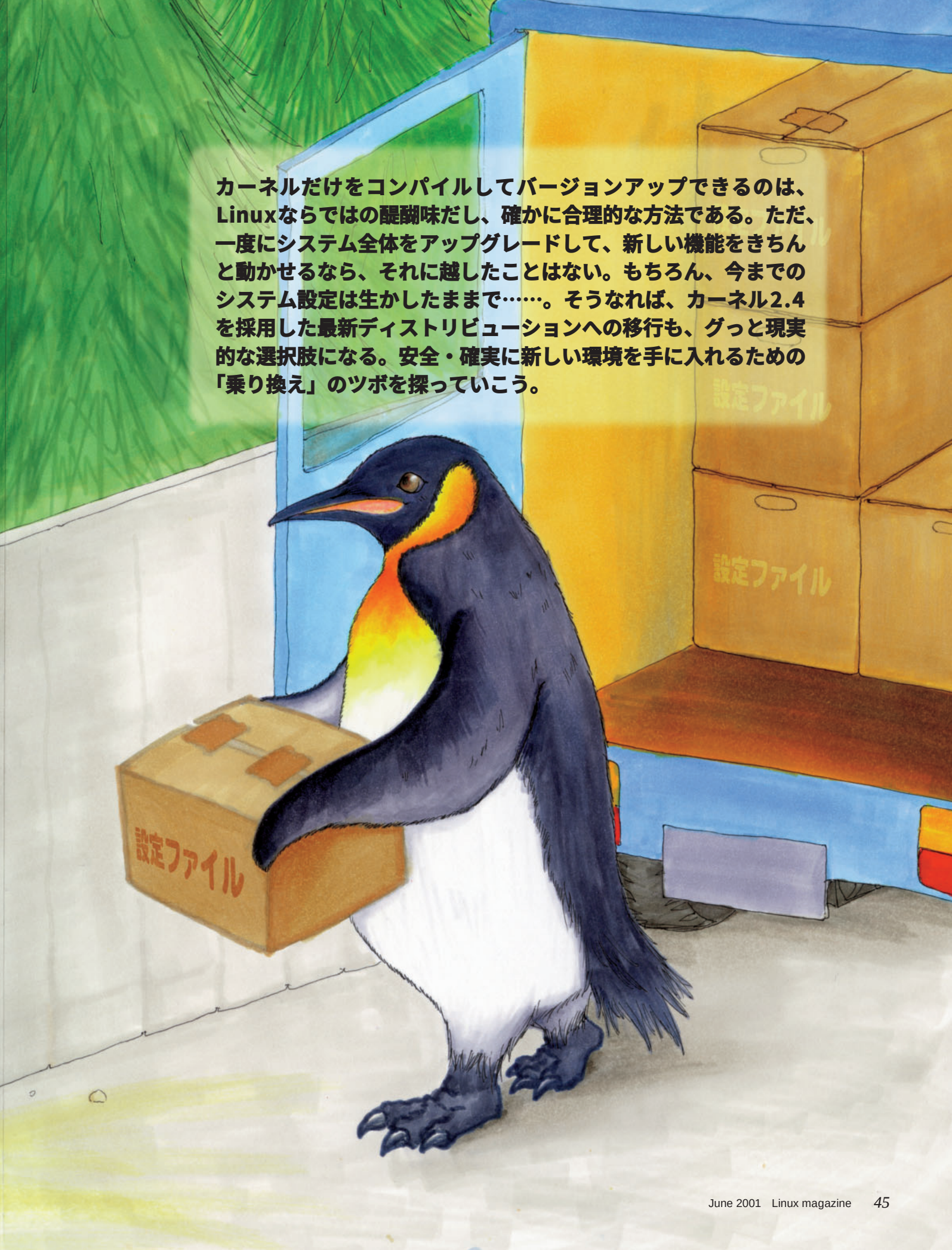
CPU	NEC VR4181 66MHz
RAM	16MバイトSDRAM
フラッシュROM	2Mバイト
ネットワーク	10BASE-T×1ポート（CS8900A）
シリアルポート	RS-232C×1ポート（10ピンコネクタ）
拡張スロット	コンパクトフラッシュType I、II準拠
拡張コネクタ	VR4181系ISA信号、I/O、サウンドなど（64ピンコネクタ×2）
LED（ボード上）	LAN用×2、ユーザー用×5、フラッシュROMアクセス用×1
ボードサイズ（mm）	91（W）×60（D）
電源	DC+5V（DC 3.65～8.5Vの電池使用も可能）
付属品	ACアダプタ（DC+5V／MAX 2.3A）、シリアル変換ケーブル、CD-ROM（Linuxクロス開発用環境ファイル、簡易取扱説明書、資料）

表1　L-Card+の主な仕様

最新ファイルシステムや新機能を活用しよう！

ディストリビューション乗り換え術

現在の環境を壊さずに▶▶▶▶▶▶▶▶▶▶▶▶新バージョンへアップグレード



カーネルだけをコンパイルしてバージョンアップできるのは、Linuxならではの醍醐味だし、確かに合理的な方法である。ただ、一度にシステム全体をアップグレードして、新しい機能をきちんと動かせるなら、それに越したことはない。もちろん、今までのシステム設定は生かしたままで……。そうなれば、カーネル2.4を採用した最新ディストリビューションへの移行も、グっと現実的な選択肢になる。安全・確実に新しい環境を手に入れるための「乗り換え」のツボを探っていこう。

姿を見せ始めたカーネル2.4採用ディストリビューション

「Red Hat Linux 7.1」のリリースによって、各ディストリビューションの2.4系への移行が急速に進んでいくと予想される。リリースラッシュに備え、まずカーネル2.4の動向を再確認し、続いて編集部いち押しの新機能「RieserSF」と「LVM」を徹底解剖していく。

Linuxユーザーがディストリビューションを選択する基準は何だろうか？「セキュリティがしかっりしている」、「日本語環境が充実している」、「開発プロジェクトを応援している」、「収録パッケージ数が多い」、「イメージキャラクターが大好き」、「雑誌を買ったら付いてきた」とまあ、ちょっと思いつくだけでも、さまざまな理由が考えられる。

これから、確実に加わりそうな新しい基準がひとつある。

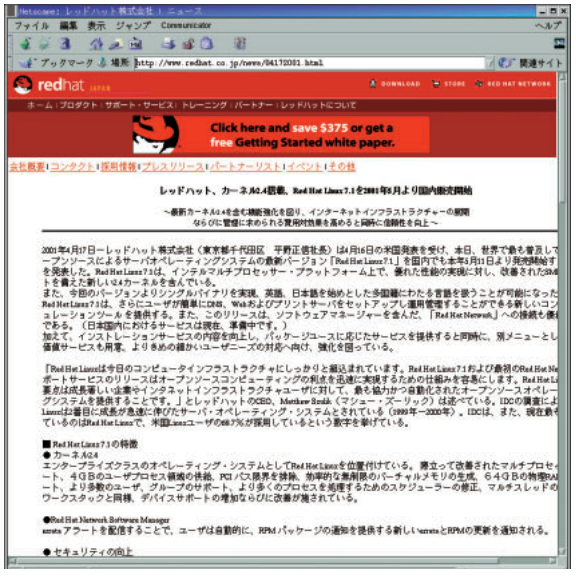
カーネル 2.4を採用している

がそれだ。

ついに登場！　Red Hat 7.1

昨年のうちから、すでにテスト版カーネルをベースにした、いくつかのディストリビューションがリリースされていた（Linux MLD5など）。今年に入って、カーネル2.4.0のリリースを受け、まず先陣を切って「SuSE Linux 7.1」が発表された。SuSE 7.1は日本語をサポートしていないが（バージョン7.2から正式サポート予定）、とにかくカーネル2.4を体験するためのベースが得られるようになったわけだ。

この間、ほかのディストリビューションの開発プロジェクトでも、着々と準備が進められていた。RPM系ディストリビューションの本家であるRed Hatは、カーネル2.4を採用した次期バージョンの最終リリースに向けて「Fisher」から「Wolverine」へとベータ版の公開を順調に消化していった。そのほかのディストリビューターもロードマップの発表やサポート表明を行っていた。



画面1　Red Hat 7.1をアナウンスするレッドハットのWebページ
米国Red Hatの発表に続いて、国内でもアナウンスが行われた。日本での製品パッケージの発売は5月11日の予定。「Delux」が1万4800円、「Professional」が3万4800円となっている。

このような状況の中、Linux magazine編集部では、カーネル2.4を採用したディストリビューションへの移行に備えるための企画が進行していた（この特集です）。そして企画内容が固まり、動作確認などのテストを開始した矢先の4月16日、Red Hat Linux 7.1のリリースが正式にアナウンスされたのであった。

そこで急遽、これを入手し、インストール、動作確認するとともに、カーネル2.4の機能を検証するためのベース環境とすることにした。なお、Red Hat 7.1で採用されたカーネルのバージョンは2.4.2である。

2.4系カーネルを採用した初の日本語対応ディストリビューションということで読者のみなさんの期待も大きいと思われるが、残念ながら制作工程の関係上、今月号の付録CD-ROMに収録することはできなかった。来月号での収録をお約束することで、どうかご容赦いただきたい。

注目の新機能

先ほども少し触れたように、この特集は2.4系カーネルを採用したディストリビューションのリリースに向けての「準備」企画としてスタートした。スムーズに2.2系から2.4系に移行するには、どのような作業が必要となるのか、カーネル2.4の新機能のうち、どれを使えば、より便利に、より有効にLinuxを活用できるのか、といったことを検討

していきながら、そのために前もって知っておきたい情報や関連する基礎知識を紹介するのが狙いである。

そこで注目したのが、ファイルシステムを始めとするディスクストレージまわりの新機能だ。改めて言うまでもないが、ディスクストレージは、OS、アプリケーション、それらの設定ファイルやデータファイルなど、システムのソフトウェア部分のありとあらゆるものを永続的に保管する「貯蔵庫」の役割を果たしている。高い信頼性と安定性が必須であると同時に、水準以上の処理能力が求められる。

カーネル2.4では、この要件を満たすべく新しいファイルシステムとして「ReiserFS」が採用された。ReiserFSはジャーナリング機能による高い障害復旧性を備えたファイルシステムだ。

また、「バランス木」と呼ばれる構造を持ち、従来のext2などと異なり、ブロックサイズが可変となっている。このバランス木構造は、高速なファイル検索やディスクスペースの効率的な利用を実現するとされている。ReiserFSについては、性能評価、ext2との比較なども含めて、このあと**48ページ**から詳細に解説する。あわせて、大きく改良された「VFS（Virtual File System）」についても取り上げる。

ReiserFSと並ぶ、ディスクストレージ関連の目玉のひとつが「LVM（Logical Volume Manager）」であろう。その名の通り「論理なボリューム」を「管理」するための機構で、カーネルに統合されたモジュールと専用ツールによって、柔軟なディスク管理を行うことができる。複数のパーティションを1つの論理的なディスクとして運用・管理するという機能は、決して目新しいものではないが、従来Linuxに欠けていた部分を補うものである。LVMの詳細は、**54ページ**以降で取り上げている。

ファイルシステムとディスク管理について知っておくことは、システムの移行やアップグレードをするうえで非常に重要なことだ。この機会に知識を深めて、ぜひとも実際のケースに活かしてほしい。

カーネルのアップグレード方法

本誌3月号のカーネル特集では、ソースファイルからコンパイルしてアップグレードする方法を紹介した。今回は、ディストリビューションごとバージョンアップするという方法について検証していく。詳しくは、**60ページ**からの実際例を見てもらうとして、この方法の長所と短所をまとめておこう。

「ディストリビュータが整備した標準的なシステム環境を利用できる」というのは、大きな利点のひとつだ。カーネルをアップグレードできるだけでなく、関連するツールや必要なライブラリなどを、まとめて一度にインストールできる。

汎用的な内容にセットアップされた設定ファイルも用意されているので、すぐにある程度整ったベーシックな環境が手に入ることになる。　また必要に応じて、各種パッチの適用、ディストリビューター独自の修正が施されている。さらには、使用するハードウェアの違いがあるにせよ、ディストリビューターによってテスト済みなのだ。これらをすべて自分で一から行うのは、大変な労力である。

マイナス面として、異なるディストリビューションに移行する場合にはアップグレードインストールできないため、設定ファイルやデータの移行作業が必要になるということがあげられる。このあたりが「乗り換え術」のポイントとなりそうだ。

5箇条の心得（再掲）

ここで、3月号の特集に掲載した、当編集部独自の「カーネルバージョンアップ5箇条の心得」を再掲しておこう。今回も、この心得は有効なはずである。

①バージョンアップは自己責任のもとで行うこと
②バージョンアップの必要性を徹底検討すること
③スキルに合わせてインストール方法を選択すること
④付属ドキュメントには必ず目を通すこと
⑤インストールの前準備は慎重に行うこと

①については、脅したり責任逃れのつもりでは決してない。「気合を入れていこう」くらいの意味にとってほしい。システムのアップグレードは、どんな場合にも、危険を伴う作業なのだ。

②について。そこに必要性がなければ、せっかくの新機能も有効に活用できないということ。本当にシステムにその機能が必要なのか再考してみよう。「適材適所」がキーワードだ。

正しくアップグレードを行うには、ある程度の知識と慣れのようなものが必要となる（③）。ただ、足りない知識は補えばいい。この特集をよく読んで、自分なりの方法論を身に付けよう。

④は常識。⑤は読んで字のごとくである（バックアップを忘れずに！）

では次ページから、Linuxの新しいファイルシステムについて、じっくりと見ていくことにしよう。

Linuxで使えるジャーナリングファイルシステム

ファイルシステムとは、ファイルをフロッピーやハードディスクの内部フォーマットのことで、WindowsではFAT／FAT32／VFATやNTFSなどが使われる。

Linuxでは標準的なファイルシステムとして、「ext2」と呼ばれるものが使われてきた。たとえばRed Hat Linuxのインストール時に、ハードディスクのパーティションをフォーマットする画面で一般に「Linux native」を選ぶが、これによってext2ファイルシステムが作成される。

数十Gバイトのハードディスクが安価に手に入るようになっているが、大きなパーティションをext2でフォーマットすると、結構な時間がかかる。また、Linuxシステム自体がなんらかの理由で落ちて、正しくシャットダウンされなかった場合、次の起動時には、fsckプログラムによってファイルシステムのチェックが行われる。このfsckプログラムのチェック時間がパーティションのサイズに比例して増加するため、1つのパーティションを大きくすると大変である。

例として、編集部のK6-2 350MHzのマシンに40GバイトのIDEハードディスク（Maxtor DiamondMax VL 40、5400rpm）を付けて試したところ、30Gバイトのパーティションをext2でフォーマット（mke2fsコマンドを使用）するのに2分30秒ほどかかった。続いて、fsckによるファイルシステムのチェックも行った。約100Mバイトのファイルを書き込んだ状態でfsckを起動したところ、終了までに約60秒、ファイルをたくさん作って20数Gバイト分書き込んだ状態では、約2分かかった。

個人で使うのならば待っていてもよい時間かもしれないが、Webサーバなどで大勢の人をサービスしているシステムでは、メンテナンスの際のダウンタイムは1秒でも短くしたいはずだ。しかし、fsckによる時間のムダは、ext2を使う限り避けられない。

また、カーネル2.2までのext2ファイルシステムでは、1つのファイルのサイズが最大2Gバイトまでとなっていて、大きなファイルを扱うビデオやデータベース用途にはext2では不満に思えるようになってきている。

■VFSを拡張するLFS

Linuxでは、WindowsやMacを始めとする複数の異なるファイルシステムをアクセス可能である。そのため、アプリケーションまたはOS側から共通のインターフェイスで扱えるように、ファイルシステムを仮想化するVFS（Virtual File System）を備えていて、各ファイルシステム用ドライバとOSの間のインターフェイスを行っている。

カーネル2.2までのext2ファイルシステムが、2Gバイトまでのファイルしか作れなかったのは、実際にはVFSがファイルサイズを32ビットint型で表すことにしていたための制限である。32ビットint型は、−2147483648〜2147483647が表現できるので、すなわち最大2Gバイト−1となる。Alphaなどの64ビット版Linuxではint型が64ビットなので、以前から2Gバイトを超えるファイルを作成できた。

カーネル2.4ベースのシステムとglibc 2.2ライブラリでは、LFS（Large File Summit）拡張によって、64ビット化されたシステムコールが追加されたために、この制限はなくなって2Gバイト以上のファイルを扱えるようになった。もちろん、lsやcpコマンドなどのLinuxに標準装備されているユーティリティプログラムも、LFSに対応している必要がある。

ジャーナリングファイルシステム

ディスククラッシュが起きた場合でも、データを復旧できるようにしたファイルシステムとして、ジャーナリングファイルシステムがある。これは同時にext2とfsckの問題も解決するものだ。まずext2とそれ以外のファイルシステムとの違いを見てみよう。

■ext2

ext2は、初期のLinuxで使われていたMINIXファイルシステムや、BSDのFFS（Fast File System）を参考にして作られたLinux専用のファイルシステムである。Linuxでのハードリンク／シンボリックリンクを効率よく実現するために、ファイルをi-nodeという呼ぶファイル情報領域とデータ領域に分けて管理している。全i-nodeと全ブロックの使用状況をビットマップによって管理していて、ファイルを書き込む際には、使用した部分に該当するi-nodeとブロックのビットマップ情報を使用済み（＝1）に書き換えることも行う。

■ext3

ext3では、ext2と互換性を持たせるために、ext2のデータ領域に隠しファイルのような形式でジャーナリングログを置いている。そのおかげで、ext2のファイルシステムからext3へ変換することも、tune2fsコマンドで簡単に行える。逆にext2しかサポートしないOSからでも、ext3ファイルシステムを、マウントし読み書きすることも可能だ。ただし、その場合はジャーナリング機能は利用されない。

ext3は、Linuxカーネルに対するパッチとして提供されている。ext3を標準でサポートするディストリビューションは、Turbolinux Server 6.5だけである。

■ReiserFS

ReiserFSは、ディスクをデータベースで使われるバランスドツリーというデータ構造で管理し、ファイルやディレクトリを記録している。ファイルのサイズやパーミッション情報などのi-node情報と、ファイルのデータ自体を連続して配置して記録するという特徴を持っている。

ファイルの読み書きを行うために、ext2（ext3）では、ビットマップ領域やi-nodeテーブルといった、ディスク上で離れたエリアにアクセスする必要があるのに対して、ReiserFSでは、近いエリアをアクセスするだけで情報が得られるという利点がある。

また、ReiserFSはデータをファイルサイズに合わせて詰めて記録するので、小さなファイルが連続していて1ブロック内に複数のファイルがある場合などでは、ディスクアクセスの回数を減らす効果がある。

ReiserFSは、ドイツのSuSEがスポンサーとなっているようことでもわかるように、SuSE Linuxがいち早くからサポートしていた。

国内でReiserFSをサポートするディストリビューションとしては、2000年の5月に発売されたサーバ用ディストリビューションKondara MNU/Linux Server 1.1が初めてで、デスクトップ用としては同8月発売のKondara MNU/Linux 2000が最初である。グラフィカルインストーラからReiserFSを選べるようになっている（**画面1**）。

Vine Linux 2.0以降では、カーネルにReiserFS用のパッチが当てられていて、未サポートながら使用することが可能になっている。このほかに、Linux MandrakeもReiserFSをサポートしている。

■ジャーナリングの実際

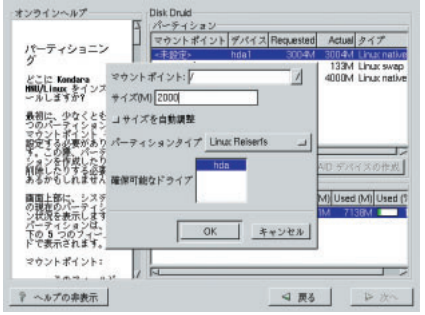
これらを例に、ジャーナリングファイルシステムの仕組みを見てみよう。

ext2、ext3、ReiserFSの、それぞれのディスク上での内部データ構造を簡略化して示したのが**図1**である。ext3とReiserFSには、ジャーナリングログ用の領域が確保されている。

ジャーナリングファイルシステムとは、ファイルシステムへのデータ書き込みの手順をログに記録しておくことで、システムのクラッシュなどでファイルシステムの不整合が起こっても、そのログを元にして再構成することができるものだ。

一般的なジャーナリングログの構造を**図2**に示す。ログ領域は固定サイズで、リング上のバッファになっている。ログを書き込んでいって最後まで行くと、先頭に戻って記録する。

ファイルを書き込む際には、まず、ファイル自身の情報やi-node、ブロックビットマップの変更などのディスク



画面1　Kondara MNU/Linux 2000のインストーラ

いち早くReiserFSに対応したKondara MNU/Linuxでは、インストーラ上でReiserFSを選ぶことが可能だ。

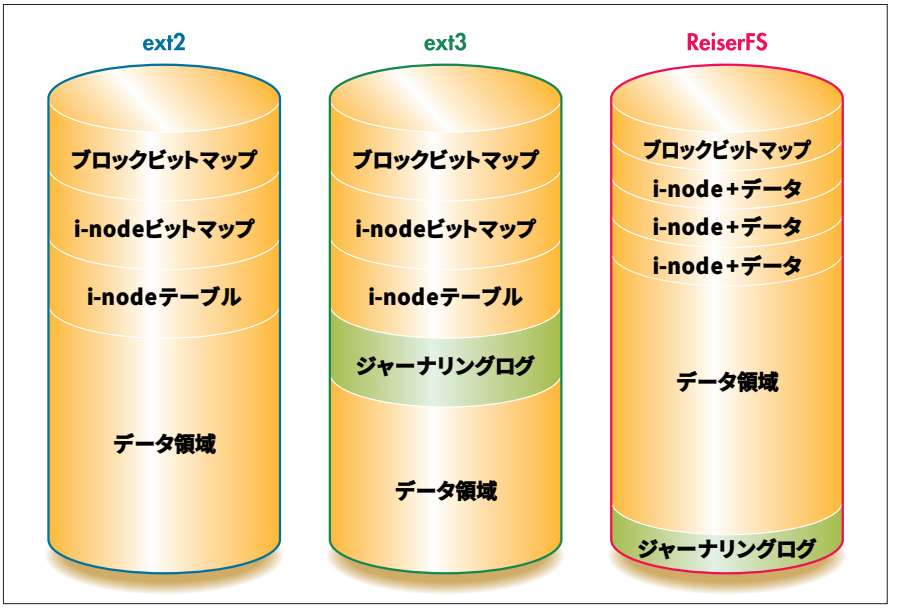


図1　ext2、ext3、ReiserFSのディスク上での内部データ構造

上の更新情報をログレコードに記録する。それから管理領域とデータ領域の書き換えを行い、それが正常に終了したならば、ログを削除する。ログの削除はログレコードを指すポインタを変更するだけである。

実際には、ディスクアクセスを効率よく行うために、ログレコードへの記録と、管理領域とデータ領域への書き込み、ログの削除は、非同期に行われる。

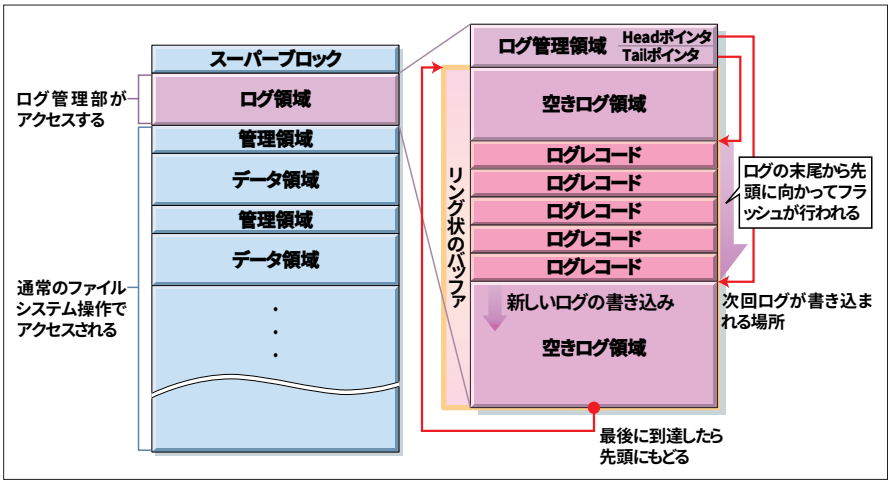


図2　ジャーナリングファイルシステムの内部構造

```
# mkreiserfs /dev/hda6

<-------------mkreiserfs, 2000------------>
reiserfsprogs 3.x.0f
Creating reiserfs of 3.6 format
Block size 4096 bytes
Block count 1024135
Used blocks 8243
Free blocks count 1015892
First 16 blocks skipped
Super block is in 16
Bitmap blocks (32) are :
        17, 32768, 65536, 98304, 131072, 163840, 196608, 229376, 262144,
294912, 327680, 360448, 393216, 425984, 458752, 491520, 524288, 557056,
589824, 622592, 655360, 688128, 720896, 753664, 786432, 819200, 851968,
884736, 917504, 950272, 983040, 1015808
Journal size 8192 (blocks 18-8210 of file /dev/hda6)
Root block 8211
Hash function "r5"
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
            ALL DATA WILL BE LOST ON '/dev/hda6'! (y/n)y
Initializing journal - 0%
...20%...40%...60%...80%...100%                                 left 0, 409
/sec
Syncing..

ReiserFS core development sponsored by SuSE Labs (suse.com)

Journaling sponsored by MP3.com.

To learn about the programmers and ReiserFS, please go to
http://www.devlinux.com/namesys

Have fun.
#
```

画面2　ReiserFSファイルシステムを作成

システムプログラムやアプリケーションなどからのリクエストが続いてきて、ログレコードがいっぱいになってしまう場合には、ログが削除されてログレコードの空き領域ができるまで待たされることになる。

システムが異常停止した場合などは、ログレコードを参照するだけでファイルシステム自体の管理情報を正常に戻すことができるので、fsckが不要になる。

ジャーナリングファイルシステムごとの実装方法の違いによって、ログに記録する内容や更新タイミングなどは異なるが、概要はいずれも同じである。

ReiserFSファイルシステムの作成

では、標準でReiserFSが含まれているディストリビューションで、ハードディスクを増設したりする場合など、手動でReiserFSファイルシステムを作成する方法を簡単に紹介しよう。

まず、fdiskでハードディスクのパーティションを確保する。

次に、mkreiserfsコマンドで、ファイルシステムを作成する（**画面2**）。

そして、ファイルシステムをマウントする（**画面3**）。

起動時に自動的にマウントするように、/etc/fstabにデバイス名やマウントポイントを追加しておく（**画面4**）。

■ReiserFSのバージョンによる違い

ReiserFSは、カーネル2.2用のReiserFS V3.5と、カーネル2.4用のReiserFS V3.6の2種類ある。V3.5は、ファイルサイズを32ビットint型で扱っているため、最大のファイルサイズは2Gバイトである。V3.6では、2Tバイトまで扱えるので、事実上制限はなくなったといえる。

カーネル2.4（ReiserFS V3.6）のシ

ステムでは、mkreiserfsコマンドを使ってファイルシステムを作成する際に、V3.6用にフォーマットされる（「mkreiserfs -V 1 デバイス名」とすれば、V3.5用にすることも可能）。

パーティションの内部フォーマットが変更されたため、カーネル2.4（ReiserFS V3.6）で作成したReiserFSのファイルシステムは、カーネル2.2（ReiserFS V3.5）からマウントすることはできない。逆に、V3.5で作られたファイルシステムは、V3.6のシステムから読み書きすることができる。必要ならば、mountコマンドのconvオプションで、V3.5→V3.6にフォーマットを変換することも可能だ。

```
# mkdir /mnt/reiser
# mount -t reiserfs /dev/hda6 /mnt/reiser
# ls -la /mnt/reiser
合計 8
drwxr-xr-x    2 root     root           48  4月 20 21:11 .
drwxr-xr-x    7 root     root         4096  4月 20 21:17 ..
# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda1              3028080   2335656    538604  82% /
/dev/hda6              4096408     32840   4063568   1% /mnt/reiser
```

画面3　ファイルシステムのマウント

```
LABEL=/                 /                       ext2    defaults        1 1
/dev/fd0                /mnt/floppy             auto    noauto,owner    0 0
none                    /proc                   proc    defaults        0 0
none                    /dev/pts                devpts  gid=5,mode=620  0 0
/dev/hda5               swap                    swap    defaults        0 0
/dev/cdrom              /mnt/cdrom              iso9660
noauto,owner,kudzu,ro 0 0
/dev/hda6               /mnt/reiser             reiserfs defaults       0 0
```

画面4　/etc/fstabの変更
fstabファイルに、最下行のようにreiserfsでマウントするように設定する。

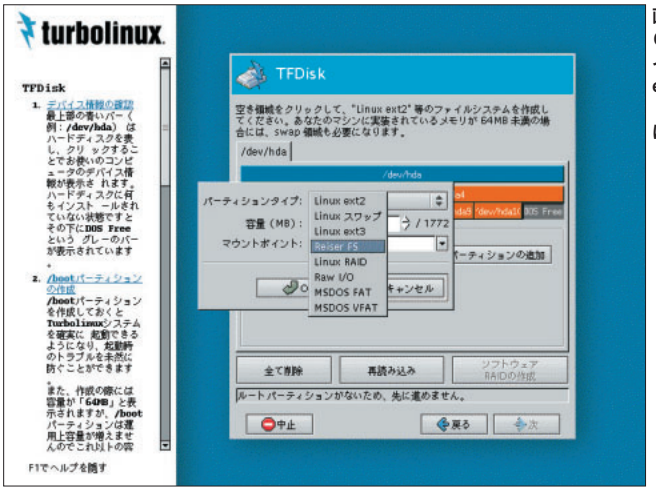
ext2とジャーナリングシステムの比較

実際にext2とext3、ReiserFSファイルシステムを使用して、ファイルの使用効率や実行時間がどれくらい違うのかを見てみることにしよう。

3つのファイルシステムを標準でサポートしているのは、現在のところTurbolinux Server 6.5だけである。カーネル2.2.18だが、LFS拡張も行われているので、ext2（ext3）では2Gバイトを超えるファイルも作成可能だ。ReiserFSは、V3.5のため1ファイル最大2Gバイトまでなのが残念なところ。

使用したマシンは、先ほども紹介したK6-2 350MHz、メモリ256Mバイトのシステムである。ハードディスクは、IBMのDeskstar DPTA-371360（13Gバイト）である。

Turbolinux Server 6.5は、インストーラからext3やReiserFSを選ぶことができる（**画面5**）。ただし、ルートパーティションにReiserFSを指定することはできないようになっている。今回は2Gバイトのパーティションにインストールして実験を行った。CPUの速度やハードディスクによって数値は変わってくるので、ファイルシステムごとの相対値を参考にしてほしい。



画面5　Turbolinux Server 6.5のインストーラ
インストーラ上でReiserFSやext3も選ぶことができる。ルートパーティションにはext2またはext3を指定する。

■ジャーナリング領域のサイズ

まず、200Mバイトのパーティションを3つ作り、それぞれ以下のコマンドで、ファイルシステムを作成した。

```
# mke2fs /dev/hda8
# mke3fs /dev/hda9
# mkreiserfs /dev/hda10
```

作成したファイルシステムをマウントして、dfコマンドでディスクの使用状況を表示させると、**画面6**のようになる。この表示は単位が1Kバイトである。ファイルシステムを作成したばかりで、何もファイルが書かれていない状態だが、まず注目してほしいのは「Used」の列だ。

一番上の行（/dev/hda8）に記録されているext2は、ルートディレクトリやlost＋foundディレクトリ用に13Kバイト使用されている。次の行のext3は、ext2での使用域とジャーナリングログ領域を合わせて約4Mバイト使用され

ている。最下行のReiserFSでは、約33Mバイトがジャーナリング用に使用されている。この結果、Availableの列やUse%の数字でわかるように、ext2が空き容量が一番多い。

今度は、1パーティションのサイズを2Gバイトにして同様にファイルシステムを作成して、dfを行ってみた（**画面7**）。ext2が少し増えて、ext3のところが4Mバイトから32Mバイトに大幅に増加しているが、ReiserFSは変わっていない。

このext2とext3での増加分は、ブロックサイズが変わったことによるもので、1パーティションが200Mバイトのときは1Kバイト／ブロック、2Gバイトのときは4Kバイト／ブロックになっている。mke2fs（mke3fs）が容量によって適切なブロックサイズを決めているのだが、−bオプションによってブロックサイズを変更することもできる。

ちなみに、ReiserFSのブロックサイズは、現在ではパーティションサイズに関係なく4Kバイトに固定のようだ。

ReiserFSでは、ext2やext3のような管理情報を記録するi-nodeエリアをあらかじめ確保しないので、パーティションの全容量がext2やext3よりも多くなる。そのため、2Gバイトのパーティションでは、ReiserFSの空き領域が一番多くなっている。

■ファイルを書き込んでみる

次に、先ほど作成した2Gバイトのパーティションにファイルを書き込んでみる。Linuxカーネル2.2.19のソース（約19Mバイト）を、tarコマンドでそれぞれのパーティションに展開してみた（**画面8**）。

dfコマンドとduコマンドで表示されるディスク使用量が、ReiserFSのほうが小さくなっている。カーネルソースには約6000個のファイルが含まれている。ext2ではブロック単位（この場合は4Kバイト）で管理されるために、ブロックサイズ未満のファイルの端数の部分が無駄なデータ領域を占めるのに対し、ReiserFSではi-node＋データサイズ分の領域だけで済むためだ。

tarコマンドの実行は、ext2で約31秒、ext3で約55秒、ReiserFSで約38秒であった。

■ファイルを検索する

ReiserFSでは、バランスドツリーという手法でディレクトリやファイルを管理しているため、複数のディレクトリからファイルを高速に検索できる。それをfindコマンドで確認してみる。

```
# find /mnt/ext2/linux/ -name "Make
file"
```

カーネルソースには、各ディレクトリにMakefileがある。ext2とext3では6秒強かかったが、ReiserFSでは約3秒だった。ただし、同じコマンドを2回以上繰り返してみると、バッファに読み込まれている情報を使用するためか、どのファイルシステムでも非常に速く終了する。

■Bonnie＋＋ベンチマーク

ファイルシステムの性能をチェックするために定番のベンチマークテストであるBonnie＋＋ Version 1.01を使用した。Bonnie＋＋では、大きなファイルへのアクセス速度を測定するテストと、複数のディレクトリに複数のファイル名を作成、ファイル情報読み込

```
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda8               202220        13    191767   1% /mnt/ext2
/dev/hda9               202220      4127    187653   3% /mnt/ext3
/dev/hda10              208788     32840    175948  16% /mnt/reiser
```

画面6　1パーティションを200Mバイトにした場合
ReiserFSのジャーナリングログ領域の大きさが32Mバイト固定なので、少ないディスク容量には向いていない。

```
# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda8              2016016        20   1913584   1% /mnt/ext2
/dev/hda9              2016016     32828   1880776   2% /mnt/ext3
/dev/hda10             2048188     32840   2015348   2% /mnt/reiser
```

画面7　1パーティションを2Gバイトにした場合
最下行のReiserFSが、パーティションの全容量（1k-blocks）も空き容量（Available）も一番大きい。

```
# cd /mnt/ext2
# tar xfz /root/linux-2.2.19.tar.gz
# cd /mnt/ext3
# tar xfz /root/linux-2.2.19.tar.gz
# cd /mnt/reiser
# tar xfz /root/linux-2.2.19.tar.gz
# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda8              2016016     93292   1820312   5% /mnt/ext2
/dev/hda9              2016016    126100   1787504   7% /mnt/ext3
/dev/hda10             2048188    118372   1929816   6% /mnt/reiser
# du -s /mnt/*/linux*
93272   /mnt/ext2/linux
93272   /mnt/ext3/linux
91124   /mnt/reiser/linux
```

画面8　tarコマンドでLinuxカーネルを展開
dfで表示される使用量（Used）の増加分とduで表示される使用量の、どちらもReiserFSが少ない。

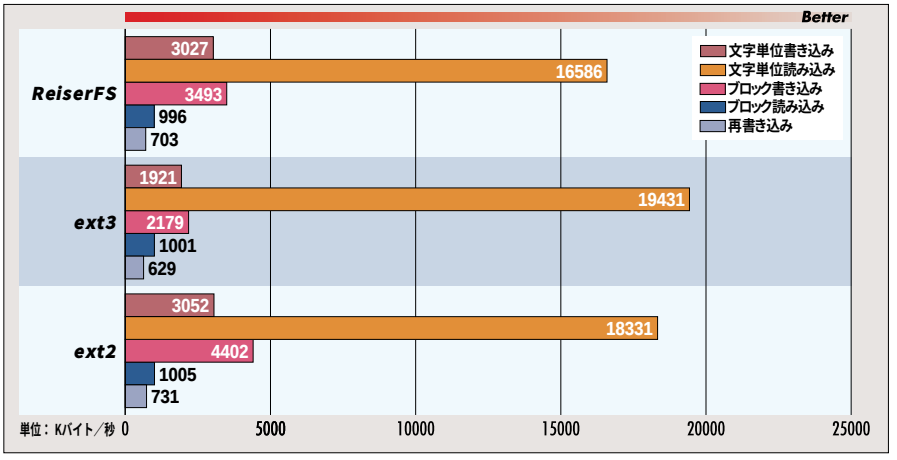
み、削除を行うテストの2種類のベンチマークができる。

前者のテスト結果は**グラフ1**の通りである。数値は1秒当たりに転送できたデータ量（Kバイト）なので、数が大きいほど高速だ。全般にext2が優秀な成績を得ている。差が出ているのは書き込みの部分で、ext3とext2の差は、すなわちジャーナリングログへの書き込み時間のオーバーヘッドと思われる。ReiserFSは、ext2より少し遅い程度だ。

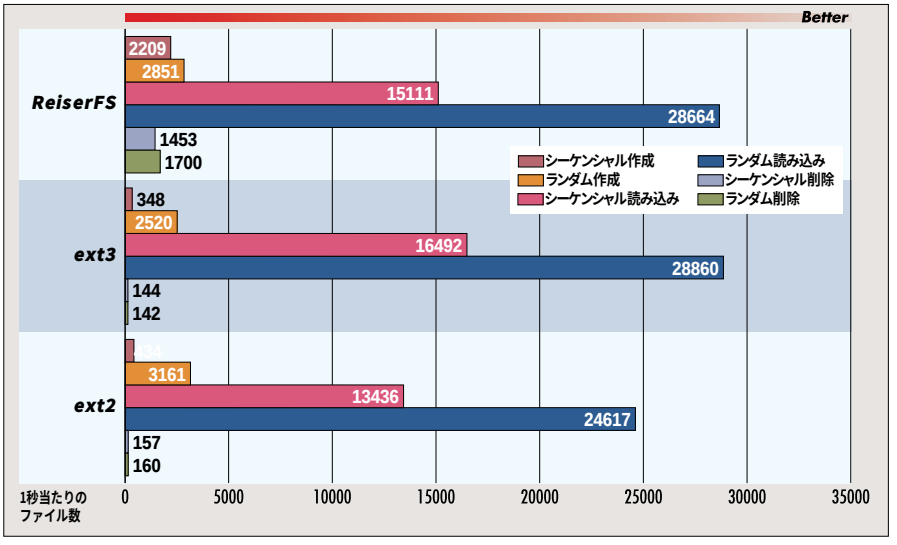
ディレクトリへ複数ファイルアクセスのテスト結果は**グラフ2**のようになった。ファイル名は、7桁の数字と文字を組み合わせて適当に作られる。グラフ中の、シーケンシャルというのは先頭の数字を順番に作っていくもので、ランダムは乱数でファイル名を生成する。ここでの読み込みとは、実際にファイルを読み込むわけではなく、statシステムコールでファイル名からファイルの属性を得ることを行っている。

グラフ2では、シーケンシャル作成、シーケンシャル削除、ランダム削除の3つのテストで、ReiserFSが圧倒的に速いことがわかる。

グラフ1　Bonnie++の結果（1Gバイトのファイルの書き込み／読み込み）



グラフ2　Bonnie++の結果（複数ファイルの作成／読み込み／削除）



ReiserFS	http://www.namesys.com/
ext3	http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html ftp://ftp.uk.linux.org/pub/linux/sct/fs/jfs/
IBM JFS for Linux	http://oss.software.ibm.com/developerworks/opensource/jfs/
SGI XFS Project	http://linux-xfs.sgi.com/projects/xfs/

表1　Linuxに対応するジャーナリングファイルシステム

ジャーナリングファイルシステムは便利

ここまで説明してきたように、ジャーナリングファイルシステムを使うことのメリットは多い。そのためか、ReiserFSのほかにも、SGIのIRIX用のXFSや、IBMのAIX用のJFSなどのジャーナリングファイルシステムがLinuxに移植され公開されている（**表1**、**表2**）。

ジャーナリングファイルシステムは、サーバ向け／ビジネス向けという感覚を持たれるかもしれないが、カーネル2.4でReiserFSが正式採用になったこともあって急速に普及していくのではないだろうか。他のジャーナリングファイルシステムも含めて今後に期待したい。

仕様	ext2／ext3	ReiserFS	SGI XFS	IBM JFS
最大ファイルシステムサイズ	4Tバイト	2Tバイト	8000Pバイト	4Pバイト（512バイト／ブロック） 32Pバイト（4Kバイト／ブロック）
最大ファイルサイズ	2Gバイト （LFSにより4Tバイトまで可能）	2Tバイト （V3.5では2Gバイト）	8000Pバイト	512Tバイト（512バイト／ブロック） 4Pバイト（4Kバイト／ブロック）
ブロックサイズ	1〜4Kバイト	4Kバイト	512〜64Kバイト	512〜4Kバイト

表2　各ファイルシステムの最大ファイルサイズ

Logical Volume Manager

Linuxカーネル2.4で導入された新機能は多岐にわたるが、ジャーナリングファイルシステムやLVM（Logical Volume Manager）は、大規模なサーバ用途に欠かせない機能である。今までLinuxは、SOHOや比較的小規模なサーバとして用いられることが多かった。カーネル2.4の新機能により、大企業の基幹サーバクラスまでもLinuxターゲットになっていくことが期待される。

LVMとは？

今までのLinuxでは、RAIDシステムを除けば、パーティションは1台のディスク上に連続した領域として確保されているものであった。

それに対して、LVMは、複数のハードディスクやパーティションをまとめて1つの論理的な領域（ボリューム）として扱うことができる。LVMで構成された領域は、カーネルからは1つのハードディスクのように見えるため、ディスクやパーティション構成に柔軟性を持たせることが可能だ。

複数のディスクをまとめて仮想的なディスクとして扱う技術としては、ほかにRAID（Redundant Array of Inexpensive／Independent Disks）がある。

RAIDの目的は、ファイルシステムに冗長性を持たせることで、ハードウェアのトラブルからデータを守ること、または複数のディスクに対して並列で読み書きを行うことでディスク性能を向上させること（RAID 0のみ）である。

一方LVMの目的は、システム運用時に動的にディスク／パーティション構成を変えられる柔軟性を持たせることにある。

目的が異なるため、LVMとRAIDを組み合わせて用いることも可能だ。

UNIXのディスクシステム

Linuxを含めたUNIX系のOSでは、複数のディスクやパーティションを各ディレクトリに割り当てて、全体として1つのディスクシステムを構成する。それぞれのディレクトリに必要なサイズや、別のディスクに分けたほうがいいパーティションなどを考慮して、適切なディスクシステムを構成するのは、管理者の腕の見せどころである。ベテランの管理者ともなれば、パーティションの切り方に独自のポリシーを持っており、メーリングリストなどで論争になることもあるくらいだ。

もっとも最近では、大容量のハードディスクが安価に手に入るため、いろいろ悩むよりは、ディスク丸ごとを「/（ルート）」に割り当てるといった使い方をしているユーザーも多いだろう。

特にクライアント用途では、「スワップ」と「/」だけ、もしくはBIOSの制限を考慮して「/boot」を分けて、2〜3個のパーティションで使うといったシンプルな構成でも特に問題はない。筆者が使用する数台のLinuxマシンも、ほとんどがこの構成だ。

ディスク管理の難しさ

だが、複数のユーザーを抱え、専任の管理者がいるようなシステムの場合は、前述のようにディレクトリごとにパーティションやドライブを分けておくのが一般的だ。そのようにすることで、レスポンスがいい快適なシステムが得られるうえに、万が一のディスクトラブル時にも、被害が及ぶ範囲を小さくできるからだ。

ディスクシステムを作成する際には、予想される使用状況を考慮して、各パーティションのサイズを最適と思われる大きさに切り分けていく。

だが「失敗する可能性があることは、必ず失敗する」のマーフィーの法則ではないが、どれだけ熟考してパーティションサイズを決めても、実際に運用していくと予想以上に消費されるパーティションが出てくるものだ。かくしてディスク全体では空きがあるのに、特定のディレクトリ以下の容量が逼迫するはめになる。

この場合、根本的に解決するためには、より大きいディスクを用意して、今あるディスクの内容をバックアップ後にディスク交換をするしかない。しかし複数のユーザーがいるシステムでは、何時間もシステムを停止してこのような作業をしたら、苦情が殺到するにちがいない。

現実には対症療法として、余裕のあるパーティションにディレクトリを作って、そこにシンボリックリンクを張るなどアドホックな対処を行うしかない。それを繰り返すにつれて、ディスクシステムは複雑怪奇になっていく。

LVMの利便性

LVMは、複数のパーティションをまとめて論理的な領域（ボリューム）として扱うことができる。このボリュームは、いったん作ったあとでも新たなパーティションを追加／削除が行える。そのため、システム運用中にディスクを追加して、容量が不足してきたディレクトリのサイズを拡大するといったことが可能だ。もちろんディスクを接続する際には、システムは停止することになるが、LVMなしでディスクを交換する場合よりも、はるかに短時間で再起動できるはずだ。

■使えるのは、事実上ext2のみ

ただし、「ディレクトリのサイズを拡大」するということは、いったん作成したパーティションのサイズを変更することが必要になる。ほとんどのファイルシステムは、サイズの変更を前提に設計されてはいないので、これは危険な作業といえる。

パーティションのサイズを変更するユーティリティとしては、PowerQuest社のPartitionMagicなどが市販されている。PartitionMagicは、Linuxのext2ファイルシステムに対応しているが、LVM上に作成したext2ファイルシステムを扱えるわけではない。

LVM自体はファイルシステムを選ばないので、たとえばLVM上にFATファイルシステムを作るといったことは可能だ。しかしそのFATパーティションのサイズを変えることはできないので、わざわざLVM上に作成する意味はないだろう。

結局LVM上で使う意味があるのは、ext2ファイルシステムのみということになる。ext2に対応したパーティションサイズ変更ユーティリティは、商用のresize2fsやFSF（Free Software Foundation）で作られたext2resizeがあるが、どちらもLVM上のext2パーティションを操作できるからだ。

用語説明

LVMには、区別しにくいうえに聞き慣れない言葉がたくさん使われている。この先混乱しないために、ここでまとめて説明をしておこう。

■物理メディア

ハードディスクそのもの、またはディスク上のパーティション。現実的にはパーティションのことと考えてさしつかえない。

■物理ボリューム（PV）

PVはPhysical Volumeの略で、物理メディアにLVMの管理データを付加したものを指す。つまり、LVM専用のパーティションのこと。

■物理エクステント（PE）

PEは、Physical Extentの略で、PV内部の最少構成単位を表す。デフォルトでは4Mバイトとなっている。

■ボリュームグループ（VG）

VGはVolume Groupの略で、複数のPEをまとめて1個の領域としたものである。まとめられるPEは、別の物理ボリューム上にあってもいいので、実体としては複数のハードディスク（パーティション）をまとめた仮想的なハードディスクのようなものといえる。

■論理ボリューム（LV）

LVはLogical Volumeの略で、VG内のPEをいくつかまとめたものである。いわば仮想的なハードディスクであるボリュームグループ上の、パーティションに相当するもので、実際にこの論理ボリューム上にファイルシステムを作成するようになっている。

各要素の関係や、LVM全体の構成図は、**図1**のようになっている。

実践LVM

実際にLVMを構成してみることにしよう。テスト環境として**表1**のスペックを持つPCを用意した。OSはVine Linux 2.1.5を使用し、カーネルのみ

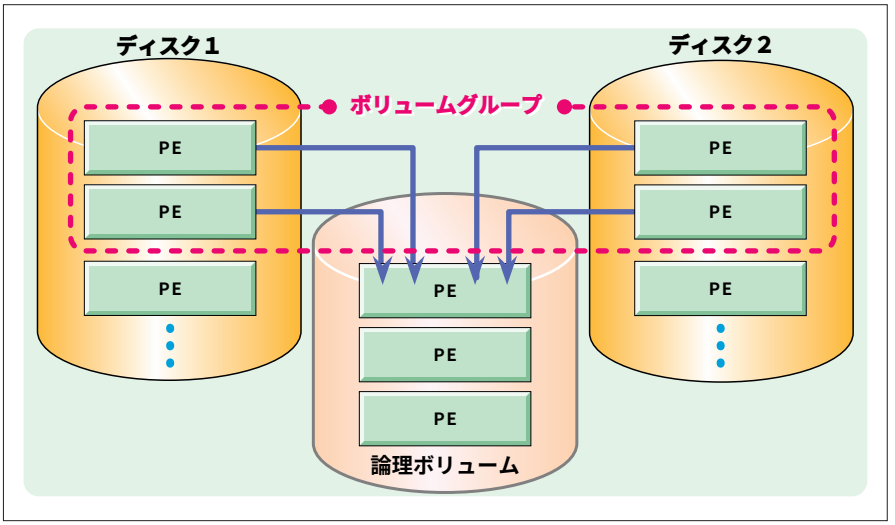


図1　LVMの概念図

2.4.3に変更している。

■開発元

LinuxのLVMは、米Sistina Softwareという会社が開発している（**画面1**）。同社はLVM以外にも、分散した大規模な記憶装置を運用するための、GFS（Global File System）の開発も行っている。

LVMは同社のWebサイトからダウンロード可能で、本稿作成時の最新バージョンは0.9.1-Beta7だった。

インストール方法

LVMの主要なコードは、カーネル2.4.xのソースに組み込まれているが、LVM 0.9.1-Beta7にはカーネル2.4.2用のパッチが含まれていた。このパッチをカーネル2.4.3のソースに適用したところ、特にエラーは出なかったので、そのまま利用した。

■カーネルコンパイル

カーネル2.4.3のソースを展開し、LVMに含まれていたカーネル2.4.2用パッチを適用する。カーネルコンパイル時の設定では、忘れずに「Multi-device support（RAID and LVM）」をモジュール化して組み込むようにしておこう。

■LVMユーティリティのインストール

LVMのtarボールを展開し、ユーティリティプログラムをインストールする。コンパイル自体は一般的な、

```
# ./configure
# make
# make install
```

の手順で行えばいい。

カーネルモジュールが自動的に組み込まれるようにするため、/etc/modules.confに以下の2行を追加する。

```
alias block-major-58  lvm-mod
alias char-major-109  lvm-mod
```

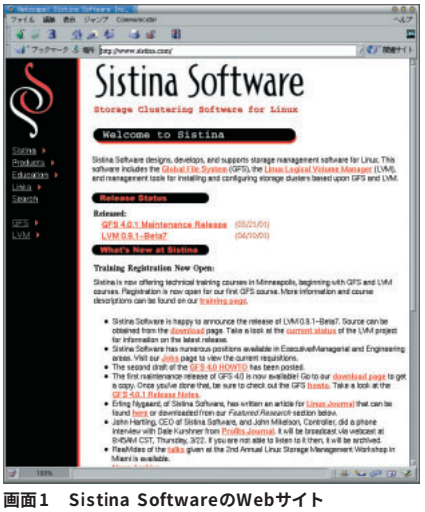
■LVM用パーティションの作成

VMで使うパーティションを準備する。今回は、ハードディスクを2台用意し、それぞれにLVM用のパーティションを複数作成した（**図2**）。パーティションサイズはそれぞれ2Gバイトとした。

パーティションの作成は、通常のext2パーティションと同じようにfdiskコマンドを使うが、作成後にパーティションIDを「8e」に変更しておく必要がある。パーティション一覧の「システム」が「Linux LVM」と表示されているのを確認しよう（**画面2**）。fdiskコマンドが古い場合は、「Unknown」と表示されることもある。

CPU	AMD Athlon 1GHz
マザーボード	ASUS A7V/WOA
チップセット	VIA Apollo KT133
メモリ	PC100 SDRAM 128Mバイト
ハードディスク	Maxtor 34098H4 40Gバイト×2
OS	Vine Linux 2.1.5
カーネル	2.4.3

表1　テストに使用したPCのスペック



画面1　Sistina SoftwareのWebサイト
Linux用LVMの開発元、Sistina Softwareのサイト。

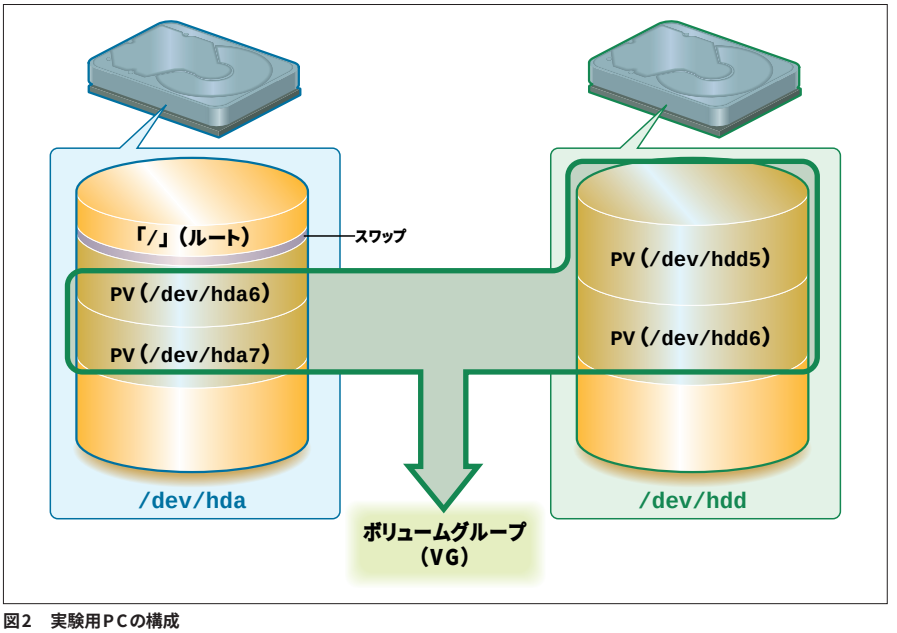


図2　実験用PCの構成

■物理ボリューム（PV）の作成

パーティションにLVM用の管理データを付加して、PVを作成する。コマンドの書式は、

```
# pvcreate パーティション名
```

である。「successfully created」というメッセージが表示されていれば、成功だ（**画面3**）。

■設定ファイルの作成

この段階では、LVM用の設定ファイルは何もないので、vgscanコマンドを用いて設定ファイルを作成する。

vgscanは、すべてのパーティションを検索し、LVM用の設定ファイルである/etc/lvmtabや/etc/lvmtab.d以下のファイルを作成・更新するコマンドだ。VG作成時だけでなく、システム起動時にも実行する必要があるので、

```
/etc/rc.d/rc.sysinit
```

に書き込んでおくといいだろう。

なおLVM用の設定ファイルは、いずれもバイナリ形式になっており、テキストエディタなどで内容を確認することはできない。

■VGの作成

PVを束ねてVGを作成する。コマンドの書式は、

```
# vgcreate VG名 PV名...
```

である。ここでは、VG名を「vg1」とした。

PVは同時に複数指定することができる。エラーや警告メッセージが出力されなければ、成功だ（**画面4**）。

VGの詳細な情報は、vgdisplayコマンドで得られる。

```
# vgdisplay VG名
```

と入力すればいい。PEのサイズ（4Mバイト）や、個数などが確認できる（**画面5**）。

■LV作成

VGからPEを割り当てることで、LVを作成する。LVのサイズは、PEの個数で指定することも可能だが、実サイズで指定するほうがわかりやすいだろう。lvcreateコマンドを用いて、

```
# lvcreate -L 500M -n lv1 vg1
```

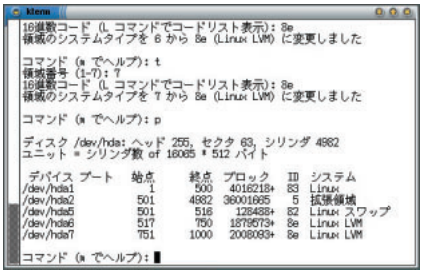
のように指定する。この例では、VG「vg1」から500MバイトをLV「lv1」に割り当てている。エラーが出なければ、このLVは、

```
/dev/vg1/lv1
```

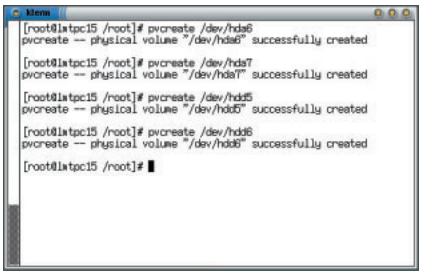
としてアクセスできる。また作成したLVの詳細は、

```
# lvdisplay LV名
```

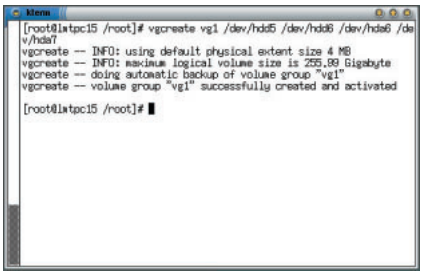
で確認できる（**画面6**）。



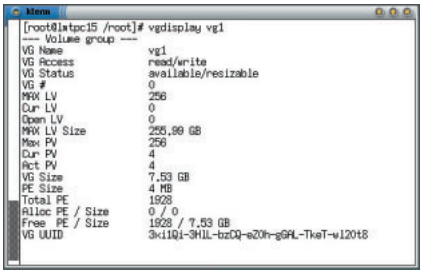
画面2　LVM用パーティションの作成
パーティション作成後、IDを「8e」に変更する。



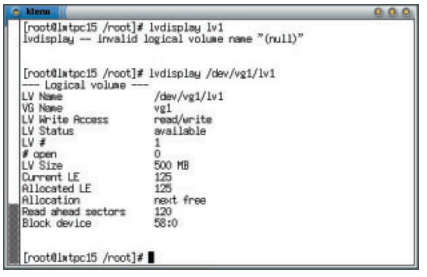
画面3　物理ボリュームの作成
pvcreateコマンドで物理ボリュームを作成する。



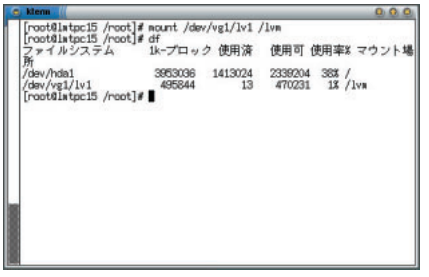
画面4　ボリュームグループの作成
vgcreateコマンドで物理ボリュームを束ねてボリュームグループを作成する。



画面5　ボリュームグループの確認
作成したボリュームグループの情報は、vgdisplayコマンドで確認する。



画面6　論理ボリュームの確認
論理ボリュームの情報は、lvdisplayコマンドで確認する。



画面7　論理ボリュームをマウントする
論理ボリュームでも通常のパーティションと同じように操作できる。

■ファイルシステムの作成

LVは、普通のパーティションと同じように扱うことが可能だ。すなわちext2ファイルシステムを作るなら、

```
# ext2fs /dev/vg1/lv1
```

とすればいい。マウントも同様に、

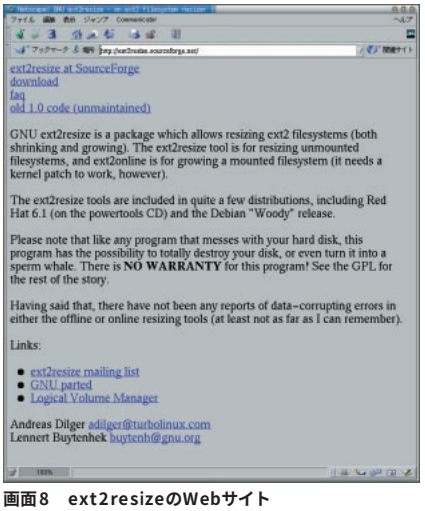
```
# mount /dev/vg1/lv1 /lvm
```

とする（**画面7**）。

パーティションサイズの変更

LVMのLVを作成・利用するには、上述の手順で行えばいい。またVGに余裕があり、ext2ファイルシステムを利用しているなら、LVMのユーティリティに含まれているe2fsadmコマンドを用いて、パーティションサイズを変更することができる。

e2fsadmは、フロントエンドプログラムなので、実際にパーティションサイズを変更するのは、デフォルトではresize2fsというプログラムである。resize2fsは、PowerQuest社の商用ソフトであったが、2000年の4月からGPLに基づいて再配布が可能になっている。



画面8　ext2resizeのWebサイト
なぜかFSFのWebサイトではなく、Sourceforge上にある。

■ext2resize

resize2fsと同様の機能を持ったプログラムとしては、Free Software Foundation（FSF）のext2resizeのほうがよく知られている(**画面8**)。e2fsadmは、ext2resizeを利用することも可能なので、今回はこちらを用いることにする。この場合、リスト1に示した2つのシェル変数をエクスポートする必要がある。

■e2fsadm

e2fsadmコマンドは、ほかのLVMユーティリティとよく似ているので、操作にとまどうこともないだろう。

「−l」オプションは、ファイルシステムのサイズを物理エクステントの数で指定するのに対して、「−L」オプションは、実際のサイズで指定することができる。通常は「−L」オプションのほうが使いやすいだろう。

```
# lvcreate -L 500M -n lv1 vg1
```

のように作成した500Mバイトの LVに作成したext2ファイルシステムを1.2Gバイトに拡張するには、

```
# e2fsadm /dev/vg1/lv1 -L+700M
```

とすればいい。自動的に

- **fsck（e2fsck）**
- **LVの拡張**
- **ファイルシステムのサイズ変更**

が行われる。個別のプログラムを操作しても同じ結果が得られるが、1つのプログラムで操作していくほうがミスも少なくなる。

▼リスト1　ext2resizeの利用に必要なシェル変数

```
# export E2FSADM_RESIZE_CMD=ext2resize
# export E2FSADM_RESIZE_OPTS=""
```

そのほかの機能

上述した基本的な機能以外のLVMの機能としては、以下のようなものがある。

■VGの拡張／縮小

いったん作成したVGに、あとからPVを追加したり、既存のVGからPVを取り除いたりするには、vgextend／vgreduceコマンドを利用すればいい。それぞれ、

```
# vgexpand VG名 PV名
# vgreduce VG名 PV名
```

の書式でPVを指定する。

■スナップショット

LVMの興味深い機能として、スナップショットがある。これは、ある瞬間のLVのコピーを記録するものであり、ディスクのバックアップを取る場合などに有効な機能だ。

一般にディスクのバックアップは、各種アプリケーション／サービスを停止して行う必要がある。そうしなければ、バックアップを取っている最中にディスクの内容が変わってしまうからだ。個人のホームディレクトリ以下程度なら、それほど時間もかからない。しかし大規模なシステムの場合は、バックアップするデータ量も大きいため、

サービスの停止が長時間におよぶことになる。

スナップショットは、lvcreateコマンドを用いて作成する。元のLVとの差分を格納するサイズを指定する。たとえば、/dev/vg1/lv1のスナップショットを作成するには、以下のように指定する。

```
# lvcreate --size 16m --snapshot
  --name snap /dev/vg1/lv1
```

これで/dev/vg1/snapに仮想的な/dev/vg1/lv1のコピーが作成される。マウントしてみると、この2つのLVが同じサイズで、同じ内容であることがわかる。だがスナップショット用のLVが消費するサイズは、元のLVよりもはるかに小さい（lvcreateコマンドの--sizeオプションで指定したサイズ）。

これは、スナップショットが元のLVとの差分を記録しているためだ。たとえば、元のLV上のファイルを削除すると、その削除されたファイルは、スナップショット用LVにコピーされる。

LVMのスナップショットは、変更されたファイルの合計があらかじめ指定したサイズ以上になると、無効になる。マシンを再起動しても同様だ。

LVMの性能

LVMによってファイルシステムに柔軟性を持たせることができるが、当然トレードオフとして性能が低下することになる。それがどの程度なのかを知るために、通常のパーティションと比較してみよう。

ディスク関係のベンチマークソフトとしては、Bonnie＋＋の最新バージョンである1.01bを使用した。

Bonnie＋＋は、POSIX標準のC関数を用いてファイルシステムにアクセスし、その転送速度とCPU使用率を測定する。単一のファイルを用いた転送速度の測定以外に、多数のファイルを生成する測定項目（ファイルシステムの性能測定に利用）がある。今回は全般的な性能の指標となる、ブロック単位のシーケンシャル読み出し／書き込みの項目を利用した。

Bonnie＋＋は多くのコマンドラインオプションを持つが、通常は、テスト領域のサイズ（Mバイト単位）、測定するディレクトリ、そしてログファイルの名前を以下のように指定する。

```
$ bonnie++ -s 300 -d /lvm -m
  test01
```

上の例では、/lvmディレクトリに300Mバイトのファイルを作成して、テストを行い、結果をtest01というファイルに出力している。テスト領域のサイズは、キャッシュの影響を受けないように、ファイルのサイズを実メモリの2倍以上の300Mバイトに設定してある。各条件で測定を3回行い、平均値を測定値としている。

測定した論理ボリュームは、実体としては/dev/hddの先頭から2Gバイトのパーティション（/dev/hdd5）にある。また比較対象の通常パーティションも、/dev/hdaの先頭から4Gバイトに位置しているので、ディスク上の記録位置による差は問題にならないと考えられる。

性能低下は許容範囲？

結果を**表2**に示す。予想どおり読み書きともに速度が低下し、あわせてCPU占有率も上昇している。速度の低下率は、読み込み時は書き込み時の3倍近い。

今回用いたハードディスクは、MaxtorのバリューPC向け製品であり、市場にはこれより速い製品も多い。回転数が7200回転／秒の高速タイプや、SCSI接続のハイエンドの製品を選べば、少々速度が低下したとしても十分な性能が得られるだろう。

またCPU占有率も、CPU性能が引き続き上がり続けている現状では、それほど問題になるとも思えない。

用途にもよるだろうが、6～17%の性能低下と引き換えに柔軟なディスク構成が利用できるならば、LVMを利用する価値はあるといえるだろう。

ただ、**表2**には示していないが、ファイルの生成／削除など、よりファイルシステムの影響が出やすい項目は、LVMによる性能低下が大きくなる傾向にあるようだ。

またLV作成時のオプションとして、複数のVG間に分散記録させることで、性能を向上させるようにすることも可能だ。RAID 0（ストライピング）と同様の手法である。

その場合、1台のディスクが故障するとLV全体が使用不能になるという危険性が、RAID 0と同様に生じることを理解しておくべきであろう。

	書き込み		読み込み	
	速度（Mバイト／秒）	CPU占有率（%）	速度（Mバイト／秒）	CPU占有率（%）
通常のパーティション（/dev/hda1）	31.8	16	28.5	11
LVMの論理ボリューム（/dev/vg1/lv1）	30.1	22	23.7	16
速度低下率	6%	—	17%	—

表2　LVMと通常パーティションの性能比較

Red Hat Linux 7.1でReiserFSを使う

信頼性、速度ともに魅力あるReiserFSを早速使ってみることにしよう。しかし、Red Hat Linux 7.1はカーネル2.4採用とはいうものの、ReiserFSを利用するにはちょっとしたテクニックが必要となる。

Red Hat Linux 7.1（以下、Red Hat 7.1）は、カーネル2.4（kernel 2.4.2-2）を採用しているものの、残念ながらReiserFSを公式にサポートしていない。

しかし、ReiserFSのツールやモジュールはRed Hat 7.1に含まれているのでReiserFSを利用することは可能だ。ただし、公式サポートでないためか、インストーラのカーネルイメージがReiserFSに対応していない。このため、インストール時にReiserFSを使用してファイルシステムを構築することはできないが、インストール後であればReiserFSを利用することが可能だ。

そこで、Red Hat 7.1のインストール後に/パーティションをReiserFSに変更する方法を紹介しよう。ただし、この手順はかなり複雑だ。手順を間違えるとパーティションの内容を失う危険もあることに留意されたい。また、前述したようにRed Hat 7.1ではReiserFSを公式にサポートしていないので、ReiserFSを使う場合は自己の責任のもとで行ってほしい。

/パーティション変更の概要

ここで紹介する/パーティションをReiserFSにする手順は、一度ext2でRed Hat 7.1をインストールし、そのあとファイルシステムをReiserFSにコピーするというものだ。このため、ext2のほかにReiserFSのパーティションがもう1つ必要となる。

ハードディスクを複数接続してパーティションを作成してもいいが、今回は1つのハードディスクですべて実行するものとし、ハードディスク内のパーティションを次のように3つに分割することにする。

```
/dev/hda1       ReiserFS
```
最終的に/となるパーティション
```
/dev/hda2       Linuxスワップ
```
スワップパーティション
```
/dev/hda3       ext2
```
最初にインストールするパーティション。ReiserFSを構築後、/dev/hda1にコピーする

それぞれのパーティションのサイズはインストールするパッケージに合わせて確保する。Red Hat 7.1では「全部」インストールで約2.4Gバイト必要となる。仮に「全部」インストールを行うのであれば、/dev/hda3は最低でも2.4Gバイト、/dev/hda1は最終的な/パーティションとして運用するうえ、/dev/hda3の内容をコピーすることになるので/dev/hda3と同等以上のサイズを確保する必要がある。

それでは、実際の手順を解説する前に、まずおおまかな手順を紹介しておこう。

①通常の方法でRed Hat 7.1をインストール（このとき、ReiserFSで利用するためのパーティションを用意しておく）
②ReiserFSを構築し、インストールしたRed Hat 7.1（ext2）イメージをReiserFSにコピーする
③システムの設定を変更し、ReiserFSを/パーティションとしてマウントできるようにする

以上の手順となる。これだけを見ると簡単そうに思えるが、実際の手順はかなり複雑だ。途中で手順を誤るとファイルシステムを壊してしまったり、起動できなくなってしまうので、一度最後までじっくり読んでから慎重に作業してほしい。

Red Hat 7.1のインストール

それでは実際の手順を順を追って解説していこう。まずはRed Hat 7.1のインストーラを起動し、通常の手順でインストールを行う。このとき、「ディスクパーティション設定」画面が表示されたら、「fdiskを使用して手動でパーティションを設定する［エクスパートのみ］」を選択し、［次へ］ボタンをクリックする（**画面1**）。fdiskを使用してパーティションを設定する理由は、Disk Druidでは使用しない（領域を確保するだけの）パーティションを作成できないためだ。面倒だが、fdiskを使用しよう。

fdiskを使用する場合、「fdiskを実行

するドライブを選択して下さい」と表示されるので、［hda］をクリックする（**画面2**）。

ここで、fdiskの実行画面がウィンドウ内に表示されるので、次のようなパーティションを作成する。

```
/dev/hda1 Linuxネイティブ（ext2）
/dev/hda2 Linuxスワップ
/dev/hda3 Linuxネイティブ（ext2）
```

作成する手順は**画面3**を参照してほしい。なお、サイズは自分の環境に合わせて設定してほしい。

パーティションの設定が終了したら、念のため「p」と入力し、意図しているとおり正しくパーティションが設定されているかどうかを確認する。正しければ、「w」と入力し、パーティション設定を保存する。**画面4**はパーティション設定を保存する直前の画面だ。もし、設定が間違っていたり、取りやめたいときは「q」と入力すれば設定は破棄され、ハードディスクは何も変更されない。

fdiskが終了すると**画面2**に戻るので［次へ］をクリックする。これでパーティションの設定は終了だ。

■/パーティションの作成

次に、各パーティションにどのマウントポイントを設定するかを指定する（**画面5**）。hda1はまだ使用しないので、「＜設定しない＞」のままにしておき、hda2も「＜スワップ＞」パーティションとして設定されているので、こちらも変更しない。最後のhda3は/パーティションに割り当てるので、［編集］をクリックし、「マウントポイント」に「/」を入力し（**画面6**）、［OK］をクリックする。設定が間違いなければ（**画面7**）、［次へ］をクリックし、hda2とhda3をフォーマットする。

あとは通常のインストール手順と同様だ。必要な設定を行いインストールを完了させよう。

ReiserFSパッケージのインストール

インストールが完了したら、システムを起動しよう。正常に起動できれば、いよいよReiserFSを構築することになる。ただし、インストール時の「インストールの種類で」、「カスタム」を選択し、パッケージの選択で「全部」以外を選択した場合はReiserFSユーティリティがインストールされないので、

画面1　ディスクパーティション設定

画面2　fdiskを実行するドライブの選択

画面4　fdiskの実行画面

画面5　マウントポイントの設定

「全部」以外でインストールした場合はReiserFSパッケージをインストールしておく必要がある。

ReiserFSパッケージのインストールはrpmコマンドを使って簡単にインストールすることができる。まず、Red Hat 7.1のDisc 2をCD-ROMドライブにセットし、rootユーザー権限で次のようにしてマウントする。

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

次に、rpmコマンドを使ってreiserfs-utils-3.x.0f-1.i386.rpmをインストールする。

```
# rpm -ivh /mnt/cdrom/RedHat/RPMS/
reiserfs-utils-3.x.0f-1.i386.rpm
```

以上でReiserFSパッケージのインストールは完了だ。

ReiserFSの構築

それではいよいよReiserFSを構築しよう。次のようにして、インストール時に作成した/dev/hda1をReiserFSでフォーマットする。

```
# mkreiserfs /dev/hda1
```

このとき、当然/dev/hda1にあるデータはすべて消去される。もともとデータは入っていないので問題ないはずだが、コマンドを実行する際には十分注意すること。コマンドを実行すると、**画面8**のようにデータが消去されるという「注意」が表示されるので、間違いなければ「y」と入力してフォーマットを実行する。

フォーマットが完了したら、システムを再起動する。これで/dev/hda1に/パーティションをコピーするReiserFSが構築できた。

ReiserFSのマウント

システムが再起動したら、次のようにして先ほど作成したReiserFSパーティションをマウントする。

```
# mkdir /mnt/reiser
# mount -t reiserfs /dev/hda1 /
mnt/reiser
```

ここでは、/mntにreiserというマウントポイントを作成し、/dev/hda1（ReiserFS）をマウントしている。

マウントしたら、正常にマウントできているかどうかmountコマンドを実行して確認してみよう（**画面9**）。

正しくマウントされていれば、「/dev/hda1 on /mnt/reiser type reiserfs (rw)」という行が表示されるはずだ。この状態で、/mnt/reiserディレクトリ以下はReiserFSとして利用できることになる。

次に、ReiserFSを自動的にマウントできるように/etc/filesystemsを**リスト1**のように変更する。

この変更は必須ではないが、このように変更しておくとmountコマンドでファイルシステムタイプを指定しない場合でもReiserFSをマウントできるようになる。このほうが便利だし、使いやすいだろう。

画面6　/マウントポイントの指定

画面7　マウントポイントの設定画面（設定終了）

```
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
    ALL DATA WILL BE LOST ON '/dev/hda1'! (y/n)
```

画面8　ReiserFSフォーマット時の「注意」

```
# mount
/dev/hda3 on / type ext2 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/hda1 on /mnt/reiser type reiserfs (rw)
```

ReiserFSがマウントされている

画面9　mountコマンドによるマウント状況の確認

/パーティションのコピー

ReiserFSが無事マウントできたところで、現状のext2の/パーティション（/ディレクトリ以下）をReiserFSにコピーする。このとき、安全のためにシングルユーザーモードに移行しよう。

```
# init S
```

シングルユーザーモードに移行したら、次のようにしてファイルシステムを/mnt/reiserにコピーする。

```
# find / -print0 -mount | cpio -p
```

```
コマンド（m でヘルプ）：n ―― 1つめのパーティション/dev/hda1を作成するために「n」を入力
コマンドアクション
   e   拡張
   p   基本領域（1-4）
p ―― 「p」を入力
領域番号（1-4）：1 ―― /dev/hda1を作成するので「1」を入力
最初 シリンダ（1-XXX，初期値 1）： ―― 初期値を使うので、Enterキーを押す
初期値 1 を使います
終点 シリンダ または +サイズ または +サイズM または +サイズK（1-XXX，初期値 XXX）： +2000M. ―― 2000M（2Gバイト）を確保するために「+2000M」と入力

コマンド（m でヘルプ）：n ―― 2つめのパーティション/dev/hda2を作成
コマンドアクション
   e   拡張
   p   基本領域（1-4）
p ―― 「p」を入力
領域番号（1-4）：2 ―― /dev/hda2を作成するので「2」を入力
最初 シリンダ（XXX-XXX，初期値 XXX）：
初期値 XXX を使います
終点 シリンダ または +サイズ または +サイズM または +サイズK（XXX-XXX，初期値 XXX）： +512M ―― 512Mバイトを確保するために「+512M」と入力

コマンド（m でヘルプ）：n ―― 3つめのパーティション/dev/hda3を作成するために「n」を入力
コマンドアクション
   e   拡張
   p   基本領域（1-4）
p ―― 「p」を入力
領域番号（1-4）：3 ―― /dev/hda3を作成するので「3」を入力
最初 シリンダ（XXX-XXX，初期値 XXX）： ―― 初期値を使うので、Enterキーを押す
初期値 XXX を使います
終点 シリンダ または +サイズ または +サイズM または +サイズK（XXX-XXX，初期値 XXX）： ―― 残り全部を割り当てるのでEnterキーを押す（必要なサイズを入力してもOK）

コマンド（m でヘルプ）：t ―― パーティションタイプを変更するために「t」を入力
領域番号（1-4）：2 ―― /dev/hda2を変更するので「2」を入力
16進数コード（L コマンドでコードリスト表示）：82 ―― 「82」と入力してパーティションコードを変更
領域のシステムタイプを 2 から 82（Linux スワップ）に変更しました

コマンド（m でヘルプ）：p ―― 変更を確認するために「p」と入力

 デバイス    ブート     始点     終点     ブロック    ID   システム
/tmp/hda1              X      XXX     XXXXXXX    83    Linux
/tmp/hda2            XXX      XXX      XXXXXX    82    Linux    スワップ
/tmp/hda3            XXX      XXX     XXXXXXX    83    Linux

コマンド（m でヘルプ）：w ―― 変更に問題なければ「w」と入力して変更を保存する（「q」で変更を中止）
```

画面3　fdiskの実行画面

```
-0 -d -m -u /mnt/reiser
```

ここではfindコマンドとcpioコマンドを組み合わせてコピーしている。これは、cpioコマンドはコピーする際のファイル名がフルパスで必要なためだ。このため、findコマンドを使ってコピーするファイルのフルパス名を取得している。

findコマンドで指定しているオプションの意味は、「/」が/ディレクトリ以下すべてを検索する指定、「-print0」がファイル名をフルパスで出力する指定、「-mount」がほかのパーティションのファイルシステムを検索しないという指定だ。「-mount」を指定することで、/procディレクトリや/mnt/reiserディレクトリそのものをコピーしないようにしている。

cpioコマンドで指定しているオプションの意味は、「-p」がディレクトリからディレクトリへのコピーを行うパススルーモードの指定、「-0」がヌル文字で終了するファイル名を処理する指定、「-d」が必要に応じてディレクトリを作成する指定、「-m」がコピー時にコピー元のファイル変更時刻を保持する指定、「-u」がファイルの自動上書きを行うという指定だ。

このコマンドを実行することで、/ファイルシステム以下のファイルを/mnt/reiserに完全にコピーすることができる。

fstabの修正

/ディレクトリのコピーが終わった段階で、2つの/ディレクトリツリーが存在することになる。このうち、ReiserFSにコピーした/ディレクトリツリーを/にマウントするために、コピーしたReiserFS側のfstabファイルを**リスト2**のように変更する。変更する際は、誤って/etc/fstab（/dev/hda3にあるfstab）を変更しないように十分注意すること。必ず、/mnt/reiserfs/etc/fstab（/dev/hda1）にある、ReiserFS側のファイルを変更するようにする。この設定を間違えた場合、システムが起動しなくなってしまうからだ。

ブート用RAMディスクイメージの作成

Red Hat 7.1のカーネルイメージにはReserFSが組み込まれていないので、システム起動後にモジュールとして組み込むことになる。しかし、ReiserFSのモジュールは/パーティションに保存されている。このため、システム起動時にReiserFSをマウントすることができない。つまり、ReiserFSモジュールを読み込むためには、ReiserFSがマウントされていなければならないというわけだ。いわば、缶詰の中に缶切りが入っている状態といえる。そこで、ブート用RAMディスクイメージを作成し、そこにReiserFSのモジュールを組み込んでおき、このモジュールを使って起動時にReiserFSをマウントするようにする必要がある。

ブート用RAMディスクイメージの作成は次のように行う。

```
# mkinitrd --with reiserfs /boot/
initrd-reiser.img 2.4.2-2
```

このコマンドを実行するこで、/bootディレクトリにinitrd-reiser.imgというブート用RAMディスクイメージが作成される。

LILOの変更

次に、このイメージをブート時に組み込むように/etc/lilo.confに変更を加える（変更は/dev/hda3のもの）。また、同時にReiserFSを/にマウントできるように変更を加える（**リスト3**）。

変更できたら、liloコマンドでMBRにlilo.confを書き込む。

```
# lilo
```

この変更により、ブート時のラベル

▼リスト1　/etc/filesystemsの変更

```
xt2
nodev proc
nodev devpts
iso9660
vfat
hfs
reiserfs        ← この行を追加
```

▼リスト2　fstabの変更（/mnt/reiserfs/etc/fstab）

```
/dev/hda1               /                       reiserfs defaults       0 0   ← この行を追加
#LABEL=/                 /                        ext2    defaults        1 1   ← コメントアウト
/dev/fd0                /mnt/floppy             auto    noauto,owner    0 0
none                    /proc                   proc    defaults        0 0
none                    /dev/pts                devpts  gid=5,mode=620  0 0
/dev/hda2               swap                    swap    defaults        0 0
/dev/cdrom              /mnt/cdrom              iso9660 noauto,owner,kudzu,ro 0 0
```

によって、ReiserFSとext2の/ディレクトリを切り替えて使うことが可能となる。もし、設定を間違えてシステムが起動できない状態となっても、「ext2」ラベルを選択すれば、/dev/hda3をマウントした状態でシステムを起動することができる。

ブートシーケンススクリプトの変更

システムの起動時には、自動的にfsckコマンドが実行される。しかし、fsckコマンドはext2用のファイルシステムの整合性チェックユーティリティなので、ReiserFSでは利用できない。もっとも、ReiserFSではマウント時に自動的にファイルシステムの整合性がチェックされるのでfsckコマンドを実行する必要はない。そこで、起動時にfsckコマンドを実行しないように設定しておく。

システム起動時のfsckコマンドは、システム起動時に実行されるrc.sysinit内で実行される。実行される部分は2カ所あるので、この部分を**リスト4・5**のように修正する。ただし、変更するのは/mnt/reiser/etc/rc.dにあるrc.sysinitファイルだ。間違えないように注意しよう。

この変更により、/ファイルシステムがReiserFSの場合はfsckを行わないようになる。

ここまでの設定が終われば、起動に関する設定は終了だ。次に、シャットダウン時のシーケンスファイルを変更する。

シャットダウンシーケンススクリプトの変更

/パーティションはアンマウントすることができない。このため、シャットダウン時に/にマウントされているパーティションをリードオンリー属性として再マウントすることによって、ファイルシステムをクリーン（ファイルシステムの整合性が保たれた状態）にし

▼リスト3　/etc/lilo.confの修正

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message.ja
linear
default=linux

image=/boot/vmlinuz-2.4.2-2
      label=linux
      read-only
      root=/dev/hda1
      initrd=/boot/initrd-reiser.img

image=/boot/vmlinuz-2.4.2-2
      label=ext2
      read-only
      root=/dev/hda3
```

（この部分を追加）

（ラベル名を「linux」から「ext2」に変更）

▼リスト4　rc.sysinitスクリプトの変更箇所1（抜粋）

```
_RUN_QUOTACHECK=0
ROOTFSTYPE=`grep " / " /proc/mounts | awk '{ print $3 }'`
#if [ -z "$fastboot" -a "$ROOTFSTYPE" != "nfs" ]; then
if [ -z "$fastboot" -a "$ROOTFSTYPE" != "nfs" -a "$ROOTFSTYPE" != "reiserfs" ];
then
        STRING=$"Checking root filesystem"
        echo $STRING
        initlog -c "fsck -T -a $fsckoptions /"
```

（この行をコメントアウト）

（この行を追加）

▼リスト5　rc.sysinitスクリプトの変更箇所2（抜粋）

```
_RUN_QUOTACHECK=0
# Check filesystems
#if [ -z "$fastboot" ]; then
if [ -z "$fastboot" -a "$ROOTFSTYPE" != "reiserfs" ]; then
        STRING=$"Checking filesystems"
        echo $STRING
        initlog -c "fsck -T -R -A -a $fsckoptions"
```

（この行をコメントアウト）

（この行を追加）

ている。しかし、Red Hat 7.1のシャットダウンシーケンススクリプトでは、クリーンが行われるのはext2ファイルシステムだけになっている。そこで、シャットダウンシーケンスファイルを修正して、ReiserFSもリードオンリー属性で再マウントしてクリーンにするように変更する。変更するのは/mnt/reiser/etc/rc.d/init.dにあるhaltファイルだ。このファイルを**リスト6**のように修正する。これも間違えないように注意しよう。

この修正を行わないとReiserFSが正常にクリーンにならず、次回起動時にマウントする際、ファイルシステムの整合性を保つためにジャーナリングに残っているログのリプレイ（再実行）が行われる。そのため、リプレイのメッセージが大量に表示されることになる。

以上ですべての設定は完了だ。あとはシステムを再起動すればよい。このとき、LILOで「linux」ラベルを選択すればReiserFSが/パーティションとしてマウントされるはずだ。システム起動後、次のようにすればReiserFSがマウントされているか確認できる。

```
# mount
```

正常にマウントできていれば/パーティションをReiseFSに移行できたことになる。

もし、システムが起動できなかったり、ファイルシステムが正常にマウントできていないのであれば、ここまでの手順で何か間違っているものがあるはずだ。なお、LILOで「ext2」ラベルを選択すればext2の/パーティションをマウントすることができるので、もう一度システムを復旧することができるだろう。

さて、/dev/hda3に残されたext2の/パーティションだが、もちろん消してしまってもいいのだが、reiserfsckコマンドなどを実行する際に利用できるので（reiserfsckコマンドはマウントされた状態では実行できない）、安定運用が確認できるまでは残しておいたほうがいいだろう。

参考までに、ReiserFSのファイルシステムチェックを行う方法を紹介しよう。

システムを再起動して、LILOで「ext2」ラベルを選択し、ReiserFSパーティションがマウントされていない状態で起動する。

次に、rootユーザー権限で次のようにコマンドを実行する。

```
# reiserfsck /dev/hda1
```

このとき、本当に実行するか確認してくるので、「Yes」と入力する。

▼リスト6　haltスクリプトの変更箇所（抜粋）

```
# Remount read only anything that's left mounted.
#echo $"Remounting remaining filesystems (if any) readonly"
#mount | awk '/ext2/ { print $3 }' | while read line; do
mount | awk '/ext2|reiserfs/ { print $3 }' | while read line; do
    mount -n -o ro,remount $line
done

runcmd $"Unmounting proc file system: " umount /proc
```

この行をコメントアウト

この行を追加（「/ext2/」を「/ext2｜reiserfs/」にする）

Column

ReiserFSとディストリビューション

インストーラがReiserFSに対応していないRed Hat 7.1で/パーティションをReiserFSにするのは難しい作業だが、SuSE Linux 7.1は標準でインストーラがReiserFSに対応しており、/パーティションをReiserFSで構築することができる。ただし、SuSEはカーネル2.2系列を採用している（インストール時にカーネル2.4を選択することもできる）。

また、同様にカーネル2.2系列のVine Linux 2.1、2.1.5も非公式ながら/パーティションをReiserFSで構築することができる。こちらは、インストーラが対応しているわけではないが、インストール作業中にコンソール画面でReiserFSパーティションをフォーマットすれば以降のインストールは自動で行ってくれる。この方法が利用できるのは、インストーラのカーネルがReiserFSに対応しているためだ。この方法の詳しい手順は、Vine LinuxのCD-ROMのdocディレクトリにある.secret.txtファイル（.secret.txt.bz2でアーカイブされている）に記載されている。

なお、ReiserFSのモジュールは比較的大きいため、インストーラのカーネルにReiserFSのモジュールを組み込むと、フロッピーに入りきらない容量となるため、CD-ROMブートによるインストールか、NFSインストールでのみこの方法が利用できる。

残念ながらカーネルにReiserFSモジュールが組み込まれていないRed Hat 7.1ではこの方法でインストールすることはできない。

いずれにせよ、カーネル2.2系列のインストーラのほうがReiserFSを容易に利用できるというのはちょっとした皮肉とも言える。

Red Hat Linux 7.1のアップデートインストール

インストーラにはアップデートインストールというインストール方法が用意されている。ここでは、Red Hat Linux 6.2からRed Hat Linux 7.1へアップデートする際のバックアップ方法や、注意点などを解説する

Linuxインストーラのほとんどに、「アップデートインストール」という項目がある。これは、インストール済みのLinuxをアップデートするためのインストール方法だ。しかし、このアップデートインストールはどのようなことを行うのだろうか。編集部でも、多くのLinuxをインストールしてきたが、あまりアップデートインストールを行ったことはない。そこで、実際にRed Hat Linux 6.2にRed Hat 7.1をアップデートインストールし、アップデート後に必要となる設定などを紹介することにしよう。

バックアップ

アップデートインストールを行う前に、念のため必要なファイルをバックアップしておいたほうがいいだろう。アップデートに失敗してシステムが起動できなくなってしまっても復旧できるし、以前の設定を参照することもできるからだ。

システムのフルバックアップはdumpコマンドが便利だ。dumpコマンドは通常、テープデバイスなどにファイルシステムを作成せずに、直接Rawデバイスとしてデータを書き出すが、ファイルシステムにバックアップを作成することもできる。ただし、別パーティションが必要となる。なお、フルバックアップを行う場合はシングルユーザーモードで実行するほうが安全だ。

次のコマンドは、/mnt/hda3にマウントされているファイルシステムに/パーティションをbackupというファイル名でバックアップするためのコマンドだ。

```
# init S
# mkdir /mnt/backup
# mount /dev/hda3 /mnt/backup
# dump -0f /mnt/hda3/backup
```

もし、CD-RやCD-R/W、MOなどがあれば/mnt/hda3/backupを保存しておけばより安心だろう。

なお、バックアップしたファイルを復元（リストア）するには、restoreコマンドを使う。

```
# restore -xf /mnt/hda3/backup
```

ここで指定している「x」オプションは、カレントディレクトリ以下にバックアップファイルをリストアするための指定だ。あらかじめ、/tmpディレクトリなどに移動してから行う必要がある。

また、ファイルやディレクトリを指定することで、指定したファイルやディレクトリのみをリストアすることができる。

```
# restore -xf /mnt/hda3/backup
/etc/inetd.conf /var
```

フルバックアップは指定したパーティションをすべてバックアップするため、かなりの時間とディスク領域が必要となる。特に、ディスク領域は重要な問題で、バックアップするための十分な領域が確保できない場合もあるだろう。このような場合は必要最低限のディレクトリだけに絞ってバックアップを取っておくといいだろう。最低限必要となるディレクトリはシステムによって異なるが、おおよそ次のようなものだろう。

/etc

いわずと知れた、設定ファイルなどが保存されているディレクトリだ。現在稼働中のLinuxの設定ファイルの多くが格納されている。

/var

DNSの設定ファイルや、ユーザーのメールスプールなどが保存されているディレクトリだ。

/home

ユーザーのホームディレクトリが保存されているディレクトリだ。Red Hat 6.2では、Webページのデータや SAMBA、FTPのデータが保存されているディレクトリでもある。

/root

rootユーザーのホームディレクトリだ。このディレクトリにはあまり必要なファイルはないはずだが、使用状況によってはバックアップの対象となる。

上から順に優先順位が高いだろう（本来なら/homeが一番優先順位が高

いはずだが、Linuxを起動させるという意味では優先順位が低くなる）。最低でもこれらのディレクトリのバックアップを取っておけば、システムの復旧はできるだろう。

ディレクトリ単位でバックアップを作成するのであれば、tarコマンドが便利だ。tarコマンドはディレクトリやファイルをアーカイブとして1つのファイルにまとめることができる。次のようにして、各ディレクトリをバックアップしておこう。

```
# tar -zcvf etc.tar.gz /etc
# tar -zcvf var.tar.gz /var
# tar -zcvf home.tar.gz /home
# tar -zcvf root.tar.gz /root
```

作成したアーカイブファイルは、できれば別のファイルシステムにコピーしておく。CD-RやCD-RW、MOなどにコピーできればベストだ。

なお、tarファイルからカレントディレクトリにファイルを取り出すには次のようにする

```
# tar -zxvf etc.tar.gz
```

バックアップが完了したら、いよいよアップデートインストールを行おう。

アップデートインストール

アップデートインストールは、その名のとおりディストリビューションをアップデートするためのインストール方法だ。しかし、実際にインストーラが行うアップデートとはどのようなものだろうか。

実は、アップデートインストールを行うと、パッケージは最新のものに置き換えられるものの、多くの設定が「デフォルト」となってしまう（ほとんどのシステム関連のファイルは書き換えられる）。以前の設定値などは.rpmsaveという拡張子で保存されるので、このファイルを参照しながら、使用していたサービスの設定を設定しなおすことになる。「以前の設定値を参照して、自動的にアップデート」とはいかないのだ。

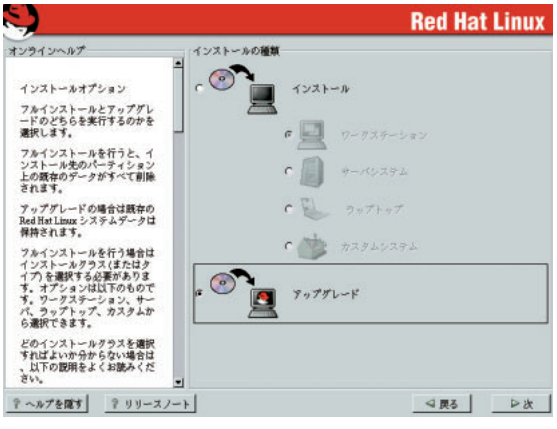
ただし、/homeディレクトリやユーザー、パスワードなどはそのまま利用することができる。

なお、アップデートされたあとでも以前の設定ファイルをそのまま使用している場合は、新しい設定ファイルが.rpmnewという拡張子で保存される。たとえば、/etc/named.conf、/etc/rc.d/rc.localなどが該当する。こちらは以前の設定が利用されているので変更する必要はないだろう。

さて、それでは実際にアップデートインストールを行ってみよう。

通常のインストールと同じようにインストーラを起動し、「インストールの種類」で「アップグレードインストール」を選択する（**画面1**）。「アップグレードインストール」を選択した場合、インストールするパッケージを選択することもできるが、通常は選択する必要はないだろう（**画面2**）。あとは、通常のインストールと同じようにインストール作業を進めるだけだ。

インストールが完了したら、システムを再起動する。



画面1　アップグレードインストールの選択

画面2　アップグレードパッケージをカスタマイズ

```
$ ls /etc/*.rpmnew /etc/*.rpmsave
/etc/inetd.conf.rpmsave  /etc/mime.types.rpmsave   /etc/smb.conf.rpmsave
/etc/ld.so.conf.rpmnew   /etc/named.conf.rpmnew    /etc/smbusers.rpmsave
/etc/lilo.conf.rpmsave   /etc/sendmail.cf.rpmsave  /etc/syslog.conf.rpmnew
/etc/localtime.rpmnew    /etc/services.rpmsave     /etc/termcap.rpmsave
```

画面3　/etcディレクトリ内の変更されたファイルの表示

設定ファイルの復旧

システムを起動すると、今まで設定していたいくつかの機能がデフォルトに戻っていることに気づくだろう。まずは、/etcディレクトリにある.rpmsaveを拡張子に持つファイルを表示してみよう（**画面3**）。ここで表示されたファイルが、アップデートによって置き換えられた/etcディレクトリ内の設定ファイルだ。

sendmail.cfなど、ほとんどの設定ファイルはそのまま置き換えれば使えるものだが、一部のサービスは、新しい設定ファイルを作成しなければならない。

まず、どれだけの設定ファイルが置き換えられているかをfindコマンドで把握しておこう。

```
# find / -name "*.rpmsave"
```

それでは主要なサービスを例に、設定ファイルを「戻す」方法を紹介していこう。

■/etc/smb.conf

Windowsネットワークサービスを提供するSAMBAの設定ファイルは、ディレクトリの構造が変更された。新しい設定ファイルは、/etc/sambaディレクトリに置かれている。SAMBAの設定ファイルであるsmb.confはそのまま利用することもできるが、パスワードの管理方法が変わったので、再度ユーザーを作成する必要がある。

■/etc/httpd/conf/httpd.conf

httpd.confはApacheの設定ファイルだ。アップデートによって、Apacheのドキュメントルートが変更されるので、httpd.confの変更が必要だ。以前のWebコンテンツは/home/httpdディレクトリ以下に保存されているので、新しいドキュメントルートである、/var/www/htmlに移動する必要がある。

■/etc/X11/XF86Config

XFree86の設定ファイルであるXF86Configも設定しなおさなければならない。こちらは、特殊な設定をしていない限り、Xconfiguratorなどを使って再設定したほうが簡単だろう。

■/etc/sendmail.cf

sendmailの設定ファイルは複雑なため、再度設定しなおすことを考えると気が遠くなるかもしれない。しかし、このファイルはそのまま使用することができるので、新しいsendmail.cfに上書きすればいいだろう。

■/etc/inetd.conf

各サーバの起動をコントロールするスーパーサーバ、inetdは、アップデートによってxinetdに変更される。xinetdでは、各サーバの設定方法が大幅に変更されているため、inetd.confの移行が最も複雑で大変だろう。

inetdの設定ファイルはinetd.confだけで、このファイル1つですべてのサーバをコントロールしていた（**リスト1**）。一方、新しいxinetdでは、xinetd.confに加え、各サーバごとに設定ファイルが用意されている。より細かく各サーバをコントロールするためだ。これらのサーバの設定ファイルは/etc/xinetd.dディレクトリに保存されている。たとえば、telnetであれば/etc/xinetd.d/telnetとなる（**リスト2**）。

inetdでは、サーバを起動する場合はinetd.confに記述された各サーバの設定行のコメントを外せばよかったが、xinetd.confでは/etc/xinet.d/ディレクトリにある各設定ファイルごとに設定しなければならない。たとえば、**リスト2**のtelnetであれば、

```
disable     = no
```

と設定することになる。これを必要なサーバごとに繰り返す必要がある。

▼リスト1　/etc/inetd.conf（抜粋）

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

▼リスト2　/etc/xinet.d/telnet

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
#       unencrypted username/password pairs for authentication.
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes
}
```

爆安Linuxマシンを実現せよ!

超実践的 電脳お買い物ガイド for Linuxers

～予算4万円・秋葉原漂流・ケチケチ主婦感覚～

文：4万円Linuxマシン特別対策班（王大人（ワンターレン）＆にゃ～@編集部）
Text : Linux magazine

4万円Linuxマシンを求めて
～事前調査とお見積り～

もはやLinux magazineの恒例となった感のある「○万円で～」企画。今回は予算4万円でPCを組み上げようというお話。

デフレスパイラル真っ只中、PCパーツの価格破壊が進む現状であるとはいえ、果たして4万円マシンは実現可能なのか？　特別対策班を結成した我々（2名）は、さっそく行動を開始することに……。

作戦会議

過去、本誌月刊化第1号（1999年10月号）においてライター・上間俊雄氏が予算5万クリアを達成（創刊当時からやってたのネ）、また2000年6月号においては、不肖・にゃ～@編集部が秋葉原で暴走、物欲のおもむくままに買い物したあげく5万円の予算を軽々とオーバーしたという実績（？）がある。ただし、今回は1万円も予算が少ないのだ。

より慎重な事前調査が必要であろうと判断した私は、秋葉原でのパーツ価格を掲載しているいくつかのWebサイト（**表1**）を参照しながら、ハードウェア＆秋葉原情報に精通するLinux magazineラボ・1号の協力を得て作戦会議を開いたのであった。以下、そこで決定された要点をまとめておこう。

■スペックと用途

いくら安くても、使いものにならないマシンではしかたがない。サクサクっとLinuxがインストールできて、XFree86＋GNOMEデスクトップが快適に動作して、さらにサウンドも再生できたりするとうれしい。で、やや具体性には欠けるが「個人ユーザーのセカンドマシン」をターゲットとして想定することにした。テストは編集部のLAN環境で行うのでネットワークインターフェイスも必須だ。

できれば、メモリは128Mバイト、ハードディスクが10Gバイト程度ほしいところだが、このあたりは予算と相談か？

■CPU＆マザーボード

Pentium 4では、いきなり予算をオーバーしてしまう。Pentium Ⅲ、Athlonでも、残金はよくて半分である。今回はCeleronも、ややキツい。ここはやはり「庶民の味方」Duronでいくしかないと決定。

ただ、ラボ1号から「Cyrix Ⅲはどうよ？」という提案があった。たしかに編集部ではLinuxでの動作も検証していないし、予算的にも見合いそうということでテストを兼ねてこの案を採用。DuronとCyrix Ⅲの2本立てでいくことにした。

マザーボードについては、別途カードを購入するより、ビデオやサウンドの機能を統合したチップセットを搭載したものがお得。DuronマシンではVIAのKM133、Cyrix Ⅲマシンなら Intelのi810/i815シリーズを搭載した機種が狙い目だろう。

エルミタージュ秋葉原	http://www.gdm.or.jp/
価格.com	http://kakaku.com/
AKIBA PC Hotline！	http://www.watch.impress.co.jp/akiba/
サハロフの秋葉原レポート	http://www2s.biglobe.ne.jp/~sakharov/

表1　秋葉原のPCパーツ価格動向を掲載しているWebサイト

想像以上に厳しい……

上記のポイントをもとに、そのほかのパーツも含めて、おおよその予算表を作成してみた（**表2**）。メモリは64Mバイトで我慢して、キーボードやマウス、ケースなど、かなり安めに見積ったつもりで、これである（SCSIは論外なのでドライブはもちろんATAPI接続）。このうえ、さらに約2000円の消費税がかかるのだ。うーむ。かなり厳しいな。ていうか、無理？

しかし、ここで挫折してしまっては特別対策班を結成した意味がないし、ページも埋まらない。なにより、痩せても枯れても、秋葉原といえば世界の電脳魔窟である。どんな掘り出し物が見つかるかもわからないのだ。

論より証拠（ちょっと違うぞ）、習うより慣れろ（全然違う！）、聞いて極楽見て地獄（それはヤだな）、とばかりに我々は現地調査に乗り出したのであった。

パーツ	見積り価格
マザーボード（KM133/i81x）	1万2000円
CPU（Duron/Cyrix III 600MHz）	6000円
メモリ（PC100 128Mバイト）	3000円
ハードディスク（10Gバイト）	1万円
CD-ROMドライブ	4000円
フロッピードライブ	1500円
LANカード（100BASE-TX）	1500円
キーボード＆マウス	1500円
ケース	4000円
合計	4万3500円

表2　4万円マシンのお見積り

秋葉原お買い物レポート
～その1・Cyrix IIIマシン編～

「今日はラーメン屋で何を食おうか。サシミあるかな……だったら黒牛でキューっと……」そうニヤニヤしながら考えていたとき、副編集長@お目付役が突然口を開いた。

「でね、買い物部隊がもう1人欲しいのよ。ほら、ケースとか大きいものあるでしょう？　つーことでよろしく」

そうだ、週末の深夜の打ち合わせ中だったのだ。どうやら私に話しているらしい。そういえば、4万円でLinuxマシンを作るとかいうトチ狂った企画の話をしていたな……。そんなの無理でしょう……と思って油断していたが、どうやら本気らしい。

「じゃ、私が土日で市場調査して、掘り出しものがあったらチェックしておきます。月曜日の1時に秋葉原で待ち合わせしましょう」と担当のにゃ～@編集部。ふだんはいい加減な男が、いつになく気合いが入ってやる気まんまんだ。なんとなく後光が差している……と思ったら、これは窓に反射した蛍光灯のせいだった。

「前回の『5万円Linuxマシン』企画のとき予算オーバーしたよね。今回はそういうことしないように。足出た分は自前……にするかもね。あっ、消費税込みだから間違えないように」とお目付役。顔は笑っているが目はマジだ。この人ならやりかねない。私は出費に備えラーメン屋を諦めてまっすぐ帰宅することにした。

写真1　CD-ROMドライブ
LEOPTICSの40倍速CD-ROMドライブ。リテールパッケージ品で、型番はCDD-ADHXらしい。パッケージの写真と実物はデザインが異なるが、製品のほうがグッドなデザイン。嬉しい誤算だ。3280円で購入。

アダルトショップ？ in 秋葉原

いよいよ今日は買い出しの日だ。秋葉原の狭い路地に車を滑り込ませる。いつも利用している100円コインパーキングはいっぱいのようだ。仕方がないので立体パーキングに車を乗り入れた。

早く着きすぎてしまったので、待ち合わせ場所そばの量販店を覗くことに。すると、なんと1万6000千円のベアボーンキットを発見。ケース、マザーボード、CD-ROMドライブ、キーボード、マウスが付いてこの値段。チップセットはi810EPなのでグラフィックスカードが必要だが、ハードディスクが1万円、メモリが4000円、CPUが9000円、グラフィックスカードが1万円だとしても残り2万3000円で揃うことに。間違いなく即買いでしょう。さい先いいぞ。

「すみません。これください」

「あ、それ、売り切れです」

かなりの脱力感が私を襲う。「売り切れ品展示してんじゃねーよ。なんなら展示現物を1万円でもいいぞ（調子いいナ）」とは思ったものの、量販店でこの値段ならもっと安い店もあるだろう。とりあえずメモを取って待ち合わせ場所に向かう。待つことしばし。しかし、いっこうににゃ～は現れない。そんなとき、ふと目の前の看板に目が止まった。「セーラー服、看護婦、体操服」と書かれてている。なんだ？　なぜこんなものが？　さらに「メンズファンシーグッズ」と書いてある。はて、メンズファンシーグッズとはどのようなものだろうか。ちょっと見てみるか。でも、店から出る姿を見られでもしたら大変だ。きっと「ふ～ん。そう（含み笑い）」とか冷たく言われるに違いない。そんな好奇心と自制心の間で葛藤を続けていると、突然にゃ～が現れた。危なかった。危うく転職しなければならないところだった。踏ん張った自分を褒めておこう。

「じゃー、行きますか」

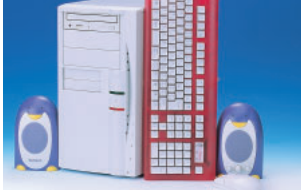
どうやらにゃ～は土日に市場調査すると言ったことなど、これっぽっちも覚えていないようだった……。

情報収集にかけろ！

まずは一緒に行動して相場を調べることに。別の量販店に行き、価格をチェック。4980円のケースを発見。かなり安いが、まだまだ掘り出し物がありそうだ。マザーボードは1万3000円台、ハードディスクは1万円前後、CPUも9000円前後だ。CD-ROMドライブは安いもので5000円台。これだけで4万2000円以上。予算オーバーである。だいたい、64Mバイトのメモリなんて売ってねぇ。先が思いやられるなぁ。

ここで見つけたものを最高値として、別行動して情報収集を図ることに。「決して安いからと思って買わないように。後悔するから」と言い残すとにゃ～は足早に去っていった。

重い足を引きずるようにして、隣の量販店へ。値段はあまり変わらない。メモリも64Mバイトは売っていなかった。気をとりなおして、すぐ前の中古ショップへ。このショップは新品も扱っているのだ。



ふむふむ。ケースが3980円。なかなか安いな。そのとき奇跡は起きた。

Socket370マザーボード5980円

おお。なんだ結構簡単に見つかるじゃないか。パッケージを見たところ、FICのマザーボードでチップセットはVIA Apollo Pro 133。UltraDMA/66にもちゃんと対応している。グラフィックスとサウンドはないけど、いいじゃんこれ。3980円のケースと合わせても1万円切ってるよ。浮かれる私に、さらなる奇跡が！

ケースを一緒に購入すると5000円

え？　我が目を疑った。一瞬中古品かとも思ったが、店員に聞くと新品に間違いないという。8980円でケースとマザーボードが我が手中に。「即買い」の衝動に駆られたが、にゃ～の言葉を思い出し、一応さらに安い店を探すことに。まだ、Cyrix IIIも見つけてないし。ま、オレの勝ちだけど（勝ち負けってあった？）一応見ておくか。CPUは5000円台で見つければOKだし、楽勝でしょう。勝ち誇った私は足取りも軽くCPUとメモリの探索に向かうのであった。

Cyrix IIIとメモリを探せ!

しかし、足取りは再び重くなっていった。やはり、64Mバイトのメモリはどこにも売っていないのだ。さらに、Cyrix IIIがまったく見つからない。そもそも、700MHz以下のCPUが売り切れなのだ。あったとしても、Celeron 600MHzはバルクで8000円に手が届く価格帯。ファンを買ったら9000円をオーバーしてしまう。諦めかけたそのとき、「Cyrix III」の文字を発見。しかし、売り切れのシールが。無理矢理シールを剥がして値段を見ると5000円台後半だ。これならなんとかなる。必ずCyrix IIIを見つけることを新たに決意。でも、本当にCyrix 置いている店ってないんだよなぁ。

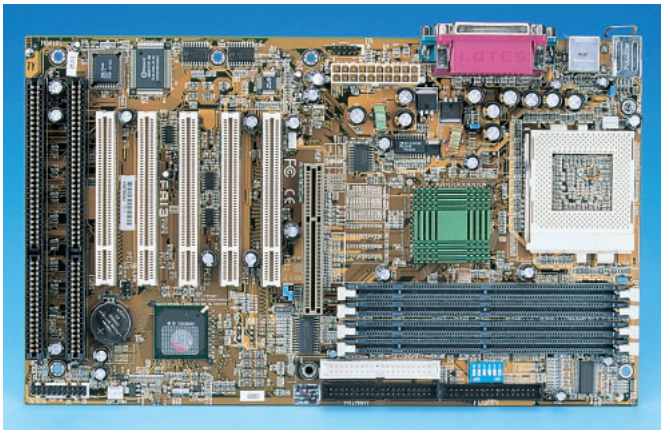


写真2　5000円のSocket370マザーボード（FIC FA13）
Pentium III 450～933MHz、Celeron 433～633MHzまで対応。もちろん、Cyrix IIIにも対応している。なぜかISAスロットが2本も。チップセットはVIA Apollo Pro 133。PC100/133 SDRAM、Ultra DMA/66対応。

ついでに安いマザーボードを探す。1万円を切るものは見つからない。ハードディスクも安いものはあるにはあるのだが、1.25Gバイトで5400円というようにバイト単価が異常に高い。

ここでにゃ～と合流し、食事をしながら情報交換をすることに。

にゃ～は280円のマウス、598円のスケルトンキーボードを見つけたという。それは買いだ！

しかし、どうやらDuron 600MHzのリテールパッケージが見つからないので何も買っていないようだ。CPUはバルクを購入してはどうかと提案をすると、ファンを買うならリテールパッケージのほうが安上がりだという。確かにそうかもしれない。

仕方がないのでCPUはあとにして、再びメモリを探すことに……。

最初のひと品は？

探索エリアを変えて、中央通りの反対側に移動。うろうろすること1時間。3280円の40倍速CD-ROMを見つけたので、この辺で手を打つことにしてこれをゲット（**写真1**）。

ここでにゃ～と再合流。やはりメモリとCPUはないらしい。にゃ～は本当に調査しているのか？　ドーナツ屋でコーヒーでも飲んで時間をつぶしているのではないか……という懐疑心が膨らむ。

エリアを変えようと歩き始めたそのとき、中央通り沿いの小さなショップで2280円の64Mバイトメモリを発見。PC100だが背に腹は代えられない。にゃ～と一緒に即ゲットした。

再び奇跡の店へ

時間も押し迫ってきたので、最初に見つけた5000円のマザーボードを購入することに。よく見ると980円のネットワークカードなんていうのも売っているではないか。

「あの3980円のケースと、このマザーボードください。一緒に買うとマザーボードは5000円なんですよね」

「そうですけど、このケース結構大きく見えますけど、マイクロATXなんでこのマザーボード入りませんよ」

絶句……。一瞬頭は真っ白に。ちゃんとマイクロATXって書いといてくれよ。え？　書いてある？　しまった。

ほかのケースは一番安いものが5980円だ。これでは値引してもらってもダメだ。しかし三度奇跡は起こったのだ！

980円相当のマウス･キーボード付属

なんと、5980円のケースを買うとおまけしてくれるらしい。壊れた思考回路で必死に考える。タダより安いものはない……。しかし、実はマウス or キーボードどちらか一方だけらしい。もういいや……「じゃ、5980円のケースをマウス付きで……マザーボードとネットワークカードも一緒に……」と、半ば投げやりてきな買い方をしてしまった（**写真3**、**4**）。しかも、よく見たらかなりアバンギャルドなカラーリングのマウスであった（**写真5**）。

大物を購入したので一度車に戻る。どうやらにゃ～もケースを購入した模様。ケース購入のいきさつを話すと、「普通キーボードでしょう。高いんだから」と冷たい声。今考えればその通りです。誰でも気づくようなミスに後悔するのであった。でも、壊れてたからね（言い訳）。

染料って値段違うの？

気を取り直して、にゃ～情報による598円のキーボードを購入することに。色はレッドとブルーの2色あるらしい。にゃ～はレッドを購入した模様。どうせだから色違いで、と私はブルーを選択（**写真6**）。レジにて支払い。

「消費税込みで732円です」

「え？　これ598円では？」

「あ、ブルーは698円なんです」

どうして色が違うだけで100円も値段が違うんだ！　しかし領収書も書かれてしまっているので仕方ない。またまた100円のムダ使いだ。

しかし、さらにこの店で格安ディスプレイカードを発見（**写真7**）。リテールパッケージでありながらなんと2980円。グラフィックスチップはTrident 3D Image975（3Dですよ！）。ただし、メモリは4Mバイトと少なめ。まぁ、いいでしょう。こんなに安いの見たことないし、XFree86もばっちりのハズだし……。

気を良くしたところでハードディスクも購入することに。ハードディスクの値段はどこも同じようなものだったので、8800円のディスクにする。10.2Gバイトで4500回転、UltraDMA/100対応、Quantum lct2010のバルク品だ（**写真8**）。

再びCPUを探しに……

三度にゃ～と合流してCPU探し。結局リテールパッケージのDuronは見つ

写真3　それなりの作りのATXケース
マザーボードとマウスを安く入手するために購入。意外としっかりしている。今思えばキーボードを選択すべきだった……。

写真6　スケルトンがいい感じのキーボード
TeknosのTN-9850IW(S)という製品らしい。どうして色によって値段が違うのかは不明。よけいなキーがいっぱい付いている。使用感はあまりよくない。

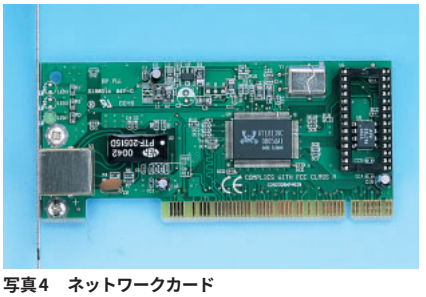


写真4　ネットワークカード
リテールパッケージ品だが、メーカーは不明。添付のマニュアルにはLinuxサポートと記載されている。チップはRealtekのRTL8139C。おなじみのカニマークチップ搭載品だ。

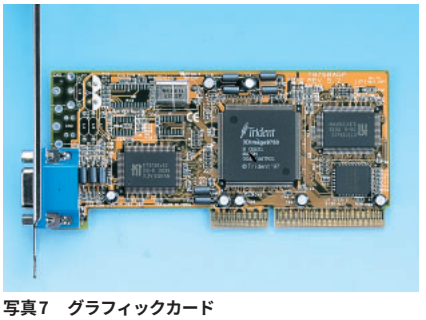


写真7　グラフィックカード
これもメーカー不明のリテールパッケージ品。詳細は不明だが、チップはTrident 3D Image975でVRAMは4Mバイト。3Dといったって、4Mバイトじゃねぇ。でも2980円と格安。

写真5　タダでゲットしたマウス
かなりインパクトのある配色だ。ホイールも付いて、見た目の割に使いやすい。製品名は「My baby mini Mouse」らしい。子供用とも書かれていた（笑）。ベビーとかミニとか謳っているが、そんなにミニサイズではない。

写真8　Quantum Fireball lct2010
容量は10.2Gバイト。シークタイムは12ms、キャッシュは128Kバイト。Quantumのハードディスク部門はMaxtorに買収されたので、MaxtorのWebサイトにあるデータシートを参照したところ回転数7200rpmとなっているが、4500rpmだと思われる。UltraDMA/100対応製品。

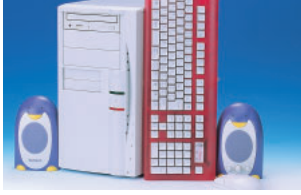


写真9　VIA CyrixIIIプロセッサ
リテールパッケージにはイカすファンが付いている。かなり小さめのファンなので、発熱量が少ないと予想される。セラミックパッケージが高級感を漂わせている。

写真10　600円でゲットしたフロッピードライブ
フロッピードライブのフタがついていないので、ケース装着時にかなり格好悪い。ホコリが溜まるという情報も。どうやら、コンパックのサービス部品らしい。バルク品。

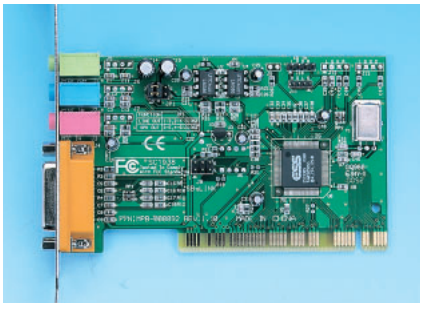


写真11　サウンドカード
リテールパッケージだが、これまたメーカー不明のサウンドカード。製品名は「Golden Melody Hi-Live PCI SOUND CARD」というらしい。チップはESS TchnologiesのSolo-1（ES1938）。

からなかったようだ。諦めてバルク品を購入したという。Cyrix IIIのバルク品を見かけたという店に行く途中、老舗量販店に寄ってみる。「リテールパッケージはこういう店のほうがあるかもね」などと話しながら最上階へ。すると、なんとCyrix IIIのリテールパッケージを発見。価格は5800円。もう買うっきゃないでしょう（**写真9**）。

さらに別の店に移動。7700円のDVD-ROMドライブを発見。にゃ〜はCD-ROMドライブをまだ購入していないらしいので、先ほど購入したCD-ROMドライブを譲ってDVD-ROMドライブを購入したいと提案したところ、「やめたほうがいい」と一蹴された。のちに、この一喝がなければ大変なことになっていたことが発覚するのであった……。

そうこうしているうちに、どうやら目的のものを購入したらしい。今度は1000円を切るCPUファンを漁っている。おかげでここでタイムアップ。サウンドカードが宿題となってしまった。会社に戻って成果を確認することに。

フロッピー欲しいよ

会社に戻って戦利品の確認と情報交換。翌日再び出撃するというにゃ〜に、残りの予算でサウンドカードとフロッピードライブをゲットしてくるように依頼。サウンドカードは1500円ぐらいで発見していたので問題なさそうだが、フロッピードライブを700円以下で購入するというかなりのインポッシブルミッションである。「フロッピーなんていらないでしょう」というにゃ〜に、ダダをこねる私。

翌日、フロッピードライブをゲットして見事任務を遂行してはくれたものの（なんと600円！）、あまりのショボさに愕然とする私であった……。

戦利品一覧

途中でムダ使いはあったが、無事予算内で購入することができた。全パーツの購入価格は合計3万7778円。消費税を入れても3万9667円。残金は333円だ。購入したパーツの一覧は**表3**のとおり。

しかし、予算内で購入できて何よりだ（フロッピードライブがショボいのだけが悲しいところだ）。なんとか自腹は避けられた。あのときDVD-ROMドライブを購入しなくて本当によかった。忠告ありがとう（買ってきたフロッピードライブはショボかったが）。

今晩は生ビールと海老の塩焼きで乾杯することにしよう。マシンの組み立てもそこそこに、早々に編集部をあとにするのであった……。

写真12　今回のパーツで組み立てた完成品
フロッピードライブを除けば、まずまずの外観。CyrixIIIのロゴもシブい。とても4万円を切っているとは（フロッピードライブを除けば）思えない。あぁ、くどいようだがフロッピードライブが……。

パーツ	購入価格
CyrixIII	5800円
マザーボード	5000円
メモリ	2280円
ケース	5980円
ハードディスクドライブ	8800円
CD-ROM	3280円
グラフィックスカード	2980円
NIC	980円
サウンドカード	1380円
キーボード	698円
マウス	0円
フロッピードライブ	600円
小計	3万7778円
消費税額	1889円
税込合計金額	3万9667円

表3　戦利品一覧（CyrixIIIマシン）

秋葉原お買い物レポート
～その2・Duronマシン編～

作戦会議で突きつけられた厳しい現実も記憶に生々しい4月のとある日、Cyrix IIIマシン担当である王大人（ワン・ターレン）と待ち合わせ、秋葉原へとくり出した。いよいよ、現地調査の開始である。

しかしながら、いきなり王大人から出鼻をくじく報告が！　i810EPチップセットを搭載した格安ベアボーンキットを発見したが、すでに売り切れだったというのだ。でも考えてみれば、こちらはDuron担当なのでカンケーないといえば関係ないのだな（＜そういう態度ではいけません）。

マザボがない！

まずは大物（マザーボード、CPU、ケースなど）を捜しながら、そのほかのパーツについても価格を調査していくことにする。

とりあえず大手PCパーツショップを4店舗ほど見てまわるが、ターゲットであるKM133を搭載したマザーボードは少ない。あっても、1万4000円～1万6000円と、見積りより若干高めの価格帯が中心だ。

しかたがないので、延々とショップ巡りを続ける。電気街をほぼ1周しても、1万3000円を切るKM133マザーは見つからなかった。ただし、ほかのパーツは収穫あり。MITSUMI製の32倍速CD-ROMドライブ（**写真13**）が3280円で発見されたほか、598円のキーボード（**写真14**）や280円のマウス、3880円のケース（**写真15**）、980円のLANカード（**写真16**）など、格安な「ブツ」がちらほら見受けられた。これらをチェックしつつ、秋葉原巡回は2周目へと突入していくのであった。

先ほど立ち寄らなかった店を中心に、（2度目の店も再確認しながら）ウロウロする。やはりKM133マザーは少ない。Duron向けとしては、KT133を搭載したものが最も多く出回っているようだ。KT133はグラフィック統合チップセットではないので、別途ビデオカードが必要になる。それでも、9000円くらいのものがあれば、王さんから報告のあった2980円のビデオカード（Cyrix III編を参照）と組み合わせて、なんとか予算内に収まる……が、やはり1万円を切る製品は皆無であった。機能ごとにバラつきがあるものの、1万3000円～1万7000円が中心的な価格帯だ。

写真13　MITSUMI製CD-ROMドライブ（32倍速）
メーカー製ながらバルクで売られていた。Cyrix IIIマシンのものと同価格だが、あちらは40倍速。8倍速分の負けである。

写真14　ナゾの格安キーボード
一応、メーカーと型番はTeknos TN-9850（R）となっている。「R」は「Red」か？　ブルーバージョンより100円お得。

それでもしぶとく調査を続けたところ、1万1980円のKM133マザー（**写真17**）を発見！　即ゲットといきたいところだが、ここは慎重に、全体的な目途が立ってから購入することにした。何にせよ、とりあえずマザーが見つかってよかった。

CPUもない!!

お次はCPU。これまたDuronは品薄で、リテールパッケージのものは800MHz以上でないと売られていない。ちなみにお値段は、800MHzと850MHzがほぼ同じで1万円弱、900MHzが1万5000円ほど。

バルク品も800～900MHzがほとんどで、まれに700MHzがある程度。お目当ての600MHzはほとんど店頭にない状況である。「品切れ」とか「入荷待ち」とかの張り紙をペロリンとめくっ

写真15　外見上は特に問題のないケース
店頭で露天積みされているところを捕獲した。これが実はクセ者で、のちに苦労するハメに……（詳細は80ページ）。

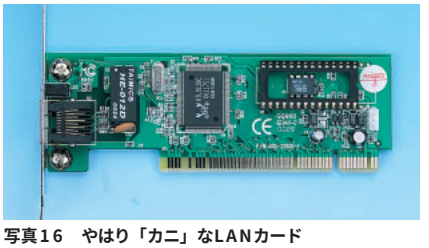
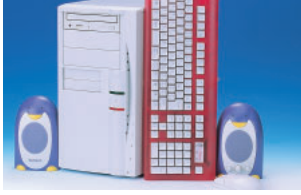


写真16　やはり「カニ」なLANカード
搭載チップは低価格のシンボルであるRealtek 8139（C）。



て確認したところ、600MHzのバルクは6000円前後で売られていた（過去形）ようである。価格は見積りどおりなので、なんとかモノを見つけなくては。

この時点で秋葉原2周回を達成し疲れきった身体に鞭打ちながら、これまでの巡回コースから外れていたショップに足を向けてみる。このショップには、2週間ほど前に訪れたときCyrix IIIが大量入荷していた。途中の情報交換では、あちらもCPU探しに苦戦しているようなので、Cyrix IIIの在庫状況も合わせて確認しておこうと思ったのだ（やさしいなぁ＜自分）。

あまり期待していなかったのだが、ここでDuron 600MHzのバルク（**写真18**）が売られていた。価格は5980円（税抜）。やはり人間、良い心がけを持つことが肝要である。なお残念ながらCyrix IIIは売り切れていた。おそらく心がけの差であろう。

メモリもまた……

メモリについては、事前の調査で、PC100の64Mバイトが3千円前後、同じく128Mバイトが5000円弱であった。今回は冒頭でも触れたように、予算の関係上、64Mバイトしか買えない。なのに、これまたモノがないのである。いまやメモリは128Mバイト単位なのだろうか！？　それでも、いくつかのショップでは扱っていて、最も安かったところで2280円だった。

これで一応の目途がついた。この時点での調査結果をまとめると、こんな感じだ。

●マザーボード：1万1980円
●CPU：5980円
●メモリ：2280円
●CD-ROMドライブ：3280円
●LANカード：980円
●キーボード：598円
●マウス：280円
●ケース：3880円

このうち、どの店で売っていたのか忘れてしまったマウス（メモぐらいはとりましょう）を除いて購入。小計は2万8978円。消費税を含めると、残金は9573円也である。購入のためにもう1周回して、へとへとになったので、この日の買い物はここまでとし、翌日改めて出直すことにした。

再び秋葉原へ

で、翌日。この日のターゲットは、マウスにCPUクーラー、最後の大物・ハードディスク、それと大人から頼まれたサウンドカードとフロッピーディスクドライブ（FDD）である。残金を考えると、なるべく安価なものを購入する必要がある。特にCyrix IIIマシンのほうは2200円ほどしか残ってないらしく、昨日見かけた1500円のサウンドカードを買ったとしても、FDDは700円で購入しなければならない（FDDなんて要らないのになぁ）。

「通常の」パーツショップにおけるFDDの相場は1980円である。安くても1580円とか1480円だ。これはもう「ジャンク屋」しかない。しばらくブラブラするうち、小さな路地の入り口付近にあやしげな立て看板を発見。「セール価格」がズラズラと書かれている。どうやらドライブ類が安いらしいので、立ち寄ってみることにした。

店の広さは3坪くらい。スチール製の棚にFDDやCD-ROMドライブが無造作に並べられている。その中に「新品600円」と値札の付いたFDDが！　フロントベゼルはないものの（たぶん）新品で、これ以上の掘り出し物があるとも思えないため即購入。「あー、いい買い物したなぁ」と気分を良くして、サウンドカード探しに移ったのであった。

安いものは安いのだ

マウスと同様に見つけた店を忘れたので、グルグルとショップ巡り。2000

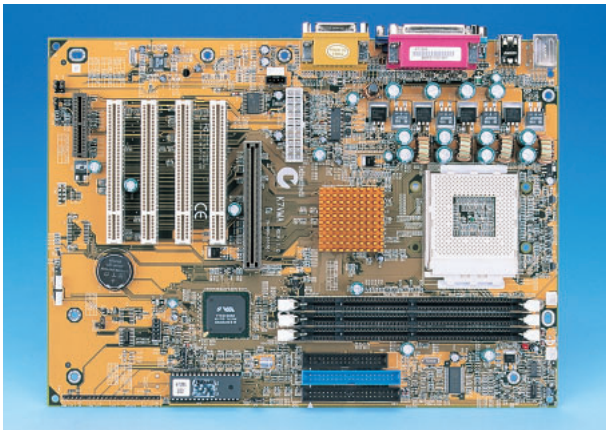


写真17　ELITE K7VMA
苦労の末に発掘したKM133搭載マザーボード。サウスブリッジはVIAのVT82C686Bで、オンボードサウンド機能、UltraDMA/100をサポートする。AGPスロット（x4）を備えており、オンボードグラフィックだけでなく外部ビデオカードにも対応。

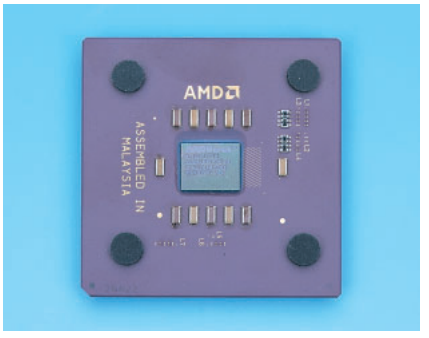


写真18　Duron 600MHz
価格対性能比ではほかのCPUを寄せ付けないDuron。特に自作派チープ族に大人気だ。そのためかどうか常に品薄状態である。

写真19　200円のマウス
値段だけで冗談のような存在だが、クリック感も奇妙である。文字で表現すると「カッツン、カッツン」という感じ。

写真20　Cool Masterシリーズの下っ端たち
Cool Masterといえば、CPUクーラでは老舗の部類。信用してもいい（のか？）。左が980円のDP5-6G11、右が880円のDP5-5H51。

写真21　絶滅寸前の「小」容量HDD
奇跡的に保護された富士通の3.2Gバイトハードディスク（MPC3032AT）。製造年月は意外に新しく2000年1月とプリントされていた。

円以下のサウンドカードは結構あって、3、4店舗めで1380円のものを発見できた。これを購入する。

「うーむ、サウンドカードも安くなったもんだ」などと、再び気を良くしていたところ、とあるショップの店先で200円のマウス（**写真19**）がカゴの中に転がっていった。しかも、Xで使ってくれとばかりに3つボタンではないか。もちろん「買い」である。

続いてCPUクーラー。これにはちょっと苦戦。Socket370用のものは1000円前後で見つかるのだが、Socket A用は安価なものがないのである。やはり、AthlonとDuronはしっかりした熱対策が必要ということなのだろうか？

結局、880円と980円の「Socket370／Socket A兼用」と張り紙に書かれたCoolMasterシリーズ（**写真20**）が見つかったのだが、冷却性能が不安だったので、店員さんに確認してみることにした。「メーカー発表のスペックでは、980円のほうは1.2GHzのAthlonでも大丈夫ということになってます。Duronなら、どちらもOKなようです」とのこと。微妙な解答である。同じソケット用とは思えないほど大きさが違うこともあり、念のため2つとも購入しておく。

小容量ハードディスクはいずこに

最後にハードディスク。ここでひとつ残金を確認しておこう。CPUクーラは一応「Athlon Ready」のほうを選択するとして、税込みで「9537－（210＋1029）＝8334円」となっている。

ただでさえ結構きついのだが、探索中に発見したペンギン型スピーカが妙に気に入っていた私は「ハードディスクは4.3Gバイトクラスのもので（予算的に）なんとかなるはず」と思い切り、大枚1869円（1780円＋税）をはたいてペンギン君を購入してしまった。実際には、税込6465円のハードディスクを探す必要が生じたわけで……。

これが実に大失敗だった。10Gバイト以下のハードディスクなど、どこにも売ってないのだ。この日も「秋葉原周遊の旅」を敢行するはめに陥りながら、発見できたのは2つだけ。しかも、4.3Gバイトで7800円、8.4Gバイトで9800円（ホワ～イ？）と、いずれも予算オーバー。このとき、実は「明日、新宿と池袋を攻めてみよう」と決意していた。しかし、2日間で推定15キロを踏破し疲れきって帰社した私を神は見捨てていなかった。

別件で秋葉原へ買出しにいっていたLinux magazineラボ・1号が3.2Gバイト5600円のハードディスク（**写真21**）を買ってきてくれたのだ。ありがとう、ラボ・1号！（それにしても、どこで見つけてきたんだろう？）

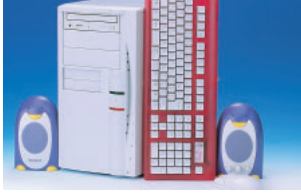
というわけで、何とか予算クリアである。最終的な明細は**表4**のとおり。ふぅ、疲れた疲れた（リフレイン）。

写真22　完成したDuronマシン
無理して買ったペンギン型スピーカを従えた雄姿を見よ。ちなみにスピーカはオウルテック製で、最大出力3W＋3W。この兄弟バージョンのほかに、ウーハーが「お母さん（？）」の親子バージョンもあり。

パーツ	購入金額
Duron 600MHz	5980円
マザーボード	1万1980円
メモリ	2280円
ハードディスク	5600円
CD-ROMドライブ	3280円
LANカード	980円
キーボード	598円
マウス	200円
ケース	3880円
CPUクーラ	980円
スピーカ	1780円
小計	3万7538円
消費税	1877円
税込合計金額	3万9415円

表4　戦利品一覧（Duronマシン）

超実践的
電脳お買い物ガイド for Linuxers



Linuxで使える格安PCパーツ情報

この特集のために秋葉原じゅうを調査した4万円Linuxマシン特別対策班が、その道すがら発見したお買い得パーツや現在の価格動向を紹介する（価格はいずれも税抜の実売価格）。

ディスプレイ

「4万円でPCを！」と言いながら、ディスプレイを無視していることは公然の秘密である。「ディスプレイはPC本体ではなく周辺機器である」と言い逃れることもできよう。それはさておき、ディスプレイの価格について。

PCに必須のパーツの中では、群を抜いて高額なのがディスプレイである。それでも、ひと昔前に比べれば、ずいぶんと安くなっている。17インチモデルでも2万5000円〜3万5000円がおおよその相場である。

人気が高いのは、コストパフォーマンスに優れた三菱電機のRDF171S（2万7000円前後）と、少し価格は高くなるが性能に定評のあるEIZOのFlexScan T561（3万5000円前後）。安さで勝負なら、ADIのMicroscan M700（2万円前後）がお勧めだ。

15インチモデルなら、1万5000円を切る製品も見かけた。ただし、現在は17インチが主流となっているようで、15インチモデルは店頭に並んでいる製品が少なくなってきてるように感じられた。

省スペースということで人気の高い液晶ディスプレイは、最安値でも14インチモデルで4万5000円程度。格安パーツとしてお勧めするには、まだまだ無理があるようだ。

IDE接続ドライブ

いまやハードディスクは「オーバー10Gバイト」が常識。10Gバイトクラスでは、本文中に登場したQuantum Fireball lct20が最も実売価格の低い製品だろう。ネット上の情報では、7900円で販売しているショップもあるようだ。ハードディスクは容量や性能、メーカーによる価格差が小さいので、あと数千円プラスするだけで20〜30Gバイトクラスが確実に手に入る。予算に余裕があるなら、そのほうがお買い得である。

いくつか製品を紹介しておくと、「騒がしくない」、「熱くならない」、「壊れない」の三拍子が揃って自作派のユーザーから圧倒的支持を得ているIBMのDeskstarシリーズ、高速大容量化競争の良きライバルMaxtorのDiamondMax Plusシリーズと SeagateのBarracuda ATAシリーズなどが定番。少しでも価格にこだわりたいなら、隠れた「価格破壊王」富士通のMPG3xxxATシリーズを探してみよう。

CD-ROMドライブは、4000円を切る製品が多く出回っている。24倍速以上であれば、体感できるほどの性能差はないと思うので、スペックに惑わされないようにしたい。また、1万円を切るCD-R/RWやDVD-ROMドライブが調査中に発見されたことをお知らせしておく。

キーボード＆マウス

キーボートとマウスは製品による価格の差が非常に激しいパーツだ。本文中では、まさに衝撃的価格の製品を紹介したが、高いものは8000円〜9000円という価格帯で売られている。実に10倍以上の価格差である。安いものはいくらでもあるが、最も人に近いパーツなので、実際にキーやボタンのタッチを確かめて、気に入ったものを購入するようにしたい。

今回の調査中に発見したお勧め製品が写真のトラックボールタイプのポインティングデバイス「使えてマウス！」である（販売元は創朋）。購入価格は2270円。通常のPS2マウスとして認識され、Linuxでも問題なく動作した。USBモデルもある（こちらは動作確認していない）。

ビデオカード

購入時の注意点はLinux（XFree86）でのサポート状況を把握しておく必要があるということ。ただ、安売りされている製品は、2、3世代前のビデオチップを搭載しているものが多いので、かえってサポートされている確率は高い。それでも、購入する前にサポートの有無は必ず確認するようにしておきたい。

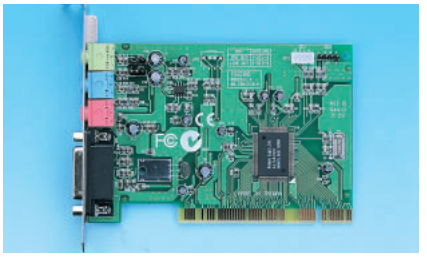
3Dグラフィックスのアクセラレーション機能が必要で、価格とともにある程度の性能を求めたいなら、NVIDIAのRIVA TNT2シリーズを搭載したものがお勧め。シリーズ中のローエンドモデルであるVANTAを搭載し、32MバイトのSDR VRAMを備えたタイプが7000円を切る価格で販売されている。なお、3Dアクセラレーションを有効にするには、バージョン4.0.1以降のXFree86とNVIDIAが配布しているLinux用ドライバ（http://www.nvidia.com/Products/Drivers.nsf/Linux.html）が必要になる。

3D処理が必要なければ、MatroxのG200搭載カードを探してみよう。6500円前後なら「買い」だ。また、グラフィックス機能統合タイプのマザーボードも選択肢のひとつ。本文中で紹介したVIAのProSavage KM133のほか、Intelのi810／i815シリーズ、SiSのSiS530／620がある。

サウンドカード

低価格な製品は、YAMAHAのYMF744／754、C-Media Electronicsの CMI8338／ 8738、 ESS TechnologyのSolo-1／Maestro-1などを搭載したカードが多く、これらはLinuxでの動作実績もある。調査中に目についたのがAOpenのAW200という製品（写真、価格は1580円）。搭載されているサウンドチップは、Avance LogicのALS4000で、これはALSA 0.5.10以降でサポートされている。

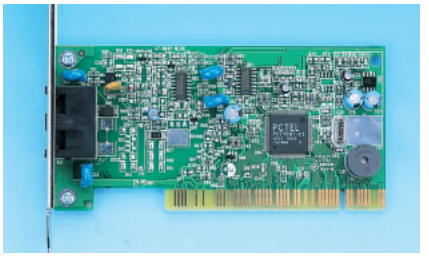
ビデオカードと同様に、チップセットに統合されたサウンド機能を利用するのもひとつの手だ。Intelのi810／i815シリーズ、サウスブリッジにVIAのVT82C686A/Bを採用したものがこれにあたる。また、上記のサウンドチップを搭載したマザーボードも狙い目。



モデム

本文中では触れていないモデムについても紹介しておこう。外付けのアナログモデムはあまり店頭で見かけなかった。Linuxではソフトウェアモデムの利用に制限があるので、できるだけシリアル接続の外付けタイプを購入したいところだが、ほとんど売られていないのが現状だ。探し出せたとしても5000円〜7000円と割高感がある。

PCIバスに装着する内蔵型は、ソフトウェアモデムが多く注意が必要である。そんな中から、「Linux Ready」を謳う製品を見つけた。ラトックシステムのREX-PCI56（写真）という製品だ。同梱されているフロッピーにLinux用ドライバのソースが含まれており、マニュアルにはインストール方法も解説されている。購入価格は2979円。



ホントにLinuxで動いたの？
～マシンの組み立てとLinuxのセットアップ～

さて、Cyrix IIIマシン、Duronマシンとも、めでたく4万円という予算をクリアしてパーツを揃えることができた。次なる関門は、はたしてLinuxでみごと動作するのだろうか？　ということ。ここでは、組み立てから、インストール、Xやサウンドの設定までの顛末をご紹介しよう。

ジャンク扱いデス！

格安パーツを求めて秋葉原をさまよった際も、メモリがない、CPUがない、ハードディスクがない、という「ないない尽くし」だったが、組み立ての段階でも「金具がない」という事態にみまわれた。

何の金具かというと、ケースの筐体にマザーボードを取り付けるためのもの。通常はPCケースに部品として同梱されている「スペーサ」というヤツである。Duronマシン用に購入したケースには、これが付属していなかったのだ。スペーサにはいろいろなタイプがあって、中には別のケースのものを流用できる場合がある。問題のケースは、筐体の取り付け部分の形状から推測して、**写真1**に示すスペーサが必要であるようだ。

編集部内で聞き込みを行ったところ、王大人（ワン・ターレン）が1つだけ所有していることが判明。試しに装着してみると、ピタリとはまる。しかし、1つだけではマザーボードを固定できない。少なくとも4カ所くらいは留めておくほうがよいだろう。ほかの編集部を訪ねてまわったが、収穫はゼロ。しかたがないので、翌日、購入したショップへと向かった。

虎の子の1個を見せながら状況を説明すると、「あのケースはジャンク品扱いなんデス」とやけにキッパリした口調で言われてしまった。ようはサポート対象外ということだ。困った。

対応してくれた店員さんは眼鏡をかけた好青年で、こちらの深い落胆を察したのか、支払いカウンターの奥から各種スペーサがぎっしりと詰まったビニール袋を取り出してきて、「これですよね」と言いながら6個のスペーサを恵んでくれた。ありがとう！　眼鏡の店員さん。

Cyrix IIIマシン用のケースは、さすがに2000円も高いだけあって（？）特に問題はなし（**写真2**）。Duronマシンのほうも、なんとか無事に組み上げることができた（**写真3**）。

写真1　恵んでいただいたスペーサ
割と珍しいタイプだと思うが、王大人によると、最近よく見かけるようになったそうだ。

サクサクといきたかったのにぃ

続いて、Linuxのインストールである。特集冒頭の作戦会議のポイントにあげたように、今回の目標は「激安マシンであっても、サクサクっとLinuxがインストールできて、XFree86のグラフィック環境で快適に使えるクライアントマシン」を作ることにあった。予算内に収めるために、格安パーツを寄せ集めた感のある2台のマシン。はたして、サクサクっといくのだろうか？

なお、以降この特集では、完成した2台のマシンをそれぞれ「Duron号」、

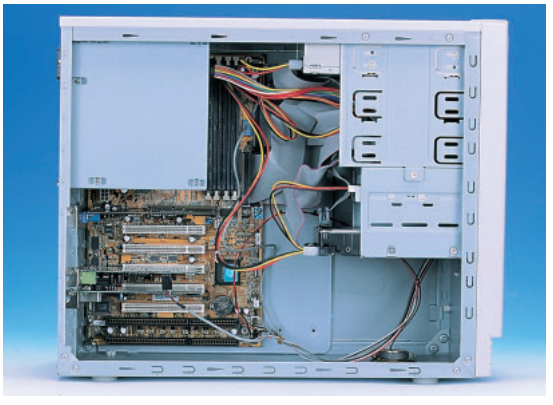
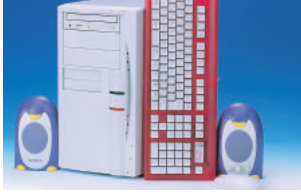


写真2　完成した「Cyrix号」の横顔
ビデオカードの装着にコンデンサが邪魔なり、無理やりコンデンサを寝かせている。ケーブル類の取り回しは、美しいほう？

写真3　ペンギンと別れて少し寂しげな「Duron号」
本文では触れなかったが、CD-ROMドライブのオーディオケーブルも同梱されていなかったのだ。こちらは、編集部にころがっていたものを流用している。



「Cyrix号」と呼ぶことにする。これは、搭載されたCPUによって「～マシン」と呼ぶ、一般的な呼称との混同を避けるためだ。

ここで、2つのマシンのスペックを再確認しておこう。**表1**を見てほしい。この中でインストール時に問題となりそうなのは、Duron号のチップセットKM133に統合されているグラフィックス機能がインストーラに正しく認識されるかという点と、Cyrix号の心臓であるCyrix IIIそのものがLinuxでちゃんと動くのか（インストーラが起動するのか）という点だ。

インストールテストは、カーネルとXFree86のバージョンがそれぞれ異なる4つのディストリビューション（LASER5 Linux 6.4、Vine Linux 2.1／2.1.5、Meister Linux Mandrake 7.2）で行った。チェック項目は以下の4つ。

①Anacondaの起動
グラフィカルインストーラ（Anaconda。Mandrakeは独自インストーラ）が起動し、正常にインストール処理を行うことができるか？

②VGAの自動検出
インストーラがグラフィックチップを正しく認識（自動検出）し、Xの設定を行うことができるか？

③NICの自動設定
インストーラがLANカード（NIC）を正しく認識し、適切な設定が自動的に行われるか？

④サウンドの自動設定
インストーラがサウンドカードを正しく認識し、適切な設定が自動的に行われるか？

結果は**表2**のようになった。

Duron号から詳細を確認していこう。表2からもわかるように、グラフィカルインストーラ自体の動作に問題はない。しかしXFree86の設定では、どのディストリビューションもKM133のグラフィックス機能を認識できなかった。実はこれ、予測された事態なのである。

	Cyrix号	Duron号
CPU	VIA Cyrix III 600MHz	AMD Duron 600MHz
マザーボード	FIC FA13	ECS（ELITE）K7VMA
チップセット	VIA Apollo Pro133	VIA ProSavage KM133
メモリ	64Mバイト SDRAM（PC100 CL2）	64Mバイト SDRAM（PC100 CL2）
ハードディスク	10Gバイト（4500rpm、12ms、UltraDMA/100）	3.2Gバイト（5400rpm、10ms、UltraDMA/33）
CD-ROMドライブ	40倍速	32倍速
グラフィックス	Trident 3DImage975	チップセット統合
サウンド	ESS Solo-1	チップセット統合
ネットワーク	Realtek RTL8139C	Realtek RTL8139C

表1　Cyrix号とDuron号の主なスペック

ディストリビューション（カーネル／XFree86）	Cyrix号 ①	②	③	④	Duron号 ①	②	③	④
LASER5 Linux 6.4（2.2.16／3.3.6）	○	○	○	×	○	×	○	×
Vine Linux 2.1（2.2.17／3.3.6）	○	○	○	○	○	×	○	○
Meister Linux Mandrake 7.2（2.2.17／4.0.1）	○	○	○	○	○	×	○	○
Vine Linux 2.1.5（2.2.18／3.3.6）	×	—	—	—	○	×	○	○

表2　インストールテストの結果
※ ①＝Anacondaの起動　②＝VGAの自動認識　③＝NICの自動設定　④＝サウンドの自動設定

KM133のグラフィックス機能は、S3のSavage2000の2DエンジンとSavage4の3Dエンジンと同等であるとされている。Savage2000とSavage4はともに、XFree86 3.3.6でサポートされているが、KM133は4.0.2からのサポートであることが事前の調査で判明していたのだ。KM133のグラフィックス機能でXを動作させるには、XFree86 4.0.2以降が必要になる。どうやら、Duron号はサクサクといきそうにない。このことについては、のちほど改めて検討することにしよう。

Cyrix号のほうは、Vine Linux 2.1.5以外では特に問題は発生していない。インストーラによるビデオチップの認識も、Trident 3D Image975という、いい具合に枯れたチップ（XFree86 3.3.6／4.0.Xともサポート）を搭載したカード使っているため問題はなし。

Vine 2.1.5でインストーラが起動しなかったのは、カーネル2.2.18固有の問題であることが、その後の調査で判明した。Cyrix IIIを使っているシステムでは、ブート時にカーネルパニックを起こすことがあるようだ（テストでも実際にカーネルパニックが発生）。

熱烈なVineファンは、Cyrix IIIを選択肢からはずしたほうが無難である。なお、この不具合は、2.2.19では解決されている。また、今回のテスト結果からもわかるように、2.2.18以前の2.2系カーネルでは、この問題は発生しないと思われる。

Wolverineでいこう!

Cyrix号はカーネルのバージョンに気をつけさえすれば、スムーズにインス

Column

カニは偉大なり

本誌のバックナンバーを丹念にチェックしていくと、LANカードの話題になるたびに「カニ」の2文字が登場していることに気づくはずだ。これはなにも、生来の蟹好きであるとか、Realtekからリベートをもらっているとかいう生臭い（前者は特に指先が）理由ではなく、編集部の「カニ密度」が高いことによるものである。ふだんの仕事で使っているPCやLinuxのテストに利用しているPCの多くにRTL8139xを搭載したLANカードが装備されているのだ。

つまり、日常的にLinuxでの動作を確認しているようなものなのである。これまで、インストール時に自動認識されなかったことはないし、ハードの故障以外で挙動がおかしくなることもなかった。ただし、編集部ではクライアントマシンでのみ使っているので、高負荷な状態が連続的に発生するような、タフなネットワーク環境での信頼性といった面は未確認であることはお断りしておく。

ちなみに、編集部で最初に「カニ」と呼び始めたLinux magazine・ラボ1号は、自宅のLAN環境でもRTL8139Bを使っているそうだ。確認したところ、これまで何の問題もないとのこと。

個人のセカンドマシンやオフィスのクライアンマシンなど、コストを抑えたいときには、やはり「カニ」がお勧めのようである。

トールできることがわかった。残るは、Duron号の「サクサク感」をどうやって醸成するかである。それには、標準でXFree86 4.0.2を採用したディストリビューションが必要なことは言うまでもなかろう。

執筆時点では、日本語ディストリビューションで該当するものはなかった。日本語対応は完全ではないものの、かろうじてSuSE Linux 7.1が選択肢としてあげられるくらいだ。ただ、SuSE Linux自体がフルインストールで4Gバイトを超えるディスクスペースを占有する巨大ディストリビューションであるため、サクサク感が薄い気がするし、今回に限って言えば、Duron号にはフルインストールさえできないのであった。なにより、日本語処理はビシっと決めたいところだ。

そこで目を付けたのが「Wolverine」である。Wolverineとは、Red Hat Linux 7.1の最新ベータバージョンのコードネーム（バージョン表記は7.0.91となっている）。Red Hat Linuxは7.1から、多言語環境を1つのバイナリでサポートする「シングルバイナリ仕様」になる予定である。つまりは、日本語処理もイケてるはず。

さっそく試してみたところ、グラフィカルインストーラはもちろん無事起動、順調に処理が進んでいった。そして問題のXの設定。残念ながら、KM133のグラフィックス機能は認識されなかった。しかし、4.0.2でサポートされていることは、本家であるXFree86 ProjectのWebサイトで確認済みであり、また、S3 Savageファミリーは共通のドライバで動作することもわかっている（http://www.xfree86.org/4.0.2/savage.4.html）。ここは慌てず騒がずビデオカードの一覧から「S3 Savage（generic）」を選択する。

インストール完了後、再起動してWolverineが快調に立ち上がる。「startx」すると、メッセージがコンソールを流れていく。緊張しながら見守る中、Xが無事起動した！　よかったー！

ベータ版とはいえ、これでDuron号も一応「サクサク・インストール状態」となることができた。当然、Red Hat 7.1でも動作するはずなので、KM133マザー＋Duron（Athlon）という組み合わせが、Linuxユーザーにとってお手軽な選択肢となる日も近いわけだ。

サウンドの設定

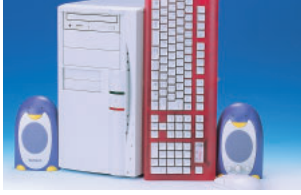
今回、作成するマシンのもうひとつの目標にしていたのがサウンドの再生だ。Cyrix号の場合、厳しい予算の中でサウンドカードを購入した。一方Duron号のほうは、ほとんど衝動買いとはいえ、スピーカまで購入したのである。せっかくなので、インストーラで自動設定できなかったディストリビューションでも、きちんと設定してサウンド再生ができるようにしよう。

Cyrix号のサウンドカードは、いわゆる「安全パイ」というやつで、編集部において過去に何度か動作確認しているサウンドチップ「ESS Solo-1」搭載機種である（ES1938と型番で呼ばれることもある）。Duron号は、グラフィックスと同様にサウンドもオンボードである。チップセットのサウスブリッジ「VIA VT82C686B」にサウンド機能が統合されているのだ。

どちらのサウンドチップ（機能）も、カーネルに付属するサウンドドライバOSS/Freeと、Linux用の代表的なサウンドドライバであるALSAの両方でサポートされている。

ここでは、ALSAでの設定を紹介しておこう。説明は、ALSAを標準でサポートしているLASER5 Linux 6.4をベースにしているが、基本的な操作はどのバージョンのALSAでも変わらない。付属のドキュメントとあわせて参照しながら、操作していけば設定できるはずだ。

ALSAのパッケージには、専用の設定ツール「alsaconf」が用意されている。実行にはroot権限が必要なので、



次のようにして起動する。

```
$ su
<rootのパスワードを入力>
# alsaconf
```

最初の画面で［OK］を押して、サウンドカードの選択画面に移る。ESS Solo-1の場合は［ESS_Solo-1_ES1938/ES1946］を、VT82C686Bなら［VIA82C686A］を選択してEnterキーを押す（**画面1**）。そのほかのサウンドカードを使っている場合も、カードの一覧から適切なものを選べばよい。

続いてドライバオプションを設定する画面が表示される。ほとんどの場合、デフォルトの値で動作するので、そのままEnterキーを押して画面を進めていこう。オプションの設定が終わるとカード選択画面に戻る。先頭にある［No_more_cards］を選んでEnterキーを押せば、設定は完了だ。

alsaconfで行った設定は、/etc/conf.modulesに反映される。conf.modulesに追加される設定の一例（ESS Solo-1）を**リスト1**に示してある。

動作確認のヒント

システムを利用していくうちに、インストール時の設定では足らない部分や間違いに気づくことがある。できればインストール直後に、システムが正しく動作するか、ある程度確認しておくほうがいい。設定変更するためのポイントとあわせて紹介しておこう。

■ネットワーク（LAN）

「ifconfig eth0」として、ネットワークデバイスが有効になっていて、IPアドレスやサブネットマスクなどが正しくセットされているか確認しよう。また、「ping ＜ホスト名＞」でほかのホストから応答があるか確認できる。

設定変更は「linuxconf」、あるいはXで動作するグラフィカルツール「netcfg」を使って行う。

■メモリ

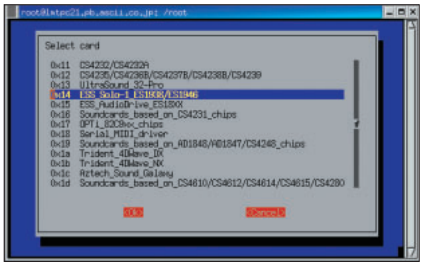
「free」コマンドを使って、搭載されているメモリ容量が正しく認識されているかチェックしておこう。64Mバイト以下の容量では、ほとんど問題ないが、それ以上のメモリを搭載している場合、たとえば128Mバイトのメモリがあるのに64Mバイトしか有効になっていない、といったケースがあり得る。

これを解決するには、/etc/lilo.confに次の1行を追加してliloコマンドを実行する（メモリサイズはシステム構成に合わせること）。

```
append="mem=128M"
```

■XFree86

とりあえず、起動すればOK（だと思う）。標準の設定ファイルは/etc/X11/XF86Config。Red Hat系ディストリビューションで3.3.6を使っている場合、設定ツールは「Xconfigurator」または「XF86Setup」、4.0.Xの場合は「xf86config」が利用できる。Mandrakeの「SaX」のように、ディストリビューションオリジナルの設定ツールが用意されていることもある。



画面1　alsaconfのカード選択画面

■ハードディスク

インストールが完了しているのだから、ハードディスクは正しく認識されて動作しているはず。問題となるのは、UltraDMA転送が有効になっているかどうかである。「hdparm /dev/hda」として、「using_dma = 1 (on)」と表示されればOK。「off」になっているようなら、「hdparm －d1 /dev/hda」として有効化しておこう。ハードディスクのデバイス名は適宜システム構成に合わせてほしい。

●

Cyrix号とDuron号についても、上記のチェックを行い、インストールが成功したディストリビューションでは、正しく動作することが確認できた。

格安パーツで組み上げたPCであっても、Linuxマシンとして立派に活躍してくれそうである。

▼リスト1　conf.modulesの例

```
# ALSA native
alias char-major-116 snd
alias snd-card-0 snd-card-es1938

# OSS/Free emulation
alias char-major-14 soundcore
alias sound-slot-0 snd-card-0
alias sound-service-0-0 snd-mixer-oss
alias sound-service-0-1 snd-seq-oss
alias sound-service-0-3 snd-pcm-oss
alias sound-service-0-8 snd-seq-oss
alias sound-service-0-12 snd-pcm-oss
```

4万円マシンは実用に耐えるのか？
～Cyrix号 vs. Duron号ベンチマーク対決～

最後のチェックポイントは「クライアントマシンとして快適に使えるのか」という点である。4万円Linuxマシンの実力を検証してみよう。

コンピュータの性能

通常コンピュータの性能は、CPUの内部クロック数、ハードディスクの平均シーク時間といったスペック値や、ベンチマークテストで得られた数値で表される。用途によっては、これらの値が重要な意味を持つ。たとえばデータベースサーバであれば、ベンチマークテストの結果として得られた「一定時間内に処理できるトランザクション数」という数値は、システムを評価するための重要な指標のひとつになる。

これがクライアントマシンとなると話がちょっと違ってくる。スペック値やベンチマーク結果は、そのコンピュータの絶対性能というか限界性能を表したもので、普通はクライアント用途でそれが求められることはない。確かにクライアント用途でも、処理が速い遅いということはある。しかし、アプリケーションの起動時間や複雑なスプレッドシートの計算処理の時間をいくら測ってみても、ユーザーが受け取る「体感的な速さ」を伝えることはできない。人によって感じ方や使用環境の違いがあるからだ。

それでもやっぱりベンチマークだ!

とまあ、堅い話はここまでにして、さっそくベンチマークだ！　「おいおい、なんでそうなるんだヨ」という疑問には「そこにコンピュータがあるからだ」とお答えしておこう……じゃなくて、文章でコンピュータの使いごこちを的確に伝えるのは難しいことに改めて気づいたからである。私には無理です。すみません。

それに、なんだかんだ言ってもベンチマークが「楽しい」のである。買ってくる（あるいは借りてくる）→組み立てる→動作確認する→ベンチマークをとる、という一連の流れはハードウェアを評価する際のPC関連メディアのルーチンでもある。そのため、一部のライターや編集者は「ベンチマークジャンキー」とも呼ぶべき症状を呈している。初めて触れるPCやパーツを前にすると、ベンチマークしたくて、したくてしょうがなくなるのだ。

というワケで、ここはひとつお付き合いいただきたい。読者のみなさんにも、まったく参考にならないということはないと思いますので。

ベンチマーク結果

今回のベンチマークテストには、733MHzのPentium IIIマシンが豪華特別ゲスト（評価の基準点）として参加している。主なスペックは表1のとおり。このマシンは「PenIII号」と命名した（ネーミングセンスなし）。では、順を追って結果を検証していこう。

■CPU

まず最初のベンチマークテストは「HDBENCH Clone Ver 0.14.1」を使ったCPUの性能チェック。HDBENCH Cloneは、Linuxにおける総合ベンチマークソフトの定番である。

やはり、Cyrix III 600MHzを搭載したCyrix号の成績が興味の焦点だろう。結果は**グラフ1**のようになった。

「Cyrix号轟沈」である。一般にCyrix IIIは浮動小数点演算に弱いとされているが、それがストレートに出てしまった感じだ。しかも、同クロックのDuronを搭載したマシンに整数演算でも大きな差をつけられてしまった。Cyrixファンのため息が聞こえてきそうな結果である。

■ハードディスク

ハードディスクのベンチマーク項目は、ブロック単位のシーケンシャルREAD／WRITE。計測には、ハードディスク専用のベンチマークツールである「Bonnie＋＋」を使用した。

```
$ Bonnie++ -s 150 > test.result
```

これがテスト時のコマンドライン。「−s 150」は計測に使用する一時ファイルのサイズ（単位はMバイト）指定だ。150Mバイトとしたのは、キャッシュの影響を除いてより正確にハードディスクの性能を計測するため。なお、PenIII号の計測では600Mバイトを指定している。

結果は**グラフ2**のようになった。

Duron号のハードディスク（富士通MPC3032）は、3.2Gバイトという容量からもわかるように2世代ほど前の製品である。転送モードはUltraDMA/33までしかサポートしていない。この結果はいたしかたないところだ。

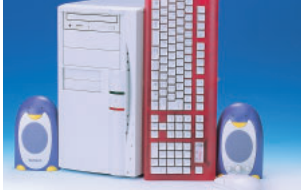
Cyrix号とPenIII号は、UltraDMA/66モードで動作している。この2台の差は、ハードディスク自体のスペックが原因だろう。Cyrix号が装備するQuantum Fireball lct20は、回転数4500rpm、平均シーク時間12ms、内部転送レート最大31Mバイ

CPU	Intel Pentium III 733MHz
チップセット	VIA Apollo Pro133A
メモリ	256Mバイト SDRAM（PC133 CL2）
ハードディスク	30Gバイト（7200rpm、8.5ms、UltraDMA/100）
グラフィックス	NVIDIA GeForce256 DDR（VRAM 32Mバイト）

表1　PenIII号の主なスペック

ソフトウェア	URL
HDBENCH clone	http://www.enjoy.ne.jp/~gm/program/hdbench/
Bonnie＋＋	http://www.coker.com.au/bonnie++/
SETI@home	http://setiathome.ssl.berkeley.edu/home_japanese.html
LAME	http://www.sulaco.org/mp3/

表2　ベンチマークに使用したソフトウェアの関連サイト



ト／秒であるのに対し、PenIII号のIBM Deskstar DTLA-307030は、7200rpm、8.5ms、35Mバイト／秒となっている（いずれもメーカー公表の数値）。Fireball lct20は、低発熱と静粛性、耐久性を重視した設計になっているのだ。

■グラフィックス処理

次にグラフィック処理性能を見てみよう（**グラフ3**）。

ここでの焦点は、Duron号のマザーボードで採用されているチップセットVIA SavagePro KM133のグラフィックス性能である。すでに触れたように、KM133に統合されているグラフィックスエンジンは、2DがS3 Savage2000、3DがSavage4に相当する。どちらも少し前の製品だが、発表当時はかなりの評価を得たグラフィックスチップだ。結果にもそれが反映されている。目をみはるほどの高性能ではないにしても、チップセット統合型としては十分な機能を備えているといえる。

Cyrix号のビデオカードで使われているTrident 3DImage975は、Savage4よりさらに古いチップで、記憶が確かなら1997年に発表された製品である。よって、過剰な期待は禁物というものだ。結果もそのとおりとなった。

■SETI@homeとLAMEによるベンチ

「SETI」は地球外の知的生命体と文明を探索するプロジェクトである。「SETI@home」は、SETIで行う必要のある大規模なデータ解析を世界中の個人のコンピュータで分散して行うためのクライアントプログラム。インターネットを利用してデータ配信と解析結果の収集が行われている。

今回のテストでは、同じ「ワークユニット」を使って解析時間を計測した（ワークユニットとは、一度に配信されるデータの単位。ユニットによって解析に必要な時間が異なる）。

「LAME」はLinuxで利用できるMP3エンコーダの代表格のひとつだ。音質に定評があり、エンコード速度もそれなりに速い。テストでは、3分33秒の音楽データをWAV形式からMP3にエンコードするのに要した時間を計測した。LAMEのバージョン3.20を使用し、特にオプションは指定していない。

結果は**グラフ4**のとおり。このグラフでは、PenIII号を100として残り2台の処理性能を相対的に表している。ほかのグラフと同じくグラフの棒が長いほど速く処理を行ったことになる点に注意してほしい。

ここでもCyrix号はDuron号に大差をつけられた。特にSETI@homeでいちじるしい。SETIのデータ解析は非常に膨大な科学計算を伴うため、CPUの演算能力がそのまま処理時間に反映する。Cyrix号の実際の処理時間は、なんと51時間51分36秒。CPUのくせに計算が苦手なのか？　大丈夫か？　Cyrix III！

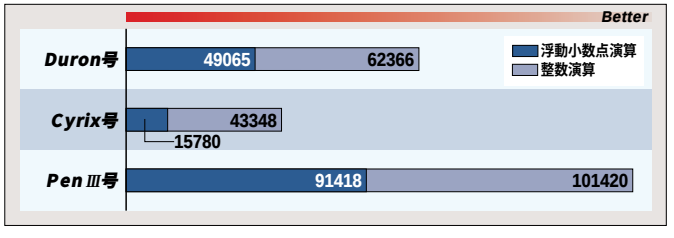
評価〜Cyrix IIIを弁護する

ご覧のように、Cyrix号はハードディスク以外ではDuron号に敗北を喫した。しかし、前のページの始めで（結構マジメに）述べたように、ベンチマークの結果がすべてではない。

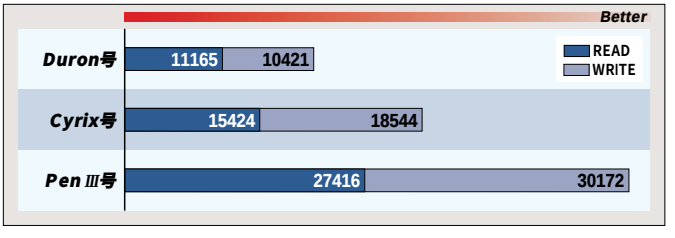
実際、動作確認のためにNetscapeを動かしたり、画面ショットをとるためにGIMPを使ったりしたが、「遅くてたまらん」ということはなかった。Webブラウジングや普通のアプリケーションを使うぶんには何ら支障はないのだ。クライアントマシンとしては、Cyrix号も十分な性能を持っている。

さらに言えば、非常に時間のかかってしまったSETI@homeのテストも、2昼夜にわたってデータ解析を黙々とこなした点を評価したい。Cyrix IIIはLinuxでも安定して動作することを証明してみせたわけだ（一部のバージョンのカーネルで不具合はあるが、すぐにも修正されるはず）。本当の意味での安定性の検証には、さらに長期間のロードテストが必要となるものの、この点をクリアできれば、低消費電力、低発熱といった長所を活かして、ルータなどの用途についてもLinuxで「使える」CPUとなるはずだ。

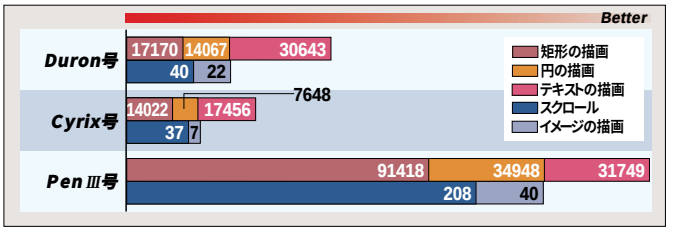
グラフ1　HDBENCH CloneのCPU項目



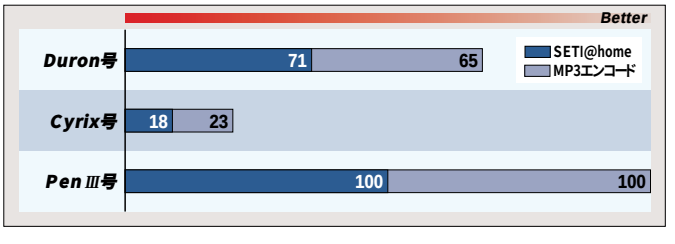
グラフ2　Bonnie++



グラフ3　HDBENCH Cloneのビデオ項目



グラフ4　SETI@homeとMP3エンコード




EPSON

最新プリンタをLinuxで活用しよう!

Linux印刷環境徹底ガイド

もう「印刷の時だけWindows」にしなくても大丈夫。
ネットワーク環境でプリンタを使いこなす秘訣を徹底解説

文:木原和明
Text : Kazuaki Kihara
Photo : Junko Kitade (Pacia)

Linuxの印刷システム

はじめに

UNIXシステムでは、lprと呼ばれるシステムを用いてプリンタを利用する。lprは、初めからネットワーク環境を前提としており、当然プリンタ共有も行えるシステムである。

LinuxもUNIXと同様にlprシステムを備えている。しかしlprシステムは、Windows／DOSの印刷の仕組みとは異なるため、これらの環境に慣れていたユーザーにとって、理解しやすいとは言えないものだ。

また、現実のネットワーク環境としては、WindowsとLinuxが混在していることが多い。後述するが、このような環境でlprシステムを用いることも可能だ。

また、LinuxやUNIX上でWindows環境用のサービスを提供するSambaを用いて、プリンタを利用するという方法もある。この方法だと、Windows側からLinuxの存在を意識しないで済むため、こちらのほうが使いやすいといってもいいだろう。

そこで今回の特集では、まず最初にlprシステムの設定や使い方に関するを解説する。続いて、WindowsとLinuxが混在している環境での、Sambaを用いたプリンタ共有について解説する。

また、エプソンやキヤノンから自社のカラーインクジェットプリンタを、Linuxシステムから利用するためのドライバが提供されるようになり、高品位なプリントが可能となった。そこで、これらのドライバのインストール方法やプリント方法についてもレポートする。

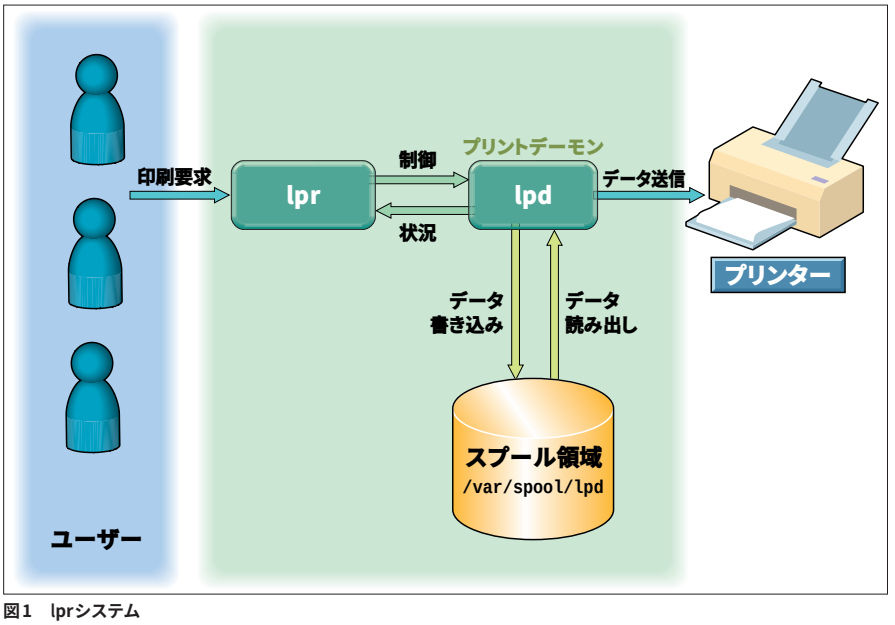


図1　lprシステム
実際の印刷を行うデーモン（lpd）とフロントエンドのlprからなる。

UNIXとlprシステムの歴史

Linuxのプリントシステムは、ほかのOSと独立したものではなく、基本的にUNIXの流れを汲むものである。そこでまずは、UNIXについて必要最低限のおさらいをしておこう。

UNIXオペレーティングシステムの開発は、1969年にベル研究所のメンバーによって行われた。稼動マシンはDEC社製PDP-11というものであり、当時のUNIXはバージョン7と呼ばれるものでであった。

その後、バークレー大学のメンバーによりviエディタやシェルなどの機能が追加されたバークレーUNIX 4.1BSD（Berkeley Software Distribution UNIX）が1981年に発表された。さらにネットワーク機能やプロセス間通信が採用された4.2BSDが1983年に発表され、研究所や大学などで広く使われるようになったのである。

バークレーでの開発と並行して、ベル研究所でもUNIXに対して数々の新しい機能の追加・更新が行われ、UNIXの普及に貢献した。その中で誕生したのがSystem Vと呼ばれるバー

ジョンである。現在利用されている商用UNIXシステムでは、サン・マイクロシステムズのSolarisがSystem Vベースだ。

BSDとSystem Vは、コマンドの一部が互いに異なるが、IBMのAIXのように、両方のコマンドをサポートするUNIXも存在する。

UNIXはマルチタスク、マルチユーザーのオペレーティングシステムである。そのため、あるユーザーが印刷している間に、ほかのユーザーがプリンタを利用できないようでは、非常に使い勝手が悪くなってしまう。

そこでUNIXでは、マルチユーザーを前提としたlprプリントシステムを利用して印刷を行うのが一般的だ。Linuxは、UNIXの優れた機能を受け継いでいるので、プリントシステムにはlprが使われており、もちろんUNIXのプリントシステムとまったく同じ機能を提供している。

lprシステム

Linuxではプリントスプーリングシステムとして、lprコマンドを使えるようになっている。ユーザーは、データを直接プリンタに送るのではなく、lprコマンドを用いて、印刷データの制御を行うデーモン（lpd）にデータを送る。印刷データはlpdによりいったんディスク上にスプールされ、要求された順番にスプールから読み出されてプリンタに出力される（**図1**）。

lprシステムは、自らが動作しているマシンに接続されているプリンタ（ローカル）と、ネットワーク上のほかのマシンに接続されているプリンタを同等に管理・運用することが可能で、用途や性能に応じて複数のプリンタを使い分けることができる。

■lprコマンドの使い方

lprコマンドには、大きく分けて2通りの使い方がある。1つは、lprコマンドをコマンドラインで起動し、印刷するファイルをコマンドの引数で指定する方法だ。

```
# lpr -Plp0 sample.txt
```

もう1つは、テキスト整形ツールなどの出力を、パイプでlprコマンドに渡す方法である。たとえば以下の例では、lsコマンドの標準出力をlprに渡している。

```
# ls | lpr -Plp0
```

このように、1つのプログラムに多くの機能を詰め込まず、単機能のプログラムを組み合わせて処理を行うというのは、UNIX以来の良き伝統であると言えよう。

lprコマンドを使う際に、よく用いるオプションを**表1**に示す。

■lprシステムのコマンド

lprシステムには、先に紹介したlprコマンドをはじめとして、プリントジョブの管理、削除を行うためにいくつかのコマンドが用意されている。ここでまとめて紹介しておこう。

```
# lpr -Plp0 sample.txt
```

lprコマンドは、プリントジョブをプリントキューに登録する。出力先のプリンタは－Pオプションで指定する。－Pオプションを指定しないと、デフォルトプリンタが選択される。プリントするファイル名を指定しないと、標準入力からの入力待ちとなる。

lpqコマンドは、プリントキューに登録され、処理中のプリントジョブの一覧を表示する。－Pオプションでプリンタ名を指定すると個別の状態を見ることができる（**リスト1**）。

lprmコマンドは、プリントキューに登録されたプリントジョブを削除する。

```
# lprm -Plp0 16
dfA016astro dequeued
cfA016astro dequeued
```

lpcはプリンタやデーモンの動作を制御する。

－P（プリンタ名）	「プリンタ名」で指定したプリンタに出力する。このオプションを指定しない場合は、デフォルトのプリンタ、または環境変数“PRINTER”で指定したプリンタに出力される。
－h	タイトルページを出力しないようにする。タイトルページには、ほかのユーザーの印
	刷物と区別しやすいように、印刷したユーザーの名前やファイル名が出力されている。
－m	印刷終了時にメールを送る。
－r	スプーリングの終了時にファイルを削除する。
－s	印刷するファイルをスプールディレクトリにコピーする代わりに、シンボリックリンクを利用する。サイズの大きいファイルを印刷する際に、指定する。
－#数字	各ファイルのコピーの枚数を指定する。
－J job	タイトルページに書くジョブ名を指定する。通常は、最初のファイル名が使われる。
－U user	タイトルページで使うユーザ名を指定する。
－i numcols	出力をnumcolsでインデントする。

表1　lprコマンドの主要オプション

- **プリンタ自体の使用可否を指定**
- **スプーリングキューの制御**
- **スプールデータの並び替え**
- **状態表示**

上記の4項目は、コマンドラインで直接指定することもできるし、オプションの指定をしないでlpcを起動し、lpcコマンド自身のプロンプトから指定することも可能だ。

```
# lpc
lpc> status
lp0:
     queuing enabled
     printing enabled
     no entries
     printer idle
lpc>
```

printcap

lprシステムは、プリンタの個別情報を“/etc/printcap”から読み出して動作する。以下にprintcapについて、**リスト2**に示した設定例を元に説明するが、詳細についてはオンラインマニュアル（man printcap）を参照してほしい。

lp0はプリンタ名である。ここで登録したプリンタ名を、lprコマンドの－Pオプションで指定する。

lpは出力デバイス名を指定する。Linuxでは、パラレルポートデバイス名は/dev/lp0、/dev/lp1、/dev/lp2のいずれかになるのが一般的だ。

shはタイトルページ（ヘッダページ）の有無を指定する。shを指定するとタイトルページは出力されない。

mxはスプールディレクトリのサイズを指定する。0は制限なしを意味する。

sdはプリントジョブのスプールディレクトリを指定する。これは各プリンタごとに分ける必要がある。

lfはlpdのエラー情報を記録するログファイル名を指定する。

ifはプリントデータをフィルタリングするプログラムまたはシェルファイルを指定する。

rmはプリンタがネットワーク接続されている場合、またはネットワークに接続されているほかのLinuxマシンに接続されている場合に、そのホスト名またはIPアドレスを指定する。

rpはrmで指定したホストに登録されているプリンタ名を指定する。

プリントシステムマネージャ

テキストの設定ファイル（printcap）をエディタで編集していくというUNIXの流儀に慣れていないと、プリンタの設定は少々手がかかる。

そのようなユーザーのために、GUIでプリンタの設定が行えるprinttoolというユーティリティが用意されている。printtoolの起動は、コマンドラインやコントロールパネルから行うことができる。実行には、root権限とX Window Systemの環境が必要である。printtoolを起動すると、メインウィンドウが表示される（**画面1**）。

■プリンタの種類を指定する

まず「追加」ボタンをクリックすると、プリンタの種類を設定するダイアログ（**画面2**）が表示される。

「ローカルプリンタ」は、ローカルマシンのパラレルポートに接続されているプリンタの設定を行う。

「リモートプリンタ（Unix lpd）キュー」は、ネットワーク上の別のマシン

▼リスト1　lpqの実行例

```
# lpq -Plp0
lp0 is ready and printing
Rank    Owner        Job   Files        Total Size
active kihara        16    sample.txt   216 bytes
```

▼リスト2　/etc/printcapの設定例

```
# Local printer
lp0:\
   :lp=/dev/lp0:\
   :sh:\
   :mx#0:\
   :sd=/var/spool/lpd/lp0:\
   :lf=/var/log/lpd/lp0-errs:\
   :if=/var/spool/lpd/lp0/filter:
# Remote printer
rlp0:\
   :rm=192.168.1.2:\
   :rp=lp0:\
   :sh:\
   :mx#0:\
   :sd=/var/spool/lpd/rlp0:\
   :lf=/var/log/lpd/rlp0-errs:\
   :if=/var/spool/lpd/rlp0/filter:
```

lp0はローカルプリンタ用エントリで、rlp0はリモートプリンタ用エントリ。

に接続されたプリンタ、またはネットワークに直接接続されたプリンタの設定ができる。

「SMB/Windows 95/98/NTプリンタ」は、Windowsマシンで共有されているプリンタの設定ができる。SMB（Server Message Block）は、Windowsで用いられているリソース共有のためのプロトコルである。

「NetWareプリンタ（NCP）」は、NetWareを利用するプリンタの設定ができる。

■プリンタ固有の設定項目

上記の4種類から設定するプリンタを選択すると、さらにプリンタデバイスのメイン設定ダイアログが表示される（**画面3**）。

「プリンタ名」フィールドには、プリンタの名前を入力する。

「スプールディレクトリ」フィールドには、プリントジョブデータをスプールするディレクトリ名を入力する。ローカル／ネットワーク接続に関係なく、1台のLinuxマシンで複数のプリンタを管理する場合は、スプールディレクトリは各プリンタごとに異なるディレクトリを設定する必要がある。

「ファイルサイズリミット」フィールドには、スプールするプリントジョブデータの容量の上限サイズをKバイト単位で入力する。たとえば、2Mバイトであれば2048と入力する。制限の必要がなければ、0を入力する。

「プリンタデバイス」フィールド（ローカルプリンタのみ）には、プリンタが接続されているデバイス名を入力する。デフォルトでは/dev/lp0となっている。

「リモートホスト名」フィールド（リモートプリンタのみ）には、プリンタが接続されているネットワーク上のマシン、またはネットワークに直接接続されているプリンタの、ホスト名またはIPアドレスを入力する。

「リモートキュー名」フィールド（リモートプリンタのみ）には、上記プリンタのキュー名（プリンタ名）を入力する。

「入力フィルタ」フィールドには、プリントジョブデータのフィルタリング（データ変換）処理をするフィルタを選択する。

■プリンタの種類

以上の項目を設定し、選択ボタンをクリックすると、続いてフィルタ設定ダイアログボックスが表示される。

「解像度」は、プリンタの解像度（1インチあたりのドット数）を選択する。

「用紙サイズ」は、デフォルトの用紙サイズを選択する。

「色深度」は、1ピクセルあたりのビット数を選択する。これはカラープリンタでモノクロプリントのみ行う場合などに有効である。

■両面印刷設定

「マージン」は、紙面の余白を指定する。単位は「ポイント」（1ポイントは1/72インチ）で、「左右」「上下」それぞれ入力する。

「追加GSオプション」には、このあとで説明するGhostscriptへのオプションを入力する。Ghostscriptは、UNIX系OS用のPostscriptインタプリタである。

すべて設定したのち「了解」ボタンをクリックすると、設定を有効にしてプリンタが登録され、メインウィンドウに戻る。登録が完了したプリンタをテストするにはテストするプリンタを選択し、「テスト」メニューからASCIIテキストページの印刷か、Postscriptテストページの印刷を選択する。

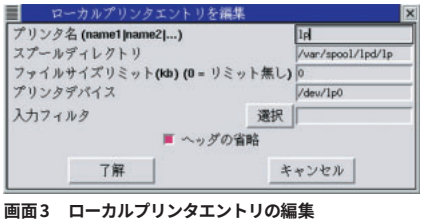
ASCIIテキストページの印刷では、Courierフォントを使用した10ポイントでのテストページが印刷される。

Postscriptテストページの印刷では、テキストとグラフィック（Red Hat Linuxの場合は、Red Hat社のロゴなど）が混在したテストページが出力される。

テストページの印刷が正しく行われない時は、パラレルやネットワークケーブルの接続やプリンタまたはデバイスの設定内容を再度確認しよう。

画面1　プリントシステムマネージャ

画面2　プリンタエントリの追加
追加するプリンタの種類を選択する。



画面3　ローカルプリンタエントリの編集
ここで設定した内容が/etc/printcapに書き込まれる。

WindowsとLinuxでプリンタを共有する

Linuxマシンに接続されたプリンタを、ネットワーク上のWindowsマシンから共有するにはいくつかの方法がある。

ひとつはlprを利用した方法で、その場合はWindowsマシンにlprクライアントソフトウェアをインストールする。その後プリンタドライバのプロパティで、印刷先のポートとして指定すればいい。つまりWindowsマシン側に、UNIXで用いられているlprシステムの仕組みを取り込ませるやり方だ。

もうひとつの方法は、Linux上でSambaを動作させ、Linuxマシンに接続したプリンタを、WindowsのSMBプロトコルを用いた共有プリンタとして利用する方法である。Linuxマシンに「Windowsのふり」をさせると思えばいい。

どちらの方法にも一長一短あるが、Sambaを使う後者の方法だとWindowsマシン側の設定が楽ということもあるので、実際にはSambaが多く用いられていると思われる。

prクライアントによる共有

UNIX系OSのマシンが主体のネットワークに、わずかなWindowsマシンが存在するような場合は、Windowsマシンがlprシステムを利用できるようにするのが自然だ（このような構成のネットワークは、あまり多くはないと思われるが）。

そのためにはWindowsマシンにlprクライアントソフトウェアをインストールして、印刷先としてネットワーク上のLinuxマシンに接続されたプリンタを指定すればいい。Windows用のlprクライアントソフトウェアは、フリーソフトウェアやメーカーが配布しているソフトウェアがいくつか存在している。

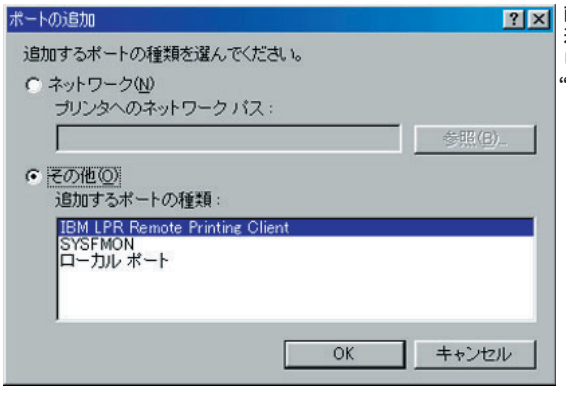
ここでは、IBMのIBMLPRプログラム（http://www.ibm.co.jp/printer/download/lpr.html）による設定方法を解説する。

まずLinuxマシン側でlprプリンタの設定がされていることが前提になる。ここでは、前のセクションを参照して正しく設定できているものとする。

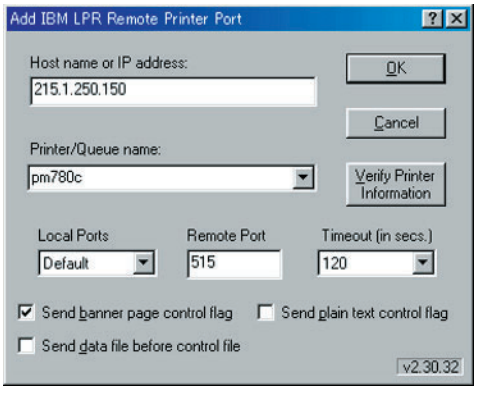
IBMLPRプログラム（instlpr.exe）は実行ファイル型のアーカイブなので、ダウンロード後に実行することでインストールされる。プリンタドライバのプロパティで印刷先のポートとしてIBM LPRを選択し（**画面1**）、さらにプリンタが接続されているLinuxマシンのホスト名とプリンタ名を追加する（**画面2**）。

Sambaによる共有

いまさら説明の必要もないと思うが、SambaはWindowsのリソース共有プロトコルであるSMB（Server Message Block）をUNIXで動作させるものである。Sambaを利用することで、UNIXマシンとWindowsマシン間でファイル



画面1　プリンタポートの追加
「その他」のポートから、“IBMLPR”を選択する。



画面2　IBMLPRの設定
出力先のホスト名またはIPアドレス、プリンタキュー名を指定する。

共有やプリンタ共有が可能となる。つまり、SambaはUNIXのlprプリンタをWindowsのネットワークプリンタの「ふり」をさせることができるのである。

Sambaは、主なUNIXプラットフォームの多くで動作している。もちろんLinuxにも対応しており、多くのディストリビューションに標準で含まれている。

最新バージョンは2.0.7であり、Windows 2000対応で起きていたいくつかの問題点が修正されている。日本語版は2.0.7-ja_2.2が最新版となる。オリジナル版のSambaでも日本語を用いたファイル名は扱えるが、日本語版のSambaではディレクトリ名にも漢字が使えるようになっている。

Sambaの実体は、smbdおよびnmbdという2つのデーモンであり、これらがWindowsマシンからの要求を処理している。smbdは、SMBプロトコルを使用してWindowsからの要求を受け、ファイルサービスとプリンタサービスを提供する。nmbdは、WindowsからのNetBIOSネームサービスの要求を受け、ホスト名をIPアドレスに変換して返す。

▼リスト1　rpmコマンドによるインストール

```
# rpm -Uvh samba-2.0.7-ja_2.2.i386.rpm
```

■入手方法

Sambaの最新版はWebサイト（http://www.samba.org/）などから入手できる（**画面3**）。また、日本語版のSambaは日本Sambaユーザ会のWebサイト（http://www.samba.gr.jp/）から入手できる（**画面4**）。本誌の付録CD-ROMにも日本語版の最新バージョンを収録している。

機能追加やバグ修正、セキュリティホールの解消、動作確認ディストリビューション情報を得るために、定期的にこれらのWebサイトをチェックするといいだろう。

Sambaのインストール

Red Hat系のディストリビューションでは、RPMパッケージが提供されていることが多いので、これを利用すればすぐに使うことができる。

特にパッケージが用意されていないディストリビューションでは、tarボールを用いてソースコードからコンパイルする。また、提供されているパッケージが古い場合にも、最新版のtarボールからコンパイルするほうがいいだろう。

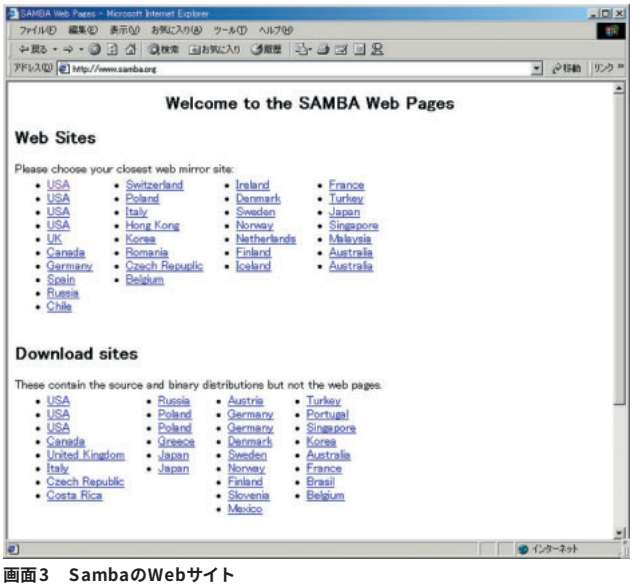
いずれの方法でも、インストール作業はrootアカウントで行う。

■RPMを利用したインストール

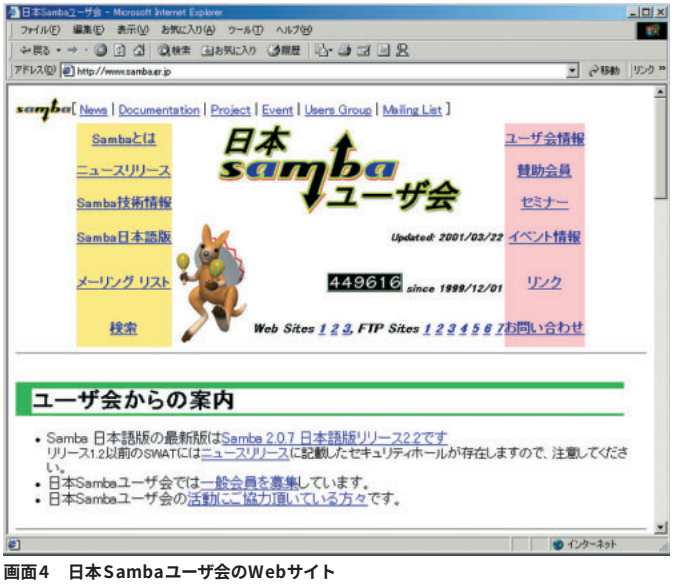
RPMパッケージを利用したインストールはrpmコマンドで行う（**リスト1**）。

最近のディストリビューションは、インストール時に用途（デスクトップ／サーバ）を指定するものが多い。ほとんどの場合、サーバ用途ではSambaは標準でインストールされる。

現在利用しているパッケージよりも新しいものが提供されている場合は、問題点の解決やセキュリティホールの解消が行われている場合が多いので、アップグレードしておくといいだろう。



画面3　SambaのWebサイト
オリジナルのパッケージはここから入手できる。



画面4　日本Sambaユーザ会のWebサイト
日本語化パッケージや日本語のドキュメントは、ここで入手可能。

■ソースからコンパイルする

RPMパッケージが利用できないディストリビューションでは、ソースからコンパイルする。まず、Webサイトまたは付録CDからソースコードを入手し、作業用ディレクトリに置き、アーカイブの展開とコンパイルをする（**リスト2**）。

SWATを用いてSambaを設定する

Sambaは、今回解説するプリンタ共有だけでなく、ファイル共有の機能も持ち、しかも用途に応じて非常に詳細な設定が行える。UNIX系OSの流儀に基づき、Sambaの設定ファイル/etc/smb.confもテキスト形式であり、エディタを用いて手動で設定することも可能だ。

また、SWAT（Samba Web Administration Tool）というGUIの設定ツールが標準で備わっており、各種の設定だけでなく、動作状況の管理をNetscape NavigatorやInternet Exploler等のWebブラウザから行うことができる（**画面5**）。

■SWAT

SWATとは、WebブラウザからSambaの設定（smb.confの設定）やデーモン（smbd、nmbd）の起動／停止、動作状況を表示するツールである。rootではすべての機能を設定できる。root以外のSamba利用者も一部の機能を除き、動作状況の表示やパスワードの設定をすることが利用できる。

■設定の変更

まず、SWATを利用するために必要な準備を行う。Red Hat Linux 6.xをベースにしたディストリビューションでは、/etc/inetd.conf、/etc/servicesの2つのファイルに変更を加える必要がある。

Red Hat Linux 6.xでは、ネットワーク系のデーモンをintedを利用して起動するようになっている。SWATをinetdから起動するためには、エディタを使って/etc/inetd.confに**リスト3**の行を追加する。追加する位置は、ファイル内のどこでもいい。7番目のパラメータ（**リスト3**では/usr/sbin/swat）は、SWATプログラムの絶対パスを指定しているので、環境に応じて変更する必要がある。さらに/etc/servicesに、**リスト4**の1行を追加する。これでWebブラウザからポート901に接続要求がされたときに、SWATが起動するようになる。

▼リスト2　Samba日本語版のコンパイル手順

```
# cp samba-2.0.7-ja_2.2.tar.gz /tmp
# cd /tmp
# gunzip samba-2.0.7-ja_2.2.tar.gz
# tar xvf samba-2.0.7-ja_2.2.tar
# cd samba-2.0.7-ja_2.2/source
# ./configure --with-i18n-swat
# make
# make install
```

rpmコマンドより手順は多いが、それほど手間はかからない。

▼リスト3　/etc/inetd.confの変更点

```
swat stream tcp nowait.400 root /usr/sbin/tcpd /usr/sbin/swat
```

inetd経由でSWATを起動するのに必要な変更点。

▼リスト4　/etc/servicesの変更点

```
service swat
{
        port = 901
        socket_type = stream
        wait = no
        user = root
        server = /usr/sbin/swat
        log_on_failure += USERID
}
```

SWAT用の設定を追加する。

▼リスト5　xinetd用の設定ファイル

```
swat     901/tcp  # Samba Web Administration Tool
```

xinetdでは、設定ファイルを個別に用意する必要がある。

▼リスト6　inetdの再起動

```
# ps ax | grep inetd
   577  ?      S  0:00  inetd
  2056  pts/0  S  0:00  grep inetd
# kill -HUP 577
```

変更した設定を反映するために、inetdを再起動する。

ディストリビューションによっては、行頭に「#」を付けてコメントアウトしている場合もあるので、その場合はコメントを外すだけでいい。

なおinetd.confもservicesも、変更にはroot権限が必要だ。

Red Hat Linux 7.xでは、inetdに替えてxinetdを利用しているため、変更点も多少異なっている。xinetdは、inetdに比べて個別のデーモンごとに柔軟な設定が可能だ。

Red Hat Linux 7.xのインストール時に「ワークステーション」構成を選択すると、xinetdはインストールされないので、その場合はRPMパッケージからインストールしよう。xinetdのインストール後、/etc/xinetd.dディレクトリ内にswatというファイルを作成する。内容はリスト5のとおりだ。

以上のようにファイルを変更したら、inetdを再起動する必要がある（Red Hat Linux 7.xはxinetd）。psコマンドでinetdのプロセスIDを調べて、killコマンドの-HUPオプションで再起動する（**リスト6**）。または、/etc/rc.d/init.d内のスクリプトを用いて、

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

のようにしてもいいだろう。

日本語版のSambaでは、SWATの画面も日本語表示される（Red Hat Linux 7.0Jでは英語表示になる問題が見つかっている）。もし英語表示になる場合は、ブラウザの設定を日本語表示になるように変更しておこう。

以上でSWATを起動し、Sambaの設定変更に利用できる状態になった。

Sambaの設定

設定が完了してSWATが動作したら、同じマシンでWebブラウザを起動して、以下のようにURLを指定する。

```
http://localhost:901/
```

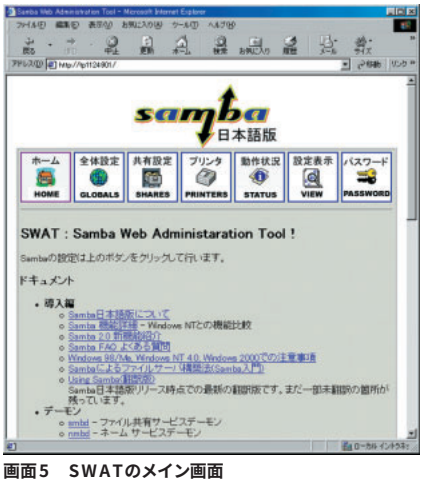
ネットワークの設定が正しく行われていれば、SWATが動作しているマシンとは別のマシンからSambaの設定を行うこともできる。その場合もまずWebブラウザを起動し、SWATが動いているマシンをホスト名またはIPアドレスで指定すればいい。たとえば、ホスト名が“tp1124”のマシン上でSWATを動かしているなら、以下のように指定する。

```
http://tp1124:901/
```

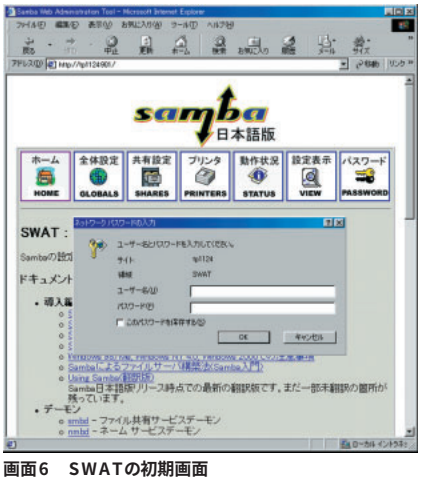
最初はSamba全体に関わる設定をする必要があるので、ユーザー名とパスワードの入力を求めるダイアログが表示されたら（**画面6**）、「ユーザ名」のフィールドに“root”、「パスワード」のフィールドにrootのパスワードを入力し、SWATを呼び出そう。

各ユーザーごとの個別の設定項目だけを変更する場合には、そのユーザー名とパスワードを入力しよう。

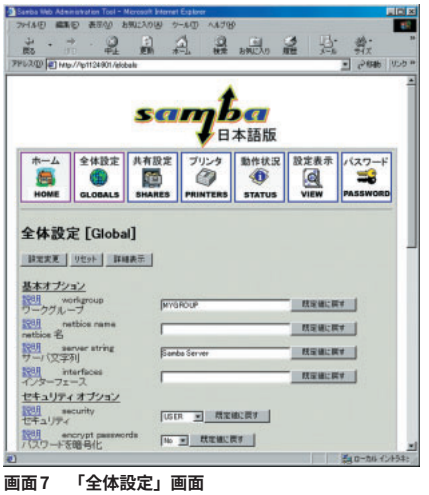
ここでのSambaの設定は、Windowsとのプリンタ共有に関する必要最低限な項目についてのみ解説する。ファイル共有など、その他の機能についてはヘルプ、マニュアルを参照して各自調べてほしい。**画面7**を見ればわかるように、SWAT画面の各パラメータの横には、「説明」のリンクがある。これらをクリックすると、詳細な説明が表示されるようになっているので、ぜひ活用しよう。



画面5　SWATのメイン画面
各種の設定や状況確認のサブ画面は、こちらから。



画面6　SWATの初期画面
WebブラウザからSWATを起動すると、ログインウィンドウが表示される。



画面7　「全体設定」画面
SWATの共通設定（ワークグループ名など）を行う。

SWATで行えるSambaの設定・管理の項目は、画面7上部の6つのボタンアイコン、「全体設定」、「共有設定」、「プリンタ」、「動作状況」、「設定表示」、「パスワード」に大別することができる。

■全体設定

「全体設定」では少なくとも以下の設定が必要だ。

“ workgroup ”は、Windowsネットワーク上でのワークグループ名を指定する。当然Sambaを利用するWindows側のワークグループ名と同じにする必要がある。同じにしないと、WindowsマシンからSambaサーバを見ることができない。

“ netbios name ”は、Windowsネットワーク上でのSambaサーバのマシン名を指定する。

“ server string ”は、マシンに関する簡単な説明を記入する。ここで指定した文字列は、エクスプローラで表示を「詳細」にした際に、「コメント」フィールドに表示される。

“ log file ”は、Sambaの動作記録（ログ）を保存するファイル名を指定する。

“ max size ”は、ログファイルのサイズの上限をKバイト単位で設定する。

これ以外の設定は、通常の使い方では、デフォルトのままでいい。

■共有設定

「共有設定」は、Windowsマシンとのファイル共有に関する設定を行う。今回はプリンタ関連の設定のみ取り上げるので、ここでは省略する。

■プリンタ

ここでは、/etc/printcapに登録されているプリンタが一覧できる。その中から設定したいプリンタを選択して、プリンタ選択ボタンをクリックする（**画面8**）。

“ comment ”には、プリンタに関する簡単な説明を記入する。「全体設定」の“ server string ”と同じように、エクスプローラ上から見ることができる。ここにプリンタの機種名など、どのようなプリンタなのかを書いておけば、共有する時に助かる。

“ path ”は、Sambaがプリントデータをスプールするディレクトリを指定する。このディレクトリは、全ユーザーから書き込み可能となるようにする必要がある。

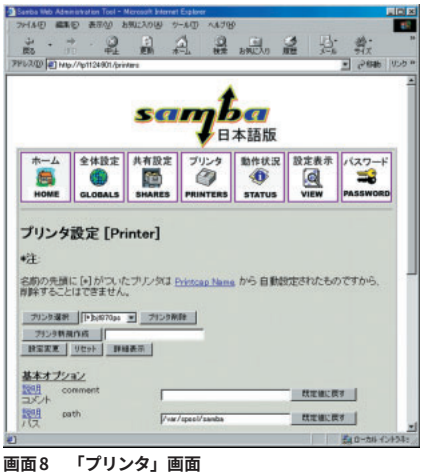
“ print ok ”は、Yesにしておかないとプリンタ共有ができないので、注意しよう。

“ browsable ”は、YesにしないとWindowsのエクスプローラやプリンタ設定の際に、共有プリンタ名が表示されない。

以上が、プリンタ共有に関する必要最低限の設定項目だが、「詳細表示」のボタンをクリックすると、さらに細かいオプションを設定することができる。

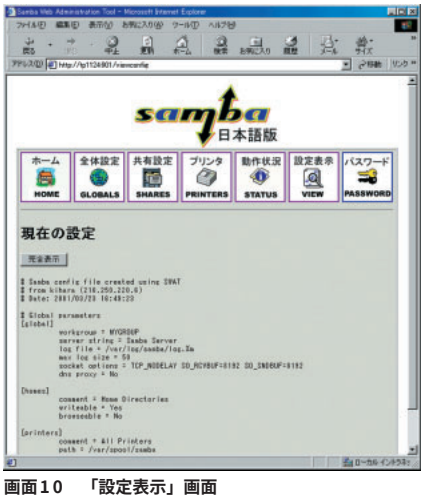
たとえば“ min print space ”は、スプールディレクトリの上限サイズをKバイト単位で設定する。デフォルトの0は、制限がないことを示す。スプールディレクトリは一般に/varディレクトリ以下に指定する。サイズの制限を設けておくことで、大量のプリントデータによって/varに割り当てたパーティションを使い切るのを防止できる。

また“ postscript ”は、プリンタがポストスクリプトプリンタの場合にはYes、非ポストスクリプトプリンタの



画面8　「プリンタ」画面
共有プリンタの設定は、ここで行う。

画面9　「動作状況」画面
smbd／nmbdの起動・停止、接続中のクライアントの確認が行える。Samba運用中のモニター画面ともなる。



画面10　「設定表示」画面
Sambaの現在の設定（/etc/smb.confの内容）を表示する。

場合にはNoを設定する。

以上の設定を有効にするためには、「設定変更」ボタンをクリックして、変更した項目を保存し、次に説明する「動作状況」の画面で、Sambaサーバを再起動する必要がある。

■動作状況

「動作状況」では、Sambaの現在の動作状況を表示する（**画面9**）。

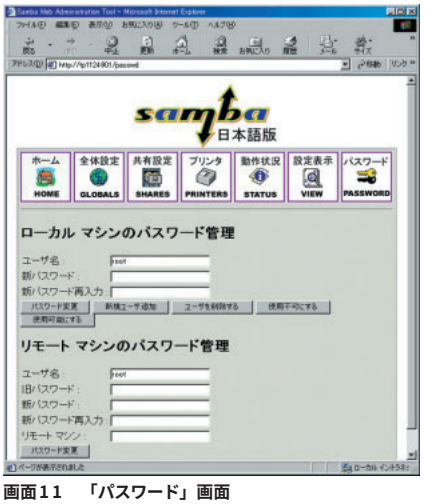
「自動再表示」をクリックすると、Sambaの動作状況の表示が一定間隔で更新される。サーバの動作をモニタするのに便利だ。

「smbd停止」や「nmbd停止」は、それぞれのデーモンの動作を停止させる。

「smbd再起動」や「nmbd再起動」をクリックすると、デーモンを再起動する。Sambaサーバに変更された設定を反映するには、デーモンを再起動する必要がある。

そのほか、現在接続中のクライアントや、Sambaサーバを共有しているクライアントマシンの情報を見ることができる。

■設定表示



画面11　「パスワード」画面
Sambaユーザーのパスワードを変更する。

「設定表示」では、/etc/smb.confに記されたSambaの設定内容を表示する（**画面10**）。

■パスワード

「パスワード」では、ローカルマシンのSambaユーザー名とパスワードを入力して、パスワード変更をクリックするとパスワードの変更ができる（**画面11**）。

「リモートマシンのパスワード管理」は、ネットワーク上のパスワード管理サーバで管理されているパスワードを変更することができる。

ここでのユーザー名とパスワードはSambaを利用するためのもので、Linuxにログインする際のユーザー名とパスワードとは異なる。

これらのユーザー名とパスワードを一致させたい場合は、Linuxのパスワードファイルである/etc/passwdからSamba利用者用パスワードファイルを作成すればいい（**リスト7**）。

また、新規にSambaユーザーを登録する場合は、smbpasswdコマンドを使用する。たとえば、ユーザー“kihara”を登録するには**リスト8**のように行えばいい。

これで、SWATやSambaが利用できる。ただし、ここでのユーザー名とパスワードはWindowsにログオンする際のユーザー名とパスワードに一致していないと、Sambaで設定した共有プリンタや共有フォルダ、SWATを利用することはできないので注意が必要である。

Sambaの自動起動

Linuxマシンを起動するたびに、SWATから手動でSambaを起動するのもめんどうなので、Linuxブート時に自動的に起動できるようにしておこう。

Red Hat 6.x系のディストリビューションでは、/etc/rc.d/rc?.d（?は0〜6の数字）に置かれたファイルで、Linuxのブート時に起動するプログラムを指定する。

0〜6の数字はランレベルを表す。たとえば、/etc/rc.d/rc3.d以下のファイルは、ランレベル3、つまりテキストログイン（Red Hat系ディストリビューションの場合）を表している。これらのファイルは、シンボリックリンクであり、実体は/etc/rc.d/init.dディレクトリ以下に置かれたスクリプトである。

Sambaを自動的に起動させるには、/etc/rc.d/init.dディレクトリ以下に、**リスト9**のシェルスクリプトを置き、適切な名前で/etc/rc.d/rc?.d以下にシンボリックリンクを張ればいい。

RPMパッケージからインストールしていれば、このファイルはすでに存在

▼リスト7　/etc/smbpasswdの作成

```
# cat /etc/passwd | mksmbpasswd.sh > /etc/smbpasswd
# chmod 660 /etc/smbpasswd
```

/etc/passswdを元にSamba用パスワードファイルを作成する。

▼リスト8　Sambaユーザーの登録

```
# smbpasswd -a kihara
New SMB password : パスワード
Retype new SMB password : パスワード
Added user kihara.
```

ユーザーの登録は、smbpasswdコマンドを利用する。

しているはずだ。

最後に、Sambaで使用するいくつかのポート（137～139）が有効になっていることを確認しよう。/etc/servicesを参照し、以下の6行があればいい。もし、コメントアウト（行の頭に#）されているようであれば、#を外して有効にしておこう。

```
netbios-ns      137/tcp
netbios-ns      137/udp
netbios-dgm     138/tcp
netbios-dgm     138/udp
netbios-ssn     139/tcp
netbios-ssn     139/udp
```

Linuxマシン上の設定は、これで終了だ。

Windowsの設定

Windows 98の設定として最初に必要なのは、Microsoftネットワーククライアントと TCP/IPの設定である。Windows 95やWindows Meも画面の構成は多少異なるものの、基本的には同じように設定すればいい。

設定は、コントロールパネルのネットワークを開いて、現在の構成に加える（**画面12**）。

■ネットワーククライアントの追加

まず、「現在のネットワークコンポーネント」の下にある「追加」ボタンをクリックし、クライアントをダブルクリックするか追加ボタンをクリックする（**画面13**）。

次にネットワーククライアントの選択画面でMicrosoftのMicrosoftネットワーククライアントをダブルクリックするかOKボタンをクリックする。

以上の設定操作で、Microsoftネットワーククライアントの追加が完了する。

■TCP/IPの設定

次にTCP/IPの設定を行うために、ネットワークコントロールパネルでTCP/IPをダブルクリックするか選択した状態でプロパティボタンをクリックする（**画面14**）。

TCP/IPのプロパティ画面が表示されたら、IPアドレスとサブネットマスクを設定する。ネットワーク内に稼働中のDHCPサーバがあるなど、自動的にIPアドレスを取得することができる環境であれば、「IPアドレスを自動的に取得」を選択する（**画面15**）。

次にバインドタブをクリックして「Microsoftネットワーククライアント」にチェックをつけてOKボタンをクリックする（**画面16**）。

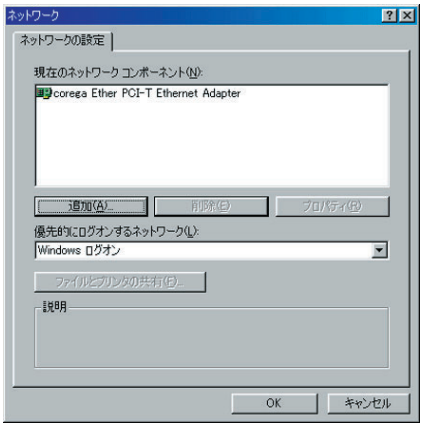
ワークグループ名を設定するためにネットワークコントロールパネルの「識別情報」タブを開き、ワークグループ名を入力する。このワークグループ名は、Sambaサーバと同じワークグループ名にしておく。そうしないとSambaサーバにアクセスすることができない。また、「コンピュータ名」のフィールドに、Windowsマシンのホスト名も設定しておく（**画面17**）。

設定が完了したら、ネットワークコントロールパネルのOKボタンをクリッ

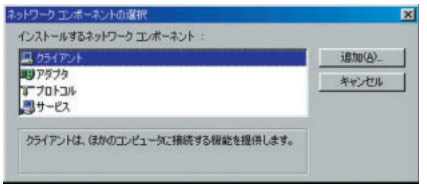
▼リスト9　Samba起動用のスクリプト

```
#!/bin/sh
case "$1" in
  start)
    echo -n "Starting SMB services: "
    daemon smbd -D
    daemon nmbd -D
    echo
    touch /var/lock/subsys/smb
    ;;
  stop)
    echo -n "Shutting down SMB services: "
    killproc smbd
    killproc nmbd
    rm -f /var/lock/subsys/smb
    echo ""
    ;;
  status)
    status smbd
    status nmbd
    ;;
  restart)
    echo -n "Restarting SMB services: "
    $0 stop
    $0 start
    echo "done."
    ;;
  *)
    echo "Usage: smb {start|stop|restart|status}"
    exit 1
esac
```

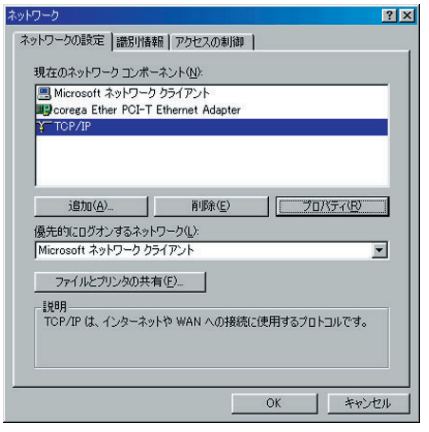
Linuxマシンブート時に、Sambaを起動できるようにする。



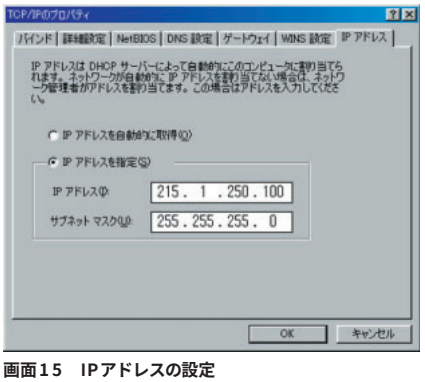
画面12　ネットワーク設定画面
コントロールパネルから開く。



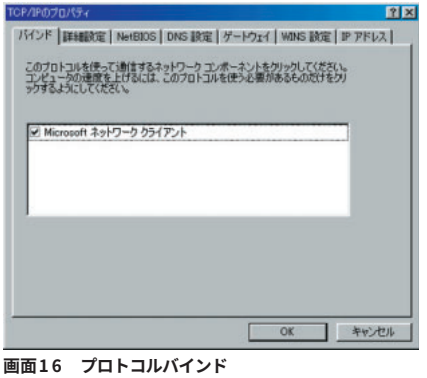
画面13　ネットワークコンポーネントの選択
クライアント、プロトコルなど必要なコンポーネントを選択する。



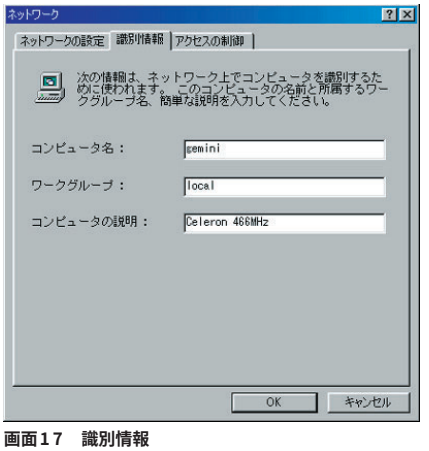
画面14　TCP/IPの設定
TCP/IPの設定はここから。



画面15　IPアドレスの設定
DHCPサーバがある環境なら、「IPアドレスを自動的に取得」を選択する。



画面16　プロトコルバインド
Microsoft ネットワーククライアントをTCP/IPにバインドする。



画面17　識別情報
マシンの名前、ワークグループなどを設定する。

クして閉じ、Windowsを再起動する。以上でWindowsのネットワーク設定は完了する。

すでにSambaサーバが起動していれば、Windowsマシンの再起動後、エクスプローラで「ネットワークコンピュータ」(Window Meでは「マイ ネットワーク」)を開き、Sambaサーバが表示されることを確認しておく。

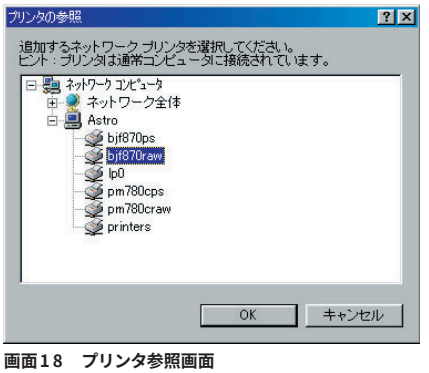
■プリンタの設定

次にWindowsのプリンタ設定を行う。

まず、コントロールパネルの「プリンタ」を開き、「プリンタの追加」をダブルクリックしてプリンタ追加ウィザードを起動する。

「ネットワークプリンタ」を選択して次へボタンをクリックする。続いて「参照」ボタンを押すと、「ネットワークコンピュータ」が表示されるのでプリンタを選択し、OKボタンをクリックする(**画面18**)。ここで、Sambaサーバ以下にプリンタ名が表示されない時は、Sambaサーバの設定を再度確認しよう。

最後は、プリンタドライバのインストールだ。プリンタの機種を一覧から選択する。新しい機種で一覧表に載っていない場合には、「ディスク使用」ボタンを押して、プリンタに付属のディスクからインストールしよう。エクスプローラ上で表示されるプリンタ名を設定すれば完了だ。テスト印刷を行って、正しく印刷できることを確認しておこう。



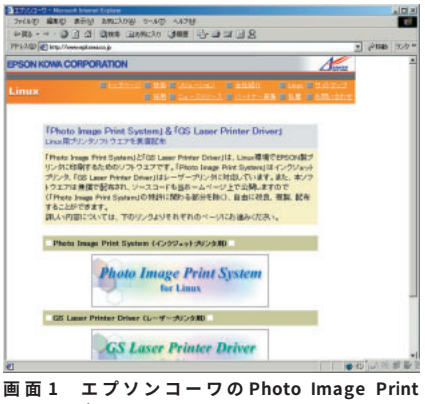
画面18　プリンタ参照画面
Sambaでの設定が正しければ、プリンタが表示されている。

最新プリンタをLinuxで使う

メーカー間の激しい競争の結果、カラーインクジェットプリンタは、高性能化・低価格化が急速に進んだ。もっともそれを利用できるのは、多くの場合Windowsであり、それ以外の環境ではMac OSがサポートされているくらいであった。

ところが最近になって、プリンタ市場で大きなシェアを持つエプソンとキヤノンからLinux用プリンタドライバが登場し、Webサイトからダウンロード可能となった。

ここではこれらのドライバのインストール方法、設定、Windowsとの共有などについてテストしてみた。



画面1　エプソンコーワのPhoto Image Print Systemダウンロード画面
このページから各プリンタ用ソフトウェアをダウンロードすることができる。

エプソンPM-780C

エプソンのカラーインクジェットプリンタ（PMシリーズ）、モノクロレーザープリンタ（LPシリーズ）に対応したGhostscript 5.50用フィルタは、同社のグループ企業であるエプソンコーワ株式会社（http://www.epkowa.co.jp/）から提供されており、Webサイトでダウンロード可能だ（**画面1**）。対応機種については、表1を参照してほしい。

カラープリンタ用のソフトウェアはPIPS（Photo Image Print System）と名付けられている。

RPMパッケージとtarボールがあるので、ディストリビューションに合わせて選択すればいい。

今回お借りしたプリンタはPM-780C（**写真1**）なので、対応するRPMパッケージはpips780-1.3-1.i386.rpmとなる。

■インストール

RPMパッケージのインストールは簡単で、まずrootでログインし、コマンドラインから以下のように入力する。

```
# rpm -ihv pips780-1.3-1.i386.rpm
```

readmeやフィルタプログラムなどは/usr/local/EPKowa/PM780Cディレクトリに、pips780は/usr/binにインストールされる。

写真1　エプソンPM-780C

写真2　PM-780Cのコネクタ
パラレルとUSBのコネクタを備え、幅広い機種に対応可能。

■pips780の起動

Photo Image Print Systemに含まれるフィルタは、lprシステムのフィルタとしても使えるし、同システムに含まれるGUIプログラムあるpips780（780はプリンタ機種を表す）を利用した印刷にも利用することができる。

主にlpr経由で使用する場合でも、設定情報を保存するために1回はpips780を起動する必要があるので、注意が必要だ。

pips780ではWindows版のドライバと同じように出力ポート（**画面2**）、用紙（**画面3**）、ハーフトーン、解像度や出力品質（**画面4**）、カラーマッチング（**画面5**）などの詳細な設定が可能になっている。設定の結果は、/etc/pipsrcに保存される。

pips780に関する設定やコマンドラインで起動する際のオプションについての詳細は、readme780を参照されたい。

■/etc/printcapの設定

Photo Image Print Systemをlprフィルタとして使用するためには、/etc/printcapへのプリンタ登録が必要である。

現バージョンでは残念ながら、プリントシステムマネージャからの登録はできないので、手動で/etc/printcapを編集して登録することが必要になる（**リスト1**）。

▼リスト1　PM-780Cの/etc/printcap

```
# EPSON PM-780C printer
pm780c:\
    :lp=/dev/lp0:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/pm780c:\
    :lf=/var/log/lpd/pm780c-errs:\
:if=/usr/local/EPKowa/PM780C/filter780:
```

pips780を利用したプリンタ設定を/etc/printcapに登録する。

■印刷方法

Linuxでの印刷は、pips780プログラムとlprの両方から行うことができる。pips780からの印刷は、PNGファイルを選択し、Printボタンをクリックするだけだ。

lprの場合は、

```
# lpr -Ppm780c sample.ps
```

とする。この場合は、ポストスクリプトファイルを指定する。

キヤノンBJ F870

キヤノンのインクジェットプリンタ

PMシリーズインクジェットプリンタ
PM-3500C/3300C/3000C
PM-2200C/2000C
PM-900C
PM-880C/820C/800C
PM-780C/770C/760C/750C
PM-670C
LPシリーズレーザプリンタ
LP-8500C/8300C/8200C
LP-3000C
LP-8600FX/8600FXN/8400FX/8400FXN/8300F
LP-1900/1900N
LP-1800

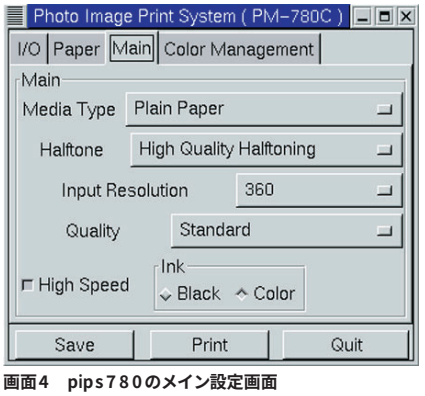
表1　対応プリンター覧

BJシリーズ用のGhostscriptフィルタ（Canon Bubble Jet Printer Filter for Linux）は、キヤノン販売（http://www.canon-sales.co.jp）のWebサイトからダウンロード可能である（**画面6**）。対応機種は、BJ F870／F860／F850、そしてF360である。

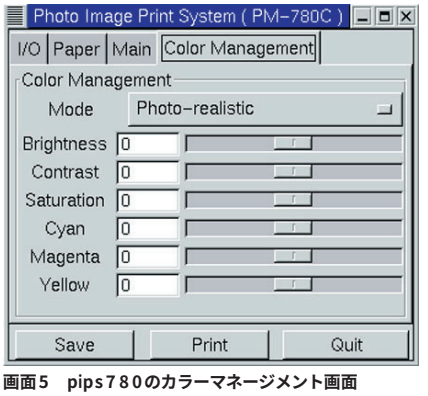
こちらもRPMパッケージ形式とtar

画面2　pips780のI/O設定画面
入力元と出力先を設定する。

画面3　pips780の用紙設定画面
用紙サイズやマージンを設定する。



画面4　pips780のメイン設定画面
用紙タイプ、ハーフトーン、解像度、画質を設定する。



画面5　pips780のカラーマネージメント画面
明るさやコントラスト、各色の設定ができる。

ボールが用意されているので、ディストリビューションに合わせて、どちらかをダウンロード、インストールすればいい。

LinuxでBJシリーズのプリンタを利用するためには、2つのパッケージが必要だ。ひとつは全機種共通で、フィルタプログラム／GUIプログラム／ランゲージモニタプログラム／ステータスモニタプログラムなどのプログラムが含まれているものだ（bjfiltercom-1.1-1.i386.rpm）。もうひとつは各機種ごとの実際に印刷を行うプログラムだ。今回は、BJ F870をお借りしたので、BJ F870用RPMパッケージを入手した。

また、インストール方法や設定についてのドキュメント（instruction.tar.gz）もダウンロードするといいだろう。

■インストール

インストールは、まずrootでログインし、コマンドラインから2つのRPMパッケージをインストールするという簡単なものだ（**リスト2**）。

フィルタプログラムなどは/usr/local/binディレクトリにインストールされる。ドキュメントは、tar形式ファイルなので任意のディレクトリで以下のように展開する。

```
# cd /tmp
# tar xvzf instruction.tar.gz
```

展開されたドキュメントはHTML形式なので、任意のWebブラウザでINDEX.HTM ファイルを開くとインストール方法やパラメータ設定、bjfilterコマンドのコマンドラインオプションの説明などを見ることができる。

■/etc/printcapの設定

今のところ、プリントシステムマネージャを用いての/etc/printcapへのプリンタ登録はできないので、手動で登録する必要がある。**リスト3**を見本に説明していこう。

“ bjf870ps ”は、Ghostscript用エントリで、ポストスクリプト形式のファイルを印刷できる。

“ bjf870raw ”は、RAWデータ用エントリで、BJ F870の制御コマンドであるBJラスタ形式のデータを印刷できる。Sambaのプリンタ共有を使用して、WindowsのBJ F870プリンタドライバやlprクライアントからプリントする場合は、このbjf870rawを共有して行う。

画面6　キヤノン販売のプリンタドライバダウンロード画面
このページから各プリンタ用プリンタドライバをダウンロードすることができる。

▼リスト2　BJ F870用プログラムのインストール

```
# rpm -ihv bjfiltercom-1.0-0.i386.rpm
# rpm -ihv bjfilter870-1.0-0.i386.rpm
```

BJ F870用プログラムと共通のユーティリティをインストールする。

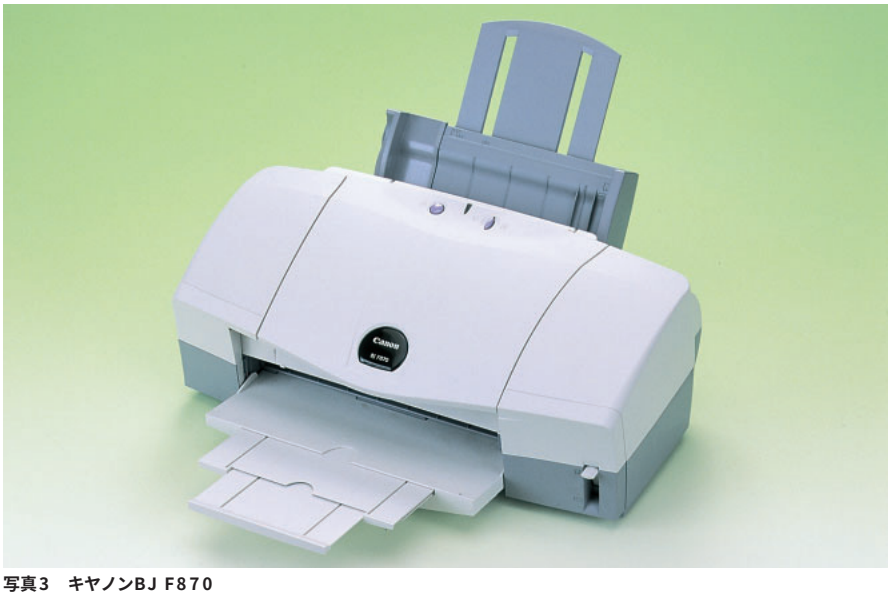


写真3　キヤノンBJ F870

写真4　BJ F870のコネクタ
PM-780Cと同様に、パラレルとUSBのコネクタを持つ。

■印刷方法

Linuxでのプリントは、lprコマンドで行う。印刷するデータは、ポストスクリプト形式を指定する。

```
# lpr -Pbjf870ps sample.ps
```

bjfilterコマンドはTIFF、BMP24、PPMファイルをBJラスタ形式データに変換できるので、lprと組み合わせることで画像ファイルのプリントも可能である。その場合は、ktermなどのコマンドラインから**リスト4**のように実行し、出力先プリンタにbjf870rawを指定すればいい。

印刷を実行すると、プリンタ設定用のGUIプログラムが起動されるので、用紙サイズや品質など必要な設定を行い（**画面7～9**）OKボタンをクリックすると印刷が実行される。

プリンタにデータが送られて実際に印刷が始まると、プリンタの状態を表示するステータスモニタが表示される（**画面10**）。

Windowsからの利用

以上のようにすれば、キヤノンBJ F870とエプソンPM-780CをLinux環境で利用できる。

Linuxマシンに接続されたこれらのプリンタを、Windowsマシンから利用するためには、前のセクションで説明したように、Sambaを利用するか、lprクライアントを利用することになる。その場合、Windows用のドライバを利用するため、Linuxマシン側のプリントキューは、RAWデータ用のものを使う必要がある（PM-780Cは“pm780c raw”、BJ F870は“bjf870raw”）。

なぜなら、Windowsドライバが生成するデータは、プリンタ用のRAWデータ（プリンタ制御コマンドとラスタデータ）だからだ。Linux側では、送られてきたRAWデータを何もフィルタリングせずに、直接プリンタに送ればいい。

Windowsマシン上で作成したポストスクリプトデータを印刷するのであれば、Linuxマシンではそれぞれの Ghostscript用フィルタ（BJ F870は“bjf870ps”、PM-780Cは“pm780cps”）を使用し、Windowsマシンではポストスクリプトプリンタドライバ（Adobe PS Driverなど）を用いることになる。

ただしその場合は、PPD（Post script Printer Description）というプリンタ用設定ファイルを個別に準備して、解像度や用紙サイズ、種類など正しい情報をプリンタドライバに伝える必要がある。

▼リスト3　BJ F870の/etc/printcap

```
# Canon BJ-F870 printer
bjf870ps:\
   :lp=/dev/lp0:\
   :sh:\
   :mx#0:\
   :rw:\
   :sd=/var/spool/lpd/bjf870ps:\
   :lf=/var/log/lpd/bjf870ps-errs:\
   :if=/usr/local/bin/bjpsprn:
bjf870raw:\
   :lp=/dev/lp0:\
   :sh:\
   :mx#0:\
   :rw:\
   :sd=/var/spool/lpd/bjf870raw:\
   :lf=/var/log/lpd/bjf870raw-errs:\
   :if=/usr/local/bin/bjprn:
```

bjfilterを利用したプリンタ設定を/etc/printcapに登録する。

▼リスト4　bjfilterをコマンドラインで起動する

```
# bjfilter --gui --model BJF870 sample.bmp | lpr -Pbjf870raw
```

bjfilterをコマンドラインで起動して画像データをBJ F870でプリントすることができる。

画面7　BJ F870用基本設定パネル
BJ F870用キューでプリントすると基本設定パネルが表示される。

画面8　印刷品質のカスタム設定パネル
印刷品質とハーフトーン設定ができる。

画面9　色調整のマニュアル調整パネル
各色の調整とカラーマッチング設定ができる。

HancomOffice

レッドハットは、デスクトップ向けLinuxディストリビューションとして、Vine Linuxの販売とサポートを2000年末より開始している。さらに2001年2月からは、韓国でオフィスソフトとして実績のあるHancomOfficeのLinux版である「HancomOffice 1.2 日本語版」と「Vine Linux 2.1CR」をバンドルした製品「Do Office」（価格は1万9800円）を出荷している。

HancomOffice 1.2 日本語版は、ワープロソフトの「HancomWord」、ペイントソフトの「HancomPainter」、表計算ソフトの「HancomSheet」、プレゼンテーションソフトの「Hancom Presenter」をパッケージにした製品である。

プリンタ特集の最後は、期待のLinuxオフィススイートであるHancom Officeに、オフィススイートに欠かせない印刷の面から迫ってみよう。

画面1　HancomShell
HancomOfficeの各ソフトウェアを起動する。

画面2　HancomWord
サンプルデータを開いた画面。なぜか旧字体が目立つ。

Do Office

Vine Linux 2.1CRは、Project Vineが開発したLinuxディストリビューションに、ダイナフォントとWnn6かな漢字変換システムを加えたもので、日本語環境を強化したデスクトップ向けディストリビューションである。

■HancomShell

HancomShellは、HancomOffice専用のランチャで、HancomOfficeに含まれる各アプリケーションを起動することができる（**画面1**）。

■HancomWord

HancomWordは、HancomOfficeの中心と言えるワープロソフトである（**画面2**）。縦書きはできないが、図形や写真などのイメージの挿入や表の作成ができる。

ほかのファイル形式のインポートもサポートしており、プレーンテキストはもちろんのことRTF、HTML、Microsoft Word、一太郎などのデータファイルの保存／読み込みもできる。

HancomWordで印刷を行うためには、まずファイルメニューの「プリンタ設定」でプリンタドライバの追加を行う。

プリンタドライバには、モノクロ用とカラー用が用意されているので、使用するプリンタに合わせて選択する。どちらもポストスクリプト用なので、プリンタもポストスクリプト対応の製品か、Ghostscriptでサポートされている製品が必要になる。

前のセクションで紹介したエプソンPM-780Cやキヤノン BJ F870であれば、Ghostscriptを使用して印刷を行える。

続いて、プリンタドライバのプロパティを修正して、「解像度（Resolution）」、「出力ポート（Port）」、「用紙サイズ（Paper）」、「用紙方向」、「ディザモード」の各項目を設定する。

「出力ポート」には、/etc/printcapに登録されているプリンタ名、またはファイル出力用のファイル名が選択できる。設定が済んだら、「Save」ボタンを押して保存しておこう。

印刷を行うには、ファイルメニューの「印刷」をクリックすればいい。WindowsやMac OSとも共通のインターフェイスだ。印刷ダイアログ（**画面3**）が表示され、印刷範囲、印刷枚数、拡大／縮小、印刷方式、印刷方向、オプションなどが設定できる。

「その他のオプション」で印刷するするオブジェクトを選択することができる。たとえば、テキストだけを印刷して内容を確認したい場合に、グラフィックスやイメージを印刷しないようにすれば、印刷時間が短くなるうえに、用紙が節約できる。

「印刷方向」でイメージ印刷を選択すると、プリンタポートではなく、イメージファイルとして書き出すことができる。イメージファイルの画像フォーマットは、BMPとPCXが選択できる。ファイル名には自動的にページ番号を示す3桁の番号が付けられて保存される。

そのほかのHancomOffice製品

HancomOfficeはオフィススイートではあるが、やはりその中心はHancom Wordであり、そのほかのアプリケーションは、HancomWordの付属品というように感じられてしまう。たとえば「印刷」ダイアログにしても、Hancom Wordよりだいぶシンプルだ。

HancomPainter、HancomSheet、HancomPresenterには、それぞれライバルともいえるソフトが存在する。機能や好み、用途に合わせて、どちらを利用するか選択するべきだ。

■HancomPainter

HancomPainterは、シンプルなペイントソフトである（**画面4**）。JPEG、GIF、BMP、PNG、PNM、XBM、XPMなど、一般的な画像フォーマットに対応しているほか、アニメーションGIFも読み込める。

印刷は、HancomWordとは設定方法が異なり、プリンタドライバの選択はできない。

ファイルメニューの「プリンタ設定」を開くと、システムに登録されているプリンタの一覧が表示されるので、その中から出力するプリンタを選択する。このプリンタ選択ウィンドウは、HancomSheet、HancomPresenterと共通だ。

印刷データは、ポストスクリプト形式なので、ポストスクリプト対応のプリンタかGhostscriptでサポートされているプリンタが必要なのは、HancomWordと共通だ。プリンタに直接出力する代わりに、ファイルに出力する機能も備えている。

■HancomSheet

HancomSheetは表計算ソフトで、Microsoft Excel形式のファイルに対応している。複数シートの読み込みや、2次元および3次元グラフの作成も可能である（**画面5**）。

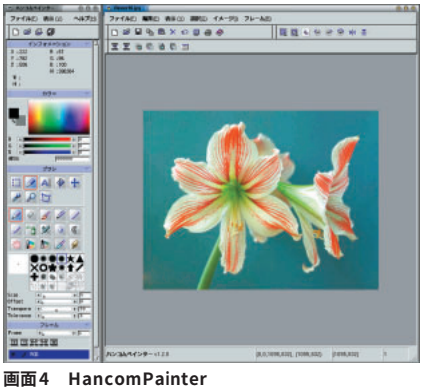
■HancomPresenter

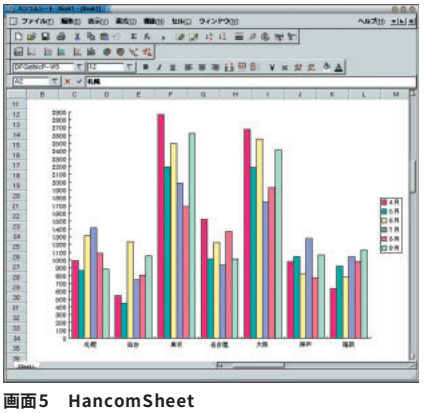
HancomPresenterは、アニメーション機能に対応したプレゼンテーションソフトで、Microsoft PowerPoint形式のファイルに対応している。テンプレートを使ったプレゼンテーション資料の作成も可能である（**画面6**）。

キヤノンBJ F870とエプソンPM-780Cの両プリンタで、HancomOfficeからの印刷をテストしたところ、Ghostscriptフィルタを利用すれば、問題なく印刷できることを確認した。

現状ではまだ粗削りな部分が残るHancomOfficeだが、最新のカラーインクジェットプリンタとの組み合わせにより、Linux上である程度のオフィス業務がこなせるレベルに到達していると感じられた。

画面3　HancomdWordの印刷ダイヤログ
印刷範囲や拡大／縮小などを指定する。


画面4　HancomPainter
シンプルな機能のペイントツール。


画面5　HancomSheet
ビジネスに欠かせない表計算ソフトだ。

画面6　HancomPresenter
プレゼンテーション資料やスライドが簡単に作成できる。

文：大野元久
Text：Motohisa Ohno

Kylix Review

Borland Kylix その機能に迫る

Linux初の本格的なビジュアル開発ツールであるKylixは、これまでのコマンドライン開発に比べ、開発時間を大幅に短縮できる。デスクトップアプリケーションの開発機能を提供するDesktop DeveloperとOracleやDB2対応のアプリケーション、Webサーバアプリケーションの開発を目的とするServer Developerがあり、開発言語としてはオブジェクト指向プログラミングのために拡張されたObject Pascalを使う。

すでに4月号の速報でも伝えたとおり、Borland Kylix（ボーランド カイリックス、以下、Kylix）は、Linux初の本格的なビジュアル開発ツールとして登場する。今回は、日本語ベータ版を使用して、特にKylixのコンポーネントを使ったプログラミングについて紹介する。

Linuxにおける ビジュアル開発環境

Kylixは、Windows用のビジュアル開発ツールであるDelphiをLinuxに移植したものである。これまで、Linux用のアプリケーション開発は、ソースコードを中心とするものがほとんどだった。実際にはKylix以外にも「ビジュアル開発」を名乗るツールは存在するのだが、Windowsで通常使われている製品レベルに相当するものは皆無であった。Kylixは、これを実現するツールである。

■ビジュアル開発について

まず、ビジュアル開発という手法について、簡単に説明しよう。

もともと、ソースコード主体でユーザーインターフェイスを持つアプリケーションを開発する場合、ウィンドウの位置、大きさ、表示する内容などは、すべてプログラムコードで指定しなければならない。これは、プログラムをコンパイル・実行してみなければ、実際にどのように表示されるかわからないということである。

これに対し、ビジュアル開発ツールでは、まずフォームと呼ばれるウィンドウにコンポーネントと呼ばれるソフトウェア部品を配置する。コンポーネントには、そのコンポーネントがどのように表示されたり動作するかといった情報を指定する。このため、設計段階からアプリケーションがどのように表示されるかを確認できるのである。これは、特にユーザーインターフェイスを持つアプリケーションを開発するときに威力を発揮する。これまで、Linuxでは「ちょっとしたツール」を作るためにもツールキットを呼び出したりしなければならず、微妙な調整なども常に実行して確かめてみる必要があった。しかし、ビジュアル開発ならば、微妙な調整も容易であり、使いやすいユーザーインターフェイスを持つアプリケーションを短時間で開発できる。

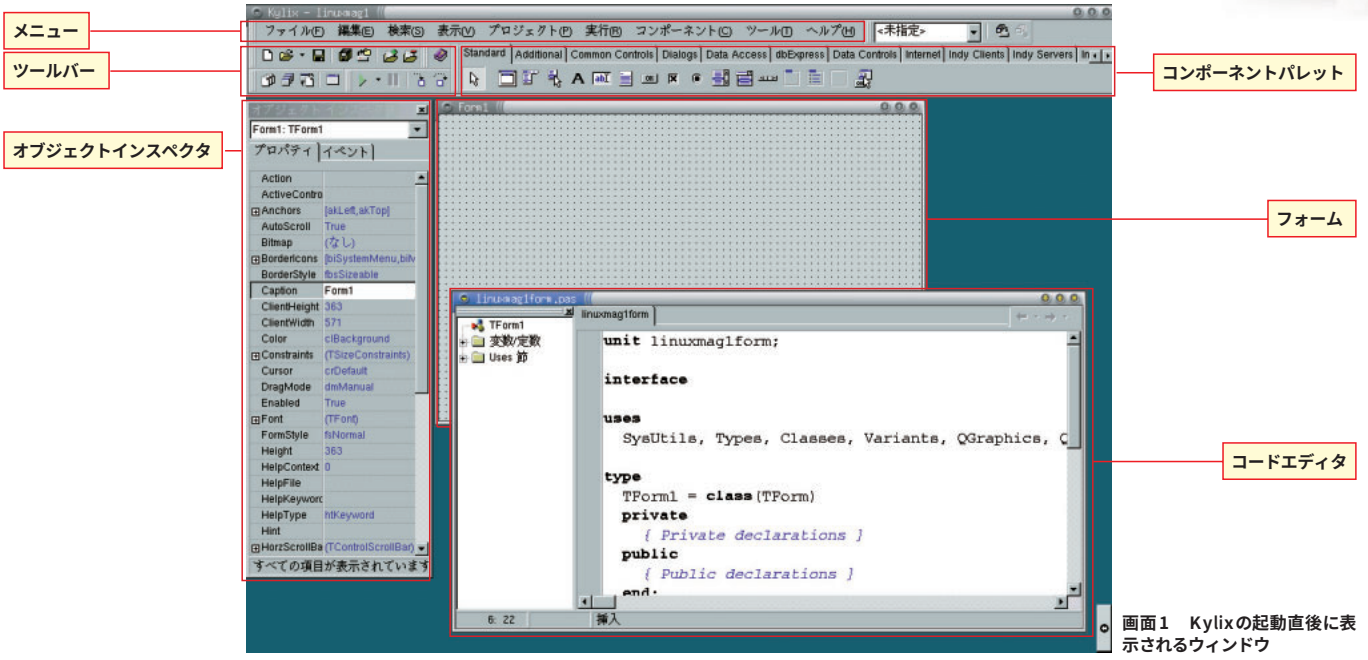
実は、ビジュアル開発のメリットは、ユーザーインターフェイスを持たないアプリケーションについても活用できるのだが、これについては後述する。

■ビジュアル開発の例

実際に、Kylixを使って単純なアプリケーションを作成してみよう。Kylixを起動した直後は、**画面1**のように画面上部のメニューやツールバー、左側のオブジェクトインスペクタ、空のフォーム（Form1というタイトルのウィンドウ）、プログラムを入力するためのコードエディタが表示される。

このフォームという場所が、アプリケーションのユーザーインターフェイスを設計する中心となる。

右上にあるコンポーネントパレットという場所には、Kylixで利用できるコンポーネント（部品）が登録されてい



画面1　Kylixの起動直後に表示されるウィンドウ

る。Kylixでは、非常に多くの部品が提供されているのだが、これらは目的別に分類されているので、必要なものを選択することはたやすいだろう。

画面2は、コンポーネントパレットで［Standard］という分類にあるButtonと［Dialogs］という分類にあるColorDialogを配置したようすである。

配置した部品には、自動的にコンポーネント名（ButtonやColorDialog）に数字が追加された名前が付けられる。ここでは、Button1とColorDialog1という名前が付いている。

ここで、コンポーネントの2つの種類について紹介しておこう。

先ほど配置したボタンは実行時にもフォーム上に表示される「ビジュアルコンポーネント」というものである。これに対し、ColorDialogは実行時には直接見えない「非ビジュアルコンポーネント」である。

ビジュアルコンポーネントは、そのままユーザーインターフェイスの設計に結びつくため理解しやすいだろう。特に、ビジュアルコンポーネントのことを「コントロール」とも呼ぶこともある。

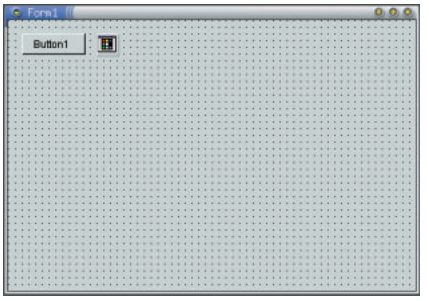
これに対し、非ビジュアルコンポーネントはプログラム内部の動作に関わるものだ。ColorDialogは、実際には色を選択する「目に見える」ダイアログを表示するのだが、ColorDialogコンポーネントが直接見えているわけではなく「目に見えるものを表示させる機能」を提供しているのである。つまり、非ビジュアルコンポーネントは「仕組み（ロジック）」を提供するものだと思えばよい。

たとえば、データベースを処理するアプリケーションで、データを「アクセス」するためのコンポーネントは非ビジュアルコンポーネントだが、データを「表示」するコンポーネントはビジュアルコンポーネントだ。

Kylixでは、両者が明確に分かれていることで、「ユーザーインターフェイスの設計」と「アプリケーションロジックの設計」を分離できるという特徴がある。

なお、フォームに配置した部品に表示される文字列は、オブジェクトインスペクタで設定できる。**画面3**は、Button1に表示する文字列を「Button1」から「Color」に変更しているようすである。Colorの前に「&」文字を追加しているが、こうすると直後の文字（この場合は「C」）がショートカットキーとして認識され、アプリケーションの実行時には［Alt］＋［C］とするだけで、ボタンを押す代わりになる。

また、フォームに配置したボタンをダブルクリックすると、このボタンをクリック（OnClick）したときの動作をプログラムできる。**リスト1**は、もうひとつのコンポーネントであるColorDialog1のExecuteというメソッ



画面2　ボタンとColorDialogを配置

ド（メンバ関数）を呼び出して、色を選択するダイアログを表示させ、［OK］が押されたとき（Trueが返されたとき）、フォームの色（Color）をダイアログで選択した色に設定する。

▼リスト1　ボタンに割り当てたイベントハンドラ

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if ColorDialog1.Execute then
    Color := ColorDialog1.Color;
end;
```

■プログラムの実行

ここまでできたら、プログラムを実行してみる。メニューから［実行］－［実行］を選んでもよいし、ツールバーの［実行］ボタンを押してもよい。プログラムは自動的にコンパイル／リンクされ、実行される。

画面4は、ボタンを押して色を選択しているようすである。

これが、ビジュアル開発という手法である。ここで示したものは、非常に単純なものだが、「フォーム」という場所に「コンポーネント」を配置して、「オブジェクトインスペクタ」でプロパティやイベントを割り当て、「コードエディタ」でプログラムを記述するという基本的な作業は、プログラムが大規模になっても同じである。

基本的なコンポーネント

Kylixのビジュアル開発の能力を支えているものが、CLXというコンポーネントフレームワークである。

CLXには、ユーザーインターフェイスの設計、データベース処理、インターネット対応、Webシステムの開発など、さまざまな目的に対応できる130種類以上のコンポーネントが登録されている。前述のとおり、これらはコンポーネントパレットに機能別に登録されているので、数が多いからといって目的のコンポーネントを選ぶのは難しくない。

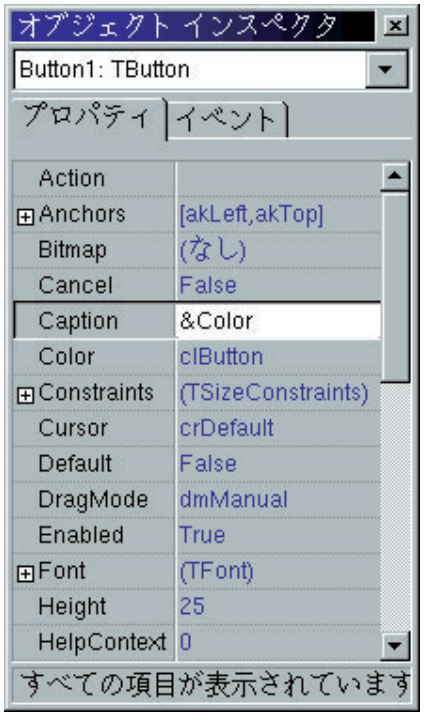
ここでは、Kylixに登録されているコンポーネントの中から、基本的なものを紹介しよう。

最初の例で使ったボタン（Button）を始め、文字列を入力するためのエディット（Edit）、テキストを表示するためのラベル（Label）、チェックボックス（CheckBox）やラジオボタン（RadioButton）、スクロールバー（ScrollBar）といった、ごく一般的に使われているものは、コンポーネントパレットの［Standard］という分類として登録されている。ウィンドウにメニューを追加するためのコンポーネント（MainMenu）もここにある。

■メニューの設計

まず、メニューの設計から見てみよう。

フォームにMainMenuを配置し、これをダブルクリックするとメニューデ



画面3　オブジェクトインスペクタ

画面4　プログラムを実行したようす

ザイナが呼び出される。あるいは配置したMainMenuコンポーネントを、マウスの右ボタンでクリックするとコンテキストメニューというプルダウンメニューが表示されるので、そこで［メニューデザイナ］を選んでもよい。

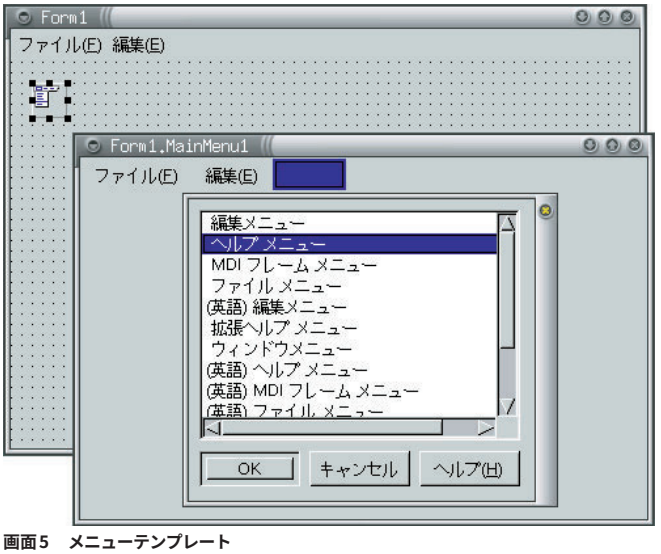
ここでは、通常のメニュー構造のように新しいメニュー項目を追加できる。たとえば、「ファイル」や「新規作成」、「開く」といった項目を追加できる。

また、マウスの右クリックでコンテキストメニューを表示させ、［テンプレートを挿入］を選ぶと、**画面5**のように定義済みのメニュー項目から一般的なプルダウンメニュー項目を追加できる。実は、メニューテンプレートにあるものは、できるだけそちらを使うほうがよい。Kylixは、追加したメニューの項目1つ1つについてもコンポーネントが追加される。このとき、メニュー項目に対応するコンポーネント名は入力した表示文字列（Caption）から自動的に生成される。そして、コンポーネント名は、プログラム中でも使われるため英数字でなければならない。FileやEditといった英語ならば、File1やEdit1といった名前が付けられるが、「ファイル」や「編集」といった日本語は英語に対応させられないため「N1」や「N2」のような名前になってしまうのだ。もちろん、あとから名前を変更することもできるが、最初からテンプレートを利用すれば、「ファイル」には「File1」といったわかりやすい名前が付けられる。

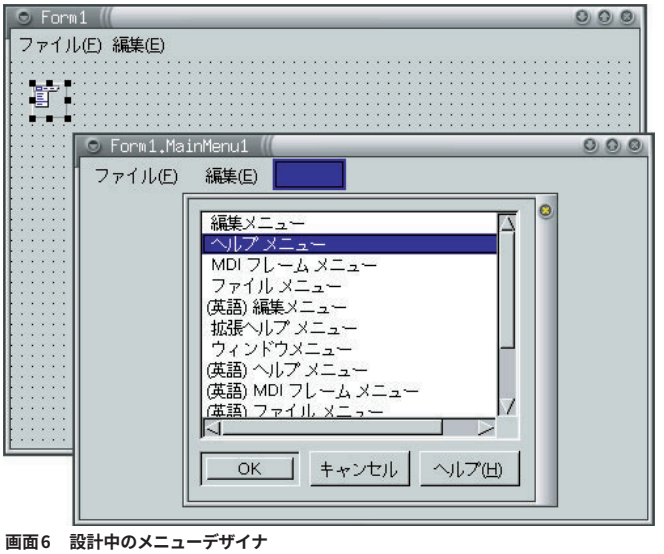
画面6は、メニューデザイナでいくつかの項目を追加しているようすである。

■コンポーネントとサイズの調整

次にメモ（Memo）コンポーネントを配置する。メモは、複数行に渡る単純なテキストを編集するためのコンポーネントである。ここで、オブジェクトインスペクタを使ってMemo1のAlignプロパティをalClientにする。す

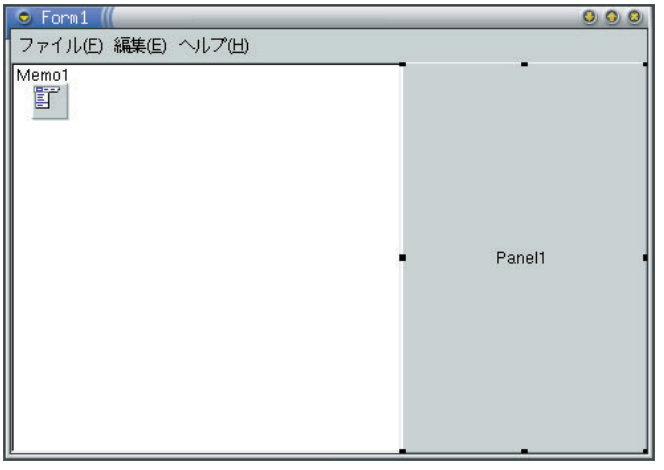


画面5　メニューテンプレート



画面6　設計中のメニューデザイナ

画面7　フォームに配置したメモ



画面8　パネルをフォームの右側に配置

ると、**画面7**のようにメモの大きさはフォームの領域いっぱいに拡大される。また、フォームの大きさを変更してみると、メモの大きさがそれに合わせて変更される。これはAlignプロパティの設定によるものだが、まさに、これがビジュアル開発の利点である。

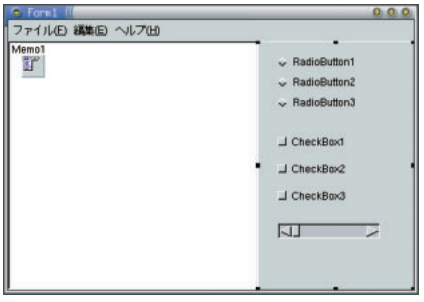
もし、フォーム上に配置したコンポーネントにAlignプロパティがなければ、「フォームの大きさが変更されたときに、メモの大きさをフォームに合わせて変更する」という処理をプログラムで記述しなければならない。しかし、コンポーネントは自分が配置された場所の大きさから、自分自身のサイズを調整する機能を持っているのだ。もちろん、ユーザーインターフェイスを設計するためのクラスライブラリを作成し、そのような自動調整機能を持たせることは可能だろう。しかし、その場合は実際に実行してみるまでどのように表示されるかを確認できない。Kylixなら、まさに「見たまま」である。

■コンテナコンポーネント

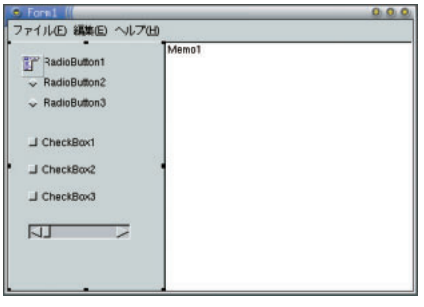
次にフォーム上にパネル（Panel）を配置する（フォームに空いている場所はないが、ここではメモの上に配置すればよい）。Alignプロパティを alRightにすると、パネルはフォームの右側に配置される（**画面8**）。なお、このときメモはPanelが占有する残りの領域のサイズに自動調整される。

パネルは、「コンテナコンポーネント」と呼ばれるものの代表的なものである。「コンテナ」とは「入れ物」のことであるが、Kylixではほかのコンポーネントを乗せることができるものをこう呼ぶ。

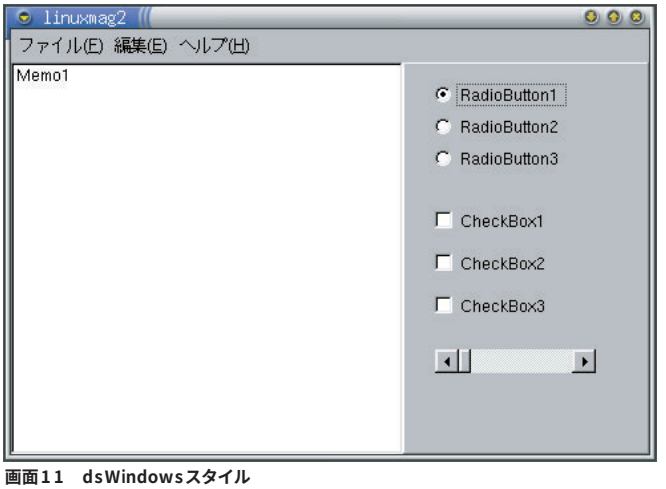
画面9は、パネル上にいくつかのコンポーネントを配置した例である。ここで、パネルのAlignをalLeftにする。すると、パネル上に配置したコンポーネントごと左側に移動する（**画面10**）。もし、メモのようにコンテナでないコンポーネント上にコンポーネントを配置した場合はこのようにはならない。また、フォームはもっとも基本的なコンテナだとみなすことができる。



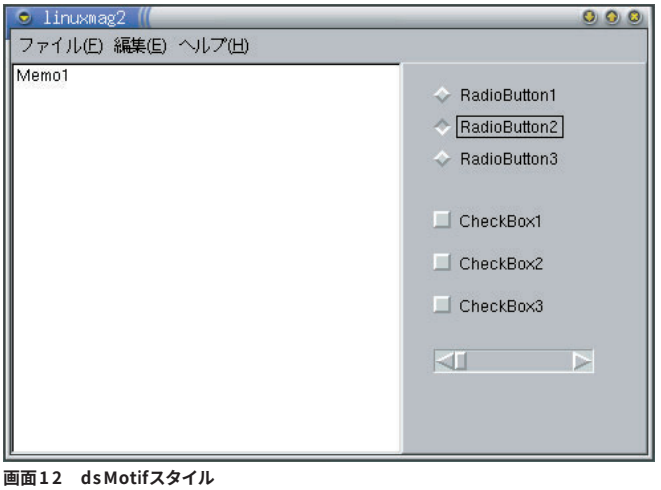
画面9　パネルにコンポーネントを配置



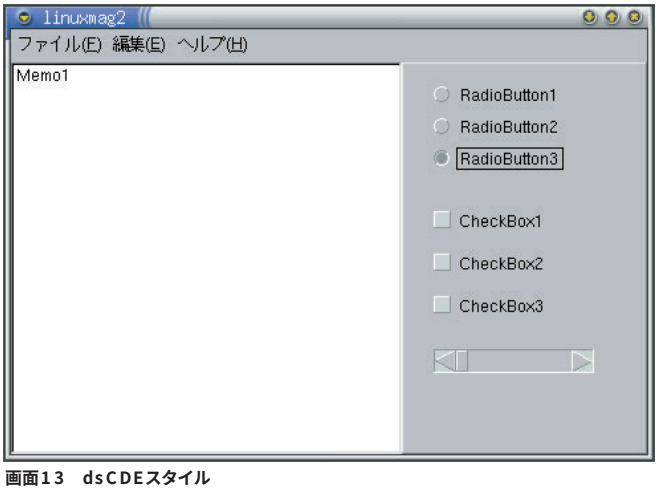
画面10　パネルを左側に移動



画面11　dsWindowsスタイル



画面12　dsMotifスタイル



画面13　dsCDEスタイル

■スタイルの変更

通常、ユーザーインターフェイスは、使用するデスクトップシステムのデフォルトで設定されているものになる。しかし、**リスト2**のようにアプリケーションのスタイルを変更するためのプログラムコードを記述すれば、**画面11**～**13**のようにWindowsライクなものやMotif、CDE（Common Desktop Environment）などに切り替えることができる。

CLXは、Windowsとの互換性を重視して設計されたコンポーネントフレームワークだが、スタイルを変更することで、よりアプリケーションの設計に柔軟性が与えられるだろう。

■アクションリスト

リスト2のイベントハンドラは、フォーム上の3つのチェックのOnClickというイベントに直接割り当てることができる。このため、別のコンポーネントのイベントハンドラを利用するために、これを呼び出すための「プログラムコード」を記述する必要はない。

このように、Kylixではオブジェクトインスペクタを使ってコンポーネントのプロパティをまとめて選択したり、複数のコンポーネントで同一のイベントハンドラを共有することで開発の効率化を実現している。

アクションリスト（ActionList）は、この共通化という考え方をさらに進めたものである。アクションリストは、プロジェクトの中のアクション（動作）を一括して処理する。アクションリストには、プログラムの中のアクションをコンポーネントとして追加する。**画面14**は、編集用のアクションなどを追加しているところである（なお、編集やデータ操作などの一般的な機能については、あらかじめ定義済みアクションが用意されている）。

アクションコンポーネントは、メニュー項目と同じく、コンポーネントパレットに登録されているものではない。アクションは、キャプション、チェックの有無、有効・無効、ヒント文字列、イメージリストのインデックス、ショートカットキー、アクションが呼び出されたときのイベントなど、共通化できる多くの要素を保持している。

ボタンやチェックボックスなど「動作を引き起こす」ためのコンポーネントには、Actionというプロパティがあり、このプロパティにアクションを指定すれば、自動的に必要な情報がコンポーネントに設定されるのだ。

アクションリストによって、ユーザーインターフェイスの設計とプログラムの機能とを分離することができる。たとえば、メニューの代わりにボタンを利用するときは、キャプションやヒント文字列などは設計しなおすことなく再利用できる。また、アクションリストは、特に後述するツールバーのボタンとメニュー項目を連動させたりするために役立つだろう。

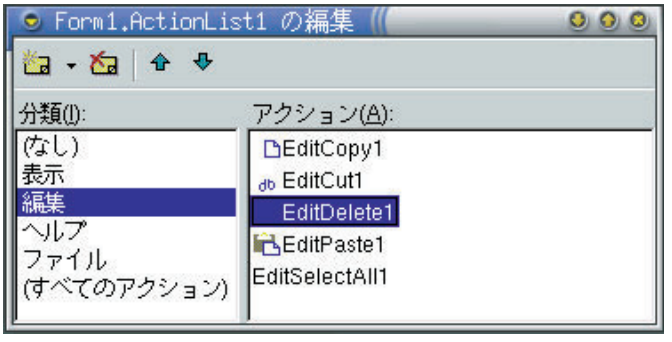
拡張コンポーネント

コンポーネントパレットの［Additional］というページには、Kylix独自のコンポーネントが登録されている。「Kylix独自」といっても、実際にはLinuxにおいてKylixのようなビジュアル開発を実現したツールはないので、前述の［Standard］ページのものもKylix独自であることに変わりはない。この区分はWindows版Delphiを引き継いでいるものと考えればよい。

以下に、主な拡張コンポーネントを紹介する。

■イメージ付きのボタン

ビットマップ付きのボタン（BitBtn）やスピードボタン（SpeedButton）は、



画面14　アクションリストエディタ

▼リスト2　スタイルを変更するためのイベントハンドラ

```
uses QStyle;

procedure TForm1.RadioButtonClick(Sender: TObject);
begin
  if RadioButton1.Checked then
    Application.Style.DefaultStyle := dsWindows
  else if RadioButton1.Checked then
    Application.Style.DefaultStyle := dsMotif
  else if RadioButton1.Checked then
    Application.Style.DefaultStyle := dsCDE;
end;
```

文字列とともにグリフと呼ばれる小さなイメージを表示できるボタンである。BitBtnには、Kindというプロパティがあり、bkOkやbkNoなどを指定すると定義済みのイメージと文字列の組み合わせが指定される。

画面15は、定義済みイメージ付きボタンのすべての組み合わせである。イメージ付きボタンにより、単純な文字列だけのボタンよりも直感的なユーザーインターフェイスを作成できる。

スピードボタン（SpeedButton）はパネルなどと組み合わせてツールバーを作成するために使うものだが、たいていの場合は［Common Controls］にあるツールバー（ToolBar）を使うほうが簡単だ。

■グリッド

Kylixには文字列を扱うための文字列グリッド（StringGrid）と、プログラムでセルを描画する描画グリッド（DrawGrid）という2つのグリッドコンポーネントがある。文字列グリッドでは、プロパティを変更することで文字列を入力することもできる。

■イメージなど

指定したイメージファイルをフォーム上に表示するのがイメージ（Image）コンポーネントである（**画面16**）。CLXでは、Windows版のDelphiで利用できるビットマップ（.bmp）、アイコン（.ico）に加え、PNG（.png）、ピクセルマップ（.xpm）、Drawing（.ddw）という形式をサポートしている。ただし、デフォルトではJPEGはサポートされない。

これは、CLXのビジュアル処理にはTrollTechのQtというライブラリを利用しているが、Qtの中にはJPEG処理をサポートしていないバージョンがあるためである。少なくともKylixとともに提供されるQtライブラリは、JPEG処理をサポートしているため、**リスト3**のようなコードをパッケージとして登録すれば、JPEGイメージを扱えるようになる。

また、JPEGイメージを描画するときはTBitmapというクラスのFormatプロパティに'JPG'を代入すればよい。

このほか、円や長方形などの図形を描画するShapeコンポーネント、LCDライクな表示のためのLCDNumberコンポーネントなどがある（**画面16**）。

■スプリッター

サイズ変更可能なフォームを使う場合、フォーム上に複数の領域を設けて実行時にこの境界線を任意の場所に移動させたいということがよくある。スプリッター（Splitter）は、こういうケースに役立つコンポーネントだ。

画面17は、2つのパネルの間にスプリッターを配置し、実行したようすである。スプリッターは、前述したAlignプロパティと組み合わせて使う。

■コントロールバー

コントロールバー（ControlBar）は、おもに複数のツールバー（ToolBar）を配置する場所として利用する。ただし、Delphiと違ってKylixでは、ツールバーやウィンドウのドッキング・フローティングはサポートされていない。

コモンコントロール

コンポーネントパレットの［Common Controls］として登録されているのがコモンコントロールである。この区分もWindows版Delphiを引き継いでいるものと考えればよい。

▼リスト3　JPEGを使うための形式の登録

```
implementation

procedure Register;
begin
TPicture.RegisterFileFormat('JPG', 'JPEG File', TBitmap);
end;
```

画面15　イメージつきボタン

画面16　拡張コンポーネントの例

■タブつきコントロール

タブコントロール（TabControl）やページコントロール（PageControl）は、「見出し」つきの場所を提供する。タブコントロールは見出しを提供するだけだが、ページコントロールは見出しごとにコンポーネントの配置場所となる場所を提供する。

タブコントロールは前述したコンテナコンポーネントの一種である。見出しによって表示内容を切り替えるといった処理のために使える。**画面18**は、タブコントロールを使ったカレンダーのプログラム例である。

なお、Delphiでは見出しを設定するTabsプロパティは文字列リスト型だが、KylixではTTabs（TTabのコレクションクラス）型となっているので注意が必要だ。

これに対し、ページコントロールは見出しごとに配置場所を提供する。この場所はTabSheetというコンテナコンポーネントになっている。ページコントロール自身は、コンテナコンポーネントではない。

見出しごとに、別々のページとしてコンポーネントを配置できるので、オプションの設定ダイアログなどで項目が多くなった場合などに、これを機能別に分けられる。ページコントロールを使ったダイアログは、Kylix自身でも多用されている。

■イメージリスト

既述のイメージ（Image）は、指定したイメージファイルをフォーム上に表示するものだが、イメージリストは複数のイメージをまとめて保持するための非ビジュアルコンポーネントである。単独でプログラムでイメージを描画するためにも使うが、ツールバーやメニューなど、ほかのコンポーネントが使うイメージを供給する目的でも使われる。

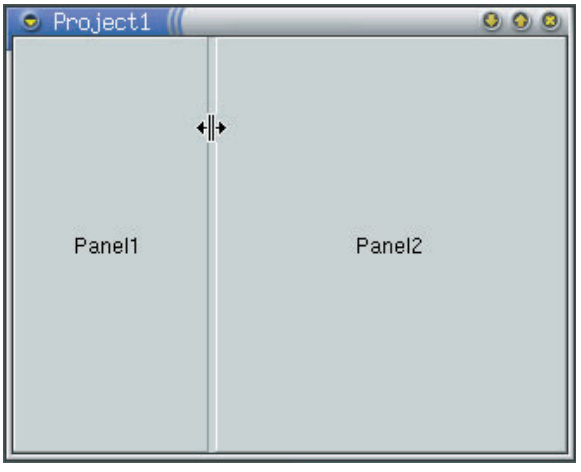
■ツールバー

アプリケーションの機能を呼び出すためにメニューコマンドを使う代わりに、イメージ付きのボタンを押すだけで呼び出せるようにしたのがツールバー（ToolBar）である。ほとんどの場合、ツールバーはイメージリストと組み合わせて使用する。**画面19**は、ツールバーを使った例である。

ツールバーは、ShowCaptionというプロパティをTrueにすることで、文字列を表示することもできる（**画面20**）。このように使う場合は、AutoSizeプロパティをTrueにしておくとよい。

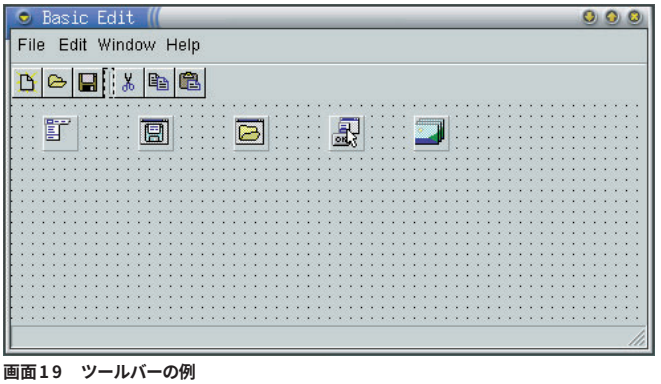
■ステータスバー

ステータスバーは、通常フォームの



画面17　スプリッターの例

画面18　タブコントロールの例



画面19　ツールバーの例

画面20　キャプションを表示した例

下部にアプリケーションの状態や簡単なヒント文字列を表示するために使うものである。**画面19**の下部にも配置されている。

ステータスバーは、Panelsというプロパティを使って複数の領域に分けて利用できる。たとえば、Kylixのコードエディタ（**画面1**）では、カーソル位置や入力モード（挿入・上書き）などを表示するためにステータスバーで複数の領域を割り当てている。

■ブラウザ機能コンポーネント

Kylixには、テキストビューア（TextViewer）とテキストブラウザ（TextBrowser）コンポーネントによって、HTMLテキストを表示する機能をアプリケーションに組み込める。

画面21は、テキストブラウザを使ったサンプルを実行した例である。テキストやリンク項目の色、アンダーライン表示の有無などはプロパティを使って切り替えられる。

コモンダイアログ

最初のプログラム例でColorDialogというダイアログコンポーネントを使ったが、このように汎用的な目的のためのダイアログを、コモンダイアログとして実装している。

Kylixが提供するコモンダイアログは、「ファイルを開く」や「ファイルの保存」でのファイル名の選択、「フォントの選択」、「色の選択」、「検索文字列の設定」、「置換文字列の設定」がある。

画面22は、「ファイルを開く」ダイアログの使用例である。

Delphiに慣れた人は、Filterプロパティの指定が変わっているので注意が必要だ。Kylix（CLX）のコモンダイアログでは、フィルタ指定は縦棒（|）で区切られるが、そこの括弧内で指定したマスクがそのまま使われる。

Webアプリケーションの開発

冒頭にも述べたが、Kylixは非ビジュアルなアプリケーションの開発でもビジュアル開発の威力を発揮できる。その代表的なものがWebアプリケーションだ。Kylix Server Developerでは、Apache対応のWebアプリケーションを開発するためのモジュールも作成できる。

［ファイル｜新規作成］で［Webアプリケーション］を選ぶと、**画面23**のようなダイアログが表示され、CGIまたはApacheのDSOモジュールのためのプロジェクトを自動的に作成できる。

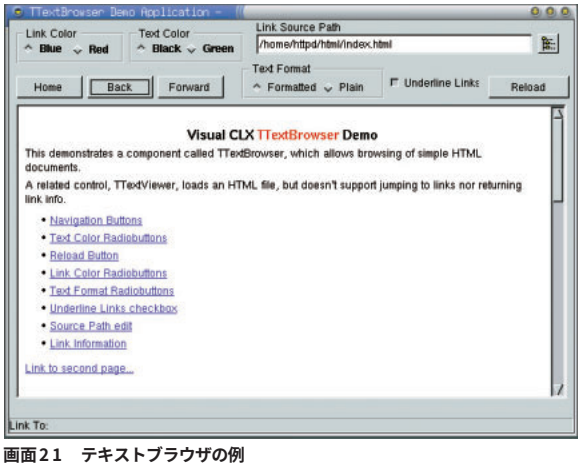
もちろん、こうしたWebアプリケーションはユーザーインターフェイスを持たない。このため、非ビジュアルコンポーネントしか配置できない、「Webモジュール」という場所が提供される。Webモジュールには、アプリケーションへの要求を処理するための「Webアクション」を登録できる。たとえば、1つのWebアクションを登録し、デフォルト（Defaultプロパティ＝True）とする。ここで、アプリケーションが呼び出されたときに、日付や時刻を文字列として返すようにプログラムする。

画面24は、このプログラムを使ってみたところである。

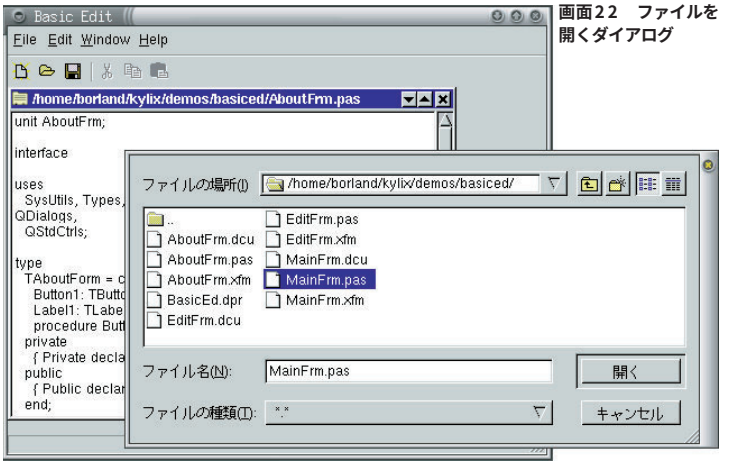
また、Kylixでは単純なHTMLテキストを返す代わりに、動的な情報をHTMLテキストに埋め込むためのページ生成コンポーネントを提供する。

このコンポーネントは、指定したHTMLテキストの中に特殊なタグ（<# …… >で囲まれたタグ）があるとイベント（OnHTMLTag）が発生し、そのタグをプログラムで作成するテキストに置き換えることができる。

また、今回は触れなかったがWebモジュールからデータベースにアクセスすることもできる。データベース対応のページ生成コンポーネントを使えば、特殊タグ名にデータベース項目名を指



画面21　テキストブラウザの例



画面22　ファイルを開くダイアログ

定するだけで自動的にそのタグを実際のデータに置き換えることができる。

インストール環境について

Kylixは、主要なLinuxディストリビューションで動作するように設計されている。

■基本的な動作環境

Kylixの動作環境としては、以下が指定されている。

- **Pentium200MHz（推奨PentiumII400MHz以上）**
- **64MバイトのRAM（推奨128Mバイト以上）**
- **200Mバイト以上のハードディスク容量**
- **CD-ROMドライブ**
- **マウスなどのポインティングデバイス**

また、動作するLinuxの条件は、以下のとおりである。

- **カーネルのバージョン2.2以上**
- **libgtk.soのバージョン1.2以上（グラフィカルインストールの場合のみ）**
- **libjpegのバージョン6.2（libjpeg.so.62）以上**
- **X11R6互換のターミナルサーバ（XFree86など）**
- **glibc 2.1.2以上（推奨glibc 2.1.3以上）**

ただし、すべてのLinux上で動作するわけではない。むしろ、「何もしていない」Linuxの場合は動作するものは限られる。これは、次に示す問題によるものだ。

■Kylixとglibcの問題

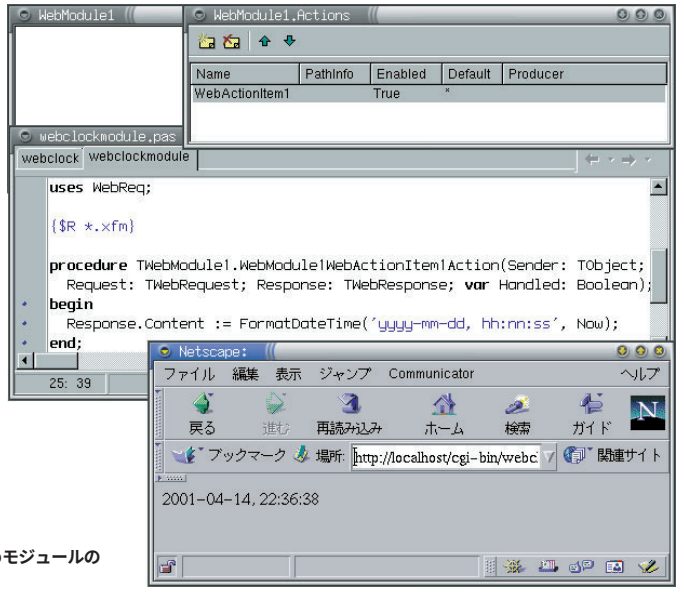
4月号の速報でも触れているが、現在使われているLinuxディストリビューションの多くはglibcに問題がある。これは、glibc 2.1.xに存在する「ローダの問題」と呼ばれるもので、この問題があるとKylixのパッケージという機能が正しく動作しない。パッケージはコンポーネントの機能をシェアードオブジェクトとして提供するものだ。Kylix自身が、このパッケージ機能を多用して設計されているため、Kylixを使うためには「ローダの問題」を解決しなければならない。なお、Kylix以外で、この問題が表面化したものは知られていない。

また、Kylixで作成するアプリケーションがパッケージを利用する場合には、アプリケーションの実行環境も、この問題を解決していなければならない。実行環境が問題を解決していることが明らかでない場合は、パッケージを使わないほうが賢明だろう。

なお、この問題の詳細は、ボーランドのWebサイト「Kylix事前テストプログラム（http://www.borland.co.jp/kylix/pretest/）で解説されている。事前テストプログラムは、Kylix自身のCD-ROMにもCのソースコード付きで収録されており、ユーザー自身がテストをする目的でも利用できる。

この問題は、glibc 2.2で解決されているが、glibc 2.1.xを使っているディストリビューションでKylixを使うためには、パッチやアップデートの適用が欠かせない。

幸いなことに、現在では多くのLinuxディストリビュータがglibc 2.2へのアップデートファイルや「ローダの問題」に対応したパッチファイルやアップデートされたglibcを提供している。また、日本語版Kylixには、これらのファイルが収録されている。最新の対応状況については、「Kylixサポート情報」ページ（http://www.borland.co.jp/kylix/devsupport/）で確認してほしい。



画面24　Webアプリケーションの実行

画面23　Webモジュールの作成

IPv6が見えてきた

最近、いちやく注目されるようになった次世代ネットワークプロトコルIPv6。アドレスがたくさん使えるようになることだけはわかったけれど、具体的にどんなものかはさっぱりです。そこで今回、IPv6のココロをやさしく解説。次いで、さらなる好奇心をもって、実際にIPv6ネットワークを体験してみます。

文：國武功一　USAGIプロジェクト
Text : Koichi Kunitake

IPv6とはなにか？

先日、ミネアポリスで行われた第50回IETFミーティングへ行ってきました。IETFとは、Internet Engineering Task Forceの略で、インターネット技術の標準化をボランティアベースで進めている団体です[*1]。皆さんのなかには、RFCやインターネット・ドラフトといった言葉を耳にしたことがある方もいらっしゃるでしょう。これらのドキュメント類は、まさにこのIETFの活動の成果なのです。IETFでは目的別に、百数十におよぶワーキンググループが活動しており、今回もそのなかの数十のワーキンググループが会場内でミーティングを開きました（ふだんの活動はメーリングリスト上で行われています）。

私が目的としていたものは、IPng、NGtransそしてmulti6と呼ばれているワーキンググループです。IPngでは、次世代インターネットプロトコルであるIPv6のコアとなる基本仕様を、NGtransではIPv6の実運用に向けてさまざまな議論が、multi6ではIPv6におけるマルチホーム環境についての問題点などが話し合われています。

IPv6といえば、我らが森首相が去年の9月に所信表明演説で触れて以来、企業から出されるニュースリリースにIPv6の文字が踊ることが多くなったように思います。そのため、IPv6という言葉を聞いたことがある方も多いでしょう。しかし、IPv6っていったいどんなものなの？　といった疑問に答えられる方は少ないのではないでしょうか。この記事では、IPv6とはなんなのか、今後どう広がっていくと予想されているのか、またLinuxでIPv6を使うにはどうすればよいか、といったことについて述べてみたいと思います。

IPv4の限界

皆さんがふだん使っている（使ってますよね？ :-）インターネットは、IPv4（Internet Protocol version 4）というプロトコルの上に構築されています。このプロトコルは、1981年にほぼ現在の仕様が固まりました。それから現在に至るまで、実に約20年にわたってインターネットを支え続けてきたことになります。そのあいだにインターネット利用者は爆発的に増え、当初1000台程度しかインターネットにつながっていなかったコンピュータが、現在では少なく見積もって7000万台以上にもなっているそうです。

最近では、アメリカの株式市場においてIT関連株が下落、将来の先行きに不安が見え始めている今日このごろで

すが、インターネットもそれに引きずられ、その成長を鈍化させてしまうのでしょうか？　答えは否、ではないでしょうか。日本だけを見ても、インターネット接続料金のさらなる低価格化や常時接続の広まりにともなって、ユーザーの増大が予想されていますし、これまで利用者の少なかった中国、東南アジア、南米、アフリカへの普及も容易に想像されます。今まで、IPv4はユーザーの要求に応えてきました。これからも応えていくことができるでしょうか？　残念ながらその先行きは、暗いと言わざるを得ません。それはなぜでしょう。ご存じのとおり、インターネットに接続した端末にはIPアドレスが割り振られます。これは世界中で一意に定まるアドレス（グローバルアドレスと呼ばれています）です。このアドレスの数は有限で、IPv4では32ビット分、つまり約43億個程度にとどまります。この数は、全人類に1つだけアドレスを割り振ったとしても、まったく足りない数です。

NATの登場

この際限ないアドレスの消費を少しでも抑えようと、NAT（Network Address Translation）という技術が発案されました。IPマスカレードとして、おなじみになっているものです。これは、1つのグローバルアドレスを、プライベートアドレスが割り振られている複数のユーザーで共有可能とするものです。ただし、これには双方向性がなくなるという弊害がともないます[*2]。

インターネットの設計思想はend-to-endモデル[*3]が基本になっているため、NAT はインターネットではないという人もいるくらいです。事実、NATはさまざまなアプリケーションに制限を加えており、その発展の妨げになっているといえるでしょう。NATの弊害については、登場時から言われ続けていることですので、将来的になんらかの解決案が提示され、end-to-endモデルを確立できるNAT＋といったものが出てくるかもしれません。しかし、そうしたものがあっても、本来のアドレスの消費問題に対しては、問題を先送りすることしかできないのです。

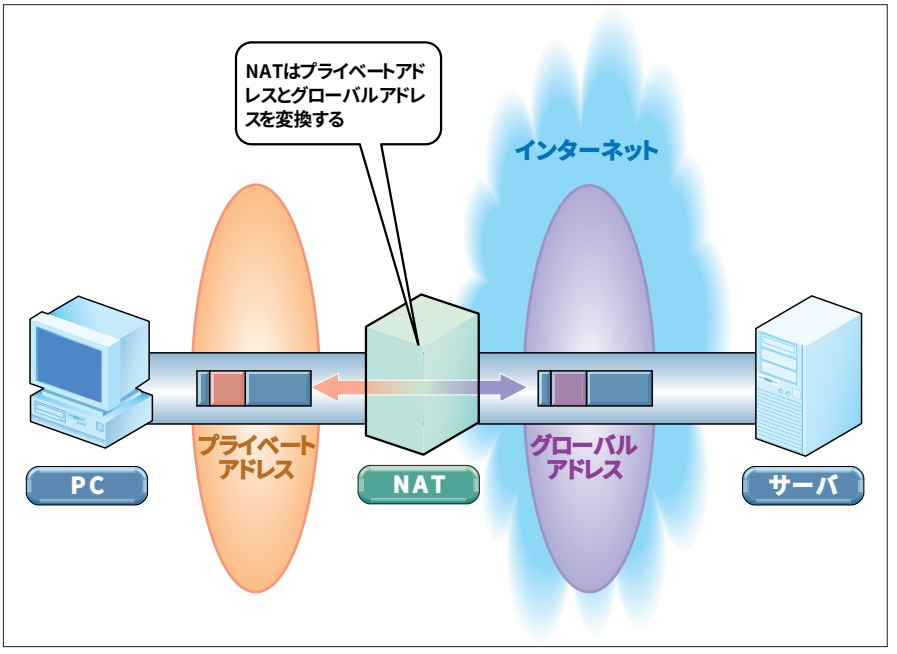


図1　NAT

NATにより[*4]、アドレスの枯渇問題については、いくばくかの時間を稼ぐことができました。しかし、先に挙げたインターネットの普及とは別に、近年ではPDAや携帯電話といった情報端末も、インターネットに接続されるようになっており、アドレスの需要はとどまるところを知りません。結局、いつかはアドレスはなくなってしまうのです。そうして、ついにはインターネットは崩壊してしまうのでしょうか？　もちろん先達たちは、そういった状況に手をこまねいていたわけではありません。その解答のひとつがIPv6（Internet Protocol version 6）なのです[*5]。

次なる飛躍に向けて〜IPv6の登場〜

IPv6はIPv4を完全に置き換えることができるようにと仕様が練られ、さまざまな改良が加えられています。その

Glossary

＊1　インターネット技術のすべてがIETFによって決められているわけではありませんが、基本テクノロジーの大部分の標準化に貢献してきたのもまた事実です。

＊2　このほかにも、NATにはIPヘッダの書き換えがともなうため、IPsecと非常に相性が悪く、基本的にNATとIPsecは同時に使うことができないという欠点があります。

＊3　end-to-endモデルとは、双方からセッションを張りにいくことができる形を指します。NATを使ってしまうと、NATの内側から外のサーバなどへ接続することはできますが、外部からNATの内側にあるホストへ接続を開始することができません。このため、PC-to-PCのインターネット電話を使おうと思った場合、NATの内側にいると、電話をかけることはできても電話をかけてもらうことはできません。お互いにNATの中にいれば、かけることも、かけてもらうことも不可能です。

＊4　NAT以外にも、CIDRやVLSMといった技術がアドレスの枯渇問題や、それにともなう経路情報の肥大化の防止に大きく貢献しましたが、ここでは触れません。

＊5　IPv4の次ならIPv5じゃないの？　と思ってしまいますね。なぜIPv5ではなかったのかというと、ST-II（RFC1819）というプロトコルがすでにバージョン番号5を使用していたためです。

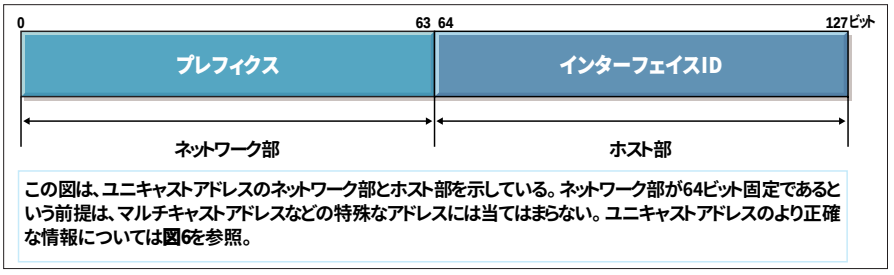


図2　IPv6アドレス

代表的な特徴のうち、いくつかを以下に挙げます。

■128ビットの巨大なアドレス空間

IPv6登場の大きなモチベーションとなったのが、このアドレス問題でした。IPv4では32ビットであったアドレス空間が、IPv6では128ビットに拡張されています。128÷32＝4で、一瞬4倍にしかなってないような気になりますが、当然それは誤りで、単純計算で$2^{(128-32)}$倍ものアドレス空間へと拡張されたことを意味します。イメージが湧きにくいのですが、このアドレス空間が広大なことを伝えるために、地球の陸地に均等にアドレスを振ると……という表現がよく使われます。いくつ割り当てられるかというと、1平方センチメートルあたり、2.2×10^{20}個割り振れるそうです[*6]。ざっと2垓個ずつですね（もっとわかりにくくなりましたか？　垓は兆の1億倍を表します）。なんにせよ、尋常ではないアドレス数です。

■自動設定（Plug and Play）

図2はIPv6アドレスの簡単な構造を示しています。上位64ビットがネットワーク部に、下位64ビットがホスト部になっています。従来はネットマスクが何ビットなのか、設定時に悩む必要がありましたが、IPv6ではネットワーク部が64ビットで固定なので、頭を悩ませる必要はありません[*7]。

それどころか、IPv6ではアドレス数の増加にともなって管理者の負担が増えることが予想されたため、管理者の手を煩わせることなく、誰でも簡単につなげられるようにと、自動設定機能が加えられました。

これは、接続先のネットワークにルータがあれば、そこからネットワークアドレス（プレフィクスと呼ばれます）が通知され、ホストは自身のネットワークカードのMACアドレスを元にインターフェイスIDを生成、この2つからアドレスを自動生成します。ルータがない場合でも、そのセグメントでのみ有効なリンクローカルアドレスというものが自動生成されますので、範囲は限定されますが、通信は可能です。

ここでIPv6で使われるアドレスについて、すこし補足をしておきます。IPv6ではIPv4とは違い、1つのインターフェイスにアドレスをいくつでもつけられます。また使用目的に応じて、

- **ユニキャストアドレス（unicast address）**
- **マルチキャストアドレス（multicast address）**
- **エニーキャストアドレス（anycast address）**

の3種類のアドレスがあります。

ユニキャストアドレスとは1対1の通

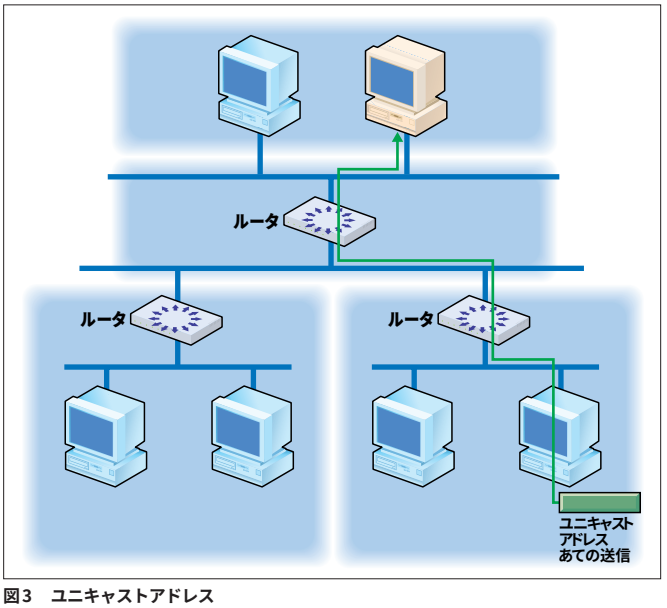


図3　ユニキャストアドレス

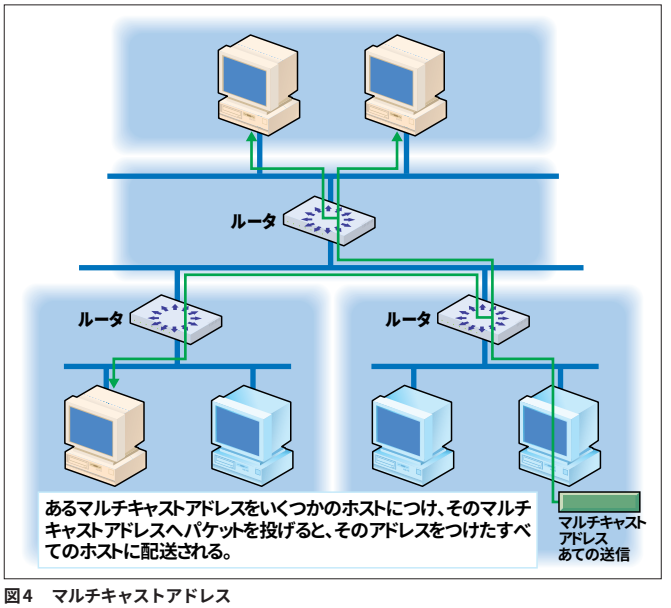


図4　マルチキャストアドレス

信に使われるもので、ふだんは主にこのアドレスが使われます（**図3**）。

マルチキャストアドレスとはあるグループを表すアドレスで、あるマルチキャストアドレス宛てにパケットを投げると、そのグループに属するすべてのインターフェイスにパケットが届けられます（**図4**）。

エニーキャストアドレスとはマルチキャストアドレスと同様、あるグループを表すアドレスではありますが、グループに属するすべてのインターフェイスに配送されるわけではなく、どれか1つに配送されると、それ以上は配送されなくなります（**図5**）。

■IPsec標準装備

インターネットの商用利用が広まった現在、通信に対するセキュリティの需要が高まっています。これを受けて、IPv6では、IPsecと呼ばれるセキュリティ機能を標準で装備しなくてはならないとされています。IPsecはIPレイヤで暗号化をするため、上位レイヤで動くさまざまなアプリケーションになんの変更も加えず、通信の秘密を守ることが可能となります[*8]。

IPv6でバラ色？ ～その幻想と欠点～

これでIPv6が広まれば、素晴らしい未来が開けるんです！　と言い切ってしまいたいところなんですが、これまたそう単純にはいきません。長い時間をかけ、検討に検討を重ねて決められた仕様ではあるのですが、なかなかすべてを満足させる完璧な仕様というものはできないようです。この節では、IPv6で誤解されやすいところや、その欠点について触れたいと思います。

■潤沢なアドレス空間？

IPv6では、とてもじゃないけれど使い切れないほどのアドレスが手に入る！　と喧伝されていますが、それは本当でしょうか？　一度、一緒に検証してみましょう。

IPv4では約43億個のアドレスがあるとされていましたが、少しでもネットワークをかじったことがある方ならおわかりのとおり、効率よくすべてのアドレスを使いきるなんてことは不可能です。IPv6も同様です。IPv6アドレスは**図2**で示したとおり、ネットワーク部とホスト部が64ビットずつに分けられています。しかし、1つのセグメントに2^{64}個ものホストをぶら下げるなんてことは現実にはまず考えられませんので、1セグメントに限界を超えるようなホスト群をぶら下げたとしても、十分なだけのアドレスが用意できることはわかります。けれどもそれと同時に、かなりの数のアドレスを無駄にしていることをも意味します。

ちょっと心配になってきましたか？　では話を先に進めましょう。結局私たちが必要とするアドレスがあるかどうかは、左のネットワークアドレス部が十分かどうかを検証すればよいことになります。次に**図6**を見てください。

IPv6では、経路の爆発的な増加という過去の反省を踏まえ、アドレスは集約可能な形で階層化して、配布されます（**図7**）。TLAはNLA組織にアドレスを割り当てます。NLA組織はSLA IDを使って自組織のネットワークを作ることができます（TLA組織の方針によってはさらにNLAを階層化し、NLA1、NLA2組織などに細分化することも可能です）。というと、ピンとき

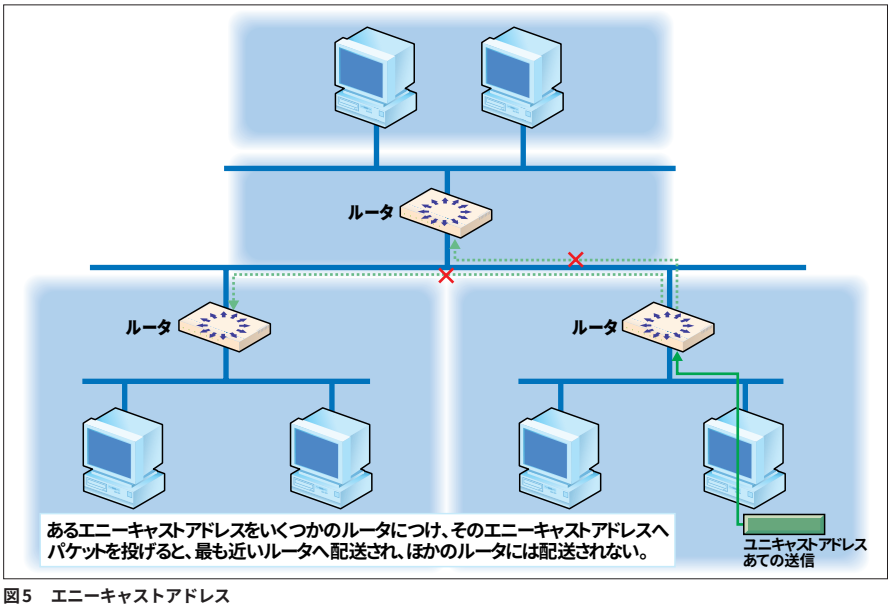


図5　エニーキャストアドレス

Glossary

＊6　WIDEプロジェクトが提供している「IPv6に関するFAQ」より引用しました。これは、WIDEプロジェクトのWebサイトに掲載されています。
http://faq.v6.wide.ad.jp/index-j.html

＊7　ネットワーク設計においては、従来と同じように、どこにどれだけのネットワーク空間を割り当てるかということに注意を払う必要があります（IPv4の場合と同様に、“/”を使って、/48といった表現をします）。

＊8　通信の中身については暗号化され、秘密は守られますが、通信していること自体は基本的に隠蔽できません。

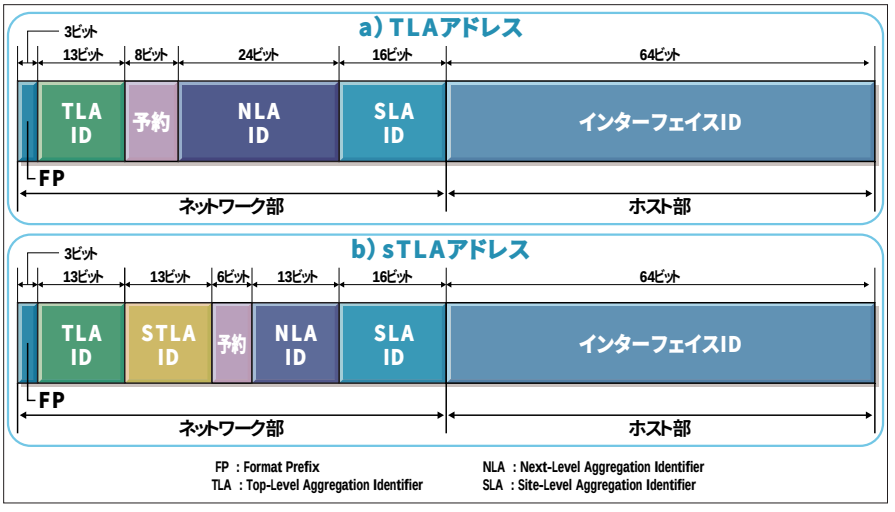


図6 TLAアドレスとsTLAアドレス

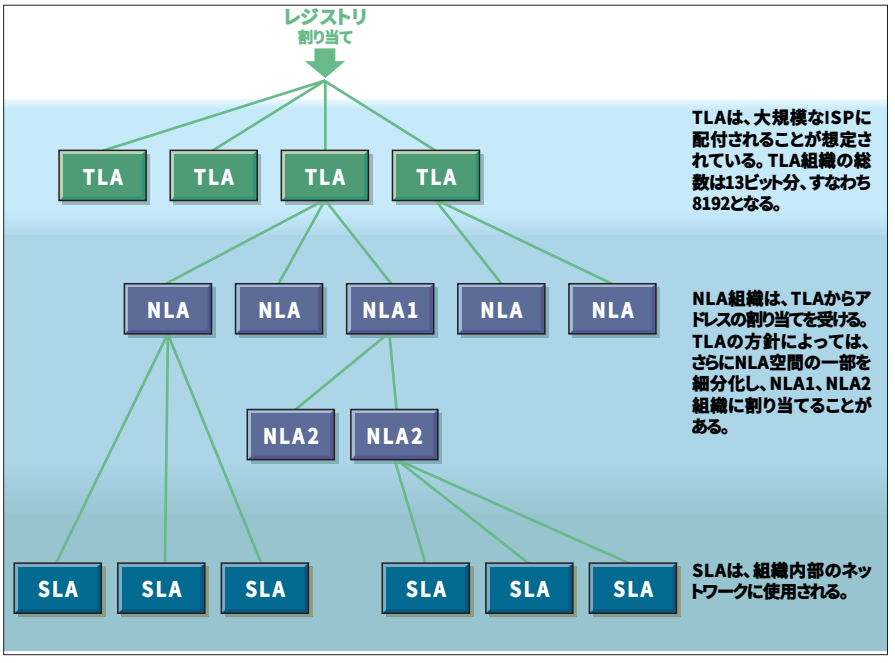


図7：階層化されるアドレスの割り当て

ませんね。実際の運用では、TLAは大手ISPなどに割り当てられ、ISPは自分の顧客にNLAを配布し、顧客はSLAを使って自分のネットワークを構築します。NLA組織がSLAをほかの組織に再配布することは禁止されています。これは、アドレスの割り当て状況が追えなくなることを防ぐためです。

さて、現在商用アドレスとしてISPなどに配付されているのは、TLAではなくsTLA（sub TLA）アドレスです（**図7**）。ここでは、sTLAについて考察していきましょう。

この場合にも、SLAには16ビットの空間が割り当てられます。これは、65536個のネットワークを構築できることを意味します。大企業と呼ばれる組織でもこれだけあれば十分でしょう。一方、sTLA組織に割り当てられるNLA空間は13ビット、これは8192個です。大手ISPになると、顧客を数万社かかえていることも珍しくありませんから、足りませんね。不足すれば、sTLA IDを新たに取得することも可能なのですが、sTLA空間自体が同じ13ビットで8192個しかありません。これでは、sTLA ID自体も早々に喰い潰されることは想像に難くありません。

■自動設定（Plug and Play）の落とし穴

IPv6は、ケーブルを差すだけですぐにネットワークが使えることになっています。これは間違ってはいませんが、また同時にすべてを表してもいません。IPv6の自動設定は、あくまでアドレスの設定を自動的に行ってくれるだけです（もちろん、どこにパケットを投げればよいかというルーティングの設定もしてくれますが）。「それで十分じゃないの？」と一瞬思ってしまうんですが、ちょっと待ってください。DNSの設定はどうするのでしょうか？

「www.linux.or.jp」とブラウザに打ち込んで日本のLinux情報が読めるのも、ブラウザが指定されたDNSサーバを使って、ドメイン名からIPアドレスへ変換し、そのアドレスへアクセスするということを裏でやっているからこそです。IPv6の機能だけでは、ネットワークにケーブルを差しただけでDNSサーバを知るすべはありません。このままではまともに通信できません。もちろん128桁のIPv6アドレスを暗記しているのであれば別でしょうが、それは非現実的です。

さらに自動設定で問題となりそうなのは、そのアドレスの作り方です。自動設定で作られるアドレスは、ルータから通知されるネットワークアドレス（プレフィクスと呼ばれます）と、NICが持っているMACアドレスから生成したインターフェイスIDとを合成して作られます。MACアドレスは、全世界で一意に定まるという建前ですから、

どんなプレフィクスが来たとしても、後ろ64ビットさえ見れば、そのアドレスから使用者が特定される可能性が出てきます。これはあまり歓迎されることではないでしょう。

■IPsecへの幻想

IPv6はIPsecを標準で搭載しているから、すべての通信は暗号化されていてセキュリティは万全！　と思ってはいませんか？　残念ながら、その認識は誤っています。それはなぜでしょうか。IPsecは必須機能として挙げられていますが、通信すべてにIPsecを使うべし……とは定められていません。ですので、IPsecを使う場合には、ユーザーがあらかじめ明示的に設定しておくことが必要です。

それでもIPv6！

と、ちょっと皆さんを不安にさせましたが、これらについては、すでに解決案が検討されていたり、もしくはすでに解決済みだったりします。

■アドレスの数について

sTLA IDが早々に売り切れそう……というのは、実は意図的なものです。IPv6が本格的に使われるようになると、**図6**のa）に示した形で、アドレスが配布されることになっています。TLA組織の数は8192とそのままですが、それに束ねられるNLA組織のアドレス空間が24ビットもあるため、まずTLA IDを複数取得しようとする組織は現れないでしょう。もし万が一TLA IDが売り切れることになったとしても、FPごとに8192個のTLAを設けることが可能なので、TLA組織自体を将来的に増やすことも可能です。

なぜわざと小さなsTLA空間を配ることにしたかというと、IPv4への反省があったためです。IPv4では、当初クラスAやクラスBを安易に配ったため、後々アドレスが足りなくなり、非常に困った状況に追い込まれました。このため、IPv6ではいきなり莫大なアドレス空間を保有するTLAを配らず、まずは比較的小さなアドレス空間でアドレスを割り当てることにしたのです。

これらはIPv6の仕様というよりも、割り当てのポリシーに起因する問題です。将来的にsTLAからTLAへと本格的に運用されれば、各TLA組織は24ビットものNLA IDを手に入れられます。まず、アドレス不足に悩まされることは考えられません。なにより、IPv6には、アドレスの付け替えが比較的簡単にできる仕組みがあるので、いざアドレスが足りないとなれば、実情に合うようにと、すべてのアドレスを振り直すことも不可能ではありません。また、求められればリナンバリングを行う必要があると定められています（あまりやりたい作業ではありませんが）。

ただ、あまり無計画にアドレスを割り振ると、内部で捌かなければならない経路が爆発的に増えることも考えられるので、ネットワーク管理者は留意しなくてはいけません。

■自動設定におけるDNS問題

DNSサーバをどうやってホストへ知らせるかといった問題も、いくつか検討されています。ひとつはエニーキャストアドレスを使う方法です。エニーキャストアドレスを、いったいどういう場面で使うのだろうと思われたかもしれませんが、これを使うことにより、ユーザーに適切なDNSサーバを知らせることが可能となります。別の方法として、従来のようにDHCPを使って知らせる方法があります。もちろんアドレスもDHCPを使って設定することができますから、よりきめ細かいコントロールをする場合には、今後もDHCPが活躍することとなるでしょう（IPv6で使われるDHCPはDHCPv6と呼ばれています）。

また、アドレスの下位64ビットから個人が特定されるのではないか、という懸念ですが、こちらはすでに解決策が提示されています。思い出してください、ホスト部にはとうてい使いきれないだけのアドレス空間があることを。これを逆手にとって、一定時間ごとにランダムなインターフェイスIDを作成、これを元に新しいアドレスへ変更

Column

経路情報と集約

ルータは、受け取ったパケットを適切な宛先に届けるために、内部に経路情報を持っています。経路情報は、宛先のネットワークアドレスとそのパケットの次の転送先となるアドレスを一組にしてリストしたもので、ルータは経路情報のなかからパケットの宛先にマッチするものを探して転送するわけです。

IPv4では、利用者の増大とともにバックボーンルータに蓄えられる経路情報の数が爆発的に増え、運用を圧迫するようになりました。これらの問題に対処するため、アドレスを実際の接続体系に沿って割り当てるなどの方法で経路の集約（複数の経路を1つに見せるものだと考えてください）が実施されました。しかし、かつて配付したアドレスが現存しているため、効率的な集約はできていないのです。

するといった仕組みがあります（もちろんランダムに生成されたアドレスですので、すでに使われていないか使用前にチェックする必要があります）。

■IPsecへの取り組み

結局のところ、IPsecとはセキュリティツールです。セキュリティツールを使うために、セキュリティポリシーが必要なのは仕方のないことです[*9]。しかしながら、機能が豊富なIPsecにユーザーは困惑するでしょう。そこで、推奨設定なるものをあらかじめ決めておいて、ユーザーの負担を小さくしようという動きがあります。また仕様上、繁雑な作業となる暗号の鍵交換については、IKE（The Internet Key Exchange）を介さない簡便な方法が提供できないか検討されています。IPsecが広まるにつれ、これらの問題は徐々に解決してゆくことでしょう。

このほかにも、IPv6で考えうる問題点はあります。しかしその問題提起の多くは、実運用を睨んだものです。これは、すでにIPv6の開発はほぼ終わり、展開と運用をどのようにしていくかというステージへ移行しつつあると捉えることができます。それが冒頭に挙げたような企業のニュースリリースラッシュへとつながっているのでしょう。

IPv6その足音　～開発から展開へ～

実際のIPv6の実験運用は、日本やヨーロッパが中心となって先行してきた感があります。しかし近年、インターネットの普及という点において日本を大きくリードしている韓国でも、IPv6は注目されているようです。インターネットの発展がめざましい国ほど、アドレス枯渇問題も顕著であり、IPv6に対する注目度が高いようです。さらに、もともと多くのIPv4アドレスを持っていたため、IPv6にはあまり積極的ではないと言われていたアメリカでさえ、最近はIPv6のトラフィックを交換可能なIX（Internet eXchange）が複数登場しています。ルータベンダーも、最近立て続けにIPv6の対応を宣言していますし、インターネットを支えるバックボーンネットワークに関しては、着実にIPv6を受け入れる体制が整いつつあるといってよいでしょう。また、実際にそのネットワークを利用するクライアントについてみると、次世代の携帯電話IMT-2000の標準規格の策定を進めている3GPPもIPv6の採用には前向きです。Microsoftも、次のWindows XPでは標準でIPv6スタックがついてくるようです（もしかしたら、サービスパックかもしれませんが）。

もうひとつ、面白いIPv6のクライアントとして、自動車（IPCar）を紹介しておきましょう。これは、自動車をインターネットに接続し、各車から集めたデータを元にして渋滞情報や天候といった情報を提供してやろうというものです。これまでに横浜で実証試験が行われましたが、今度は舞台を名古屋に変え、より大規模な2000台で実験が行われます。次の実験では、IPv6の導入が決定したようです。

LinuxでIPv6を使ってみよう！

IPv6は、ふだん使っているパソコンはもちろんのこと、これまでに紹介したように、携帯電話、家電、自動車など、さまざまなものに載せられることが期待されています。振り返って我らがLinuxでの対応はどうなっているのでしょうか？　LinuxのIPv6への対応は意外と歴史が古く、カーネル2.1.xのころから対応をうたっています。とい

Column

IPCar

自動車のGPSや各種センサーから得た情報をネットワークを介して収集、交通情報や路面情報、天気情報などに役立てようという試み。横浜での実験では、タクシーやバスなど約300台が参加しました。

運転席の下の車載システム。集めたデータはDoPaを使って送られる。

天候を表示している例。ワイパーが動作している車が青く表示され、雨が降っていることがわかる。

うことは、Linuxを使ってれば、いつIPv6の世の中になっても大丈夫！　とは、残念ながらいきません。実は、LinuxのIPv6の実装は、IPv6の仕様が出始めたころにPedro Roque氏が書いたものです。IPv6の仕様は、長期にわたって検討を重ねるあいだに、仕様が変更された部分も出てきました。LinuxのIPv6の実装がほとんどメンテナンスされていなかったこともあって、現在ではRFCに準拠しない古い仕様になってしまっています。必須とされるIPsecも実装されていません（すくなくともカーネルと一緒には配布されていません）。また、glibcのIPv6対応も、お世辞にも良いものとはいえませんでした。

USAGIプロジェクトの誕生

ほかのプラットホームが次々とIPv6の対応を進めるなか、Linuxのカーネルおよびライブラリの改善は遅々として進みませんでした。このままでは将来、Linuxが取り残されてしまう！　そういった危機感を募らせたLinux IPv6 Users Group JPの有志たちは、カーネルとライブラリを改善しようと活動を始めました。しかしながら、ユーザーグループとしての活動に限界を感じ、BSD系のKAMEプロジェクト[*10]の開発形態を参考にしてUSAGI（UniverSAl playGround for Ipv6）プロジェクト[*11]を開始したのです。

昨年、このプロジェクトがさまざまなメーリングリストを通じて開発者を募集しました。その結果、基本的にフルタイムでIPv6に情熱を注げる人をコアメンバーに、仕事の関係で難しい方にもプロジェクトメンバーとなっていただき、前者はカーネルおよびライブラリを、後者はおもにアプリケーションのIPv6対応や、各ディストリビューションへのフィードバックを進めています。

USAGIで何が良くなるの？

では、このプロジェクトはいったいLinuxの何を改善するというのでしょうか。「そもそもLinuxのIPv6対応って、本当にあなたが言うほど悪いの？」と疑問に思われる方もいらっしゃるでしょう。ということで、次のページをご覧ください。

```
http://www.linux-ipv6.org/
tahitest/summary-index.html
```

これは、IPv6の実装を高品質なものにするために活動しているTAHIプロジェクト[*12]の協力を得て、どれだけ正しく仕様に沿っているかを、オリジナルのカーネルとUSAGIプロジェクトで改良を加えたものとで比較したものです。すべてはこれが物語っていると思います。

USAGIを走らせる

次に実際にUSAGIを使ってLinuxをIPv6化してみましょう。えっ？　IPv4が使えなくなると困るから嫌だって？　大丈夫です。すべてのネットワークが一気にIPv6に移行すると考えるのは現実的ではありません。そこで、IPv4とIPv6が共存できるように、移行時期にはIPv6を話すノードはIPv4も話せなくちゃダメ、ということになっています。ですので、手持ちのLinuxをIPv6に対応にさせたからといって、IPv4が話せなくなるわけではありません。また、家庭内LANでIPv6を使う際に、特別な機材も必要ありません。普通のハブが使えます。これは同じLAN上に、AppleTalkやIPX、それにIPv4など、異なるプロトコルが共存できていることを考えれば、不思議なことではないでしょう。共存できない特殊なハブもないわけではありませんが、ふだんお使いのハブやスイッチングハブでは、まずそんなことはないでしょう。

さて、USAGIプロジェクトの成果は、カーネル＋ライブラリ＋アプリケーションをすべて1つの圧縮ファイルにまとめて配布されています（圧縮されてはいますが、ファイルサイズが大きいので気合いを入れて落としてください）。立ち読みをせず、ちゃんとこの雑誌を買ってくれた人は、本誌の付録CD-ROMにひととおり収録されていますので、そちらをどうぞ。一度USAGI kitを手に入れれば、CVSを利用して常に最新のUSAGI kitへと更新できますので、ぜひそちらもお試しください。

このUSAGI kitは、USAGIプロジェクトのFTPサーバからも手に入れることができます。

```
ftp://ftp.linux-ipv6.org/pub/
usagi/snap/kit/
```

こちらには、カーネル2.2.x系とカー

Glossary

＊9　NATは簡易ファイアウォール代わりになるという人もいます。が、それを鵜のみにして、それを目的にNATを導入することは、お奨めできません。セキュリティがほしいのであれば、ネットワーク管理者は必ずセキュリティポリシーを定め、それを満足する技術を導入すべきです。

＊10　KAMEプロジェクトとは知る人ぞ知る、IPv6では有名なプロジェクトです。このプロジェクトでは、すべてのIPv6の実装の参照コードとなるべく精力的に活動し、BSD系のOSにIPv6のスタックを提供しています。詳細はhttp://www.kame.net/をご覧ください。

＊11　KAMEに対してUSAGIとは、ちょっと不吉な名前じゃないかって？　一番にゴールできなくてもいいんです。遅れても着実にゴールすることが一番大切なことだとは思いませんか？ :-)

＊12　TAHIプロジェクトの詳細については、http://www.tahi.org/をご覧ください。

ネル2.4.x系が用意されています。カーネル2.4.x系を使用する際には、modutilsなどのバージョンアップが必要となる場合がありますので、ご注意ください。

カーネル2.2.x系：

```
usagi-20010402-linux22.tar.bz2
```

カーネル2.4.x系：

```
usagi-20010402-linux24.tar.bz2
```

次に落としてきたファイルを展開します。ディスクに十分な空き容量があることを確認しておいてください。ソースの展開とコンパイルに必要なディスク容量は180Mバイトを超えます。

```
$ bzip2 -cd usagi-2001XXXX.tar.bz2
| tar xvf -
$ cd usagi
$ make prepare
```

最後のmake prepareは忘れずに実行してください。実行しないとコンパイルできません。次にカーネルのコンパイルです。ディストリビューションによっては独自のパッチがカーネルに当てられていることがありますので、ご注意ください。場合によっては独自パッチを当てないと起動できないものもあるようです（ReiserFSなど）。

```
$ cd kernel
$ cd linux22　（カーネル2.4系なら“cd
linux24”）
$ make mrproper
$ make menuconfig
```

ここで、カーネルのオプションを選択します。“ make xconfig ”のほうが慣れているかもしれませんが、このあたりは各自の好みで。カーネルをIPv6に対応させるには、すくなくとも**画面1**の**a**）に挙げた2つの項目をチェックする必要があります。

さらに、USAGIでの新機能を試したい場合の推奨コンフィグは、リスト1のb）のとおりです。ちなみにこれはカーネル2.2.xの場合です。カーネル2.4.xをお使いの場合は、同じく展開されたディレクトリ内の、doc/CONFIG.linux24をご覧ください。

いよいよコンパイル＆インストールです。

```
$ make dep
$ make bzImage
$ make modules
$ su
# make install; make modules_install
```

あとはいつものとおり、ブートローダにliloを使っている方は/etc/lilo.confを編集したあと、/sbin/liloを実行すれば、OKです。では新しいカーネルを立ち上げてみましょう。無事に立ち上がりましたか？　さっそくIPv6のアドレスがついているかどうか確認してみましょう。

```
$ /sbin/ifconfig -a
```

a)これは必須

```
  Code maturity level options  --->
   [*] Prompt for development and/or incomplete code/drivers

  Networking options  --->
   <*>   The IPv6 protocol (EXPERIMENTAL)
```

b)推奨するオプション

```
  Networking options  --->
   [*]     IPv6: Loose scope_id
   [*]     IPv6: drop packets with fake ipv4-mapped address(es)
   [*]     IPv6: ignore too small Valid Lifetime for Address
Autoconfiguration
   [*]     IPv6: allow default route when forwarding is enabled
```

画面1　選択するカーネルオプション

この結果を**画面2**に用意しましたが、この行の意味はなんなんだ？　と思われる方がほとんどではないでしょうか。ここで、簡単に説明することにしましょう。この行の真ん中に位置する16進数の並びが、今までお話ししていたIPv6アドレスなのです。

IPv4のアドレスは、10進数で表した数字を“ . ”で区切って、たとえば“ 192.168.0.1 ”のように表記してきました。IPv6ではアドレスが128ビットと長いこともあり、次のように16ビットごとに“ : ”で区切って表記します。

```
xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:
xxxx　（xは16進数）
```

では、さきほどの例を見ましょう。

```
fe80::2d0:b7ff:fea0:beea
```

ん？　なんだか短いですね。Linuxのバグ……ではありません。IPv6では、連続する0を1カ所だけ“ : : ”を使って省略できます。ですので、上記の例を“ : : ”を使わず書くと、

```
fe80:0000:0000:0000:2d0:b7ff:fea0:beea
```

となります。えっ？　“ 2d0 ”って

桁が足りないんじゃないかって？　目ざといですね。ご指摘のとおり、4桁のなかで最初に0がきた場合は、0が省略できます。ですので、いっさい省略せずに書くと……

```
fe80:0000:0000:0000:02d0:b7ff:fea0:beea
```

だいぶ長くなりましたね。逆にいえば、IPv6では長すぎるアドレスをいかに省略して書くかに心を砕いていることがわかります。では、次の場合はどう省略できるのでしょうか？

```
xxxx:0000:0000:0000:xxx0:0000:0000:000x
```

答は、

```
xxxx::xxx0:0:0:x
```

または、

```
xxxx:0:0:0:xxx0::x
```

です（前者のほうが短くていいですね）。

```
xxxx::xxx0::x
```

じゃないのか？　と思われる方もいらっしゃるでしょうが、2カ所で“：：”を使ってしまうと、それぞれの“：：”にどれだけの0が連続していたかが、判別できなくなってしまいます。

みなさんのマシンにも“inet6 addr”の項目は表示されましたか？　もし表示されていたら、おめでとうございます！　これであなたのLinux BoxもIPv6対応となりました。もし表示されなかったら、本当に新しく作り直したカーネルが立ち上がったのかどうか確かめたり、ifconfigのバージョンアップを試してみてください。

```
$ /sbin/ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:D0:B7:A0:BE:EA
          inet addr:192.168.0.1  Bcast:192.168.1.255
Mask:255.255.255.0
          inet6 addr: fe80::2d0:b7ff:fea0:beea/10 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:4575 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3275 errors:0 dropped:0 overruns:0 carrier:0
          collisions:496 txqueuelen:100
          Interrupt:9
```

画面2　ifconfigでネットワーク情報を見る

アプリもね

と、せっかくカーネルがIPv6化しても、このままではなにもできません。アプリケーションもIPv6に対応させたものを入れる必要があります。しかしご安心を。USAGI kitには、表1に挙げたIPv6化したアプリケーションも一緒についてきます。次のステップとして、このIPv6対応アプリケーションをインストールしてみましょう。

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

簡単ですね？ デフォルトではすべて/usr/local/v6以下にインストールされるようになっています。自分自身にpingを打ってみましょう。**画面3**のように表示されていれば成功です。ほかにもいろいろと試してみてください。

IPv6ネットワークへつないでみよう！

せっかくカーネルもアプリケーションもIPv6に対応させたんだから、次に、実際にIPv6ネットワークへと接続してみましょう。最近はいくつかのISPがIPv6の接続サービスを提供しています。とはいえ、現状ではダイヤルアップでのIPv6接続サービスを提供しているISPはほとんどなく、専用線接続が主です。さすがにそれは敷居が高いので、別の手段を検討しましょう。その手段とはトンネルによる接続です。どういうものかというと、IPv6のパケットをIPv4パケットで包んで配送するという方法です。

トンネルというだけあって、その両端にIPv6パケットをIPv4パケットでカプセル化する端末が必要となります（**図8**）。IPv6パケットはトンネルの始点でカプセル化され、トンネルの終点で再びIPv6パケットへと戻されます。一般ユーザーに、このトンネルサービスを提供してくれる対向側の端末を、トンネルブローカと呼びます。私たちは、このトンネルブローカとトンネルを張ることによって、IPv4の接続先を

finger、fingerd
ftp、ftpd
inetd
ping6
radvd
rcp、rexec、rexecd
rlogin、rlogind
rsh、rshd
rwhod
talk、talkd
telnet、telnetd
tftp、tfptd
tracepath6、traceroute6
write、writed
v6p

表1　IPv6対応アプリケーション

介してIPv6ネットワークに接続することができます。しかし、日本のISPでもトンネルによる接続を提供するところは増えていますが、固定のグローバルIPv4アドレスを必要とすることがほとんどです。ダイヤルアップによる接続では固定IPアドレスを割り振られることは少ないので、これらのサービスを受けるのもまた現状では難しいといえます。結局あきらめるしかないのでしょうか？　あきらめるのはまだ早すぎます。実は、FreeNet6というサイトがフリーで6boneと呼ばれるIPv6実験ネットワークへのトンネルによる接続サービスを提供しています。

```
http://www.freenet6.net/
```

さっそく試してみましょう。ここから［Get your own IPv6 tunnel］をクリック、次にLinux Debian GNUから［Registration Form］へと進んでください。Red Hatなどを使っている方も同様です。次に簡単なフォームを埋め、［Submit］を押すと、トンネルを張るための設定が書かれたPerlスクリプトがダウンロードされます。Debianをお使いの方は、これを実行するだけでIPv6ネットワークへのトンネルが張られます。ほかのディストリビューションを使っている人も、このスクリプトを参考にすれば、比較的容易にトンネルを張ることができるでしょう。

どうでしょうか、ちゃんと設定できましたか？　おめでとうございます。これでIPv6ネットワークに接続されました！　しかし、本当に接続されたのかよくわかりませんね。実際になにかアプリケーションを動かして確かめてみましょう。ブラウザはNetscape6をお使いでしょうか？　このブラウザはIPv6に対応しているのです。このブラウザを使って、先ほど紹介したKAMEプロジェクトのサイトへアクセスしてみてください。

```
http://www.kame.net/
```

実はこのサイトには仕掛けがしてあって、IPv4を使ってアクセスしてきたときとIPv6でアクセスしたときとでは、マスコットのカメに変化があります。いったいどんな変化があるのか？　実際にアクセスして、自分の目で確認してみてください。残念ながら、USAGIプロジェクトのサイト（http://www.linux-ipv6.org/）にはこういった仕掛けはありません。USAGIプロジェクトのマスコットが決まったら、ぜひなんらかの仕掛けを作ってみたいですね。

```
$ /usr/local/bin/ping6 -c 2 ff02::1%eth0
PING ip6-allnodes(ff02::1%eth0) from fe80::2d0:b7ff:fea0:beea%eth0 :
56 data bytes
64 bytes from ::1: icmp_seq=0 hops=64 time=65.000 usec
64 bytes from ::1: icmp_seq=1 hops=64 time=20.000 usec
```

画面3

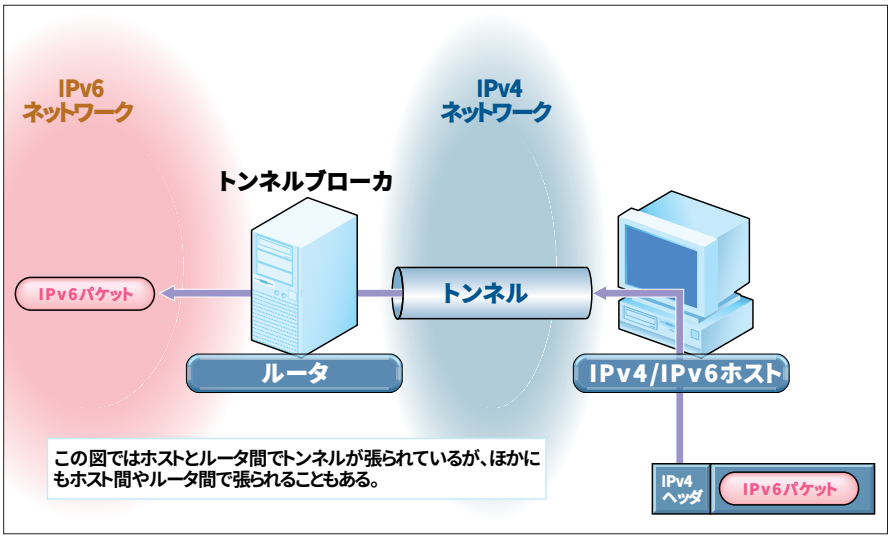


図8　トンネル接続とトンネルブローカ

最後に

IPv6の基礎となるアドレス体系の話などは、あえて大きく取り扱わず、ざっと、「IPv6ってこんなもんなんだよ」という話をしました。実際に活用していくうえで必要な基礎知識は、回をあらためてまたお話するので、ちょっとお待ちください。それでも、少しでもIPv6に興味を持っていただくきっかけになれば幸いです。といった感じで、長々と書いてきましたが、結局我々Linuxユーザーに必要なのは、こんな話ではなく、次の一言なのかもしれません。「IPv6って新しい技術があるんだけど、使いたくならない？」

新技術＝正義

なのです？！

では、次なる展開を見せるインターネットの海へと、みんなで飛び込んでゆくことにしましょう！　もちろんLinuxと一緒にね :-)

短期集中連載

Linuxでテレビ録画に挑戦！

1万円で作るテレビ自動録画サーバ

第3回　電子番組表を使ったメール録画予約

今回は予定を変更して、電子メールを利用した遠隔地からの録画予約に挑戦する。テレビ録画機能付きのWindows PCで利用できる「電子番組表」を、Linuxマシンでも使えるようにするのだ。利用するのは「ON TV JAPAN」（http://www.ontvjapan.com/）の電子番組表だ。これに対応すれば、iモード携帯電話からの録画予約もできるようになる。

文：竹田善太郎
Text : Zentaro Takeda

前回は、「tvmaster」スクリプトを中心とした録画予約のしくみを完成させ、Webを使った簡単なユーザーインターフェイスを追加してみた。できるだけ単純な構成にしてあるので、さまざまな機能を追加することが、比較的簡単にできるのがわかってもらえただろうか。

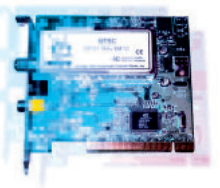
前回の予告では、今回はキャプチャするビデオデータを圧縮して、より高画質の録画データを大量にキャプチャできるようにする予定だったが、圧縮用のツールの選定やインストールの方法を絞りきることができなかったため、やむなく、予定を変更させていただく。

いいわけがてら、ビデオ圧縮ツールの現状を簡単に述べたいと思う。本連載のテレビ録画システムでは、「mp1e」というビデオデータ圧縮ツールを使うつもりでいる。このツールは、bttvドライバから直接キャプチャデータ（非圧縮）を受け取って、それをリアルタイムにMPEG-1形式に圧縮する。bttvドライバやVideo 4 Linuxとは密接な関係を持ち、これらのドライバ類を直接制御するので、ドライバのバージョンの違いに、かなり敏感である。

最新版のmp1e（バージョン1.8.1）は、カーネル2.2系にバンドルされている古いbttvドライバでは動作しない。また、bttvドライバを最新のものに取り替えようとすると、今度は、カーネル2.3以降が必要になる。ところが、原稿執筆時点で流通しているLinuxディストリビューションは、一部ベータ版を除いて、カーネル2.2系のものがほとんどだ。

カーネルのバージョンアップそのものは、本誌でもたびたび掲載されているように、それほど難しい作業ではない。しかし、実際にカーネルをバージョンアップしてみると、さまざまなコマンド類やドライバ類との「相性」の問題が生じて、システム全体の安定性が著しく落ちることになる。特に、最近の「肥大化」したディストリビューションでは、このような問題がよく起こるので、「新カーネルにも完全対応！」と明記されているディストリビューションを除くと、仕事に使っているような実用Linuxマシンにおいて、素人がカーネルのバージョンアップを行うのは、事実上不可能といってよいだろう。

本連載のテレビ録画システムは「シンプルであること」を至上の課題にしているので、カーネルのバージョンアップというやっかいな作業は、できれば避けたい。また、カーネル2.4を搭載したディストリビューションの登場も近いので、ビデオ圧縮については、カーネル2.4搭載の各ディストリビューションの動向がはっきりしてから、改めて挑戦することにしたい。



画面1　ON TV JAPANのホームページ（http://www.ontvjapan.com/）

画面2　「MAIL予約」は電子番組表から直接できる

電子番組表とは？

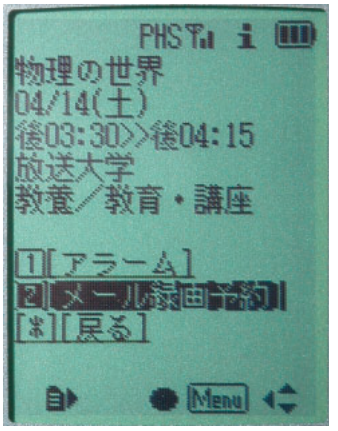
最近の「機能てんこ盛り」の家電系メーカー製デスクトップPCでは、「電子番組表」を使ってテレビの録画予約が簡単にできることを売り文句にしている製品が多い。白状すると、このような「イロモノPC」にはまったく興味がなかったので、店頭でもまるっきり無視していたし、ましてや購入したことなどない。このため、電子番組表がどのような使い心地なのか、実用に耐えるのかどうかはわからなかったのだが、要するに、PCの画面上にテレビの番組表を表示して、その上で録画したい番組名をクリックすると、自動的に録画予約が完成するというものらしい。

番組表そのものは、電子化された番組データをダウンロードして、PC上の専用プログラムに読み込ませて表示するもの、あるいはWebサーバ上に番組データを用意して、PCのWebブラウザで表示できるようにするもの、などの方式が考えられるが、最近ではWebブラウザで表示できる番組表が主流になっているようだ。このような、パソコンで扱えるテレビの番組表のことを「電子番組表」や「EPG（Electronic Program Guide）」などと総称している。

このような電子番組表は、現在ではいくつかの会社が提供している。ユーザーの視聴環境（地上波放送、CATV、衛星放送）の違いや、ユーザーの好み（好きな番組のジャンルなど）に合わせて、カスタマイズされた番組表を表示できるようにするなど、各社でさまざまなサービスの工夫をしている。その中で、今回利用するのが、イサオが運営するインターネット電子番組表サービス「ON TV JAPAN」（**画面1**）だ。

ON TV JAPANでは、Webページとして表示された番組表から「iEPG」形式の予約データをローカルマシンにダウンロードしたり、「ネットリモコン」形式の予約データをあらかじめ登録したアドレス宛てに電子メールで送信したりすることで、PCのテレビ録画予約を行えるようになっている（**画面2**）。iモード携帯電話からの予約も可能なので（**画面3**）、外出先でも手軽に録画予約ができるようになる。

電子番組表の形式としては、日本ではiEPGのほかに「ADAMS-EPG」という方式も使われているが、こちらは「地上波データ放送」の仕組みを使って、番組表のデータそのものをPCにダウンロードするものなので、Linuxで利用す



画面3　iモード携帯電話機からの「MAIL予約」

るのはかなり難しい。

メール予約のしくみ

前述したように、ON TV JAPANで予約データをメール送信すると、ネットリモコン形式の予約データを含むメールが、指定されたアドレス宛てに送信される。このメールをPCで受信して、予約データの内容を解析することで、外部からPCのテレビ予約ができるようになるのだ（**図1**）。ちなみに、「ネットリモコン」とはテレビ朝日データが提唱しているリモート予約メールの形式で、iEPG、ADAMS-EPGの両方の電子番組表で共通に利用されている。

ネットリモコン形式の予約メールは、**図2**のような内容になっている。このメールの中で重要なのは、「open　…」で始まる1行で、ここに録画するチャンネル、開始時刻、終了時刻、日付などの情報が含まれている。このメールをLinuxマシンで定期的に受信し、内容を解析して前回作成したtvtimerスクリプトを呼び出せば、メールを介したリモート予約ができるわけだ。

●予約メールの内容

ここで、ネットリモコン形式の予約メールの内容を、もうすこし詳しく見てみよう。

予約データの内容は、**図3**のようになっている。openという単語に続いて、ユーザーがあらかじめ指定するパスコード（パスワード）が続く。次に「tv」という単語と「SC」で始まる4桁の数字が続く。この「SCxxxx」は、選択した放送局を示すコードのようだ。続いて、4桁の数字で録画開始時刻、同じく4桁の数字で録画終了時刻、最後にこれも4桁の数字で録画する月日のデータが続いている。

時刻などのデータについては、すべて24時間制の数字でそのまま埋め込まれているので、Linuxで処理するのは簡単にできそうである。本記事の自動録画システムでは、録画終了時刻ではなく、録画の継続時間を分単位で指定しているが、この計算もごく簡単な数式でできるので問題はない。

唯一問題になりそうなのは、チャンネルを指定する部分だ。「SCxxxx」という放送局識別コードには、単純なチャンネル番号が割り当てられているわけではない。これは、ON TV JAPANサービスが地域を限定したサービスではなく、全国

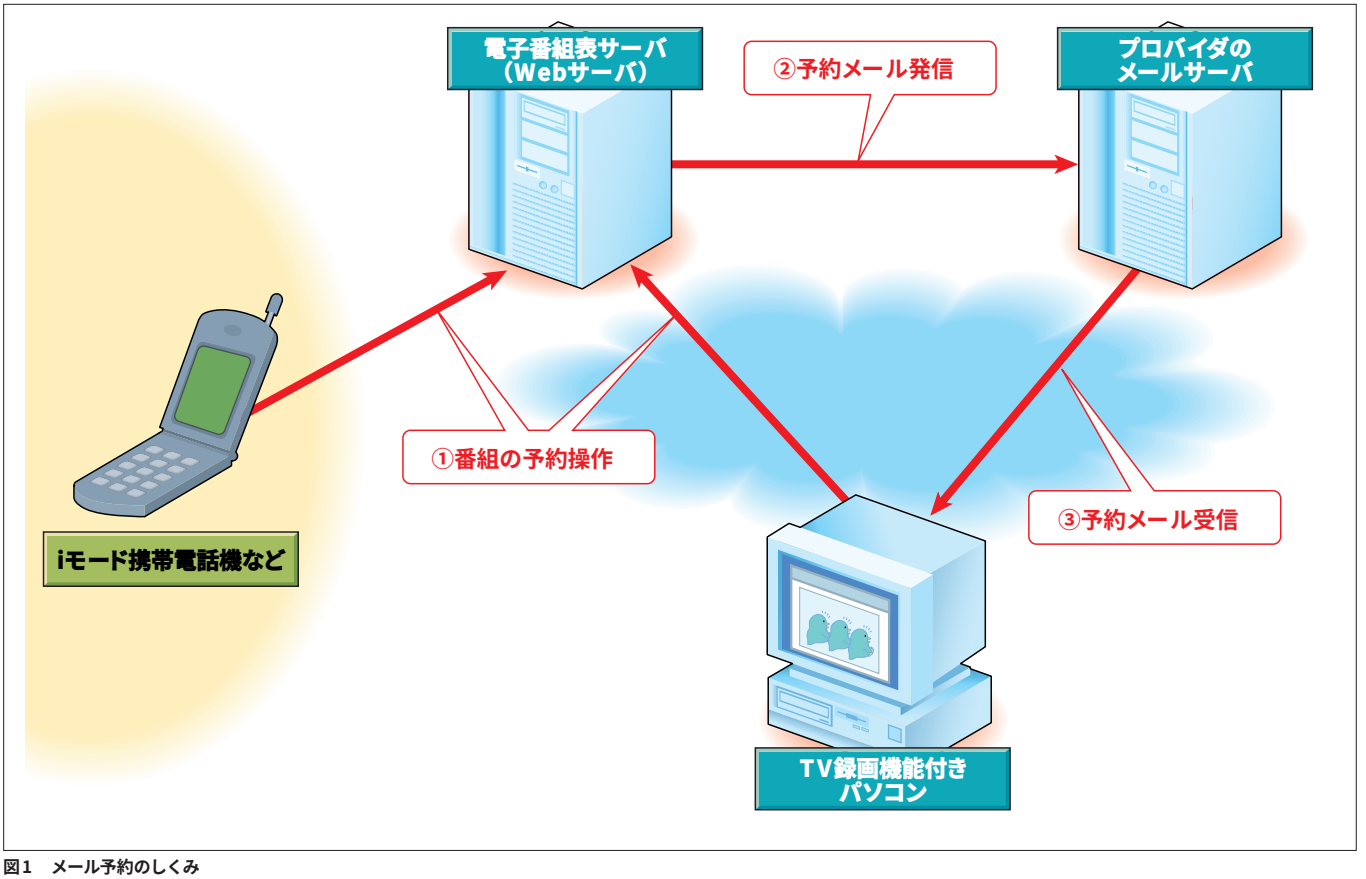
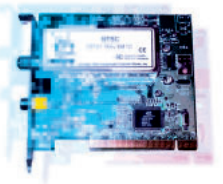


図1　メール予約のしくみ
ON TV JAPANの場合、電子番組表はWebサーバ上に存在する。メール予約の操作をすると、予約情報が含まれたメールが指定されたメールアドレス（すなわちプロバイダのメールサーバ）に発信される。このメールを録画するパソコンで受信して、録画予約が完了するのだ。



で利用できるサービスになっていることと関係している。地域が変われば受信できる放送局も変わるし、割り当てられているチャンネルも当然変わる。放送局名とチャンネル番号を単純に1対1に結びつけることはできないのだ。

ON TV JAPANを利用する場合、まず最初に自分が住んでいる地域や視聴環境を登録する。これに合わせて、Webページ上で表示される番組表の内容が変わり、メール予約に埋め込まれる放送局識別コードも変わってくるのだ。

結局、どの放送局識別コードがどの放送局に対応しているかは、実際に予約メールを自分のメールアカウント宛てに送ってみて、ひとつずつ調べるのが確実なようだ。

システム構成を考える

メール予約の概要がわかったところで、システムの構成を考えてみる。まず、テレビ録画マシン（Linuxマシン）で外部からのメールを、常時受信できるようにしなければならない。しかし、個人が自宅などで使っている環境では、自分のLinuxマシンをメールサーバにするのは現実的ではない。たとえ「フレッツ」シリーズやADSLなどの（疑似）常時接続環境を使っているとしても、グローバルIPアドレスが使えなかったり、セキュリティ上の不安があったりして、自分専用のメールサーバを立ち上げられないことも多い。なにより、正式に自前のメールサーバを立ち上げるには、自分専用のドメイン名とIPアドレスを取得して、DNSサーバを立ち上げなければならない。そこまで個人でやるのには、かなりの暇と手間とお金と知識が必要になるだろう。

●fetchmail

Linuxには、常時接続の環境でなくても、外部のメールサーバに届いた個人宛てのメールを定期的にローカルのLinuxマシンに「転送」してくれる便利なツールがある。それが「fetchmail」である。fetchmailは、接続先のメールサーバ（通常はインターネットプロバイダのメールサーバ）とメール読み出し用のパスワードなどを設定ファイルに記述しておけば、そのメールサーバに接続して到着しているメールを定期的に読み出し、それをLinuxマシンのローカルのアカウントへ転送してくれる。最近のLinuxディストリビューションの

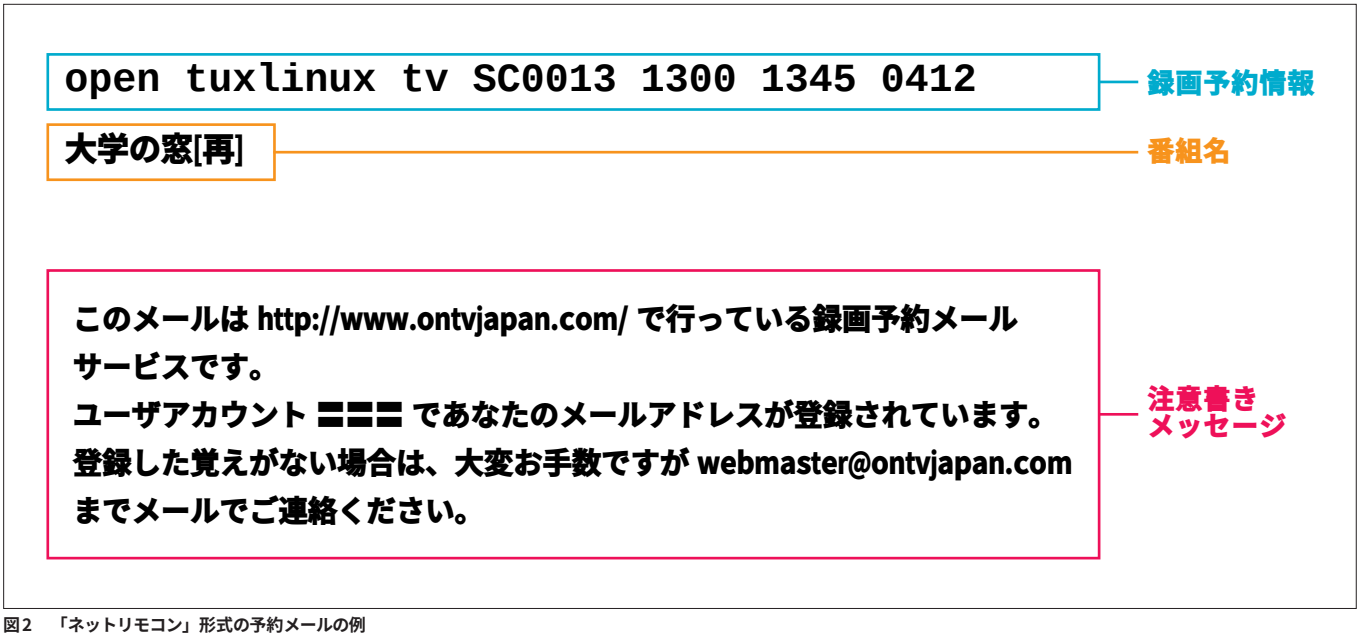


図2　「ネットリモコン」形式の予約メールの例
メール予約を行うと、このような内容のメールが指定されたメールアドレスに送られる。

図3　予約データの詳細構造
予約メール中の予約データは、このような簡単な構成になっている。ただし、将来の機能拡張で、コマンドの構造などが変わる可能性があるかもしれない。

大半は、fetchmailが標準でインストールされているはずだ。

フレッツなどの常時接続環境ならばむろんのこと、ダイヤルアップルータを使っている場合や、Linuxの各種PPPツールで自動ダイヤルアップできるような設定にしている場合なら、fetchmailの設定ファイルを作成するだけで、プロバイダのメールサーバに届いたメールを定期的にチェックできるようになる。ただし、本記事ではPPPで自動ダイヤルアップを行う方法などには触れないので、本誌の関連記事や関連書籍を参考に、自力で設定してほしい。

fetchmailで受信したメールはすべて、前回作成した「tvmaster」ユーザー宛てに転送する。このため、通常利用しているメールアカウントと予約メール用のアカウントを共用するのはお勧めできない。現在、多くのプロバイダでは、メールアカウントの追加を比較的安価な料金でできるようになっているので、このようなサービスを利用して、メール予約専用のメールアカウントを作成したほうがよいだろう。ただし、特定のアドレス宛てに届いたメールを別のアドレスに転送するだけの「メール転送サービス」や、「Yahoo！メール」のようにWebブラウザ上でしかメールの閲覧ができないようなメールアカウントサービスでは、fetchmailによるメールの読み出しはできない。いわゆる「無料」のメールアカウントには、このような形態のサービスが多いので注意すること。

●procmail

tvmaster宛に転送されたメールは、その内容を読み出してスクリプトで処理することになる。このような処理にうってつけなツールが「procmail」である。procmailもfetchmail同様、ほとんどのLinuxディストリビューションで、標準でインストールされるようになっている。

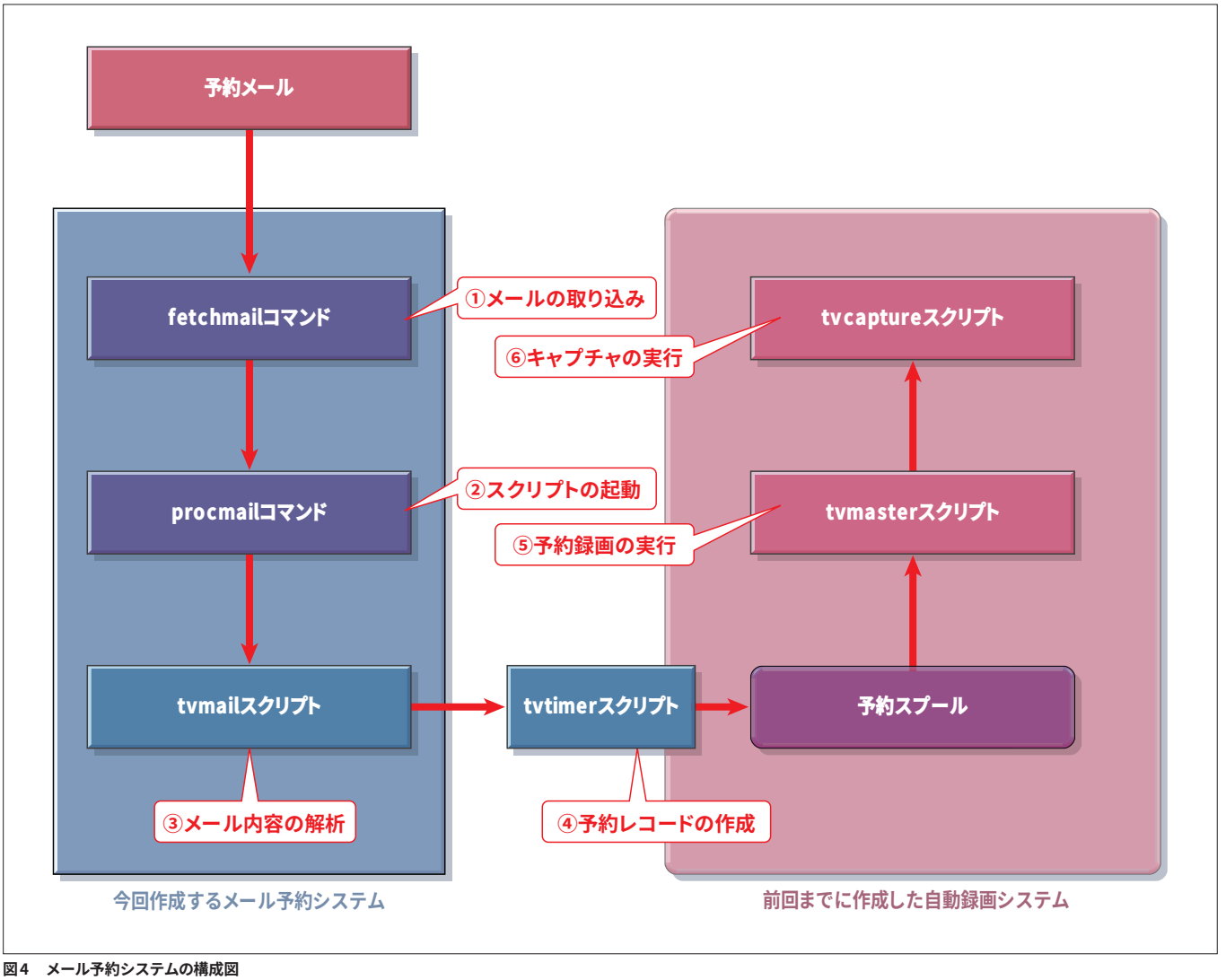
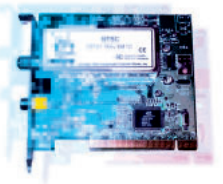


図4　メール予約システムの構成図



procmailは、到着したメールを即座に読み出し、その内容に従って、メールを別の場所に転送したり、ファイルに保存したり、コマンドを起動してメールの内容を標準入力に渡したりすることができる。前掲の**図3**のようなテキストデータを処理して、録画開始時刻、録画時間、チャンネル名などをtvtimerスクリプトに渡すようなスクリプトを作成し、それをprocmailから起動できるようにすれば、メール予約の目的は達成できる。このスクリプトを「tvmail」スクリプトと命名することにしよう。

●システムの全体像

以上のように、fetchmail、procmail、tvmailスクリプトなどを含めた、メール予約システムの構成図は**図4**のようになる。tvmailスクリプトからtvtimerスクリプトを呼び出してしまえば、あとは前回作成したtvmasterの予約システムで、予約が自動的に実行されるようになるはずだ。

ON TV JAPAN側の準備

メール予約を行うには、まずON TV JAPANにユーザー登録しなければならない。登録や利用にあたって、会費や利用料が必要になることはない。登録の手順については、ここでは説明しない。ON TV JAPANのホームページ（http://www.ontvjapan.com/）をWebブラウザで開き、左側のメニューにある「メンバー登録」の項目をクリックして、あとは画面上に表示される指示に従えばよい。

唯一注意が必要なのが、メール録画に関する設定だ。メンバー登録の途中や、トップページのメニューから「パーソナル設定」を選択すると表示される画面（**画面4**）で、「メール録画予約」の項目をクリックする。すると、**画面5**のようなページが表示されるので、「合い言葉」のテキストボックスに、予約メールに付加させたい「パスコード」を適当な文字列で入力する。ここには、日本語（全角）文字も入力できるが、文字コードの扱いなどがやっかいなので、半角の英数字にしておいたほうがよいだろう。「送信先アドレス」の部分には、メール予約専用に使うメールアカウントのアドレスを入力する。

●放送局識別コードの調査

メンバー登録や予約用メールアドレスの設定が終了したら、自動録画システムの設定を始める前に、前述した「放送局識別コード」の調査をしておこう。

ON TV JAPANのトップページに戻ってから、「テレビ番組表」という項目で「パーソナルチャンネル」のラジオボタンをオンにして、「GO」ボタンをクリックする。すると、個人の視聴環境に合わせた番組表が表示される（**画面6**）。ここで、メール録画の対象にしたい放送局の適当な番組名をクリックすると、**画面7**のように番組の詳細情報が表示され、画面左下に「iEPG録画」、「MAIL録画予約」、「MAIL CLIP」というボタンが表示されるはずだ。ここで「MAIL録画予約」をクリックすると、設定されたメール録画用メールアドレス宛てに、録画予約メールが送信される。この操作を、自動録

画面4　ON TV JAPANのパーソナル設定画面

画面5　メール録画予約の設定
「合い言葉」の部分にパスコードを、送信先アドレスの部分には、メール予約専用のメールアドレスを入力する。

画させたい放送局ごとに順次繰り返せば、各放送局に割り当てられた放送局識別コードを含んだメールが、予約用のメールアドレスに届けられることになる。

ところで、ON TV JAPANでは、すべての放送局がメール予約に対応しているわけではない。たとえば東京地区の場合、テレビ東京については番組表は表示できるものの、メール予約しようとすると、**画面8**のように「この放送局は対応していない」という意味のメッセージが表示され、予約メールは送信されない。これについては、ユーザー側では対応のしようがないのであきらめるしかない。

テスト用のメールの送信が終わったら、メール録画用アカウントのメールサーバに接続して、メールを読み出してみる。メールの送信順や番組名などを頼りに、どのメールがどの放送局の予約に対応しているものなのかを割り出して、放送局名と放送局識別コード（SCxxxx）の対応表をメモしておく。ちなみに、東京地区のVHF地上波放送（およびUHFの「放送大学」）の放送局識別コードは、**表1**のようになっている。

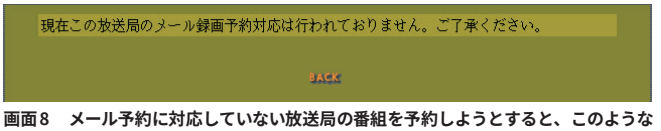
fetchmailの設定

fetchmailは、大半のディストリビューションに付属しているようなので、ここではfetchmailそのもののインストールについては説明しない。fetchmailに関しては、現在は英文の文書しかないようだが、http://tuxedo.org/~esr/fetchmail/を参考にしてほしい。また、Config-HOWTO（http://www.linux.or.jp/JF/JFdocs/Config-HOWTO-4.html）に、ごく簡単だが日本でfetchmailについての解説がある。

fetchmailの設定は、tvmasterユーザーのホームディレクトリ上に.fetchmailrcというファイルを作成してその内容を編集する方法と、「fetchmailconf」という専用の設定ソフト（X Window System上で動作する）を使う方法の2通りがある。

fetchmailconfを使えば、GUI上で設定が行えるので一見便利なようだが、いくつものウィンドウを開いたり閉じたりしなければならないのが面倒だ（**画面9**）。.fetchmailrcの内容はそれほど複雑ではないので、直接ファイルを編集したほうが早いだろう。いずれにせよ、必要になる設定項目は以下のとおりである。

- **接続先のメールサーバのアドレス**
- **メールサーバがサポートするプロトコル（通常はPOP3）**
- **定期的にメールをチェックする場合、その間隔**
- **メールサーバに接続するときに必要なユーザーIDとパスワード**
- **ローカルマシンに転送する際の転送先アカウント名（ここではtvmaster）**



画面8　メール予約に対応していない放送局の番組を予約しようとすると、このようなメッセージが表示される

画面6　ON TV JAPANのパーソナル番組表
自分の視聴環境に合わせた番組表が表示される。番組名をクリックすれば、その番組の詳細が表示される。

画面7　番組名をクリックした後の状態
左下のボタンをクリックすれば、録画予約を行える。

.fetchmailrcの例を**リスト1**にあげておく。このリスト中で、メールサーバの名前、ユーザーID、パスワードなどの部分を、自分の環境に合わせて書き換えればよい。また、手動でダイヤルアップ接続する場合には、「set daemon 300」の行は削除する。このように設定した場合、ダイヤルアップした後で「fetchmail」コマンドを手動で起動して、予約メールを受け取ることになる。

なお、.fetchmailrcには、接続先メールサーバのユーザーIDとパスワードを「生のまま」記述することになるので、誰かに見られるようなことがあっては大変である。このため、.fetchmailrcファイルは、自分以外は読み書きできないように設定する必要がある（誰にでも読み出しできるような設定になっている場合、fetchmailは警告を表示して強制終了する）。tvmasterユーザーとしてログインしている状態で、次のようにファイルのアクセス権を設定すること。

```
$ su tvmaster
$ chmod 600 /home/tvmaster/.fetchmailrc
```

ところで、fetchmailを使用する場合、ローカルのマシン上で「sendmail」が動いている必要がある。このsendmailは、ローカルマシン上やリモートマシンとの間のメール転送を請け負うプログラムなのだが、働きの割には肥大化したプログラムで、特に起動時にかなりの負荷をマシンにかけることがある。このため、はじめからsendmailをインストールしていなかったり、起動しないように設定しているLinuxユーザーも多いかもしれない。そのような場合は、改めてsendmailプログラムをディストリビューションのCDからインストールしなければならない。また、停止させている場合には、以下のような手順で起動するように再設定する。

```
$ su
# /etc/rc.d/init.d/sendmail start
```

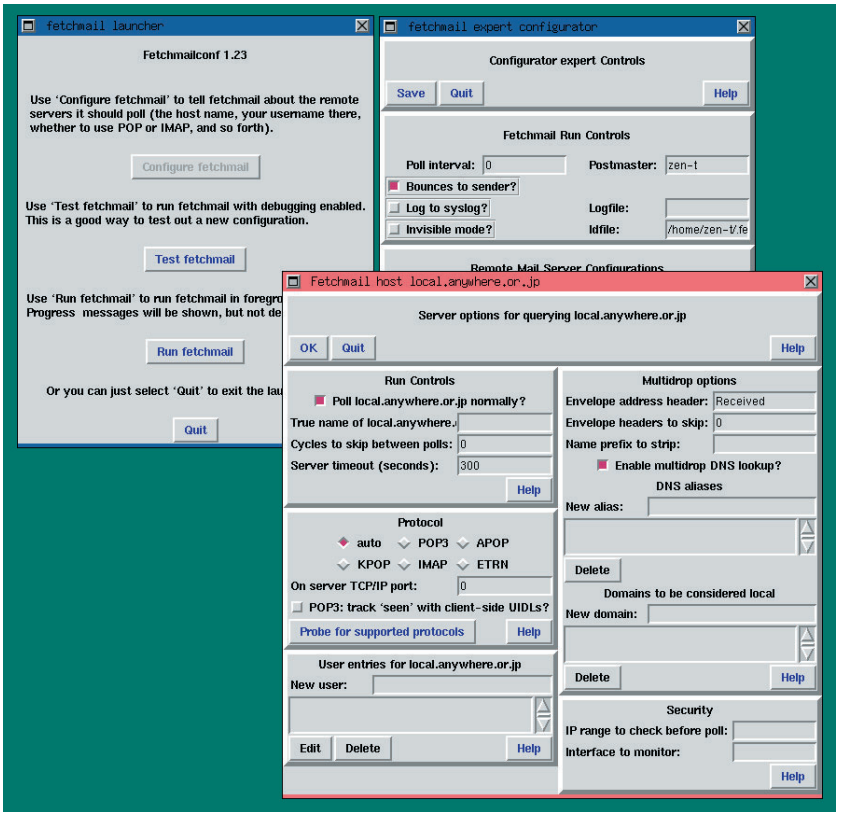
sendmailの代わりに、「qmail」や「Postfix」などのより高速で使いやすいメール転送プログラムを使うこともできるが、これについては話が長くなるのでここでは説明しない。個人の興味に応じて、挑戦してもらいたい。

procmailの設定

ここまでで、ON TV JAPANから送られた予約メールは、ローカルのLinuxマシンのtvmasterユーザー宛てに定期的に届くようになった。つぎは、届いたメールの内容を解析する

識別コード	放送局
SC0031	NHK総合
SC0041	NHK教育
SC0004	日本テレビ
SC0005	TBS
SC0006	フジテレビ
SC0007	テレビ朝日
SC0013	放送大学（UHF放送）

表1　（参考）東京地区のおもな地上波放送局の放送局識別コード一覧



画面9　fetchmailの設定を行うfetchmailconfアプリケーション
開くウィンドウの数が多く、簡単な設定をするだけならかえって使いにくいかもしれない。

スクリプトを自動的に起動するための設定が必要になる。この起動処理を行うのに便利なのがprocmailだ。本来、procmailはメールサーバに届いたメールを解析して、その宛先に応じてローカルのメールスプールに送ったり、さらに別のマシンに転送したりする処理を行うためのプログラムだが、一般ユーザーが自分に届いたメールを解析して、プログラムに処理させたり、別のメールアカウントに転送したりするのにも利用できる。

procmailもfetchmail同様に、ほとんどのディストリビューションで標準でインストールされているので、procmailのインストールそのものについては言及しない。また、これもfetchmail同様に、ローカルマシンでsendmailが起動している状態でないとprocmailは使えないので注意すること。

procmailで設定が必要になるのは、tvmasterユーザーのホームディレクトリに置く次の2つのファイルだ。

- **.forward**
- **.procmailrc**

.forwardの内容を**リスト2**に、.procmailrcの内容を**リスト3**に掲載する。どちらも「暗号」のような内容になっているが、特に.forwardの内容については、この設定が「定番」になっているので、この通り変更せずにファイルを記述することをお勧めする。特に、.forwardの内容を、

```
| /usr/bin/procmail
```

のような簡単なものにしておくと、悪意を持った第三者からいたずら目的でprocmailを誤動作させられる危険性があるので注意すること。

リスト3の.procmailrcでは、ON TV JAPANから届いたメールのみを、/usr/local/bin/tvmailというスクリプトの標準入力に渡すように設定している。その他のメールについてはいっさい無視するようになっているが、ここにさらに自分で記述を追加することで、予約メール以外のメールを別のアカウントに転送したり、ファイルに保存したりするようにもできる。procmailのオンラインマニュアルなどを参考に、各自で挑戦してみてほしい。

ところで、procmailを正しく動作させるためには、.forwardと/home/tvmasterディレクトリのアクセス権を適正に設定しておかなければならない。

まず.forwardだが、第三者の書き込みができないようにするのは当然のことだが、第三者からの「読み出し」は可能になっていないとうまく動かない。このため、tvmasterユーザーとしてログインしている状態で、次のように設定する。

```
$ chmod 644 /home/tvmaster/.forward
```

一方、/home/tvmasterディレクトリが、同一グループの第三者を含めて、第三者が書き込み可能な状態になっていると、procmailは動作しないようになっている。このため、同じくtvmasterユーザーとしてログインしている状態で、次のように設定する。

```
$ chmod 755 /home/tvmaster
```

tvmasterユーザーのホームディレクトリには、録画済みファイルや予約ファイルなどがあるため、最低限読み出しアクセスとディレクトリへの移動（実行属性）がついていないと不便なので、上のように設定している。

ちなみに、もうひとつの設定ファイルである.procmailrcについては、アクセス権の設定に関する制限は特にないようだが、念のため、第三者が書き換えられるような設定にはして

▼リスト1　/home/tvmaster/.fetchmailrcの例

```
set postmaster "tvmaster"
set nobouncemail
set properties ""
set daemon 300
poll ■■■.■■■.ne.jp with proto POP3
       user ■■■ there with password ■■■ is tvmaster
here warnings 3600
    antispam 571 550 501 554
```

「■■■」の部分を、各自が利用しているメールサーバ名、メールアカウント名、パスワードに置き換える。また、作成後に「chmod 400 /home/tvmaster/.fetchmailrc」を実行して、他人に中身を読まれないようにすること。

▼リスト2　/home/tvmaster/.forwardの内容

```
"| IFS=' '&&p=/usr/bin/procmail&&test -f $p&&exec $p -f-
||exit 75#hoge"
```

ファイル作成後「chmod 644 /home/tvmaster/.forward」を実行して、第三者からの読み出しができるように設定する。末尾の「#hoge」というコメント文字列は、クラッキング対策として必要なものなので、省略しないこと（#以降の文字列はなんでもよい）。

▼リスト3　/home/tvmaster/.procmailrcの内容

```
:0
* ^From:.*admin@ontvweb
| /usr/local/bin/tvmail
```

予約メール以外のメールを保存するなどの処理を追加したい場合は、このファイルにさらに記述を追加すればよい。

おかないほうがよいだろう。

```
$ chmod 644 /home/tvmaster/.procmailrc
```

以上の設定が済んだら、念のため、tvmasterユーザー宛てにメールを送ってみて、エラーにならないかどうかテストしてみよう。tvmaster以外のユーザーとしてログインしている状態で、次のコマンドを実行してみる。

```
$ mail tvmaster
Subject: test［リターンキーを押す］
this is test［Ctrl-Dを押す］
Cc:  ［リターンキーを押す］
$
```

もし、procmailによるメールの処理がうまくできないと、エラーメールが自分宛てに返ってくるはずである。特に、/home/tvmasterディレクトリや.forwardファイルのアクセス権設定が間違っていると、

```
Address <tvmaster@localhost> is unsafe for mailing to
programs
```

というように、このアドレスは「安全でない」というような意味のエラーメールが返ってくることになる。

メールの配送と処理の下準備はできたので、いよいよ核心部分のメール解析処理を行うスクリプトを作成しよう。

メール処理スクリプトの作成

メールを処理するPerlスクリプト「/usr/local/bin/tvmail」の内容は、**リスト4**の通りである（付録CD-ROMに収録）。核心部分といいながらずいぶん短いスクリプトなので、拍子抜けした読者がおられるかもしれない。

スクリプトの前半部分では、%channelcodeという連想配列に、ON TV JAPANから送られてくる放送局識別コードと、xawtvで使っている放送局名（前回、前々回に説明した、~/.xawtvファイルで定義しているチャンネル名）の対応表を設定している。また、$passcodeという変数には、ON TV JAPANのユーザー設定時に決めたパスコードを設定する。

あとは、簡単な日付計算や時間の計算をして、/usr/local/bin/tvtimerスクリプトを呼び出しているだけだ。処理はかな

▼リスト4　/usr/local/bin/tvmail

```
#!/usr/bin/perl
#
# tvmail : process iEPG mail and invoke tvtimer script
#
# copyright (c) 2001 by Zentaro Takeda
#
# only for personal use

%sctable =
  ("SC0031" => "NHK",
   "SC0041" => "NHK-EDU",
   "SC0004" => "NTV",
   "SC0005" => "TBS",
   "SC0006" => "FUJI-TV",
   "SC0007" => "TV-ASAHI",
   "SC0013" => "U-AIR");

$passcode = "tuxlinux";

while (<>) {
  if ($_ =~ /^open $passcode/) {
    @lwords=split(/ /,$_);
    $channelcode=$lwords[3];
    $channelcode =~ s/\s//g;
    $channelcode = $sctable{$channelcode};
    $starttime=$lwords[4];
    $starttime =~ s/\s//g;
    $endtime=$lwords[5];
    $endtime =~ s/\s//g;
    $startdate=$lwords[6];
    $startdate =~ s/\s//g;
  }
}
#

$start_hour=substr("$starttime",0,2);
$start_minute=substr("$starttime",2,2);
#
$end_hour=substr("$endtime",0,2);
$end_minute=substr("$endtime",2,2);
#
$start_month=substr("$startdate",0,2);
$start_day=substr("$startdate",2,2);

if ((0 + $start_hour)>(0+$end_hour)){
  $end_hour=$end_hour+24;
}

$rec_length = ($end_hour - $start_hour) * 60 +
$end_minute - $start_minute;

($sec, $min, $hour, $mday, $mon, $year, $wday, $yday,
$isdst) = localtime();

$year = $year + 1900; # Y2K
$mon = $mon + 1;

if ($mon > $start_month){
  $start_year=$year + 1;
} else {
  $start_year=$year;
};

$start_date=substr("0000$start_year",-
4).substr("00$start_month",-2).
  substr("00$start_day",-2);

$commandline = "tvtimer -c $channelcode -d $start_date -
s $starttime -l $rec_length -r 0 -p MAIL";

exec $commandline;
```

り簡略化しているが、日をまたがった予約（夜中の23時過ぎに始まって、深夜の0時以降に終了するような番組）にも対応している。また、ON TV JAPANの予約メールには「年号」の情報が入っていないのだが、年をまたがった予約（来年の番組）もできるようにしておいた。ただし、2年先の番組の予約はできないが、そのような予約をすることはまずないだろう。

過去の番組を予約しても、そのまま処理されて、予約レコードがスプールに作られるが、これはtvmasterスクリプトが検出して自動的に排除してくれるので、tvmailスクリプト内では特にチェックしないことにした。

●テストしてみる

tvmailスクリプトを作成したら、次のように実行可能属性をつけておく。

```
$ su
# chmod +x-w /usr/local/bin/tvmail
```

つぎに、いきなりON TV JAPANから予約してみてもよいのだが、**リスト5**のようなテスト用のダミーデータを作ってみて、これをtvmailスクリプトに読み込ませてみる。

```
$ cat dummy.txt | /usr/local/bin/tvmail
```

/home/tvmaster/spoolディレクトリを調べて、テスト用の予約レコードができていれば、tvmailスクリプトは問題なく動くことになる。

```
$ ls /home/tvmaster/spool
MAIL_NTV_200104211000_60_0.tvr
```

不要なら、この予約レコードは削除しておく。

ちなみに、今回のシステムでは、メール予約した番組の「番組名」は、すべて「MAIL」という文字列になるようにしている。ON TV JAPANのメール予約では、番組名などの情報もメールで送られてくるのだが、Linux上での日本語文字列の扱いに自信がないので、予約ファイルなどへの反映はいっさいしていない。現在、「予約レコード」のファイルの中身は空のままなので、あるいは番組名などの情報をファイルの中身として保存してみてもいいかもしれない。これについては、今後の課題としたい。

▼リスト5　tvmailスクリプトのテスト用データの例

```
open tuxlinux tv SC0004 1000 1100 0421
jkasdjlf;ajksd;faklsjdf
shg;dshf
open linuxtux tv SC0005 1100 1200 0421
```

パスコードを違えた行などを混ぜてみて、正しく動作するかどうかをチェックする

運用開始

すべての設定ファイルやスクリプトの準備が済んだら、ON TV JAPANの電子番組表から、適当な番組を選んで「MAIL予約」してみる。次に、tvmasterユーザーとしてログインしている状態で、fetchmailを起動する。

```
$ fetchmail
```

設定に間違いがなければ、なにも表示されないでコマンドラインプロンプトに戻るはずだ。しばらく待ってみてから、/home/tvmaster/spoolディレクトリを調べて、先ほど予約した番組に相当する予約レコードファイルができていれば、予約は成功である。

fetchmailプログラムは、前掲のリスト1のような設定をしている場合、いったん起動するとそのままデーモンプログラムとして常駐して、指定した間隔で定期的にメールを読み出すようになっている。この処理は、tvmasterユーザーがログアウトしても、そのまま継続される。ただし、マシンをリブートしたりシャットダウンしたりすると、次にシステムが起動してもfetchmailは起動しない。改めて、tvmasterユーザーとしてログインして、fetchmailを手動で起動しなければならない。

これでは面倒ということであれば、次のような行を/etc/rc.d/rc.localファイルの末尾に追加してみるとよいだろう。

```
(su tvmaster; fetchmail)
```

ただし、このような記述がすべてのディストリビューションで有効かどうかはわからない。場合によっては、fetchmailをデーモンモードで動かすのはあきらめて、tvmasterユーザーのcrontabを編集して、定期的にfetchmailを起動するようにしたほうがよいかもしれない。

次回は、懸案のMPEG圧縮に今度こそ挑戦する予定だ。

箱の中のペンギンたち

文：みわよしこ
Text : Yoshiko Miwa miwachan@pp.iij4u.or.jp

第5回　組み込みLinuxシステムの開発ツール

システムの開発には非常に多数のステップがあります。システムの「心臓」となるCPUの開発だけを大まかに見ても、単体トランジスタの設計・回路の設計・レイアウトの設計・外形の設計といった、多段階の設計工程が必要となります。さらに、それぞれの工程で設計ツールが別途必要になりますし、それらを組み合わせるためのツールが必要になることもあります。

もちろんシステムはCPUだけでは作れません。周辺機器と組み合わせ、適切なOSやプログラムを搭載することではじめて構築でき、また適切な筐体に収められる必要があります。これは、システムがスーパーコンピュータであろうが4ビットマイコンであろうが同じことです。そして、そのすべての段階でツールが必要になります。

マイコン開発の場合、たいていの場合は大手半導体メーカーの製品ラインナップの中から、適切なCPUを選択することになります。

マイコンの世界では、未だに4～16ビットのCPUが健在です。マイコンは使用目的に応じて、消費電力、入出力インターフェイス、実装形態など、非常に多数のバリエーションを必要とするため、大手半導体メーカーが多数の選択肢を用意しているのが普通です。その中でも日立のH8シリーズは、一般的な製品に広く利用されているマイコンで、8ビット、16ビットの多様なラインナップがあります。

Linuxで利用できるGCC（GNU C Compiler）には、数年前からH8シリーズ向けのマシンコードを作成するためのコンパイルオプションがすでに用意されていました。コードをマイコンに搭載するには、PCのハードディスクに相当するマイコンのROMにコードを書き込まなくてはなりません。また、ROMの種類によっては「書き込み・消去可能な回数が100回程度まで」といった制約があるため、開発時にはプログラムを実際にROMに搭載しないで動作を検証するためのツールも必要になります。

マイコンシステムのプログラム開発の流れ

通常、パソコン上でC言語などのプログラムを作成する場合に必要な作業は、適切なライブラリを利用してソースコードを作成し、GCCなどのコンパイラでソースコードから実行形式を生成することだけです。実行にあたって、プログラムがメモリのどこにどのように読み込まれるのかを、開発者が気にする必要は通常はありません。

一方、マイコンシステムのためのプログラム開発の場合、開発者が決めなくてはならない要素が多いため、開発の流れはかなり複雑になります。たとえばプログラムがメモリのどこに読み込まれ、どのように実行されるかを開発者が決める必要があるのです。このため、マイコンシステムの開発者は、必然的にそのマイコンの性格を熟知することになります。

エディタを利用してソースコードを作成するのは、パソコン

上の通常のプログラム開発と同じです。異なる点は、最初にI/Oポートアドレスの定義を行わなくてはならないことです。パソコンの場合、I/Oポートアドレスはある程度「決め打ち」されていたり、自動割り当てを行うシステムが存在したりしますが、マイコンのI/Oポートアドレスは機種ごとに異なります。

さて、マイコンは小さいといえども独立したコンピュータです。パソコン上でLinuxを起動するのにLILOなどのブートローダを利用する必要があるのと同様、マイコンに組み込むプログラムにもブートローダに相当する「スタートアップルーチン」が必要です。これも、開発者がマイコンごとに、また、プログラムごとに個別に開発する必要があります。スタートアップルーチンは、I/Oポートやメモリのチェック（初期化）を行い、スタック領域を設定し、プログラムのメインルーチンを呼び出す作業を行います。あとはパソコン上と同様に、プログラムが実行されることになります。データやプログラムをメモリのどこに配置するかも、開発者が定義する必要のある部分です。

スタートアップルーチンは、C言語などの高級言語ではなく、アセンブラで書く必要があります。したがってスタートアップルーチンの開発には、アセンブラ、ディスアセンブラ（逆アセンブラ）が必要になります。

開発したプログラムは、実際のマイコンのROMに書き込む必要があります。ここで必要となるのが「ROMライタ」と呼ばれるものです。これはパソコンから転送したデータをROMに書き込む装置で、コントロールソフトとセットでROMライタと呼ばれることもあります（最近では、フラッシュROMを内蔵しているCPUもあるので必ずしもROMライタを必要としないこともあります）。

マイコンに搭載されたROMにはいくつかのタイプがありますが、一般的に利用されるROMの多くは、紫外線や電気によって、データの書き込み・消去が行えるタイプです。これを利用すれば、開発はマイコンチップそのものを用いて行えるかに思えます。ところが実際には、そうはいかないのです。

たとえば「EPROM」と呼ばれる、データ書き込み・消去が何度も可能なタイプのROMがあります。EPROMは、データの書き込みは電気的に行い、消去は紫外線を照射することによって行います。データの消去には1時間以上かかります。データが消去されていないと新しいデータを書き込むことはできませんから、すぐに新しいデータに差し替えるということはできないのです。また、メーカーが保証している書き込み・消去回数にも、たとえば「100回まで」といった制約があり、ハードディスクのように実用上無限に等しい書き込み・消去が可能なわけではありません。

プログラムの開発には「作成→実行形式作成→実行→不具合修正→実行形式作成……」といった繰り返しがつきものですが、この過程を実際のマイコンのROMを使って行うことは、時間的にも、マイコンの耐久性という面でも現実的ではありません。

そこで必要になるのが「モニタプログラム」です。マイコンに搭載したモニタプログラムと開発用のLinuxマシンを

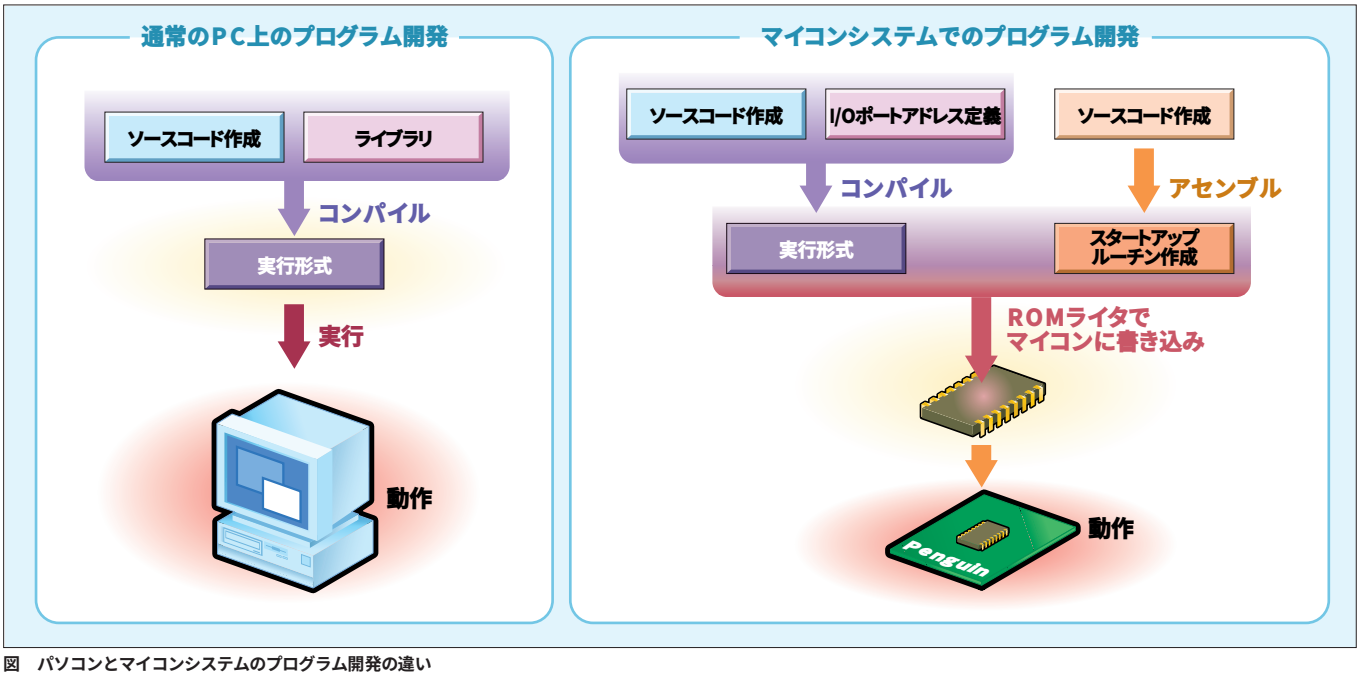


図　パソコンとマイコンシステムのプログラム開発の違い

RS-232Cなどの通信ケーブルで結び、開発したプログラムをマイコン上のRAMに転送して、リモートで実行します。モニタプログラムを利用してマイコン上のメモリ内容を確認したり、接続した周辺機器の初期設定の状態や、レジスタなどの状態、実行のようすを詳細に知ることができます。

モニタプログラムを使って十分なテストを行い、不具合のなくなったプログラムをROMライタでROMに転送することで、マイコンへのプログラムの組み込みはひとまず完了します。

なお、組み込みマイコンシステムの構築では、マイコンに組み込むプログラムを開発するだけでなく、周辺回路の設計・基板の設計などの作業と、ツールが別途必要です。

Linux上で日立H8シリーズマイコンの開発を行うためのツール

ここでは、今回インタビューした三岩氏が開発した多数のツールも含め、開発段階ごとに利用するツールを紹介します。

・ソースコードのエディタ

プログラマーはEmacsを利用することが多いでしょう。プログラム作成を支援する機能が豊富に含まれているからです。

・コンパイラ

Intel版Linux上でH8シリーズ用のコードを生成するクロスコンパイラを利用します。それを用いてソースコードのコンパイルを行うことで、H8上で動作する実行形式を作成することになります。

・組み込み用ライブラリ

glibcを利用すると、プログラム実行形式のサイズが大きくなってしまい、マイコンの小容量のRAM／ROM（多くは2～256kバイト）に搭載できなくなります。このため、必要な関数だけを静的にリンクして実行形式のサイズを抑制できるライブラリ「newlib」を利用します。

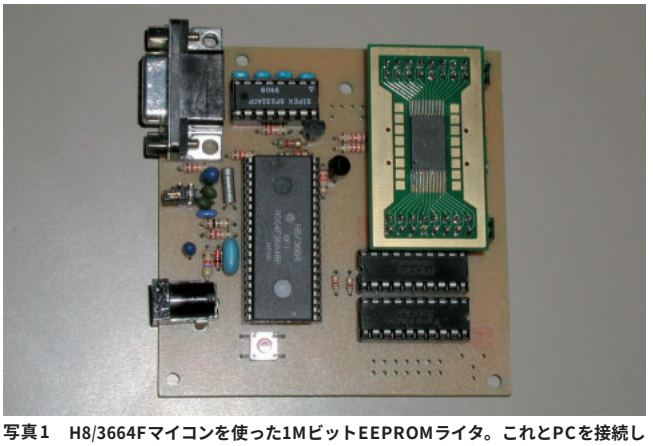


写真1　H8/3664Fマイコンを使った1MビットEEPROMライタ。これとPCを接続して、プログラムの実行形式をEEPROMに書き込む。右上にあるのがEEPROM。

・アセンブラ

Linuxには「binutils」という開発ユーティリティ群がありますが、これに含まれているGNUアセンブラ（as）には、H8のアセンブラプログラムをH8の機械語に翻訳する機能があります。

・ROMライタ

H8シリーズのF-ZTATはフラッシュROMを内蔵しているので、書き換えを電気的に行えます。Linux上からプログラムを書き換えるソフトを三岩氏が開発し、公開しています。

・モニタプログラム

Linuxマシン上でマイコンの動作を確認できるモニタプログラムを三岩氏が公開しています。

なお、三岩氏の開発したROMライタコントロールソフト・モニタプログラムは現在、H8/3664、H8/3048、H8/3067の3つの種類のマイコンに対応が可能になっており、Linux上でH8シリーズマイコンの開発を行うためのほとんど唯一のツールとなっています。

■その他のツール

マイコンには日立H8シリーズ以外にも、多種多様の種類があります。また、マイコンシステムはマイコン本体だけでなく、周辺回路やそれらを実装するための基板とセットで初めて完成します。

・GUI統合開発環境

三岩氏はLinux上でのマイコン開発のためのGUI統合環境「MIDE」を開発しました。現在はまだ評価版ですが、十分に実用に耐えるということです。

・回路図エディタ

周辺回路の設計には回路図が必要になります。Linux上では「xcircuit」や「gschem」が存在します。

・プリント基板開発ツール

多くの場合、プリント基板も設計する必要があります。Linux上で利用できるツールには、「PCB-CAD」があります。プリント基板作成でもっともポピュラーな感光基板の作成に使用できますが、NC自動工作機械にも対応していますので、NCルータマシンを利用した基板作成も可能です。

このように、現在ではLinux上のツールを利用すれば、マイコン開発のほとんどのフェーズに対応できるようになっています。

三岩幸夫氏インタビュー

Interview

ふだんの生活で、なかなか意識されることのない組み込みマイコンシステム。その開発ツールは、マイコンシステム以上に一般人の生活の場には縁のない存在です。今回は、マイコンシステムの開発ツールを多数開発してきた三岩幸夫氏に、ツールを開発するということや日々の思いについてお尋ねしました。

―はじめまして。私、「マイコンシステムは企業で開発するもの」という思い込みがありまして、三岩さんが個人でマイコンシステムの開発ツールを開発されているというのが、非常に意外に感じられるんですよ。

三岩：そうですか？

―三岩さんはお仕事として、障害者の方にマイコンシステム開発を教えていらっしゃるわけですが、ツールの開発は業務の一環なのでしょうか？

三岩：いえ、ふだんは朝から夕方まで、ずっと生徒さんと接しています。定員は10人で、今は1人で7人の生徒さんを見ています。ここは職業能力を開発する学校ですから生徒さんがたも熱心で、夕方5時近くまで質問を受けたりしています。

―ではツールの開発はいつやっていらっしゃるのでしょう？

三岩：通勤電車の中ですとか、帰宅後ですとか……。だいたい食事をして帰宅するのが夜の8時頃ですので、それから12時頃まで自分の時間があります。それに休日もありますし。

―そういったご自分の時間には、ずっとツールの開発をやっていらっしゃるのですか？

写真　三岩幸夫（みついわゆきお）氏
1967年生まれ。石川工業高校電気科を卒業後、日立製作所で汎用機の検査に従事する。1991年より神奈川県公務員。電気工事を担当したあと、1996年より障害者の電子機器制御分野の職業能力開発に従事し、現在に至る。余暇を利用してLinux上でマイコンシステム開発ツールや、デスクトップツールを開発している。また、Plamo Linux開発スタッフの一員でもある。コンピュータ以外の趣味はヨーロッパ史の研究やサイクリング。特に東ローマ帝国、ハプスブルグ王朝、近代ドイツ帝国の幕藩体制などに関する造詣が深い。

三岩：いえ、歴史の本を読んだり、ぼうっとしていたり。ヨーロッパ史が趣味なんです。歴史の物語は面白いですし、現在、たとえばボスニアで発生している民族紛争の遠因が、案外歴史的な経緯にあったりするので、国際ニュースが興味深く読めたりもしますし。

―コンピュータに関わる時間はどのくらいなのでしょうか？

三岩：まったく触らない日もありますね。だいたいいつも、作りたいツールや解決したい課題のことを心の片隅で考えていて、アイディアがあったらメモしておき、形に出来そうになったところで一気に試作しています。試作にかかりはじめると、1日～2日はそればかりですね。

―ご家族は何も言われませんか？

三岩：私、独身ですので（笑）。

―ヨーロッパ史とLinux上でのツール開発の組み合わせは意外に思えるんですが、ご自分で何か通じるものを感じていらっしゃいますか？

三岩：そうですねぇ……あえて言えば、どちらも「やっている人が日本にあまりいない」ところが共通していますね。ヨーロッパ史で私が特に興味を引かれているのは、ハプスブルグ家の歴史なんですが、日本ではこのあたりをやっている方があまりいないんです。Linux上でのマイコン開発ツールも、断片的にはほかにいくつか見受けられますが、徹底してやっている方が日本にはほかにいなくて……。

―それでご自分で開発されてきたわけですね。

三岩：ええ、気が付いたら同じようなことをやっている方が誰もいなかったわけです。

―ところで、ご経歴を見ていて不思議に思ったことがひとつあるんです。現在の職場に来られるまで、マイコン開発に関係するようなお仕事にはついていらっしゃらなかったわけですよね。

三岩：ええ。それに、まともにパソコンを触りだしたのが3年前で、Linuxに関わりはじめたのもそれからなんです。

―高校は工業高校の電気科に進まれたわけですが、学校の授業でマイコンシステム開発など学ばれましたか？

三岩：いえ、まったくそういうことは学んでいません。電気科に進んだのもなんとなくで……。学校のカリキュラムは電気工事が主になっていて、一応、第二種電気工事士の資格も取得しました。

――その後、日立製作所に就職されたのですね。そこでマイコンを学ばれたわけですか？

三岩：いえ、プリント基板の穿孔作業や、メインフレームの検査に従事していました。ただ、1年ちょっと研修所で情報処理を学びまして、そのあと、今度はメインフレームのモニタプログラムの検査に従事していました。バグがある部分の頭にジャンプ命令を入れて、バグ部分を回避するといったこともやりましたね。

――現在は公務員でいらっしゃるわけですが、転職のきっかけは何だったのでしょう？

三岩：友人が公務員試験の募集の切り抜きを持ってきまして、一緒に受けにいったんです。友人は落ちて私だけ受かってしまって（笑）。「電気」の区分で受験していましたし、高校で取得した第二種電気工事士の免許もありますので、電気工事の訓練に5年ほど従事していました。

――そのお仕事もマイコンシステムとは関係なさそうですね。

三岩：ええ、ありません。5年前に現在の職場（神奈川障害者職業能力開発校）に転勤して、そこで教える立場として、マイコンシステムと接することになったんです。

――それがマイコンシステムとの初めての出会いだったのですか？

三岩：ええ。最初はWindows版の電子回路設計ソフトや、マイコンプログラミング用のソフトを使って勉強しました。

――それからたった3年の間で、独自にツールを開発されるようになられたわけですね。才能を感じてしまいます。

三岩：そうですか（笑）。マイペースで世の中にないものを作ってきて、気が付いたらほかに誰もいなかったんですが。

――Linuxと出会われたのはいつごろですか？

三岩：3年くらい前のことです。面白そうなのでインストールしてみて、3、4カ月はいわゆる「インストールユーザー」でした。

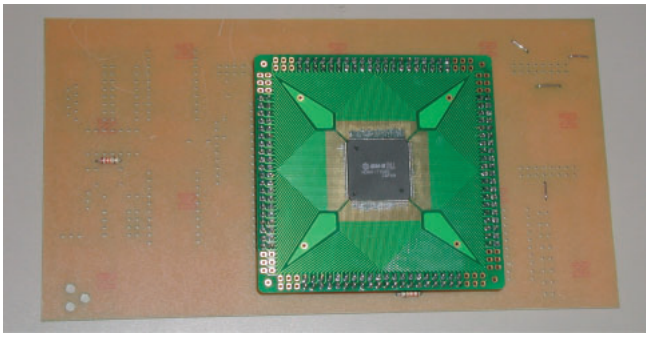


写真2　三岩氏が試作中のSH-3ボード。H8の周辺機器で蓄積したノウハウをSH-3に移行したいと三岩氏は考えている。

――Linux上のツールの開発を始められたきっかけは何だったのでしょうか？

三岩：学校の予算が潤沢でなかったから、フリーの環境を利用して教育用の環境が作れないかな……と思ったのがきっかけです。Linuxはフリーで、しかもプログラム環境が含まれていますから。

学校で、Z80ベースのマイコンシステム開発をWindows環境を利用して教えていたのですが、ある時、これはLinuxだけでできそうだと思いました。やってみたらできたというわけです。

――どういったところから始められたのでしょうか？

三岩：Z80向けの開発ツールとして、秋月電子通商から出ているAKI-80というROMライタがあるんです。PCのシリアルポートに接続して使うタイプなんですけど。ある時、FreeBSD上で動くAKI-80のコントロールソフトが公開されているのに気がついたんです。ところがLinuxで利用しようとすると、BSDとLinuxではシリアルポートの扱いがまったく異なるので、シリアルポートの制御部分をLinux向けに作りなおさなくちゃいけないという問題があることがわかりました。それからSerial-HOWTOなどを読んで理解して、シリアルポートを使うプログラムを試作してみて、FreeBSD上のコントロールソフトのシリアルポート関連部分を置き換えてみたら、とりあえず動いたんです。それがきっかけでしたね。

――予算が潤沢でないという制約から、そうやってツールが生まれてきたんですね。

三岩：そうです。それに同じものでも、開発コストを下げることではじめて使えるようになるということがありますね。マイコンの場合は特に、コストダウンによって応用面が広がるという側面がありますし。

――そのあたり、Linuxがフリーであればこその醍醐味ですね。開発を行う立場で、LinuxとWindowsの違いをどのようにお感じになりますか？

三岩：マイクロソフトのツールは、マイクロソフトが想定している用途には非常に簡単に使えるんです。だけど、その範囲外の用途に利用しようとすると非常に難しくなってしまいます。ツールが想定していない用途に対応するのが難しいのはLinuxも同じなんですが、マイクロソフトのツールの場合、その落差が激しくてLinuxの比ではないですね。また開発環境も高価なので、予算が潤沢でない環境では使いにくいです。その点、Linuxはインストールすれば開発環境も含めて、すぐ使えていいですね。

―開発する人間にとって、開発環境がフリーであるということのメリットは大きいですものね。

三岩：それに組み込みシステムの場合、「裸」のハードウェアを扱うわけですから、開発環境がオープンソースでないと厳しい面があるとも言えます。ただ、マイクロソフトのツールが現在主流ですので、個人的にはVisual Basicを自在に扱うくらいのスキルはつけたいですね。

―お作りになったツールを、世の中でどんなふうに利用してほしいと思っていらっしゃいますか？

三岩：自分のために作って、ついでに公開しているという形なのですが……。基礎的なツールなので、Z80やH8の開発者すべてに利用してもらえればと思っています。商業的にライセンスをどうこうしたいとはあまり思いませんが、改造して隠して使うということだけはしないでほしいですね。

―ツール自体で報酬を得るというお気持ちはないのですね。

三岩：それは無理です。私、公務員ですから（笑）。ただ、1人でも多くの方が、Linuxで開発を行うようになってくれたらいいと思っています。Linuxでの開発が普及すればメーカーも無視できなくなるでしょうし、自分よりできる方がいろんなことをやってくれたら楽しいと思いますし……。公開したツールが普及すること自体が、自分にとってメリットがあると思っています。現在、自分と同じような基礎的なツールを供給している企業はありませんし。

―今は「Linuxで開発がやれます」ということを、これから世間に認知してもらうという段階ですね。

三岩：そうですね。世の中がLinuxに流れてくるといいな……と思っています。

―ところで三岩さんの「Linux研究所（http://www.linet.gr.jp/~mituiwa/）」を拝見すると、マイコン開発ツール以外にもいろんなツールやアプリケーションの開発をしていらっしゃいますね。

三岩：ええ。世間にないものを作っていこうとしているうちにそうなってしまいました。たとえば、Linuxネイティブなワープロはまだ少ないので開発しようとしています。

―そうですね。私も思い浮かぶものがオムロンソフトウェアのdp/NOTEくらいしかありません。

三岩：ええ、Javaですとか、Applixwareみたいに中間言語を使うものですとか、HancomWordのようにWineをベースにしているものはあるんですが、Linuxネイティブなものは少ないんですよ。Linuxネイティブな英語版のものを日本語化する手もありますが、日本語特有の部分の機能がどうしても貧弱になりがちですしね。だから、そういったものを、フリーでなおかつオープンソースで配布すれば、市販のツールに比べて貧弱でも広まるでしょうし、市場で販売される製品にも影響を及ぼすだろうと思うんですよ。

―ディストリビューションは開発されないのでしょうか？

三岩：あ、私、Plamo Linuxの開発に関わっています。Plamo Linuxには、私の開発したマイコン関連のツールがいくつか採用されています。

―本当に広くご活躍していらっしゃるのですね。これからやろうと計画していらっしゃる活動には、どのようなものがありますか？

三岩：自分の作った作品の情報をキット会社に提供して、製作・販売をしてもらおうと思っています。私自身は公務員なので、そういう営利活動をするわけにはいきませんので……。最近の作品には、H8ベースのミニネットワークサーバがあるんですが、TCPやHTTPを搭載可能で、PPPを入れればダイヤルアップサーバにもなって、もちろんLinux上のネットワークプログラムと通信が可能なんです。H8ベースなのでコストは3000円くらいでできてしまうんですよ。

―それは凄いですね。それにH8だと発熱の問題もあまりありませんね。

三岩：そうなんです。あと、H8向けの開発ツールは一通り作りましたので、今度はSH-3向けのツールも開発したいと思っています。H8の周辺機器ノウハウを移行して、パソコン用の通常のキーボードを接続して使えるようにしたいですね。マイコンによっては専用のキーボードがメーカーから販売されているんですが、それは通常のパソコン用のキーボードより高価なんですよ。

―コストを抑えなくてはならないというマイコンシステムの宿命が、次から次に新しいものを生み出していっている感じですね。そこにLinuxとオープンソースの思想が加わることで、これまでになかった動きが生まれそうな予感がします。今日は、楽しいお話をありがとうございました。

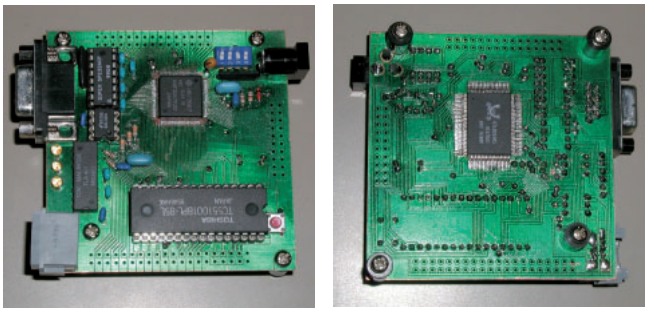
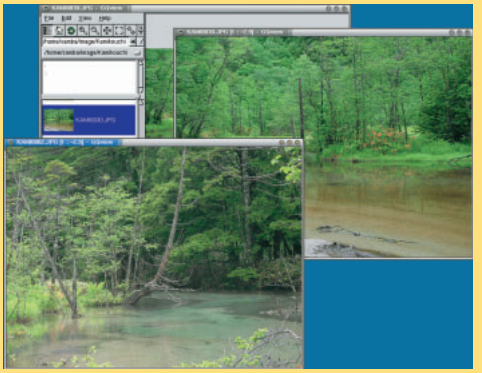


写真3　三岩氏が開発した、H8/3067Fマイコンを使ったネットワークサーバ。TCPやHTTPが搭載可能で、Linux上のネットワークプログラムと通信が可能。裏面にNE2000相当のコントロールチップが実装されている。

Free Application Showcase

文：出井 一
Text : Hajime Dei



GQview　P.152

Xplanet　P.154

GnomerMind　P.158

※紹介したソフトは、すべて付録CD-ROMに収録されています。

あの天文シミュレーションソフトが日本語表示に

Xplns

バージョン：3.2.0　　ライセンス：フリー

http://www.astroarts.com/products/xplns/index-j.html

Xplnsは、星空をウィンドウに表示する天文シミュレータだ。アストロアーツの「StellaNavigator for Windows 1.0」のサブセットで、高速・高精度な天文計算エンジンを利用している。地平座標や星座早見、太陽系モードといった表示モードを持ち、恒星や星雲、惑星などの運行を正確にシミュレートする。今回の国際化対応により、メニューやダイアログなどが日本語で表示されるようになった。

●バイナリのインストール

Xplnsは、StellaNavigatorの商用コードを含んでいる関係で、バイナリ形式の配布のみ行われている。glibc2を使うRed Hat系ディストリビューションでは、RPMバイナリパッケージを利用すればいい。

提供されるRPMバイナリパッケージは、Xplns本体（xplns）と追加恒星データ（xplns-cat）、起動要素データ（xplns-elm）、画像データ（xplns-img）の4つ。「su -」としてrootになった状態で、これらを「rpm -Uvh xplns-＊3.2.0-1.i386.rpm」としてまとめてインストールしよう。

一方、RPMが採用されていないディストリビューション（DebianやPlamoなど）では、Alien（http://www.kitenet.net/programs/alien/）を使って、RPMバイナリパッケージをDEBやTGZパッケージに変換するといいだろう。

●ロケールファイルLC_CTYPEに注意

今回のバージョンから国際化対応が行われ、ロケール設定に応じてメニューやダイアログ、星座名などが英語・フランス語・日本語で表示される。ただし、いくつかの国産ディストリビューションの日本語ロケールのLC_CTYPE（/usr/share/locale/ja_JP.eucJP/LC_CTYPE）に不具合があり、メニューなどが文字化けしてしまう。

筆者が試した限りでは、Kondara 1.1および2000、Plamo 2.1などでこの問題による文字化けが発生した。一方、Vine 2.1以降では正常に表示されることを確認している。

本誌付属CD-ROMには、Vine 2.1付属のLC_CTYPEを収録している。Red Hat 6系ディストリビューションでは、rootになった状態で、このファイルを/usr/share/locale/ja_JP.eucJP/LC_CTYPEに上書きコピーすることで、文字化けが解消されるはずだ。

なお、文字化けを解消できない場合でも、環境変数LANGの設定や、オプション設定などによって、以前と同じ英語表示で利用できる。

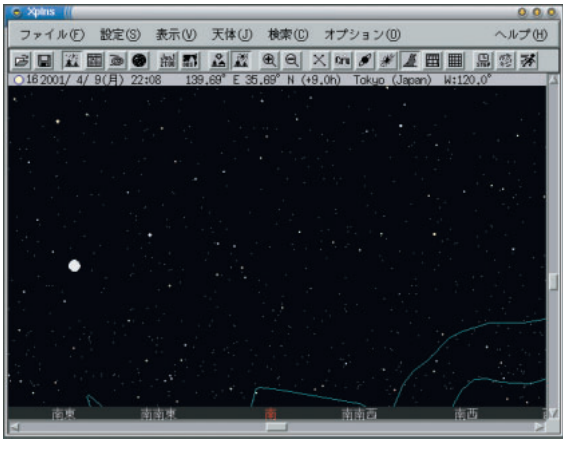
●現在位置を設定する

ktermなどのコマンドラインで「xplns&」とすると、Xplnsのウィンドウが開いて、現在の星空が表示される（**画面1**）。なお、メニューなどが文字化けするなら、「LANG=C xplns&」として起動すればいい。以下の説明や画面はすべて日本語表示の場合だ。

星空を正確に表示するには、現在位置を設定する必要がある。ツールバーの[場所]ボタンでダイアログを開き、リストから「Tokyo（Japan）」を選択して[OK]ボタンを押そう（**画面2**）。その際、[デフォルトにする]をチェックしておけば、次回からはこうした操作を行う必要はない。

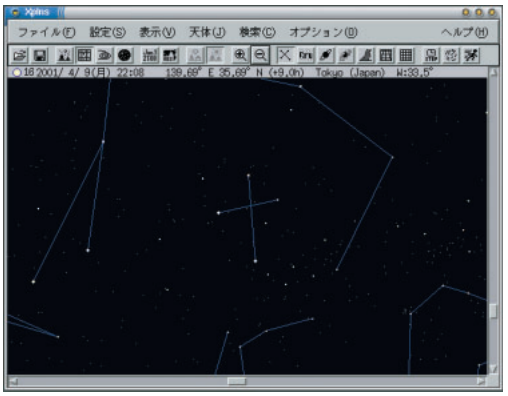
●表示モードを豊富にサポート

Xplnsには、どのように星空を画面に投影するかを決める「表示モード」が全部で6つ用意されている。以下では、代表的な4つの表示モードについて説明しよう。これらは、ツールバー左から3〜6番目のボタンでいつでも切

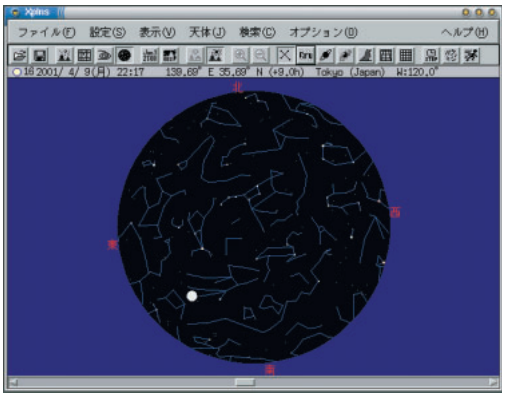


画面1　Xplnsのウィンドウ。メニューなどが日本語表示されている。

画面2　現在の場所をリストから選択する。場所の新規追加も可能だ。



画面3　「赤道座標モード」で南天の南十字星を画面中央に表示してみた。



画面4　星座早見盤風に全天を円形に投影する「星座早見モード」。

り替えられる。

・地平座標モード（初期設定）

現在位置から見た星空を半球に投影する（**画面1**）。地平線や方位が表示されるので、実際に星空を眺める際の比較対象として使うと便利だ。

・赤道座標モード

地平線より下の星空を見たくなったら、全天の星空を球に投影して中心から眺める赤道座標モードに切り替えよう（**画面3**）。地平線に邪魔されずに、あらゆる方向の星空を表示できる。

画面6　星空の表示日時はこのダイアログで自由に変更できる。

・星座早見モード

星座早見盤のように、現在位置から見た星空を円内に投影する（**画面4**）。特定の星座の位置を素早く見つけたい場合に使うといい。

・太陽系モード

他の表示モードと違って、太陽系を外部から眺められる（**画面5**）。指定した日時の惑星の位置関係や、惑星の公転のようすを眺めるのに最適だ。

なお、地平座標モードと星座早見モードでは、太陽の位置によって画面表示が変化する。初期設定では、日の出前から背景（空）がしだいに明るくなり、日中は青空となって太陽や月などしか表示されない。

もし、昼間でも星空を眺めたいなら、ツールバー中央付近にある[昼光]ボタンをオフにすればいい。一方、赤道座標モードと太陽系モードでは、[昼光]ボタンの状態に関係なく常に星空が表示される。

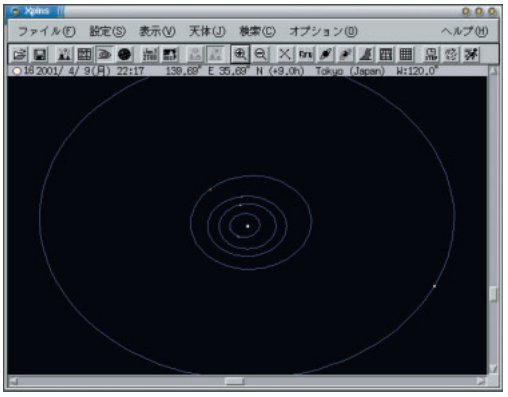
●表示範囲や日時の変更

どの表示モードでも、右と下のスクロールバーをドラッグすることで、星空の表示がスムーズにスクロール（あるいは回転）する。星空自体をドラッグしてスクロールすることも可能だが、こちらはボタンを離すまで画面が更新されない。

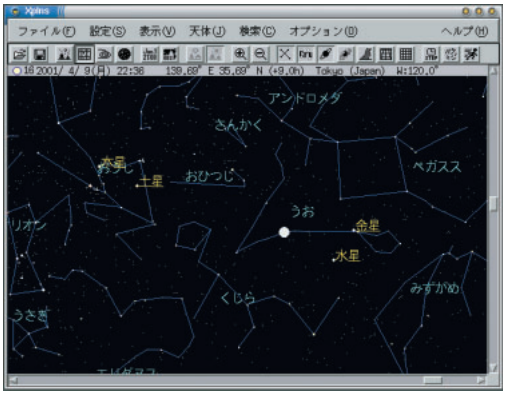
表示範囲を拡大・縮小するには、ツールバー中央のルーペのボタンを押せばいい。また、ウィンドウ自体のサイズを変更すると、星空の表示も自動的に追従する。

このほか、[表示]メニュー以下の項目により、視野角（初期値120度）を変更したり、方位（東西南北）を選択したり、表示を180度回転するといった操作を行える。

なお、通常はXplnsを起動した日時



画面5　太陽系の惑星などを外部から眺める「太陽系モード」。



画面7　星座線と星座名、惑星名の表示をオンにした状態。

画面8　惑星や一部の星雲などには、こうした画像が用意されている。

の星空が表示されるが、[設定]－[日付/時刻]で開くダイアログで、現在時刻や指定した日時の星空に切り替え可能だ（**画面6**）。

●星座線や惑星名などを表示

ツールバー中央左寄りのボタングループは、星座線や星座名、惑星名、星雲、天の川、地平・赤道グリッド（格子線）の表示を切り替えるためのものだ。初期設定では天の川の表示のみオンになっている。

たとえば、[星座線]、[星座名]、[惑星名]ボタンをすべてオンにすると、星座を構成する星と星の間に線が引かれ、星座名や惑星名が日本語で表示される（**画面7**）。

その他の天体（恒星や彗星など）についても、[天体]メニューの各項目で開くダイアログにより、天体自体や名前表示の有無を設定可能だ。

おもな天体をクリックすると、位置や方位などを表示する情報ダイアログが開く。たとえば、惑星の場合は、位置や等級、直径、衛星数などのデータが表示される。

さらに、ダイアログ右上のボタンを押すと、その天体の画像が別のウィンドウに表示される（**画面8**）。こうした

画面9　「今日のデータ」では、日の出、日の入の時刻などを確認できる。

画像は、惑星のほか、太陽や月、木星のガリレオ衛星、一部の星雲などにも用意されている。

このほか、[オプション]－[今日のデータ]で開くダイアログには、設定されている日付や位置、ユリウス日（JD）、太陽・月・各惑星の出・南中・没時刻といった情報がまとめて表示される（**画面9**）。

●アニメーション表示を行う

Xplnsでは、時間とともに移り変わる星空のようすをアニメーション表示できる。この機能に関連したボタンは、ツールバー右端のボタングループにまとめられている。

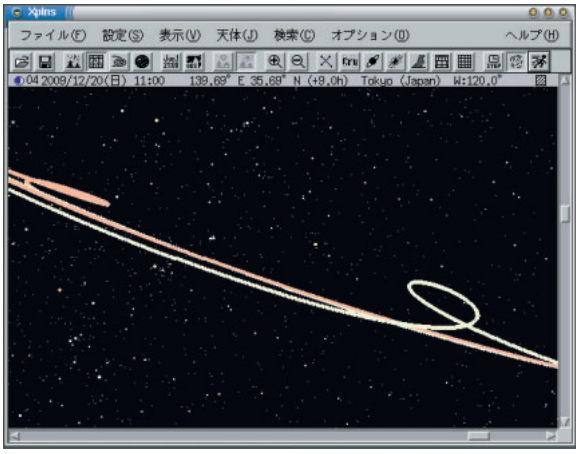
まずは、地平座標モードで恒星の日周運動をアニメーション表示してみよう。ツールバー右端の[プレイ]ボタンをオンにすると、3分ごとの星空の変化がアニメーションで表示される。時間間隔（タイムステップ）を変更するには、ツールバー右から3番目の[タイムステップ]ボタンダイアログを開く（**画面10**）。分単位から年単位まで柔軟な設定が可能だ。

画面10　アニメーション表示の時間間隔はこのダイアログで設定する。

続いては、赤道座標モードで惑星の複雑な動きを観察しよう。タイムステップを6時間程度に変更してからアニメーションさせると、時々逆戻りする惑星の動きを確認できる。

さらに、ツールバー右から2番目の[光跡残し]ボタンをオンにすると、惑星の光跡が画面に描かれる（**画面11**）。太陽や月の表示が邪魔になる場合は、[天体]－[太陽]（あるいは[月]）を選択し、ダイアログでそれぞれの天体自体の表示をオフにすればいい。

このほか、太陽系モードでは惑星の公転をアニメーション表示できる。タイムステップの設定は、水星～火星では6時間、木星～冥王星では10日程度にすると動きがわかりやすい。さらに、[光跡残し]ボタンをオンにすることで、それぞれの惑星の移動量がより明確になる。



画面11　金星と火星の見かけ上の動きが光跡を残す設定でより明確に。

Project Gutenbergのテキストを検索、閲覧する

EText Reader

バージョン：1.0.0　　ライセンス：GPL

http://linux.techass.com/projects/etr/
http://www.gutenberg.net/（Project Gutenberg）

EText Readerは、Project Gutenbergの電子テキストを検索、ダウンロード、閲覧するソフトだ。いちいちブラウザでWebサイトを検索し、ヒットした電子テキストをダウンロードし、展開して表示するといった手間を省ける。閲覧用のビューアは、文字や背景の色、フォントの種類や大きさなどを柔軟にカスタマイズ可能だ。実行にはGUI作成用C++フレームワークのwxWindows（2.2.6以降）が必要となる。

●ビルドとインストール

EText Readerは、tarボールとRPMパッケージが両方配布されているので、使っているディストリビューションに合わせて選択しよう。

EText ReaderのRPMバイナリパッケージには、wxWindowsのGTK＋対応版がスタティックリンクされている。このため、wxWindowsをインストールする必要はなく、EText Readerだけを「rpm -Uvh etr-2.2.6-0.i386.rpm」としてインストールすればいい。

一方、EText Readerのtarボールを利用する場合は、先にwxWindowsのGTK＋対応版をtarボールから導入する。どちらも「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールするという一般的な手順だ。

ただし、wxWindowsをビルドする前にソースを一部修正する必要がある。具体的には、展開先ディレクトリのinclude/wx/protocol/ftp.hの60行目を「const wxString& GetLastResult() { return m_lastResult; }」に変更すればいい。

この修正を忘れてwxWindowsをビルド、インストールした場合は、EText Readerのビルド時にエラーが表示されるので注意しよう。

●Project Gutenbergについて

Project Gutenberg（プロジェクト・グーテンベルグ）は、1971年にMichael Hartが開始したボランティアベースのプロジェクト。著作権が消滅して自由に配布できる古典作品を電子テキスト（EText）として提供する「電子図書館」だ。

現在では、3400以上の電子テキストが提供されており、本家サイトや各地のミラーサイトから自由にダウンロードできる。なお、Project Gutenbergの電子テキストはすべて英文だ。日本語で書かれた電子テキストに興味のある方は、「青空文庫」（http://www.aozora.gr.jp/）を参照されたい。

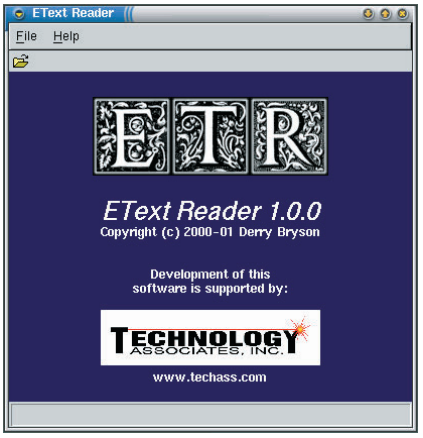
通常、Project Gutenbergの電子テキストを取得するには、Webページで作者名や作品名による検索を行い、結果のページから目的の作品のページに飛んでダウンロードするという手順を踏む。取得した電子テキストの内容は単なるテキストなので、lessなどのテキストビューアやXEmacsなどのエディタで内容を閲覧できる。

しかし、時間帯によってはWebサーバの処理が遅く、検索結果が表示されるまでだいぶ時間がかかることがある。また、技術文書ではなく文学作品を読むのだから、文字や背景の色使いやフォントなどにもそれなりの雰囲気が欲しい。こうした不満はEText Readerによって改善される。

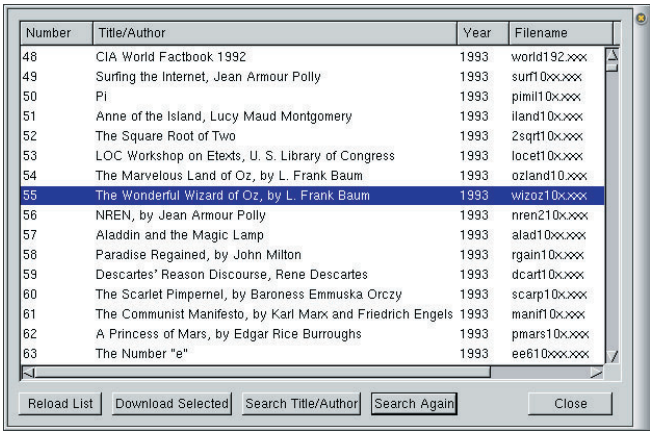
●電子テキストの検索

ktermなどのコマンドラインで「etr&」とすると、EText Readerのメインウィンドウが開く（**画面1**）。

[File]－[Browse Project Gutenberg]を選択すると、現時点での全作品リストが自動的にダウンロードされ、別ウ



画面1　EText Readerは、電子テキストを快適に検索・取得・閲覧する。



画面2　現時点での全作品リストから閲覧したい作品を探し出そう。

ィンドウに一覧表示される（**画面2**）。作品の選択や検索などの操作は、このウィンドウを利用してオフラインでじっくり行える。

作品リストには、通番や作品名／作者名、電子化された年、ファイル名が含まれている。ファイル名以外の項目に関する並び替えも可能だ。さらに、ウィンドウ下部の[Search Title/Author]ボタンで、作者名／作品名に対する文字列検索を行える。

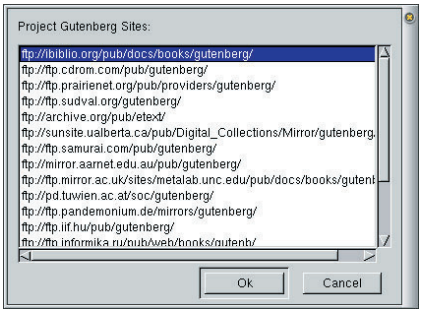
●電子テキストのダウンロード

リストに気に入った作品が見つかったらクリックで選択し、[Download Selected]ボタンを押そう。Project Gutenbergのミラーサイト一覧が表示されるので（**画面3**）、適当なサイトを選択すると、電子テキストの存在確認が行われる。

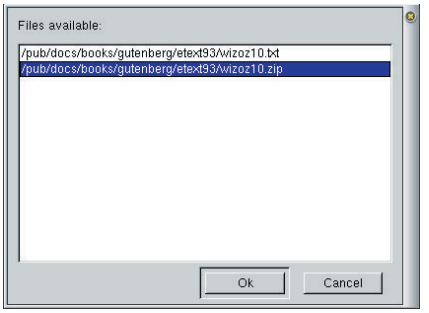
なお、各作品の電子テキストは、未圧縮のテキスト（拡張子.txt）と、圧縮されたZIPファイル（拡張子.zip）の両方で提供されることが多い。

こうした場合は、ダウンロードするファイルをリストから選択する（**画面4**）。EText Readerでは、ZIPファイル内の電子テキストをそのまま閲覧できるため、どちらを選択してもいい。

最後に、電子テキストの保存先ディレクトリ（通常はホームディレクトリ以下のMyLibrary）を指定すると、ダウンロードが開始される。



画面3　電子テキストをダウンロードするミラーサイトを選択する。



画面4　未圧縮テキストまたはZIPファイルを選択してダウンロード。

●電子テキストの閲覧

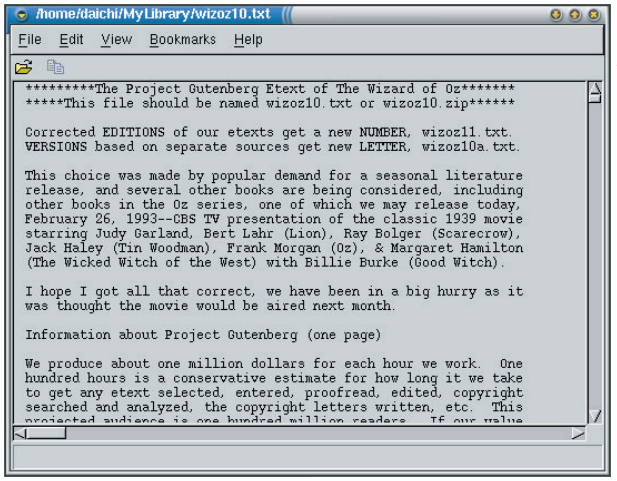
EText Readerの内蔵ビューアには、電子テキストを快適に閲覧するための機能が豊富に用意されている。たとえば、電子テキスト中の任意の位置を記憶するブックマーク（しおり）機能もそのひとつだ。

[Bookmarks]－[Add Bookmark]で現在位置に名前（日本語可）を付けると、[Bookmarks]メニューからその名前を選択するだけで、即座に記憶させた位置に切り替わる。

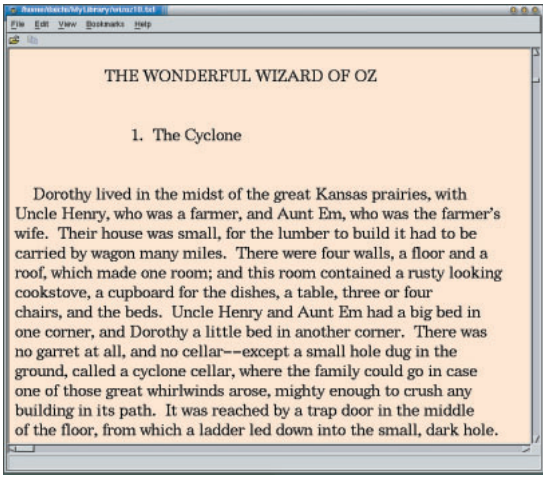
どの電子テキストの冒頭にも、Project Gutenbergに関する説明が必ず含まれているため、実際の作品が始まる位置にブックマークを設定しておくとあとあと便利だ。

このほか、[View]メニューの各項目により、フォントや文字・背景色、行間スペースを変更できる。これらを調整することで、閲覧がさらに快適になるだろう（**画面6**）。

スクロールには、カーソルキーやPageUp／Downキーを使用するほか、マウスのホイールにも対応している。ホイールのないマウスの場合は、[View]－[Mouse Scroll]をチェックするといい。マウスの左／右ボタンを押している間、順／逆方向に連続してスクロールするようになる。



画面5　ダウンロードした作品は即座に内蔵ビューアーに表示される。



画面6　内蔵ビューアはフォントや表示色などをカスタマイズ可能だ。

ワンクリックでOKの画像ビューア

GQview

バージョン：0.11.0　　ライセンス：GPL

http://gqview.sourceforge.net/
ftp://ftp.gnome.org/pub/GNOME/unstable/sources/gdk-pixbuf/（gdk-pixbuf）

GQviewは、ファイル名を一度クリックするだけで画像を表示できる画像ビューアだ。サムネイルやフルスクリーン表示、スライドショー、画像の類似性チェックなどの機能も用意されている。マウスを使わずにキーボードだけでも操作できる点も特長だ。付属の日本語カタログにより、メニューなどの一部が日本語表示される。実行にはイメージライブラリのgdk-pixbuf（0.9.0以降）が必要だ。

●ビルドとインストール

まずは、イメージライブラリのgdk-pixbufから先にインストールする。GQviewの実行にはgdk-pixbuf（0.9.0以降）が必要で、たいていのディストリビューションに含まれるgdk-pixbufはこれよりも古いからだ。

gdk-pixbufの最新版（現時点では0.10.1）は、GNOMEのFTPサイトでtarボールのみ配布されている。以下の手順でRPMパッケージを作成しよう。まず、「su -」としてrootになり、「rpm -ta gdk-pixbuf-0.10.1.tar.gz」とすると、/usr/src/redhat/RPMS/i386にgdk-pixbufとgdk-pixbuf-develパッケージが作られる。これらを「rpm -Uvh gdk-pixbuf-＊0.10.1-1.i386.rpm」としてインストールすればいい。

GQviewもtarボールのみ配布されている。gdk-pixbufとまったく同じ手順（tarボールのファイル名はgqview-0.11.0.tar.gz）でRPMパッケージを作成可能だ。tarボールのまま導入する場合は、「./configure」→「make」としてビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

●キーボードでも操作できる

ktermなどのコマンドラインで「gqview&」とするか、GNOMEメニューの[プログラム]－[グラフィック]－[GQview]を選択すると、GQviewのウィンドウが開く（**画面1**）。

ウィンドウの左側には、上から順に現在のディレクトリ、サブディレクトリ一覧、画像ファイル名一覧が表示される。ツールバー左端のボタンを押すと、画像ファイル名一覧にサムネイル（縮小画像）が追加される（**画面2**）。一方、ウィンドウの右側にはGQviewのロゴが表示されており、画像ファイル名やサムネイルをクリックすると、その画像のイメージが表示される。

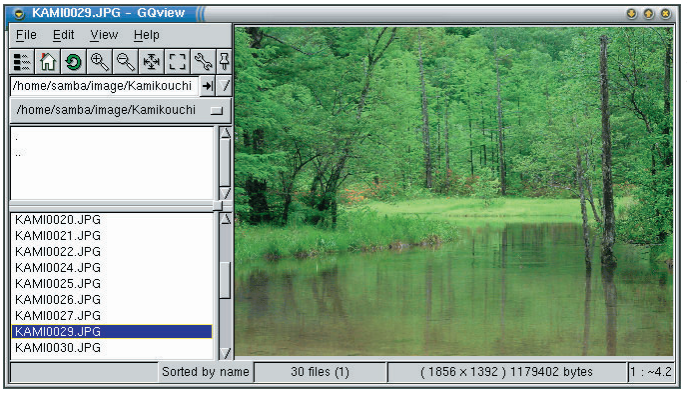
ディレクトリ内の画像を順次眺めるには、画像上でクリックすればいい。前の画像に戻るには中ボタンをクリックする。また、ツールバーのボタンや画像上での右クリックメニューを利用して、画像の拡大や縮小、現在のウィンドウに合わせたサイズ調整（フィット）などが可能だ。

なお、画像が切り替わる際には元のサイズで表示されるが、設定ダイアログの[イメージ]ページの設定（**画面3**）により、自動的にウィンドウにフィットさせることもできる。

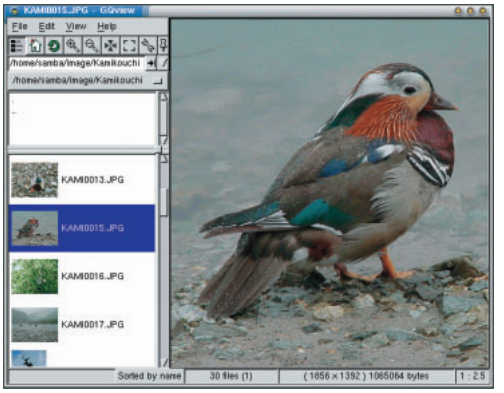
なお、GQviewのほとんどの操作はキーボードにも割り当てられている。たとえば、前後の画像への切り替えはPageUp／Downキー、フィットはXキー、設定ダイアログの表示はCtrl－O（オー）キーといった具合だ。また、Tabキーで画像にフォーカス（黒枠）を移動させれば、カーソルキーによるスクロールや、スペース／BSキーによる前後の画像への切り替えも可能になる。

●フルスクリーンとスライドショー

画像が大きい場合には、右クリックメニューの[フルスクリーン]を選択して（またはVキーを押して）、フルスクリーン表示に切り替えよう。この状態でもマウスやキー操作は有効だ。

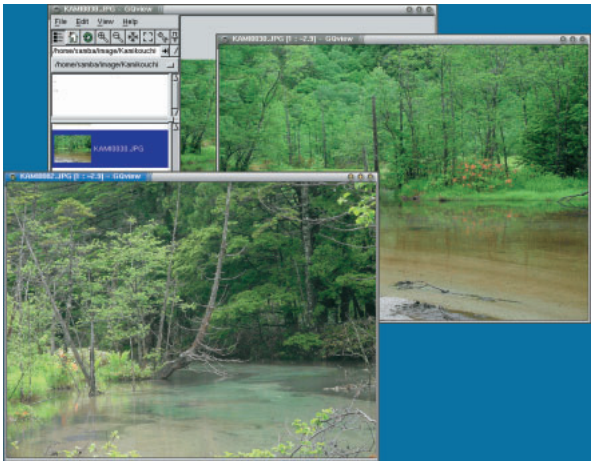


画面1　画像ファイルの一覧が右に、実際の画像が左に表示される。



画面2　一覧表示をサムネイルに切り替えると、このような表示になる。

画面3　画像を切り替える際に自動フィットさせる設定に変更する。



画面4　それぞれの画像を独立したウィンドウに表示することも可能だ。

一方、複数の画像を並べて見るには、右クリックメニューの[新しいウィンドウでみる]を選択して、それぞれの画像を独立したウィンドウに表示すればいい（**画面4**）。

スライドショーを実行するには、画像ファイル名（またはサムネイル）一覧から、対象となるファイルをShift－クリックやCtrl－クリックを使って複数選択する。その後、右クリックメニューの[Start slideshow]を選択する（またはSキーを押す）と、一定時間（初期設定では15秒）ごとに自動的に次の画像に切り替わる。フルスクリーン表示の状態のままスライドショーを実行することも可能だ。

●コレクションの作成と表示

GQviewでは、複数の画像を登録した「コレクション」を作成できる。[File]－[New collection]を選択し（またはCキーを押し）て、白紙のコレクションを作成しよう。

コレクションに画像を登録するには、右クリックメニューの[Append from list]を選択すればいい。GQviewで一覧表示している画像ファイルがまるごと取り込まれる。また、GNOME Midnight Commander（gmc）などから画像ファイルをドラッグし、コレクション上にドロップしてもいい。

コレクションにはサムネイルが縦横に並んで表示され（**画面5**）、ダブルクリックでメインウィンドウに画像が表示される。別ウィンドウに画像を表示したり、コレクションから画像を削除するといった操作も可能だ。

作成したコレクションは、右クリックメニューの[Save collection]で保存でき、メインウィンドウの[File]－[Open collection]（またはOキー）でいつでも読み込める。

●画像の類似性をチェックする

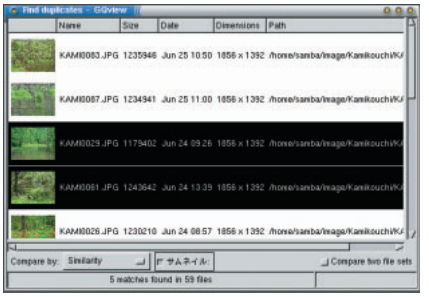
最後に、最新版のGQviewで取り入れられた画像の類似性チェック機能を試してみよう。[File]－[Find duplicates]を選択する（またはDキーを押す）と専用のウィンドウが開く。

チェックする画像ファイルの登録は、gmcなどからのドラッグ＆ドロップで行う。調べたい画像ファイルをまとめてこのウィンドウにドロップすればいい。もちろん、後からでも画像ファイルを追加できる。

左下のリストを「Similality （high）」や「Similality」に切り替えると、画像が類似しているファイルだけが一覧表示される。サムネイルを表示することも可能だ（**画面6**）。また、比較基準をチェックサムや画像サイズ、ファイル名などに切り替えたり、特定のファイル（複数可）に対する類似性だけを調べることもできる。

なお、コレクション内の画像については、コレクションの右クリックメニューから[Find duplicates]を選択するだけで類似性をチェックできる。

画面5　サムネイル（縮小画像）が並んだコレクションを作成できる。



画面6　複数の画像の類似性を調べる。さまざまな比較基準を切り替え可能だ。

リアルな地球や他の惑星を2D／3D表示

Xplanet

バージョン：0.81　　ライセンス：GPL

http://xplanet.sourceforge.net/

Xplanetは、Xのルートウィンドウ（背景）に、地球をはじめとする惑星や月の画像を表示するソフトだ。地表全体をカバーするマップ画像を使って、各種の表示図法によるリアルな画像を表示する。一定時間ごとに画像を再描画したり、設定をGUIで行う関連ソフトも付属している。このほか、ウィンドウ中に惑星を3D表示して、キー操作で自由に回転させることも可能だ（Mesaライブラリが必要）。

●ビルドとインストール

XplanetはtarボールとRPMパッケージの両方で配布されている。ただし、RPMパッケージはRed Hat 7用なので、Red Hat 6ベースのディストリビューションでは、rpmのバージョンを3.0.5以降に更新しないと利用できない。また、RPMパッケージを利用する場合、アニメーション機能が標準で組み込まれるため、Mesaライブラリが必須となる。

tarボールを利用する場合は、「./configure」→「make」としてビルドし、「su」でrootになってから「make install」とする一般的な手順だ。Mesaの使えない環境では、「./configure --without-animation」として、アニメーション機能を組み込まないように指示する必要がある。

なお、FreeType2ライブラリがインストール済みなら、地名（マーカー）表示の際に利用される。FreeType2の有無はconfigureスクリプトが自動的に判別してくれるし、無くても描画が遅くなる以外に問題はない。

●各惑星のマップ画像を入手

Xplanetで惑星の画像を表示するには、地表全体をカバーするマップ画像が必要だ。JPEGやPNG、TIFFなど各種の画像形式に対応している。

XplanetのtarボールとRPMパッケージには、地球の通常マップ画像（earth.jpg）と夜間マップ画像（night.jpg）が含まれている。しかし、他の惑星や月などのマップ画像は同梱されていないので、マップ画像を配布しているWebサイトから取得しよう。

たとえば、「J.H.T.'s Planetary Pixel Emporium」（http://apollo.spaceports.com/~jhasting）には、Xplanet同梱の地球のマップ画像のほか、各惑星のマップ画像（JPEG形式）が用意されている（**画面1**）。画像サイズも手頃なものが多いのでお勧めだ。

ダウンロードした画像は、/usr/share/xplanet/images（tarボールの場合は/usr/local/share/xplanet/images）にコピーする。Xplanetで利用する際に便利なように、ファイル名を「惑星名.拡張子」（たとえばvenus.jpgなど）に変更しておくといい。

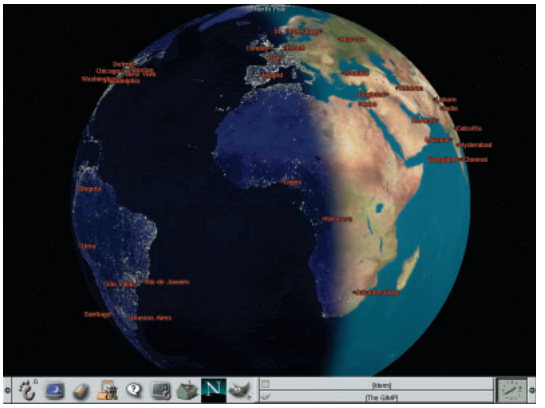
●オプションで表示図法などを指定

Xplanetの各種設定は、起動時のコマンドラインオプションで行う。たとえば、「xplanet -projection orthographic -markers」とすると、しばらくたってからXのルートウィンドウ（背景）に円形の地球の画像が表示されるはずだ（**画面2**）。

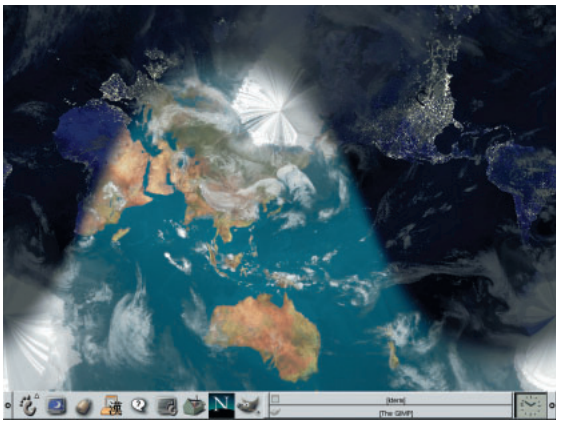
-projectionは表示図法を指定するオプションで、上の例では「正射方位図法」（orthographic）による表示が行われる。これらのオプションや引数は、ほかと区別できる部分まで入力すればいいので、「xplanet -p or -markers」としても同じだ。

このほか、「正距円筒図法」（rectangular）、「メルカトル図法」（mercator）、「モルヴァイデ図法」（mollweide）、「正距方位図法」（azimuthal）、「ピーターズ図法」（peters）による表示も可能だ。また、-projectionを省略すると、正距円筒図法による平面表示が選択さ

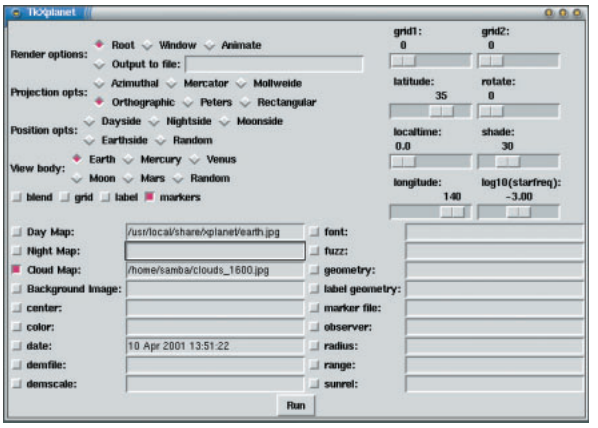
画面1　地球以外のマップ画像はこうしたWebサイトから入手しよう。



画面2　ルートウィンドウに地球の画像を表示する（正射方位図法）。



画面3　メルカトル図法で日本を中心に表示。雲の画像を重ねている。



画面4　GUIで設定を行うtkxplanetでお気に入りの設定を見付けよう。

れる。

もうひとつの-markersは各地の地名（マーカー）を表示するオプションだ。Xplanetには、地球・火星・月のマーカーデータが付属している。

●中心地の指定や他の惑星の表示

このほかにも、表示の中心となる緯度・経度を指定する-latitudeと-longtitude、地球以外の惑星や月を表示する-bodyなどのオプションが用意されている。各オプションの詳細は「man xplanet」で表示されるマニュアルを参照されたい。

たとえば、「xplanet -p or -lat 35 -lon 140」とすると、東京を中心とした地球の画像が表示されるし、「xplanet -p or -bo Venus」とすると金星の画像が表示される。ただし、地球以外の画像を表示するには、それぞれのマップ画像（惑星名.拡張子）を用意する必要がある。

また、地球の雲のマップ画像を重ねて表示するオプション-cloud_imageも用意されている。XplanetのWebサイトには、世界各地の気象衛星データに基づく雲のマップ画像が6時間ごとにアップロードされるので、これを取得するシェルスクリプト（内容はWebページに記述されている）と組み合わせることで、現在の雲の状態を地球に重ねて表示できる（**画面3**）。

●tkxplanetとxplanetbg

コマンドラインでオプションを指定するのが面倒な人のために、Xplanetの設定をGUIで行うツール「tkxplanet」が付属している。ktermなどのコマンドラインで「tkxplanet&」として起動しよう。

ウィンドウには設定用のラジオボタンやスライドバーが並んでいる（**画面4**）。たとえば、東京を中心とした表示にするには、[latitude]を「35」、[longtitude]を「140」付近に設定すればいい。コマンドラインで「-lat 35 -lon 140」と指定した場合と同じ結果が得られる。

さまざまな設定を変更した後で[Run]ボタンを押すと、現在の設定でXplanetが実行され、しばらくすると画像が表示される。

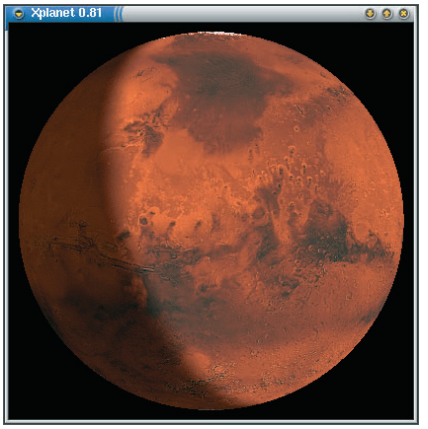
なお、ktermなどの端末画面には、現在のGUI設定に対応するオプションを持つコマンドラインが表示される。これを利用して、現在の設定をシェルスクリプトやエイリアスに登録することも可能だ。

このほか、一定時間ごとにルートウィンドウの表示を更新するツール「xplanetbg」も用意されている。コマンドラインオプションのほとんどはXplanetとまったく同じだ。初期設定では5分ごとにXplanetを呼び出す。この時間間隔は、独自のオプション-waitで秒単位に変更できる。

●ウィンドウ表示でアニメーション

Xplanetを-windowや-animate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、-animateを指定した場合は、Mesaライブラリを利用した3D表示の惑星が自転するようすを眺めることができる（**画面5**）。

さらに、カーソルキーで眺める角度を変えたり、Home／Endキーで拡大／縮小したり、＋／－キーで自転速度を制御したりできる。キー操作の詳細は、hキーでウィンドウ上に表示されるヘルプを参照されたい。



画面5　通常のウィンドウに火星の3D画像をアニメーション表示。

複数ファイルを効率よくダウンロード

Downloader for X

バージョン：1.25　　ライセンス：Artistic

http://www.krasu.ru/soft/chuchelo/

Downloader for Xは、HTTP／FTPを利用してファイルのダウンロードを行うソフトだ。指定されたURL以下のファイルを再帰的に取得したり、ファイル種別を限定することも可能だ。また、レジューム転送をはじめ、巨大なファイルの分割受信機能や、指定時刻に受信を開始するスケジュール機能などさまざまな機能を備えている。日本語環境ではメニューなどが日本語で表示される点もうれしい。

●ビルドとインストール

Downloader for Xは、tarボールとRPMパッケージの両方で配布されている。RPMバイナリパッケージはRed Hat 6.2用と7.0用がそれぞれ用意されているので、使っているディストリビューションに合わせて選択しよう。

また、RPMソースパッケージからバイナリパッケージをリビルドすることも可能だ。「rpm --rebuild nt-1.25-1.src.rpm」とすると、/usr/src/redhat/RPMS/i386にバイナリパッケージが作成されるので、「rpm -Uvh nt-1.25-1.i386.rpm」としてインストールすればいい。

一方、tarボールを利用する場合は、展開先のmainディレクトリに移動して「make」でビルドし、「su」でrootになってから「make install」でインストールする。

●起動とファイルの追加

ktermなどのコマンドラインで「nt&」とするか、GNOMEメニューの[プログラム]－[インターネット]－[Downloader for X]を選択すると、上下2ペイン構成のウィンドウが開く（**画面1**）。上部にはファイルリスト、下部にはメインログが表示される。

ファイルをリストに追加する方法は、いくつか用意されている。もっとも簡単なのは、Netscapeなどでブラウズしているページ中のリンクをドラッグし、Downloader for Xのウィンドウにドロップする方法だ。自動的にプロパティダイアログ（**画面2**）が開いて、URLが設定される。

このほか、ツールバー右端のボタンで画面上に表示される「ドラッグ&ドロップバスケット」（**画面3**）にもリンクをドロップできる。また、クリップボード監視機能を有効に変更した場合は、他のアプリでURLをクリップボードにコピーした段階でプロパティダイアログが開く。

なお、プロパティダイアログの[メイン]ページでは、ファイルを保存する「セーブフォルダー」や、URLとは別の名前で保存する「セーブファイル」なども設定可能だ。

●ファイルをダウンロードする

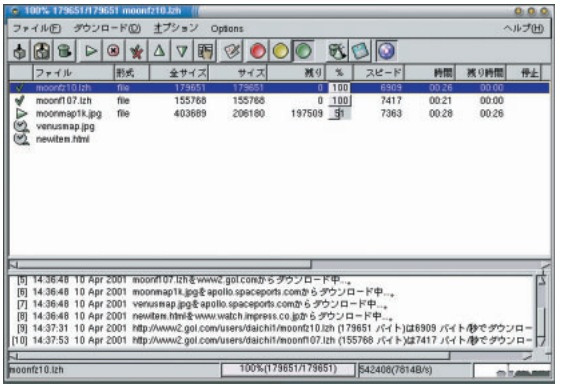
プロパティダイアログの[了解]ボタンを押すと、ウィンドウ上部のファイルリストに追加され、リストの先頭から順番にひとつずつファイルがダウンロードされる。なお、同時にダウンロードするファイル数はオプション設定で変更できる。

ダウンロード中は、ファイルごとに受信済み・残りサイズなどが数値やグラフで表示される。各ファイルの右クリックメニューから、ファイルごとの詳細なログの参照（**画面4**）や、プロパティの変更が可能だ。

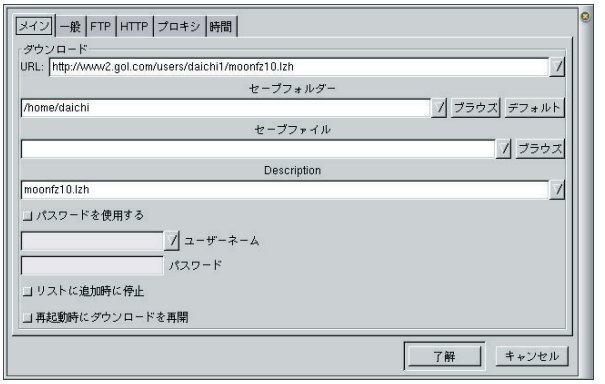
このほか、ツールバーのボタンを利用して、リスト中のファイルの順番変更やリストからの削除、ダウンロードの停止・再開などを行える。

●Webページをダウンロードする

Webページに含まれる画像やリンク先のページを取得するには、プロパティダイアログの[HTTP]ページで、[再帰取得する深さ]を変更する（**画面5**）。設定値は0以上の整数で、「0」は制限



画面1　上部にダウンロード対象のファイルリストが表示される。



画面2　ファイルのURLなどを入力するプロパティダイアログ。

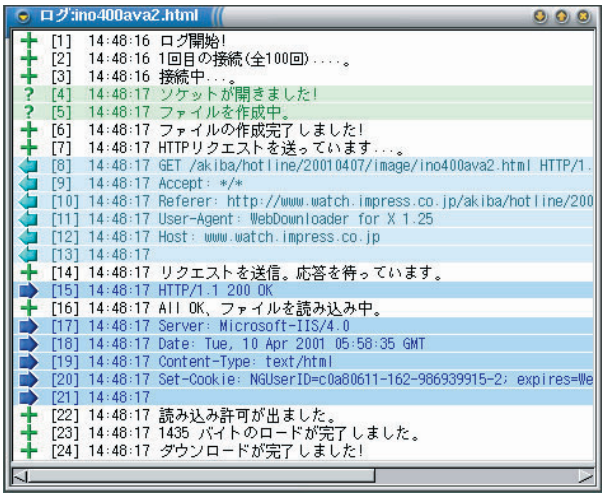
なしの再帰取得、「1」は再帰取得しないことを意味する。

初期値は1なので、URLで指定したファイルしか取得しない。このため、Webページに含まれる画像やリンク先ページを取得するには、2以上の値を設定する必要がある。

なお、再帰取得を行う場合、指定したURL（ドメイン部分を除く）に基づいたディレクトリ構造がセーブフォルダに作成される。たとえば、「http://www.hoge.com/users/hogehoge/index.html」の再帰取得を行うと、セーブフォルダにusers/hogehogeディレクトリが作られ、そこにindex.htmlなどが格納されるわけだ。

これらのファイルをNetscapeなどのブラウザで開けば、Webページをオフラインで閲覧できる。なお、[HTTP]ページの[Change links in HTML files to local]をチェックしておけば、ダウンロードしたHTMLファイル内のリンクの書き換えも行われる。

画面3　リンクのドラッグ&ドロップを受け付ける小型の「バスケット」



画面4　各キューごとにダウンロード時のログが保存されている。

●そのほかにも便利な機能が満載

FTPを利用したダウンロードについても、プロパティダイアログの[FTP]ページで再帰取得の設定が可能だ。FTPサイトのディレクトリの内容を、サブディレクトリ以下も含めてまるごと取得できる。

また、FTPの場合は、URLの末尾にワイルドカードを含むファイル名を指定することで、取得するファイルの種別を制限できる。たとえば、URLの末尾に「*.jpg」を付けると、JPEG画像だけを取得できるわけだ。

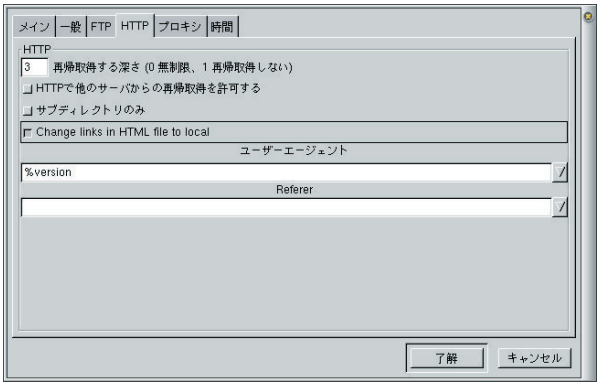
時間帯によっては接続先のサーバが混雑していて、つながらなかったり、ダウンロードに時間がかかり過ぎることがある。そのような場合は、[時間]ページでダウンロード開始日時を設定するといい。この設定を有効にしてファイルをリストに追加すると、Downloader for Xは指定された日時になるまで待機状態になる。

一方、xDSLやCATVなどを導入してバンド幅に余裕がある環境では、巨大なファイルを複数の部分に分割して別スレッドで同時受信する機能を活用するといいだろう。[一般]ページの[複数の部分に分けてダウンロード]の数値を変更すればいい。

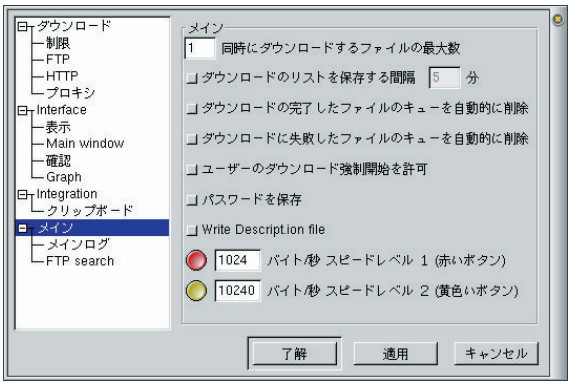
●ジャンル別のオプション設定

各ファイルのプロパティダイアログで設定した内容は、そのファイルに対してのみ有効だ。全体の設定を変更するには、ツールバーの[オプション]ボタンを押してオプションダイアログを開く（**画面6**）。

設定項目は多岐にわたるため、左側のツリーでジャンル別に分類されている。セーブフォルダの初期値などは[ダウンロード]ページで、表示に関する設定は[Interface]ページで、クリップボード監視などは[Integration]ページで、同時にダウンロードするファイル数などは[メイン]ページで設定を行う。



画面5　Webページ中の画像などを取得するには再帰取得の設定が必要。



画面6　さまざまな設定がツリーで分類された設定ダイアログ。

グラフィックが美しいマスターマインド

GnomerMind

バージョン：0.9.0　　ライセンス：GPL

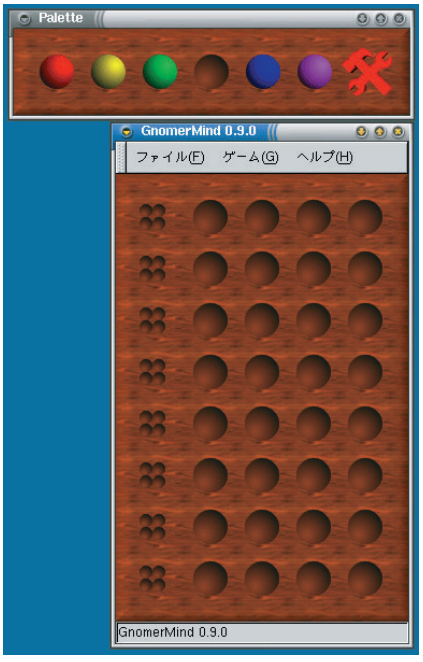
http://gnomermind.sourceforge.net/

GnomerMindは、マスターマインドタイプのパズルゲームだ。6種類のトークン（駒）を4つ並べる順番を推理するだけの簡単なゲームだが、同じトークンが複数使われるため、8ラウンドで正解を見つけるのはなかなか難しい。トークンの種類や並べる数、最大ラウンド数などは柔軟に変更でき、盤面やトークンのデザインを切り替えるテーマ機能も用意されている。実行にはgdk-pixbuf（0.7.0以降）が必要だ。

●ビルドから起動まで

GnomerMindの最新版は、ソース一式をtar＋bzip2したtarボールのみ配布されている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

コマンドラインで「gnomermind&」とするか、GNOMEメニューの[プログラム]－[ゲーム]－[GnomerMind]を選択すると、メインウィンドウと小型のパレットウィンドウが開く（**画面1**）。



画面1
パレットウィンドウに並ぶトークンを盤面に並べよう。

●正しい配置を見つけよう

まず、パレットウィンドウに並んだ6種類の「トークン」（駒）のいずれかを選択し、メインウィンドウの盤面（下列から順番に使用）の4つの穴のいずれかに配置する。注意してほしいのは、同じ種類のトークンを複数の穴に配置できることだ。

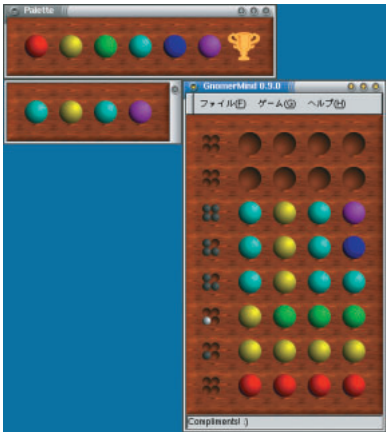
4つの穴にすべてトークンを配置して、パレットウィンドウの右端のボタンを押すと、トークンの種類・位置ともに合っている数と、種類だけが合っている数が盤面左端に小さなアイコンでそれぞれ示される。アイコンの意味が分かりにくい場合は、ステータスバーの表示を参考にしよう。

これらの情報を手がかりとして、8ラウンド以内に正しい配置を見付けるのがゲームの目的だ。正解が見つかると、ファンファーレとともに正解の配置が表示される（**画面2**）。

●画面のデザインをテーマで切替

トークンの種類や並べる数、最大ラウンド数は、デフォルトではオリジナルのマスターマインドと同じ値に設定されているが、設定ダイアログの[Game settings]ページで変更することもできる。また、ラウンド数を無制限に設定することも可能だ。なお、現在の設定値に基づく難易度がダイアログ下部にパーセント表示されるので参考にしよう。

このほか、操作に使用するキー割り当てや、盤面やトークンのグラフィック（テーマ）の切り替えも可能だ。初期値のClassicのほか、ペンギンが登場するPingus、数字で構成されるScribbleなど全部で4つのテーマが付属している（**画面3**）。



画面2
見事に6ラウンドで正解に到達。その秘訣は…。

画面3
テーマを切り替えると、盤面やトークンのデザインが一変する。

国産の縦スクロールシューティングゲーム

Geki2

バージョン：0.9.0　　ライセンス：GPL

http://www2.mwnet.or.jp/~fc3srx7/

Geki2は、縦スクロールタイプのシューティングゲームだ。敵を殲滅する「ボム」を持たないため、雨あられと降ってくる敵の弾の動きを読みきって避けるという熱い弾よけを楽しめる。通常弾とレーザーを切り替え可能だ。全部で5つのステージが用意されており、各ステージの最後には特徴あるボスキャラが待ちうけている。実行には同じ作者のKXLライブラリが必要だ。

●ビルドから起動まで

Geki2とKXLライブラリは、どちらもtarボールのみ配布されている。まずは、KXLライブラリを「./configure」→「make」としてビルドし、「su」でrootになってから「make install」でインストールしよう。Geki2についても、まったく同じ手順でビルドとインストールを行える。

ktermなどのコマンドラインで「geki2&」とすると、ウィンドウが開いてタイトル画面が表示される（**画面1**）。Zキーを押すとプレイ開始だ。

●基本的な遊び方

自機を操作して画面上方から迫り来る敵を倒そう（**画面2**）。操作にはキーボードを使用する。カーソルキーが上下左右への移動、Zキーが武器の発射だ。連射可能なので、Zキーは常に押し続けていればいい。

自機の武器は、威力は低いが広範囲に攻撃できる通常弾（初期値）と、攻撃範囲が狭いが威力の高いレーザーの2種類が用意されている。縦スクロールシューティングゲームの定番といってもいいだろう。

ときどき登場する敵の「タンク」を破壊すると円形のアイテムが現れ、これを取ることで武器の切り替えとパワーアップを行う。アイテムが「S」の時に取れば通常弾が、「L」の時ならレーザーがパワーアップする。通常弾は弾の飛ぶ方向が増加し、レーザーは威力が増加していく。

各ステージの最後には、「お約束」の巨大なボスキャラが待ちうけており、それぞれ特徴ある方法で攻撃してくる（**画面3**）。見事にボスキャラを倒せばステージクリアだ。全部で5ステージが用意されており、すべてクリアすると最初のステージに戻って2周目が開始される。

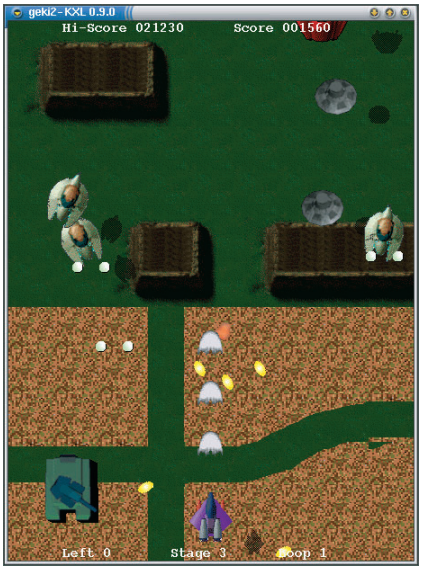
●無限にコンティニュー可能

プレイ中にSキーを押すとゲームの一時停止（再開はZキー）、ゲームオーバー後のタイトル画面でCキーを押すと、最後のステージからプレイを再開できる。

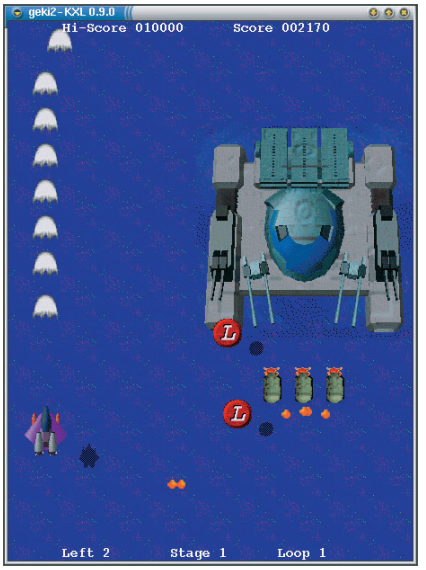
一面の敵でさえ大量の弾を発射してくるという、かなり高めの難易度に設定されており、最初は何度もコンティニューのお世話になるだろう。

なお、作者のWebページでは、このゲーム以外にも、KXLライブラリを利用した横スクロールシューティングゲーム「Geki3」などが公開されている。これらのゲームについても近いうちに紹介したい。

画面1　Geki2のタイトル画面。Zキーでプレイ開始だ。



画面2　敵は怒涛のごとく弾を撃ってくる。動きを見極めろ。



画面3　各ステージの最後には、巨大なボスキャラが待ちうけている。

マニュアルページをGUIで操作

gman

バージョン：0.9.2　　ライセンス：GPL

http://homex.coolconnect.com/user/xkwang/gman/

gmanは、コマンドのマニュアルページの表示や印刷などをGUIを使って行うソフトだ。調べたいコマンドをリストから選択するか、キー入力（インクリメンタルサーチ付き）すると、xtermやgv、Netscapeなどの外部コマンドによってマニュアルページが表示される。一部の設定を変更するだけで、xtermの代わりにktermを利用したり、日本語のマニュアルページを表示することも可能だ。

●基本的な使い方

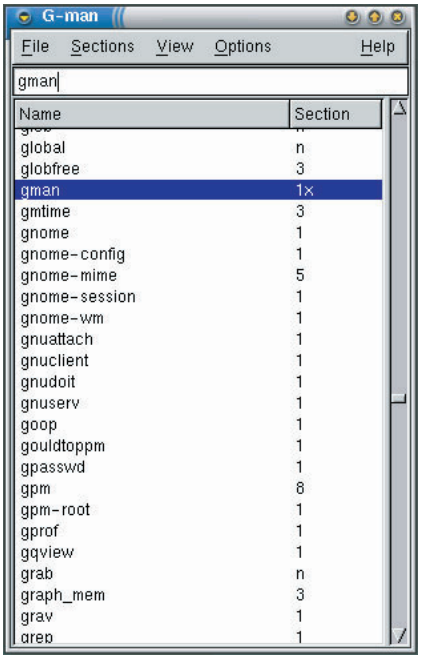
gmanはtarボールとRPMバイナリ、ソースパッケージで配布されているので、ディストリビューションに合わせて選択しよう。tarボールを利用する場合は、「make」でビルドし、「su」でrootになってから「make install」でインストールすればいい。

ktermなどのコマンドラインで「gman&」とするとウィンドウが開く（**画面1**）。初期設定ではシステムコールなどのエントリーも含まれているので、リストは膨大だ。

リストを一般のコマンドだけに絞り込むには、[Sections]－[Only]を選択後、その下の[1: User Commands]だけチェックされた状態にする。また、rootで作業するユーザーは、さらに[8: Sys.Administration]もチェックしておくといいだろう。

コマンドの入力は、リストからクリックで選択するか、上部のテキストボックスに直接キー入力する。1文字入力するたびに検索するため、目的のコマンドが検索された時点でEnterキーを押せばいい。

マニュアルページは、初期設定ではxtermによって表示される（**画面2**）。また、[View]メニューにより、gvによる表示・印刷や、Netscapeでの閲覧に切り替えることもできる。



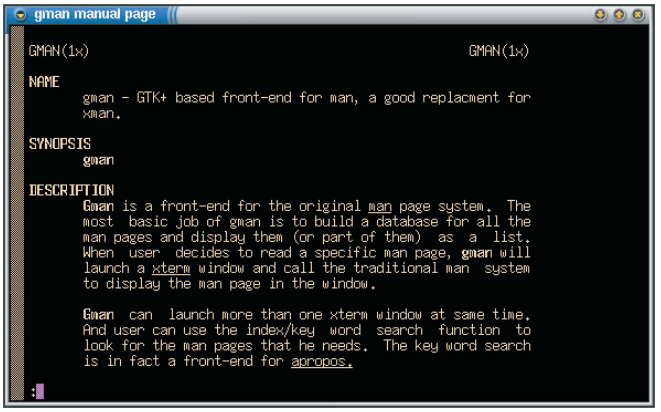
画面1　コマンドやシステムコールの一覧がリストに表示されている。

●日本語のマニュアルページを表示

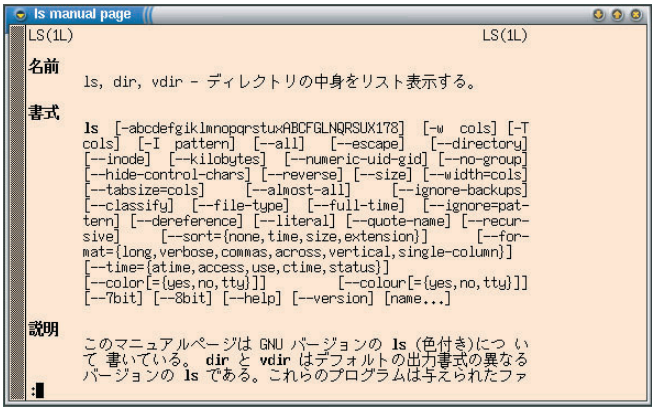
日本語のマニュアルページを追加するには、まず[Options]－[Man Paths]を選択して、マニュアルページのパス設定ダイアログを開く。テキストボックスに「/usr/man/ja」と入力して、[Add new]ボタンを押せば、リストに日本語マニュアルページのエントリーが追加される。同じエントリーが複数並んでいる場合は、下が日本語版だ。

続いて、xtermの代わりにktermを使うように設定ファイルを書きかえる。ホームディレクトリの設定ファイル（.gman）をエディタで編集し、「xterm_command = kterm」という1行を追加すればいい。これで、日本語のマニュアルページも正常に表示されるようになる（**画面3**）。

なお、man2htmlが日本語に対応していないディストリビューションでは、Netscapeでの閲覧時に日本語が文字化けしてしまうので注意されたい。



画面2　選択したエントリーのマニュアルページをxtermで表示。



画面3　ktermを利用すれば日本語マニュアルページの表示も問題ない。

ディスク使用量をグラフで表示

xdiskusage

バージョン：1.43　　ライセンス：GPL

http://xdiskusage.sourceforge.net/
http://www.fltk.org/（FLTK）

xdiskusageは、ディレクトリ単位（またはファイル単位）のディスク使用量を、グラフ表示するソフトだ。コマンドラインベースのduと違って、数段階下のサブディレクトリまで同時にグラフ表示されるため、各ディレクトリの使用比率を感覚的に理解できる。サブディレクトリに切り替えて細かな部分を確認したり、グラフを印刷することも可能だ。ビルドと実行にはFLTKライブラリが必要となる。

●ビルドとインストール

xdiskusageの前に、実行に必要なFLTK（Fast Light Tool Kit）ライブラリをインストールしておこう。

tarボールをそのまま導入する場合は、「./configure --enable-shared」→「make」としてビルドし、「su -」でrootになってから、「make install」でインストールすればいい。

また、tarボールからRPMパッケージを作成する場合は、「su -」としてrootになってから「rpm -ta fltk-1.0.10-source.tar.gz」とすると、/usr/src/redhat/RPMS/i386にfltkとfltk-develパッケージが作成される。これらを「rpm -Uvh fltk-＊1.0.10-1.i386.rpm」としてインストールする。

xdiskusage自身は、ソース一式をtar＋gzipしたtarボールのみ配布されている。「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

●ディスクやディレクトリを選択

ktermなどのコマンドラインで「xdiskusage&」とすると、ディスクやディレクトリを選択するウィンドウが開く（**画面1**）。また、コマンドライン引数で「xdiskusage /home&」のように、直接ディレクトリを指定することも可能だ。

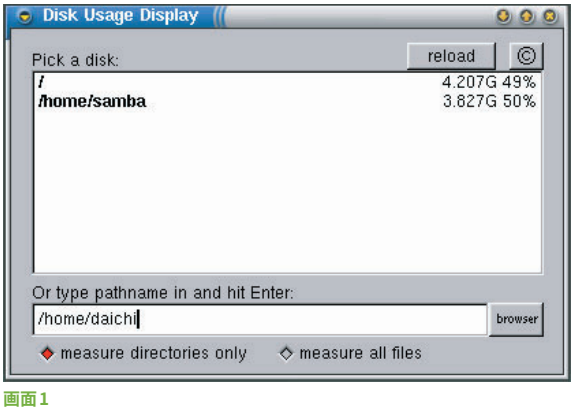
ウィンドウ上部には、各ディスクのマウントポイント先ディレクトリ一覧が表示されている。これらをクリックすると、ディスクから情報を取得した後、別ウィンドウにグラフが表示される（**画面2**）。なお、一般ユーザーではアクセスできないディレクトリがあるため、ディスク全体のグラフを表示する場合はrootになってからxdiskusageを実行しよう。

ディレクトリを指定してグラフを作成するには、その下のテキストボックスを使う。一般ユーザーのまま利用するなら、ここに自分のホームディレクトリ（/home/hogeなど）を入力すればいいだろう。Enterキーを押すとグラフが表示される。

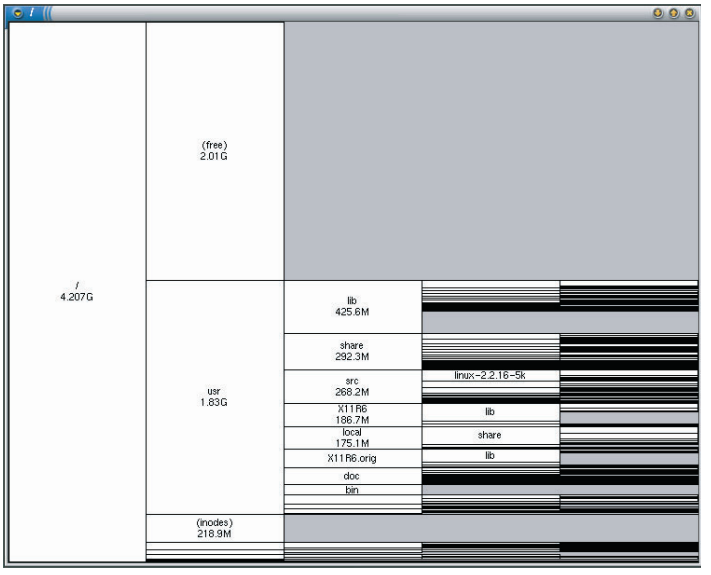
これらのグラフは、複数（初期値は4個）の列に分かれており、左から指定したディレクトリ、そのサブディレクトリ…という順に並んでいる。なお、「(free)」と表示されたグラフは、そのディスクの空き領域を表わす。

グラフ中のディレクトリをクリックすると、グラフの左端がそのディレクトリに切り替わり、さらに下のサブディレクトリが表示される。ウィンドウサイズを広げて、グラフを拡大することも可能だ。

このほか、右クリックメニューにより、グラフの列数（2～11個）や並び順（サイズ順・名前順など）の変更、グラフの印刷、PSファイルに保存などの操作を行える。



画面1
ディスクのリストから選択するか、ディレクトリをキー入力する。



画面2
指定したディレクトリのディスク使用量がグラフで表示される。

Software Catch Up

Javaで書かれた純国産表計算ソフトウェア

Choco

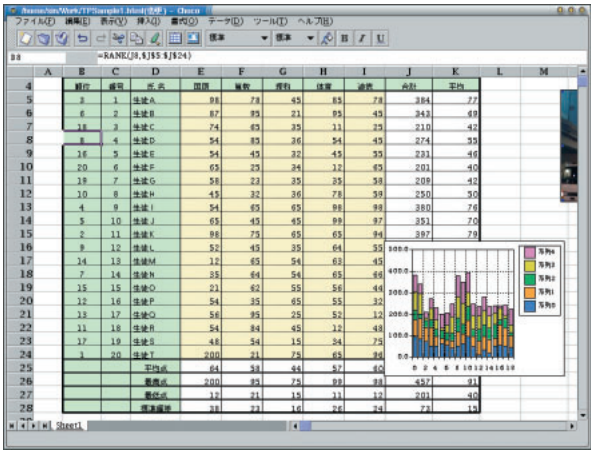
Chocoは、ジャストシステムが開発中の表計算ソフトだ。現在は、Technical Preview版をダウンロードして試用することができる。すでに発売している一太郎Arkと合わせ、新しいオフィススイートとして登場する予定だ。

文：塩田紳二
Text : Shinji Shioda

価格	未定
問い合わせ先	ジャストシステム 03-5412-3939 http://www.justsystem.co.jp/ark/

Chocoは、ジャストシステムが開発中の表計算アプリケーションである（**画面1**）。すでにここでも紹介した一太郎Arkと同じ考え方で作られ、Javaで作成されている。このChocoは、一太郎Arkと、プレゼンテーションソフトであるMuffinの3つを合わせて、いわゆるオフィススイートを形成するわけである。ただし、一太郎Ark以外は現在開発中で、そのTechnical Preview版が公開されているだけである。今回は、このChoco Technical Preview版を紹介することにしよう。なお、これはあくまでも開発途中のものであり、動作はするものの、いくつかの不具合もある。このため、ここではおもにChocoの概要を紹介するにとどめ、あまり細かい評価は行わないことにする。

ちなみに、Chocoは開発中のコードネームで正式名称ではない。一太郎Arkと組み合わされてオフィススイートになることを考えると、製品版は別の名前になるのではないかと思われる。もっとも、ボーランドのKylixのように、コードネームがそのまま製品名になってしまう例もあるので、必ず変わるともいいきれないのだが……。



画面1　表計算ソフトChoco
Chocoは、Javaで記述された表計算ソフトだ。グラフの作成が可能なほか、セルに対する書式の設定なども可能。

Chocoの概要

Chocoはいわゆる表計算（スプレッドシート）ソフトウェアで、一太郎Arkと同じく、すべてJavaで記述されている。現在のサイズは、1Mバイト程度で、この手のアプリケーションとしては非常に小さいといえるだろう。コンセプト的にも、コンパクトなプログラムを目指しているため、製品版のサイズも現状とあまり違わないものになるのでないかと思われる。

また、データ形式としてはXMLを使い、文字サイズなどの属性はCSS2（Cascading Style Sheet）を使って行われる。このため、標準のファイルフォーマットは、XHTMLとなり、ネットスケープナビゲーターなどのWebブラウザを使って開くこともできる（**画面2**。HTMLのTableとして解釈できるような形式になっている）。これにより、ファイルの内容を見るだけならば、Chocoを起動する必要もないしインストールしてある必要もない。ある意味、どこにでも持って行きやすい形式であるともいえる。

また、シート内のデータを使ってグラフを作成することも可能で、基本的な表計算の機能はひととおり備えているといっていいだろう。

セルに対しては、文字サイズやフォント、属性、文字色や背景色、数値などの表示形式を指定することができ、

このあたりは一般的な表計算ソフトと同じ機能を持っている。

ChocoのGUI

Chocoはメニューバーやツールバーなどを持つ、一般的なウィンドウになっている。基本的なGUIはJavaを使っているため、Javaのフレームワークを使ったものになっている。Chocoで使われているJDKについてだが、現時点ではJDK 1.2で検証されており、JDK1.3での動作は未検証となっている。ただし、JDK1.3でも起動はするようである。

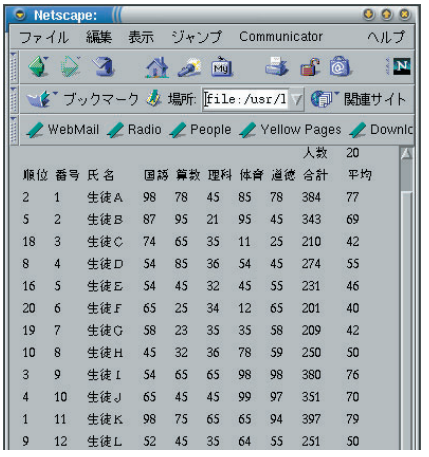
セルへの入力方法は、マウスなどでセルを選択し、キーを打てば入力される。ただし、入力は、ツールバー下の「入力ライン（Microsoft Excelなどでいう数式バーに相当するもの）」で行うが、セルへの直接入力（セル自体に入力カーソルであるキャレットが表示される）もできるようだ。

1つのファイルで複数のシートを扱うことができ、その切り替えはウィンドウ下部にあるタブで行う（**画面3**）。このあたりは、Microsoft Excelと似ている。個人的には、横スクロールバーが小さくなるとか、タブの数が多くなると表示ができなくなるので、この形式はあまり好みではないのだが、すでに多く使われているソフトと同じ形式を踏襲することでユーザーが理解しやすくなるというメリットはある。多数のタブがある場合には、その左端にあるボタンでタブ自体を横スクロールさせて表示する。

このタブ／水平スクロールバーの下にはステータスバーがある。通常、Microsoft Excelでは、ここにキーボードなどの状態が表示されるのだが、今回のバージョンでは何も表示されないようである。これとは別に、さらにこの下にファンクションキーの表示も行える。ただし、現状では機能名は表示されないため、たんなる飾りになっているようだ。

ツールバーは、メニューバーの下のほか、左右にも配置でき、さらにフローティング状態にすることもできる（**画面4**）。また、ボタンサイズを大（**画面5**）、小（**画面6**）の2つに切り替えることができるため、高解像度表示をしている場合でもボタンが見にくくなるようなことがない。

セルの幅や高さの変更は、表の見出し部分の境界線上をマウスでドラッグすることで可能である。また、「書式」メニューの「行」「列」を見ると、「幅」や「高さ」の指定とは別に「最適化」

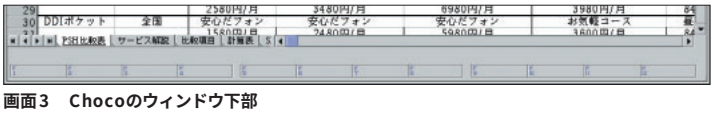


順位	番号	氏名	国語	算数	理科	体育	道徳	人数 20 合計	平均
2	1	生徒A	98	78	45	85	78	384	77
5	2	生徒B	87	95	21	95	45	343	69
18	3	生徒C	74	65	35	11	25	210	42
8	4	生徒D	54	85	36	54	45	274	55
16	5	生徒E	54	45	32	45	55	231	46
20	6	生徒F	65	25	34	12	65	201	40
19	7	生徒G	58	23	35	35	58	209	42
10	8	生徒H	45	32	36	78	59	250	50
3	9	生徒I	54	65	65	98	98	380	76
4	10	生徒J	65	45	45	99	97	351	70
1	11	生徒K	98	75	65	65	94	397	79
9	12	生徒L	52	45	35	64	55	251	50

画面2　Webブラウザでの表示
画面1のファイルをWebブラウザ（ネットスケープナビゲータ）で開いてみたところ。Webブラウザから見ても、ちゃんとしたテーブルとして表示される。

といった項目があり、入力されている文字などのサイズに合わせた調整が可能なように見える。Microsoft Excelでは、見出しの境界部分にカーソルを置き、カーソルの形が変わったところでダブルクリックすると自動的にサイズ調整が行われるのだが、Chocoの最終的な製品でもこうしたショートカット機能を入れておいてくれるとありがたい。

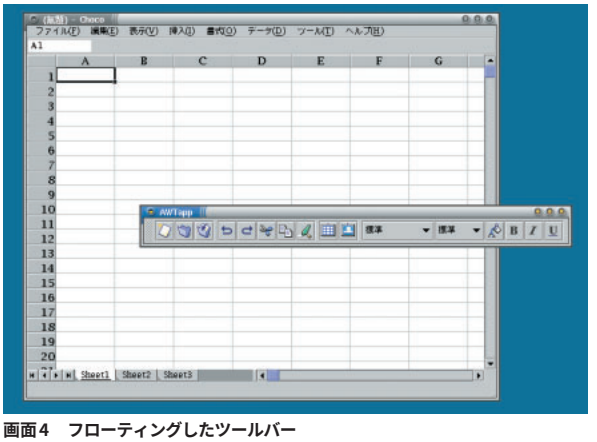
また、多くの表計算ソフトでは作成した表の見出しに相当する部分のスクロールを固定する、「ウィンドウ枠の固定」といった機能があるが、Chocoにもこの機能が用意されている（**画面7**）。実際、画面よりも大きな表を作成する



画面3　Chocoのウィンドウ下部
シートの切り替えは水平スクロールバー横のタブで行う。一太郎ARKと同じくファンクションキーの表示を行わせることもできるが、現状では機能名は表示されない。

画面5　アイコン（大）
ツールバーのアイコンは大、小２つの切り替えが可能で、画面の解像度に合わせて自動で切り替えを行うこともできる。

画面6　アイコン（小）
画面5に比べて、ツールバーのアイコンが小さく表示されている。



画面4　フローティングしたツールバー
標準でメニューバー下に置かれるツールバーは、取り外してフローティングツールバーとして利用することも可能だ。

場合、この機能がないとほとんど実用にならないといってもいいぐらいの機能である。

なお、Windowsなどの多くのソフトで一般的なマウスの右ボタンクリックによるメニュー（ショートカットメニューや、コンテキストメニューと呼ばれるもの）や、選択中のセルを表すセルポインタのドラッグによるコピー機能などは実装されていない。前者は、Readme.txtファイルに「未実装」である旨が記載されており、後者はセルポインタにハンドルが付いていることから、実装する予定はあるものと推測される。

表計算機能

Chocoでは、セルは俗に言うA1形式で指定する。これは、セルの横方向にA、B、C……と番号を振り、縦方向は数値で指定するものだ。かつてLotus 1-2-3が採用していた形式である。

以前、マイクロソフトはMS-DOS用のマルチプランで縦横ともに数値を振り、R1C1という形でセルを指定していた。これをR1C1形式というが、いまではMicrosoft ExcelもA1形式が標準となり、R1C1形式はあまり使われていない。

数式は、Microsoft Excelと同じく、最初に「=」を付けることで数式として認識される。それ以外は、文字列もしくは数値、日付などのデータとして認識される。数式中の関数でセルの範囲を指定する場合、その左上と右下のセルのA1形式アドレスを「：」で繋いで表記する。たとえば、一番左上にあるA1というセルから4×4マスのセル範囲を指定するにはA1：D4とする。

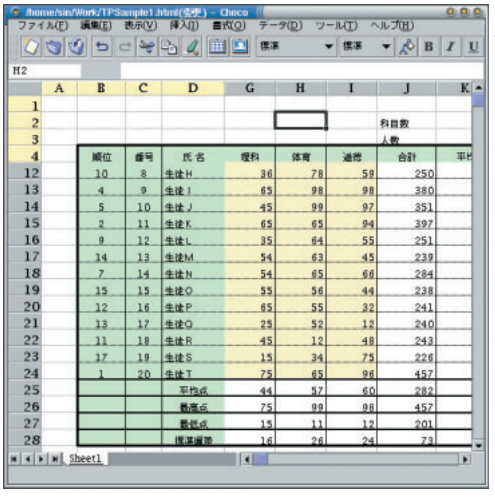
数式中で利用できる関数は、165個（附属の関数を説明するシートファイルで数えた数）あり、特に少ないというわけではない。なお、プラグインを使って関数を定義できるため、必要ならばJavaを使って関数を作ることができる。ただし、これは少し技術を要する。通常の表計算ソフトであれば、たいていはマクロ機能を持っており、これを使って関数を記述することができる。Chocoもこうしたなんらかのマクロ機能を持って欲しいところだが、このTechnical Preview版にはマクロ機能は装備されておらず、ジャストシステムのホームページにも「Technical Preview版に対する要望」として、VBAやマクロなどが要望されている。ジャストシステムも重要な機能と考えてはいるようだが、プログラムをコンパクトにするというChocoのコンセプトと相入れない部分もあって、現在検討中ということになっている。

ただ、ちょっとした条件判断ぐらいであればIF関数を使って行えるし、複雑な数式であってもセルのコピーで簡単に複数のセルへの入力が可能なので、入力の手間を省くために長い数式を関数として記述するといった方法はなくても構わない。実際、筆者はかなり長い数式を利用する場合でも、関数として定義することはあまりしない。もっとも、頻繁に使うようなものであればライブラリ的に関数定義したくなることもあるだろう。

なお、マクロを付けるとなると、ちゃんとした繰り返しや条件判断など、本格的な言語としての体裁が必要で、その編集やデバッグ環境を含めてしまうとかなり大がかりなものになってしまう。ここはかなり悩ましいところであろう。

一太郎ArkのPlugInの解説を見る限り、PlugIn内では組み込み先アプリケーションの内部的なクラスを使ってアクセスを行う必要があり、簡単にいえば、内側が完全に見えていて、それを使って何でも実現してくれという形である。たとえて言えば、Cのプログラムでソースコードを見て、適当に内部の関数を使ってくれといっているようなものである。これに対して、この手のアプリケーションのマクロ機能は、内部構造を見せずに、表やセル、あるいは編集中の文書などを抽象化するように設計されている。このため、原理的には、マクロがアプリケーションをクラッシュさせてしまうようなことができないようになっているわけだ。

一太郎Arkでは、編集対象のデータはジャストシステムが一太郎Ark用に実装したDOM（Document Object Model）を使うようになっている。こ



画面7　ウィンドウ枠固定
作成した表の見出し部分を動かさずに一部分のみスクロールさせる「ウィンドウ枠固定」機能も備わっており、固定される位置の行、列番号の部分が黄色く表示される。

のあたりの構造をうまく使えば、マクロは実装できなくもなさそうである。

各セルには、罫線（**画面8**）、文字色／背景色（**画面9**）、書式（**画面10**）、フォント（**画面11**）などの指定が行える。このあたりはメニューからダイアログボックスを使って行ったり、ツールバーを使う。

前述のようにChocoの基本ファイル形式は、XML＋CSS2であるが、実際のファイルはXHTMLというかHTMLとしても解釈が可能なもので、表をHTMLのTABLEタグを使って表現している。このため、通常のWebブラウザに読み込ませれば、そのまま内容が表示できる。ただし、逆に任意のXHTML／HTMLファイルを読み込めるわけではなく、一定のルールに従って記述されたXHTML／HTMLファイルのみ読み込み可能となっている。

このほか、PlugInとしてMicrosoft Excelのファイル形式の読み書きを可能にするものが附属しており、Windows上などで作成したMicrosoft Excelデータなどもそのまま利用することができる（ただし、Technical Preview版には制限がある）。また、CSVやテキスト形式のファイルを読み込むこともできる（**画面12**）。

ファイルのセーブは、ChocoのXHTML／HTML形式のほか、Microsoft Excel 2000／95形式の3つうちのどれかを選択することができる（**画面13**）。ただし、CSVやテキスト形式での書き出しはTechnical Preview版では行えなかった。

また、Chocoで作成したHTMLファイルは、一太郎Arkで読み込むことも可能である。ただし、一太郎Ark内では単純な表となり、オブジェクトとして配置した画像やグラフなどは文書のうしろに図版として配置される。

グラフ作成

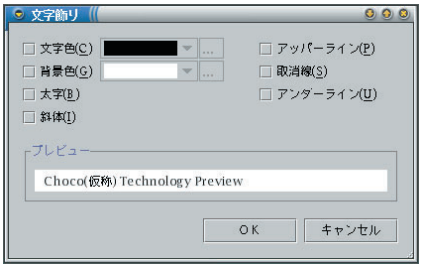
Chocoは、シート内の表からグラフ（ビジネスグラフ）を作成する機能を持つ。グラフの作成は、グラフ化したい範囲をマウスなどで選択しておき、ウィザード形式で設定を行う。手順としては、Microsoft Excelなどと同じである。

グラフの種類は13種類ほどあり、さらにこの個々のグラフ形式に対してパターン（たとえば、棒グラフなら積み上げなのか、割合の表示なのかといったパターン）がある。Chocoではこれを「グラフ書式」と呼んでいるが、会社内のレポートや実験レポートといった、普通の用途であればこれでなんとかなるはずである。

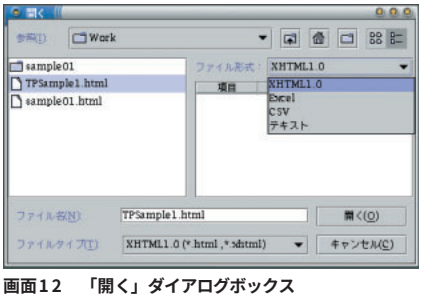
グラフの作成は、以下の手順で行う。

①グラフの種類の指定（画面14）
②グラフ書式の指定（画面15）
③データ系列などの指定（画面16）
④グラフタイトルや軸タイトルなどの指定（画面17）

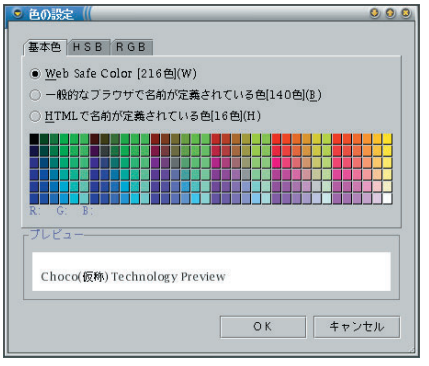
画面8　「罫線」ダイアログボックス
各セルに罫線を表示させることが可能で、ダイヤログボックスを使って罫線を引く場所や線種、色なども指定可能。



画面10　「文字飾り」ダイアログボックス
セル内の文字は、太字やアンダーラインといった属性を指定できる。これらはツールバーやメニューからも個別に指定できるが、文字飾りとして文字色や背景色、属性などをまとめて指定できる。



画面12　「開く」ダイアログボックス
ファイルを開くときには、拡張子がHTMLのXHTML/HTML形式ファイルのほか、Microsoft Excelのデータやカンマ区切り、テキスト形式などが選択できる。



画面9　「色の設定」ダイアログボックス
セルの背景や文字に対しての色指定が可能で、その指定方法も、Webブラウザで一般的に使われる色を選択したり、RGBやHSBで数値として指定することも可能。

画面11　「言語・フォントの設定」ダイアログボックス
Chcocoで利用できるフォントはJavaから利用できるフォントだが、内部で使われている仮想的なフォント名に対して、実際のフォントの対応を指定することができる。

画面13　「名前を付けて保存」ダイアログボックス
ファイルのセーブは、当然、ネイティブ形式のHTMLで行えるが、そのほかMicrosoft Excel2000／Excel95形式も可能。

以上の手順が終了すると、別ウィンドウにグラフが表示される（**画面18**）。ただし、このウィンドウはグラフを表示するだけのもので、作成されたグラフはワークシート上に配置される（**画面19**）。なお、このワークシート上のグラフは、どうやら画像ファイルと同じ扱いのオブジェクトになっているようで、グラフ作成対象となったセルの値を変更してもグラフは変化しない。

画面14　「グラフウィザード対話形式 1／4」ダイアログボックス
グラフウィザードを起動すると、最初はグラフの種類の選択を行う。

画面15　「グラフウィザード対話形式 3／4」ダイアログボックス1
指定したグラフの種類に対して、さらに細かく形式を指定する。この部分は、グラフの種類で違ってくる。

この状態でセーブを行うと、JPEGファイルが別途作成される。このため、グラフのサイズを大きくすると文字のジャギーなどが出てしまい、品質がかなり下がってしまう。このあたりは、XMLによるベクトルグラフィックス記述言語であるSVGなどを使ってほしいところだ。

これが、Technical Preview版のみの状態なのか、最終製品もそうなのかは不明だが、グラフはできあがったのちに元データに修正が加わったり、グラフ形式を変更することもよくある。現状の仕様だと、再度作り直しになり、一部のみ変更するといった作業が困難になる（以前の設定値を完全に覚えている必要がある）。最終的にできあがるものは画像でも構わないが、設定値や元データの範囲などを保存しておいて、

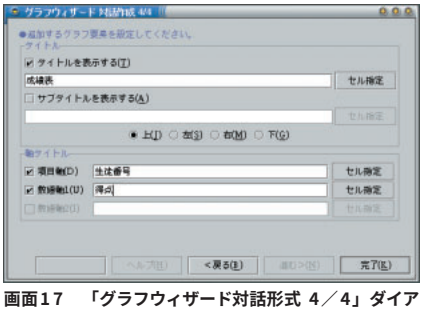
画面16　「グラフウィザード対話形式 3／4」ダイアログボックス2
選択したセル範囲をどうやって表にするか（系列の方向）などの指定を行う。ダイアログボックス内の左側に見えている部分は、最終的にはプレビューとなるのではないかと思われるが、現状では単なる絵が表示されているだけだ。

一部を変更して再度実行できるような機能がほしいところだ。ぜひ、最終製品で対応してほしい。

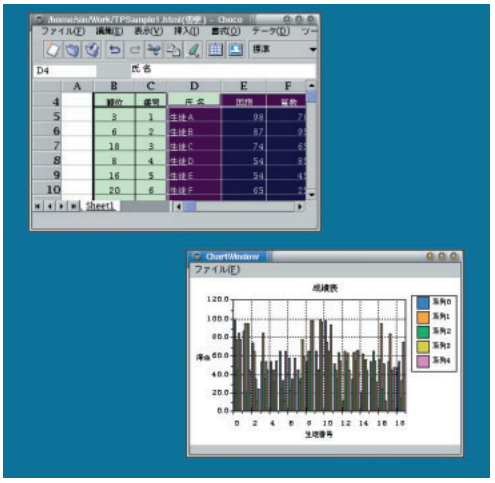
ワークシート上には、グラフのほかに画像を配置することもできる（**画面20**）。これは、ファイルを指定して外部から読み込むが、元画像ファイルに対してリンクおよび、埋め込みの設定がある（**画面21**）。

グラフはともかく、ワークシート上に画像を配置できることは、それほど必要な機能ではないと筆者は考える。あって邪魔になる機能ではないが、多くの表計算ソフトが肥大化している原因のひとつは、表計算ソフト自体で文書を作成できるように、書式やレイアウト機能が多数追加されたためともいえる。

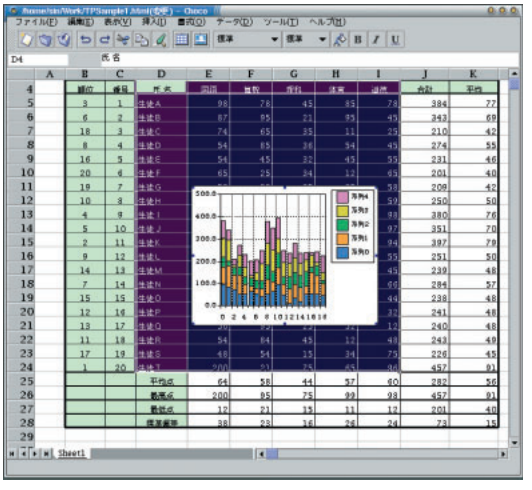
本来なら、表計算ソフトでは表やグ



画面17　「グラフウィザード対話形式 4／4」ダイアログボックス
ウィザードの最終段階では、グラフの見出しやタイトルなどを指定。最後に「完了」ボタンをクリックするとグラフが作成される。



画面18　作成されたグラフ（表示）
作成したグラフは、最初別ウィンドウに表示される。ただし、このウィンドウはグラフを表示する機能しか持たず、メニューも「終了」という項目しかない。



画面19　ワークシートに配置したグラフ
作成したグラフは、ワークシート内のオブジェクトとなる。ただし、最終的に作られたのは単純な画像（JPEGファイル）であり、セルの値を変化させてもグラフは変化しない。

ラフのみを作成して印刷を行う文書などであれば、ワードプロセッサへそれらのデータを貼り付けて作成すべきという気がする。実際、表計算を使った場合、説明用に矢印やテキストボックス（付箋紙のようにセルの脇にメモを付けるなどの用途）などの描画機能があると便利なのだが、単なる画像ファイルだとちょっと使い道は少ないのではないかと思われる。

グラフはXHTML／HTMLファイルとは別の、JPEG画像としてにセーブされる（名前はワークシートファイル名から作られる）。

Javaアプリケーションとしての Choco

Chocoを使ったとき、セル入力などでは特に問題は感じないものの、マウスによる範囲選択などのGUI操作では一部動作がもたつき、たとえばマウスで矩形範囲を指定した状態で、ドラッグしたままマウスをウィンドウ外まで動かすといった動作では、イベントがバッファに溜まってしまうのか、ボタンから手を離してもスクロールが続いてしまう。また、スクロールなども速いとは言い難い。これがCPU（Celeron 450MHz）の処理能力によるものなのかは不明だが、製品版ではこのあたりがちゃんと動くことを願う。

処理効率が問題となるような動作としては、たとえばMicrosoft Excel形式での表のセーブなどがある。こちらはかなり待たされる感じで、もう少しCPU能力が欲しくなる。今後のチューンナップに期待というところか。

GUI操作やセーブ以外の面では、あまり中間言語による処理というJavaの遅さを感じることは少ない。入力などもそれなりのスピードでできるし、計算もそんなに遅いという感じはしない。最終的なチューンナップが済めば、かなり実用的なものになるような感じだ。

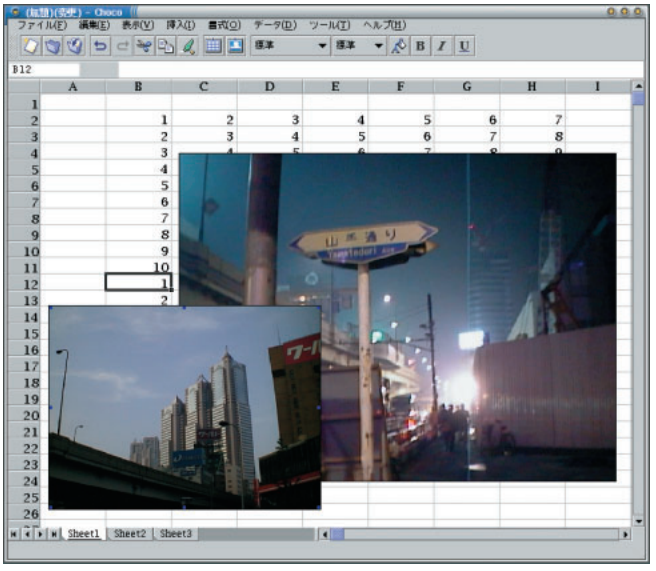
ワープロや表計算といったビジネス用アプリケーションは、結局のところ日本語サポートが最低限の条件となるが、国内メーカー開発による製品ということで、このあたりは安心できる。

ざっと使った印象では、レポートに付ける集計表やちょっとしたグラフを作るという程度であれば、なかなか使えるし、ちょっとしたデータの整理や、リストの作成用といった用途でも大丈夫だろう。しかし、毎日表計算を使って、複雑な計算を行うような仕事にはちょっときついかもしれない。

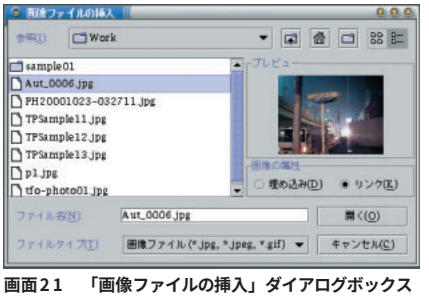
文書を打ち込むワープロソフトには、機能を削って「軽く」して、エディタ感覚で使うといった方向はありえるのだが、表計算は本来の使い方からすると高度な計算ツールであるため、あまり機能を軽くすると利用範囲を狭めてしまう可能性がある。ただ、日本のオフィス内での使われ方を見ると、単純な集計表やリスト、簡易データベース的なものも多いため、ある程度のユーザーの要求には応えられるだろう。しかし、シミュレーション的な計算を行うようなユーザー（最近では、技術系の計算も表計算で行う場合がある）の要求は満たせなくなってしまう。このあたり、ターゲットユーザーをはっきりさせないと、なかなか機能を絞り込めないかもしれない。

また、PlugInによる機能拡張に期待したいところだ。たとえば、データベースと連携できるような機能があるとかなり応用がききそうな感じがする。

このほか、一太郎Arkに埋め込みドキュメント的に作成した表などが入れられ、表を修正すると自動的に一太郎Ark文書も更新されるようになるといいだろう。すでにWindowsでOLEによる埋め込みや、リンクドキュメントに慣れてしまったユーザーにとっては、静的なデータの貼り付けだけの文書間連携には戻れないと思うからである。



画面20　画像の貼り付け
JPEGファイルなどを自由にワークシート上に配置することができる。また、画像オブジェクトの端の部分にあるハンドルを使って拡大縮小も行える。



画面21　「画像ファイルの挿入」ダイアログボックス
ワークシートに挿入する画像の選択画面。ここでは、実際に画像を貼り付ける前に内容を見ることができる。

隠喩としてのコンピュータ

諸関係の総体のデザインにむけて

文：豊福 剛
Text : Tsuyoshi Toyofuku
illustration : hmm

そもそも情報って何なのかを考え出すと、よくわからなくなる。

本やCDには、情報が詰まっている。Webにもたくさんの情報がある。たまに渋谷の街を歩いたりすると、くらくらしてしまうのも、情報があふれているからだ。見るもの、聞くもの、感じるもの、すべてが情報だ。でも、重要なのは、自分にとってそれらの情報に、価値があるかどうかだ。つまり、価値のある情報とそうでない情報がある。その判断は、きわめて主観的なものだ。情報は、いつでもどこでも価値があるわけじゃなくて、時と場合によって、情報の価値は変ってしまう。

そう考えると、情報そのものに価値があるわけではない。たぶん、情報がトリガーになって何らかの反応なりアクションが生起したときに、はじめてその情報の価値が認められるのだ。

情報の価値と関係の価値

ところで、ほとんどの情報は、誰かが何らかの目的で発信したものである以上、情報の価値とは、自分とその発信者との関係の価値である、と拡張解釈できるだろう。

だから、情報の洪水とか、情報があふれている、といった言い方は、誤った言語の用法なのだ。でも、それが誤った用法と見なされずに、納得的な表現として受け入れられてきたのは、おそらくその関係が一方的だったからだろう。20世紀になって、情報は爆発的に増大し、それは現在も進行中だ。情報はますます一方的に送り付けられてくる。でも、人間の情報処理能力には、限界がある。コンピュータみたいにクロック速度が倍々で向上するわけでもないし、どこかのサーバみたいに24時間365日ノンストップで稼動させることもできない。

それに何よりも、世界はあまりに複雑になってしまった。どんな人でも、自信をもって判断が下せる分野は、2つか3つに限られるだろう。それ以外の分野については、専門家の意見なり見解を聞いて判断するしかない。でも、その分野の専門家の意見が正しいかどうかなんて、ほんとはわからないのだ。そして、わからないものについては、基本的に無視してしまおう、そのような問題と自分とは何の関係もないと考えることにしよう。どうしても、決定を下さなければならない問題については、他人の判断に追随しよ

う。このようにして情報の爆発に対応してきたのだ。

20世紀は情報爆発の世紀だった。そして、おそらく21世紀は関係爆発の世紀なのだ。そして、インターネットの本質が、エンド・ツー・エンドのコミュニケーションにあるとしたら、それが革命的であるのは、あらゆる領域において関係を爆発させる起爆剤になるからだろう。

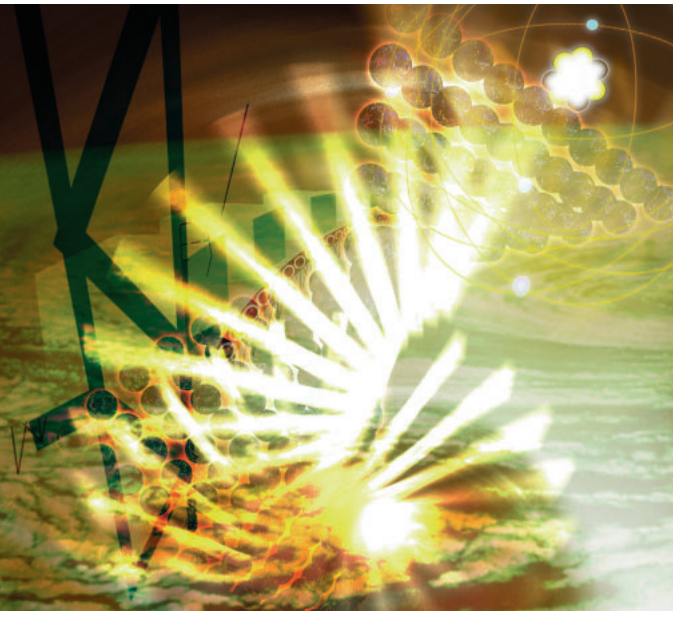
人間とは諸関係の総体であるとマルクスは考えた。インターネットが関係の爆発をもたらすものである以上、インターネットによって、いままでの関係の総体は必然的に揺さぶられ、破壊され、そして新しい関係の線が引かれることになる。インターネットは、地球規模での関係の爆発であるグローバリゼーションを駆動する。

「政治」を関係調節装置として実装するには

はたして人間は、この歴史の津波に対処することができるのだろうか。グローバリゼーションが進行する一方で、人間はシェルターとしての民族主義や宗教への回帰に逃げ場を求める現実がある。それは関係の爆発に対する人間の不能と恐怖がそうさせているのかもしれない。それは旧ソ連圏の国々の悲惨というだけではない。日本においても、そのような悲惨は、着々と進行しつつあるのではないか。

あらゆる関係という関係は、それが人と人の関係である以上、そこにはかならず利害がからんでいる。そして、両者が納得いくかたちで利害を調整するのに困難を極めることは少なくない。その調整に失敗したとき、最後は暴力が行使されてしまう。暴力は、最終的な権力のかたちだ。そして、暴力を抑止するのもまた権力だ。権力は関係において生じるのだから、権力とは、決定するのは誰なのか、という問題ともいえるだろう。

時代閉塞とは、関係の閉塞である。政治の閉塞もまた、関係の閉塞である。政治の無能は、政治と自分との関係が無能であることの反映である。政治は、テレビのニュースや新聞のうんざりする記事によって、一方的に送りつけられるだけだった。政治との関係は、選挙というチャンネルでしかなかった。政治家は、いつも景気のことばかりを語っていた。景気がいいときは、そのような政治でも、よしとされてきた。でも、景気は一向に良くならないままだ。だから、景気をなんとかするための政治を求める、というのではなく、政



治そのものとの関係をなんとかするしかないのだ。

ITは、彼らの経済のためだけにあるのではないし、彼らの政治のためだけにあるのでもない。それは、同時に、ぼくたちの政治のため、ぼくたちの経済のための道具でもあるのだ。それはすでに与えられた道具なのではなく、爆発する関係を調整する装置として実装し、使いこなさなければならない。

関係をデザインし直すには、まずコミュニケーションの条件をデザインし直すことが必要だ。思うに、一番の問題は、法律の文体とぼくたちの言語とのセマンティック・ギャップにあるのではないだろうか。法律の文体は、ぼくたちのためにあるのではなく、彼らのためにあるものの、典型のような気がする。それに、法律は読めば一意的に意味が解釈できるものではなくて、それがどのように解釈され、適用され、判断されてきたのかという歴史とも深く関係している。それが、法律の専門家と非専門家の間に、絶対的な知の格差をもたらし、彼らに権力を与える根拠になっているのであれば、そのギャップを埋めるための知識ベースをオープンソース運動として展開できないものだろうか。

投票的評判のためのアルゴリズム

関係調整装置を実現するためには、電子メール、Web、BBSといった既存の道具だけでは、おそらく十分でない。Webよりもっと双方向的に編集できるテキストベース、BBSよりもっと重層的な構造をもった会議システム、さらには、断片的な情報をベースにした検索・照会システムを超えて、関係の調整や意思決定のプロセスを支援するシステム。そうした方向への展開が求められる。以前紹介したLETSタイプの地域通貨の設計・実装プロジェクト、GETSを主宰されている鈴木 健氏は、貨幣、投票、所有の情報論的融合という、非常に刺激的な構想を提起されている。その概要は「ネットコミュニティ通貨の玉手箱」（http://sacral.c.u-tokyo.ac.jp/~ken/gets/tamatebako.html）にまとめてある。これを読みながら漠然と予感したのは、P2Pはたんにサーバ集中から分散処理へのシフトというだけでなく、鈴木氏が「評判言語」と名付けているサムシングへの指向が決定的に重要であるということだ。

それは、これまでの常識的な社会観を補強するための装

置というのではなく、むしろ、そうした常識に対する意識変革を同時に要求する装置であるように思う。たとえば、「相対値貨幣」のアイデアは、商品の購入—販売という常識的に理解されている関係を、投資—被投資の関係に変容させるものである。つまり、買うとは売る人に対する投資である、というラジカルな発想の転換が打ち出されているのだ。おそらく、ここで提起された装置は、より抽象化することで、より広範な領域に適用可能なツールボックスになる気がしている。

たぶん、これまでのインターネットのユーティリティにおいて足りなかったのは、ノードとノードの間のダイナミックな関係をシステムの中に内包することだったのではないだろうか。たとえば、Webにおいては、リンクの機能が重要な役割をはたしているわけだが、それは、一方向的であり、固定的であるという限界がある。そうしたHTMLベースのリンクのもつ限界を、検索エンジンはかなりの部分、補完している。最近とみに人気を集めている検索エンジンにGoogleには、ページの重要度（PageRankと呼ばれる）を判定するアルゴリズムが実装されているらしい。このPageRankの概要については、山本 篤「Googleの基礎技術PageRankについて」（http://aglaia.c.u-tokyo.ac.jp/~yamamoto/pagerank/pagerank.html）、馬場 肇「Google の秘密 - PageRank 徹底解説」（http://www.kusastro.kyoto-u.ac.jp/~baba/wais/pagerank.html）が参考になる。

これらのPageRankについての解説を読んで興味深かったのは、ページからページへのリンクを「投票」とみなしている点だ。また、ページ間の遷移確率を表わす行列から、固有値と固有ベクトルを計算することで、各ページの重要度を求めるアプローチは、「相対値貨幣」で評判行列から評判ベクトルを求めるアプローチと家族的類似関係にあるように思う。

システム工学や制御理論には疎いので、よくわからないのだけれども、このような遷移確率をベースにしたモデリングは、常套手段であり、とりたてて珍しいものではないのかもしれない。でも、こうした理論や方法を、実際に何かに応用できるためには、独特のセンスが要求されるのだろう。

とりあえずGNU Octaveをインストールして、線型代数の勉強しなおしだ。そうそう、GoogleはRed Hatのサーバ6000台で動いているらしい。どうやって管理しているんだろうなあ。

Profile

とよふく つよし

1962年東京生まれ。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。

バーチャル・シットインの論理

文：安田幸弘
Text : Yukihiro Yasuda

不安な時代のはけ口

日本の景気は一向に回復する気配すらなく、アメリカ経済の不調のおかげで、まだまだ景気は悪くなりそうな気配である。経済大国だなんて威張り散らしていた日本も、不景気に加えて財政は破綻状態、失業者は増える一方で、希望も何もあったもんじゃない。

こういう時代には国粋主義がはやる。外国人や外国の文化、外国の流儀を目の敵にして、日本の伝統、日本の歴史、日本の道徳、日本の美徳といった、わけのわからない価値に一抹の心の拠り所を求め、日本は偉いんだとブイブイ吹きまくり、不景気のウサを晴らそうという心理なんだろう。

そんな時代だからなのか、日本では妙な教科書が検定を通過してしまった。個人的には右翼教科書でも左翼教科書でも、歴史の教師が趣味で選べばいいと思うのだが、役人が検定制度なんてものを作るからややこしい。「日本政府が歴史歪曲教科書を合格させた」と、中国や韓国の愛国者たちをカンカラカンに怒らせてしまったのだ。

ネットの座り込み

中国・韓国の抗議の中には若干誤解もあるようだが、彼らが怒るのもまあ無理はない。日本政府は「歴史観は検定の対象ではない」と言っているが、旧植民地国民が怒っているのは、まさに植民地史観を日本政府が公教育の教材として認めたことなのだから。

だが日本政府は、「厳正な手続きを通過したので、再修正はできない」の一点張りで、それがまた火に油、とうとう怒り心頭に発した韓国ネットワーカーたちが、去る3月末に文部科学省をはじめとするサイトに「バーチャル・シットイン（仮想座り込み）」を仕掛け、やられた側はそれに対して「違法なサイバーテロだ」と逆ギレするという事件が起きた。

バーチャル・シットインは、1995年頃からアメリカで始まったインターネット上の抗議手法のひとつで、要するにブラウザの更新ボタンを押し続け、対象となるWebサイトの負荷を高めてサーバをストップさせてしまおうというもの。

スレッドを使うIISやApacheバージョン2はともかく、Apacheをはじめとする現在のWebサーバは、リクエストごとにプロセスをfolkする。ご存じのように、UNIXにとってfolkはかなり重い仕事で、同時に大量のプロセスの生成、消滅を繰り返せば、マシンの負荷は跳ね上がる。実メモリが少なければ、これに加えてスワップの嵐が発生し、下手をすればカーネルがパニックで落ちてしまうかもしれない。何とかしようと思ってログインしても、負荷は上がり切っていてコマンドを打っても反応が返ってくるまでに何分も待たなければならない。こうなると、ネットワークのケーブルを引っこ抜く以外にコントロールを取り戻す方法はない。

確かに抗議としてはかなり過激な方法だが、海外では何度かバーチャル・シットインが呼びかけられている。有名なところでは、'98年のメキシコのサパティスタによる金融機関への抗議、'99年のRTMarkによるeToys.comへの抗議などがあり、韓国では昨年、いわゆる通信秩序確立法に反対して政府機関のサイトを対象にバーチャル・シットインが行われているが、日本のサイトがバーチャル・シットインの対象になったのは、おそらく初めてのことだろう。

数年前のサーバと違って、最近のサーバは高性能になり、回線も太くなった。100人や200人が集まっても、そう簡単にサーバがダウンするもんじゃない。はたして抗議の対象にされた日本のサイトがいったいどんな対処をするのか大いに興味があって、当日はサイトの様子を見ていたのだが、対象にされたサイトはタカをくくっていたのだろうか、完全にストップしてしまったのである。

抗議された側はこれを「サイバーテロ」、「明らかな犯罪」として非難の声明を出し、日本の政府

は非公式に韓国に打診をしたというのだが、これは恥ずかしい。彼らの立場を認めるとしても、せいぜい「不当な抗議」ぐらいの声明にしておくべきだった。何しろバーチャル・シットインという抗議手法は、決してお上品な抗議であるとは言えないにしても、明確な犯罪行為ではない。事実、法的な責任を問おうとしても、少なくとも法治国家では根拠となる条文がなかったり、証拠不十分その他の理由で立件ができず、これまでに法的な責任が問われたケースはない。

合法的なサーバ攻撃

意外に感じる人もいるかもしれないが、バーチャル・シットインは、サーバへの侵入、ファイルの改変や破壊などがなく、通常の公開されたプロトコルしか使わないという点で、いわゆる不正アクセスの範疇に入らない。歴史教科書への抗議という意図は同じでも、中国からのサイト改ざん攻撃が明確な不正アクセスとされるのとは大きく異なる点である。

ただし、バーチャル・シットインと似たような効果をもたらすツールに、Trinooのような分散DOS攻撃ツールがある。もし今回のバーチャル・シットインでこれらのツールが使われていれば、これは完全な違法行為として問題になっていただろう。ツールを使った攻撃がバーチャル・シットインと異なるのは、これらのツールさえあれば1人でも攻撃が可能で、予告もなく、防衛はほとんど不可能だ。つまり攻撃のための攻撃でしかない。

それに対して、バーチャル・シットインは対策が取りやすい。今回のケースも、単に韓国からのパケットをフィルタすればいいだけで、実際、4月10日の「第二次バーチャル・シットイン」ではパケットを拒否して運用を続けたらしい。

だが技術的にはともかく、バーチャル・シットインは、数千人、数万人が手動で原始的な「更新」を繰り返すという点が、単なる攻撃ではなく、抗議の行動とみなされる部分だ。つまり公開の呼びかけに相当数の人々が同意しなければ成立しないという点が、一種の正当性の担保になっているのである。

日本のように、お上に楯突いて抗議するなんてのはもってのほかという社会では、抗議の権利、抵抗の権利なんてとんでもない話かもしれないが、海を渡れば抗議の群集が時の権力者を失脚させることさえある。韓国だって、今でも大規模なゼネストで交通が止まり、ソウルの中心部に火炎瓶が飛ぶのは日常茶飯事の社会である。さすがに火炎瓶はヤバいと思うし、抗議なら何でも許されるわけではないだろうが、権力による不当な行為への抗議は、およそ民主主義社会では広く民衆に認められた権利だ。今回の件でも、日本側がサイバーテロだ、違法行為だと興奮しているのに対し、韓国の当局はバーチャル・シットインは言論の自由に属するものだと判断しているという。

民衆の抗議の権利なんてものが存在しない中国からの抗議がサイト改ざんというクラッキングで、ノーベル平和賞受賞者を現職大統領に持つ韓国からの抗議がバーチャル・シットインという形で現れたのは、このあたりのセンスの違いを象徴しているのだろう。

しかし最近、情報で国家経済を支えようとする米国を中心に、サイトの運営の障害になる行為を国際的に違法化しようとする動きがあり、これにはバーチャル・シットインなども含まれかねないという。もちろんインターネットが無法地帯であっていいわけはないが、しかしやみくもに「サイバーテロ」の恐怖を煽り、政府や企業に都合が悪い行為をすべて違法化することでしか秩序を保てないのがIT社会だとすれば、そんな恐怖社会で暮らしたくなんてないもんだ。

Profile

やすだ ゆきひろ

生業はテクニカルライター。原稿書きのかたわら、(株)市民電子情報網のボランティア社長兼技術スタッフとして、NGO向けプロバイダのネットワーク運営に携わる。

ドクターShiodaの

ギョーカイSnapShot

PC業界今月の話題

文：塩田紳二
Text : Shinji Shioda

3・16　ソニー、エミュレーターでConnectixと和解
3・19　Intel、1GHz版モバイルPentiumⅢ発表
4・2　Office XP発表
4・5　Wind River Systems、BSDiを買収

世間では、ADSLやら光ファイバやらが話題だが、東京のすぐ隣の市に住んでいるとはいえ、東京とは反対の端にあたる筆者の家のあたりには、まだブロードバンドの波も来ていない。駅まで遠いとか、近所にコンビニが少ないなんてことはどーでもいいのだが、やはりネットワーク的に置いてきぼりを食うのはどうにも寂しい。やはり今、住居を選ぶなら、立地条件よりもネットワーク条件か。

IntelはPentium 4を急激に普及させるつもり

Intelは、当初Pentium 4の普及はゆっくりとしたもので、当面PentiumⅢが主流だし、0.13μプロセスを使ってPentiumⅢを強化するなんてことを去年言っていたのだが、どうも状況が変わってきたようである。おそらく、この号が出ているころには、1.7GHz版のPenitum4が発表されているはずだし、たぶん、今年末にはデスクトップ向けCPUの出荷の半分はPentium 4にするなんて話が届いているのだと思う。つまり、これから急激にPentium 4への切り替えを行おうというわけなのだ。

ここに来て、急に切り替えを行う背景には、Athlonへの対抗措置という意味もある。基本的にPentium 4は、高クロックを実現するために設計されたCPUで、同じクロックならば、PentiumⅢよりも遅くなるが、代わりにクロックを高くできるような設計になっている。もうPentium Ⅲでは、Athlonの対抗にはならないので、「選手交代」させるわけだ。また、米国の景気が悪くなったということもある。これは、どっかの国みたいに「デフレ」傾向で安くなったのではなく、景気が悪くなったので、予想出荷台数が減って、それならば、現在の製造能力でカバーできるという判断になったためだ。もし、米国の景気が良ければ、Pentium 4は常に不足気味という状態になったのだろうが、景気が悪くなって、かえってよかったわけである。

世間の予想として、Pentium 4はいまのPentiumⅢ並みの値段まで下がってくる。しかも、Pentium 4のリテールパッケージには、メモリまで付いてくるから、お買い得感は強い。また、メーカーのPentium 4搭載マシンも出そろい始めた。今のところ、メーカーセットよりも、マザーボードとCPUを買ってきて組み立てたほうが安そうな感じだが、これもいまに逆転することになるだろう。

Intelの急ぐ気持ちはわからないでもないが、このクロック競争には、ユーザー不在という感じがないわけでもない。日本にしても米国にしても、景気が悪いわけで、ユーザーの気持ちとしてはスピードより値段という感じである。それに現在のPentiumⅢを使っているユーザーなら、速度的な不満もそれほど感じていないのではないかと思う。Windowsにしても、CPUのクロックを上げるより、メモリを増やしたほうが、調子が良くなる場合が多いし、Windowsに比べて効率がいいLinuxならば、PentiumⅡでも不満を感じないぐらいである。特にシェルを使っている限り、ディスクアクセスなどで待たされることはあっても、処理が重いという感じにはならず、クロックが高くなればなるほど、CPUが休んでいる時間が多くなって、なんだか高いCPUほど、無駄な気がしないでもない。

MicrosoftはXで忙しい

Microsoftは、Windows XPに続いてOffice XPを正式に発表した。もっ

とも、出荷は、2001年の第2四半期つまり、4〜6月のどこかである。まあ、早くても5月の終わり、たぶん6月の出荷ということになると思われる。このOffice XPからはProduct Activationという機能が入って、インストール後にMicrosoftからライセンス認証を受けなくてはいけなくなるらしい。これが、ハードウェアの構成により生成される情報を含んでいる関係で、たとえばマザーボードを交換したりすると認証をやり直す必要があるのだとか。筆者のようにハードウェア構成を頻繁に変更するユーザーにとっては結構大変そうな気がします。いちおう24時間体制で受けつけるらしいですけど。

Office XPのXPとはExperienceの略だと言われているが、Officeはそれまで Office 10と呼ばれていた。MacintoshのOSが10をローマ数字で表す“X”を使っているのを考えると、なんだかOffice 10 → Office X → Office XPという感じで奇妙だが、Microsoftって“X”が好きなんですよね。そういえば昔ありましたよね、MSX。これはたしか“Microsoft X”の略だったと記憶しておりますが……。

4月に行われた東京ゲームショウ2001では、ビルゲイツが来日して基調講演で“Xbox”の話をしていた。Xboxってスペックだけを見ると、筆者が今原稿書きに使っているマシンよりも高機能なんだけど、これ、遊ばなくなったらパソコンとして再活用できないかしらね。この不況の影響か、東京ゲームショウも来場者が減ったとか、ゲームがどれも月並みなんて話が出ていて、去年とは状況が大きく違ってきているよう。そんな中で、これから始めるXboxってどうなってしまうのでしょうかね。米国のバブル景気時代の計画なんで、ビジネス上は大きく方向修正する必要があると思うんですけど。

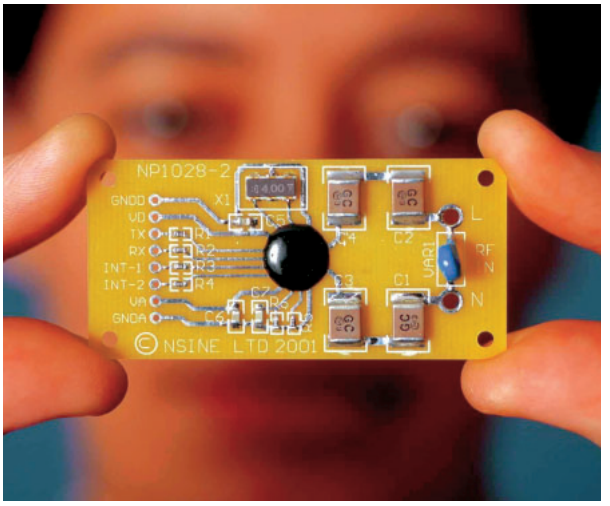
電灯線ネットワークの時代は来る？

普通の家なら必ず各部屋にある電気のコンセントだが、これを使ったネットワークという可能性が出てきた。米国などではすでに製品が出ているようだが、日本では、法律で高い周波数の信号を流すことができないため、いままであまり注目されていなかった。しかし、来年には電波法が改正される予定で、そうなると、電源コンセントを使って高速なネットワークの構築が可能になる。すでに家庭内の電話線を使ったネットワーク機器はあるが、やはり電話のモジュラジャックよりも、確実に部屋に付いているコンセントのほうがネットワークは組みやすいかも。また、電力会社経由での高速インターネット接続なんて話もあり、そうなると、パソコンを買ってきて、コンセントにつなぐだけで、もうインターネットにアクセスできるなんて時代がやってくるのかも。

この電灯線を使ったネットワークを手がける企業を、ソフトバンクが英国のメーカー、nSineと協同で設立するらしい。電波を使うスピードネットや、光ファイバを使うアイ・ビー・レボリューションなどという会社を持つソフトバンクとしては、電灯線のネットワークも押さえておきたいらしい。

余談だが、ソフトバンクの孫社長が、衆議院で「基本的人権にネットアクセス権を加えるべき」と主張したとか。でも、強制的にメールアドレス持たされたりするといやだな。市役所いくと、書類にメールアドレス書かされたりして……。

ただ、電灯線のネットワークには、解決すべき問題もいくつかあるらしい。たとえば、冷蔵庫などが出すノイズの問題とか、電気的に接続しているマンションの部屋同士なんて問題である。そういえば、インターネットマンションで、Windowsユーザーが「マイネットワーク」を開いたら、マンション中のパソコンがみんな見えたなんて事件もあったけど、電灯線ネットワークもそうなる可能性はあるわけだ。ハードディスクの容量が不足したら、他人のマシンのハードディスク使ったりしてね。マンション内でチャットとか、それはそれで楽しそうな気もするが、世の中には、マンション管理上の意見の食い違いから住人同士の仲がよくないマンションもあるしね。やっぱりファイアウォールとか必要なのかも。



電灯線をネットワークに使うためのアダプタ、“nPlug”チップ。コストはわずか5USドル程度。
nSine（http://www.nsine.com/）
ソフトバンク ネットワークス（http://www.softbank-net.com/）

正しきタコへの道

初級Linuxer養成講座

第21回　シェルスクリプト入門〜シェル変数

シェルスクリプトの基本は、ふだんコマンドラインに入力するコマンド文字列を、そのままファイルに書き込んだものである。しかし、本格的なプログラミング言語のようなプログラムを作ることもできる。今回は、プログラミングに必須の要素のひとつ「変数」について紹介しよう。

文：竹田善太郎
Text : Zentaro Takeda

先月に書いたように、自宅でもいよいよADSLが使えるようになる。今、この原稿を書いているまさにその隣で、回線の増設の工事が行われている。サービスの申し込みをしてから約3週間、開通までの期間は、おおむね**プロバイダの公約どおり**だっだ。この原稿を書き上げて、編集部にメールで送信する頃には、我が家も「ブロードバンド接続」の仲間入りをしているはずだ。

今回のADSL加入では、いままで使っていたISDN回線はそのまま残したかったので、新たにADSL専用のアナログ回線を増設した。電話との共用はしないので、いわば**ADSL専用線**を引いたことになる。昔は個人で専用線を引くなどというのは、とてつもなくお金がかかり、一種の**ステータス**になっていたことを思い出す。NTTが未使用のメタル線、いわゆる**ドライカッパー**をADSL業者に開放したことの恩恵だと思うが、月々数百円の回線使用料と、ADSLの接続料数千円だけで済むようになったのは、やはりありがたい。実際のパフォーマンスに付いて、額面通りの期待をしていないのは前回書いた通りだが、新しい技術が自分でも使えるようになるのは、やはりうれしいものだ。

ところで、マンションの配電盤内の配線が済んで、次は回線のテストでもするのかと思って見ていると、工事担当の技術者はおもむろに携帯電話を取り出し、どこかに電話している。どうやら、NTTの交換局で作業している同僚（？）に電話しているようだ。「これからショートテストをします」と伝えているのだが、こちら側にはそれらしい測定機器はいっさいない。被覆を剥いた電線を、ニッパーで挟んだり離したりしているだけである。

電話の会話をそれとなく聞いていると、どうやら交換局にいる技術者が、ショートされた回線の抵抗値を計っているらしい。「××を取り付けたから、少しは抵抗が高くなっていると思います……」なんてことを話している。二言三言やりとりがあったのちに、テストはOKということになったようで、モジュラコンセントを取り付けて工事は終了した。

宅内工事をしてくれた技術者は、こちらから聞くともなく「いやぁ、回線テスターのようなものがあればいいんですけど、**今のところはこんなテストしかできないんですよね**」と話してくれた。ADSL回線のテストをする専用の機器が世の中になかったり、あったとしても現場にまで行き渡っていないということらしい。

自宅にISDN回線を初めて引いたときのことを思い出すと、ラップトップ型の大がかりな回線テスターを使って、回線の状態をチェックしていたのだが、ADSLの場合はそこまで手をかけられないのだろう。ADSLでは回線の品質は保証されておらず、保証の手間をかけないぶん、価格も安くできるのだということを目の当たりに体験できたような気がする。

スクリプトとプログラム

前回説明したように、ふだんコマンドラインから入力しているコマンド文字列を、そのままテキストファイルに書き込めばシェルスクリプトができあがる。スクリプトの中で、リダイレクトやパイプを使って、コマンドの結果を保存したり別のコマンドに渡したりすることもできる。このように、簡単な**バッチ処理**をするのにシェルスクリプトを利用することも少なくないのだが、シェルスクリプトの**本当の実**

力は、そのプログラミング機能にある。

前回は、コマンドラインを実行させたい順にテキストファイルに書き込んだだけの簡単なスクリプトを作成してみたが、もう少しプログラムらしいプログラムを書けるようになると、日常の作業がずっと楽になることがある。特殊な例かもしれないが、ばらばらな名前の付いたたくさんのファイルに、アルファベット順に数字の番号を付けて整理したいような場合があったとする。これを、ひとつひとつ、

```
$ mv hogehoge.mp3 001_hogehoge.mp3
$ mv hugehuge.mp3 002_hugehuge.mp3
        :
```

などと手作業でやるのは、コマンドラインであってもGUIのツールであっても面倒なことこの上ない。このような作業こそ、スクリプトを使って片付けるべき作業なのだが、単に、コマンドラインを書き並べただけのスクリプトでは、手間がかかることは手作業でやるのはまったく変わらなく、やはり意味がない。

このような場合には、たとえば、プログラミング上の標準的な機能である繰り返し処理を使えば、数行のプログラムを書くだけで済んでしまう。使い方を覚えるのが面倒と感じるかもしれないが、いったん覚えてしまえばあとでずっと楽ができるのがプログラムの利点なのだ。

「プログラミング」に必要な要素

世の中にはいろいろな種類のプログラミング言語があるが、これらのプログラミング言語すべてに共通する機能がいくつかある。なかでも重要なのは変数と繰り返し処理である。

書店などでプログラミングの教科書を覗いてみると、ずいぶんと難しそうな項目が並んでいて、たとえパソコンに抵抗がなくても通常の神経を持った人なら、まず9割は拒絶反応を示すだろう。あれだけの内容をすべて理解しないとプログラムを書くことすらできないのかと考えると、気が遠くなってしまうかもしれない。

しかし、我々のような素人にとっては、オブジェクト××とか●●APIなどというものはあまり関係ない。先に述べたような変数と繰り返し処理の使い方だけ知っていれば、十分なのである。

これはシェルスクリプトに限った話ではない。C言語でもPerlでも、変数と繰り返し処理（あえていえば、このほかに「入出力」くらい）の使い方だけマスターすれば、あとは必要なときに必要な機能を調べて追加したり、ほかの人のプログラムを見てまねをするだけで、十分に実用的なプログラムを書くことができるのだ。

「シェル変数」とは

では、まずbashにおける「変数」について見てみよう。試しに、次のようなコマンドラインを実行してみてほしい。

```
$ a=hoge
$ echo $a
hoge
```

この一連のコマンドラインで繰り返し使っているaあるいは$aの部分が、bashの変数あるいはシェル変数と呼ばれるものである。プログラミングにおける変数とは、データを一時的に保存する場所のことだと思っていてほしい。変数には名前が付けられていて、その名前を指定することで変数にデータを保存したり、保存されているデータを呼び足したりすることができる。

上の例では、「a」という名前をもつ変数の中に「hoge」というデータを保存したり呼び出したりしている。このようすを図に表してみると、図1のようになる。頻繁に使われているたとえだが、変数は箱のようなものと思っておけばわかりやすい。いまの例でいえば、「a」という名札の付いた箱の中に、いろいろなデータを保存したり取り出したりすることができるのだ。

シェル変数の使い方

ところで、先の例で変数にデータを入れるときは「a」という書き方だったのに、変数の中身を呼び出すときには「$a」というように、先頭に$文字が付いているのを不審に思った読者もいるかもしれない。筆者自身も、この違いに最初はとまどった。普通のプログラミング言語なら、変数にデータを入れるときも、データを読み出すときも、同じように記述するのが常識だからだ。

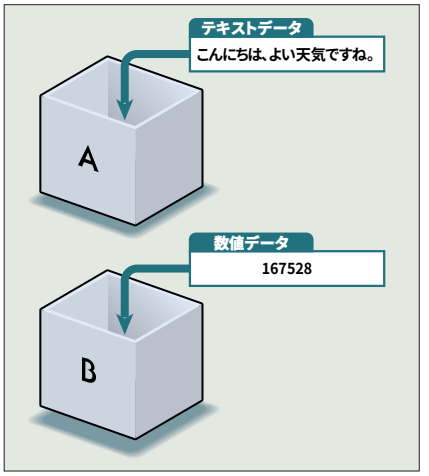


図1　「シェル変数」の概念
「変数」とは、データを一時的に保存しておく「箱」のようなものと考えることができる。

しかし、細かい理由はあえて問わないことにして、**シェル変数にデータを入れるときは変数名だけ、データを読み出すときには$を付ける**ということを、丸暗記しておいたほうがよいだろう。あるいは**$記号は変数の中身を読み出せという命令**というように理解してもよいかもしれない。

もうひとつ、初心者がつまずきやすいのが、**空白文字**(スペース文字)の入れ方だ。変数にデータを入れるときに、

```
$ a = hoge
```

というように、イコール記号の前後にスペースを入れてしまうと、

```
bash: a: command not found
```

というエラーメッセージが表示されてしまう。「『a』などというコマンドは知らない」という意味である。なぜこのようになってしまうかを説明すると長くなるのだが、後学のために簡単に述べておこう。

bashがコマンドラインを読み込んで実行する場合、スペース文字で区切られたひとかたまりの文字列を「単語」としてとらえる。そして、先頭にある「単語」をコマンド名であると判断するのだ。このため、「a = hoge」のようにスペースで区切ってしまうと、先頭の単語「a」をコマンド名として指示されたと解釈して、結果として「そのようなコマンドは見つからない」という結果になるのだ。「a=hoge」というようにひと続きの「単語」の中に「=」文字が含まれている場合、bashはこれを「変数にデータを入れたいのだな」と判断して、適切な処理をする。

●bashの不思議

ところで、少し話は脱線するのだが、以上のようにコマンドラインでシェル変数に値を設定する場合、いろいろと不思議なことがある。

bashでは、

```
$ a=foo b=bar c=buzz
```

というように、1行で複数のシェル変数に値を設定できるようになっている(**画面1**)。この例では、変数a、b、cのすべての値が指定したとおりに設定される。

では同じように、変数を設定する命令に続けて、通常のコマンドを実行させようとしてみると、一見するとうまく動く。

```
$ a=foo ls
```

上の場合なら、なんのエラーも表示されずに、lsコマンドの結果が表示される。ところがこの場合、「a=foo」という部分は**完全に無視**されてしまって、シェル変数aの値は以前のまま変わっていないのである。もし、変数の値を設定してから、別のコマンドを続けて実行させたいという場合には、スペースではなく「;」などの区切り文字を使って、コマンドを区切らなければならない。

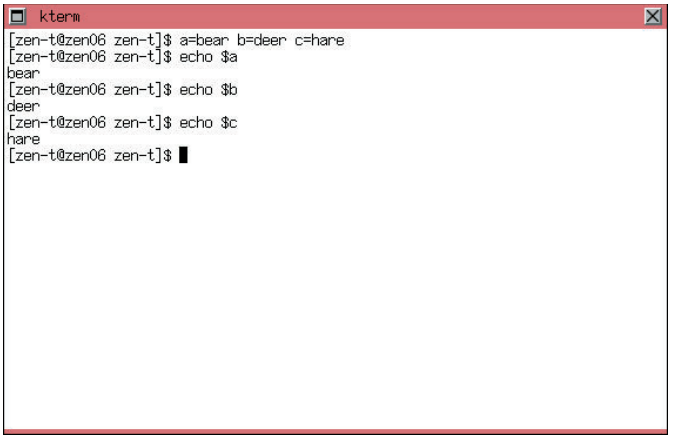
```
$ a=foo; ls
```

上のようにすれば、シェル変数aの値は変更され、lsコマンドも実行される。

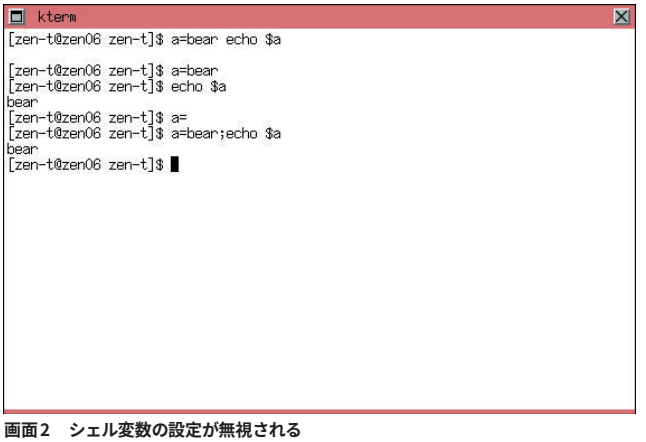
シェルスクリプトの中でも、これと同じようなことが起こる。改行を入れるのが面倒だからと、次のような内容のスクリプトを書いたとしよう。

```
a=foo echo $a
```

このスクリプトは、予期したように



画面1　複数のシェル変数を一度に設定
このような記述方法はあまりお勧めできないが、このようなこともできる。



画面2　シェル変数の設定が無視される
シェル変数を設定してからコマンドを実行したつもりでも、実際には設定されていない。1行ずつ実行するか、「;」などで区切って実行すべきだ。

は動かない。変数aの値は設定されないので、実行しても画面にはなにも表示されないことになる（**画面2**）。

結局何がいいたいかといえば、スクリプトを書くときには**横着はしないで**、

```
a=foo
b=bar
echo $a
```

というように、1行に1つずつ変数の設定やコマンドを書いたほうが、**トラブルに遭わずにすむ**ということである。

●シェル変数は「その場限り」

シェル変数の内容は、実行中のシェルを終了するとすべて**無の状態**になる。たとえば、X Window Systemの環境でターミナルウィンドウを閉じたり、ログアウトしたりすると、それまでに設定したシェル変数はすべてクリアされる。改めてウィンドウを開いたり、マシンにログインしても、以前に設定したシェル変数の内容は残っていない。いわばその場限りのデータ格納場所なのである。

シェル変数の「名前」

いままでは、シェル変数の例として「a」や「b」といったごく簡単な名前を付けていたが、べつにアルファベット1文字でしか変数の名前を付けられないというわけではない。シェル変数では、アルファベットの大文字、小文字、数字などを組み合わせて、比較的自由に変数の名前を付けることができるのだ。たとえば、

```
ThisIsMyVariableNo1
```

というような、説明的な長い名前を付けることもできる。英数字のほかに、「_」（アンダースコアを使うこともできる。ただし、そのほかの記号類（$%&－/<>．＊など）は基本的に使えない。

bashのシェル変数の名前では、英語の**大文字小文字は区別される**。つまり、

```
ThisIsMyVariableNo1
```

と、

```
thisIsMyVariableNo1
```

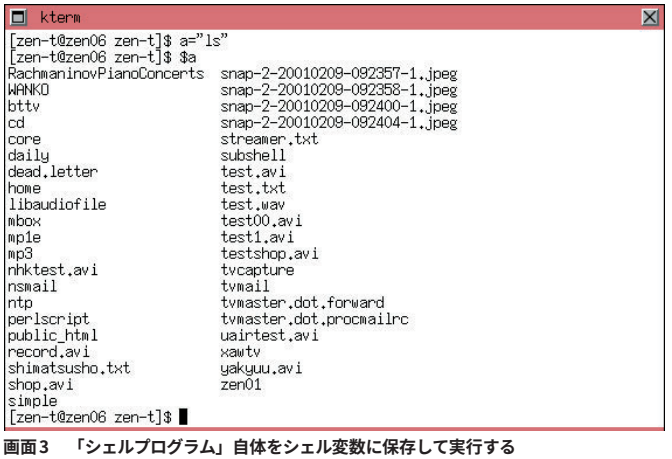
は、**別の変数**であるとみなされる。

シェル変数に入れられるデータの種類

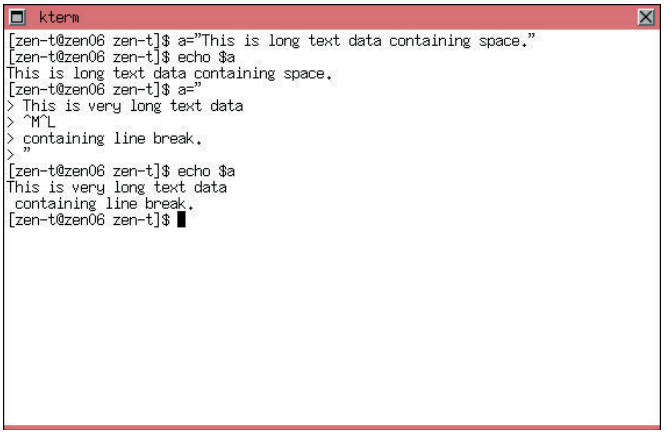
シェル変数には、いろいろな種類のデータを入れることができる。数値、文字列、ファイル名、ディレクトリ名、はては、シェルプログラムそのものを変数にしまうことすらできる（**画面3**）。ところが見方を変えると、シェル変数には**文字列だけ**を保存できるという言い方もできる。じつは、シェルにおいては、数字であろうが、ファイル名であろうが、プログラムであろうが、すべては**単なる文字列**として扱われているのだ。たとえば、

```
a=100
```

というように、変数aに「100」という値を保存した場合、この「100」は、**数値の百**を意味すると考えることができるが、「1」、「0」、「0」の3つの文字が並んだ、**ただの文字列**としてとらえることもできるのだ。これらのうちのどちらの意味になるかは、変数のデータを利用するコマンドによって



画面3　「シェルプログラム」自体をシェル変数に保存して実行する
このような使い方は、比較的一般的なシェルプログラミングのテクニックだが、システムへの不正侵入の手段としても使われることがあるので要注意だ。



画面4　長い文字列を変数に保存する
スペース文字を含んでいたり、複数の行にまたがるような文字列は、「"」などの引用符でくくることで変数に保存できるようになる。

変わってくる。

たとえば、

```
a=100
echo $a
```

のように使っている場合は、echoコマンドは変数aの中身をただの文字列としてしか見ていない。ところが、

```
a=100
head -$a file1.txt
```

のように、headコマンドのオプション（表示する行数を数値で指定する）として変数aを使った場合は、数値の100を意味するようになる。

しかし、これとてもっと詳しく見てみれば、単にheadコマンドが、オプションの「100」という文字の並びを読み込んで、内部でそれを数値に変換しているだけなので、やはり**シェル変数の値は「文字列」でしかない**と考えることもできる。

ちょっと混乱させてしまったかもしれないが、要するにシェル変数の中には**何でも入れることができる**と覚えておいてよいだろう。

●長い文字列を変数に入れる

ところで、シェル変数に「シェルプログラム」も入れることができると書いたが、プログラムのような**長大な文字列**を変数に保存するときは、どのようにすればよいのだろうか。プログラムに限らず、たとえばスペース文字を含むような文章やデータを変数に保存するときは、コマンドラインでもシェルスクリプトでも、少々やっかいである。

```
a=This is text data.
```

上のように文字列を変数に保存しようとしても、エラーになってしまう（**画面4**）。このような場合には、”などの**引用符**を使って、保存したい文字列全体を囲んでやるとよい。

```
a="This is text data."
```

これで、目的通り「This is text data.」という文字列データを変数aに保存できるようになる。シェルスクリプトの中では、引用符を使えば、複数の行にまたがるような長い文章やプログラムなどを変数に保存することもできる。

```
a="
This is multi line text data.
You can put anything you like in
your variables.
"
```

実は、bashでは「”」以外にも「’」、「‘ 」などの引用符が使えるのだが、これらの使い分けに付いてはいずれ説明することにしたい。

とにかく、シェル変数に「文章」のような長いデータを保存したい場合には、**「”」で囲めばよい**ということを覚えておけばよいだろう。

シェル変数と「環境変数」

ところで、WindowsやMS-DOSなどの知識がある程度ある人なら「環境変

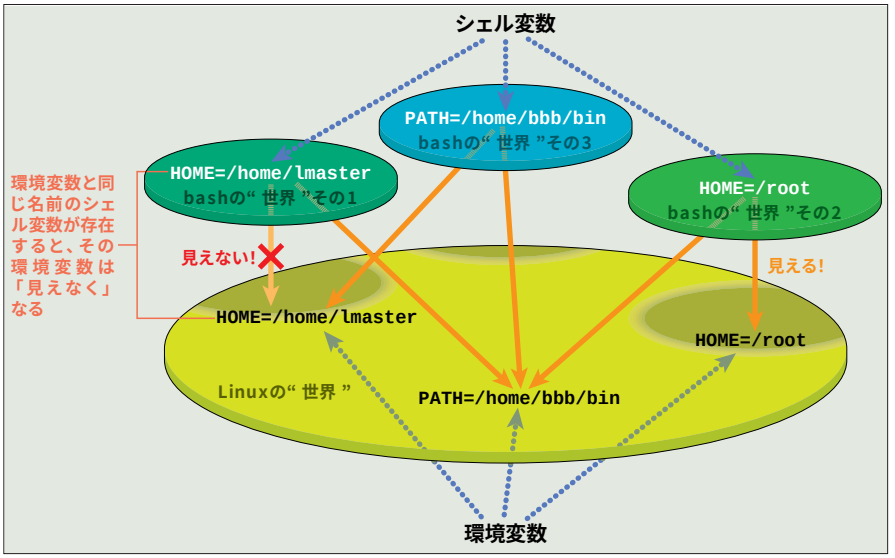


図2　環境変数とシェル変数
環境変数は、どのプログラムからでも共通に利用できる「共用記憶領域」と考えることもできる。一方のシェル変数は、1つのシェルの内部（あるいはシェルスクリプトの内部）でのみ利用できる記憶領域だ。

画面5　シェル変数の一覧を表示させる
setコマンドを使えば、その時点で使われているシェル変数の一覧を見ることができる。

数」という言葉を聞いたことがあるだろう。プログラムの基本的な設定などを行うために、

```
SET MYCOMPORT=1
```

というような形で、数値や文字列などを保存するものだ。これと同じく、UNIXやLinuxにも「環境変数」というものがある。というより、DOSやWindowsの環境変数のほうが、UNIXの環境変数を**真似**したのだ。

Linuxでは、いままで説明した「シェル変数」とは別に、この「環境変数」も使うことができる。環境変数は、bashなどのシェルの内部に限らず、あらゆるプログラムからアクセスできるデータ保存領域なのだが（**図2**）、bashのコマンドラインやシェルスクリプトの中から見る限り、シェル変数と環境変数を見分けることは難しい。どちらも、**$変数名**の形式で内容を読み出すことができるからだ。

実は、bashでも、シェル変数と環境変数は別の記憶領域として存在しているのだが、これについては改めて説明することにしよう。とりあえずは、「シェル変数」とは別に「環境変数」というものが存在するということだけ覚えておけば問題ない。

特別なシェル変数

その時点で使われているシェル変数の一覧を表示する方法がある。setというコマンドを使うのだ。

```
$ set
```

すると、**画面5**のように、さまざまなシェル変数の変数名と、その変数に保存されているデータの一覧が表示される。

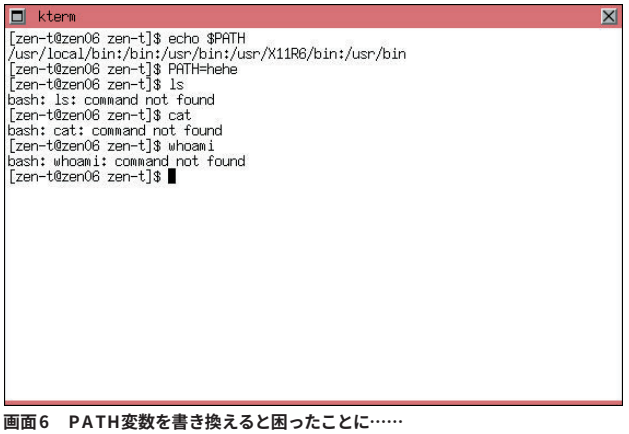
Linuxにログインしたばかりの状態でも、かなりの数の変数があらかじめ設定されていることにびっくりするかもしれない。実際に、どのような変数が使われているかに付いては、ディストリビューションやユーザーの環境によって大きく変わってくるが、たとえば、**HOME**や**PATH**といったような変数については、ほとんどの場合、設定済みになっているはずだ。

これらの変数の多くは、Linuxのシステムにおいて**特別な意味**を持っている。たとえば、PATH変数は、システムの中でコマンドファイルが納められているディレクトリの一覧を指定するもので、この変数にディレクトリが含まれていれば、コマンドラインでいちいちディレクトリのパス名を指定しなくても、コマンドを起動できるようになっているのだ。逆に、PATH変数を勝手に書き換えてしまうと、それ以降、**ほとんどのコマンドが使えない状態**になってしまう（**画面6**）。

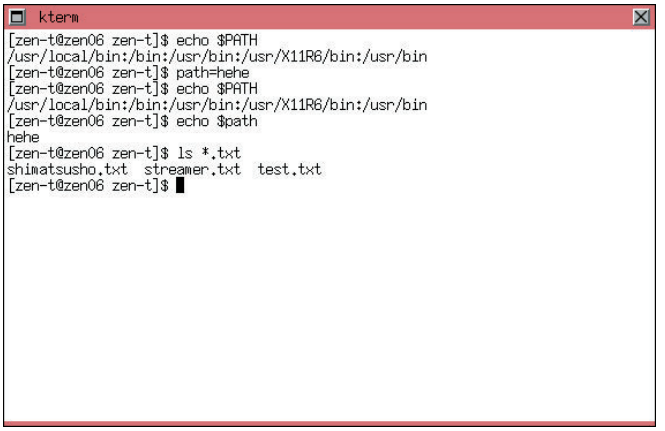
このように、シェル変数（そして環境変数）の中には、特別な役割を持つものが多い。このため、コマンドラインやシェルスクリプトの中では、これらの特別なシェル変数を書き換えてしまわないように注意しなければならない。

幸い、このような特別な変数では、**すべて大文字**の変数名がつけられていることが多い。従って、コマンドラインやシェルスクリプト中で、ユーザーが変数を使うときには、**すべて小文字**あるいは**大文字と小文字を混在**させた名前を付けるようにすればよい。こうすれば、たとえば**path**という名前の変数を使ったとしても、PATH変数とpath変数はまったく別のものと見なされるので、コマンドラインに不都合が生じるようなことはないはずだ（**画面7**）。

次回は、いままで説明した「シェル変数」のもっとも便利な利用法である**繰り返し処理**に触れることにしよう。



画面6　PATH変数を書き換えると困ったことに……
PATH変数（PATH環境変数）を書き換えてしまうと、コマンドが実行できなくなって困ったことになる。PATH以外でも不用意に書き換えるとやっかいなことになる変数は少なくない。



画面7　path変数とPATH変数は「共存」できる
PATH変数の代わりにpathという名前の変数を作った場合なら、なんの問題も起こらない（bashの場合）。

InterBase 6.0

IBPerlはPerlからInterBaseへ簡単に接続できるユーティリティです。このIBPerlを使用することで、CGIを利用したWebベースのプログラムを作成することができます。

第7回　IBPerlの概要について

文：加藤大受
Text:Daiju Kato　kato@jcom.home.ne.jp

前回、InterBaseのセキュリティについて解説し、独自のシステムを作成するときはトリガー、ストアドプロシージャを効果的に利用した、独自のセキュリティシステムを作成したほうが将来的に拡張しやすい細かなユーザーの権限管理を実現できることを解説しました。今回は、Linuxで最も使用されているスクリプト言語であるPerlからInterBaseのデータベースを操作する方法について解説します。PerlからInterBaseを利用するには、ユーティリティであるIBPerlを利用します。Perlについては詳細に解説しませんので、Perlの使い方についてはPerlに関する市販の書籍などを利用してください。

IBPerlは、PerlからInterBaseのデータベース操作を簡単に実現できるPerlモジュールです。このプログラムの作者であるBill Karwin氏は、当時のインプライズコーポレーションの100％子会社であるInterBaseソフトウェアコーポレーション（現在はボーランドソフトウェアの一部）にて、ドキュメントマネージャを行っていた人で、InterBaseの開発チームにてドキュメント作成のマネージメントを行うかたわら、個人的にIBPerlの開発を行っていました。InterBaseのオープンソース化の発表とともに、ボーランドソフトウェアを離れ、現在はデータベース関連のコンサルテーションを行っているそうです。また、現在でも定期的にIBPerlは更新されており、InterBaseの進化とともにIBPerlも少しずつ進化し続けています。

また、IBPerlは欧米のInterBaseを使用するエンジニアには非常に有名なツールで、日本でも数多くのユーザーが使用しています。欧米のメーリングリストではIBPerlだけのメーリングリストが存在しています。これはGPLのライセンスで公開されていることも理由のひとつですが、やはりInterBaseの開発メンバーが開発しており、メーリングリストで直接彼とやりとりができる部分にも起因しているでしょう。

それでは、IBPerlの特徴をまとめてみましょう。IBPerlの特徴は次の4点に集約できます。

- **Perl5に対応し、Perlベースでオブジェクト指向のパワフルなプログラムが作成可能**
- **IBPerlにより、CGIから簡単にInterBaseのデータを操作できる**
- **IBPerlにより、非常に軽快に動作するデータのインポート・エキスポートプログラムの開発が可能**
- **スクリプト言語であるPerlにより、プラットフォームに依存しないプログラムの開発が可能**

筆者は、個人的にPerlの良さは非常に簡単でありながら、パワフルなプログラミングが可能なスクリプト言語であることだと思っています。

この使いやすいPerlから、簡単にInterBaseのデータ操作を実現できるIBPerlは、ちょっとしたWebシステムの開発時に力を発揮すると思っています。

IBPerlの使い方

IBPerlはすべてPerlで書かれているPerl用のモジュールです。最新版は、Bill Karwin氏のWebサイトからダウンロードすることができます。また、同サイトにてIBPerlのマニュアルを参照することができます。

実際にIBPerlをWebサイトからダウンロードし、インストールしてみましょう。Webサイト（**表4**）にてWindows用のzip形式とUNIX用のtar.gz形式が入手可能です。UNIX用のtar.gz形式をダウンロードし、**画面1**のようにIBPerlのインストールを行います。インストール手順は非常に簡単ですが、環境変数INTERBASEを設定しないとInterBaseがどこにインストールされているかがわからないため、エラーが発生します。環境変数をセットすることを忘れないようにしてください。

インストールが終了したら、サンプルファイルを実行して正しくインストールされているかどうかを確認してください。**画面1**では、従業員データベース内のPROJECTテーブルを参照するサンプルを使用して、IBPerlが正しくインストールされているかどうかを確認しています。

サンプルディレクトリにはCGIを使ったWebページのヒット数を管理するサンプルなどが入っていますので、一度どのようなサンプルが入っているかを確認してみるといいでしょう。

インストールがうまくいかない場合は、IBPerlのドキュメントを参照してみてください。

インストールが正しく行えたことを確認したところで、簡単にIBPerlの使い方について説明してみましょう。IBPerlはPerlからInterBaseを操作するためのユーティリティですので、次のような動作を行います。

- **InterBaseのデータベースへの接続**
- **トランザクションの管理**
- **SQL文の実行**
- **データの読み取り**

```
# cd /tmp
# tar zxvf IBPerl-08p3.tar.gz          ダウンロードしたIBPerlを解凍する
# cd IBPerl-0.8p3/
# perl Install                         インストールスクリプトを起動してmakeファイルを作成
# export INTERBASE=/opt/interbase
# perl Makefile.PL                     メイクファイルの実行
# make install                         作成したモジュールのインストール
# cd examples
# cp /opt/interbase/examples/employee.gdb .
# perl select.pl                       サンプルプログラムの実行

IBPerl version 0.8p3
1..8
ok
ok
ok
ok
ok
PROJ_DESC:      Design a video data base management system for
controlling on-demand video distribution.

PRODUCT:        software
PROJ_NAME:      Video Database
TEAM_LEADER:    45
PROJ_ID:        VBASE

  …… 省略 ……

ok
Committing... ok
Disconnecting... ok
```

画面1　IBPerlのインストールの流れ

・BLOBデータの読み取り
・データベースからの切断

SQL文の実行では行セットを返すSELECT文を実行した場合にはフェッチを行い、各行のデータの読み取りを行います。このように、IBPerlは基本的なデータベース操作を行うための機能をすべて提供しています。

それでは、データベースへの接続から説明していきましょう。データベースの接続はConnectionオブジェクトのインスタンスの作成で行います。構文およびパラメータは**表1**のようになります。データベースハンドルが0以下の場合はエラーになりますので、エラーハンドリングが可能です。

接続できたら、続いてトランザクションを開始します。一般的に、SQL文によるデータ操作を行うにはトランザクションを開始しなければなりません。ほとんどのリレーショナルデータベースでは、明示的にトランザクションの開始を宣言しない暗黙のトランザクションをサポートしていますが、IBPerlでは明示的にトランザクションの開始を宣言しないとSQL文によるデータ操作が正しくできません。

トランザクションは、Transactionオブジェクトのインスタンスを作成することで開始することができます（**表2**）。トランザクションの終了は、commit()メソッド、またはrollback()メソッドにて行います。つまり、トランザクションをコミットするか、あるいはロールバックするかによって判断することになります。

現在のIBPerl Version 0.8では、トランザクションモードはInterBaseのデフォルトである、READ WRITE WAIT SNAPSHOTモードにしか対応していません。ただし、IBPerlのソースであるIBPerl.pmを参照してみると、Mode、Resolution、Isolationというパラメータが用意されています。現在のバージョンにて細かいトランザクション管理を行いたい場合は、IBPerl.pmを修正してみるといいでしょう。

トランザクションを開始したら、SQL文操作を行うことになります。SQL文には、SELECT構文のような値を参照するものと、INSERT、UPDATE、DELETE構文のようにデータ操作を行うもの、EXECUTE構文のようにストアドプロシージャを実行するものの3種類があります。それぞれStatementオブジェクトのインスタンスの作成、SQL文のセット、SQL文の実行となり、SELECT文ではfetch()メソッドを繰り返してデータ値の読み取りを行います。最後に、トランザクションのコミット処理、またはロールバック処理によってSQL文操作を終了します。
Statementオブジェクトの構文は**表3**のようになります。

構文例では行データをすべて表示する形となっていますが、次のように列名を指定することで特定の列のデータを取り出すことが可能です。

```
$st->fetch( ¥%result );
print "$result{LAST_NAME}¥n";
```

すでに説明したように、SQL文操作を行った場合は、必ず明示的にトランザクションを終了させなければなりませ

構文	
IBPerl::Connection(＜パラメータ1＞,＜パラメータ2＞,……);	
パラメータ	**内容**
Server	サーバ名の指定。ローカルサーバなら省略可能
Path	データベースへのパス
Protocol	使用するプロトコル。TCP/IP、NetBEUI、IPX/SPXの値をセット可能。TCP/IPなら省略可能
User	ユーザー名。環境変数ISC_USERを設定している場合は省略可能
Password	パスワード。環境変数ISC_PASSWORDを設定している場合は省略可能
Dialect	Dialectモードの設定。1または3を設定可能。省略時は1
Charset	キャラクターセットの設定。EUCならEUCJ_0208、シフトJISならSJISを指定。省略時はNONE
Role	ロールの設定。省略時はロールなし
メソッド	**内容**
new()	データベースへの接続
create()	データベースの作成
disconnect()	データベースから切断
戻り値	
データベースハンドル。0以下なら接続失敗	
例	
$db = new IBPerl::Connection(Server => 'dkato-linux', Path => '/opt/interbase/examples/employee.gdb', User=>'sysdba', Password=>'masterkey');	

表1　IBPerl::Connection構文

ん。これはSELECT構文でデータを参照する場合も同じです。コミット処理、またはロールバック処理を実行してトランザクションを終了させてから、次の処理を行う形となります。この点を忘れずに覚えておいてください。

IBPerlではBLOBデータの取り出しも可能ですが、テキストBLOB以外のデータを扱うことはお勧めできません。これは、IBPerlが1Mバイトまでしか BLOBデータを取り出せない仕様になっているためです。また、Perlを利用して画像フォーマットの変換などを行うのは、かなり難しい作業となります。ですから、基本的に文字・数字・日付データのみを扱うものだと思ってください。

ログインページの作成

IBPerlの簡単な使い方がわかりましたので、見積書発行システムの作成に入りましょう。まずはログインページを作成します。ログイン処理の流れをまとめると**図1**のようになります。

まずはログイン時に使用するHTMLファイルを作成します。用意するファイルは**リスト1**のようになります。ログインボタンが押されると、ib-login.plが呼び出されるようにFORMタグのACTIONに設定しておきます。

続いて、login.htmlで入力されたユーザー名とパスワードからログイン可能かどうかを判断する、check_loginプロシージャを呼び出すPerlスクリプトを作成します。作成するPerlスクリプトはCGI形式となりますので、今回はcgi-lib.plというPerlのモジュールを使用します。このモジュールはブラウザから送られてきた内容をデコードしてハッシュテーブルに格納後、扱いやすいようにスカラー変数に格納してくれるなどの処理を行ってくれる便利なモジュ

構文	
IBPerl::Transaction(＜パラメータ1＞,＜パラメータ2＞,……);	
パラメータ	**内容**
Database	データベースハンドル
Active	トランザクションがアクティブかどうか。読み取り専用プロパティ
メソッド	**内容**
new()	トランザクションの開始
commit()	トランザクションのコミット
rollback()	トランザクションのロールバック
戻り値	
トランザクションハンドル。0以下なら接続失敗	
例	
my $tr = new IBPerl::Transaction(Database=>$db);	

表2　IBPerl::Transaction構文

構文	
IBPerl::Statement(＜パラメータ1＞,＜パラメータ2＞,……);	
パラメータ	**内容**
Transaction	トランザクションハンドル
SQL	SQL文
TimestampFormat	タイプスタンプ型データの場合のフォーマットの指定
DateFormat	日付型データの場合のフォーマットの指定。'%m-%d-%Y %H:%M'などの指定が可能
TimeFormat	時間型データの場合のフォーマットの指定。'%H:%M'などの指定が可能
メソッド	**内容**
new()	Statementオブジェクトの作成
execute()	SQL文の実行。戻り値が0なら成功
fetch(＜数値＞)	フェッチの実行。行IDをパラメータに指定
rollback()	トランザクションのロールバック
戻り値	
ステートメントハンドル。0以下なら接続失敗	
例	

```
my $tr = new IBPerl::Statement(Transaction=>$tr, SQL=>"SELECT * FROM PROJECT");
#全データの表示
if ($st->execute())==0 {
   while ( ($status = $st->fetch( \$result )) == 0) { print "$result\n"; }
}
```

表3　IBPerl::Statement構文

ールです。

ib_login.plではまず、login.htmlで入力されたユーザー名とパスワードを変数に格納します。続いて、InterBaseへの接続を行い、ストアドプロシージャを動作させるためにトランザクションを開始します。トランザクションを開始したら、ストアドプロシージャを実行するためのStatementオブジェクトを作成し、先ほど格納したユーザー名とパスワードをパラメータとして、check_loginプロシージャを起動します。

システムにログイン可能なときはcheck_loginプロシージャの戻り値が1になりますので、1の場合はメニューを表示し、0の場合はログインページに戻るためのリンクを表示します。ib_login.plのソースファイルは**リスト2**のようになります。IBPerlを利用してInterBaseへの接続、トランザクション操作、ストアドプロシージャの呼び出しが実行されていることがわかると思います。

PerlからInterBaseを利用できるIBPerlは、非常に便利なPerlモジュールです。IBPerlを利用することによって、Perlによる柔軟性のあるシステムを構築することができます。また、Webベースのシステムでさまざまな機能を実現することが可能となります。

▼リスト1　login.html

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html;
charset=EUC-JP">
<TITLE>見積書発行システム</TITLE>
</HEAD>
<BODY>
<FORM METHOD=POST ACTION="/cgi-bin/ib-login.pl">
<P align="center"><IMG src="title.gif" width="332"
height="103" border="0"></P>
<HR>
<P align="center"><FONT size="+2" color="#000099">
<B>システムへのログイン</B></FONT></P>
<CENTER>ユーザー名：<INPUT size=" 20" type="text"
name="login_users"></CENTER><BR>
<CENTER>パスワード：<INPUT size="20" type="password"
name="login_pass"></CENTER><BR>
<CENTER><INPUT TYPE="SUBMIT" VALUE="ログイン">
<INPUT TYPE="RESET" VALUE="リセット"><BR>
<HR>
</FORM>
</BODY>
</HTML>
```

IBPerlのダウンロードサイト	http://www.karwin.com/ibperl/
IBPerlのドキュメント	http://www.karwin.com/ibperl/ibperlug/
cgi-lib.plのダウンロードサイト	http://cgi-lib.berkeley.edu/

表4　URL一覧

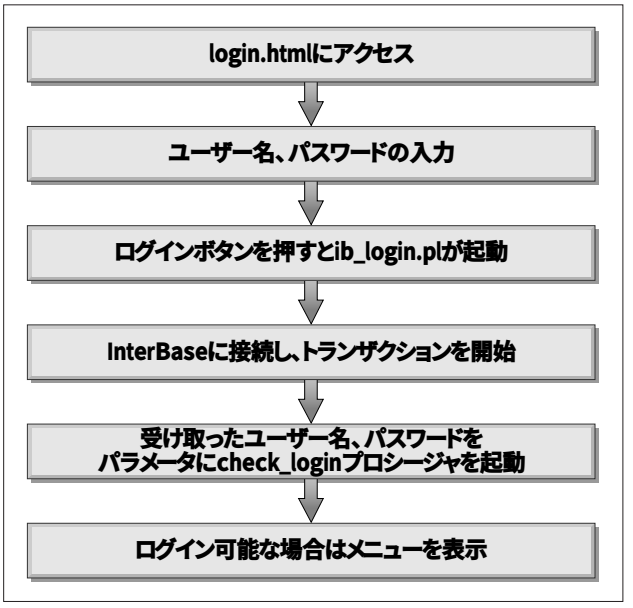
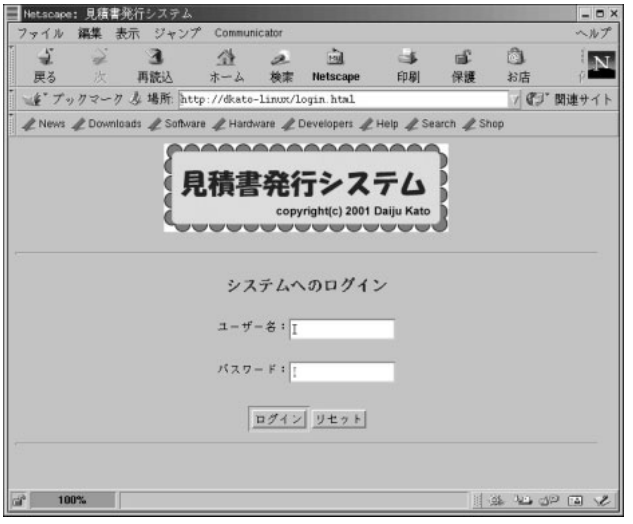
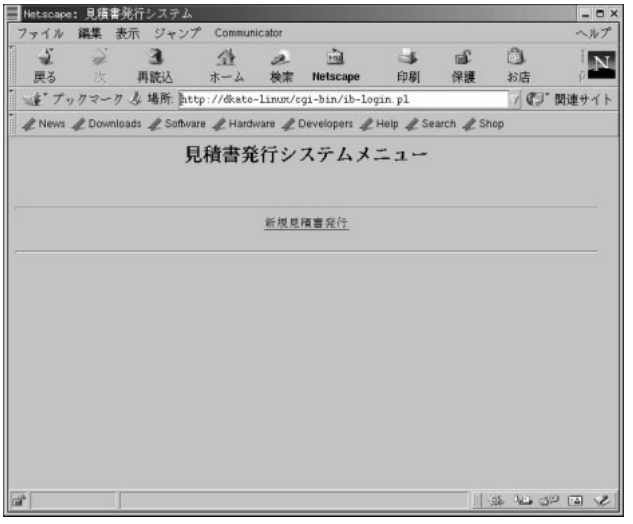


図1　ログインまでの流れ



画面2　login.htmlの画面



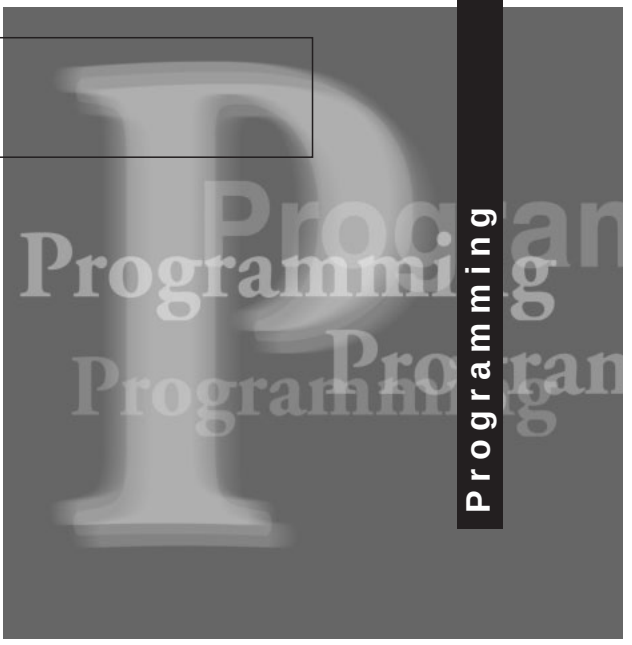
画面3　ログインに成功したところ

▼リスト2　ib-login.pl

```
#!/usr/bin/perl
$|=1;

#初期設定
use lib '/home/db';
require 'cgi-lib.pl';
use IBPerl;

#my ($db, $tr, $st, $user, $pass);

&ReadParse(*in_data);
   $user = $in_data{'login_users'};
   $pass = $in_data{'login_pass'};

print "content-type: text/html; charset=EUC-JP¥n¥n";
print "<HTML><HEAD><TITLE>見積書発行システム</TITLE></HEAD>¥n";
print "<BODY>¥n";

$db = new IBPerl::Connection(
   Path     => '/home/db/estimate.gdb',
   User     => 'SYSDBA',
   Password => 'masterkey',
   Charset  => 'EUCJ_0208');
die "Connection->new() error:¥n$db->{Error}¥n" if ($db->{Handle} < 0);
if ($db->{Handle}>0) {

   #データベースに接続できた場合
   $tr = new IBPerl::Transaction( Database => $db );
   if ($tr->{Handle}<0) { print "Handle error!";}

   #ISLOGINプロシージャの呼び出し
   $query = "EXECUTE PROCEDURE CHECK_LOGIN('$user','$pass')";
   $st = new IBPerl::Statement( Transaction => $tr, SQL=>$query);
   #プロシージャの実行
   $st->execute();

   #返値の取得
   $st-> fetch(¥%row)<0;
   foreach (keys %row)
   {
   $islogin=$row{ISLOGIN};
   }

   #ユーザー名が正しいかどうかのチェック
   if ($islogin eq 1) {
      print "<CENTER><H1>見積書発行システムメニュー</H1></CENTER><BR>";
      print "<HR><CENTER><A HREF='new_estimate.html'>新規見積書発行</A></CENTER><BR><HR>";
   } else   {
      print "<H1><CENTER>ユーザー名が正しくありません。</H1></CENTER><BR>";
      print "<CENTER><A HREF='http://dkato-linux/login.html'>[戻る]</A></CENTER><HR>";
   }

   #データベースから切断
   $db->disconnect();

} else {
   #データベースに接続できない場合
   print "エラーが発生したため、InterBaseに接続できませんでした<BR>";
}

print "</BODY></HTML>";
__END__
```

プログラミング工房

今回からはサーバ管理をするプログラムを作成しながら、C言語のプログラム手法について解説していくことにする。今月号では、これから作成するシステムの概要を示したあとに、Webブラウザでユーザーを登録する部分までを解説する。



第18回　サーバ管理プログラム（1）

文：藤沢敏喜
Text : Toshiki Fujisawa

サーバ管理プログラミング

インターネットがあたりまえとなった現在では、メールサーバやファイルサーバがいたるところで利用されるようになった。そのため、書店にはサーバ管理の本が山積みされ、だれにでも簡単に管理できるという触れ込みのサーバ製品も多数販売されている。

しかしながら、ネットワーク使用環境は会社や学校によってさまざまであり、一般的な管理方法では対処できないことが多い。また、いわゆる「簡単サーバ」に内蔵されている管理ツールでは、細かい設定ができないこともある。それに、他人が作成したツールの中身を理解せずに使っていると、問題が起きたときに自分自身で対処することが難しい。やはり、サーバ管理を適切に行うには、自分自身で使いやすいプログラムを書きたいものである。

そこで、今回からサーバを管理するツールを作成しながら、それを作るために必要なプログラミングテクニックについて解説する。なお、作成するツールはあくまでもサンプルであり、実際に運用する環境用のプログラムを自分自身で作成できるようになることが、この連載の主眼である。

対象とするサーバの規模

ひとくちにサーバといっても、家庭内で1人で使うものから、プロバイダのように数万人が使うものまでさまざまである。

家庭内で使うものであれば、ツールを作成するよりも、viで設定ファイルを編集してしまったほうが簡単だ。また、数万人規模のサーバを管理できるツールの解説は、この連載の範囲ではない。

そこで、今回は学校の研究室サーバや企業の部門サーバなど、1台あたり20人～500人くらいのユーザーを対象とする中規模なサーバ管理について考えてみたい。

このサーバではメールサービスと、Windowsクライアント向けのファイル共有を扱うものとする。クライアントがWindowsマシンであるため、このサーバは安全なファイアウオールの中にあり、直接インターネットには接続していないという前提で話を進めていくことにする。

ここでは、架空の「フジサワ宝飾株式会社」を想定し、**図1**のようなネットワーク環境を持つものと仮定する。この会社は宝石の製造販売が主体のため、情報処理関連の知識を持つ社員はほとんど皆無で、ネットワークを管理できる社員はたったの1名だけしかいないものとする。もちろん、各支店にもネットワークの知識を持つ人はまったくいない。

作成するシステムの概要

管理するユーザーの数がある程度増えると、ユーザー登

録や各種設定変更に手間がかかるようになる。たとえば3650人の会社で、年間1割の人が、入社・退職・異動・改姓……をすると、毎日1人以上の設定変更が必要になる計算になる。

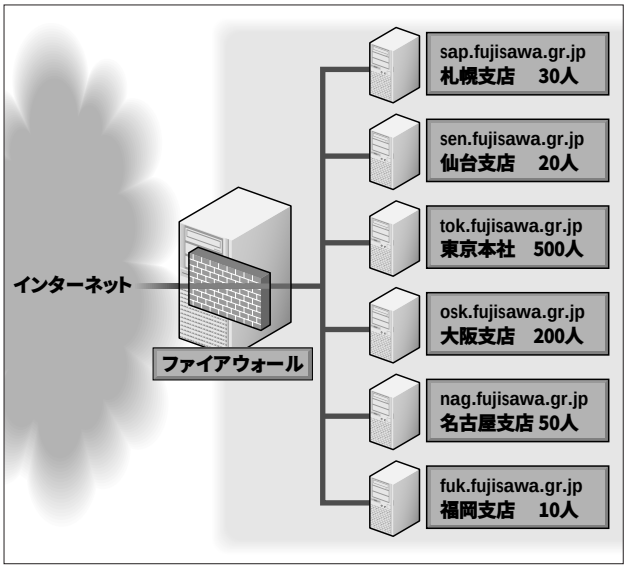


図1　フジサワ宝飾株式会社におけるネットワークとサーバの構成

設定変更作業のために、viやEmacsを自由自在に使える人間を育成することは非常にたいへんである。また、慣れない人にroot権限を与えて、Linuxconfで設定をメチャメチャにされても困る。

したがって、管理するユーザーが多い環境ではだれにでも簡単に登録や変更が可能で、必要のない操作はできないようなツールを作成する必要がある（素人はまったく考えもつかないことをしでかすものである）。

メールサービス程度ならば管理項目が少ないため、手動での設定でもさほど問題はないが、ファイルサーバは設定項目が多く、慣れた人がやっても手動ではミスが発生する。やはり専用のプログラムを準備して、間違いが起こらないようにするほうが望ましい。

そこで今回は、**画面3**や**画面4**に示すように、Internet ExplorerなどのWebブラウザから、ユーザーの登録やファイルサーバのディレクトリ管理などができるようなシステムを作成する。また、数百人を一度に登録することも想

Column

セキュリティと利便性

多くのLinuxディストリビューションに用意されているuseraddコマンドでは、ユーザーの登録時に「-p」オプションを用いて、コマンドラインからパスワードを設定できるようになっている。ここで指定するパスワードは、暗号化されたパスワードでなければならない。

なぜならば、コマンドラインでパスワードを受け渡すのはセキュリティ上危険な場合があるからである。たとえば、悪意のある一般ユーザーが、

```
while : ; do \
    ps ax | awk '$5~"useradd"{print}' ; \
done
```

というようなコマンドを実行しているときにユーザー登録を行うと、

```
3042 pts/1    R      0:00 useradd -p パスワード
```

というように表示され、パスワードが生のままだと簡単にそれがわかってしまう。

ところで、FreeBSDのadduserコマンドでは、ユーザー登録時にコマンドラインからはパスワードを設定できない仕様になっている。しかしながら、数百人単位で一括してユーザーを登録する場合などに、いちいちpasswdコマンドを用いて登録していては、サーバ管理者の睡眠時間がなくなってしまう。

このため、FreeBSDではpwコマンドを用意してバッチ的な処理を可能としている。pwコマンドでは、「-h」オプションでファイルディスクリプタを指定できるようになっていて、標準入力経由で生のパスワードを受け渡すことができるのである。

今回作成したsedusrコマンドでは、Webからの管理者用パスワードについてはセキュリティを重視し、FreeBSDのpwコマンドと同様に標準入力を用いて受け渡している。しかし、ユーザー登録時における各ユーザーのパスワード受け渡しは、セキュリティ上問題があるコマンドライン経由としている。なぜなら、サーバ管理をしていると、コマンドラインからのパスワード指定は非常に便利で、捨てがたいものがあるからである。

また、サーバ管理をしているとさまざまなトラブルがあり、実際にトラブルが起きているユーザーのIDで、Windowsにドメインログオンして、テストしなければならないことも頻繁にある。このときに、ユーザーの生パスワードを知っていないと対処できないため、sedusrコマンドでは各ユーザーのパスワードを生のままファイルに保存している。

これはセキュリティ上非常にまずいのであるが、悪意のユーザーがいない場所でのセキュリティよりも、トラブル解決における利便性を優先しなければならない場合も存在するのである（もちろん、telnetやFTP、そして余計なCGIを実行できないようにするなどの工夫が前提）。

セキュリティレベルを上げるには、公開鍵暗号を用いてパスワードを別サーバに保存するなどの方法もあるが、システムを複雑にするとトラブルもまた増えるし、開発負荷も大きい。セキュリティと利便性は相反するものなのである。

セキュリティを重視するあまり、トラブルが発生して頻繁にサーバが落ちてしまうのでは本末転倒なので、使用するネットワーク環境を考えて、セキュリティと利便性とのバランスを考える必要がある。

定し、コマンドラインから一気に登録するプログラムも作成する。

なお、Linux magazineの創刊号には、筆者が「Samba — Windowsから行うユーザー管理テクニック」というタイトルで書いた記事が記載されている。このときはExcelのCSV形式を使って、Sambaユーザーを管理する運用法について書いた。今回のシステムはこれを発展させ、Webインターフェイスを付加したものである。

ちなみに、以前の記事は、

http://linux.ascii24.com/linux/allascii/linuxmag/no_01.html

からPDFファイルとして入手可能なので、参照してみてほしい。

Webインターフェイスとコマンドラインツール

Webから管理するツールを作成する場合、まず思いつくのはシェルを使ったCGIプログラミングである。/bin/shはプログラミング言語として結構強力であり、sedやawkなどと組み合わせればかなりのことができる。

実際、本誌創刊号の記事では、簡単なシェルスクリプトのみを用いてユーザー登録やSambaの共有ファイル設定を行っている。しかしながら、root権限を持たない一般ユーザーが登録操作を行うようにするには、シェルスクリプトだけでは難しい面もある。

シェルスクリプトで機能が不足するような場面では、Perlがよく使われる。特に、ユーザー登録のように文字列処理を多用するプログラミングでは、プログラムが簡単に書けるので便利である。今回作成したシステムも、最初のバージョンはPerlで書いてあった。

このようにPerlでプログラムを作ることは可能ではあったが、プログラムが大きくなるに従いメンテナンスしにくくなった。また、運用しているうちに、細かいアクセス制御やメール用のMD5パスワード、そしてSamba用のMD4パスワード演算ルーチンなどのように、C言語で書かれたライブラリをリンクする必要も生じてきた。

また、「プログラミング工房」のメインテーマはC言語の解説であるので、この連載のネタとすることも視野に入れて、PerlからC言語に書き直すことにしたのである。

なお、C言語で記述（2700行）したのは純粋に登録作業を行うコマンドラインプログラムの部分だけであり、Web部分のインターフェイスはPerlで記述してある。

sedusrコマンド

前述のとおり、今回のシステムの最も重要な部分はC言語で記述されたコマンドである。このコマンドは、sedusr（Super EDit USeR）と呼ぶことにした。

sedusrコマンドは、WebサーバのCGIプログラムから呼ぶことを前提にしているが、シェルのコマンドラインからの使用も考慮してある。たとえば、

ユーザー名	sato
ユーザーID	1234
パスワード	mQ9W7i
英語名	K.sato
個人用ディレクトリ名称	MY
共有するディレクトリ	all plan

というようなユーザーを登録するとしよう。この場合、root権限でログインし、シェルのコマンドラインから、

```
# sedusr add 'sato,9569,mQ9W7i,K.Sato,MY,all/plan'
echo '================================>' sato
groupadd -g 9569 sato
adduser -u 9569 -g 9569 -d /smb/mail/sato -s /noshell
-c 'K.Sato' -p 'mQ9W7i' sato
edquota -g -p sedusr sato
echo 'sato,mQ9W7i' >> /smb/etc/rpw.txt
mkdir -p /smb/mail/sato
chown sato.sato /smb/mail/sato
chmod 700 /smb/mail/sato
smbpasswd -a sato 'mQ9W7i' > /dev/null
ln -s STD.BAT /smb/logon/sato.bat
chown -h sato.sato  /smb/logon/sato.bat
mkdir -p /smb/home/sato
chown sedusr.sedusr /smb/home/sato
chmod 755 /smb/home/sato
ln -s ../../data/all /smb/home/sato/all
chown -h sato.sato /smb/home/sato/all
ln -s ../../data/plan /smb/home/sato/plan
chown -h sato.sato /smb/home/sato/plan
usermod -G 'all,plan' sato
mkdir -p /smb/home/sato
chown sedusr.sedusr /smb/home/sato
chmod 755 /smb/home/sato
mkdir -p /smb/home/sato/MY
chown sato.sato /smb/home/sato/MY
chmod 700 /smb/home/sato/MY
```

画面1　ユーザー登録の実行

```
# sedusr -x add 'sato,9569,mQ9W7i,K.Sato,MY,all/plan'
```

というようなコマンドを実行すると、登録が行われる。

なお、「-x」オプションを付けないと、このコマンドの結果として、どのようなことが行われるかを示したのが**画面1**である。したがって、

```
# sedusr 引数 | sh -x
```

というような使い方も可能である。

また、

```
sato,9569,mQ9W7i,K.Sato,MY,all/plan
tanaka,8462,40PLB8,K.Tanaka,MY,all/design/product
saito,9658,4lRqdo,A.Saito,MY,all/plan/sales
aoki,8880,OjvA7N,T.Aoki,,
```

という内容を、たとえばinpというファイル名で保存し、

```
# sedusr -x add -f inp
```

として、多数のユーザーを登録することもできるようになっている。

なお、引数なしで実行すると、**画面2**に示すヘルプの表示を行う。

```
sedusr ver-0.01 Apr 1 2001 Copyright (c) 1999 2000 2001 Toshiki Fujisawa
(1) user add/del/chg by admin
     # sedusr add -f sui_file [-x]
     # sedusr add 'fujisawa,6104,pass,T-F,MYDIR,plan/proj' [-x]
     # sedusr chg 'fujisawa,6104,pass,T-F,MYDIR,plan/proj' [-x]
     # sedusr get 'fujisawa' [-x]
     # sedusr listup
(2) mail list add/del by admin
     # sedusr add-ml 'name,adm_email,passwd,no_reply,no_subj'
     # sedusr del-ml 'ml_name'
(3) alias edit by admin
     # sedusr get-alias'
     # sedusr set-alias 'inp_fname'
(4) system initialize by root
     # sedusr init
(5) make share dir by root
     # sedusr add_dir 'dir_name,group_id'
(6) passwd change by normal user
     # sedusr chg-pw 'tanaka,old_password,new_password'
(7) ML member change by normal user
     # sedusr get-ml 'ml_name'
     # sedusr set-ml 'ml_name,ml_password'
(A) sample of 'sui_file'
     fujisawa,6104,pass1,Toshiki Fujisawa,MYDIR,plan/design
     suzuki,,pass2,Taro Suzuki,,
     tanaka,,pass3,Jiro Tanaka,,plan
(B) FILES
     /smb/home              samba home dir
     /smb/data              share dir
     /smb/mail              mail home dir
     /smb/logon             samba logon dir
     /smb/dust              dust box dir
     /smb/ml                ML member dir
     /smb/etc/alias.txt     alias file
     /smb/etc/rpw.txt       raw passwd file
     /smb/etc/dir_name.txt  share dir list
     /smb/etc/ml-conf.txt   ML config file
     /tmp/.sedusr           lock file
```

画面2　sedusrのhelp画面

sedusrシステムのインストール

sedusrシステムは、FreeBSD-3.3/4.2とVMware上にインストールしたVine Linux 2.0で動作を確認してある。このシステムのインストールを行うには、まずWebサーバを動作させなければならない。Linuxconfなどでhttpdを動作させ、Webブラウザで Webサーバの動作を確認しよう。そして、Sambaをセットアップし、smbpasswdなどが動作することも確認しておく。

準備ができたらroot権限でログインし、付録CD-ROMをマウントしてから、本連載のディレクトリに移動する。そして、

```
# make install
```

を実行すると、sedusrコマンドとCGIプログラム群がインストールされるはずだ。

もし、HTMLのデータディレクトリにusrやadmといったディレクトリがある場合は、それを消してからmakeを行う必要がある。また、このシステムでは、Sambaのデータファイルをルートディレクトリにある/smb以下に置くことにしている。/smbディレクトリ以下にそのシステムで使用するディレクトリが存在する場合は、インストールできないので注意してほしい。

さらに、現在のバージョンでは上書き登録ができないので、何度も繰り返して実験する場合はvipwコマンドやvi /etc/groupを実行して不必要なものを削除し、make danger-clean を実行するとよい。ただし、danger-cleanでは/smbやHTMLファイルが全部消えるので十分注意してほしい。

なお、sedusrはもともとFreeBSD専用として作成し、今回の連載のためにLinux上でも動作するように改造したものである。そのため、今月号の解説範囲であるユーザー登録以外の機能についてはLinux上ではまだ動作していない。来月以降に順次実装していく予定である。

sedusrシステムの使用例

インストールが正常に終わったら、まず管理者用のユーザーであるsedusrのパスワード設定をするためにルートでログインをする。そして

```
# passwd sedusr
```

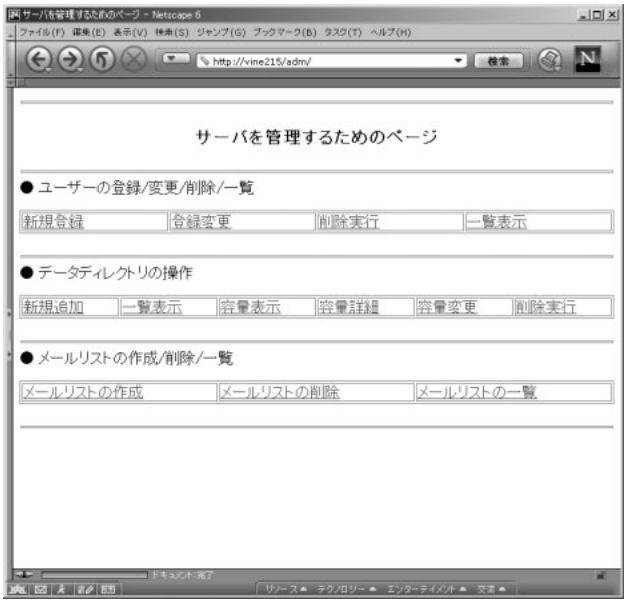
として、sedusrシステムの管理者パスワードを変更する。

次に、ブラウザで、

```
http://サーバ名/usr/
```

というURLを指定すると、**画面3**が表示されるはずである。ただし、このページは今月号のバージョンではまだ動作しない。

画面3　ユーザー情報を管理するためのページ



画面4　サーバを管理するためのページ

その次に、

```
http://サーバ名/adm/
```

を指定すると、**画面4**のようなサーバを管理するためのページが現れる。

ここで、「●データディレクトリの操作」の［新規追加］を選ぶと、**画面5**が表示される。

このページで、

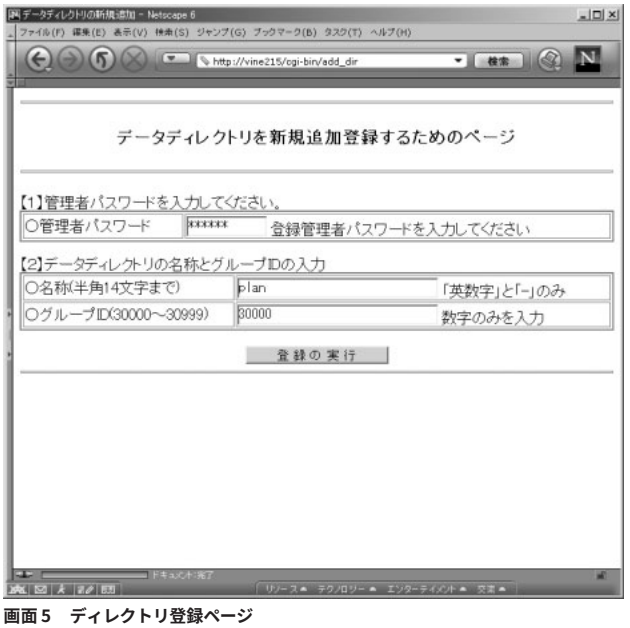
管理者パスワード　※先ほど登録したもの
名称　plan
グループID　30000

と入力して、［登録の実行］ボタンをクリックすると、**画面6**のように正常に登録が終わったことが表示され、/smb/data/planディレクトリが作成される。同様に、designおよびproductディレクトリを登録してみよう。

さて、次はユーザーの登録である。

```
http://サーバ名/adm/
```

からユーザーの［新規登録］を選ぶと画面7が表示される。ユーザー情報を入力して［登録の実行］ボタンをクリックすると、ユーザーの登録が行われる。なお、今月号のバージョンでは、まだ暗号化パスワードの処理を行っていないので、パスワードは正しく設定されない。



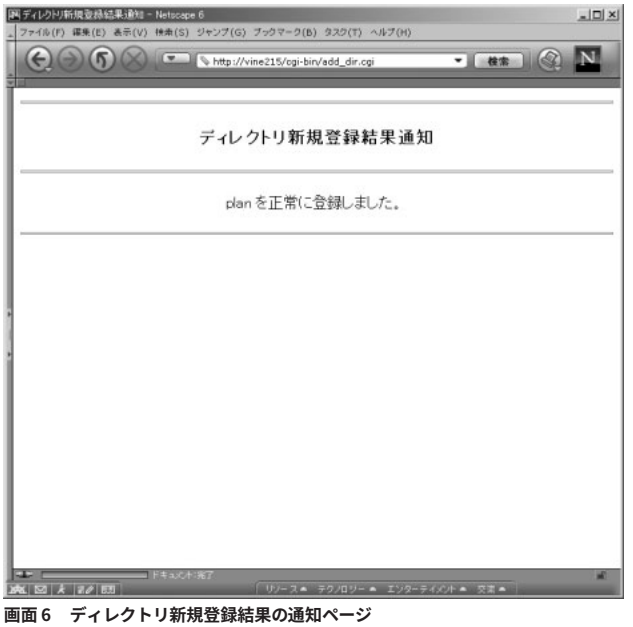
画面5　ディレクトリ登録ページ

sedusrコマンドからの外部コマンドの呼び出し

さて、本題のsedusrコマンドの中身の解説である。

このコマンドでは、ユーザー登録をする際にuseraddコマンドやchmodコマンドなどの、外部コマンドを呼び出すように設計されている。これは、実際の運用をしながら開発を進めていった都合上、何が起きるかを目で見てから実行したかったからである。

この外部コマンドの呼び出しは、たとえば、



画面6　ディレクトリ新規登録結果の通知ページ

画面7　新規ユーザーを登録するためのページ

```
int   mode  = 0600;
char *path  = "/etc/smbpasswd"
cmd("chmod %o %s", mode, path );
```

というようにして行われる。ここで、cmdは可変長の引数をとる関数で、次のように定義されている。

```
static void cmd( const char *fmt, ... )
{
    va_list ap;
    char    buf[1024];

    va_start(ap, fmt);
    (void)vsnprintf(buf, sizeof(buf), fmt, ap);
    va_end(ap);
    if( global_is_cmd_exec == false ){
        printf("%s\n", buf );
        return;
    }
    if( system(buf) != 0 ){
        fatal("[%s]", buf);
    }
}
```

上記のva_xxxxは可変長引数を扱うためのものである。詳しくは、コラムの「特別講座　ステップアップC言語」を参照していただきたい。

vsnprintfは、cmdに与えられた引数をprintfと同じようにfmtで指定されたフォーマットで、bufで示されたバッファに展開するものである。

コマンドライン引数に「-x」が指定されていない場合は、global_is_cmd_exec変数がfalseになるので、printf関数によりbufバッファに展開された内容が表示される。

一方「-x」が指定された場合は、system関数によりbufの内容が実際に実行されることになり、もし実行の結果エラーが起きてゼロ以外の値が返ってくると、fatalマクロによりエラー処理を行うことになる。

今後の予定

今月は、システムの概要とユーザー登録の方法を解説しただけで、誌面が尽きてしまった。

なお、先ほど述べたように、CD-ROMに収録したバージョンはユーザー登録の部分しか実行できない不完全なものであり、セキュリティの問題も多々ある。来月から当分の間は、足りない部分の実装や不備な部分の改善について引き続き解説していく予定である。

Column

ユーザーIDとグループID

UNIX系のOSでは、ユーザーを登録するときに、ユーザーIDと呼ばれる各ユーザーごとのユニークな番号が必要になる。OSの内部では、この番号ですべて管理されるため、これをどのようなポリシーで管理するかは重要である。

OSのユーザー登録コマンドでは、ユーザーが登録された順にユーザーIDが割り振られるが、これではサーバを変えたときに別のユーザーIDが割り振られることになる。

もちろん、これでも問題がない場合もあるが、今回想定しているように全国の支店に複数のサーバが存在する場合には、管理がきわめて面倒になることがある。たとえば、札幌支店から東京本社への人事異動があった場合、登録するサーバを変えることになるが、このときにメールスプールやデータをコピーする方法によっては、ユーザーIDが違うことによってさまざまなトラブルを引き起こす。

したがって、ユーザーIDは社員番号のように重複なしに割り当てることができ、将来に渡って変更が生じないような番号を使用することが重要である。なお、最近では派遣社員が活躍する場面も多いが、派遣の場合は人事上の社員番号を持たないこともある。このような場合でも、全社で統一的な番号管理を行わないと、あとあと収拾がつかなくなってしまう。面倒でも最初から派遣社員番号領域を定め、しっかりとしたデータベースで管理しなければならない。

一方、グループIDもユーザーIDと同様な管理が必要であるが、このIDはユーザーIDと同じ番号とするのが管理上も運用上も都合が良い（理由は、FreeBSD上でjman adduserを実行すると表示される）。

この方式では、ユーザーごとにユニークなグループIDを割り振ることにより、各ユーザーのディレクトリの制限容量をグループクオータを用いて管理することができるので便利である。

なお、今回のシステムでは、ディレクトリのアクセス権限をコントロールするため、各ディレクトリに固有のグループIDを割り当てている。このグループ番号は、社員番号領域と重ならない番号領域を割り当てておく必要がある。さらに、支店から部署がまるまる本社に異動することもあり得るので、全社のサーバで番号が重複しないように、きちんと管理しなければならない。

人間や組織（ディレクトリ）に番号を割り振るのは簡単に見えるが、きちんとした管理はたいへん難しく、そのうちいいかげんになってくる。しかし、サーバを運用するうえでID管理はたいへん重要なので、初めからしっかりとした管理体制を作り上げる必要がある。

特別講座

ステップアップC言語

可変長引数関数

■printf関数

printf関数はいくつもの引数を持つことができる。このような関数を「可変長引数関数」と呼ぶ。

このような関数は、次のようなプロトタイプ宣言がされる。

```
void printf(char *fmt, ...);
```

printfは、ご存じのように第1引数のfmt文字列を解析し、あとに続く引数の個数とその型を決め、それに従って第2引数以降を表示する。

■可変長引数関数の実現

この可変長引数関数を理解するために、次のように第1引数によって指定した型で、第2引数を順に表示するpr関数を考えてみる。

```
pr( "csl",
    (char)  'A',
    (short) 123,
    (long)  12345678 );
```

ここで、pr関数の第1引数の文字列にある「c」はchar型、「s」はshort型、「l」はlong型を示している。第2引数以降は、第1引数で指定した順番に並んでいる。

このpr関数は、次のように定義することができる。

```
#include <stdarg.h>

void pr( char *t, ... )
{
    va_list ap;
    va_start(ap, t);
    vpr(t, ap);
    va_end(ap);
}
```

上記のva_list、va_start、va_endの3つは可変長引数を実現するマクロであり、<stdarg.h>で定義される。

va_list型は、引数のアドレスを保持するためのものであり、

```
va_list  ap;
va_start(ap, 第1引数)
```

とすることにより、apに第2引数の位置を示す情報がセットされる。

これらのva_list、va_startはコンパイラによって実現方法が異なる。コンパイラによっては、後処理が必要なこともある。この後処理を行うのがva_endである。

さて、第2引数以降を処理するvpr関数であるが、これは次のように定義される。

```
void
vpr(char *t, va_list ap)
{
  char  c;
  short s;
  long  l;

  while(*t){
    switch(*t++){
    case 'c':
      c = va_arg(ap, char);
      printf("c=%c\n", c );
      break;
    case 's':
      s =  va_arg(ap, short);
      printf("d=%d\n",s );
      break;
    case 'l':
      l = va_arg(ap, long);
      printf("l=%ld\n", l);
      break;
    }
  }
}
```

ここで、tには「csl」が渡されるので、その順番に与えられた引数を取得して、表示している。引数の取得にはva_argマクロが用いられるが、

va_arg(**現在の引数位置**, **取得する型**)

として使用することにより、順番に引数の値を得ることができる。

■stdarg.h

さて、上記で説明したva_xxxxはどのようにして実現されるのであろうか？

C言語の場合、関数の引数はスタックに順番にプッシュされるので、その順番とそれぞれの引数のサイズがわかれば、ポインタを用いてアクセスすることができる。

これはさほど難しいことではなく、i386のgccで使用しているstdarg.hでは、**リスト1**のように定義されている。

__va_sizeでは32ビット単位にアライメントを行い、va_startではlastnの分だけポインタを進め、va_argではtypeのサイズ分取り出し、ポインタを次に進めていることがわかる。

```
typedef char *va_list;
#define __va_size(type)    (((sizeof(type) + sizeof(int) - 1) / sizeof(int)) * sizeof(int))
#define va_start(ap, last) ((ap) = (va_list)&(last) + __va_size(last))
#define va_arg(ap, type)   (*(type *)((ap) += __va_size(type), (ap) - __va_size(type)))
#define va_end(ap)
```

リスト1　i386のgccで使用するstdarg.h

Perlスクリプト入門
すぐにできる実践Perl

今回は複数のデータを記憶できる配列変数について解説します。また、一度に配列変数に値を入れるリストについても解説します。配列変数は、さまざまな場面で活躍するものです。必ずマスターしましょう。

第4回　配列とリスト

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

ここまでは、変数といえば$aのように1つの値（文字列）を記憶するものでした。これに対して、複数の値や文字列をひとまとめにして記憶する変数があります。それが配列変数です。今までの単純な変数が、1つの引き出しだとすれば、配列変数はタンスのようなものだと思ってください（**図1**）。

配列変数に対して、今まで使っていた$aのような変数をスカラー変数と呼びます。スカラー変数では、変数名の先頭に$を付けていましたが、配列変数の場合は、@を付けます。

```
@list, @kingaku, @a
```

スカラー変数と配列変数はまったく別のものなので、名前の部分が同じでも別のものとして扱われます。ですから、$aと@aは別のものとして扱われるのですが、このようにして使うとまぎらわしいのでできるだけ同じような名前を付けるのは避けましょう。

配列変数の中のひとつひとつの値を使うには、変数名の後ろに[0]、[1]、[2]……といった添字（そえじ）を付けます。添字の番号は0から始まることに注意してください。たとえば、配列の中の3番目の値は[2]になります。

配列全体を表すときは@なのですが、添字を付けて配列の中の1つを扱いたいときは、先頭が$になります。たとえば、@list配列の6番目は、

```
$list[5]
```

になり（$list[6] ではありません。6番目は[5]です）、これは普通のスカラー変数と同じですから、今までのスカラー変数と同じように、

```
$list[5] = "あいうえお";
```

というように使えます。@と$ を間違えないようにしてください。

```
@list = "あいうえお";
@list[5] = "あいうえお";
```

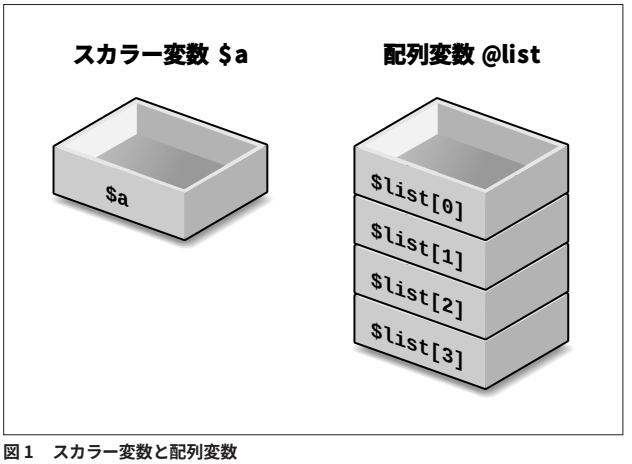


図1　スカラー変数と配列変数

はいずれも間違いです。

リスト

では、配列に数値や文字列を入れるにはどうすればよいのでしょうか。

Perlで扱う（変数の中に入れられる）データの種類は、

- **数値　1、5000、0.25など**
- **文字列　"あいう"、"abc\n"、'1234'など**

でした。これらに加えて、

```
("大根","白菜","玉ねぎ","ゴボウ")
```

のように、複数の数値か文字列を「,（カンマ）」で順に並べて、両側を( )でくくったひとかたまりのデータをリストといいます。リストも1つのデータとして扱うことができます。

リストはデータの集合体であり、配列はまさにそのような集合体のデータを入れるのにぴったりです。

配列変数に一度にデータを入れるには、

```
@yasai = ("大根","白菜","玉ねぎ","ゴボウ");
```

のように行います。このとき、配列の要素（引き出し）の数は自動的に4個になり、それぞれの内容は、

```
$yasai[0] = "大根";
$yasai[1] = "白菜";
$yasai[2] = "玉ねぎ";
$yasai[3] = "ゴボウ";
```

となります。先頭の$に気を付けてください。@ではありません。

また、リストの中には変数を並べて書くこともできますから、

```
$a = "apple";
$b = "orange";
$c = "lemon";
```

のとき、

```
@fruits = ($a,$b,$c,"tomato");
```

と書くこともできます（tomatoがfruitsかどうかはさておき……）。

さらに、リストの中には配列変数を並べて置くこともできますので、前述の@yasaiも使って、

```
@food = (@yasai,@fruits);
```

と書くことができ、結果は、

```
$food[0] = "大根";
$food[1] = "白菜";
$food[2] = "玉ねぎ";
$food[3] = "ゴボウ";
$food[4] = "apple";
$food[5] = "orange";
$food[6] = "lemon";
$food[7] = "tomato";
```

となります（これで、tomatoがfruitsかどうか気にする必要がなくなりました）。

応用すれば、

```
@food = (@yasai,"カボチャ",$a);
```

のように、配列と文字列とスカラー変数を混在させても構いませんし、

```
@yasai = (@yasai,"キュウリ");
```

と、リストの中に自分自身を書いても構いません。この場合、配列の要素が増えていくだけです。つまり、

```
@yasai = ("大根","白菜","玉ねぎ","ゴボウ");
```

のとき、

```
@yasai = (@yasai,"キュウリ");
```

とすると、結果は、

```
$yasai[0] = "大根";
```

```
$yasai[1] = "白菜";
$yasai[2] = "玉ねぎ";
$yasai[3] = "ゴボウ";
$yasai[4] = "キュウリ";
```

になります。要素が5つになりましたね。

このように、リストを使えば配列の要素をどんどん増やしていくことができるのです。

逆に、配列変数の中身を1つずつ取り出してそれぞれスカラー変数に入れるためには、リストを左辺に書いて、

```
($a,$b,$c) = @yasai;
```

とします。これで、$a,$b,$cには@yasaiの要素が[0]から順番にコピーされていきます。このとき、

```
($a,$b,@kirai) = @yasai;
```

のようにリストの最後に配列変数を書けば、結果は、

```
$a = "大根";      # ← $yasai[0]
$b = "白菜";      # ← $yasai[1]
@kirai = ("玉ねぎ","ゴボウ","キュウリ");
                  # ← $yasai[2]、[3]、[4]
```

となり、最後の配列変数（@kirai）に残り全部が入れられます（**図2**）。

実はこのように、配列変数が「＝」の左辺のリストの中にある場合、配列変数は欲ばって右辺から渡される残り全部のデータを自分の要素にしてしまいます。ですから、上の例を、

```
(@kirai,$a,$b) = @yasai;
```

と、配列を先に書いてしまうと、右辺の@yasaiの内容はすべて@kiraiに入ってしまい、$aと$bには何も入らなくなってしまいますので注意してください（**図3**）。

このような場合は、必ず配列変数を一番最後に書くようにしましょう。

特殊なリストの書式

リストには順に数値や文字列を並べて書けばよいのですが、次のような場合は簡略化して書くことができます。

```
(1,2,3,4,5,6,7,8)      →    (1..8)
(1,2,3,4,5,10)         →    (1..5,10)
(1,2,3,4,11,12,13,14)  →    (1..4,11..14)
```

また、文字列の" "が面倒なときは、

```
("算数","国語","理科","社会")
  →   qw(算数 国語 理科 社会);
```

と、空白かタブ、改行で区切ってqw( )で囲めば、" " を省略することができます。ただし、これができるのは各要素に空白が含まれていないときだけです。

配列の要素の数

配列にどんどん要素を追加していくと、今、その配列にいったいいくつの要素があるのかを知りたい場合があります。この場合、

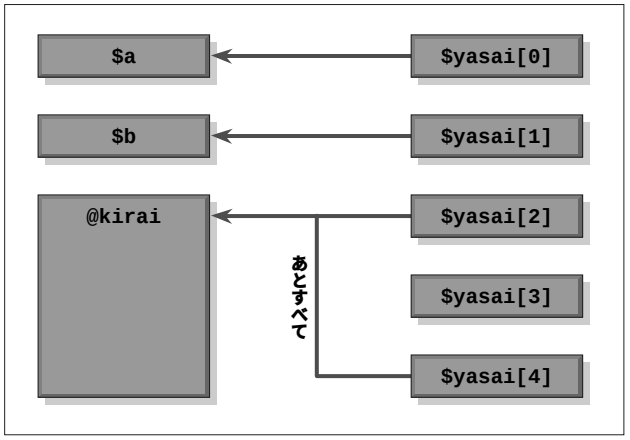


図2　($a、$b、@kirai) = @yasai;のとき

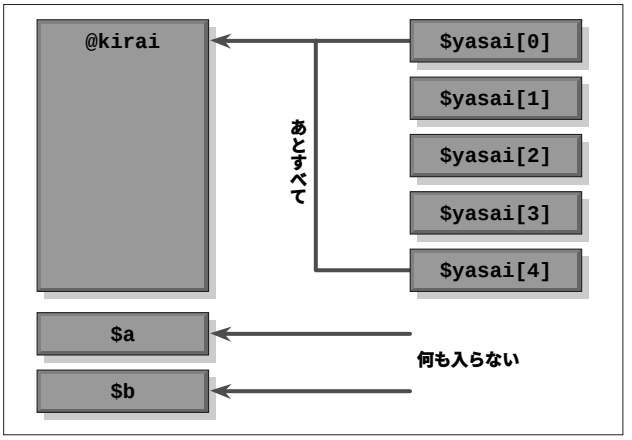


図3　(@kirai,$a,$b) = @yasai;のとき

```
$a = @yasai;
```

のように、配列変数を無理やりスカラー変数に代入してやると、$aの値は配列@yasaiの要素数になります。本来、タイプの違うスカラー変数と配列変数を＝で結ぶのはおかしいのですが、Perlにはこの場合に限って、「配列変数をスカラー変数として評価すると、その要素数を返す」という決まりがあり、エラーにならないのです。たとえば、

```
@kamoku = ("算数","国語","理科","社会");
```

のとき、

```
$a = @kamoku;
```

とすると、$aは4になるのです。ややこしいですが、間違えないようにしてください。次の例をよく見てください。

```
$a = $kamoku[0];        # $a = "算数"
$a = $kamoku[1];        # $a = "国語"
$a = $kamoku[2];        # $a = "理科"
$a = $kamoku[3];        # $a = "社会"
$a = @kamoku            # $a = 4
```

また、配列の要素の数を知る方法には別の方法もあります。@kamokuという配列に対して、

```
$#kamoku
```

と書きます。これは「配列@kamokuの最後の要素の添字」を表します。つまり、上記の例では@kamokuの最後の要素は$kamoku[3] ですから、$#kamokuは3ということになります。しかし、要素の数は4個ですから、

```
$#kamoku + 1
```

が要素の数を表すことになります。

$#kamoku は先頭の記号が$ですから、一目見てスカラー変数だということがわかるので、

```
$a = @kamoku;
```

のように、わざわざ「スカラーとして評価すれば……」などということを意識しなくても、どう代入されようがれっきとしたスカラー変数の$# を使ったほうがわかりやすいかもしれません。

ただし、$# を使う場合は、1を足さなければ要素の数にはならないことに注意してください。ですから、配列の要素数が4個かどうかを判定するには、

```
$a = @kamoku;
if ( $a == 4 ) { …… }
```

と書くか、

```
if ( ($#kamoku + 1) == 4 ) { …… }
```

と書きます。どちらも一長一短なので、好きなほうを自分のスタイルとしてください。

配列の扱いは、@、$、$#、[ ]などの記号によるものが多いため、ちょっとした記述ミスがまったく違った意味を持ってしまいます。Perlがエラーメッセージを出してくれるようなミスなら大丈夫ですが、エラーにならない場合もありますので、最初にスクリプトを書くときに記号には十分注意して、また、スクリプトの動きがおかしいときには、この配列まわりの記号表記を疑ってみてください。とはいえ、配列はとても便利なものですから、怖がらずにどんどん使って慣れるようにしてください。

繰り返し（for文）

for文は、同様の処理を複数回行うときに、繰り返し回数やいつ繰り返しを終了すればよいかを記述するものです。for文の書式は、

```
for ( 初期値; 条件式; 増減式 ) { …… }
```

になります。具体的には、

```
for ( $i=0; $i<10; $i++ ) { …… }
```

のようになり、この場合は「$iを最初に0とし、$iが10未満である間繰り返し、1回繰り返すごとに$iに1を加える」という意味になります。わかりやすく言うと、「10回繰り返す」です。

「10回繰り返す」だけでは少々乱暴すぎて、なぜそうな

るかがすぐには理解できないかもしれませんが、とりあえず、このfor文は、{ …… }の中に書かれたスクリプトを10回繰り返すものだと丸暗記してみてください。

そうすれば、15回繰り返すには……、100回繰り返すには……という場合にどうすればよいか、見当が付くと思います。条件式の$i<10の「10」を15に変えれば15回に、100に変えれば100回になるのではないかという予想は正解です。

「初期値」「条件式」「増減式」といった各部には当然それなりの意味があり、正しく解釈すれば最初の長ったらしい文章になるのですが、式とにらめっこして「これがああなって……あれが……」とうなるよりも、実際にどういう動きをするのかを試してみたほうが理解の助けになります。

次のスクリプトを実行してみましょう。

```
for ( $i=0; $i<10; $i++ ) {
    print "$i¥n";
}
```

どうですか、0から9までの数字がprintされたでしょう。このとき、print文は何回実行されたのでしょうか。printされた行数を数えてみてください。10回ですね。

「0から9ではなくて1から10までのほうがわかりやすい」と思うのも当然ですが、残念ながらコンピュータは私たちと違って、0から数を数えるのが得意なんです。そのほうが何かと都合が良いようです。たとえば、0から9なら全部1桁ですが、1から10なら10だけ2桁になるといった問題があります。

どうでもいい問題のような気もしますが、コンピュータにとっては結構大きな問題であったりします。2000年問題も、そもそもは桁数の問題だったのですから。

余談はさておき、配列の添字もそうですが、プログラミングではあらゆる場合に0から数字を数えます。文句を言っても始まらないので、そういうものかと慣れてください。

ということで、10回繰り返す場合のカウントは、0、1、2、3、4、5、6、7、8、9、になるため、最初の値は0になり最後は10未満ということになるのです。

0から始まるか1から始まるか、未満なのか以下なのかということはとりあえず気にしないのであれば、重要なのは「10回繰り返す」場合に、このfor文には「10」という数字が書かれているということです。となれば応用は簡単ですね、15回なら10を15に、100回なら10を100にというように、この数字を変えれば繰り返しの回数は自由自在に設定することができます。

実際にスクリプトを書き換えて、実行してみてください。

変数 $i

ここで使われている$iという変数は、特に特別なものではありません。繰り返しをカウントする変数として$iが慣習的に使われるのです。何の意味もない、ただの繰り返しであれば$iでよいでしょうし、もし回数を数える変数が意味を持つのならほかの変数名でも構いません。

意味を持つということは、単なる回数カウントのためだけではなく、{ }内で、何かの目的のために使われるということです。

前述のスクリプトのように、回数を数えるためだけなら0から9のカウントで構わないのですが、たとえば出席番号のように1からの数字をprintしたい場合は、for文の初期値$i=0 を変えるのではなく、{ } の中を操作します。つまりprint文を修正して、

```
for ( $i=0; $i<10; $i++ ) {
  print $i+1,"¥n";
}
```

としたり、

```
$bango = $i + 1;
```

というように、表示用の別の変数を使ったりします。この方法を使えば20から30までをprintしたり、掛け算を使えば、偶数だけをprintしたりすることができるのです。
for文を使うときは、for文内の変数の初期値はできるだけ0にして、「{ }内で数値を加工する」ほうが、あとで見やすいプログラムになります。

foreach文

先ほど配列を学習しましたが、配列の要素それぞれに同じような処理を行うとき、繰り返しが威力を発揮します。たとえば、@kamokuに4つの要素がある場合、それぞれの要素に何らかの処理を行うときは、

```
for( $i=0; $i<4; $i++ ) {
    $kamoku[$i] = ……
```

```
}
```

とすれば、$iが0から3に変化してくれるので、$kamoku[$i]は、

```
$i = 0のとき   $kamoku[0]
$i = 1のとき   $kamoku[1]
$i = 2のとき   $kamoku[2]
$i = 3のとき   $kamoku[3]
```

となって、うまいぐあいに@kamokuの全要素を処理することができるのです。

もし、@kamokuの中に要素がいくつあるかわからなくても、前述したように$#kamoku + 1で要素の数がかわりますから、

```
for ( $i=0; $i<($#kamoku+1); $i++ ) { …… }
```

または、

```
for ( $i=0; $i<=$#kamoku ; $i++ ) { …… }
```

とすれば、要素数だけ繰り返しを行うことができることになります。

しかし、もっと良い方法があるのです。上記のfor文はいずれも、「@kamokuの全部の要素に対して処理を行う」のですから、たとえ要素数がはっきりわからなくても、とにかく全部の要素に対して繰り返されればよいのです。特に要素の数を知る必要もなく、全要素に対して繰り返したいだけなら、

```
foreach (@kamoku) { …… }
```

という、foreach文が役に立ちます。実例を見ながら、for文とforeach文を比べてみましょう。

@kamokuの中身を1つずつprintする場合、for文だと、

```
@kamoku = ("算数","国語","理科","社会")
for ($i=0; $i<=$#kamoku ; $i++ ) {
    print $kamoku[$i];
}
```

となりますが、foreach文を使えば、

```
@kamoku = ("算数","国語","理科","社会")
foreach (@kamoku) {
    print;
}
```

だけでよいのです。シンプルですね。

つまり、foreachは( )内のリストの中身を順に1つずつ取り出して、デフォルト変数$_に格納（コピー）し、それを繰り返すということなのです。

1つずつ取り出された要素は都合よく$_に格納されていますから、print文にも何も変数を書かなくて済みます。正規表現を使う場合でも$_は簡単に使えたことを思い出してください。

$_以外のforeach文

foreach文でのデフォルトは$_ですが、取り出したひとつひとつのデータを格納する変数を変えたければ、

```
foreach $a (@kamoku) {
    print "$a¥n";
}
```

のように、( )の前に変数名を書くだけです。これで、$aには繰り返されるごとに@kamokuの要素が1つずつ順番に格納されていきます。

foreach文を使って並べ替え

foreach文で1つずつ取り出されるデータの順番を制御することもできます。配列@listの要素を後ろから逆順に取り出したいときは、

```
foreach (reverse @list)
```

と書きます。データの順番を昇順（文字列の場合はコード順）に並べて取り出したいときは、

```
foreach (sort @list)
```

と書きます。コードの逆順に並べたいときは、

```
foreach (reverse sort @list)
```

でよいわけです。次のようなスクリプトを実行して、ぜひ実感してください。

```
@list = ("a","dddd","ccc","eeeee","bb");
foreach (@list) {
   print "$_¥n";
}
```

このスクリプトのforeach行を以下のように変更して実行結果を確認してみましょう。

```
foreach (reverse @list) {
foreach (sort @list) {
foreach (reverse sort @list) {
```

whileとfor、foreach

繰り返し（ループ）については、以前にwhileを学習しました。whileは「何かが起こるまで」ループし続けるというイメージですが、ここで学んだforとforeachは、同じ動作を何度も繰り返すというイメージとして捉えてください。どちらもループには変わりなく、while文で書くところをfor文でも書けたりするのですが、そうなると「whileとforのどちらを使えばいいのだろう」ということで迷ってしまいます。おおよその判断基準として、

- **for**
  繰り返し回数が決まっている場合
- **foreach**
  配列変数の要素すべてに対して繰り返し行う場合
- **while**
  そのいずれでもなく、外部の状況が変わるまでループしたい場合（ファイルの終わりまでなど）

というのを目安にすればよいでしょう。ただ、必ずこうしなければならないということではありませんから、いろいろ使ってみて、自分で判断できるようになりましょう。

また、for文、foreach文においても、while文と同じようにlast;を使って{ }内ループから脱出することができます。たとえば配列の処理をしていて、途中で空の要素があればそこで繰り返しをやめるなどの場合に使います。

```
foreach (@list) {
  if ( $_ eq "" ) { last; }
  print "$_……";
}
```

このスクリプトでは、@listの要素を繰り返し表示しながら、空文字列"の要素になったときに繰り返しをやめて脱出します。

配列とパターンマッチ

時刻を表す、"11:29:30"という文字列から、時、分、秒を別々に取り出したい場合、前回学習したパターンマッチを使って、

```
$tm = "11:25:30";
$tm =~ /([0－9]＋):([0－9]＋):([0－9]＋)/;
```

とすれば、後方参照によって$1、$2、$3にそれぞれ、時、分、秒の値が取り出されます。それぞれを出力するなら、

```
$tm = "11:25:30";
$tm =~ /([0－9]＋):([0－9]＋):([0－9]＋)/;
print "$1¥n";
print "$2¥n";
print "$3¥n";
```

となるのですが、実はパターンマッチの結果を配列に代入してやると（つまり、リストととして評価されると）、

```
@list = ($tm =~ /([0－9]＋):([0－9]＋):([0－9]＋)/);
```

の結果は、

```
$list[0] = "11"       # = $1
$list[1] = "25"       # = $2
$list[2] = "30"       # = $3
```

となって、配列@listに$1、$2、$3と同じ結果が格納されるのです。これを応用すると、繰り返しのパターンマッチである、//gを使えば、

```
$s = "This is a Perl";
@word = ($s =~ /(¥w＋)/g);
```

とした結果は、

```
$word[0] = "This"
$word[1] = "is"
$word[2] = "a"
$word[3] = "Perl"
```

となってくれます。これは使えそうですね（\wは英数字にマッチ。[_A－Za－z0－9]と同じ）。正規表現の復習もかねて、いろいろ試してみましょう。

配列と<IN>

ファイルから1行ずつデータを読み込んで処理するスクリプトの基本形は、

```
open(IN,"test1.txt");
while(<IN>) {
    print;
}
close(IN);
```

というものでした。

ここでの<IN>は、whileによる繰り返しで1行ずつ読み込むためのものですが、場合によってはとりあえず全行を読み込んでから処理したいこともあります。

そんな場合にも<IN>は便利で、<IN>をそのまま配列に代入すると、なんとファイルの中身を全部配列に取り込んでくれるのです。もちろん、配列の要素それぞれがデータ1行ずつになってくれます。

ですから、ファイルの中身を全行配列に読み込んでから1行ずつ出力するスクリプトは、

```
open(IN,"test1.txt");
@line = <IN>;
foreach (@line) {
    print;
}
close(IN);
```

になります。

この例だと、whileを使った場合に比べて、かえって複雑なスクリプトになっていますが、大きく違うのはファイルの全データが、常に配列@lineに記憶されているということです。

whileを使った場合は、<IN>で読み込まれた各行は次の行が読み込まれるときには捨てられてしまうので、前の行に戻って処理したりすることができません。先頭行から最終行まで、順に処理するような場合に向いているのです。

一方、全行を配列に読みこんでしまう後者は、いつでもどの行にでもアクセスすることができます。ファイルをいったん最後まで読んでみないと処理が決まらない場合などに向いています。

スカラーコンテキストとリストコンテキスト

今回は配列を中心に学習しましたが、考え方として重要なのは配列そのものではなくて、配列変数とスカラー変数の違い、すなわちPerlにはリストとスカラーという2つのコンテキスト（文脈）があるということなのです。

いきなりコンテキストなどといってもピンとこないかもしれませんが、今回何度か触れた「スカラー変数として評価」、「（リストとして）配列に代入」ということがまさにスカラーコンテキストとリストコンテキストを意味するものです。つまり、Perlのスクリプトではその局面によって、式や変数をスカラーとして扱う場合（スカラーコンテキスト）とリストとして扱う場合（リストコンテキスト）があるということなのです。

具体例として、今回出てきた式や変数において、使われる状況がスカラーコンテキストかリストコンテキストかによって意味や振る舞いが変わるものを挙げておきます。

- **並列変数（@listなど）**
  - スカラー　要素の数
  - リスト　配列全体のリスト
- **パターンマッチの結果**
  - スカラー　マッチしたかどうか（真／偽）
  - リスト　後方参照のリスト
- **<IN>などの<>演算子**
  - スカラー　データ1行
  - リスト　データ全行のリスト

リストおよびリストコンテキストは慣れるまで少しわかりにくいかもしれませんが、リストを使いこなすのがまさにPerlを使いこなすコツです。さまざまな場面で役立ちますので、じっくり理解してください。

Rubyで行こう

「車輪を再発明する」という言葉を聞いたことがありますか？　「すでにあるものをまた作り直すのは無駄」というような意味で使います。プログラミングが楽なRubyも、ライブラリとして提供されている機能を活用すれば、ますます簡単になります。今回は、ライブラリを眺めてみます。

最終回　大いなる遺産

文：赤松智也
Text : Tomoya Akamatsu

ライブラリこそすべて

Rubyの仕様は、大きく分けると「文法」と「ライブラリ」に分かれます。文法はRubyインタプリタに組み込まれたものであり、制御構造や代入などがそれに当たります。一方、ライブラリは「クラスライブラリ」であり、メソッド呼び出しによって利用されます。

Rubyの大きな特徴は、言語仕様のうち、ライブラリの占める割合が非常に高いことです。仮にRubyからクラスライブラリを取り除いてしまえば（インタプリタそのものの実装に用いられているので、取り除くと動かないのですが）、文法だけでできるのは若干の制御構造の実行と代入くらいでしょうか。

Rubyは、クラスライブラリあってこそのRubyなのです。このことは、『Rubyデスクトップリファレンス』（オライリー発行）の本文138ページ中、実に94ページがクラスライブラリの記述であることからもうかがえます。割合でいえば7割近くになるのですから。

組み込み、標準、コントリビューション

クラスライブラリは、「組み込みライブラリ」、「標準添付ライブラリ」、「コントリビューションライブラリ」に分類することができます。

「組み込みライブラリ」は、Rubyインタプリタに初めから組み込まれているクラスライブラリです。文字列クラスや配列クラスがこれに当たります。Rubyの組み込みクラスは強力で、これだけでも結構なことが実現可能です。

「標準添付ライブラリ」は、Rubyインタプリタと同時にインストールされるライブラリです。

最後の「コントリビューションライブラリ」は、RAAなどから独自に入手する必要があるライブラリです。もっとも、いくつかの著名なライブラリは、DebianやKondaraでは、パッケージによって簡単にインストールすることができます。

巨人の肩に乗る

表1を見てください。Rubyの「標準添付ライブラリ」の一覧です。いろいろあるでしょう。

これらのライブラリを使うためには、まずrequireによってロードする必要があります。たとえば、cgiライブラリを使うためには、

```
require "cgi"
```

とします。サブディレクトリに格納されているもの、たとえばnet/ftpライブラリについては、

ライブラリ	内容
cgi	CGIライブラリ
cgi/session	CGIセッションライブラリ
complex	複素数クラス
curses	cursesインターフェイス
date	日付クラス
dbm	DBM（ndbm）
debug	Rubyデバッガ
delegate	Delegateデザインパターン
e2mmap	例外ユーティリティライブラリ
English	記号変数に対する英語の別名
etc	etcアクセスライブラリ
fcntl	fcntl定数
finalize	ファイナライザ
find	ディレクトリツリーを走査
ftools	ファイル操作
gdbm	DBM（gdbm）
getoptlong	GNU getoptlong compatible
importenv	access environment variables as globals
jcode	日本語文字列処理（Stringクラスのメソッドを置換）
kconv	日本語文字コード変換（nkfを利用）
mailread	メール読み込みライブラリ
mathn	拡張数学操作（有理数を使用）
matrix	行列クラス
md5	MD5ライブラリ
mkmf	Makefile maker
monitor	スレッド用排他モニタ
mutex_m	スレッド用Mutex Mix-in
net/ftp	ftpライブラリ
net/http	HTTPライブラリ
net/imap	IMAP4ライブラリ
net/pop	POP3ライブラリ
net/smtp	SMTPライブラリ
net/telnet	telnetライブラリ
nkf	nkfインターフェイスライブラリ
observer	Observerデザインパターン
open3	サブプロセス通信用open
parsedate	日付文字列解析ライブラリ
ping	ネットワークping
profile	Rubyプロファイラ
pstore	簡易オブジェクト指向データベース
pty	pty（疑似tty）クラス
rational	有理数クラス
readline	GNU readlineインターフェイス
sdbm	DBM（パブリックドメイン）
singleton	Singletonデザインパターン
socket	ソケットライブラリ
sync	2フェーズロックライブラリ
tcltklib	Tcl/Tkインターフェイス
tempfile	temporary file that automatically removed
thread	スレッドライブラリ
thwait	スレッド同期ライブラリ
timeout	provides timeout
tk	Ruby/Tk
tracer	実行トレース
weakref	weak reference class
Win32API	Win32 APIライブラリ

表1　標準添付ライブラリの一覧

```
require "net/ftp"
```

という形でロードします。

これらをグループ（族）に分類して、紹介します。

ネットワーク族

まず最初に紹介するのは、ネットワーク関係のライブラリです。

■mailread

mbox形式のメールスプールを読み込んで、メールヘッダーを解析するライブラリです。Rubyの初期から提供されているライブラリで、今となってはなんとなく古臭いのですが、スプールにたまってるメールのFrom行の情報を切り出すといった処理には、それなりに便利です。

■socket

ネットワークソケットを実現する拡張ライブラリです。このライブラリは、BasicSocket、IPSocket、TCPSocket、TCPServer、UDPSocket、UNIXSocket、UNIXServer、Socketの各クラスを提供します。おそらく最もよく使われるのは、クライアントとしてはTCPSocket、サーバとしてはTCPServerでしょう。

Socketクラスは、OSの全ソケットインターフェイスを利用するためのクラスですが、OSのインターフェイスに直接アクセスするので、プログラミングはそれなりに繁雑になります。RubyのソケットはIOのサブクラスなので、ソケットに対してもふつうに入出力ができる点は便利です。

■ping

本物のpingはICMPを使っているのですが、こちらはTCP/IPのechoサービスを使って実現しています。でも、最近はセキュリティなどの観点から、echoを始めとする不要なサービスは止めていることが多く、あんまり役に立たないようです。本物同様、ICMPを使ったpingの実装（ICMPping）もRAAに登録されていますが、スーパーユーザー権限が必要なのが難点です。

■net/telnet

telnetサーバと通信するライブラリで、プロキシにまで対応している本格的なものです。このライブラリがあれば、

telnetクライアントが簡単に書けます。telnetクライアントをなぜ、と思われる方もいるでしょう。

Rubyがtelnetアクセス機能を持つということは、telnetにマクロ言語としてRubyが組み込まれているのと同じ意味を持ちます。Rubyほど強力なマクロ言語はほとんどありませんから、オートパイロットとして最高の機能を実現できることになります。

実際、niftyのオートパイロットプログラムであるnif.rbも、このライブラリを使って実現されています。

■net/ftp

FTPアクセスライブラリです。これも、たとえばFTPをミラーリングするプログラムや、インテリジェントなFTPクライアントに応用できます。

■net/http

HTTPアクセスライブラリです。以下同文（笑）。

これらのライブラリを使うことのメリットは、Rubyの強力な機能を使ってネットワークアクセスを自動化できることでしょう。もちろん、ふつうにHTTPアクセスを行うためには、NetscapeなどのWebブラウザを使えばよいわけですが、Webページの更新状況の確認やWebページのダウンロード、Webサイトのミラーリングなどに、Rubyの機能を活用する余地があります。なにより、思い通りにプログラミングできるのはすばらしいことです。

net/httpライブラリのサンプルとして、超簡易HTTPクライアントを紹介します。HTTPのアクセスが、実質2行でできていることに注目してください。

```
require 'net/http'

host = "localhost"
port = 80
path = "/"
resp, body = Net::HTTP::new(host, port).get(path)
print body
```

■net/pop、net/smtp

POP3プロトコルを話すライブラリと、SMTPを話すライブラリです。これらは、2000年12月号で紹介しました。

■net/imap

IMAPプロトコルを話すライブラリです。以前メール関係について紹介したときには、まだ標準には含まれていませんでした。基本はPOPと同じようなものです。ただし、ただ単にメールをサーバから受け取るだけのPOPと違って、IMAPはメールをサーバ側に保存し、サーバ側で多くの処理を行うために機能が圧倒的に豊富です。サーバ側でサーチまでできます。

■cgi

CGIは、Common Gateway Interfaceの略で、Webの動的ページを生成する基本的テクニックです。CGIについては、別の機会があれば、きちんと説明したいと思っています。

■cgi/session

HTTPプロトコルには状態という概念がないので、セッションやトランザクションを実現するのはたいへんです。1人が連続してページを参照しても、サーバ側ではそれが一連のアクセスであることを認識するのをHTTPはプロトコルとしては支援しないのです。

cgi/sessionライブラリはCookieなどの技術を使って、HTTPプロトコル上でのセッションを実現します。セッションは、Webを使ってオンラインショップを作ったりするためには欠かせない技術です。

時間族

時間族には、日付を表現するDateクラスを提供するdateライブラリと、「日付っぽい」文字列から時間情報を取り出すparsedateがあります。

dateライブラリは、Dateクラスを提供します。Dateクラスと組み込みクラスのTimeクラスは、どう違うのでしょうか？　Timeクラスは主に時刻情報を表現するクラスで、グリニッジ標準時で1970年1月1日以前を表現することができません（Ruby 1.7.0では、OSが許せば、1902年まで表現できるようになりました）。

一方、Dateクラスは時刻を表現することはできず、日付の情報しか持ちません。しかし、表現できる範囲にはほぼ制限がありません。利用可能なメモリサイズだけに制限されます。Dateクラスの日付の基点は、紀元前4712年1月1日です。この日は、西洋では宗教的に重要な日のようです。詳しくは知りませんが。

これらのクラスと組み込みのTimeクラスについては、この連載の2000年9月号で詳しく紹介しました。

デザインパターン族

デザインパターンとは、クラス構成などのプログラム設計上のパターンのことを言います。ライブラリが、主に具体的な機能を1つのクラス（とその下請けクラス）で実現しているのに対して、デザインパターンはむしろ機能ではなく、構成を主眼にしています。

代表的なデザインパターンとして、『デザインパターン』（ソフトバンク発行、ISBN4-7973-1112-6）には、以下のようなものが紹介されています。

- **Observerパターン**
- **Delegatorパターン**
- **Strategyパターン**
- **Templateパターン**

デザインパターンは、一般的にはクラス図などで表現されます。しかし、Rubyのような柔軟で強力な言語では、いくつかのデザインパターンについてはライブラリとして提供することが可能です。Rubyの標準ライブラリに含まれるデザインパターンを実現しているライブラリは、delegate、observer、singletonの3つがあります。

・delegate

Proxyパターンを実現する。オブジェクトに送られたメッセージを、別のオブジェクトにそのまま転送する。

・observer

Observerパターンを実現する。あるオブジェクトが変化したことを、あらかじめ登録したObserver（観察者）に通知する。提供するのは観察者のクラスではなく、観察されるオブジェクトにMix-inするためのモジュールObservableである。

・singleton

あるクラスのオブジェクトが、システム全体で唯一であることを保証するSingletonパターンを実現する。このライブラリが提供するSingletonモジュールをインクルードしたクラスは、インスタンスを1つしか生成できない。

Rubyとデザインパターンについては、『オブジェクト指向スクリプト言語Ruby』（アスキー発行、ISBN4-7561-3254-5）の第5章に詳しく紹介されています。

ユーザーインターフェイス族

GUI、CUIなどのライブラリのうち、標準添付されているのはcurses、readline、tcltklib、tkです。

cursesは、キャラクタベースのインターフェイスライブラリで、Window（キャラクタ画面上の矩形領域）なども扱うことができます。

readlineは、bashやgdbなどでも使われているGNUreadlineライブラリへのインターフェイスで、1行編集やヒストリ機能を提供します。irbなどでも利用されています。

tcltklibは、Tcl/Tkをリンクして、その機能を呼び出すための拡張ライブラリです。tcltklibは、Tclの生の機能をそのまま呼び出すことができますから、Tclスクリプトからはほぼ機械的に翻訳できますが、インターフェイスはやや不自然という印象があります。tcltklibを使ったプログラムは、以下のようになります。なんというか、結局Tclを呼んでるって感じですね。

```
require "tcltklib"

ip = TclTkIp.new
ip._eval("button .b -text {hello world} -command exit")
ip._eval("pack .b")
TclTkLib.mainloop
```

一方、tkライブラリは、tcltklibを使ってよりRubyらしいインターフェイスを提供するライブラリです。tkライブラリのサンプルは、以下のようになります。こちらのほうがRubyらしいですね。

```
require "tk"

b = TkButton.new(nil, "text"=>"hello world",
                                  "command" => "exit")
b.pack
Tk.mainloop
```

Ruby用のGUIライブラリには、ほかにもGtk、Fox、Fltk、Qtなどがありますが、これらは別に入手して、インストールする必要があります。

データベース族

標準ライブラリでは、本格的なデータベース機能は提供されていません。ですから、ちゃんとしたデータベースを使おうと思えば、RAAなどからデータベースライブラリをダウンロードする必要があります。現在、Rubyには、Oracle、PostgreSQL、MySQL、InterBaseなどに対応したライブラリがあります。

しかし、標準ライブラリの範囲内でも、簡易的なライブラリ機能はあります。それはdbmとpstoreです。

dbmは、文字列から文字列へのハッシュをファイルに記録するライブラリです。実装には、OSのdbmライブラリを使いますが、なぜか3種類（ndbmを使うdbm、gdbmを使うgdbm、実装込みで配布されているsdbm）が提供されています。

それぞれには、以下のような特徴があります。

・dbm

OS提供のndbmを使う。ほとんどのUNIX系OSで提供されるが、性能などはまちまち（遅かったり、データサイズの制限があったり）。Linuxでは、実際にはgdbmが使われる。データファイルは、ほかのOSやCPUでは使えない。

・gdbm

データサイズの制限がなく速度的にも有利だが、すべてのOSで提供されているとは限らない。

・sdbm

パブリックドメインの実装が含まれているので、どこでも使えるという利点がある反面、速度はさほど速くないうえ、キーと値の文字列の長さが合計1Kバイト弱という制限はかなり厳しい。

pstoreのほうは、オブジェクト指向データベースです。pstoreの使い方は、以下のようになります。

```
require "pstore"       # pstoreをロードする

d = PStore.new(path)   # データベースオブジェクト作成
d.transaction {        # トランザクションの開始
  d["root"] = 1234     # 自由にデータを操作する
  ...
}                      # トランザクションが終われば、
                       # dから参照される
                       # オブジェクトがすべて
                       # データベースに書き出される
```

pstoreについては、2000年7月号でも紹介しました。

数学族

Rubyは数学もできます。標準ライブラリによって、数のデータ型を追加できます。

`rational`	**有理数**
`complex`	**複素数**
`matrix`	**行列**

有理数とは、要するに分数のことです。割り切れなくても、正確に計算できます。

`mathn`	**数学拡張ライブラリ**

mathnをロードすると、Rubyの数値計算が上記のライブラリを利用して行われるようになります。たとえば、

```
a = 1/3
```

は、普通なら1/3が切り捨てられて0になりますが、mathnを使えば、きちんと1/3になります。

数の挙動までライブラリで変えてしまえるのがRubyのすごさですが、ここまで変えていいのかという気もします。

スレッド族

Rubyには組み込みのスレッド機能がありますから、どこでも同じように動くスレッドが使えます。標準添付ライブラリには、スレッドプログラミングを支援するライブラリがいくつか含まれています。

■thread

スレッドプログラミングに基本的なクラスを提供します。このライブラリで提供されるのは、Mutex（排他用フラグ）、ConditionVariable（条件変数）、Queue（キュー）、SizedQueue（サイズ限定キュー）です。

■mutex_m

Mutex_mモジュールを提供します。Mutex_mモジュールをインクルードしたクラスのオブジェクトは、それ自身がMutexとして排他機能を持ちます。

■monitor

特定のコード領域において、排他を行うモニタ機能（Java のsynchronizedに似た機能）を実現します。このライブラリは、MonitorクラスとMutex_mと同様に、オブジェクトに排他機能を追加するMonitorMixinを提供します。

■sync

2フェーズロックを実現します。このライブラリは、Syncクラスとオブジェクトに2フェーズロック機能を追加するSync_mを提供します。

■thwait

複数のスレッドを同時に待つ機能を提供するThreadsWaitクラスを提供します。

OS API族

OSの機能を利用するものです。これらのライブラリの多くは、OSごとの差を埋める働きがあります。

■etc

/etcの情報、特に/etc/passwdの情報を獲得します。

■fcntl

OSごとに異なるfcntlシステムコール用の定数を提供するためのモジュールを提供します。このモジュールは定数だけで、メソッドは定義しません。fcntlは、IOクラスのメソッドです。

■find

findコマンドのように、ディレクトリツリーのファイル名を順番に処理するライブラリです。次のようにすると、path以下の各ファイルに対して繰り返すことができます。

```
Find.find(path) do |f|
  ...
end
```

■ftools

コピー、移動、比較などのファイル操作メソッドをFileクラスに追加するライブラリです。

■open3

サブプロセスを起動し、標準入力、標準出力、標準エラー出力に対応するIOを3つ返します。組み込みのopen（）メソッドは、標準エラー出力をつなぎかえないので、標準エラー出力を捕捉したいときにはこちらを使います。

```
in, out, err = Open3.popen3(command)
```

■pty

pty（疑似tty - pseudo tty）クラスを提供するライブラリです。ptyは、プロセスをttyにつながっているかのように起動するものです。open3に似ていますが、起動されたプロセスからは標準入出力がttyにつながっているように見え、起動したプロセスからは普通のIOに見えます。

```
in, out = PTY.spawn(command)
```

▼リスト1　getoptlongサンプル

```
GetoptLong.new(["--size", GetoptLong::REQUIRED_ARGUMENT],
               ["--quiet", "-q", GetoptLong::NO_ARGUMENT],
               ["--help", "-h", GetoptLong::NO_ARGUMENT]).each{|name,val|
    case name
    when '--max-size"
      ....
    when '--quiet"
      ....
    when '--help"
      ....
    end
}
```

■Win32API

WindowsのAPIをコールします。もちろん、Linuxからは使えません。私にはよくわかりませんが、『Rubyを256倍使う本　邪道編』を読みこなすと、使えるようになるかもしれません。

開発ツール族

プログラム開発に便利なライブラリもたくさんあります。

■getoptlong

GNUのgetoptlongライブラリと同等のインターフェイスを持つライブラリです。**リスト1**のようにして使います。

これだけで、面倒なARGVの解析やエラーチェックまで勝手にやってくれます。

■weakref

GCを妨げない参照を実現します。WeakRefで参照されているオブジェクトは、他からの参照がなくなれば、GCの対象になり回収されます。参照先が回収されているWeakRefを参照すると、WeakRef::RefError例外が発生します。

■tempfile

一時ファイルを作ります。一時ファイルは、プログラムが終了するときに自動的に削除されます。

■timeout

処理が時間内に終了しなかった場合に、強制的に終了させます。以下のようにして使います。

```
timeout(10) do  # 10秒以内
   ...
end
```

処理が時間内に終了しなかった場合には、TimeoutError例外が発生します。

■debug、profile、tracer

debug（デバッガ）、profile（プロファイラ）、tracer（実行トレース）の3つのライブラリは、コマンドラインから、-rオプションで指定して使います。

■md5

MD5ハッシュを計算します。MD5は文字列に対応する16バイトの「ハッシュ値」を計算するものです。文字列が変化すれば、ハッシュ値も変化します。MD5はファイルが変更されていないことの検証や暗号化などに用いられます。ファイルのMD5ハッシュを計算するmd5sumというコマンドがあるのですが、これと同様の結果は以下のようにして求められます。

```
puts MD5.new(open(file).read).hexdigest
```

国際化族

Rubyは日本生まれですから、日本語を扱うためのライブラリも含んでいます。

■kconv

日本でよく用いられる文字コードであるEUC-JP、SJIS、JIS（ISO-2022-JP）を相互に変換するライブラリです。kconvは後述するnkfモジュールを使って実装されています。Kconv.kconv（文字列, 変換先文字コード［, 変換元文字コード］）として、文字列を変換します。文字コードはKconvモジュールの定数で指定します。

Kconv::AUTO	自動判別
Kconv::JIS	JIS（ISO-2022-JP）
Kconv::EUC	EUC-JP
Kconv::SJIS	シフトJIS（MS漢字コード）
Kconv::BINARY	バイナリ
Kconv::UNKNOWN	不明（AUTOと同じ）

また、文字列の文字コードを推測するguessメソッドも提供されています。

```
Kconv.guess(文字列)
```

戻り値は、上述の定数のうちのいずれかです。

■nkf

nkf（Network Kanji Filter）のソースコードを流用して作った漢字コード変換ライブラリです。こちらは、kconvと比較すると、nkfの引数などがそのまま見える作

りになっていてやや使いにくいのですが、nkfのすべての機能を使うことができます。変換機能は、NKF.nkf（オプション, 文字列）として呼び出します。具体的には、NKF.nkf（"-Sxe", str）のように実行します。このオプションは、「SJISからEUCに変換。1バイトカナを保存」という意味になります。オプションの意味については、nkfのマニュアルを参照してください。

■jcode

Rubyの正規表現は、日本語を文字単位に処理することができますが、文字列処理は基本的にすべてバイト単位です。ですから、

```
a = "あいうえお"
puts a[0]
```

とすると「あ」ではなく、「あ」の前半のバイトが得られます。jcodeライブラリは、この文字列処理を文字単位で行うようにしてくれるライブラリです。ただし、以下のような欠点があります。

- **全部の文字列処理が文字単位になるので、バイト単位の処理ができなくなる**
- **文字列処理が全般に遅くなる**

実際には、正規表現による文字単位の処理で十分な場合が多いので、jcodeを使うことはあまりないようです。

Rubyの将来のバージョンでは、文字単位に文字コードを指定することができるようになるそうなので、スピードを犠牲にすることなく、文字単位で操作することが可能になると思います。

まとめ

というわけで、今回は組み込みでない「標準添付ライブラリ」について紹介してみました。これらのクラスのより詳細なリファレンスについては、『Ruby ライブラリ編』（アスキー発行、ISBN4-7561-3697-4）に詳しく紹介されています。ところで、この本ってプログラム例があるともっと良かったと思いませんか？　惜しいなあ。

意外（？）にも、1年半以上の長期連載となった「Rubyで行こう」ですが、今回をもって最終回とさせていただきます。**表2**は、本連載の軌跡です。長い間、本当にありがとうございました。いつかまたどこかでお目にかかりましょう。

1999年12月号 第1回	Rubyとの遭遇（紹介）
2000年1月号 第2回	ユー・ガット・Ruby（紹介、テキスト処理）
2000年2月号 第3回	スキャナーズ（正規表現）
2000年3月号	休載
2000年4月号 第4回	ストリーム（入出力）
2000年5月号 第5回	DNA（他言語からの影響）
2000年6月号 第6回	コールバック（ブロック）
2000年7月号 第7回	データよ永遠に（永続化）
2000年8月号 第8回	ウォーゲーム（ゲーム）
2000年9月号 第9回	タイム・アフター・タイム（時間と時刻）
2000年10月号 第10回	十戒（落とし穴、べからず集）
2000年11月号 第11回	新たなる未知へ（Ruby 1.6）
2000年12月号 第12回	ポストマン（メール送受信）
2001年1月号 第13回	アウトブレーク（Rubyの発展と歴史）
2001年2月号 第14回	ツインピークス（Perl/Ruby Conferenceレポート）
2001年3月号	休載
2001年4月号 第15回	チェーンリアクション（プロモーション）
2001年1月号 第16回	コレクター（用語解説／海外事情）
2001年6月号 最終回	大いなる遺産（標準ライブラリ）

表2　今までの連載内容

Column

Rubyを256倍使う本　魔道編

今度は魔道編ですか。魔道編と言っても、魔術か何かを取り扱うわけではありません。魔道編のテーマは「RD」です。

RDは「Ruby Document」の略で、なんとなくプレーンテキストのように見えるが、ちゃんと構造も表現しているという、Rubyさながらの「お手軽さ」を実現しているドキュメントフォーマットです。

最近話題になるドキュメントフォーマットが、HTMLにしてもXMLにしてもマークアップが目立ちすぎて、本文が埋没してしまいそうなのと対象的です。RDについては、先月も解説しましたね。RDは、Rubyのドキュメントを書くために開発された（のだと思う）のですが、すでにそれを越えて、一般的な原稿やWeb日記を書いたりするためなどにも使われています。

本書は、RDとその処理ツール「RDtool」について詳しく説明しています。しかも、それだけにとどまらず、EmacsモードなどのRD支援周辺ツールまで扱っています。表を簡易にするRDtoolの兄弟「RTtool」なども魅力的です。でも、やっぱりこれは『RDを256倍使うための本』ですよね。ま、「Ruby256倍」シリーズはみんなそんな感じですし、決して悪いことではない（かえって良い感じ）と思います。

あと、ちょっと気になったのですが、著者のるびきちさんは、RDtool作者（Toshさん）とRDの発案者（まつもとさん）を混同しているように感じられました。ちょっと残念です。

[超]入門シェルスクリプト

今回は、スクリプトの動作を実行時に変更するコマンドラインオプションの基本的な処理方法や、オプションを解析するコマンドgetopt／getopts、getoptsとペアで使われるwhile制御構造などについて説明し、画像をウィンドウに表示するシェルスクリプトを作成する。

第8回　getopt(s)を使ったオプション処理

文：大池浩一
Text:Koichi Oike

本連載の第1回（2000年11月号）でも説明したように、LinuxなどのUNIX系OSでは、あるファイルが実行可能かどうかは、そのファイルの実行パーミッションの設定による。シェルスクリプトや、Perl、AWK、Rubyなどのスクリプト言語で書かれたスクリプトの場合、ファイルの内容は単なるテキストなのだが、「chmod +x ファイル名」として実行パーミッションを設定することで、C言語などで作成された通常のコマンドと同様に実行できるのだ

ところで、実行したコマンドが意図した通りに動作していれば、一般的なユーザーはそれが通常のコマンドなのかスクリプトなのかは気にしないものだ。たとえば、Xを起動するために使用するコマンドstartxが実はシェルスクリプトなのだということを、いったいどれほどのユーザーが意識しているだろうか。

このことは、たとえシェルスクリプトであっても、できるだけ通常のコマンドと同じようにコマンドラインを取り扱うべきだということを示唆している。たとえば、「－」で始まる引数は動作を変更するオプションで、引数で指定されたファイル名はすべて処理する、ファイル名が1つも指定されなかった場合は標準入力から読み込む（あるいは使い方を表示する）といったインターフェイスだ。

特に、自分以外のユーザーにも使われるシェルスクリプトを作成する場合、できるだけ通常のコマンドと同じようにコマンドラインを処理して、スクリプトであることを意識させないことが重要になる。

シェルスクリプトでのオプション処理

UNIX系OSのコマンドラインインターフェイスでは、コマンドの動作を変更するために、「－」で始まる起動時オプションが使われる。たとえば、カレントディレクトリのファイル一覧を表示するlsは、-lオプションで詳しい情報を表示し、通常は表示されないドットファイルも-aオプションで表示されるようになる。

シェルスクリプトでこうしたオプションを処理する場合、オプションを記述する形式が何種類もあることが問題になる（たとえば、複数のオプションを「-la」のようにまとめて書くか、「-l -a」のように分けて書くかなど）。すべての形式に対応するスクリプトを一から作成するのは大変なので、オプションの解析を代行してくれる外部コマンドgetoptや、さらに機能が強化されたbashの組み込みコマンドgetoptsが用意されている。

以下では、

- **if制御構造などを使った簡単なオプション処理**
- **getoptを使った本格的なオプション処理**
- **getoptsとwhile制御構造の使い方**

などの説明のあと、実際にgetoptでオプションを処理するスクリプトの解説と作成を行う。

●if制御構造などを使ったオプション処理

まずは、if制御構造やcase制御構造などを組み合わせたオプション処理について説明する。題材としてこの連載の第1回で作成した「wless」を取り上げる。これは、指定したファイルの内容を、独立したktermのウィンドウにlessを使って表示するスクリプトだ。

```
#!/bin/sh
kterm -geometry 80x25 -fg black -bg white -e less
"$@" &
```

スクリプト中でバックグラウンド実行されるktermには4つのオプションが指定されている。「-geometry 80x25」はウィンドウサイズを80桁×25行、「-fg black」は文字色を黒、「-bg white」は背景色を白に設定する。最後の「-e less "$@"」は、ktermのウィンドウ中でlessを起動する指定だ。「"$@"」は、スクリプト実行時のコマンドライン引数（ここではファイル名）を1つずつ「"」で囲んで並べたものに展開される。

このスクリプトに、コマンドラインオプションの処理を追加して、動作を変更できるようにしていこう。

(1)オプションが1つだけの場合

使用するオプションが1つだけなら、if制御構造を使って比較的簡単に処理できる。たとえば、スクリプトの実行時に-bオプションを指定して、

```
$ wless -b hoge
```

とすると、通常とは逆の配色（文字色を白、背景色を黒）で表示するスクリプトは次のようになる。

```
#!/bin/sh
colors='-fg black -bg white'
if [ "$1" = -b ]; then
  colors='-fg white -bg black'
  shift
fi
kterm -geometry 80x25 $colors -e less "$@" &
```

まず、シェル変数colorsに、文字色を黒、背景色を白に設定するktermのオプション文字列「-fg black -bg white」を格納する。colorsは最後の行で実行するktermの引数で参照され、-bオプションが省略された場合はオリジナルのスクリプトと同じ配色になる。なお、オプション文字列には空白文字が含まれているため、シェル変数に格納する際には両端を「'」（または「"」）で囲む（クォーティングする）必要がある。

続いては、if制御構造を使って、-bオプションが指定されたかどうかを判断する。条件部でコマンドラインの最初

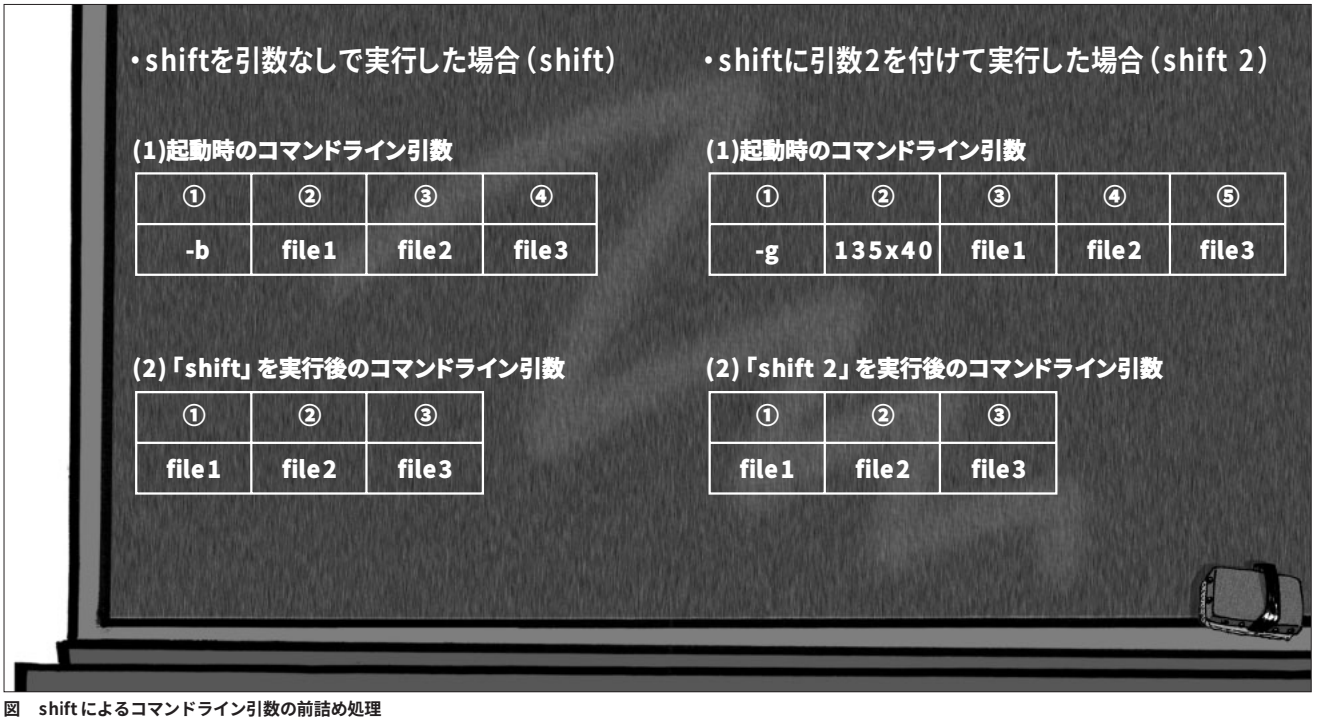


図　shiftによるコマンドライン引数の前詰め処理

の引数（$1で参照）を「-b」と比較し、一致する場合はcolorsの内容を通常とは逆の配色「-fg white -bg black」に変更する。さらに、組み込みコマンドのshiftを使って引数の内容を1つ前に詰め、先頭の「-b」を引数から削除している（**図**）。

このように、シェルスクリプトのオプション処理では、shiftを使って処理後のオプションを引数から削除することがポイントだ。これを忘れると、その後の処理で問題が起きる。たとえば、このスクリプトの場合は、「-b」がファイル名のひとつとして扱われてしまうのだ。

(2)パラメータを持つオプションに対応する

オプションの中には、数値や文字列などをパラメータとして持つタイプもある。たとえば、

```
$ wless -g 135x40 hoge
```

のように、-gオプションとパラメータ「135x40」が指定されたら、ktermのウィンドウをパラメータのサイズ（135桁×40行）で開くスクリプトは次のようになる。

```
#!/bin/sh
size=80x25
if [ "$1" = -g ]; then
  size="$2"
  shift 2
fi
kterm -geometry $size -fg black -bg white -e less
"$@" &
```

まず、シェル変数sizeに、ウィンドウサイズの初期値「80x25」を設定する。sizeは最後の行で実行するktermの-geometryオプションの直後で参照され、-gオプションが省略された場合はオリジナルのスクリプトと同じサイズで表示される。続くif制御構造では、-gオプションが指定されたかどうかを判断している。指定された場合は、パラメータが設定されている2番目の引数（$2で参照）の内容をシェル変数sizeに設定し、shiftを使って引数の前詰めを行う。-g自身とパラメータを一度に削除するため、「shift 2」として2つ前に詰める点に注意しよう（**図**）。

(3)複数のオプションに対応するには

(1)と(2)のスクリプトを組み合わせて、複数のオプションを指定できるようにしよう。たとえば、

```
$ wless -b -g 135x40 hoge
```

とすると、通常とは逆の配色で、なおかつ135桁×40行のウィンドウが開くわけだ。

任意の順番で並ぶ複数のオプションに対応するには、繰り返し処理を行うfor制御構造と、前回学んだ多分岐処理を行うcase制御構造を組み合わせる。具体的には、次のように書けばいい。

```
#!/bin/sh
colors='-fg black -bg white'
size=80x25
for opt in "$@"; do
  case "$opt" in
  -b ) colors='-fg white -bg black'
       shift ;;
  -g ) size="$2"
       shift 2 ;;
  esac
done
kterm -geometry $size $colors -e less "$@" &
```

シェル変数colorsとsizeに、オプションが指定されなかった場合の初期値を設定した後、for制御構造によってコマンドライン引数の内容が1つずつループ変数optに格納され、do～doneのcase制御構造を繰り返し実行する。case制御構造では、ループ変数optの内容を「-b」や「-g」と比較し、マッチする場合にはそれぞれのコマンド群を実行する。コマンド群の末尾には、次のパターンとの区切りとして「;;」が必要だ。

これで、-bオプションと-gオプションが同時に指定された場合にも、それぞれの処理がちゃんと行われる。したがって、最後の行でktermが起動される時点では、オプションやパラメータはすべて引数から削除され、colorsとsizeが設定済みの状態になっている。

要点をまとめよう。

- **オプションが1つだけの場合は、if制御構造で簡単に処理できる。**
- **処理済みのオプションはshiftで削除する。パラメータが必要なオプションの場合は、パラメータも忘れずに引**

数から削除すること。

・複数のオプションに対応する場合は、for制御構造とcase制御構造を組み合わせて使う。

●外部コマンドgetoptでオプションを解析する

これまでの説明のように、for制御構造やcase制御構造を使えば、シェルスクリプトでもコマンドラインオプションを処理できる。しかし、一般のコマンドのオプション処理はもう少し複雑だ。

たとえば、「-l -a」のように各オプションを独立した引数で指定するだけでなく、「-la」のように複数のオプションをまとめて指定しても受け付ける。パラメータを持つオプションについても同様で、「-w 60」のようにパラメータを独立した引数で指定するほか、「-w60」のように1つの引数にまとめてもいい。

こうした柔軟性のあるオプション処理をシェルスクリプトの機能だけを使って行うことは不可能ではないが、オプション関係の記述が長くなってしまい、実際に行いたい処理に到達するまでに作成者が疲れてしまうだろう。そこで、UNIX系OSにはこうした柔軟なオプション処理を代行してくれる便利な外部コマンドgetoptがある。Linuxのディストリビューションでも、/usr/binにgetoptが用意されているはずだ。

getoptのコンセプトは、「さまざまな形式で指定されるコマンドラインオプションを解析して、ある決まった形式に整形する」というものだ。たとえば、実際のオプションが「-la」や「-w60」などと指定されている場合でも、「-l -a」とか「-w 60」というように、オプションやそのパラメータがそれぞれ1つの引数に独立した形式に整形される。これなら、前節で紹介したfor制御構造とcase制御構造の組み合わせで簡単に処理できる。

getoptの書式は以下の通りだ。

```
getopt オプション文字列 コマンドライン引数
```

「オプション文字列」には、「−」の後に指定する文字を並べて書く（「−」は含まれない）。その際、パラメータを持つオプションは後ろに「：」（コロン）を付ける。たとえば、前節で作成したwlessの場合、パラメータなしの-bオプションと、パラメータを持つ-gオプションがあるので、オプション文字列は「bg:」となる。

一方、「コマンドライン引数」には、コマンドラインで指定するオプションやファイル名などを並べて書く。通常のシェルスクリプトでは、この部分には「"$@"」が指定されるはずだ。

getoptはスクリプトから実行されるケースがほとんどだが、コマンドラインで直接実行すると、どのような処理を行うのか理解しやすい。たとえば、

```
$ getopt bg: -bg135x40 hoge
 -b -g 135x40 -- hoge
```

といった具合だ。最初のコマンドライン引数には、2つのオプションとパラメータをまとめた「-bg135x40」を指定しているのに、出力ではそれらが1つずつ独立した引数に分かれていることがわかる。「-b -g135x40」や「-b -g 135x40」など、さまざまな形式でコマンドライン引数を指定しても、出力される内容はまったく同じだ。

もうひとつ注目すべき点は、オプションではない最初の引数「hoge」の前に「--」が挿入されていること。これは、「これ以後のコマンドライン引数はオプションではない」ことを示すためにUNIX系OSで使われる文字列で、特に「−」で始まるファイル名を指定する際に使われる。

getoptは、ユーザーがコマンドラインで「--」を指定しなかった場合でも、オプションの並びの終わりに「--」を自動的に挿入する。後ほどお見せするように、引数が「--」になった時点でオプション処理を終了するようにスクリプトを記述することで、通常のコマンドと同様に「−」で始まるファイル名を処理できるのだ。

ところで、getoptはコマンドライン引数の内容を書き換えるのではなく、解析結果を標準出力（通常は端末画面）に出力するだけだ。この出力をどのようにスクリプトで扱えばよいだろうか。

コマンドライン引数が格納されている位置パラメータ（1、2、…）は参照のみ可能な特殊な組み込み変数なので、内容を「$1」などで参照することはできるが、「1=hoge」として内容を書き換えることはできない。位置パラメータの内容を書き換えるには、シェルの組み込みコマンドであるsetを利用する。具体的には、新しい位置パラメータの内容を引数に列挙して、

```
set -- 新たな位置パラメータの内容
```

とすればいい。現在の位置パラメータの内容がすべて破棄され、新たな位置パラメータの内容で置き換わる。なお、set自身のオプションと区別できるように、最初の引数に

「--」を指定する点に注意されたい。

新たな位置パラメータとしてgetoptの出力を利用するには、シェルのコマンド置換機能を使う。getoptのコマンドライン全体を「`」（逆引用符）で囲んでsetの引数に記述すればいい。「`」で囲まれた部分がgetoptの出力で置換され、新たな位置パラメータとして設定される。

前節のスクリプトwlessに、getoptによるオプション処理を組み込むと以下のようになる。

```
#!/bin/sh
set -- `getopt bg: "$@"`
colors='-fg black -bg white'
size=80x25
for opt in "$@"; do
  case "$opt" in
  -b ) colors='-fg white -bg black'
       shift ;;
  -g ) size="$2"
       shift 2 ;;
  -- ) shift
       break ;;
  esac
done
kterm -geometry $size $colors -e less -- "$@" &
```

まず、「`」で囲まれたgetoptがサブシェルで実行され、オプションやパラメータが1つずつ独立した引数になるよう整形する。実行結果はコマンド置換によりsetの引数に使われ、新たな位置パラメータとして設定される。あとは、for制御構造とcase制御構造を組み合わせて、オプション処理を行えばいい。

case制御構造のパターンには、「-b」「-g」のほかに「--」が追加されている。この部分は、引数が「--」にマッチした時点でオプション処理を終了する典型的な記述だ。shiftによる引数の前詰めで「--」自身を引数から削除した後、組み込みコマンドのbreakを使ってfor制御構造による繰り返し処理から強制脱出している。これで、「--」よりあとの引数は、たとえ「－」で始まっていてもオプションとして処理されないことが保証される。このほか、最後の行の「"$@"」の前にも「--」を追加して、lessが「－」で始まるファイル名をオプションとして処理しないようにする必要もある。

このスクリプトは、前節のスクリプトにたった3行追加しただけだが、「-b -g 135x40」といった形式だけでなく、「-b -g135x40」や「-bg135x40」など、さまざまな形式のオプションをすべて正しく処理できるし、「－」で始まるファイル名にも対応するものになった。

この節の要点をまとめると、

- **getoptで、さまざまなオプションの指定に対応できる。**
- **引数には「オプション文字列」と「"$@"」を指定。**
- **オプション文字列に記述するオプションのうち、パラメータを持つものは後ろに「:」を付ける。**
- **スクリプトの最初のほうで、setの引数に「`」で囲んだgetoptを記述する。**
- **引数が「--」とマッチしたらオプション処理を強制脱出するようにする。**

ということになる。

●bashの組み込みコマンドgetoptsを使う

getoptの欠点は、オプション解析中の動作をスクリプトから制御できないことだ。たとえば、オプション文字列に含まれない不正なオプションを指定すると、「getopt: invalid option -- （文字）」というエラーメッセージが表示され、そのオプションはgetoptの出力には含まれない。このため、不正なオプションに対する処理は、for制御構造とcase制御構造を組み合わせたオプション処理では行えない。getoptを実行した直後に、getoptの終了ステータスを参照して別に処理する必要がある（具体的な処理内容は「今月のスクリプト」を参照）。

そこで、bashにはオプション処理用の組み込みコマンドgetops（こちらは末尾に「s」が付く）が用意されている。シェル自身に内蔵されているため実行速度が速く、機能も拡張されていて使いやすい。ただし、getoptsを利用すると、bash以外では動作しないスクリプトになってしまうことに注意しよう。getoptsの書式は以下のようになる。

```
getopts オプション文字列 変数名
```

「オプション文字列」の指定方法はgetoptと同じだ。ただし、オプション文字列の先頭に「:」を付けると、不正なオプションに対するエラーメッセージを抑制できる。なお、getoptsは現在のコマンドライン引数を先頭から順番に解析するので、getoptsの引数には指定する必要はない。その代わりに、解析結果を格納するシェル変数の「変数名」

を指定する。解析結果には「−」は含まれないので気をつけよう。また、不正なオプションの場合は実際の文字に関係なく「？」が格納される。

ところで、getoptsは繰り返し処理を行うwhile制御構造と組み合わせて使われる。while制御構造については本連載ではまだ取り上げていないので、ここで簡単に説明しよう。書式は以下のようになる。

```
while コマンド; do
  コマンドが正常終了した場合に実行する内容
done
```

while制御構造の条件部に指定する「コマンド」の扱いは、if制御構造とよく似ている。コマンドの終了ステータス（0＝正常終了、1〜255＝異常終了）に応じて、doとdoneに囲まれたコマンド（複数可）を実行するかどうかが決まるのだ。if制御構造との違いは、doneに達したら再度条件部のコマンドが実行される点にある。

このwhile制御構造とgetopts、case制御構造を組み合わせた典型的なオプション処理は次のようになる。

```
while getopts オプション文字列 opt; do
  case "$opt" in
  文字1 ) 「-文字1」オプションの処理 ;;
  文字2 ) 「-文字2」オプションの処理 ;;
        (中略)
  ¥?    ) 不正なオプションの処理 ;;
  esac
done
shift $((OPTIND-1))
```

まず、while制御構造の条件部でgetoptsを実行する。getoptsは、先頭の引数から順に解析を行い、シェル変数optにオプション文字（「−」を除く）を1つずつ格納する。オプションを処理している間はgetoptsが正常終了するため、do〜doneが繰り返し実行される。一方、オプションではない引数（「--」を含む）まで達すると、getoptsが異常終了して繰り返し処理から抜ける。

do〜doneの中では、case制御構造によるオプション処理を行う。シェル変数optには「−」が含まれないので、パターンはオプション1文字になることに注意しよう。getoptsは、どこまで引数を解析したか自分自身で把握しているため、shiftによる引数詰めを行う必要はない。パラメータを持つオプションでは、シェル変数OPTARG（$OPTARGで参照）にパラメータの内容が設定されている。

不正なオプションが指定された場合は、シェル変数optに「？」が設定されるので、通常のオプションと同様にcase制御構造で処理できる。ただし、パターン中の「？」はワイルドカード（任意の1文字）として扱われるため、「？」そのものは「\?」と書く必要がある。

さて、getoptsはshiftによる引数詰めを必要としないが、オプションやパラメータを残しておくとあとで困ったことになるので、繰り返し処理から抜けた時点でまとめて引数から削除する。シェル変数OPTINDに、オプションではない最初の引数の位置パラメータ（1以上の値）が設定されていることを利用して、OPTINDの値より1小さな数だけ引数を前に詰めればいい。bashの$((…))構文を使うと、「shift $((OPTIND-1))」と書ける。

getoptsとwhile制御構造、case制御構造を組み合わせると、wlessは次のように書ける。bashだけで動作するため、先頭は「#!/bin/bash」にしている。

```
#!/bin/bash
colors='-fg black -bg white'
size=80x25
while getopts :bg: opt; do
  case "$opt" in
  b ) colors='-fg white -bg black' ;;
  g ) size="$OPTARG" ;;
  ¥? ) echo "Usage: wless [-b] [-g size] files... "
  > /dev/stderr
      exit 1 ;;
  esac
done
shift $((OPTIND-1))
kterm -geometry $size $colors -e less -- "$@" &
```

以上、要点をまとめると、

- **getoptsとwhile制御構造、case制御構造の組み合わせで、　オプションの解析と処理をまとめて行える。**
- **while制御構造の条件部でgetoptsを実行する。**
- **引数には「オプション文字列」と「変数名」を指定。**
- **シェル変数には「-」を除いたオプションが格納される。**
- **オプションのパラメータは「$OPTARG」で参照できる。**
- **不正なオプションはパターン「\?」で一括処理する。**

・while制御構造による繰り返しを抜けてから、「shift $((OPTIND-1))」でまとめて引数詰めを行う。

ということになる。

今月のスクリプト

後半は、与えられたテーマを実現するスクリプトの作成手順を説明する。今月は、

・指定した画像ファイル（複数可）を1つ（あるいは複数）のウィンドウに表示する「disp」

を作成する。

画像の表示には、数多くの画像形式に対応している画像ビューア「ImageMagick」のコマンドdisplayを利用する。たいていのLinuxのディストリビューションにはインストールされているはずだ。

複数の画像をひとつのウィンドウにまとめて表示するか、画像の数だけウィンドウを開くかは、シェルスクリプトのコマンドラインオプションで指定する。ウィンドウサイズの上限もオプションで指定できるようにしよう。

スクリプトの構成は、大まかにいって、

・コマンドラインオプションの処理
・displayを利用した画像の表示

という2つの部分に分かれる。以下では、それぞれの処理について詳しく説明しよう。

●コマンドラインオプションの処理を行う

スクリプトdispでは、以下の2つのコマンドラインオプションを処理することにする。

-i 画像の数だけ個別にウィンドウを開く
-g ウィンドウサイズの上限をパラメータで指定する

上記以外のオプションが指定された場合は、使い方（Usage）を表示して、終了ステータス1でスクリプトの実行を終了させる。また、「--」から後ろはファイル名と見なしてオプション処理を終了する。

今回学んだ外部コマンドのgetoptを利用してオプションを解析し、整形した結果を得るには、

```
getopt ig: "$@"
```

と書けばよさそうだ。-gオプションはパラメータを持つので、後ろに「:」を付ける必要がある。

次に考えるのは、不正なオプションが指定された場合の処理だ。前半で説明したwlessのように、setの引数として直接getoptのコマンド置換を指定すると、不正なオプションに対する処理を行えない。getoptとsetの実行は分けたほうがいいだろう。

オプション文字列に含まれない不正なオプションが指定されると、getoptは異常終了する（終了ステータス1）。このため、if制御構造による分岐を利用可能だ。

```
if opts=`getopt ig: "$@" 2> /dev/null`; then
  set -- $opts
else
   echo "Usage: disp [-i] [-g size] files..." >
/dev/stderr
  exit 1
fi
```

if制御構造の条件部でgetoptを実行し、正常終了した場合だけ位置パラメータの更新を行う。getoptの実行結果はコマンド置換によりシェル変数optsに格納され、その内容（$optで参照）をsetの引数に指定する。なお、コマンド置換部分の末尾に書かれた「2> /dev/null」は、標準エラー出力をリダイレクトして、エラーメッセージを端末画面に表示しないようにする仕掛けだ。

一方、getoptが異常終了した場合は、echoで使い方（Usage）の説明を標準エラー出力（通常は端末画面）に出力し、「exit 1」により終了ステータス1（異常終了）でこのスクリプトを終了する。

続いては、for制御構造とcase制御構造を組み合わせてオプション処理を行う。

```
for opt in "$@"; do
  case "$opt" in
  -i ) ind=yes
       shift ;;
  -g ) size="$2"
       shift 2 ;;
  -- ) shift
       break ;;
```

```
  esac
done
```

この構造はwlessの場合とほぼ同じなので、細かい説明は省略する。-iオプションの処理ではシェル変数ind、-gオプションの処理ではシェル変数sizeに、あとの処理で利用する内容をそれぞれ設定している。

●displayを実行して画像を表示する

displayのコマンドラインで複数の画像ファイルを指定すると、最初の画像だけがウィンドウに表示される。残りの画像は、キー操作（スペース／BSキー）などにより順次切り替えて表示できる。

今回作成するスクリプトdispでも、-iオプションを指定しない場合はこの動作をそのまま受け継ぐことにしよう。一方、-iオプションを指定した場合は、それぞれの画像ファイルに対してdisplayを起動して、複数の画像が個別のウィンドウに表示されるようにする。

-iオプションの有無はシェル変数indの内容で判断できる。よって、if制御構造とtest（あるいは［）を組み合わせて、次のように書けばいい。

```
if [ -z "$ind" ]; then
  display -geometry "$size" "$@" &
else
  for f in "$@"; do
    display -geometry "$size" "$f" &
  done
fi
```

条件式「-z "$ind"」は、シェル変数indが未定義か、内容が空文字列の場合に真になる。つまり、-iオプションが指定されていない場合にだけ、「display -geometry "$size" "$@" &」が実行される。なお、displayは自分のウィンドウを持つXアプリなので、末尾に「&」を付けてバックグラウンド実行している。

何回か説明したように、「"$@"」はすべての引数をひとつずつ「"」で囲って並べたものに展開される。この時点ではオプションやパラメータはすべて削除されているため、画像ファイル名だけが並ぶわけだ。

シェル変数sizeには、-gオプションの引数で指定したウィンドウサイズ（「640x480」など）が格納されている。これをdisplayの-geometryオプションのパラメータとして指定することで、ウィンドウサイズの上限値を設定できる。一方、-gオプションを指定しなかった場合はパラメータが「""」になり、元のサイズで画像が表示される。

-gオプションのパラメータで指定するのはあくまでもウィンドウサイズの上限値で、実際に表示される画像は元の画像の縦横比を保っている。ただし、「640x480！」のように末尾に「！」を付けることで強制的に変形させることも可能だ。なお、これらはdisplay自体に備わっている機能で、スクリプトによるものではない。

一方、-iオプションを指定した場合の処理はもう少し複雑だ。「"$@"」を直接displayの引数に指定する代わりに、for制御構造のリスト部分に指定して、画像ファイルに対して1つずつdisplayをバックグラウンド実行する。これで、各画像ファイルが独立したウィンドウに表示されるわけだ。画像ファイルを多数指定した場合は、メモリの消費が激しくなるので気をつけよう。

以上の処理を統合して完成させたスクリプトdispが**リスト1**だ。腕に自信のあるユーザーは、getoptの代わりにbashの組み込みコマンドgetoptsを利用するように書き換えたり、複数の画像を一定時間ごとに自動的に切り替えて表示する「スライドショー」を指示するオプションを新たに追加してみるとよいだろう。

▼リスト1　スクリプトdisp

```
 1: #!/bin/sh
 2: if opts=`getopt ig: "$@" 2> /dev/null`; then
 3:   set -- $opts
 4: else
 5:   echo "Usage: disp [-i] [-g size] files..." >
    /dev/stderr
 6:   exit 1
 7: fi
 8: for opt in "$@"; do
 9:   case "$opt" in
10:   -i ) ind=yes
11:        shift ;;
12:   -g ) size="$2"
13:        shift 2 ;;
14:   -- ) shift
15:        break ;;
16:   esac
17: done
18: if [ -z "$ind" ]; then
19:   display -geometry "$size" "$@" &
20: else
21:   for f in "$@"; do
22:     display -geometry "$size" "$f" &
23:   done
24: fi
```

Linux日記

第21回　メール配送（8）

sendmailがメールを送受信するためには、メールアドレスを処理しなければなりません。sendmail.cfに書かれたルールセットの中から、これらの処理を行う部分について解説します。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

みなさんはなぜLinuxに興味を持っているのだろうか？　Windowsに嫌気がさしたから？　毛色の違うシステムに触ってみたいから？　仕事や勉強の都合だろうか？　流行してるみたいだから？　実のところは、安いからかもしれない。

もう少し根源的なところで、みなさんはなぜコンピュータをいじるようになったのだろうか？　何となく興味を持ったから？　できないと就職できないから？　インターネットにアクセスしてみたかったから？　ゲームをしたかったから？

まぁ、こんなことは人それぞれだ。

筆者は、もう40歳を過ぎている。コンピュータとの付き合いは、10代前半からだから、もう25年以上だ。マイクロコンピュータが登場して間もない頃から本を読んだり実際にいじっていることになる。読者の多くが生まれたか生まれてないかといった時代のワンボードマイコンのシミュレータを作ったりするのも、歳のせいである。

筆者がコンピュータに興味を持ったのは、最初は技術的な関心からだった。その後、マイクロコンピュータが登場し、それをいじってみた。ワンボードマイコンの時代だ。そして、いわゆるコンピュータらしいことがあまりできない、つまり、FORTRANなどの高級言語を使い、紙やテレビに結果を表示するには、あまりに非力である（あるいはそれを実現するためにとてつもなくお金がかかる）ということを認識した。

筆者はその頃、電子楽器にも興味があって、シンセサイザという楽器をいじっていた（キーボード楽器として弾けたわけではない）。で、マイクロコンピュータには高度な計算は無理でも、シーケンサ（自動演奏装置）くらいは作れるのではなかろうかと思ったのである。しかしその後、マイクロコンピュータの可能性を深く知り、同じようなことをやっている仲間と出会い、そして周辺機器を安価に仕入れるツテを知ったことによって、いつしかシンセサイザのことなど忘れてしまい、コンピュータどっぷりの生活になった。

そんなこんなで25年経ち、ひさびさに電子楽器に触れることになった。子供が音楽教室に通っていたので、中古のエレクトーンを買ったのである。同じ頃、実家から当時のシンセサイザを回収してきた（「持っていかないと捨てる」と実家から脅されたのだ）。エレクトーンに触発されたこともあり、久々

Column

1970年代の少年たちの野望

筆者は、マイクロコンピュータを使えばシーケンサを作れるかもしれないと思ったのであるが、当時からの友人のひとりは、また別の理由でマイクロコンピュータに手を出したのだという。曰く、「自作ロケットの姿勢制御系に使えるんじゃないかと思った。」

もちろん彼も航空宇宙分野に進むことはなく、今は半導体設計関連の仕事をしている。すなわち、Rocket Boysになり損なったわけだ。

にいじると、アナログシンセサイザの安直で電子的な音色が妙に懐かしかった（70年代後半の音楽シーンに欠かせない音色だ）。で、懐かしさついでに、インターネットオークションで同時期のシンセサイザをもう1台購入してしまった。

1万円ちょい出せば、非常に高度なサウンドカードが入手できる今の時代に、当時20万円以上したアナログシンセサイザの機能と音は、あまりに新鮮かつ回顧的で、しばらく2台で遊んでしまった。子供もおもしろがって弾いている。山ほどあるボリュームやスイッチが面白いらしい。

■アナログシンセサイザ

現在はさまざまな音源方式があり、小さなLSIでリアルな音色が生成されているが、25年前のアプローチはもっと原始的なものだった。まず、発振周波数を変化させられるオシレータがある。何種類かの波形を選択することができて、これで基本的な音色の要素（周波数成分）が決まる。この発振信号を、フィルタに通す。フィルタのパラメータをいくつかいじることにより、音色が変わる。

これらの要素に対して、副次的なパラメータを与えることが可能だ。キーの操作に対して、音の立ち上がり時間、持続時間、余韻時間などを設定できる。この信号をアンプに加えれば、音量変化をいろいろいじることができる。持続的に鳴るように設定すれば、オルガンのようになり、減衰するようにすれば打楽器やピアノのようになる。立ち上がり時間をちょっと長くすれば、管楽器や弦楽器のようになる。また、この信号をフィルタに加えれば、時間経過とともに音色が変化する。また、音源であるオシレータとは別に低周波のオシレータがあり、これを音源オシレータやアンプに適用すると、トレモロやビブラート効果が得られる。

こういった要素とは別に、ノイズジェネレータや特殊なモジュレータなどがあり、さまざまな特殊効果をかけることができる。

これがアナログシンセサイザの基本的な仕組みだ（**図1**）。これらの制御が、すべてアナログ信号処理によって行われていたのである。シンセサイザは、数十個のボリューム、スイッチを備えた楽器だったのだ。筆者が持っているのは、2台ともステージでの使用を意識した固定配線のものだが、スタジオ録音用などには、これらをパッチコードで自由に配線して音を作るタイプのものもあった（当時は高くて手が出なかった）。

アナログシンセサイザは、当時としては画期的にいろいろな音を出せる楽器だったのである。難点は、基本的に1ボイスだったことだ。つまり、和音が出せないのである（筆者が70年代に購入した機械は、当時あまりなかった2ボイスモデルだ）。ほかの楽器とともに使って、効果的な音を出すか、マルチトラックテープレコーダを使って多重録音するという使い方が多かった（日本では、富田勲のアルバムが有名だった）。

今となっては、ソフトウェアエミュレーションでもできてしまうようなチープな楽器であるが、昔から重厚長大マニアであった筆者にとっては、こんなにおもしろい楽器はなかった。キー

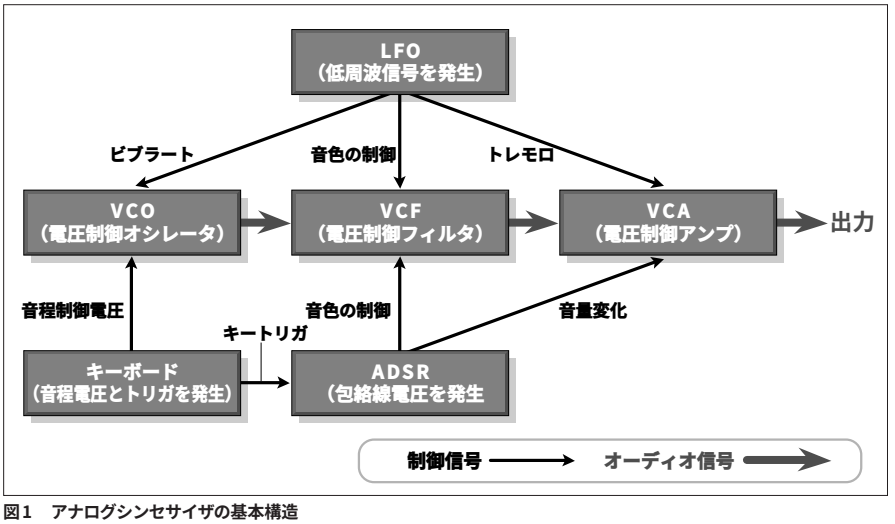


図1　アナログシンセサイザの基本構造

Column

昔の人はえらかった

これだけのつまみやスイッチがある機械である。使い方は恐ろしく複雑だ。基本原理や用語を知っていればともかく、シンセサイザの場合、何も知らないユーザーは音も出せないという状況がままあったらしい。それにも関らず、当時は、これだけの機械に20ページ程度のマニュアルしか付いていなかった（初期のコンピュータも似たようなものだったが）。

当時のユーザーは今のユーザーより我慢強かった。自分が操作できないのを、人のせいにしなかったのである。うまく動かなければ、自分でいろいろ調べ、実験し、詳しい友人に聞き、知識を自分のものにしていった。自分の頭も使わずに、こんな難しい機械を作るほうが悪いなんて開き直るようなユーザーはいなかったような気がする。

の数と同じくらいつまみがあるのだから（筆者は昔から、スイッチやつまみが多い機械が好きだ）。和音が出せないため、キーボードが弾けない人差し指奏法でもどうにかなったということもある（所有者である筆者より7歳の子供のほうが上手に弾いているのを複雑な心境で見ている）。

こういった25年も前のアナログ電子機器の中を見ると、結構感動もんである。今のデジタル機器は、どんなに性能がすごいといっても、すごいのは半導体チップの中身であり、人間の目で見ると、数枚の板に四角い部品が並んでいるだけだ。当時の機械はそんなつまらない光景ではない。値段相応の複雑さが目に見えるのである。基板上に並んだ多数のディスクリート部品、基板とスイッチ類をつなぐ電線の束。あぁ、人間が手間暇かけて作った機械なんだぁと実感できる。

現在の半導体チップの内部写真も美しいし、プリント基板もきれいに作られている。しかし、それらはCADで設計された幾何学的美しさである。手作りの見事さとは違う。いま考えると、これだけの部品が並び、無数の配線がなされている機器が20万円ちょいで作られていたというのは、逆にすごいことのように思えるのだ。

惜しむらくは、25年の時間の経過である。どちらのシンセサイザも、経年劣化が見受けられる。ボリュームのガリ、スイッチの接触不良などに起因する動作不良があるのだ。暇になったら直そうと思っているが、いつやれることやら。

さて、シンセサイザで音を出すより難しいsendmailの話の続きだ。前回は、ルールセット0と配信エージェントの関連付けについて説明した。今回はメールアドレスを処理するルールセットについて説明しよう。

■メールアドレス

アドレスの処理というのは、さまざまな形式で表記されたアドレスを解析し、実際のメール配信情報である＜ユーザー名＞@＜ドメイン名＞という情報を抽出する処理である。前回解説した配信エージェントの選択では、メールアドレス中のドメイン名に基づいて、メールをどのように配信処理するかを判定していた。ローカル配信を行う際には、メールアドレスのユーザー名に基づいてメールボックスへのスプール処理を行う。このような処理を適切に行うために、メールアドレスに対する前処理などが必要になる。

電子メールに使われている発信者、宛て先のメールアドレスにはさまざまな形式がある。

日常目にするのは、**リスト1**に示したようなものだろう。最初の「masanori」というように、@以降のドメイン名が省略されている形式は、ローカルホスト上、あるいはローカルサイト内の配信のための表記なので、実際には、適当なデフォルトのドメイン名（ここではascii.co.jp）が付加されることになる。実質的には2番目の形式と同じだ。

メールアドレスに加えて、本名（あるいはハンドルなど）を付加した形式も有効である。通常、これらはメールのFrom：行に現れる形式であるが、To：やCc：にこの形式で表記することもできる。宛て先を手でタイプする場合は、このような形式を使うことはほとんどないだろうが、受信メールに返信する場合などは、元メールのFrom：をそのままTo：に転記するので、宛て先アドレスがこの形式で指定されることになる。

これらの表記をまとめると、次のよ

▼リスト1　さまざまなメールアドレスの例

```
masanori
masanori@ascii.co.jp
Masanori Sakaki <masanori@ascii.co.jp>
榊 正憲 <masanori@ascii.co.jp>
masanori@ascii.co.jp （Masanori Sakaki）
masanori@ascii.co.jp （榊 正憲）
```

Column

Dr. MoogとMOOGシンセサイザ

アナログシンセサイザの基礎を築いたのは、Dr. Moogである。オシレータやフィルタを組み合わせていろいろな音色を作るということは、以前から行われていたが、Dr. Moogは、これらを直流電圧で統括的に制御するという仕組みを考え出した（らしい）。たとえば、オシレータが発振する周波数、フィルタがカットオフする周波数、アンプのゲインなどを、すべて電圧制御化した。なんでもないことのように思えるかもしれないが、この仕組みが非常に高い自由度を産み出したのである。

Dr. MoogはMOOGというブランドでいくつかのシンセサイザを製品化した。多数のモジュールをパッチコードで配線するモジュール式のシステムや、ある程度固定配線し、ステージでも使えるようなシステムである。ステージ用として代表的なものが、mini MOOGと呼ばれる製品だ。当時は60万円以上し、とても手が出なかった。今でも中古品が30万円くらいから入手できる。

また、mini MOOGの音をサンプリングした音源というのもあるらしい。合成すればよさそうに思うのだが、どうも違うらしい。

うになる。

メールアドレス
名前 <メールアドレス>
メールアドレス（名前）

もちろん、実際の配信に必要なのはメールアドレスだけだ。

これらのメールアドレスは、ユーザーがメールを作成するMUAか、そのメールを最初に発信、中継するMTAによって生成される。WindowsやMacintoshなど、MTA機能を持たないコンピュータ上で動作するMUAで生成する場合は、MUAの初期設定として、メールアドレス、本名やハンドルなどを指定する。MUAはこれらの情報に基づいて、上記の形式のFrom：行を生成する。sendmailが動作するホスト上で実行されるMUAを使う場合は、つまりMUA内からsendmailを実行する場合は、MUA側ではFrom：行を生成せず、sendmailに生成させることもできる。sendmailプロセスを実行したユーザーの情報、つまりプロセスのユーザーIDからpasswdファイルのユーザー名、本名（GECOSフィールド）などの情報を取得し、メールアドレスとユーザー名から構成されるFrom：行を作成するのである。

sendmail.cf中で記述するヘッダの定義についてはあとで解説するが、From：行については、以下のような行で定義されている。

```
H?F?From: $q
```

■ルールセット3

sendmailでメールを処理する場合、配信に使用される各種メールアドレスは、まず最初にルールセット3で前処理が行われる。ルールセット3の前処理

▼リスト2　あるsendmail.cfのルールセット3、ルールセット96

```
# handle null input and list syntax (translate to <@> special case)
R$@                     $@<@>

# strip group: syntax (not inside angle brackets!) and trailing semicolon
R$*                     $:$1<@>                 mark addresses
R$*<$*>$*<@>            $:$1<$2>$3              unmark <addr>
R:include:$*<@>         $::include:$1           unmark :include:
R$*[$*:$*]<@>           $:$1[$2:$3]             unmark IPv6 addrs
R$*::$*<@>              $:$1::$2                unmark host::addr
#R$*:;<@>               $:$1:;                  unmark list:;
R$*:;<@>                $@                      list:; special case
R$*:<@>                 $:$1:                   unmark something: (bad)
R$*:$*;<@>              $:$2                    strip group:; if marked
R@$*:$*<@>              $:@$1:$2                unmark @route:addr
R$*:$*<@>               $:$2                    strip group: if marked
R$*;<@>                 $:$1                    strip semi if marked
R$*<@>                  $:$1                    unmark

# basic textual canonicalization -- note RFC733 heuristic here
R$*<$*>$*<$*>$*         $2$3<$4>$5              strip multiple <> <>
R$*<$*<$*<$+>$*>$*>$*   $4                      3-level <> nesting
R$*<$*<$+>$*>$*         $3                      2-level <> nesting
R$*<>$*                 $@<@>                   MAIL FROM: <> case
R$*<$+>$*               $2                      basic RFC821/822 parsing
# make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@$+,@$+                @$1:@$2                 change all "," to ":"

# localize and dispose of route-based addresses
R@$+:$+                 $@$>96 <@$1>:$2         handle <route-addr>

# find focus for list syntax
R$+:$*;@$+              $@$1:$2;<@$3>           list syntax
R$+:$*;                 $@$1:$2;<@>             list syntax

# find focus for @ syntax addresses
R$+@$+                  $:$1<@$2>               focus on domain
R$+<$+@$+>              $1$2<@$3>               move gaze right
R$+<@$+>                $@$>96 $1<@$2>          already canonical

# convert old-style addresses to a domain-based address
R$-!$+                  $@$>96 $2<@$1.UUCP>     resolve uucp names
R$+.$-!$+               $@$>96 $3<@$1.$2>       domain uucps
R$+.!$+                 $@$>96 $2<@$1>          extended domain uucps
R$*!$*                  $@$>96 $2<@$1.UUCP>     uucp subdomains

# if we have % signs, take the rightmost one
R$+%$+                  $1@$2                   First make them all @s.
R$+@$+@$+               $1%$2@$3                Undo all but the last.
R$+@$+                  $@$>96 $1<@$2>          Insert < > and finish

# trap invalid address
R@$*                    $@<@$1>

###  Ruleset 96 -- bottom half of ruleset 3  ###

#  At this point, everything should be in a "local_part<@domain>extra" format.
```

```
S96

# strip trailing dot off
R$*<@$+.>$*             $1<@$2>$3

# resolve numeric addresses to name if possible
R$*<@[$+]>$*            $:$1<@$[[$2]$]>$3
R$*<@$+.>$*             $1<@$2>$3                       strip trailing dot off

# header/envelope alias rewriting
#R$*<@$*ALIAS_NAME>$*   $:$1<@$2REAL_NAME>$3
#R$*<@$+ALIAS_NAME>$*   $:$1<@REAL_NAME>$3
#R$*<@ALIAS_NAME>$*     $:$1<@REAL_NAME>$2

R$*<@$=w.UUCP>$*        $:$1<@$j>$3
R$*<@$=w>$*             $:$1<@$j>$3

# exit immidiately, if fake domains or numeric address spec
R$*<@$*.$=X>$*          $@$1<@$2.$3>$4

# acceptable domains are OK
R$*<@$=Z>$*             $@$1<@$2>$3
# lowers are OK
R$*<@$*.$m>$*           $@$1<@$2.$m>$3
# official name is OK
R$*<@$=w>$*             $@$1<@$2>$3
# canonicalize with nameserver lookup
R$*<@$*>$*              $:$1<@$[$2$]>$3
# strip trailing dot off possibly canonical name
R$*<@$+.>$*             $1<@$2>$3
```

```
$ sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3 masanori@ascii.co.jp
rewrite: ruleset   3   input: masanori @ ascii . co . jp
rewrite: ruleset  96   input: masanori < @ ascii . co . jp >
rewrite: ruleset  96 returns: masanori < @ ascii . co . jp >
rewrite: ruleset   3 returns: masanori < @ ascii . co . jp >
> 3 masanori@ascii.co.jp (SAKAKI Masanori)
rewrite: ruleset   3   input: masanori @ ascii . co . jp
rewrite: ruleset  96   input: masanori < @ ascii . co . jp >
rewrite: ruleset  96 returns: masanori < @ ascii . co . jp >
rewrite: ruleset   3 returns: masanori < @ ascii . co . jp >
> 3 SAKAKI Masanori <masanori@ascii.co.jp>
rewrite: ruleset   3   input: SAKAKI Masanori < masanori @ ascii . co . jp >
rewrite: ruleset  96   input: masanori < @ ascii . co . jp >
rewrite: ruleset  96 returns: masanori < @ ascii . co . jp >
rewrite: ruleset   3 returns: masanori < @ ascii . co . jp >
>
```

画面1　sendmailのアドレステストモードでルールセット3を試す

は、ほかのルールで処理するために必要な共通処理を行うものであり、実際にどのような書き換えを行うかは、ほかのルールセットの処理内容に依存する。しかし、ユーザーの本名などを含んだ前述の形式のメールアドレスから、本来のメールアドレス部分を抽出するという処理は共通している。

あるsendmail.cfのルールセット3の定義を見てみよう。このsendmail.cfは、CFというsendmail.cf作成ツールで生成したものである（CFについては夏頃には触れられるかもしれない）。このルールセット3は内部でルールセット96も呼び出しているので、それもまとめて示しておく（**リスト2**）。

いかにもsendmail.cfというリストである。これは、前述した形式以外の処理も行っているので、かなり長くなっている（締め切りすれすれなため、短くする暇がなかった）。1行ずつ読んでいってもいいのだが、実際にsendmailで処理させたほうが簡単だ（**画面1**）。

アドレスの表記形式に関らず、名前部分が取り除かれていることがわかる（@以降のドメイン部が<>で括られているのは、あとの処理の都合である）。

■ルールセット4

配信エージェントの選択は、宛て先アドレスについてまずルールセット3を適用し、その後、ルールセット0で実際の選択処理を行う。これについては前回説明した。この処理とは別に、宛て先アドレス、発信アドレスについても処理を行う。宛て先アドレスについては、3→2→R＝→4というルールセットが、発信アドレスについては、3→1→S＝→4というルールセットが適用される。S＝とR＝は、前回説明したように、配信エージェント定義のMコマンド中で定義されるルールセットである（**図2**）。

ルールセット4は、後処理のためのものである。前述したルールセットでは、@以降のドメイン部を<>で括ったが、このような形式は、内部処理には便利に使えても、外に出すわけにはいかな

い。ルールセット4は、このような内部的な書き換えを元に戻すのに使える。実際に試してみよう（**画面2**）。アドレスをルールセット3で前処理し、その後、ルールセット4で後処理してみる。ルールセット3で付加された<>が、ルールセット4によって除去されているのがわかる。

このような形にすることにより、共通する前処理と後処理を独立したルールセットに切り出し（ルールセット3と4）、宛て先アドレス、発信アドレスに固有の処理を別々のルールセットで処理することができる。また、配信エージェントの定義のところで指定するS＝とR＝は、ヘッダ中のアドレスとエンベロープのアドレスについて別々のルールセットを定義できるので、さらに細かく処理を分けることができる。

たとえば、宛て先アドレスであれば、すべてのアドレスに共通する前処理と後処理はルールセット3と4で行う。そして、宛て先アドレスについて、エンベロープ情報でもヘッダでも共通する処理はルールセット3で、宛て先アドレスとエンベロープ情報で異なる処理についてはR＝で指定したルールセットで行うことができる。

これらのルールセットによる処理は、必須ではない。たとえば、S＝、R＝によるルールセットが必要ないのであれば、ルールセット番号として0を指定すればよい。また、必須のルールセット番号であっても、処理を行う必要がない場合は、ルールセットに何も記述しなければよい。

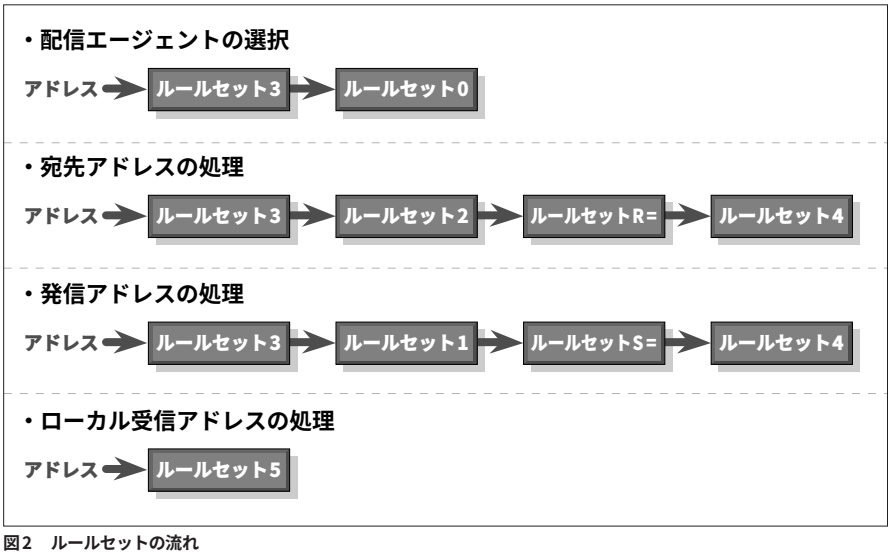


図2　ルールセットの流れ

```
$ sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,4 masanori@ascii.co.jp
rewrite: ruleset   3   input: masanori @ ascii . co . jp
rewrite: ruleset  96   input: masanori < @ ascii . co . jp >
rewrite: ruleset  96 returns: masanori < @ ascii . co . jp >
rewrite: ruleset   3 returns: masanori < @ ascii . co . jp >
rewrite: ruleset   4   input: masanori < @ ascii . co . jp >
rewrite: ruleset   4 returns: masanori @ ascii . co . jp
>
```

画面2　ルールセット3、4を試す

■ルールセット5

ルールセット5は、ローカル配信を行う際に、宛て先アドレスについて呼び出されるルールセットである。

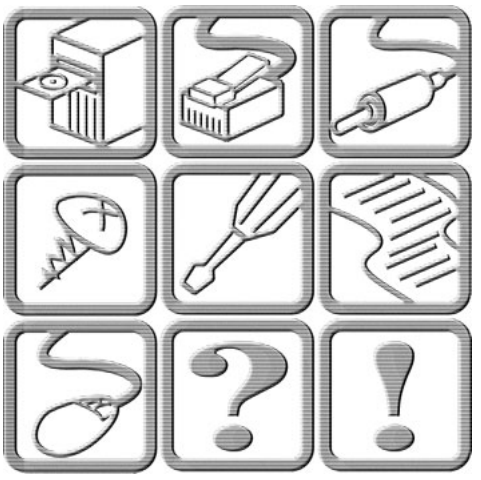
ルールセット0でローカル配信を行うと判定されたメールは、ルールセット3やルールセット4で処理されたあと、エイリアスの評価を受ける。エイリアスについてはもう少ししたら解説するが、基本的には、あるアドレスについて別名を定義したり、特定アドレスを複数ユーザーに配信したり、受信メールをプログラムで処理したりするための機能である。エイリアスで何らかの再送信処理、つまり宛て先アドレスの書き換えが指定されていた場合は、新しいアドレスでメール配信処理をすべてやり直すことになるが、何も処理されなかった場合は、指定されたローカル配信処理が行われることになる。通常であればbinMailを呼び出して、各ユーザーのスプールファイルに追加書き込みを行うことになるだろう。

ルールセット5は、エイリアス評価を行い、何も書き換えが行われなかったメールについて、つまり実際にローカル配信を行うことが確定した時に、実際にローカル配信エージェントを呼び出す前に呼び出されるルールセットである。単にbinMailでローカル配信を行うだけであれば、特殊な処理は何もないので、ルールセット5を記述する必要はないが、ローカル配信の時だけに何らかの特殊な処理を行う必要がある場合は、ルールセット5で記述できる。

■さて次回は

非常に簡単にではあるが、ルールセットの仕組みについて説明した。次回は、sendmail.cf中のヘッダの定義について解説する予定だ。

Try & [trái] Try



我が家のネットワークの調整（2）

文：政久忠由
Text : Tadayoshi Masahisa

『Darwin Kernel』

いろいろと言われているMac OS Xのカーネルには、こんな名前が付けられていた。

Mac OS Xが欲しくなる

Linuxと異なり、Mac OS Xはマイクロカーネルベースなので、Darwinというのは、そのマイクロカーネルの名前というよりは、そのカーネルモジュールとそれを取り巻くサーサービスモジュールを含めてのMac OS Xのベースコンポーネント全体を表す呼び名である。もちろんネットワーク部やファイルシステムもサービスということになる。

僕は単に店頭で15分程度デモ機と戯れただけで、そもそもMacユーザーではないし、今からMacユーザーになる気もさらさらないのだけれど、実際のところ、そんなに悪くないじゃん、という印象である。

実は個人的には、Apple（いちおうモトローラとIBMも）の作るマシンは買いたくないけど、その昔開発されていたPowerPC 620に淡い期待を抱いていたひとりで、PowerPC G4やその先のG5、G6がアセンブリパーツとして店頭に並んでくれたらなぁ、といまだに願っていたりする。

まぁそんなことは置いておくとして、当のMac OS Xは、一般にはあまり評判がよろしくないようだ。

Mac OS Xは、NeXTStepとMacの融合というか、Mach 3.0ベースのマイクロカーネルアーキテクチャを採用し、BSD4.4系のベース環境サービスシステムを実装して、その上にCocoaとかCarbonといったアプリケーション環境フレームワーク、そしてユーザーインターフェイスのAquaといった構成になっている。

この構成は、従来から言われていた次世代OSの姿のひとつではあるが、今さらという意見もないわけではない。

マイクロカーネルアーキテクチャは、システムの堅牢性に大いに寄与するが、逆にメモリやCPUのリソースの無駄も多くなる。すなわち、各コンポーネントモジュールは個々の機能別に小さくなるものの、全体の動作としては重くなる傾向にある。まぁこれは予想されるトレードオフだから仕方がない。またデバイスドライバレイヤがきちんと整備され比較的安定したものが容易に作成できるようになったものの、すべては新たに作り直す必要があるため、サポートハードウェアが当面制限される。古臭いハードウェアを持っている人にはハッキリ言ってその恩恵がないというのは、商用のOSのバージョンアップではよくあることだ。

しかし、メモリやCPUのリソースが足りない時代でもないし、従来のMac OSからすると、メモリ管理やスレッドのスケジューリング、またネットワーク機能などは格段に向上するのだから、悪い話ではない。それに、忘れ去られようとしていたBSD系UNIXのNeXTStepとその技術がやっと広くに利用されようとしているのだ。Mac OS Xのネイティブアプリケーション環境であるCocoaは、実はNeXTStepで培われてきたものだ。Mac OS Xは、従来のMac

OS用のアプリケーションが動作するための環境サブシステムとMacのユーザーインターフェイスをベースにしたフレームワークを実装したNeXTStepそのものといってもいい。まぁAppleが手持ちのカードから切るとなるとこうなっちゃうわけだ。でも僕はNeXTStep自体嫌いじゃない。NeXTStepは当時のハードウェア環境が貧弱だったためにユーザーに受け入れられなかったが、今なら大丈夫だ。

さてMac OS Xのマイクロカーネルアーキテクチャやちょっと古いCランタイムライブラリ、あと迎合主義的なハードウェアリソースを不毛に浪費するユーザーインターフェイスフレームワークのことを悪く言う人もいるけれども、僕は悪くないと思う。

カーネルやベースコンポーネントの実装形態は思想の問題で結論が出ないので、ここでは述べないけど、実は個人的にはマイクロカーネルのほうが好きだったりする。まぁ一部のコアなユーザーは別として、多くの一般ユーザーはベースコンポーネントがどうなっているかなんて、普通重要視しない。それに理論構造上の設計が単純にそのまま実装されているわけではないし、それぞれの構造には一長一短があるから、つまるところは好みの問題なんだよね。結果として、そのことによる差がそれほど生じていないんだなぁ、これが。

OSは、安定して、無駄が少なく、効率よくハードウェアを利用できる、すなわち利用者がハードウェアの性能以外で不必要なストレスを感じないよう動いてくれることが重要で、それは実績という形で評価され表される。ただ、カーネルに注視すれば、誰もがクソだと認めるWindows 9x系が多数を占めている現状は、その程度のものでも一般ユーザーは許容しているということだし、OSの信頼よりもアプリケーション環境の充実が重要だという表れでもある。

ユーザーインターフェイスは、そのアプリケーション環境の重要なファクタとなるものだが、現時点では、Linuxで利用できるまやかしのような陳腐なGUIと比べたら、Windowsにしろ、Macにしろ、格段に洗練されていることは確かだ。ちょっと操作しただけだけど、Macに対して無知な僕でもストレスを感じなかったのだからAquaはいい感じに仕上がっているのだと思う。

Linuxにはさまざまなguiが実装されているけれども、僕はいまだに使いたいと思うものにめぐり合っていない。これこそ相性の問題なのだが、どれもこれも使ってみるとイライラがつのってくるのだ。編集の出来の悪い映画を見せられたような感じでもあるし、キーボードのタッチとも通じるところもある。どれも似たようなものなのだけれども、ピッチとストロークとフィーリングが合わないのだ。

僕は、Mac OS Xは何だかんだ言ってもまずまずの出来だと思うし、我が家でも使いたい欲しいなぁとしばらく想いをめぐらせてはみたものの、如何せんそれを動かすハードウェアを持っていないし、それ用のアプリケーションを買い揃えることなんて毛頭考えられなかった。いい加減Appleもハードウェアとソフトウェアを切り離してくれないかなぁ。x86版のMac OS Xとか、この新しいUIフレームワークをLinux上にも提供してくれたりすると、個人的にはうれしいのだけれども…。それかx86をPowerPCが凌駕するような状況を作り出してくれたり、あとDarwin上にLinuxやWin32の環境サービスを乗っけたりするなど、広がりを見せてくれるといいのになぁ。しかしまぁ、クローズドなのがMacの世界で、それゆえ今まで生き残ってこれたような状況なので、ちょっと無謀だし危険かな？

あっそうそう、Mac OS Xを店頭で触ったあと、BSD系ってことで、家に帰って何となくFreeBSDをあるLinuxマシンのディスクの空きスペースにインストールしてみた。4.3unstableだったと思うけど、ブートフロッピーイメージをダウンロードして、FTPサイトを指定してネットワークからのインストール。シックリくればゲートウェイマシンをリプレースしようかとも思い、久しぶりにLINTファイルを読みながらカーネルを作り直したり、ゲートウェイとして必要な設定したりしてみたものの、今の僕にはちょっとトラディショナルすぎるのか、何か噛み合わない。これといってダメなところがあるわけじゃないのだけれど、なぜかなぁ、うむ、相性なんだろうなぁ、きっと。

インターネットとMTU

では前回に引き続き、ADSLの導入に関連したネットワークの調整を見ていくことにしよう。

前回の説明でMTU（最大転送ユニット）とMSS（最大セグメントサイズ）についてはある程度理解できたと思う。最低限、ネットワークの物理レイヤによっては、その伝送経路での転送ユニット（パケット）サイズが異なることがあるという点と、異なった場合、パケットのフラグメント（断片）化が生じるという点、またTCPのセッションの場合は、MTUを基にした最大セグメントサイズのやり取りがコネクションの確立の際に行われているという点の3つが理解できていればいいと思う。

今回はまず、ネットワークの物理レイヤによってMTUにどのようなサイズが採用されているのかを示しておこう。

MTU	Comments	
65535	Official maximum MTU	RFC 791
65535	Hyperchannel	RFC 1044
17914	16Mb IBM Token Ring	ref. [6]
8166	IEEE 802.4	RFC 1042
4464	IEEE 802.5 (4Mb max)	RFC 1042
4352	FDDI (Revised)	RFC 1188
2048	Wideband Network	RFC 907
2002	IEEE 802.5 (4Mb recommended)	RFC 1042
1536	Exp. Ethernet Nets	RFC 895
1500	Ethernet Networks	RFC 894
1500	Point-to-Point (default)	RFC 1134
1492	IEEE 802.3	RFC 1042
1006	SLIP	RFC 1055
1006	ARPANET	BBN 1822
576	X.25 Networks	RFC 877
544	DEC IP Portal	ref. [10]
508	IEEE 802/Source-Rt Bridge	RFC 1042
508	ARCNET	RFC 1051
296	Point-to-Point (low delay)	RFC 1144
68	Official minimum MTU	RFC 791

これはRFC1191から引用したもので、MTUの最小は68オクテット（バイト）、最大は65535、Ethernetは1500などと規定されている（まぁこれらはあくまで推奨なんだけど、オープンなネットワークでコミュニケーションを実現するためには準拠する必要があるし、実際従っている）。個々の詳細はそれぞれのRFCを参照して欲しい。

僕たち一般ユーザーが通常利用するのは、EthernetとPPPということになるが、これらのデフォルト値はともに1500ということがわかる。そのため、ほとんどの実装ではこのデフォルト値が設定されているのである。

前回Windows 9xのMTUのデフォルト値が576であることを紹介したが、これは上記のX.25 Networksに該当する。ということは、Windows 9xは、X.25 Networksの存在を前提にそのようなデフォルト値になったのだろうか？　これに関してはそうかもしれないし、そうでないかもしれない。本当の理由は僕にはわかりかねるので、悪しからず。ただ、576というのは、インターネットコミュニケーション仕様を規定しているRFC1122で実質的なMTUの最小サイズとして紹介されていて、とりあえずこのサイズにしておけば、難しいことを考えないで通信可能であるとも言い換えられる。インターネットを構成するネットワーク群では、576以下のMTUを採用した物理レイヤは存在しないと考えてもいいよ、と言っているのだ。

ちなみにX.25 Networksは、僕は使ったことないけど、DTEとかDCEといった用語が使われるシリアルポートベースのパケット通信プロトコルネットワークのことで、PPPやSLIPの登場以前、コンピュータ同士を接続する手段として、またインターネットを構成するネットワークの一部にも、これを利用している経路が結構あったらしい。

そのためRFC1122には576という記述があったというわけだ。しかし現在では、EthernetのMTUデフォルト値1500がインターネットの基準MTUサイズと言っても差し支えないと思う。

Path MTU Discovery

LANであってもネットワーク全体の物理レイヤを必ずしも統一できるとは限らないし、ましてインターネットでそのようなことができるわけがない。そのためネットワーク全体としては、複数のMTUサイズが存在することになる。

異なったMTUサイズの場合、ネットワークからネットワークの橋渡しをする、いわゆるルータは、それぞれのネットワークのMTUサイズに応じて、パケットを再度分割して転送するようになっている。しかし前回説明したように、IPフラグメントと呼ばれるこの手法は、受け取り側にそれを組み立て直す作業を生じさせ、DoS攻撃などにも悪用される危険性があった。どちらかと言うと疎まれていた機能だった。

そのためパケットのIPヘッダには、フラグメント化を制限するためのフラグが用意されている。これがセットされたパケットは、ルータがそれらをホップする際に参照され、その処理を抑制できるようになっている。

しかし単にこのフラグで抑制するだけだと通信できない、送信したパケットがどうなったかわからないという状況で終わってしまう。そこで、Path MTU Discoveryという規格が作られた。これは、RFC1063、1191で規定されている。

Path MTU Discoveryの基本動作は次のようなものだ。Aは送信元、Bは送信先、Cはルータである。またそれぞれのパケットの(DF)というのは、Don't Fragment、IPヘッダのフラグメント化の抑制フラグが有効になっていることを表している。

```
A > B: S 3183102292:3183102292(0) win 16384
```

```
    <mss 1448,nop,wscale 0,nop,nop,timestamp 12128 0> (DF)
B > A: S 2022212745:2022212745(0) ack 3183102293 win 49152
    <mss 1448,nop,wscale 1,nop,nop,timestamp 1592957 12128> (DF)
A > B: . ack 1 win 17248  (DF)
A > B: . 1:1449(1448) ack 1 win 17248  (DF)
C > A: icmp: B unreachable - need to frag (mtu 576)! (DF)
B > A: . ack 1 win 32768  (DF)
A > B: . 1:525(524) ack 1 win 17248  (DF)
```

これはホストAからホストBへのTCPセッションなんだけど、先頭の3つはそのハンドシェイクだ。wscaleやtimestampなどのオプションヘッダを付加しているので、MSSは1448となっているが、前回説明したようにこの場合のMTUは1500である。一応最初のハンドシェイクは問題なく完了している。

しかし実際にホストAからホストBに1448バイト分のデータを含むMTU1500のパケットを送信してみると、ルータCからICMPメッセージが返されている。そのメッセージ内容は、IPフラグメント化しないで、ホストBに送信するには、MTUを576にする必要があるというものだ。

次にホストBからホストAにACKが返されている。これはルータCがICMPでホストAに通知したMTUサイズの情報をホストBにも通知し、最初のハンドシェイクで交わしたMSSの情報を再調整することの同意パケットである。

その後、ホストAは524バイト分のデータを含むMTU576のパケットをホストBに送信している。ルータCからの指示をきちんと処理した結果だ。

このMTUサイズの自動調整機能がPath MTU Discoveryということになる。なお、ホストA、B、ルータCの通信に関連するすべてのノードがPath MTU Discoveryに対応していないと、この調整はうまく機能しないので注意しよう。

経路がもっと複雑、多くのルータを通過しなければならない場合でも処理の内容としては何も変わらない。ルータは、パケットのIPヘッダの送信先をもとに、転送すべきネットワークを見つけてそのインターフェイスに送り出す際、DFフラグもチェックしており、そのネットワークIFのMTUサイズが、そのパケットの実サイズよりも小さい場合には、分割して転送する手段は利用せず、送信元（必要に応じて送信先にも）に「転送できない、送信先にはこのMTUで送ってね」というICMPメッセージを生成して送信する。ただ、あるルータが経路全体のMTUサイズを知っているわけではなく、把握しているのは自分の周辺だけなので、その場合は、同様の調整が複数回行われることになる。

しかしながら最近は、ICMPパケットをホスト／ルータレベルやファイアウォールレベルでフィルタリングしてしまうことが多いのでPath MTU Discoveryが機能しないケースが増えてきている。ICMPは、pingなどでも使用されているネットワークの管理用プロトコルなんだけど、これまたこれを悪用する人がいて、DoSなどの被害が生じることがあるので、上記の措置が施されるようになったのだ。気の利いた管理者だと、ICMPのタイプに応じたフィルタリング（router-advertisementやrouter-solicitationを許可）を設定している場合もある。でも、インターネットでEthernetのMTU1500が基準MTUサイズとしてデファクトスタンダードになった今、Path MTU DiscoveryでMTUを調整する必要ってないに等しいし、それ以外のunreachable、redirect、time-exceeded、timestampなども提供しなくたって各送信元のホストが適当なタイムアウトでもって処理するように実装が変わってきているので、ICMPを一切受け付けないホスト／ルータも少なくないのが現状だ。まぁ、流量やアクセスが多いとICMPの処理も馬鹿にならないからね。

なお、ある経路を通過するのにパケットのフラグメント化が避けられない場合でも、通過後対応ルータが責任をもってそのフラグメントを再構築して元通りにするのであれば、DFフラグは無視される。

あとPPPoEでよくあるMTUサイズの問題の場合、残念なんだけどPath MTU Discoveryの効果はまったくない。Ethernetにしろ、PPPoEにしろ、ほかのインターネット経路にしろ、MTU1500で問題ないのだから、MTUサイズでのアドバイスは意味をなさないのだ。TCPだとMSSを指定すれば対応できるが、PPPoEは通過経路なのでMSSのやり取りに関与できない。つまりなすすべがないのである。結局、手動で設定するしかない。でも前回も述べたようにPPPのMTUが1500というのは、インターネットのデファクトスタンダードに合わせているだけなので、本来特定の目的で使用することがわかっていたなら、MTUを1540などのPPPヘッダで必要なサイズ分、大きく設定しておけばよい話なんだけどなぁ。

実際に自分のネットワークをチェックしてみる

これらを踏まえて、実際に自分のADSL経由のインターネット接続状態をチェックしてみたいと思う。

■IPフラグメントのチェック

まずフラグメントに関しては、tcpdumpでちょっとパケットの流れをチェックするだけで確認できる（linux01が内側のホスト名、linux02がインターネット上のホスト名）。

```
# tcpdump -i eth1 -vvv
linux02.www > linux01.36487: . 1461:2921(1460) ack 587
win 33580 (DF) (ttl 46, id 46634, len 1500)
linux01.36487 > linux02.www: . [tcp sum ok] 587:587(0)
ack 2921 win 11680 (DF) (ttl 64, id 0, len 40)
```

これはWebサイトとのやり取りの一部なんだけど、パケットのフラグメントの状態をチェックするには、外部のホストからのパケットサイズを確認すればよい。上記の場合、linux02からのパケットのサイズはlen 1500とMTU1500の最大サイズで分割されることなく届いていることがわかる。実はチェックとしてはこれだけでいい。簡単でしょ。MSSサイズから求める方法やpingなどを利用する方法もあるけど、tcpdumpのログの拡張表示を利用するとパケットサイズも一目瞭然なのだ。PPPoEでフラグメントが余儀なくされている場合は、同様のチェックで表示された値、len 14xxをMTUとして該当するインターフェイスに設定すればよいということになる。

■MTUサイズと性能のチェック

次にMTUサイズと性能の関係についてチェックしておこう。

テストは内部ネットワークのクライアントからテスト回線よりも高速と考えられるインターネット上のホストからのファイルのダウンロードスピードである。閉じた環境ではないため、さまざまな外的要因も加わるので、それほど厳密なテストとは言えないが傾向は見えてくる。以下が我が家のADSL回線の性能チェックということになる。

```
MTU576  166.03 KB/s
MTU1382 187.91 KB/s
MTU1400 184.30 KB/s
MTU1420 184.08 KB/s
MTU1430 188.06 KB/s
MTU1440 183.58 KB/s
MTU1452 185.22 KB/s
MTU1456 183.66 KB/s
MTU1458 185.98 KB/s
MTU1460 186.21 KB/s
MTU1462 186.67 KB/s
MTU1464 186.54 KB/s
MTU1466 187.02 KB/s
MTU1468 187.47 KB/s
MTU1470 187.32 KB/s
MTU1472 186.01 KB/s
MTU1474 187.79 KB/s
MTU1476 188.10 KB/s
MTU1478 188.92 KB/s
MTU1480 182.79 KB/s
MTU1488 184.09 KB/s
MTU1490 183.89 KB/s
MTU1500 185.85 KB/s
```

まずMTUサイズを1400から10刻みで1500まで変更しつつチェックを行い、その後気になる周辺を2刻みで調整してテストを行った結果である（例：ifconfig eth1 192.168.1.4/28 mtu 1478）。一応参考としてMTU576の場合も紹介している。

さて結果を見るとMTUが1400から1500の間で180Kバイト／秒以上でほとんど差がないということがわかると思う。MTU576でもそれほど差があるわけでなない。この結果からすると我が家のADSL回線の最適なMTUサイズは特定できず、ってことになる。実際問題として特に調整する必要はないってことだ。デフォルトのMTU1500で問題ない。

ただ、結果を細かく見ている誤差と思われていたデータの揺らぎに、とある傾向が見えてくる。MTU1478と1480の比較的大きな性能の違い、1478、1430、1382をピークにしたちょっとした性能の向上。ポイントは傾向が48バイト周期であることと、1500よりも1382のほうが、性能が若干良いという結果だ。

結論から言うと、これはATMネットワークによるものだ。我が家のADSL回線では、PPPoEではなく、PPPoAが利用されているのである。

ATMの特徴を簡単に説明しておくと、Ethernetと同じレイヤに位置し、パケットはATMセルと呼ばれる53バイトの固定長が使用され、そのうち5バイトがヘッダ、48バイトがデータ格納部で、大勢の客を同時に相手にする回転寿司のベルトコンベアのように、よどみなく膨大なコネクションを効率よく伝送するために開発された規格だ。Ethernetは可変長パケットだが、ATMは小さな固定長パ

ケットという点が重要なポイントとなる。

現在ATMはインターネットのバックボーンの主流になっている。我が家から米国の主要なサイトにアクセスしたある時点において次の経路が利用されていた。少し前まで155MbspのSTM-1程度だったと記憶していたのだけれど、622MbpsのSTM-4に置き換わっているようだ。そのうち2.4GbpsのSTM-16かぁ。

```
5  30 ms  ATM6-0-100.ar1.NRT4.gblx.net
6  40 ms  pos0-2-622M.cr1.NRT4.gblx.net
7 120 ms  pos5-1-622M.cr1.SEA1.gblx.net
```

さてPPPoAを利用するメリットだが、これはひとえに基地局のアクセスサーバの構成によるところが大きい。通常基地局アクセスサーバは、それぞれのISPの集約センターに接続され、そこからバックボーン、エクスチェンジ（IX）につながっているわけだが、接続距離や収容ユーザー数を考えると、これらの接続に直接Ethernetを使うのは無謀だ。そこで登場するのもATMというわけだ。パケットが小さく、ヘッダの占める割合も高いが、ATMスイッチは高速にハードウェア処理できるので、効率は非常に良いからだ。

ISPの上位ネットワークとATMで接続されている各アクセスサーバが各ユーザーと接続する場合、無駄な変換が不要になるATMを利用するのは当たり前だ、と言いたいところだが、個人向けサービスの場合、動的IPアドレスの割り当て作業などが必要となるので、変換が不要になることはない。う〜とまぁ、パケット（セル）が固定長なので処理が非常に楽になることは間違いない。上記のテスト結果を見ればわかるようにMTUサイズにも影響されにくいといったところがメリットかな。オーバーヘッドは1割と比較的大きいんだけどね。

処理に関してだが、EthernetからATMネットワークに転送される場合、EthernetのMTU1500のパケットは、ATMセルのデータ部のサイズ48バイトに分割されて32個のセルとして経路を通過することになる。これをフラグメント化と呼べなくもないが、通常のIPフラグメントと違い、それぞれに送信先（IPヘッダ）が付加されているわけではないので、必ずATM経路の終点で再構成が行われる。ATMでは、輻輳などでセルが捨てられることはあっても、フラグメントパケットが送信先に到着することは決して起こらない。

先ほどの性能テスト結果だと、MTUが1478、1430、1382のときに最も良い結果となっているわけだが、セルのデータ部のサイズ48バイトから計算すると、1488、1440、1392のときに隙間なくデータが格納されていることになる。この差の10バイトはPPPヘッダによるものだ。

ちなみにMTU1478のときの188.92Kバイト／秒を48で割って、53をかけてみると208.60、回線速度の理論値は200Kバイト／秒。PPPoEの場合、パケットオーバーヘッドの理論値はPPPoAより小さいけど、実質レートがどの程度なのかは知らない。でもまぁ、テスト上のピーク性能だと198Kバイト／秒程度になるような気がする。

性能テスト結果は、ADSL回線でPPPoAが使用されていることと、その特徴を表していて、この経路の転送効率だけに注目するとMTU1478が最適と言っている。MTU1500より2%近く効率が良いのだ。気にする人は気にするだろうし、どうでもいいと言えばどうでもいい差である。

MTU1478とMTU1500。僕は通常MTU1500のまま何も設定をいじらないで使っている。でも、CD-ROMのISOイメージファイルなど巨大なサイズのダウンロードを行う際、気が向けばifconfigコマンドでMTU1478を設定するようにしている。実際、計測上の違いが出る。でもめったに設定しないんだけどね。

なおこれらは、我が家のADSL回線がボトルネックと仮定した場合の話で、一般的なインターネット接続では、ボトルネックはさまざまな部分に生じるので、MTU1500が最適であることに変わりはない。

■

はてさて、ネットワークの調整個所をいくつか見てきたけれども、結局何も変える必要がないという結論に達したわけだ。しかしこれは予想された結果だ。デフォルト値に問題があるようなハードウェアやソフトウェアは、早く消えてしまったほうがいい。ユーザーに不必要な設定を求めるものもだめだ。ADSLが本当に普及するには、そうでないと困るでしょ。

でも次回はゲートウェイとADSLモデムのパケットフィルタの設定などを行う予定。

●時間合わせとこ

インターネット上で日本標準時を提供するNTPの試行サービスが開始されている。近くの正確な時間に合わせちゃおう。詳細はここ参照http://www.jst.mfeed.ad.jp/。今使っているNTPサーバと、ホップ数とかピンポンの時間を比べるなどしてね。もちろん近くて反応時間が安定しているサーバ群のほうがよりよいから。

Linux ガベコレ

Linux Garbage Collection

目からウロコの用語辞典

文: しのはらひろあき
Text : Hiroaki Shinohara

第14回
【組み込みLinux】（くみこみ・りなっくす）
【リアルタイムOS】（りあるたいむ・おーえす）
【Midori Linux】（みどり・りなっくす）
【de-Modo Linux】（でもど・りなっくす）
【戦隊シリーズLinux】（せんたいしりーず・りなっくす）

組み込みLinux
【くみこみ・りなっくす】

ソースばっちり

Linuxの普及とIT化の進展により、今までは考えられなかった用途にもこのOSを利用しようという意気が盛んである。ひとつの例が家電や工業機器への応用であり、これら機器をLinuxを用いて制御するものである。こういった用途に特化したLinuxを「組み込みLinux」と呼ぶ。

組み込みLinuxの恩恵は多大であり、家庭にある何の変哲もない冷蔵庫や炊飯器、電気ポットなど、あらゆる電化製品でLinuxを動かすことにより、いつでも身近な機器を使ってカーネルのリコンパイル作業が可能になる。なにより、Linux最大の特長はオープンソースだという点であり、たとえ家電に身をやつしてもその長所は健在である。Linux炊飯器なら、炊きあがりが水っぽいときでもソースを読んで問題がどこにあるか確認できる。また、固めのご飯が食べたいという亭主関白のわがままなニーズにも、ハードコーディング値を直接書き換えることによって応えることが可能だ。

また、現在試作が進んでいるLinux冷蔵庫では、食料品、飲料、野菜などを高度に抽象化することによって、あたかもファイルシステムであるかのようにシームレスに扱うことができる。このため、買ってきた冷凍ギョウザを冷凍庫に入れるにも、

```
# mount -t freezedry /dev/freezer /mnt/gyouza
```

と、1行の処理ですむ。食べる場合にはアンマウントして取り出すだけと、ここでもごく一般的なLinux文化を踏襲している。

なお、組み込みLinuxの類似品として、性格とガラの悪い女性をエミュレートした「クミコLinux」（編集部注：自称、性格もガラも悪くないクミコさんからの苦情は編集部では受け付けておりません。命名に至った理由などは開発元にお問い合わせください）、マイラインプラスに登録すれば月々電話基本料金のみで2時間分のインターネット接続がタダという「コミコミLinux」などがある。

リアルタイムOS
【りあるたいむ・おーえす】

組み込みOSでは、イベント発生やリクエストに応じて決められた時間内に確実に処理をこなすことが要求される。このような目的に適した設計のOSを、リアルタイムOSと呼ぶ。家電のような機器では、リアルタイム処理が行えなければ各種のイベントがかみあわず、処理がうまくいかないことがある。たとえば、もし炊飯器がリアルタイム動作しなければ、米の投入というイベントに対して“炊きあげ”という処理が時制上適切に起動しないかもしれない。米を入れてもいつまでも炊き始めなかったり、米を入れる前にほかほかご飯が炊きあがるといった、物理常識を超越した炊飯器が登場することも予想される。すると米を買わなくてもご飯が食べられるわけで、お米屋さんがかわいそうだから安定した

リアルタイムOSの開発は至上命題だ。

現在では、リアルタイムOSの人間への搭載も研究され始めている。締切を守らないライターの脳にリアルタイムOSを埋め込めば、編集者のリクエストに応じて決められた時間内に（通常数十μsec～数msec）原稿をアップする「リアルタイムライター」の誕生も夢ではない。実はこのコーナーの編者の脳にも、Linux magazine編集部謹製リアルタイムLinuxが埋め込まれているが、脳と腕デバイスをつなぐPCIバスのボトルネックにより、脳内原稿完成から校了までに数日～数週間の遅延が発生するという問題がすでに判明している（編集部注：ということにしたいのですね）。

Midori Linux
【みどり・りなっくす】


みどりだに"ょ

Linuxの生みの親が勤めている某企業が発表した、組み込み用Linux。同社が省電力CPUとともに組み込み用途に本気で取り組んでいく姿勢の表れとして注目されているが、同時に発表されたイメージキャラクターの画像は業界により大きな衝撃を与えた（http://midori.transmeta.com/manual/）。その画像がグラマラスなボディをタイトなSF風コスチュームに包んだ、まさにジャパニメーション的デザインの女性だったからであり、これによって「Linuxコミュニティにはアニメファンが多い」という事実があらためて確認されたのである。

一部には「デッサンが狂っているのでは？」、「目が小さい」、「唇が武器」などという評価もあるMidori Linuxだが、怪電波により流れてきた信頼すべき情報によれば、開発元ではキャラクター商品としての展開も考えているとのこと。20分の1サイズガレージキットの販売を手始めに、OVA・DVDボックス（初回限定ソースコードつき）、そして恋愛シミュレーションゲームが予定されているという。来年の今ごろには、ネットでも「Midori萌え～」な人々がファンサイトを立ち上げ、2ちゃんねるでは「でじきゃらっとMidori、どっちが萌える？」といった関連スレッドが大量に並ぶ光景が見られることだろう。

de-Modo Linux
【でもど・りなっくす】

Midori Linuxを採用した家電メーカーが不謹慎にもエンベデッドWindowsにうつつを抜かしていると、Midoriちゃんが怒って実家に帰ってしまう。これを「出戻りナックス」と呼ぶ。このネタ、書いてみたらかなり苦しかった。

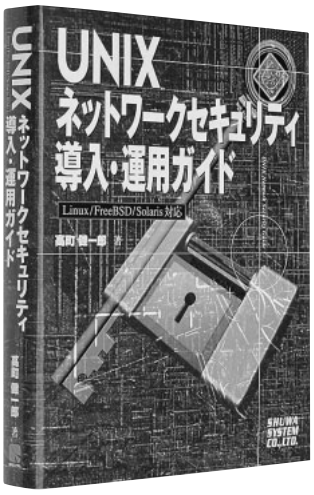
戦隊シリーズLinux
【せんたいしりーず・りなっくす】


キメポーズ

いかにも近未来的にして正義の味方風のMidori Linuxマスコットコスチューム、そして色をもとにしたネーミングに触発され、日本のLinuxコミュニティで進行しているのが「戦隊シリーズLinux」である。これはスシ、フジヤマにつぐ日本の伝統芸能である「スーパー戦隊」ものにならい、それぞれカラーリングの異なる用途別単機能サーバアプライアンススイートをLinuxで作ろうという計画で、すべての特撮ファンの夢を実現するものといわれている。

まず、「Linux red」は、某大手ディストリビューションに商標侵害の疑いをかけられながらも闘う熱血漢サーバ。勢いあまってときおりコアダンプすることもあるが、たいせつなDNSサーバを担当する、リーダー肌の頼りになるやつだ。中国人クラッカーになんか負けるな。「Linux blue」は知的なクールガイ。本当なら自分のほうがリーダーとして適任なのに、と思いながらもredをサポートして戦隊全体を支える、縁の下の力持ち。担当はセカンダリDNSサーバだ。「Linux black」は、かつて親兄弟をすべて敵にクラッシュさせられたという悲しい過去をもつメールサーバ。番組終盤が近づくころ、実は不治の病に侵されているという衝撃の事実が明らかにされ、女性ファンの涙を誘う予定。管理者は事前に代替機を購入しておく必要があるので注意されたし。「Linux yellow」はカレー好きな大型サーバで九州出身。HTTPサーバを担当。口ぐせは「～でごわす」……の予定だが、最近の風潮に合わせて知的美女タイプのキャラクター実装となる可能性もあるという。最新のソースをanonymous CVSで確認してほしい。「Linux pink」は正当派特撮ヒロイン仕様で、ケースの周りにミニスカート風のヒラヒラをつけるオプションも用意される。今のところNTPサーバを担当予定。「メンバーにファイルサーバがいない」という指摘がしばしばなされるが、2クール目ごろにredの生き別れの兄という設定で「Linux fire」として登場する（4クール目にredの身代わりに火を噴きながらクラッシュする）。乞うご期待！

Books



UNIXネットワークセキュリティ導入・運用ガイド

高町健一郎 著
秀和システム
B5変形判／400ページ　　本体価格 3200円

ネットワークセキュリティに関する本の中には、やたらとクラッカーの脅威を強調していたり、攻撃の詳細な手口を延々と解説していたりするものがある。それはそれでおもしろい（といっては語弊があるかもしれないが）し、興味を引かれる話題である。実際にネットワークを管理する立場から見ても、クラッカーの行動を理解しておくことは必要だろう。しかし、より重要なのは、どのようにセキュリティを適用し、それを管理していくかということだ。

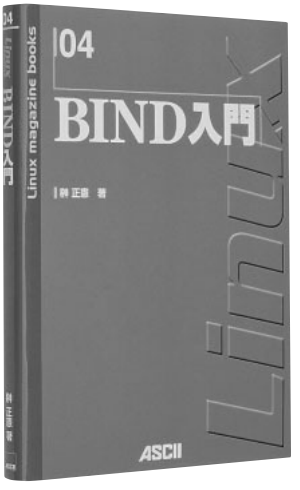
本書はまさに、その点にフォーカスした内容になっている。Web、メール、DNS、プロキシなどの各種サーバでのセキュリティ設定、SSHやパケットフィルタリングに必要なツールの紹介と設定について、「すべきこと」と「その手順」が丁寧にとてもわかりやすく解説されている。xinetd、netfilterといった新しいツールにも対応している。

BIND入門

榊 正憲 著
アスキー
A5判／200ページ　　本体価格 1600円

DNS（Domain Name System）は、ドメイン名とIPアドレスを相互に変換する。世界中にある膨大な数のホスト名を解決をするこの仕組みは、いったいどのように実現されているのだろうか。また、DNSサーバプログラム「BIND」を運用するにはどう設定すればよいのだろうか。このような疑問を持っている方も多いだろう。

DNSの解説書籍といえば、オライリーの『DNS＆BIND 第3版』が定番となっているが、これは本格的すぎて、誰でも気軽に読めるものとは言い難い。『BIND入門』は、本誌で連載中の「Linux日記」で好評だったBINDの解説記事をもとに、大幅に加筆して単行本にまとめたものだ。書籍として読みやすいよう再構成されており、DNSの動作原理から、BINDの運用方法までをわかりやすく解説している。インターネット常時接続環境を手に入れ、DNSサーバを運用したいと考えている方に広くお勧めしたい一冊である。

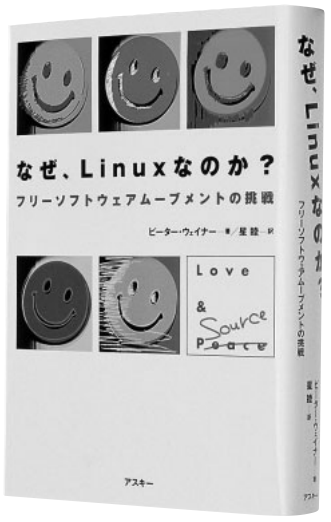


フレッツ・ISDN／ADSLでサーバーを立てる本

清水隆夫／坪山博貴／津守美弘 共著
インプレス
B5変形判／256ページ　　本体価格 2200円

このところ、ますます活況を呈しているのがインターネットへのブロードバンド接続サービスだ。Linux magazine編集部でも、ネットワークにチトうるさい編集長を筆頭に、「あのサービスがいい」、「こっちは料金が下がった」、「ウチのは○○Mbpsになった」などと盛り上がっていることしきり。読者のみなさんの中にも、そろそろ加入しようかと思案中の方がいるのではないだろうか。

そこで気になるのが、サービスの種類と選択のポイント、必要な機器や料金などのコストである。この本を読めば、このあたりの実情をチェックすることができる。サーバ構築についても、ひと通りの手順を確認しておくことが可能だ。ユニークなのが、ルータの設定、LinuxとWindowsでのネットワーク設定に加えて、OpenBlockSとCobalt Qube 3を取り上げているところ。幅広い選択肢を提示している点が特色となっている。



なぜ、Linuxなのか？フリーソフトウェアムーブメントの挑戦

ピーター・ウェイナー 著／星 睦 訳
アスキー
四六判／400ページ **本体価格 2000円**

もともと「Linux」とはカーネルに付けられた呼称である。カーネルにLinuxを採用し、主にGNU Projectによって開発されたシステムプログラムやユーティリティ（bash、glibcなど）と組み合わせたOS（つまり現在、単に「Linux」と呼ばれることの多いディストリビューションのOS部分）は、厳密には「GNU/Linux」と呼ばれるべきものなのである。

とまぁ、薀蓄を垂れてみたわけだが、本書にはこのほかにも、「フリーソフトウェア」と「オープンソース」の違い、「Linux＝Linux Is Not UNIX」の持つ意味、といったLinuxに関連する薀蓄ネタがてんこ盛りである。こう書くと「Linuxの小ネタを集めた薀蓄本か？」と誤解されるかもしれないが、実際には非常にまじめな書籍で、Free Software Foundationの設立から始まるフリーソフトウェアの潮流を詳細に追っていき、最後にその意義と未来について考察している。Linuxを生み出したムーブメントの全容に迫る力作である。

UNIX／Linuxコマンドリテラシー

アミール・アフザル 著
小嶋隆一／太田耕三 訳
ピアソン・エデュケーション
B5変形判／524ページ
本体価格 3400円

sendmail
ーメールサーバーの設定・運用・管理ー

江面 敦 著／
矢吹道郎 監修
テクノプレス
A5判／252ページ
本体価格 2500円

Linux IPスタックコメンタリー
オープンソースコード詳解

Stephen T. Satchell／
H.B.J. Clifford 著
小嶋隆一 訳
セレンディップ／小学館
A4変形判／672ページ／
CD-ROM1枚付き
本体価格 8500円

正規表現の達人

IDEA・C 著
ソフトバンク
パブリッシング
A5変形判／256ページ
本体価格 2400円

読者の声

俺にもいわせろ！

編集部では、ゲームボーイアドバンス（略称：ゲボア）が大流行です。先日、数人の編集部員が「打ち合わせ」と称して、会議用スペースで対戦ゲームをしているのが発覚しました。なんと、Linux magazineやUNIX MAGAZINEなどの編集部を統括する局長に現場を目撃されたのだそうです。くわばらくわばら。

Itaniumに期待大

Itaniumの特集面白かったです。いつになったら正式にリリースされるのでしょうかね。ちょっとテスト段階が長いと思います。さて、気になるお値段はいくらになるのでしょう。個人レベルで購入できるようになるのは何年（何十年？）か先でしょうかね。

（村山一馬さん）

4月号からガベコレがなくなったと思っていたら、5月号は復活していましたね。一番好きな連載なので、安心しました。また、Itanium（IA64）の特集も良かったです。これからはIA64対応のディストリビューションが出てくると思うので、また特集をお願いします。

（森岡祐一さん）

Itaniumは、2.5インチのハードディスクをひとまわり小振りにしたような外観をしています。大きさのみならず、ずしりと伝わる重量感にも驚かされました。

IA64のCPUとはいえ、Red Hat Linuxや、Turbolinuxが動作している画面を見ると、ふだん見慣れたIA32用Linuxとほとんど変わるところがないので、少しがっかりしましたが、考えてみれば、コレってすごいことなのかもしれませんね。Itaniumが普及したら、みなさんは何に使いたいですか？

話は変わりますが、森岡さん、4月号のガベコレはカラーページにあります。5月号からは元通り、モノクロページでお届けします（ガベコレ担当は、年に1回くらいはカラーページに掲載したいと言っています）。

クライアントOSとしてのLinux

今年の1月号から読ませていただいています。今回の「Vineで極めるLinuxクライアント活用術」は私としては、今一番欲しいと思っていた情報でしたので大変うれしいです。本を片手にソフトウェアをアップデートしたりして、自分で「なるほどな」と感心しています。これからも初心者にやさしく、また経験者にも納得できるような誌面作りを頑張ってください。

（山田光晴さん）

Linuxを使っていて、いつも疑問に思っているのですが、ソフトをインストールするとき、tarボールを展開する場所や、インストールしたあとに残ったファイルを消してよいものか色々悩みます。

そのほかに、本当のLinux初心者向けの記事を載せてほしいです。専門書などより、雑誌に載っているほうが、初心者にはとっつきやすいのですが。専門書などでは、地域によっては取り寄せないと入手しにくかったりするので（あと、経済的にも）。

（もぐらちゃん）

音楽、画像、動画など、Linuxでもマルチメディアデータを気軽に扱えるようになってきました。また、USBやIEEE 1394への対応など、周辺機器も活用できる環境が整いつつあります。Linuxは、ますます活躍の場を増やしていくことでしょう。

さて、もぐらちゃん（さん？）がお持ちの疑問にお答えします。tarアーカイブを展開する場所は、基本的にどこでもよいのですが、自分のホームディレクトリに適当な名前のディレクトリを作り、その中に展開するとよいでしょう。インストールがすんだら、そのディレクトリごと消してしまって大丈夫です。念のため、tarファイルは残しておきましょう。

プリンタに挑戦！

先月、インターネット接続に成功したので、次はプリンタだ！　と思い3日前に購入したEPSONのPM-900C……ではなく、メインプリンタの座を奪

われ、部屋の片隅に追いやられていたCanon BJC-420JをLinux専用プリンタにしようとたくらみました。が、現実は厳しいもので、GhostscriptをBJC600にして印刷を試みたところ、“Unrecoverable error: unknownerror in kfVFlib”とかいう英文に続き、文字列がだーっと印刷されて、正しく印刷できません。プリンタとかインターネットとかインストールとか、初心者がぶち当たりやすい壁を打ち破るための特集をしてもらえないでしょうか？

（角谷千穂さん）

エラーメッセージを元に、Webで検索したところ、/usr/share/ghostscript/5.10/kanji以下にある、kconfig.psのリンクを、kconfig-basic.psからkfvflib.psに張り替えるという方法で、問題を解決した例があるようです。ファイルの置かれているディレクトリは、ディストリビューションによって少々違うかもしれません。たとえば、Vine Linux 2.1.5では、/usr/share/ghostscript/5.50/vflib/になります。プリンタの設定については、今月号の第3特集もご覧ください。

ステップアップにはムックをどうぞ

連載記事のムック化をまたやってもらいたいです。今度は、「viはじめました」＆「Emacsはじめました」、「Rubyで行こう」、「賢く使うUNIX」とか期待してます。

（黒田 信一郎さん）

まだ、Linuxの知識はいまいちですけど、毎月Linux magazineを買って勉強していきたいです。内容がとても充実していると思います。この雑誌を集めることで、必要なときに調べられそうです。初心者向けの記事を作ってくれるととてもありがたいです。

（北見大岳さん）

5月23日に初級者を応援するムックを発売する予定です。その名も『Linux magazine for beginners』。

Linux mgazineで好評だった特集の中から、「Linuxの設定ファイル徹底ガイド」、「Linuxレスキュー『疑問・難問・200問』」など、Linuxの基本的な設定や使い方を解説したものを厳選して掲載しました。付録CD-ROMには、Vine Linux 2.1.5を収録して、価格は880円（税込）です。Linuxを使い始めた方、バックナンバーを買い損ねた方にお勧めの1冊です。

Linux習得の秘訣

Linuxという言葉が有名になってだいぶたちますが、まだ扱い切れていません。自分は今、システムエンジニアを目指して学校に通っていますが、これからのことを考えるとやはりLinuxはぜひ習得したいです。これからもよろしくお願いします。

（馬場伸哉さん）

購読しはじめて1年経ちましたが、前はわからなかったことが今はかなりわかるようになってきました。やっぱ、継続は力なりですね。

（國信さん）

LinuxなどのUNIX系OSをうまく使えるようになるには、慣れるのが一番です。では、どのようにすれば早く慣れるのでしょう？　それは、毎日使うということにつきるでしょう。國信さんのおっしゃるとおり、継続は力です。外国語を習得するのにも似ているかもしれません。担当は外国語はカラキシだめなんですが……。

慣れてくれば、初めての作業でも、どうすればよいのかが類推できたり、作業を省力化できるようになりますので、苦労することは徐々に減っていきます（たぶん）。

実運用システム構築例

病院勤務をしています。昨年、大混雑した介護保険開始時に、予算がなかったので、Linuxサーバ、PostgreSQL、PHP3、Javaスクリプトを使い、Webで動く介護給付費請求システムを作りました。複数のサービス事業所からネットワークを介して利用できるようになっています。ハードも手作りマシンで、本当にダウンせず、国保連合会に給付費請求できるのか不安でしたが、この1年間ノンストップで動作し、きちんと介護給付費請求もできました。メーカー製品を買えば、2000万円くらいかかったのですが、Linuxのおかげで、投資額は30万円程度と私の人件費でシステム構築ができました。これに気をよくした私は、Linuxサーバから基幹業務のサーバに検索をかけ、結果をHTMLで表示するとか、色々作ってみました。しかし、COBOLでHTMLを書くとは思いませんでした。

（和田博知さん）

すばらしい。ビジネスの現場で、Linuxを活用する方が増えてきましたが、ご自分でシステムを構築してしまうなんて！　Linuxユーザーの底力を見せていただきました。

今月も、みなさまからのメッセージをお待ちしております。

付録CD-ROMに収録した
Turbolinux Server 6.5のインストール

本誌付録CD-ROM収録のTurbolinux Server 6.5はFTP版です。非商用ソフトだけが含まれており、製品版を販売しているターボリナックスジャパン株式会社からサポートを受けることはできません。

CD-ROMのメディアに不良があった場合は、お手数ですが（linux-cd@ml.ascii.co.jp）宛にご連絡くださるようお願いいたします。

1 インストールの前準備

これからTurbolinux Server 6.5のインストールを始めます。多くのネットワークカード、ビデオカード、モニタは自動認識されますが、自動認識されない場合に備えて、あらかじめハードウェアのマニュアルなどを用意しておくとインストール中にあわてなくてすみます。

また、レスキュー用のフロッピーを作成するために、空のフロッピーを1枚用意しておいてください。CD-ROMブートができないマシンを使う場合は、インストーラ起動用のフロッピーも用意します。

2 ブートフロッピーの作成

CD-ROMブートできないマシンにTurbolinuxをインストールする場合は、以下の手順でブートフロッピーを作成します。

まず、Windowsマシンに付録CD-ROMと空のフロッピーディスクをセットします。次に、エクスプローラを使ってCD-ROMの［DOSUTILS］を開き、［BOOT］をダブルクリックします。するとDOS窓が立ち上がるので、［Enter］をタイプしてインストーラ起動用のブートディスクを作成します。

3 インストーラの起動と使用言語の選択

CD-ROMやブートフロッピーをセットしてマシンを起動すると、Turbolinuxのインストーラが起動します。［boot］プロンプトが表示されるので、［Enter］を押します。しばらくすると、使用する言語の選択場面に切り換わるので、［↓］キーで［Japanese］を選択し、［Tab］キーと［Enter］キーを使って［OK］を押します。しばらくすると、Xを使ったグラフィカルなインストール画面が表示されます。

なお、［boot］の箇所で［text］と入力して［Enter］を押すと、テキストベースのインストーラが起動します。また、インストーラがグラフィカルな画面の表示に失敗した場合は、自動的にテキストベースのインストール画面が起動します。

インストールクラスの選択

新規にTurobolinuxをインストールする場合は［標準インストール］を選択し、古いTurbolinuxをアップグレードする場合は、［アップグレード］を選択します。

ここでは、新規にインストールするものとして［標準インストール］を選択して次へ進みます。

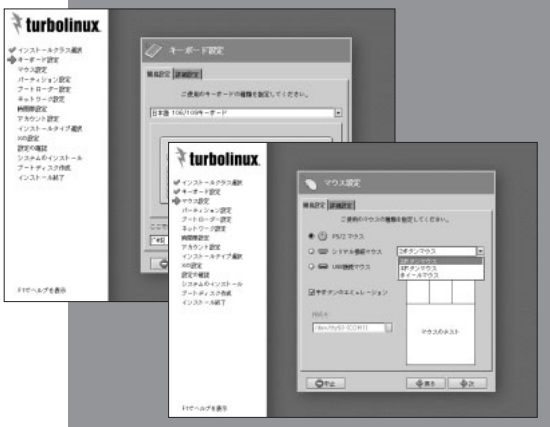
4

キーボードとマウスの設定

使用するキーボードタイプをプルダウンメニューで選択します。たいていのキーボードはプルダウンメニューに含まれていますが、Sunのキーボードなど特殊なものを使う場合は、［詳細設定］タブで画面を切り替えて、キーボードタイプを選択します。キーボードを選択したら、［ここで選択した設定をテストしてください］の欄でキーボード入力をテストします。［Shift］キーを押しながら［1］、［2］、［3］を押して、キーボードの刻印どおりの記号が表示されたら、キーボードの設定は成功しているでしょう。

次にマウスを設定します。使用するマウスタイプをプルダウンメニューから選択します。［2ボタンマウス］を選択した場合は、［中ボタンのエミュレーション］をチェックします。これは、2ボタンマウスの左右のボタンを同時に押すことで、3ボタンマウスの中ボタンを押す効果を出すオプションです。マウスタイプを選択したら、［マウスのテスト］の領域でマウスの動作をチェックしましょう。

5

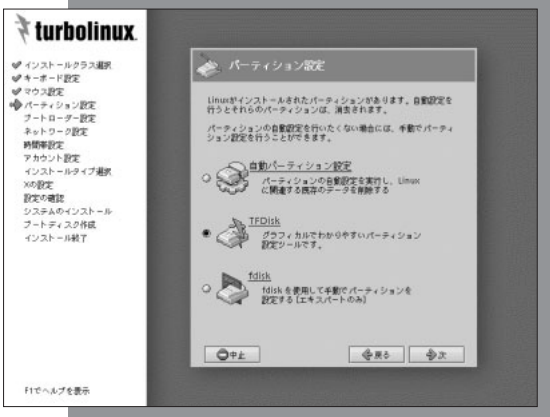


パーティション作成ツールの選択

Turbolinuxのインストーラには、パーティションを作成するために3つのツールが用意されています。［自動パーティション設定］を選択すると、既存パーティションの削除後にTurbolinux用のパーティションが自動で作成されます。［TFDisk］ではグラフィカルな画面で手動でパーティションを作成し、［fdisk］ではテキストベースの画面でパーティションを手動作成してから［TFDisk］を使ってパーティションにマウントポイントを割り当てます。

ここでは［TFDisk］を選択してインストールを進めます。

6



パーティションの作成

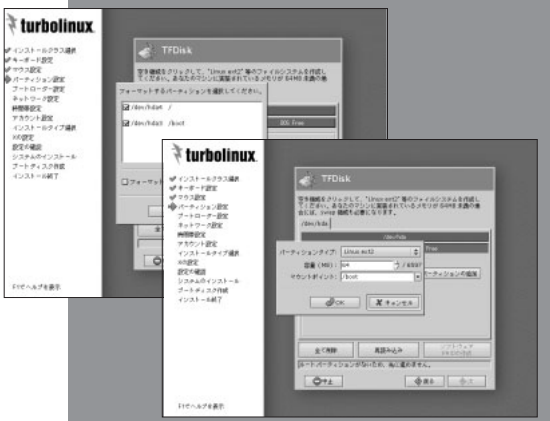
Turbolinuxのインストーラでは、Linuxシステム用パーティション［/］とSwapパーティションのほかに、Linuxカーネルを格納する［/boot］パーティションの作成を推奨しています。以下は、それにしたがって3つのパーティションを作成した例です。

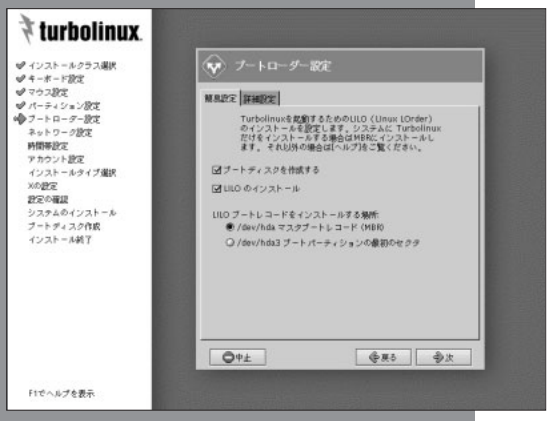
まず、ハードディスクの空き領域をマウスでクリックして［パーティションの追加］を押します。すると、ダイアログ画面が別に開かれるので、

パーティションタイプ	容量	マウントポイント
Swap	マシンが搭載するメモリの1～2倍	なし
Linux ext2	64Mバイト	/boot
Linux ext2	2Gバイト以上	/

の3つのパーティションを作成します。［/］パーティションには、新しく選択項目に追加されたext3ファイルシステムを選択してみるのもおもしろいでしょう（ReiserFSは選択できません）。このあと作成したパーティションをフォーマットします。

7





8

ブートローダLILOの設定

ブートローダLILOのインストール先を選択します。TurbolinuxをWindowsと共存させる場合は、［LILOブートレコードをインストールする場所］に、

Windows 9x／Me	/dev/hda マスターブートレコード（MBR）
Windows NT／2000	/dev/hdax ブートパーティションの最初のセクタ

をそれぞれ選択します。いずれのWindowsと共存させる場合でも、ブートディスクを作成するために［ブートディスクを作成する］をチェックしておいてください。

9

ネットワークとタイムゾーンの設定

TurbolinuxをDHCPサーバからネットワーク情報を取得する環境で使う場合は、［DHCPを使用して設定する］を、DHCPサーバのない環境で使う場合は［DHCPを使用して設定する］のチェックをはずして、IPアドレスなどのネットワーク情報を入力します。なお、この画面ではダイヤルアップ接続の設定はできません。

次にタイムゾーンを設定します。デフォルトでは日本時間がセットされているので、たいていのユーザーはタイムゾーンを設定する必要はありません。［システムクロックでUTCを使用］は世界標準時に合わせるためオプションです。Windowsなどと共存させる場合は、このオプションをはずしておくのがよいでしょう。

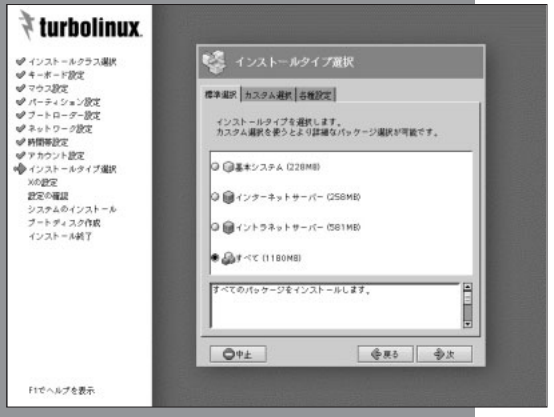
Turbolinuxのインストーラでは新たにNTPを設定できるようになりました。NTPを使ってマシンの時刻を設定する場合は、［タイムサーバー］タブをクリックして使用するNTPサーバを入力します。

10

ユーザーアカウントの作成

まずはLinuxの管理者用ユーザー（ログイン名はroot）のパスワードを設定します。［rootパスワード］と［確認］に同じパスワードを入力します。［確認］欄の右側に「確認しました」と表示されれば設定は成功しています。

次に、メールの読み書きなどを行う一般ユーザーを作成します。まずは［ユーザー作成］タブをクリックして画面を切り替えます。［ユーザー名］に一般ユーザーのログイン名を入力して、［パスワード］と［パスワード（確認）］に一般ユーザー用のパスワードを入力します。［パスワード（確認）］の右側に「確認しました」と表示されたら［追加］を押し、さらに［次］を押して次に進みます。



11

インストールタイプの選択

インストールするパッケージグループを選択します。各グループを選択すると、そのグループの説明が下の欄に表示されます。ここでは、［すべて］を選択してインストールします。［すべて］を選択した場合は、ハードディスクを約1.3Gバイト消費します。

モニタとXの設定

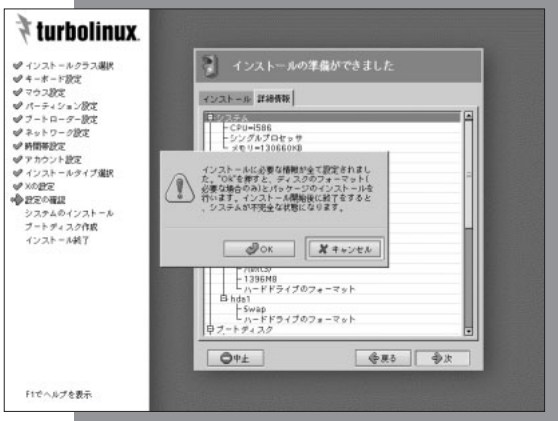
使用するモニタが自動認識されなかった場合は、リストの中からモニタを選択します。使用するモニタがリストにない場合は、［メーカー］から［Generic］を選択して、モニタのマニュアルを参照しつつ［モデル名］から無理のない解像度のものを選択します。また、液晶モニタの場合は［Laptop Display Panel］から選択します。

次にXを設定します。ほとんどのビデオカードは自動認識されるでしょう（画面の例では［S3 968］と認識されています）。ひとまずインストーラが自動設定した［デスクトップカラー］と［デスクトップサイズ］のままで、［この設定をテストする］を押してXの表示をテストします。表示に成功したら、Turbolinuxのデスクトップ画面が表示されるので、表示される質問に［はい］と答えてテストを終了します。テストに失敗した場合は、［Xの設定を行わない］をチェックして、インストールを終えてからturboxcfgを使って、じっくりとXを設定してください。

なお、［ログイン方法を選択してください］では、Xの表示テストに成功した場合のみ［グラフィック］を選択してください。［グラフィック］を選択すると、Linuxの起動後にグラフィカルなログイン画面が表示されます。

パッパッケージインストール前の確認

［詳細情報］タブをクリックすると、ここまでの設定情報を確認できます。設定情報を変更する場合は、［戻る］で該当画面まで戻って再設定します。［インストールの準備ができました］で［次］を押すとパッケージインストールの確認を求めるダイアログが表示されます。［OK］を押してインストールを始めましょう。



パッケージのインストール

［インストールタイプの選択］で［すべて］を選択した場合は、パッケージのインストールが終わるまでに、128Mバイトのメモリを搭載したAMD K6-II/350MHzマシンで約30分かかります。画面の下段にある［詳細を表示］を押すと、インストール中のパッケージ名などより細かな情報が表示されます。

パッケージのインストールが終了すると、追加パッケージのインストールに進みますが、本誌付録のCD-ROMには追加パッケージを収録していません。［追加パッケージのインストールをスキップする。］をチェックしたまま次に進みます。

ブートディスクの作成

Linuxをハードディスクから起動できなくなった場合などに備えて、レスキュー用のフロッピーディスクを作成します。空のフロッピーディスクをセットして、［次］を押してください。

フロッピーディスクの作成が終わるとインストールは終わりです。［完了］を押し、CD-ROMとフロッピーディスクを抜いてください。お疲れさまでした。