PentiumIIIXeonを最大8基搭載するハイエンドサーバ PowerEdge 8450

デルコンピュータは、PowerEdgeシリーズ初の8-Wayサーバ「PowerEdge 8450」を11月15日より発売した。PowerEdge 8450は、同シリーズの最上位モデルで、PentiumIIIXeon 700MHzを最大8基搭載することが可能なエンタープライズ向けサーバだ。

IntelのProfusionチップセットを採用し、最大32GバイトのECC SDRAMが搭載可能である。オンボードにUltra2/LVD SCSIコントローラを2個、Intel EtherExpress Pro/100＋ネットワークを搭載している。PCIスロットは、64ビット（66MHz）×4と64ビット（33MHz）×6の合計10スロットを装備しており、全スロットがホットプラグに対応している。

エントリー構成の場合、PentiumIIIXeon 700MHz（1Mバイトキャッシュ）、256MバイトECC SDRAM、512Kバイトキャッシュコヒーレンシフィルタ、オンボードUltra2/LVD SCSIコントローラ×2、18GバイトUltra2/LVD SCSIハードディスク（10000rpm）×1台、40倍速CD-ROMドライブ、オンボードIntel Pro/100＋ネットワーク、HP OpenView NNM Special Edition、OSなし、冗長化電源、ラックマウント用日本語キーボード、ラックマウントキットで、243万7000円となっている。

高さは7U（311mm）、奥行きは711mmで、200Vの電源が必要。電源ユニットと冷却ファンは標準で二重化（最大で三重化が可能）されていて、ホットプラグに対応する。

OSはRed Hat Linux 6.2Jを選ぶことが可能で、今後Red Hat Linux 7Jにも対応する。

PowerEdge 8450	
発売日	2000年11月15日
発売	デルコンピュータ株式会社
TEL	044-556-6190
価格	243万7000円〜
URL	http://www.dell.com/jp/

Hardware　発売日　2000年12月1日

FAパソコンで初めてTurboLinuxを搭載 HF-W25F/LX

URL　http://www.hitachi.co.jp/Div/omika/prdcts/com/com.htm

発売　株式会社日立製作所
TEL　03-5295-5143
価格　オープン価格

日立製作所は、FAパソコンでは初めてLinuxを搭載した「HF-W25F/LX」を12月1日から発売した。

CPUにCeleron 566MHz、64MバイトECCメモリ、10GバイトIDEハードディスク、フロッピードライブを装備する。拡張スロットには、PCI（ハーフ）×2、PCI/ISA（ロング）×3、ISA（ロング）×1の合計6スロットを備えている。

製品は発売開始から3年間の長期安定供給が行われる。24時間連続稼動、使用期間10年を想定した高信頼性を備えており、保守サービスとして、納入後最大10年間の保守契約が可能。

OSは、TurboLinux Workstation日本語版6.0リミテッドエディション、またはTurboLinux Server日本語版6.1を搭載する。

Linuxに強力なRAS機能や障害時メモリダンプ機能、OSトレース機能を搭載し、障害解析が容易になっている。

Hardware　発売日　2000年10月30日

PCサーバPRIMERGYに1Uラックマウントサーバを追加 PRIMERGY TS120、PRIMERGY TS220

URL　http://www.fujitsu.co.jp/hypertext/granpower/

発売　富士通株式会社
TEL　03-3216-7976
価格　29万5000円〜

富士通は、1Uの薄型ラックマウントサーバ「PRIMERGY TS120」および「PRIMERGY TS220」と、デスクサイドまたは4Uサイズの2WAYエントリーサーバ「PRIMERGY ES320」を10月30日より発売した。

TS120は、PentiumIII933MHz／800EBMHzを搭載し、30.7GバイトのIDEハードディスクを搭載している。TS220は、デュアルCPUに対応可能で、9.1GバイトUltra160 SCSIハードディスク（ホットプラグ対応）を搭載している。ES320は、PentiumIII1GHz／933MHz／800EBMHzのデュアルCPUに対応可能で、ホットプラグ対応ハードディスクベイを5台分用意している。価格は、TS120が29万5000円から、TS220が43万8000円から、ES320が63万円からとなっている。

TurboLinux Server日本語版6.1のインストール代行サービスをバンドルした「Linuxサービスバンドルタイプ」が用意されている。

Hardware　発売日　2000年12月1日

ハードディスク3基搭載可能な1Uラックマウントサーバ hp netserver lp1000r

URL　http://www.jpn.hp.com/go/netserver/

発売　日本ヒューレット・パッカード株式会社
TEL　03-5344-7181
価格　42万8000円〜

日本HPは、1U（高さ44.45mm）の超薄型ラックマウント専用サーバ「hp netserver lp1000r（エルピー1000アール）」シリーズを、12月1日から発売した。出荷開始は2001年1月上旬の予定。

CPUはPentiumIII866MHz／1GHzのデュアルプロセッサに対応し、メモリは標準で256Mバイト、最大4Gバイトまで増設が可能。ハードディスクは、RAIDにも対応可能なホットスワップハードディスクベイを3基内蔵し、合計で最大108Gバイトまで拡張できる。

OSは、Red Hat LinuxとTurboLinuxに対応する。組込み式リモートアシスタント「Top Tools Agent」と「Top Tools RMC support」により遠隔地から集中管理が行える。

価格は、PentiumIII866MHz、256Mバイトメモリ、ハードディスクなしの構成で、42万8000円から。

Hardware　発売日　2000年12月1日

ネットワーク対応1Uサイズラックマウント UPS BIROS-Jupiterシリーズ

URL　http://www.gsee.co.jp/

発売　ジーエス・イーイー株式会社
TEL　03-3502-6554
価格　7万9800円

GSEEから、1Uサイズ19インチラックマウントタイプのUPS（無停電電源装置）「BIROS-Jupiter（500VAモデル）」を12月1日より発売した。

内蔵バッテリは、容量が500VA/300Wで長寿命タイプを採用している。寿命判定機能によって警告するため、バッテリ切れを未然に防ぐことができ、ラックに取り付けたままで前面パネルから交換が可能になっている。

常時商用給電方式で、バックアップ時間は4分、重量は7kgである。

10BASE-Tネットワークを標準搭載しており、Webサーバ機能、SNMPエージェント機能を備えている。WebブラウザやSNMPマネージャソフトを使用して、UPSをリモート管理できる。また、専用のパワーマネージメントソフト「パワーバイザv3」とオプションパーツを利用することで、クラスタサーバなどのネットワークシステム全体をサポートすることが可能である。

Hardware 発売日 2000年11月8日

デュアルCPU対応1Uラックマウントサーバ Express5800/120Ra-1

URL http://www.express.nec.co.jp/

発売 日本電気株式会社
TEL 03-3455-5800
価格 63万円〜

NEC（NECソリューションズ）は、1Uサイズの薄型ラックマウントケースにPentiumIIIを2基搭載可能なサーバ「Express5800/120Ra-1」を11月8日より発売した。

チップセットにServerSetIIIを採用し、FSB 133MHzで動作するPentiumIII800EB／933MHzを2基まで搭載可能。メモリは128Mバイト〜4Gバイト、ハードディスクはSCSIタイプで最大36.3Gバイトを2台搭載可能。オプションでRAIDにも対応する。64ビットPCIスロットを2スロット装備して優れた拡張性を実現した。10/100BASE-Tネットワークインターフェイスを2ポート搭載している。シリアルポートは前面にも用意され、保守の際の利便性を向上させている。

価格は、PentiumIII800EBMHz、128Mバイトメモリ、ディスクレスの構成で63万円から。

Hardware 発売日 2000年11月24日

インターネットアプライアンスサーバを拡充 Express5800/MailWebServer、CacheServer（Lite）

URL http://www.express.nec.co.jp/

発売 日本電気株式会社
TEL 03-3455-5800
価格 45万8000円〜

NEC（NECソリューションズ）は、ISP／ASP向けの1Uサイズインターネットアプライアンスサーバ「Express5800/MailWebServer」と「Express5800/CacheServer（Lite）」を、11月24日より発売した。

同MailWebServerは、Webサーバとメールサーバ機能を1台に統合した製品で、バーチャルホスティング機能を使用することで、複数の仮想Webサーバを構築可能。CPUにCeleron 667MHz、メモリ128Mバイト、20Gバイトハードディスクを2台装備しソフトウェアRAIDでミラーリングしている。価格は45万8000円から。

同CacheServer（Lite）は、インターネットのWebアクセスを高速化する製品で、従来の同CacheServerが搭載していたソフト「Novell Internet Caching System」を、別のソフト（squid）に変更することで、価格を約26％値下げした。CPUにPentiumIII850MHz、メモリ256Mバイト、20Gバイトハードディスクを2台装備している。価格は72万5000円から。

Software 発売日 2000年12月20日

マルチプラットホーム対応電源管理ソフト FULLBACK Manager Pro

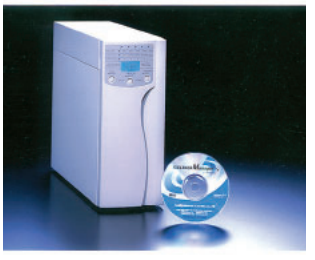
URL http://www.sanken-ele.co.jp/

発売 サンケン電気株式会社
TEL 03-3986-6701
価格 1万5000円

サンケン電気は、システムの電源を一括して統合管理できる電源管理ソフトウェア「FULLBACK Manager Pro」を発売する。Windows版は12月20日から、UNIX版は2001年2月中旬より出荷開始する。Linux版は2月上旬より同社のホームページから無償公開の予定。

FULLBACK Manager Proは、同社製のUPS「FULLBACKシリーズ」と併用することで、異なったOSを使用している複数台のコンピュータをシャットダウンしたり、ネットワーク上から監視／制御することができる。

OSをシャットダウンする前に、アプリケーションのファイル保存を行い終了させることや、ユーザーが指定したコマンドファイルを実行することが可能。



Software 発売日 2000年11月1日

高機能Javaアプリケーションサーバ Lutris Enhydra Professional

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

発売 NECソフト株式会社
TEL 03-5569-3399
価格 7万円

NECソフトは、米Lutris社のアプリケーションサーバ「Lutris Enhydra」の日本国内での販売とサポートを開始した。同ソフトはオープンソースで開発されている「Enhydra」の製品版で、Lutris Enhydra 3.0 Professional（英語版）は、11月1日出荷開始で、7万円。

Lutris Enhydraは、プラットフォームに依存せずJava2に完全対応した、生産性の高い開発環境を備えたアプリケーションサーバで、e-ビジネス向けのインターネットWebサイトを効率よく構築することができる。開発環境として、アプリケーションウィザードやXMLC（XMLコンパイラ）が用意されている。Webサーバ機能として、ロードバランシング、フェイルオーバーなどのクラスタリング機能や、管理コンソール機能などを備えている。

JavaアプリケーションからJDBCによって、Oracle、SQL Server、Informix、Sybase、PostgreSQLデータベースに接続できる。

News Headline from 日刊アスキーLinux

www.Linux24.com

待望の「Netscape 6」正式リリース、日本語版もダウンロード可能

2000年11月16日

米America Online（AOL）の子会社、米Netscape Communicationsは、新しいレイアウトエンジン「Gecko」を搭載した最新型Webブラウザ「Netscape 6」の各国語版を米国、日本、ドイツ、フランス、イギリスにおいて同時リリースした。

HTML4.0、XML1.0、CSS1などをフルサポートするレイアウトエンジン「Gecko」を搭載し、多言語文字データ（Unicode3.0）などもサポートする。また、ブラウザの表示速度も高速化。特に、今までNetscape Navigatorの弱点とされていたテーブルレンダリングの速度が向上した。

ブラウザ画面の左サイドをパーソナルスペースとして利用可能になり、国内外のサイトと連携した600ものメニューを使用して、最新ニュースや株式ポートフォリオなどを表示できる。コンテンツには、「CNN」「Asahi.com」「eBay」「日刊スポーツ」などが用意されている。

URL入力フィールドにキーワードを入力することでの検索が可能になり、検索の作業効率が向上した。また検索結果の表示には、同社の「Smart Browsing」サービスと、「First Page Results」システムを組み合わせ、最初のページに正確で関連性の高い情報を表示するようになっている。各種サイトへのログイン名とパスワードをすべて記憶しユーザーの入力の手間を軽減する「Password Manager」（どのログイン名とパスワードを記憶させるかはユーザーが管理可能）や、Cookieの記録の表示・削除が可能な「Cookie Manager」などを搭載する。

サイトの翻訳機能を搭載し、メニューバーから[表示]－[翻訳]を選択すると日英／英日翻訳などを行う。

「Netscape 6」ダウンロードページ
(http://home.netscape.com/ja/download/)

メルコがLinux端末事業を開始

2000年11月15日

メルコは11月15日、Linuxを搭載したインターネット端末事業を展開すると発表した。インターネットのサービス（たとえばインターネットモールやISP、SIベンダーなど）を行う企業を対象に、システム構築のためのコンサルティング、インターネット端末の提供、実際のシステム構築などを提供する。

Linuxの技術面においては、組み込みLinuxを扱う台湾のIST（Internet Solutions Technology Corporation）と提携するとともに資本出資し、協調を図っていく。資本出資は、ISTの総株式の10%を50万ドルで取得する。

提携の具体的な内容は、ISTのソフトウェアやソリューションの国内における独占的販売権（サーバ製品を近日発表）、ISTのソフトウェアのソースコード取得、ISTからメルコに対する技術教育の3点となっている。

ISTからどのように技術が提供されるかについては具体的な説明はなかった。ただし、組み込みLinux上で動作させるアプリケーションは、自社開発ではなく、Netscape製品などを想定しているという。

メルコ（http://www.melcoinc.co.jp）

「デスクトップとしてのLinux」を強烈に打ち出すGNOME用ファイルマネージャ「Nautilus PR2」

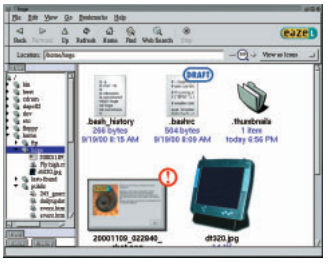
2000年11月10日

米Eazelは、GNOME向けファイルマネージャの最新版「Nautilus Preview Release 2」をリリースした。Eazelは、Linuxをデスクトップとしての選択肢に引き上げることを目的として1999年に設立されたベンチャー企業で、Macintoshのユーザーインターフェイスの設計を手がけた元Apple社員たちが設立者に名を連ねている。

NautilusはまだPreview Releaseの段階にすぎないが、すでに「一般的な」ユーザーが必要とする機能をほぼ備えている。その特徴は、アンチエイリアスされた文字やスタイリッシュなデザイン、ドラッグ&ドロップを活用した直感的な操作、サムネイルとして働く拡大可能なアイコンなどだ。

Nautilusの画面構成は、WindowsのExplorerと同じ2ペイン。左側のペインにはディレイトリのツリー表示のほか、タブを選択することによりヒストリやメモを参照することができる。右側には、選択したディレクトリのファイルが表示される。アイコンはファイルのタイプだけではなく、内容まで示している。たとえばテキストファイルなら最初の数行が、画像ファイルならそのサムネイルがアイコンになっている。

米Eazel（http://www.eazel.com/）



ぷらっとホーム開発のインターネット端末、Linuxを採用

2000年11月9日

ぷらっとホームは、11月販売予定のインターネット端末「DTシリーズ」に、インフォマティックジャパンの組み込みLinux「JNT」を採用した。DTシリーズは、PCは使いたくないがインターネットは利用したいユーザーのためのインターネット専用端末で、モバイル／モニタレス／デスクトップの3種類から構成される。

DTシリーズは、シンプルな操作性をユーザーに提供する。ハードディスクレスのため、トラブルや騒音がなく、終了後はワンボタンでシャットダウンできる。

DT320は、Webブラウジングに機能を絞ったモバイルタイプの端末で、ワイヤレスでインターネットに接続することが可能。DT500は、10.4インチ液晶ディスプレイを装備した省スペース設計のデスクトップタイプの端末、DT240は、モニタをすでに所有しているユーザー向けの

モニタレスタイプの端末。

インフォマティックジャパンは、ドイツInfomatecの日本法人で、インターネットにおけるLinux／UNIX技術を利用したソリューションを提供している。

ぷらっとホーム
(http://www.plathome.co.jp/)
インフォマティックジャパン
(http://www.infomatec.co.jp/)

OADG、“Linux分科会”を発足
2000年11月4日

PCオープン・アーキテクチャー推進協議会（以下、OADG）は、日本におけるLinux普及促進を目的とした“Linux分科会”を10月23日付で設置した。

Linux分科会は、OADGにLinuxディストリビューション会社を加えたメンバーで、Linuxビジネスの普及拡大、Linuxを安心して使える環境の整備、オープンソース環境への対応を目標に活動を行う。

普及活動計画として、“ソリューション・カタログ”の作成、Webサイトを利用したLinux関連情報の公開、OADG会員およびLinux関係者向けの無料ワークショップ／セミナーの開催、LPI等との協力による認定エンジニアの支援を行う。

また、環境整備のために、Linuxディストリビューターや PCメーカーなどと協力し、ハード／ソフトのLinux環境における互換性の検証と動作保証の統一基準の作成、事務局内に設置した開発テスト環境の提供、Linuxアプリケーション開発ガイドの作成や、Linux国際化への支援を行う。

さらに、プリンタやディスプレイ、サウンドカード、周辺機器インターフェイスなどのドライバを、メーカーと共同で開発し、GPLに基づくオープンソースとして2001年後半に公開するとしている。

OADG（http://www.oadg.or.jp/）

ローソン全7600店に、1万5000台のLinuxマシンが登場
2000年10月30日

日本IBMは、コンビニエンスストア「ローソン」店頭に設置された無人端末「Loppi（ロッピー）」のバックエンドサーバとして、Linuxを採用した。

Loppiは、全国のローソン7600店舗に設置された無人端末で、チケットやギフト商品販売などを行っている。日本IBMは、Loppiのほかにもコンビニエンスストアのマルチメディアステーションを手がけており、従来のLoppiもIBM製だ。

今回このLoppiのバックエンドを司るPCサーバ「eServer xSeries」を、Linuxで運用する。このバックエンドサーバは、ローソン各店舗にメインとバックアップ用途で2台ずつ配置されるので、7600店舗×2台＝1万5000台のLinuxサーバが全国のローソンに配置されるというわけだ。

バックエンドを強化したLoppiでは、デジタルカメラデータのDPEプリントサービス、音楽、ゲーム配信などコンテンツ配信機能の強化、インターネットや衛星経由のタイムリーな情報提供などを行っていく。

日本IBM（http://www.ibm.co.jp/）
ローソン（http://www.lawson.co.jp/）

注目を浴びるLinux用新ファイルシステム「Tux2」
2000年10月28日

また1つ、Linuxの次世代ファイルシステムが登場する。名前は「Tux2」。まだコードは一切リリースされていないものの、アルゴリズムが目新しいためにSlashdotなどで話題になっているものだ。

fsckのいらないファイルシステム、ReiserFSは、キャッシュをディスクに書き込む前にログを保存しておいて、もしクラッシュしても、ログを「リプレイ」することで一貫性を保つことができる。この機能は「ジャーナリング」と呼ばれ、商用UNIXでも広く使われている。

Tux2の仕組みはジャーナリングファイルシステムとは異なっている。用いているのは、「Phase Tree」と呼ばれる新しいアルゴリズムである。

fsckのいらないファイルシステム、Phase Treeは、3つのツリーでファイルシステムを構成、書き込み専用のツリーに対してデータを保存し、すべてのデータを書き込んだところで一度に（アトミックな操作で）きれいな状態のツリーに「遷移」させる。この方法では、書き込み中の状態というのが在存しないので、どのタイミングでも一貫性が保たれている。つまりどのタイミングでクラッシュしても、ファイルシステムを修復する必要がないということだ。

Tux2のもうひとつの特徴は、既存のext2との親和性が高いことである。ext2を“Tux2”としてマウントするだけでファイルシステムを自動的に変換、Tux2として使用することができるという。これはReiserFSにはない大きな利点だ。

そのほかにTux2は、UNIXのセマンティクスをできるだけ維持したまま（Macのリソースフォークや Windows NTのストリームのような）ファイルにメタ情報を持たせるための仕組みや、1ブロックに満たないファイルの末尾をまとめて収納することでディスクの使用効率を高めるアルゴリズムの導入なども、目標に入れているという。

Tux2の公開の時期は未定。いまはオリジナル開発者のDaniel Phillipsが未公開のまま開発を進めている段階だが、コードはGNU GPL（General Public License）でリリースされる予定だ。

Tux2ホームページ
(http://innominate.org/~phillips/tux2/)

KDE 2.0「Kopernicus」リリース
2000年10月25日

KDEプロジェクトは10月23日、次期デスクトップ環境「KDE 2.0」をリリースした。KDEは、GNOMEと並んで広く利用されているUNIX向けデスクトップ環境で、UNIXにWindows並みかそれ以上の操作性を与えるべく開発されている。主要なディストリビューションはすべて、GNOMEとKDEの両方を採用している。

KDE 2.0は、WindowsのExplorerと同じくWebブラウザとファイルマネージャの機能をあわせ持つ「Konqueror」や、オフィススイート「KOffice」、新しい開発ツールなどを備える。操作性も1.xから進歩しており、ルックアンドフィールをテーマでカスタマイズ可能になった。

KDEプロジェクト（http://www.kde.org/）

IBMのオープンソース戦略
Linux開発コミュニティとの連携

オープンソースへの流れはソフトウェア開発の手法に新たなパラダイムをもたらしつつある。PCからメインフレームまでを擁するIBMはこの動きを敏感に察知し、Linuxの開発コミュニティと連携を取るべく活動を始めている。今回は、IBMのLinux関連ソフトウェアの開発を行っているIBM Linux Technology CenterのDaniel D. Frye氏に、IBMのオープンソース戦略について話を聞いた。

Interview

――IBM Linux Technology Centerの基本戦略から聞かせてください。

Frye:インターネットが既存のネットワークを統合したように、Linuxはアプリケーションを統合すると考えています。すでに、eビジネスのアプリケーションでは、Linuxがリファレンスプラットフォームになっています。私たちのねらいは、あらゆるところにLinuxを導入し、この動きを加速することなのです。

Linuxは、Webサーバ、ファイル／プリントサーバなどの用途に対応できる十分な能力を備えています。私たちは、このような用途でのソリューションを提供するため、すべてのハードウェア、ソフトウェア、サービスをLinuxに対応させています。ハードウェアについては、PCサーバからメインフレームまで、幅広いプラットフォームでLinuxが動いています。そして、データベース、Webサービス、システム管理などのミドルウェアもLinuxで動作しています。サービスに関しても、プロフェッショナルサービス、ワールドワイドサービス、教育とトレーニングを提供しています。

ただ、現状のLinuxは、エンタープライズレベルのOSとしては不十分なところがあります。そこで、Linuxをエンタープライズレベルの OSにするために、オープンソースコミュニティと共に作業を行うのが私たちの戦略です。したがって、私のチームの仕事は、Linuxコミュニティと共にLinuxをより良いものにしていくことなのです。今後2年ほどで、Linuxはエンタープライズレベルの OSとして、より広範な用途で使えるようになると考えています。

私たちの戦略はシンプルです。Linuxは、現在でもいろいろな用途に対応する能力を持ちますが、さらに広い用途に対応できるように拡張しよう、私たちがそれを手伝おうということなのです。

――IBM Linux Technology Centerの概要を聞かせてください。

Frye:IBM Linux Technology Centerとは、オープンソース開発をコミュニティと共に行うIBMエンジニアの集まりです。ここでは、ネットワーク、ファイルシステム、システム管理に関する開発を行っています。スケーラビリティ（拡張性）、標準化など、OSとして必要なすべての要素について、Linuxに向上させるべき点があれば、コミュニティと共に作業を行います。

IBM Linux Technology Centerのチームは、日本を含めた世界6カ国に配置されています。日本のほか、インド、アメリカ、イスラエル、ドイツ、イギリスです。日本のチームは国際化に焦点を絞り、開発を進めています。Linuxがエンタープライズレベルに成熟するためには、国際化が重要です。私たちは、コミュニティと共にオープンソースで開発を行っていきます。

――日本で行われている国際化とは、韓国語や中国語のような文字コードへの対応などを含めたものでしょうか。

Frye:はい。このチームが行っているのは、世界中の言語をサポートするためのコア部分の開発です。この中には2バイト、マルチバイトの文字コードも含まれています。

――コミュニティと共同で開発を行うのですか？

Frye:そうです、共同で開発を行います。多くの場合は既存の開発コミュニティに我々のチームが参加することになります。ときにはIBMの技術を提供することもあるでしょう。この場合は、まずその技術をオープンソース化してから、コミュニティと共有する形となります。基本的に、私たちの内部だけで作業することはありません。すべてをコミュニティと共同で行います。

――IBMのハードウェア製品に関するLinuxサポートなども行うのですか？

Frye:特定のハードウェアに関するサポートはハードウェアグループが行います。これら、他のグループとも連携をとりながら仕事を進めていますが、私のグループは、すべてのLinuxユーザーのための仕事に注力しています。とはいえ、研究開発レベルではなく、製品レベルの仕事をしていると認識しています。

――LinuxをメインフレームのS/390へ移植したのはLinux Technology Centerなのでしょうか？

Frye:これはLinux Technology Centerの仕事とはいえません。実をいうと、この作

業はとても簡単でしたので、夏期のみ働いているドイツの実習生たちが数週間で行いました。これはLinuxの設計デザインがすばらしいことを実証しているといえるでしょう。従って、私たちが手を出す前に完了していたんです。

――現在Linux Technology Centerで話題となっている分野は何ですか？

Frye:ストレージとI/O、ファイルシステム、スケーラビリティ、パフォーマンスの向上がホットな分野ですね。

――スケーラビリティの向上とは、具体的にはどのようなものですか？

Frye:スケーラビリティには、垂直と水平2つの方向があります。ひとつは、SMPによる垂直方向への向上、もうひとつはクラスタリングによる水平方向の向上です。

――Linuxカーネル2.4ではSMP性能が向上していますが、さらなるスケーラビリティの向上をねらい、研究をしているというわけですね。

Frye:そうです。私たちはまだLinuxの開発コミュニティにおけるリーダーとはいえませんが、カーネル2.4においてもスケーラビリティを向上させる小さな修正をいくつか提供しています。今後のカーネルについてもスケーラビリティ向上に向け作業を進めていきます。

クラスタについては、クラスタインストールに関するチームとクラスタマネージメント技術のチームを構成しています。これらのチームによる成果をコミュニティにもたらしたいと考えています。クラスタインストールのチームは、VA Linux、Red Hatと共に、クラスタインストールの手順を標準化しようとしています。エンタープライズ化するにあたって、標準化されたソリューションが必要になるのです。

――ファイルシステムについてはどうでしょう？　IBMのJFSは？

Frye: Linuxにはジャーナル機能を持つ4種類のファイルシステムが準備されつつあります。どれも良いファイルシステムですが、それぞれに得手不得手があります。通常、コミュニティではこれらすべてを実際に動作させ、そのうちの1つを採用するという手法を取ります。従って、カーネル2.4の次にリリースされるLinuxカーネルでは、標準的に搭載されるジャーナルファイルシステムは1種類になっていると思います。もちろん、IBMのJFSが採用されれば嬉しいのですが、それは、あまり本質的なことではないと思っています。私たちIBMとしては、どのファイルシステムがカーネルに含まれるようになってもサポートしていきます。

――ビジネスの話になりますが、IBMではLinuxをどう位置づけていますか？ WindowsなどのOSとの関係はどうなるでしょう？

Frye:共存すると思います。お使いになるOSは、お客様に選んでいただくということになるでしょう。ですから、Windowsソリューションで最高のものが欲しければIBMへどうぞ、Linuxソリューションで最高のものが欲しくてもIBMへどうぞということになります。

――開発を含め、Linuxディストリビューターとの協業は行いますか？

Frye:私たちは、Red Hat、Caldera Systems、SuSE、TurboLinuxの4社と強いパートナーシップを結んでいます。たとえば、PCサーバ「eServer xSeries」ではこれら4社のLinuxディストリビューションを利用できます。また、開発などにおいてもパートナーシップを結んでいます。私たちは、IBM単体で流通を行うというよりも、これらのパートナーと最高の関係を続けていくという戦略をとっています。

――日本では、主にサーバ用途でLinuxが普及しつつあります。今後はデスクトップOSとしても普及が進むとお考えですか。日本より先行していると思われるアメリカではどうでしょうか。

Frye:そうですね、現状のLinuxはサーバ用途において、より成熟したOSだと思います。しかし、デスクトップ用途においても成熟しつつあります。また、Linuxについては、必ずしもアメリカが先頭を切っているとはいえないでしょう。Linuxの普及は世界的な現象なので、どの地域が進んでいるということもないと思います。

――デスクトップ用途で使うとなると、オフィススイートなどのアプリケーションソフトウェアが必要になりますが、IBMからこれらを提供するということはありますか？

Frye:それはありません。オフィスアプリケーションについては、ソフトウェアプロバイダなどの会社とパートナーシップを結び、それらの会社から提供する戦略をとっています。

――今後のIBM Linux Technology Centerは？

Frye: Linuxは、これまで存在したどのOSよりもはるかに速い速度で成熟を続けていくと考えています。そして、将来的にはLinuxで対応できない分野はまったくなくなるでしょう。IBM Linux Technology Centerは、Linuxのコミュニティにおいて、信頼できる仲間として認められることを目標にしています。

IBM Linux Technology Center Director Daniel D. Frye氏

LinuxWorld Conference & Demo/Tokyo 2000

5月のLinuxWorld Expo Tokyo/2000に続き、LinuxWorld Conference & Demo/Tokyo 2000が開催された。2日間で1万8000人以上の来場者があった展示会場のようすをお伝えしよう。

10月31日、11月1日の2日間にわたって、LinuxWorld Conference & Demo/Tokyo 2000が東京ファッションタウンで開催された。「少し前にもあったのでは?」と思われる方もいるだろうが、今年の5月11、12日に開催されたのは、LinuxWorld Expo Tokyo/2000である。

主催者のIDGジャパンによれば、Linuxビジネスの隆盛に伴い、今後は、初夏にLinuxWorld Expo、秋にLinux World Conference & Demoと題して、年2回の定期開催になるそうだ。

大企業の参加がさらに増える

昨年9月にも同じ会場でLinuxWolrd Expo Tokyo/'99が行われたが、今回は展示スペースが約1.5倍に拡大されていた。広くなった分、大手ハードウェアベンダーやディストリビューターの出展スペースが大規模になっているのが目立っていた。

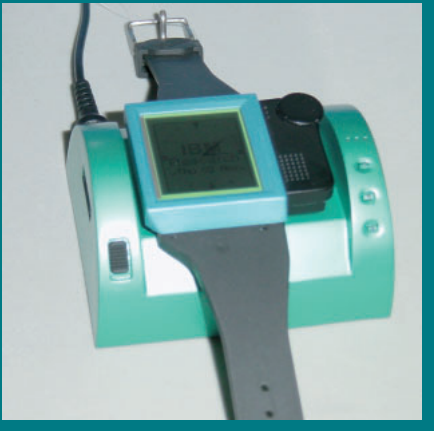
中でも日本IBMは、同社のメインフレームS/390上で動くTurboLinuxや、ラインアップを一新したばかりのeServerシリーズを全面に押し出していた。基調講演も同社のパーソナル・システム事業部長である、堀田一英氏が行っており、IBMのLinuxビジネスへの取り組みの本気さが感じられた。

また、8月下旬に米国で公開された「Linuxウォッチ」も日本初公開され、注目を集めていた。見た目は大きめの腕時計だが、Linuxカーネル2.2、X Window Systemが搭載されているという。説明員によれば、実用的かどうかは別として、Linuxウォッチ上でWebサーバを動作させることも可能ということだ。

上から下までLinux

Linuxの得意分野であるサーバでは、インテルのItanium（アイテニアム）を搭載したハイエンドサーバが日立、NECから出展されていた。Itaniumは、現行のx86とは異なるアーキテクチャの64ビットCPUで、ここ数年「もうすぐ出る」と言われ続けていたものだ。

なかでもNECは、Itaniumを16個搭載したAzusA（あずさ）で大規模科学計算のビジュアル化のデモを行っており、64ビットCPUを複数用いた性能を



本邦初公開のLinuxウォッチ。こういう製品は日本企業のお家芸だったはずだが、IBMもがんばっているようだ。

誇示していた。

Itaniumのようなハイエンドだけでなく、一般的なx86ベースのサーバ、さらにはSOHO、個人宅をターゲットにした小型のサーバ／ルータも多数発表されていた。

SOHO向けのサーバ／ルータとしては、おなじみの子羊ルータ（ワイルドラボ）、PathNavigator（サイバネテック）、blue grass、white neon（アクアリウム）などに加えて、アクアリウムの新型機green tetra（仮称）や12C SolutionsのiCOMサーバといった新顔の製品も展示されていた。

これらの製品は、比較的低価格でSOHOや個人宅に必要十分な機能を提供するものである。また、いずれもスタイリッシュな外観を持っており、個人の常時接続が一般的になってくれば、ますます普及していくと思われる。

ハイエンドサーバからSOHO／家庭用の超小型サーバまで、同じOSが使えるのは、ソースが公開されており、カスタマイズが自在に行えるオープンソースならではの強みと言えるだろう。

クライアント用途も着々

サーバ用途だけでなく、クライアントマシン用OSとしてのLinux環境も着々と整いつつある。

オフィススイートとしては、Applix ware Office for Linux 5.0やHancom Office 1.0のベータ版が展示されており、日本語表示も問題なく行えていた。ただ、どちらも必要なリソース量は大きく、展示されていたマシンのメモリ（64Mバイト）ではスワップが多発していた。「少々古いマシンでも軽々動く」というわけにはいかないようだ。

ソフトウェアDVDプレーヤのWinDVDを販売しているインタービデオ社からは、Linux用のLinDVDが出展されていた。DVD再生補助機能を持ったSiS630チップセットと、Celeron 500MHzを用いたマシンでDVDの再生が行われていたが、現状ではわずかにコマ落ちやノイズが見られた。

富士通では、発表されたばかりの小型ノートPCであるLOOXに、Linuxをインストールしたものを展示していた。LOOXは、Linus Torvalds氏が所属していることで知られている、Transmeta社のCrusoeを搭載している。CrusoeはLongRunと呼ばれる機能を用いて、電力消費を低減しているが、現状ではLongRunはWindows Meでのみ利用可能なようだ。

定着した「Linuxでビジネス」

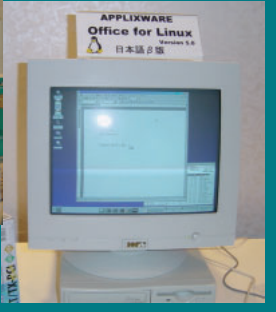
「Linuxのブームは終わった」という言葉を聞くことがあるが、実際そのとおりであろう。もはやブームではなく、IT産業従事者にとっては、必須科目といってもいいだろう。「Linuxわかりません」では通用しないのだ。会場を埋めたビジネスマンたちを見て、そのような印象を強く受けた。

SOHO向け小型サーバ群。必要十分な性能とカラフルな外観で、人気上昇中だ。

16 way Itaniumサーバ、AzusA。後継機は「AzusA 2号」になるのだろうか。

Linux用DVDプレーヤ、LinDVD。現状では画面にノイズが残るなど、完成まであと一歩という状態だ。

Crusoeを搭載したノートPC、富士通LOOXにインストールされたLinux。省電力機能には、まだ未対応のようだ。



Linux用オフィススイートのHancomOfficeとApplix Ware for Linux。

Distribution

新着ディストリビューション

Linux MLD 5

Windowsアプリケーションとして簡単に起動できるLinux MLD 5は、WindowsのFAT領域にインストールして使うディストリビューションだ。インストール作業は他のWindowsアプリケーション並の容易さだ。ハードウェア情報やネットワーク情報をWindowsマシンから流用するので、ユーザーはわずらわしい設定から開放される。多くのディストリビューションの中でも異色の存在だ。

Vine Linux 2.1CR

完成度の高さはトップクラス。日本語ディストリビューションの雄Vine Linuxの最新版、Vine Linux 2.1がリリースされた。安全性と設定の容易さのポリシーをどこまでも追求し、細かい仕様が変更されている。PPC、SPARC、Alpha版もリリースされ、異なるプラットフォームでも同一の操作環境を実現した。

Official Red Hat Linux 7J改訂版

これで安心？ 改訂版のRed Hat Linuxは、約3週間でシステムが停止するというOfficial Red Hat Linux 7Jの致命的なバグをはじめ、細かい不具合が修正された。

Linux MLD 5

設定の容易さを追求するディストリビューションLinux MLDの最新版、Linux MLD 5（以下、MLD 5）がメディアラボから12月15日に発売される。

表1のように、日本語入力プログラムWnn6と5書体のDynaFontの商用ソフトなどを収録し、9800円で販売される。

収録物	概要
インストールCD	FAT用とext2用の2枚を収録
フリーソフトCD	追加パッケージを収録
ソースCD	パッケージのソースRPMを収録
マニュアル	約200ページの分量でインストール方法から活用法までを解説
Wnn6 Ver.3	UNIX環境で定番の日本語入力プログラム
DynaFont 5書体	日本語表示のための商用フォント

表1　Linux MLD 5の収録物
収録物はCD-ROMとマニュアルというベーシックな構成だ。ext2領域用のインストールCDも収録される。

全自動インストールで設定不要

MLD 5の特徴は、なんといってもボタンワンクリックで終わるインストールだ。これは、LinuxをWindowsのFAT16/32ファイルシステム上へイメージファイルとしてインストールするのが理由で、MLD 5はWindowsアプリケーションのひとつとして扱われる。

WindowsマシンにCD-ROMをセットするとインストーラが立ち上がり、Linux領域のサイズを指定して[OK]ボタンを押せば、K6-2/350MHzマシンだと約10分でインストールが終了する。

MLD 5はWindowsの持つ設定情報を利用する。たとえば、インストール先のWindowsマシンではすでにネットーワークが設定されているとする（**画面1**）。するとMLD 5は、この設定情報を引き出してLinuxのネットワークを設定するというわけだ（**画面1**）。ビデオカードやネットワークカードといったハードウェア情報もWindowsから流用するので、ユーザーはLinux導入時の基本的な設定が不要だ。

また、メディアラボ独自のxconfというツールを使えば解像度の設定も含めて簡単にXを設定できる。Linux初心者がつまずきがちな設定項目は、ほぼすべてカバーされているというわけだ。

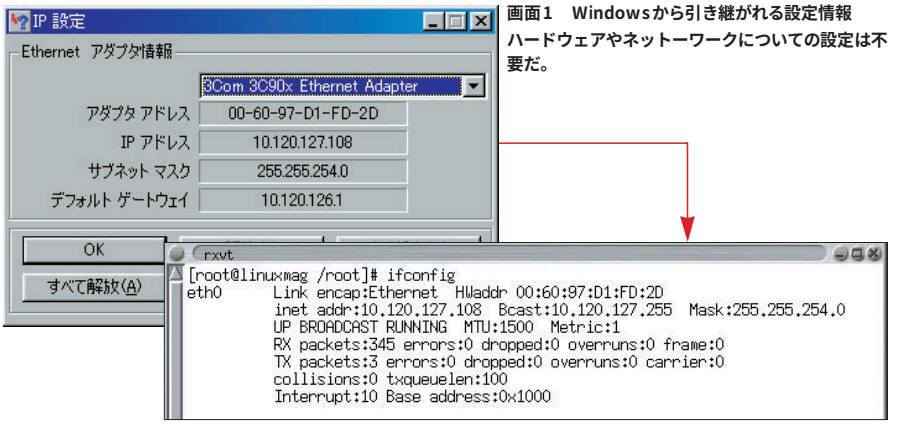
カーネル2.4を採用

優しい顔をしたインストーラの裏には最新の基本システムが隠れている。Linux MLD 5は標準でカーネル2.4のtest8、XFree86 4.0.1、Cライブラリのglibc2.2.1、RPM 4.0を採用している。リリース版でカーネル2.4とglibc2.2を使うのは現時点でMLD 5だけだ。

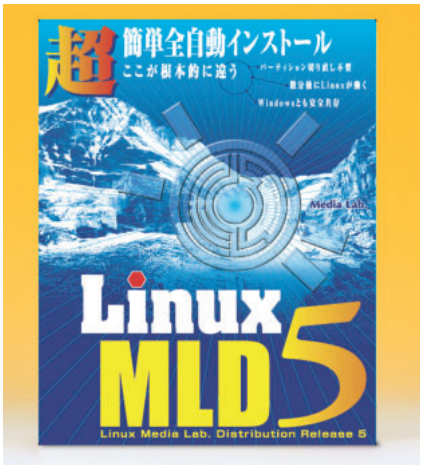
これら基本システムの最新バージョン収録によって、各種USBデバイスの利用、ハードウェアアクセラレーションを利用した高速な3Dグラフィックス表示が可能になった。

また、カーネルはメディアラボによって、より多くのSCSIカード、サウンドカード、ビデオキャプチャカードなどを使えるように拡張されている。

システム設定もハードウェアの利用も、Windowsのように行いたいユーザーに特にお勧めのディストリビューションである。



画面1　Windowsから引き継がれる設定情報
ハードウェアやネットーワークについての設定は不要だ。



製品名　Linux MLD 5
価格　9800円
問い合わせ先　メディアラボ株式会社
03-5294-7255
http://www.mlb.co.jp/

Vine Linux 2.1CR

日本語ディストリビューションの雄であるVine Linuxの最新版Vine Linux 2.1（以下、Vine 2.1）がリリースされた。商用ソフトなどをバンドルした製品版Vine Linux 2.1CR（以下、Vine 2.1CR）は、レッドハットから11月9日より発売されている。

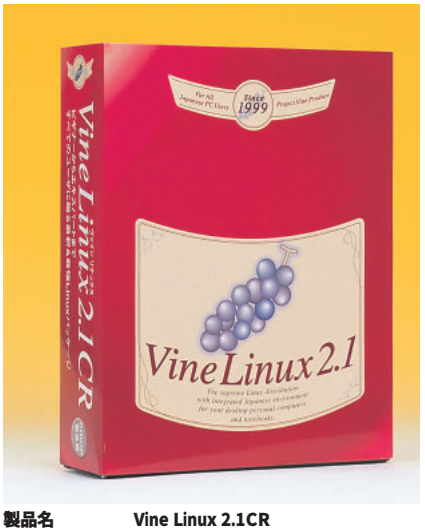
今回のバージョンアップでIntel版にPPC版、SPARC版、Alpha版の3つが新たに加わり、サポートされるプラットホームはIntel版と合わせて4つとなった（Alpha版は開発版扱い）。

Vine 2.1はRed Hat Linux 6.2をベースにしており、カーネルに2.2.17を、Cライブラリにglibc2.1.3を、XにXFree86 3.3.6を採用している。

製品版はレッドハットから

これまで技術評論社が行っていたVine Linux製品版の販売は、今回からレッドハットが担当する。これは今年7月に発表されたレッドハットとProject Vineとの間に敷かれた協力体制の一環である。レッドハットは製品版の販売とインストールサポートを担当するだけで、Vine Linuxの開発体制はこれまでと変わりない。

製品版には**表1**のとおり日本語入力プログラムのWnn6やダイナフォントといった商用ソフトなどが収録される。付属のユーザーガイドは約270ページの分量で、インストール解説に加えて、プリンタの設定、メーラの使用法などが解説されている。



製品名 Vine Linux 2.1CR
価格 1万2800円
問い合わせ先 レッドハット株式会社
03-3257-0411
http://www.redhat.com/jp/

Vineすみずみまで浸透

新しいVine 2.1CRには、Intelだけでなく、PPC、SPARC、Alphaといったプラットホーム用のインストールCDが付属する。

おウチで使わなくなったMac、研究室の隅でホコリをかぶっている古いSunワークステーション、はたまた大規模な数値計算用として現役バリバリのAlphaマシン、これらのマシンでVine Linuxを使えるなんてVineファンにはたまらない話だ。

ワークステーションでも作り込まれたVineの日本語環境を堪能しよう。

本誌付録CD-ROM収録のVine Linux 2.1はFTP版（インテルアーキテクチャ）です。非商用ソフトだけが含まれ、製品版を販売しているレッドハット株式会社からのサポートを受けることはできません。

安全軽快主義

Vine Linuxでは高機能よりも、安全性と設定の容易さを追求するという明確なポリシーに基づいたソフトウェアの選択が行われている。今回のバージョンアップで新たに収録されたパッケージ（**表2**）を見ても、そのポリシーが強く反映されている。

■w3m

w3mはktermやrxvtといったX上のターミナルのほか、コンソール上で動作するWebブラウザだ。日本人によって開発されており、日本語やフレームの表示に対応している。

■LFTP

LFTPはシェルライクなインターフェイスを持ち、コンソール上で動作する高機能なFTPクライアントだ。このツールは、リモートマシンでの使用中

収録物	概要
インストールCD	Intel版、PPC版、SPARC版、Alpha版の4枚を収録
ソースCD	収録されるソースパッケージは各プラットホームで共通
VinePlusCD	JGやKDEなどの追加パッケージを収録
アプリケーションCD	Hancom Officeと翻訳魂や、各種試用版の商用ソフトを収録
ブート用FD	インストーラ起動用のフロッピー（ノートPC用もあり）
ユーザーガイド	インストールから活用方法までを解説
Wnn6 Ver.3	UNIX環境で定番の日本語入力プログラム
DynaFont 5書体	日本語表示のための商用フォント
HancomWord	Microsoft Word互換を目指すオフィスアプリケーション
翻訳魂	和訳、英訳ソフト

表1　Vine Linux 2.1CRの収録物
製品版にはIntel用のほかにも、PPC、SPARC、Alpha用のインストールディスクが収録される。付属の商用ソフトは4種類だ。DynaFontはすべてのプラットホームで使用できるが、ほかの3つはIntel版のみとなる。

に回線が切断されても、自身のプロセスをバックグランドへ回し、ファイル転送をそのまま継続するという優れた機能を備えているほか、ファイル転送時の最大接続数を指定できたり、バッチ処理にも対応するなど、多くの機能をサポートする。

■Sylpheed

今までのLinuxに足らなかったもの、それはマウスで直感的に操作できるWindows環境でお馴染みのメーラだ。

UNIX系OSには、EmacsやXEmacsで使えるWanderlustやMewといった優れたツールが存在するが、いかんせんWindowsユーザーにとってはEmacsの操作が難しい。

Sylpheed（**画面1**）はGTK+をベースに開発されており、Windows環境のメーラのように直感的な操作が可能だ。POPのほかにNetNewsやIMAPも利用できる。

■Namazu

定番の日本語全文検索エンジンNamazuは最新のバージョン2.0が収録されている。Namazu 2.0は新たにMicrosoft WordとExcel、PDFなどをサポートして、その用途はさらに広がっている。NamazuはWebブラウザやTknamazu、あるいはEmacsなどをフロントエンドとして利用可能だ。

■ProFTPD

ProFTPDはWU-FTPDに代わって収録されたFTPサーバプログラムだ。WebサーバApacheと同じような感覚で設定できることを目指していて、ディレクトリごとのアクセスを制御するApacheの.htaccessに類似した.ftpaccessなどの設定ファイルがその例だ。

画面1　マウスでの直感的操作ができるSylpheed
マウスで直感的な操作ができるSylpheedは、Windows環境のメーラと比べても決してひけをとらない。POP形式のほかにIMAPやNetNewsにも対応している。

また、FTPデーモンはinetdデーモンから起動する以外にプログラム単体で起動できるので、さらなるパフォーマンスとセキュリティの向上が可能だ。

■Postfix

Postfixはメールサーバを構築するためのプログラムだ。現在多くのサイトで稼動しているSendmailに、メール配送の速さやセキュリティの高さでアドバンテージを持つとされている。Webminというツールを使えばWebブラウザからの設定も可能だ。

やはり完成度は高い

Vine 2.1は安定した環境を提供するだけでなく、先進的な機能も積極的に取り入れている。

たとえば、ReiserFSというデータ更新のログを逐一記録するジャーナリング機能を備えた先進的なファイルシステムの採用だ（動作についてはサポート範囲外）。

昨今では一般的になった数十Gバイトクラスの大容量ディスクを、現在Linux標準であるext2ファイルシステムで使用すると、不正にシャットダウンした際、再起動時に実行されるファイルシステムのチェックに、場合によっては数十分を要する。

ReiserFSはこの長い待ち時間を解消してくるので、サーバ管理者だけでなく、大容量ディスクを使う一般ユーザーにも有効な機能だ（ReiserFSの導入方法についてはインストールCDの/docs/secret.txt.bz2を参照）。

このほかにも、LBA32モードに対応したLILOの採用でディスクの先頭から8Gバイトを超える領域にあるカーネルの起動に対応したほか、usbmgrの採用でUSBデバイスの認識をサポートするなど、デスクトップ用としてもサーバ用途としても、完成度はさらに高くなっている。

初心者から上級者までお勧めできるディストリビューションだ。

ツール名	URL
w3m	http://ei5nazha.yz.yamagata-u.ac.jp/~aito/w3m/
LFTP	http://lftp.yar.ru/
ProFTPD	http://www.proftpd.net/
Postfix	http://www.postfix.org/
Sylpheed	http://sylpheed.good-day.net/

表2　Vine Linux 2.1で新たに採用されたツール
新たに採用されたソフトウェアは使いやすさと安全性を考慮したものが収録されている。

Official Red Hat Linux 7J改訂版

Official Red Hat Linux 7J（以下、Red Hat 7J）の不具合修正版となるOfficial Red Hat Linux 7J改訂版（Red Hat 7J改）が11月13日にリリースされた。

現時点ではRed Hat 7J改というパッケージが販売されているわけではなく、Red Hat 7Jの購入者へアップデートCDを配布するという形態だが、今後は改訂内容を取り込んだパッケージ販売が予定されている。

すでにRed Hat 7Jを購入しているユーザーは、レッドハットのFTPサイト（http://www.redhat.com/download/mirror.html）から改訂内容をダウンロードできるほか、1050円の実費でCD-ROMを入手できる。

致命的なバグを修正

Red Hat 7J改では**表1**のような不具合が修正された。

Red Hat 7Jには新しく採用されたアップデートプログラムが使用するデーモンプログラムrhnsdが、システムのリソースを開放しないために、起動後約3週間でLinuxシステムが停止するという大きなバグが存在した。Red Hat 7J改ではこの不具合が修正されている。

このほかにも、起動スクリプトの場所が変わったFreeWnnのデーモンプログラムがデフォルトで起動するようになったり、kinput2と同時にCannaが呼び出されるようになるなど、日本語入力プログラムの周辺に細かい修正がほどこされている。

付録CD-ROMを利用してアップデート

本誌の付録CD-ROMに修正されたRPMパッケージを収録したので、現在Red Hat 7Jを使用しているユーザーは、本誌付録のCD-ROM（disc2）をRed Hat Linux 7Jマシンへセットして、

```
$ su
# mount /mnt/cdrom
# cd /mnt/cdrom/RedHat7Jupdate/
# ./install
```

としてアップデートしていただきたい。

製品名 Official Red Hat Linux 7J改訂版
価格 1万2800円（Deluxe）
2万9800円（Professional）
問い合わせ先 レッドハット株式会社
03-3257-0411
http://www.redhat.com/jp/

パッケージ	修正内容
FreeWnn	起動スクリプトを/etc/init.dから/etc/rc.d/init.dに戻してデフォルトでWnnデーモンを起動
LPRng	フォーマットされたストリングが使われるのを防ぐためにuser_syslogdを修正
Xconfigurator	「｜」や「＿」などのキーマップを修正
anaconda	インストール時に言語Englishを選択してもタイムゾーンがAsia／Tokyoになるように修正
e2fsprogs	16キャラクタラベルに変更
esound	0.2.20にアップデート
glibc	互換性のためja_JP.ujisをaliasに追加
gnorpm	ゼロ問題のためにdivisionを修正
initscripts	すでにセットされているOUTPUT_CHARASETが壊れないようにして/etc/sysconfig/i18nをlang.shにエキスポート
iputils	ラージバッファでルートの時にセグメンテーションフォルトにならないようss001010にアップデート
kpackage	qt1x+i18nのパッチのためリビルド
kterm	muffの表示を修正
mount	16キャラクタラベルを扱ってマウントさせるようにパッチを追加
namazu	SPECとtypoを修正
redhat-release	日本語版のリリースを更新
rpmdb-redhat	rpmデータベース作成時に、--notriggersオプションが必要となった
sawfish	日本語メニューを修正
slang-j	Kon用にBol_charをNULLにするよう修正
sysklogd	セキュリティホールを修正
sysstat	crontabへの登録を修正
tcsh	日本語カタログのディレクトリ名を修正
tmpwatch	正しい戻り値を得るためにsystem()の代わりにexecle()を使用
traceroute	パケットの最大長を正しくチェックさせるように修正
up2date	起動後に約3週間でシステムが停止するバグを修正
usermode	haltとrebootが引数をとらないようにパッチを修正
xinetd	前バージョンのバグ修正と内部サービスを変換しないように修正
xinitrc	kinput2がcannaを使うようにパッチを適用

表1　改訂版で修正された不具合
Red Hat 7Jに存在した3週間でシステムが停止するという大きなバグなどが修正された。

Distribution ▶▶▶

LASER5 Linux 6.4 Develリリース

レーザーファイブは、「LASER5 Linux 6.4 Devel」を12月8日より発売した。

LASER5 Linux 6.4は、Red Hat Linux 6.2（US）をベースに、Red Hat Linux 7（US）の主要な最新機能を取り込んだディストリビューションで、レーザーファイブにより、安定した日本語環境が実現されている。

今回より、ゲームソフト用ライブラリであるSDLや、ATOK、Wnn6がデフォルト設定でインストールされるなど、使い勝手が向上している。

主なバンドルソフトは、ATOK、Wnn6、eWnn、翻訳魂、ThinkFree Office、System Commander Lite、Sylpheed（メーラ）、Borland JBuilder 4、商用フォントなど。

なお、COBOL、JDK、JBuilder4などの開発用ソフトウェアを除いた「LASER5 Linux 6.4デラックス」も同時に発売する。両パッケージともはがき作成ソフト「K筆」がバンドルされている。

カーネルは2.2.16（2.2.17/2.4test8も収録）、glibcは2.1.3となっている。X Window System関連のバージョンは、XFree86が3.3.6（4.0.1も収録）、GNOME 1.2.4、KDE 1.1.2（2.0も収録）となっている。

価格は、Develが1万6800円、デラックスが9800円で、サポートは90日間。

レーザーファイブ株式会社（http://www.laser5.co.jp/）

TurboLinux DataServer 6 for Oracle8iリリース

ターボリナックスジャパンは、オラクルのデータベースに最適化されたディストリビューション、「TurboLinux DataServer 6 for Oracle8i」を11月30日より発売した。価格は5万円。

DataServer 6は、ターボリナックスとミラクル・リナックスが共同開発したもので、ミラクル・リナックスが発売している、「Miracle Linux Standard Edition Version 1.0」と同一製品である。今後、両社のダブルブランドで販売協力を行っていく。

ターボリナックスでは、ミラクル・リナックスとの連携により、今後、Oracle8iの新バージョンに迅速に対応してくことになる。

対応するデータベースは、Oracle 8i R8.1.5 for Linuxと、R8.1.6 for Linux。

カーネルは2.2.16、Glibcは2.1.3となっている。

主なバンドルソフトは、ATOK12、HDE Linux Controller 2.0 Express Edition、リョービTrueTypeフォント5書体、Oracle8i Workstation Server R8.1.6評価版（120日間トライアル版）。

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

OpenLinux eServer2.3日本語版の発売元が日本SCOに変更

日本SCOは、「OpenLinux eServer 2.3日本語版」を11月より発売した。eServerは、以前はネオナジーが販売していたが、eServerの開発元であるCaldera Systemsによる米SCOのサーバソフトウェア部門の買収により、日本SCOより販売されることとなった。なお、製品に変更はない。

ネオナジーが販売したeServerについてのサポートは日本SCOに引き継がれ、今後、Caldera Systems製品の国内販売は、日本SCOによって行われることになる。

なお、Caldera SystemsとSCOから買収した部門はCaldera,Inc.となる予定で、日本でもCaldera日本法人が設立される見通しだ。

日本SCO株式会社（http://www.nihonsco.co.jp/）

IA-64に対応したTurboLinux Workstation Pro 6.1英語版リリース

ターボリナックスジャパンから、開発者向けディストリビューションである「TurboLinux Workstation Pro 6.1英語版」が国内向けに発売された。

Pro 6.1英語版は、10月17日に米国のTurboLinux Inc.から販売開始されたものと同じもので、商用ディストリビューションとしては初めて、IA-64（Intel Itanium Processor：インテルの次期64ビットCPUアーキテクチャ）への対応が行われている。さらに、Power PCや、SPARCバージョンも収録されている。

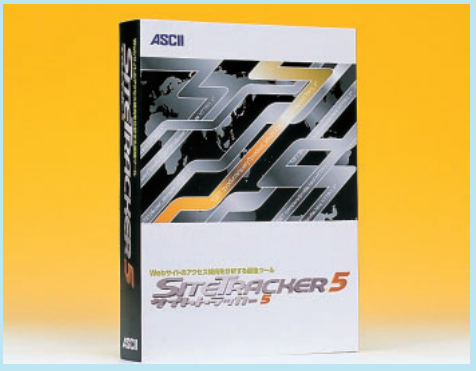
バンドルされる商用ソフトウェアはすべてIA-32向けのもので、それ以外のプラットフォーム用のソフトウェアはバンドルされていない。また、IA-32以外に関してはデベロッパーリリースとして、サポートも一切用意されていない。価格は9800円。

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

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

Products

Webサイトのアクセス傾向を分析するログ解析ツール



サイトトラッカー5

Webサイトがどのくらいアクセスされているかは気になるところだ。また、どこを経由してアクセスしたかといった情報は、重要なマーケティング情報となる。サイトトラッカーはそんなWebサイトのアクセス状況を分析するソフトウェアだ。

価格　9万8000円～
問い合わせ先　株式会社アスキー　ネットメディア事業部
Tel:03-5351-8590
http://www.ascii.co.jp/netmedia/

サイトトラッカーは、Webサイトがどのように見られているかを、HTTPのログから解析し、ホームページのアクセス状況など、さまざまな情報を分析することができるソフトウェアだ。Webサイトが効果的に提供されているか、どのくらいの人がサイトを見ているか（ページビュー）、どこのページからたどり着いたのか、どのくらいの間ページを表示していたのか、どのページが人気があるかなど、Webサイトを使ったe-ビジネスを効率よく展開するためのマーケティングツールである。

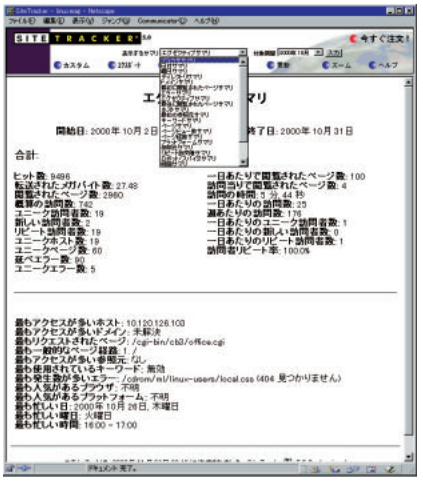
このサイトトラッカーのバージョン5が、11月10日にアスキーより発売された。

サイトトラッカー自体は、米国Sane Solutions, LLC.の製品で、米国ではNetTrackerとして販売されているものだ。アスキーでは、このNetTrackerの日本語版をサイトトラッカーとして販売している。

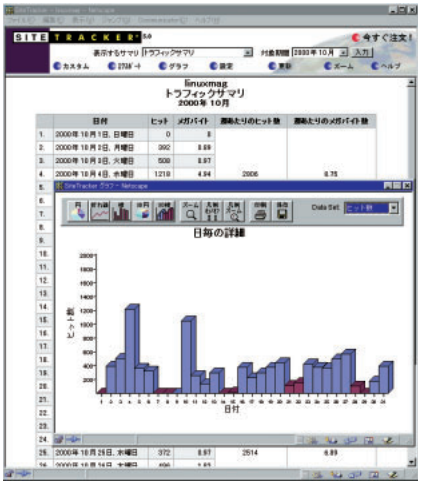
動作環境

Webサイトの分析結果の表示は、ブラウザベースで行うため、Javaの使えるWebブラウザであればクライアントを選ばない。ただし、今回記事作成用に使用した評価版では、Linux上では一部文字化けする部分があった（このため、画面はWindowsでキャプチャリングしてある）。

サーバ側の動作環境は、Linux（Intel）、Cobalt Linux、FreeBSD 3.x、4.x、AIX 4.x、Solaris、Windowsなどとなっており、対応するWebサーバは、Apacheを始め、多くのサーバに対応している。なお、分析結果を表示するためには、Webサーバが稼働している必要がある。



画面1　エグゼクティブサマリ
総ヒット数やアクセスの集中しているページなどサイト全体の情報を表示。ドロップダウンリストでサマリが選択できる。



画面2　トラフィックサマリとグラフ
転送されたデータ量やヒット数などを表示。棒グラフとして表示することもできるほか、円グラフ、折れ線グラフなどで表示することもできる。

多彩な機能

サイトトラッカーは、Webサーバのアクセスログを解析し、さまざまなサマリ（要点）で表示することができる。サマリは、サイト全体の情報を分析する「エグゼクティブ」、ユーザーが使っているブラウザを分析する「ブラウザ」、どのくらいのユーザーがアクセスしたかを分析する「クッキー」、ユーザーがどのようにページをアクセスしたかを分析する「ページ経路」、サイトがアクセスされた時間帯を分析する「時間」、1回のアクセスで表示されたページ数を集計する「ページビュー数」など、多彩な分析機能を持っている。

バージョン5ではさらに、リピート訪問者を分析する「リピート訪問者」、バナー広告による効果を測定する「広告キャンペーン」、日付や曜日による分析を行う「日付サマリ」「曜日サマリ」、人気ページのランキングを分析する「ページ」、サイトのエラーを分析する「エラー」などのサマリが追加され、合計35種類ものサマリが提供されるようになった。

これらの分析結果は、折れ線グラフや円グラフによって、ビジュアルに表示することができる。また、CVS形式や、Microsoft WordやExcelで読み込むことのできるRTF形式で出力することができ、データマイニングを自由に行うことができる。

そのほかサイトトラッカー5の新機能として、メールによるレポート配信や、さまざまなデータを自由に選択して、任意のレポートを作成するトレンドレポート機能が追加された。

サイトに合わせた3つのバージョン

サイトトラッカーには、基本となる「プロフェッショナル版」に加え、「エンタープライズ版」「eビジネス版」の2つの上位バージョンが用意されている。

エンタープライズ版は、複数のWebサーバの分析、プロキシ／ファイアウォールサーバ、FTPサーバの分析が可能だ。さらに上位のe-ビジネス版は、一日あたり数十万ページビュー以上の大規模サイト向けで、Oracleや、Microsoft SQL Serverを使ったデータマイニングが可能となっている。

マーケティングの新たな手法

通常の店舗販売のようなビジネスにおいては、来店者がどんな商品に興味を持っているのか、広告がどれほどの効果があったのかを掴むことは難しいが、インターネットビジネスにおいては、こういった情報を基にした効果的なWebサイトの構築が成功のカギを握っているといっても過言ではないだろう。そういった意味で、サイトトラッカーのようなツールは、マーケティング戦略において重要な意味を持ってくるだろう。

なお、サイトトラッカーの機能を試用できるオンラインデモと（実際はWebページ）、評価版が以下のWebサイトにて提供されている。

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

評価版は30日の限定版だが、実際にログファイルを分析できる。

クライアント	解像度	800×600ドット以上表示可能なデスクトップ
	ブラウザ	Microsoft Internet Explorer 4.0日本語版以上または、Netscape Navigator 4.0日本語版以上
サーバ	CPU	Pentium以上（WindowsではPentium II以上）
	メモリ	32Mバイト以上の空き容量
	ハードディスク	解析ログファイルの合計＋40MB以上の空き容量
	OS	Linux（Intel）、Cobalt Linux（MIPS、Intel）、FreeBSD 3.x、4.x、Solaris（SPARC、Intel）、AIX 4.x、Windows 95/98/Me/NT4.0/2000など
	Webサーバ	Apache、IPlanet Web Server、Lotus Domino、Microsoft IIS、NCSA、Netscape FastTrack/Enterpriseなど

表　動作環境

Express5800/SURFNAVI ＜Liteモデル＞

Webサーバ、メール、ファイアウォール、キャッシュなどのインターネット接続に必要な機能が、あらかじめセットアップしてあるアプライアンスサーバは、設置してすぐに利用が可能で、専任の管理者を割り当てられないユーザーには最適だ。

製品名	Express5800/SURFNAVI（Liteモデル）
価格	34万8000円
問い合わせ先	日本電気株式会社（NECソリューションズ） TEL 03-3455-5800 http://www.express.nec.co.jp/

NEC（NECソリューションズ）から、スリムタワーケースにインターネット接続のための機能を搭載したオールインワンサーバ「Express5800／SURFNAVI＜Liteモデル＞」（以下SURFNAVI）が発売された。

SURFNAVIのハードウェアは、CPUにCeleron 566MHz、メモリ64Mバイト、15GバイトIDEハードディスク、CD-ROMドライブ、フロッピードライブといった構成である。10/100BASE-Tネットワークは2ポートあり、ひとつはLAN用で、もうひとつはルータ接続用だ。OSには、TurboLinuxをカスタマイズして使用している。

初期設定はWindowsで行う

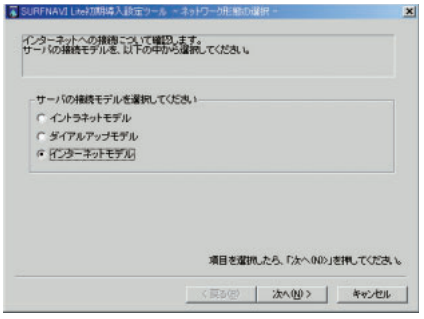
SURFNAVIを設置したらすぐに電源を入れてはいけない。SURFNAVIの設定はWebブラウザで行うため、まずはネットワークの設定が必要なためだ。ちょっと変に思われるかもしれないが、Windowsマシンに初期導入設定用フロッピーを入れて、「SURFNAVI Lite初期導入設定ツール」を起動する。そこで、まずサーバの接続モデルを選び（**画面1**）、管理者パスワードを変更し、IPアドレスやDNSなどのネットワーク情報を設定すると、フロッピーにその情報が記録される。

そのフロッピーをSURFNAVIに入れて電源を投入すると、SURFNAVIが起動中にフロッピーから情報を読み込んでネットワークの設定が行われるというしくみだ。起動したら、他のマシンでWebブラウザを起動し、「http://（SURFNAVIのIPアドレス）:50080/」に接続すると、Management Consoleで［一般ユーザ用］［管理者用］［ヘルプ］というシンプルな画面が表示される。管理者用をクリックするとユーザー名とパスワードを要求されるので、ユーザー名に「admin」、パスワードに先ほどの初期導入設定ツールで設定したパスワードを入力すると、**画面2**のように管理者画面のトップページになるので、あとは左側のメニューを選んで、SURFNAVIの設定を行っていく。

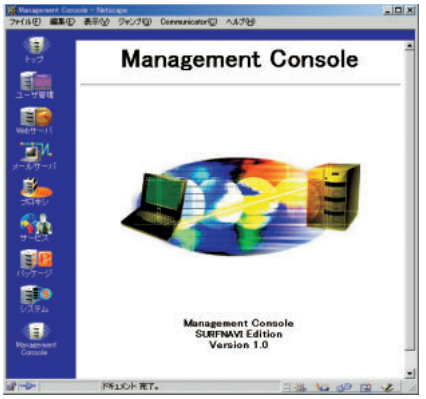
Webブラウザで環境設定

まずは「ユーザ管理」から新規ユーザーを作成する（**画面3**）。ここでは、ホームディレクトリとメールスプールのディスク使用量を、ユーザーごとに上限を設定できる。

次に、Webサーバを設定する（**画**



画面1　SURFNAVI Lite初期導入設定ツール
イントラネット、ダイヤルアップ、インターネット接続を選び、IPアドレスなどのネットワーク情報を設定する。



画面2　Management Console管理者トップページ
WebブラウザからSURFNAVIのさまざまな設定の変更や状態の確認ができる。

画面3　「ユーザ管理」の「新規ユーザ」設定
ダイヤルアップ接続の場合には、メールサーバ（POP）とパスワードを入力する。

画面4　Webサーバの設定
Apacheの設定ファイルをエディタで修正するのは面倒だが、これなら簡単に行える。

画面5　システムメニュー
システムの状態を見ることやその他の設定機能が集められている。

画面6　ディスク使用状況の表示
ハードディスクの使用状況がわかりやすくグラフで表示される。

面4)。エディタでApacheの設定ファイルを修正する場合に比べて非常に簡単だ。そしてメールサーバ、プロキシサーバと順に設定していく。

サービスのメニューには、「DHCPサーバ」「ファイル転送（FTP）」「Windowsファイル共有（Samba）」「ネットワーク管理エージェント（SNMP）」の設定がある。それぞれ必要に応じて変更すればよいだろう。

システムメニューには多くの機能が用意されていて、上段にシステムの停止／再起動を行うボタンがある（**画面5**）。

中段にはシステム状態を表示するボタンがあり、「ディスク使用状況」をクリックすると、ハードディスクの使用状況が表とグラフでわかりやすく表示される（**画面6**）。そして、「WEBアクセス統計」をクリックすると、ブラウザの別ウィンドウが開き、WebサーバとしてSURFNAVIを利用した場合に記録されているアクセスログを、Webalizerというソフトで解析して表示する。

下段の「その他」には、ネットワークの設定変更、バックアップ処理とスケジュールの設定、管理者パスワードの変更などサーバ管理に必要な機能が割り当てられている。ログ管理では、サーバの/varディレクトリに記録されているログの表示だけでなく、ログローテート周期の設定などを行うことも可能だ。

また、パッケージメニューでは、NECやターボリナックスから提供されるサービスパックやソフトウェアをRPMパッケージのCD-ROMやURLで指定することで、SURFNAVIにインストール（アップデート）を行うことができる。

写真1　SURFNAVIの内部
コンパクトPCとほぼ同じ部品構成である。

CPU	Celeron 566MHz
RAM	64Mバイト（最大384Mバイト）
ハードディスク	15Gバイト
CD-ROM	40倍速CD-ROM
フロッピードライブ	3.5インチ1.44Mバイト
グラフィックス	ATI 3D RAGE PRO（4MバイトSGRAM）オンボード
ネットワーク	10/100BASE-T×2ポート
インターフェイス	RS-232C×1、パラレル×1ポート、PS/2×2ポート
PCIスロット	2スロット
サイズ（mm）、重量	85（W）×367（D）×316（H）、10kg
電源	200W

表1　Express5800/SURFNAVI＜Liteモデル＞の主な仕様

サーバ統合管理ツール ESMPRO

添付されるサーバ統合管理ツール「ESMPRO/ServerManager」（サーバマネージャ）で数百台のサーバを管理することが可能だ。ネットワーク管理エージェントは、SURFNAVIの状態を常に監視し、温度／電圧／ファンなどのハードウェアの異常や、ディスクの容量不足／回線障害／リソース不足などの警告を、サーバマネージャに伝える。

なお、データベースと連係したWebページを作るのに便利なPHPが利用できるように、Apacheが設定されている。フリーのRDBMSソフトPostgreSQLもインストールされているので、それと連動した本格的なWebサーバとして利用することも可能だろう。

箱の中のペンギンたち

文：みわよしこ
Text : Yoshiko Miwa

第1回
Linuxを「組み込む」とは？

「組み込みOS」という言葉を聞いて何か具体的なイメージを思い浮かべられる読者は少ないでしょう。組み込みOSは一般ユーザーの目に見えないところで働いている、いわば「縁の下の力持ち」なのです。

現在の電気機器のほとんどは、見える見えないにかかわらずコンピュータ制御されています。

たとえば炊飯器もそうです。マイコン制御が一般的になる以前は、熱を感知して形状の変わる金属部品（バイメタル）で温度制御をしていましたが、現在では言うまでもなく、炊飯器のほとんどはコンピュータが温度制御を行っています。ある統計によれば、現在の日本国民の30%ほどは将来においてもパソコンを購入する予定がないそうですが、その人たちも炊飯器でお米を炊いている以上、コンピュータを家庭で「利用して」いるのです。

ほかの多くの電気機器も同様にコンピュータが組み込まれています。したがって、乳児を除くほとんど全国民が家庭で「コンピュータを利用して」いることになるのです。

電気機器だけではありません。ガスや石油を燃料とする暖房器具も、車も、今やコンピュータを内蔵していないものを探すほうが難しいでしょう。今やコンピュータは当たり前のように、「縁の下の力持ち」として、ありとあらゆる機器に組み込まれています。そして、そこで利用されているのが「組み込みOS」というものです。

多機能さは求められない組み込みコンピュータ

一般のパソコンには、高性能さが求められます。またソフトウェアを追加してゆくことにより、1台のパソコンをワープロ、表計算、Webブラウザ、メール送受信、AV……と多様な用途に利用するのが普通です。

一方、機器に組み込まれたコンピュータは、極めて限定された目的に従い、一定条件での計測・制御を行います。先ほどの炊飯器の例で見てみましょう。

炊飯器を制御するコンピュータが果たすべき最低限の役割は、次のものです。

①炊飯スタートの指令を受けたら「始めチョロチョロ」で釜を弱く加熱する
②一定の温度に達したら「中パッパ」で釜を強く加熱する
③米に火が通ったら「赤子泣くとも蓋取るな」で釜をごく弱く加熱する
④炊飯が終了したら加熱をやめる

したがって炊飯器に組み込まれるコンピュータに必要な機能は、

- **温度を測定する**
- **加熱を制御する**
- **「炊飯中」「炊飯終了」などの状況を、インジケータ・LEDなどでユーザーに知らせる**

の3点です。通常のパソコンに当然のこととして含まれている、文字入力、画像表示、音声出力などの機能はほとんど必要ありません。炊飯に関する指示を与えるのには、キーボードではなく、いくつかのボタンで十分です。ユーザーと炊飯器がコミュニケートするには、ディスプレイへの表示ではなく、たとえばモードを示すLEDが点灯・消灯したり、数字など、限られた文字を表示する液晶パネルがあれば十分ということになります。炊飯終了を知らせるのはブザーでもいいのです。

また炊飯器のコンピュータそのものにも、それほど高い機能は求められません。測定したデータは、ほぼ即時に（リアルタイムで）、加工を加えることなく、温度制御に利用されます。

まずメモリ、つまり扱うデータについて見てみましょう。扱うのはほぼ温度のみです。炊飯器の場合には、温度センサーから入力されたデータに対して、リアルタイムに「このまま加熱を続ける」「加熱を停止する」といった処理を行います。データを蓄積する必要はありません。したがってメモリはKバイト単位でよいのです。

次にプロセッサ、つまりデータをどう扱うかについて見てみましょう。炊飯器の行う処理はさほど複雑ではなく、リアルタイムでの処理がほとんどです。したがって、プロセッサの性能も高くなくてよいのです。このようなことから、組み込み用途では現在でも往年の8ビットCPU「Z80」が利用されることすらあるのです。

通常のパソコンの性能は「高ければ高いほどよい」のですが、組み込み用途では「必要にして十分」が求められます。炊飯器に組み込まれているコンピュータは、炊飯器に必要な、ごく限定された機能だけを持ったコンピュータなのです。

ここまで炊飯器の例を見てきましたが、他の機器に組み込まれているコンピュータでは、必要とされる機能が異なります。たとえば車の制御用のコンピュータであれば、温度だけではなく、たとえば車軸の回転数やブレーキの圧力の測定・制御の能力も必要でしょう。ガス暖房機器の制御用のコンピュータであれば、温度制御のほかに、空気中の一酸化炭素ガス濃度を測定し、一定濃度を超えたら運転を停止する機能が求められることもあるでしょう。

いずれにせよ、極めて限られた目的のために限定した機能を搭載し、その目的でチューニングされているのが組み込み用途のコンピュータというものです。

コンピュータを「組み込む」とは？

通常のパソコンと測定ボード、温度センサー、電熱器、鍋を利用して、炊飯器を作ることはもちろん可能です（**図1**）。

しかし、温度を測定して制御するだけのために、サーバにもワープロにも利用できるパソコンを使用するのは、あまりにも効率が悪いといえます。炊飯の進行状況を表示するのにディスプレイを利用する必要もないでしょう。さきほど述べたように、LEDや液晶パネルで十分です。

そうなれば、グラフィックカードが不要になります。拡張バスも必要ありません。マザーボードと測定ボードを一体化したものがあれば機能的には十分です。マザーボードは小さければ小さいほどいいですし、もちろんCPUのスペックも、

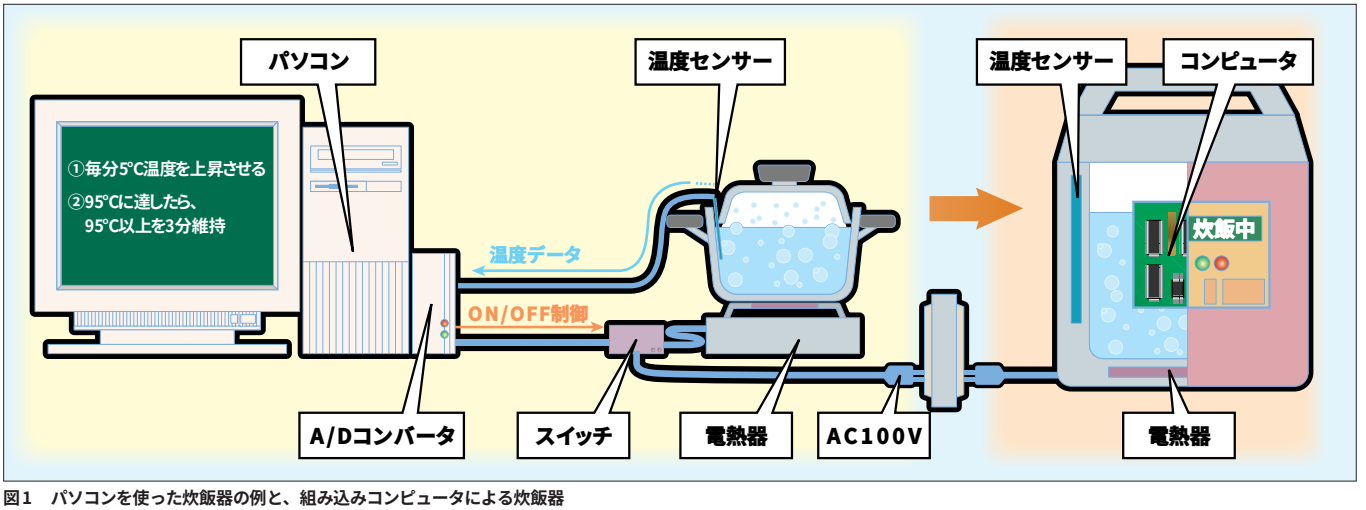


図1　パソコンを使った炊飯器の例と、組み込みコンピュータによる炊飯器

8ビット程度で十分です。これはボードを小さくできるということでもあります。

また、狭い台所にパソコンの筐体があっては邪魔です。通常のパソコンやディスプレイの筐体は防水加工されているわけではないので、台所に置いておくのは危険ですらあります。8ビットCPUを搭載した小さなコンピュータであれば、炊飯器の本体内に、防水・断熱して格納してしまうことができるのです。

このようにして、必要かつ十分と考えられる極限まで機能を減らし、また用途に適した実装を行った結果が、通常見かける電気炊飯器です。

コンピュータを「組み込む」とは、このように、コンピュータの機能を必要かつ十分なレベルに限定し、使用条件に適した実装を行うということなのです。

組み込みOSの特徴

通常、パソコンやパソコンOSに求められる、①機能の多さ、②性能の高さ、③拡張のしやすさの優先順位は、組み込み用途のコンピュータやOSではけっして高くありません。このことは、炊飯器の例でも理解いただけたと思います。

逆に、組み込みコンピュータ・組み込みOSに求められるのは次のようなものです。

①機能が限定しやすい

たとえばOSは、利用しない用途をできるだけ削り、少しでもコンパクトにすることが求められます。

②必要に応じた性能が選べる

Z80で十分な用途にPentiumを利用する必要はありません。性能の高さより、必要に応じた性能が選択できることが求められます。もちろんメモリ容量についても、単に多いことより、必要にして十分な容量を搭載できることが望ましいのです。

③必要に応じて入出力機器に対応できる

パソコンの「シリアルもパラレルもUSBもOK」という世界は、組み込みコンピュータには無縁の世界です。用途に対して必要にして十分な入出力機器を接続できればよいのです。

加えて、次のような機能も求められます。

④リアルタイム性

高速で動くものを制御する場合、細かいタイムステップでの入力・応答が可能であることが求められます。リアルタイム用途のOSでは一般に、100マイクロ秒以下のタイムステップに対応できることが必要といわれています。

⑤動作途中に電源をオフできること

機器に搭載されたコンピュータは、機器のスイッチが切られれば電気が供給されなくなります。このため、電源オフと同時にOSの動作を停止できることが求められます。

組み込みOSは見えない

以上、炊飯器を例にとって組み込みコンピュータ・組み込

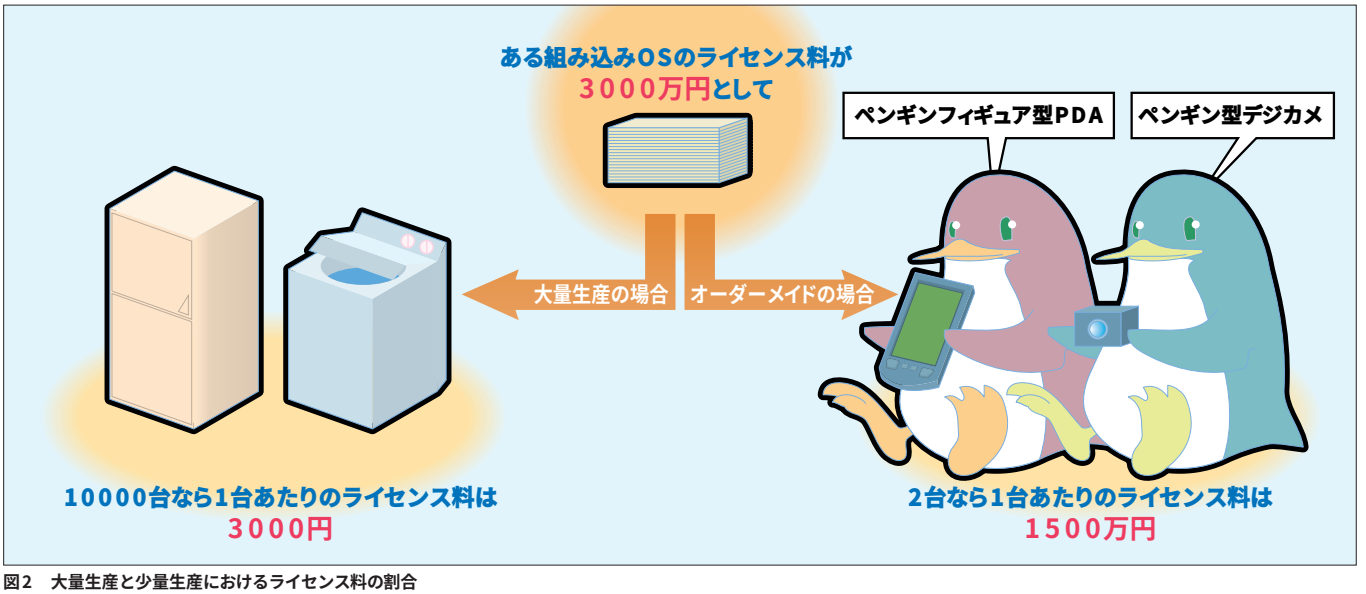


図2　大量生産と少量生産におけるライセンス料の割合

みOSの世界を解説してきました。しかしながら、自分の持っている炊飯器がどういうプロセッサと、どういうOSを利用しているかを知っている人はほとんどいないでしょう。筆者も知りません。そういった情報が製品マニュアルに記載されているわけでもありません。炊飯器を利用するうえでそんな情報は必要ないのです。

組み込みOSの世界は、「CPUは……」「プレインストールOSは……」といったことが宣伝上有効なパソコンの世界とはまったく対象的です。プロセッサやOSの名前は、完成した製品が出荷される時には、往々にしてどこにも出てきません。製品を利用するユーザーは、組み込まれているプロセッサやOSを意識する必要がないからです。

では、なぜLinuxの組み込み用途での利用が、このところホットなのでしょうか？ Linuxであることを意識せずに利用されるかもしれないLinuxに、どういう意味があるのでしょうか？

性能におけるLinuxのアドバンテージ

組み込みOSに求められる5つの条件を前述しました。これらのそれぞれについて、Linuxにどういう利点があるか見てみましょう。

①機能が限定しやすい

Linuxカーネルは、よく知られているとおり非常に小さいものです。組み込むモジュールによってサイズは変化しますが、多くともせいぜい数Mバイト以下です。機能を限定したい場合は、必要最小限のモジュールだけでカーネルを構築すればよいのです。

②必要に応じた性能が選べる

Linuxは80386でも動作します。これは、要求される性能によって、80386からPentiumに至る多様な選択肢からCPUを選択できるということになります。80386ですら高性能すぎる用途に対しては、4〜16ビットプロセッサや、ほかのOSを利用することになるでしょう。これに関しては、後出の日本エンベデッドリナックスコンソーシアム会長の中島氏のインタビューを参照してください。

③必要に応じた入出力機器への対応

Linuxは、シリアル、パラレル、SCSI、USBなど、一般的なバスのほとんどに対応しています。これらを利用すれば、たいていの入出力機器への対応が可能になります。

④リアルタイム性

リアルタイム用途に特化したLinux（RT-Linux）のタイムステップは15マイクロ秒となっており、一般的にリアルタイム用途で要求されるタイムステップ100マイクロ秒以下をクリアしています。

⑤動作途中に電源をオフできること

通常、Linuxマシンの電源をオフするにはシャットダウン操作が必要です。しかし、リアルタイム対応ディストリビューションの一部は、シャットダウン操作なしにマシンの電源をオフにしても障害が発生しないようになっています。

組み込み機器の発展も推進する？ オープンソースの思想

Linuxがオープンソースの思想のもとに発展・普及したことは今や「常識」です。では組み込み用途に対して、オープンソースはどういった影響を与えるのでしょう。

通常、商用の組み込みOSを利用するにはライセンス料が必要になります。ライセンス料は製品1個ごとに支払うのではなく、「使用料」の形で包括して支払うのが通常です。

したがって、家電製品のように「数の出る」製品を大量製造・大量販売する場合には、製品1つあたりのライセンス料は安価になります。OSを自社開発するコストを考慮すれば、ライセンス料を支払っても採算が取れることが多いでしょう。

しかし研究・開発的要素の強い製品をごく少量製造する場合や、用途や状況に個別対応したオーダーメイド品が主である場合には、製品1つあたりのライセンス料は高価になってしまいます。このような場合には、ライセンス料を考慮すると、いかに性能がすぐれていても商用組み込みOSは利用しづらいでしょう。

オープンソースのLinuxの場合、もちろんライセンス料はかかりません。ライセンス料がネックになって開発がやりにくいケースの場合、Linuxを利用することによって開発が容易・可能になるということになります。たとえば、研究・開発段階の売れるかどうかの見通しの立たない製品や、大量製造・大量販売を見込めない製品の開発が、ライセンス料の重圧がかからなくなったために、容易になったり可能になったりするのです。いわばLinuxは、ユニークな組み込み製品や世界に1つだけの製品を開発しようと思っている人々の援助をしているのです。

もちろん、OSに対するライセンス料が必要ないというメリットは、大量製造・大量販売する製品にとっても同様です。

日本エンベデッドリナックスコンソーシアム会長 中島達夫氏に聞く

Interview

一般ユーザーには馴染みの薄い組み込みLinux。ましてや「組み込みLinux業界」となると、さらに馴染みの薄い世界でしょう。
今回は、2000年7月13日に発足したばかりの「日本エンベデッドリナックスコンソーシアム（以下、Emblix）」会長の中島達夫氏（早大助教授）に、コンソーシアム設立の経緯、なぜLinuxなのか、今後の展望などについてお話をお伺いしました。

――この「Emblix」は、どういったことから設立されたのでしょうか？

中島：これまで組み込みシステムの対象が比較的小規模で機能が低かったので、OSとしてはITRONなどのリアルタイムOSが広く使われてきました。しかし、昨今のインターネット応用機器に代表されるように、これまでと比べて大規模化、高機能化するに伴い、「ファイルシステムがない」「ブラウザもウインドウシステムもない」といった機能的な限界が見えてきました。

また、将来の組み込みシステムを考えると、さまざまな分散システムのためのミドルウェアが必要となり、新しい組み込みOSが必要になってきました。同時に、世界的に大流行しはじめたLinuxが、組み込みOSが必要とする機能を十分に持っていました。そういった状況から、この「Emblix」設立の話が自然発生してきました。

私どものコンソーシアムは、つい先日の2000年7月13日の設立発表会で発足したばかりで、これから組織が形になっていくという段階です。コンソーシアムに関する詳しい情報はホームページを見てください（http://www.emblix.org/）。

写真　日本エンベデッドリナックスコンソーシアム会長　中島達夫氏
1961年生まれ。慶応義塾大学理工学部博士課修了後、カーネギメロン大学、ドイツ国立計算機研究所訪問研究員、北陸先端科学技術大学院大学、情報科学研究科助教授を経て現在、早稲田大学理工学部情報学科助教授。専門はリアルタイムOS、耐故障オブジェクト指向システム、大規模ユビキタスコンピューティング、モーバイル／マルチメディアシステム。

――ロゴマークがお堅くなくて可愛いですね。

中島：ええ。広告代理店のデザイナーの作品です。

――今後の活動のご予定をお聞かせください。

中島：活動内容としては、具体的なテクノロジーや技術標準に関するワーキンググループ活動、技術動向調査や特許に関する専門小委員会を設けての活動を中心にしたいと考えています。ただ現在はまだ、どういうものが必要か、誰が適任かを議論している段階です。議論の結果は本年11月に有明国際展示場で開催された組み込み技術展示会で発表いたしました。

ただ、単に組み込みLinuxをプロモート・標準化するのみではなく、新しい組み込みOS、新時代のコンピュータ環境を作る手伝いをしてゆけたらいいと思っています。また技術的な話だけではなく、それが社会におよぼす影響についても考えてゆきたいと思っています。

――「社会におよぼす影響」とは、具体的にはどういったことでしょうか？

中島：たとえば、ありとあらゆる電気機器にLinuxが組み込まれるようになったら、どんな社会生活が可能になるか、ライフスタイルはどう変化するかといったことです。

――これまで広く利用されてきたITRONの今後が気になるところですが。

中島：Linuxは堅牢で、安定していて、機能も充実しているなどの長所があります。もともと、80386以後の32ビットCPUを用いた比較的大規模な応用を対象にしていますしね。一方、規模の小さい応用に対しては、従来どおりITRONがメインで用いられてゆくと思います。つまり小規模で比較的機能が単純なものにITRON、大規模で機能が豊富なものにはLinuxといった棲み分けができるでしょう。ここでいう規模はCPUのビット数ではありません。小規模でも性能を重視する場合には、ITRON+32ビットCPUが使われています。

また、1つのマシンの中でLinuxとITRONを同時に使うという「共存」もありうるでしょうね。

そういったことを考えると、「Linuxを組み込み用途で利

用するための技術」だけではなく、LinuxとITRONを棲み分けさせる、使い分ける、共存させる技術も必要になってくるだろうと思います。

――「Emblix」は、そこでどういう役割を果たすのでしょうか？

中島：まず、Emblix自身では開発はやらない、規制もしないということを申しあげておきたいと思います。

開発は各企業でそれぞれ行っていますので、各企業の成果に対してコンソーシアムが示唆やまとめをするといった形を考えております。各企業の立場はいろいろですので、どういった形になるか現段階では未知数ですが……。

コンソーシアムの活動成果が一般に見えてくるのは、数年先のことになりそうです。組織が回っていったあとに何をしていくかが、コンソーシアムにとっての課題でしょうね。

――Linuxは、組み込み用途でより広く利用される可能性がありますか？

中島：Linuxの組み込みは当たり前になるでしょうね。ネットワークに最初から対応していますから、たとえば家電のネットワーク対応といったことが容易に実現できます。

進んだシステムを作ろうとする場合も、Linuxではすでに開発されている多くのソフトウェアを利用できます。ただし、ネットワークに接続して何をするかはこれからの課題でしょうね。

Linuxは組み込み用途のインフラとしてはReadyなんです。むしろ、使ってどうするか、将来の組み込みをどうするかという段階にきています。

――組み込みOSの世界の文化が、Linuxで変わる可能性もありますか？

中島：そうですね。私どものコンソーシアムは、もともとオープンソースの思想のもとに開発されたLinuxの技術の情報を、みんなに見えるようにする組織でありたいと思っています。開発者のコミュニティを作るのが最大の目的かもしれません。

また、オープンソースの組み込みOSが普及することにより、大学における組み込みシステムの研究を、実際の製品に利用することが容易になると思います。

――同じ時期に米国でも組み込みLinuxのコンソーシアムが発足したようですが。

中島：「Embedded Linux Consortium（ELC）」ですね（http://www.embedded-linux.org/）。Linuxはネットワークに最初から対応していますので、将来の組み込みシステムのインフラにはまさにうってつけのOSです。世界的に組み込み用途で注目されるのは、当然のなりゆきでしょう。

画面　日本エンベデッドリナックスコンソーシアムホームページ（http://www.emblix.org）

――組み込みLinuxの将来を、中島会長はどう予測されますか？

中島：まず、ひとつ申しあげたいのは、現在のパソコンは「組み込みの特殊形態」だということです。汎用性を非常に高くした形態といいますか……。ですから、Linuxの組み込みが普及していくと、パソコンの形態が現在のようではなくなってゆくでしょうね。

それに、組み込みだから特別な技術が必要ということはないんです。組み込みには、一般的なLinuxの技術がそのまま利用できるわけです。ですので、組み込みに利用したからといってLinuxが変わるということはないでしょう。

ただし、組み込み用途ではさまざまなユーザーインターフェイスや環境に対応する必要がありますので、そこが難しいかもしれません。

――OSをLinuxに限定する必要もありませんね。

中島：そうですね。Linuxと同じインターフェイスを持っているんだけど、中身は別のOSという形態で、新しいOSを開発するという方向性も考えられます。

――最後に、他団体とどのような関係を持たれるご予定かについてお聞かせください。

中島：他団体との協力、情報交換は積極的に行なっていくつもりです。

特に米国の組み込みLinuxのコンソーシアムである、ELCとは積極的に協力関係を築いていく予定です。

実際、9月のELC総会にもEmblixメンバーが参加しましたし、11月にMST2000と同期して開催された第1回Emblix総会にはELCの幹部が来日しました。

――技術的な面でも組織活動の面でも、広がりのある活動が期待できそうですね。ありがとうございました。

これであなたもXマスター!

XFree86 セットアップ徹底ガイド

Linuxの標準X Window SystemとしておなじみのXFree86。GUI環境を実現するために欠かせないソフトウェアコンポーネントである。フリーソフトでありながら、そのハードウェアサポートは高いレベルにある。最近では設定ツールも充実しインストーラで一発設定できるケースがほとんどだが、逆にここでうまくいかないと深みにはまってしまうことも……。しかし、あきらめてしまうのはまだ早い。基礎からXFree86を見つめ直して、セットアップに再挑戦だ!

illustration : Chata Tachibana

XFree86設定ツール大全

文：竹内充彦
Text : Michihiko Takeuchi

多くのディストリビューションでは、ユーザーにわかりやすいXFree86の設定ツールを採用し、インストーラのセットアップ手順にも組み込んでいる。これらのツールは、なにもインストール時にだけしか起動しないわけではない。インストール時に設定できなかった場合や、あとからビデオカードやモニタを交換した場合などに、いつでも起動して設定作業をすることができるのだ。

ここでは、多くのディストリビューションに採用されているXの設定ツールの使い方を解説していく。

事前に調べておきたいこと

各種ツールを使う前にまず、Xを設定するために必要となる項目について触れておこう。実際に設定作業を始める前に以下の項目を調べておけば、あとからあたふたしなくてすむはずだ。

画面1　メーカーのWebページ
これはCreative Technologyのホームページ。搭載されているチップの種類やビデオRAMの容量、RAMDACスピードなどのスペックが記載されている。設定を行う前にチェックしておこう。

●ビデオカードのメーカーと型番
●ビデオチップのメーカーと型番
●搭載されているビデオRAMの容量
●モニタのメーカーと型番
●モニタの同期周波数

ビデオカードのメーカーと型番は、たとえば「Creative 3D Blaster」とか「AOpen PA256MX」のような、パッケージに書かれた商品名のことだ。ビデオチップのメーカーと型番は、「nvidia RIVA-TNT2」とか「ATI RAGE128」のような、ビデオカード上に乗っている具体的なチップの名称である。ビデオ統合型のチップセットを搭載したマザーボードの場合は、i810やi815といったチップセットの通称を控えておけばいいだろう。

さらに搭載しているビデオRAMの容量を調べておく。パッケージやマニュアルに記載されているはずだ。バルク品の場合は、メーカーのWebページで調べるか、PC起動時に画面に表示される場合もあるので、それを控えておこう。i810などでは、メインメモリの一部をビデオRAMとして使用するため、BIOS等で設定した値を控えておく必要がある。

バルク品や中古品でかつヒートシンクでチップが隠れているようなビデオカードの場合には、どうにもならないのだろうか。実はそんなことはなく、そのビデオチップがこれから設定しようとするXのドライバデータベースに登録されているものであれば、XFree86に標準添付のSuperProbeというコマンドを実行して検出することができる。SuperProbeはスーパーユーザー（root）で実行する必要がある。

```
$ su （スーパーユーザーになる）
# /usr/X11R6/bin/SuperProbe
```

実行すると一瞬画面が切り替わり、検出したビデオカードに搭載されるビデオチップとメモリ容量、RAMDACの種別を表示してくれる。

RAMDACとは、ビデオメモリの内容をビデオ信号に変換するD/Aコンバータのことだ。最近ではこの機能自体もビデオチップに統合されているため、チップがきちんと検出されていれば、さほど気にする必要はない。しかし、たとえば手元にあるATI Mach64（1994年の製品）という旧タイプのビデオカード上には、別のチップとして搭載されている。このように別チップで搭載されている場合は、カードの型番やリビジョンに応じてRAMDACのチップが異なることもある。

とはいえ、カードの製品名がわかっていたほうが設定が容易になる局面もあるので、パッケージやマニュアルがあるなら製品名は調べておきたい。

モニタのメーカーと型番は、「Sony CPD-1730」とか「iiyama MT8617」のような、モニタの正面や背面、あるいは底面に記載されている製品名のことだ。メジャーな製品ならば、型番がわかるだけで、面倒な同期周波数の設定を避けることができるだろう。メジャーでない製品の場合、垂直同期周波

数と水平同期周波数を調べておく必要がある。マルチスキャンタイプ（現在一般的なモニタはほとんどがこのタイプ）の場合、スキャン可能な周波数帯域が「水平同期31.5－50kHz、垂直同期50-70Hz」のように、範囲で示されているはずだ。この範囲は、製品のマニュアルに記載されていることが多い。また、モニタの筐体の背面や底面に記載されていることもある。ただし、メーカー製PCのセット売りモニタの場合、これらの周波数がどこにも記述されていないことも多い。不明な場合は、標準的なSVGA、XGAモニタとして使うしかない（それでも1024×768ピクセル程度の解像度は期待できる）。

さて、これらの情報を控えたら、次は設定ツールを実際に使ってみよう。

設定ツールあれこれ

もともとXFree86には、xf86configというテキストベースの設定ツールと、XF86SetupというGUIを用いた設定ツールが用意されていた。しかし、これらのツールは初心者にはわかりにくく、あまりポピュラーではない。そこで、各ディストリビューションとも、初心者にもわかりやすい設定ツールを開発・調達し、インストールするようにしている。

ここでは、あなたが使うべきXの設定ツールを見極めてみよう。まずは**図1**のチャートを使って設定ツールを判断しよう。

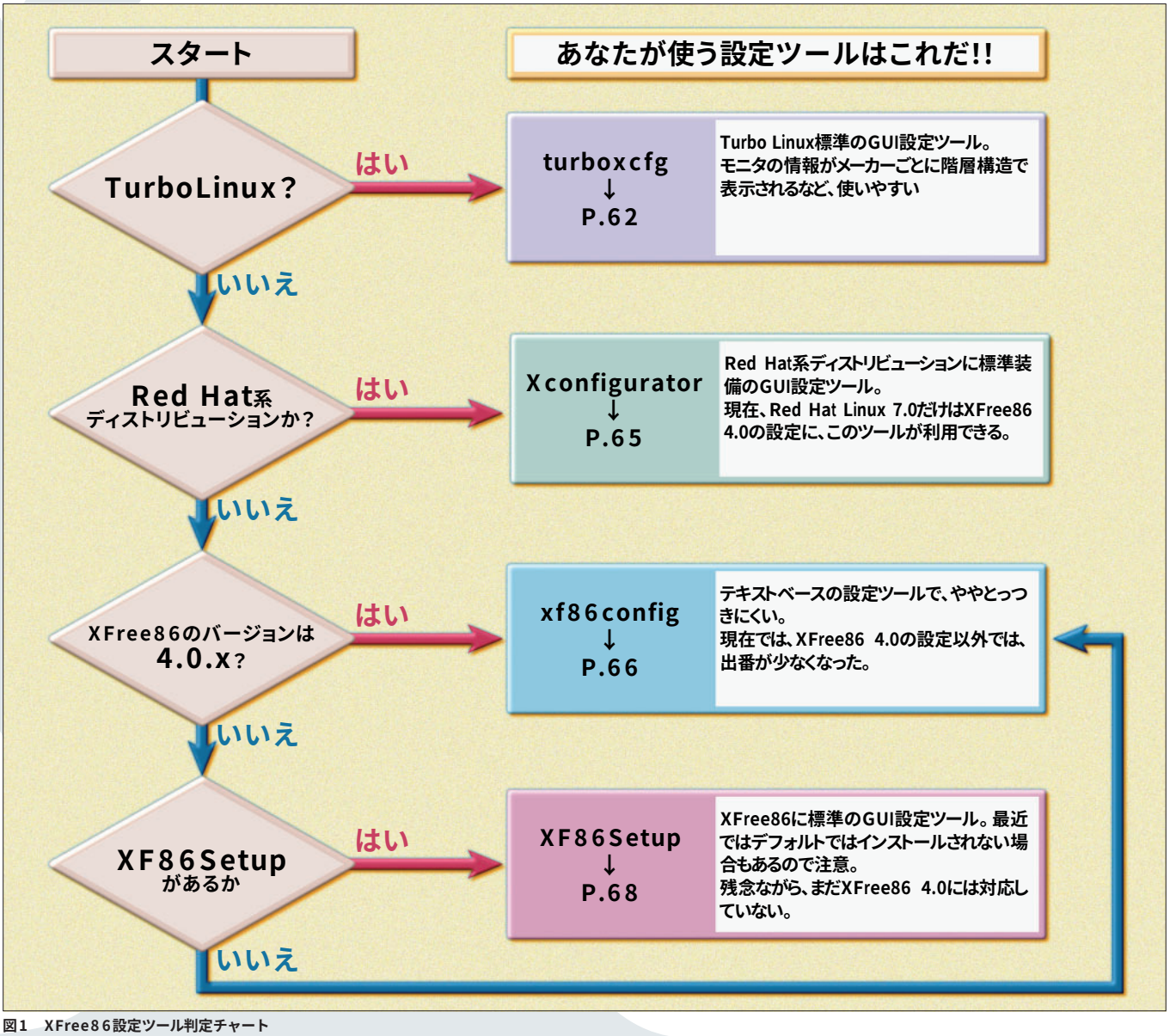


図1　XFree86設定ツール判定チャート

■turboxcfg

turboxcfgはTurboLinuxが開発したXFree86の設定ツールだ。TurboLinuxは、このXの設定ツール以外にも、ネットワークやプリンタなどの設定用に独自のツールを開発し提供している。もし、あなたがTurboLinuxをインストールしていて、かつXFree86 3.xを設定するならば、このツールを使うのがベストだ。

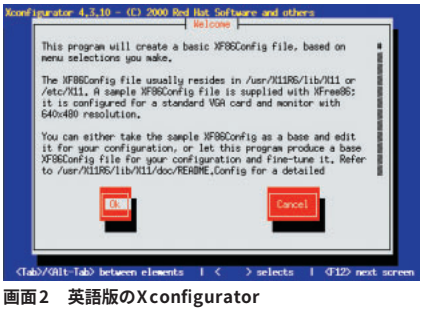
■Xconfigurator

Xconfiguratorは、Red Hatが開発したXFree86の設定ツールだ。したがって、Red Hatをベースにした、いわゆるRed Hat系ディストリビューションでは、このツールを採用していることが多い。XFree86 3.x対応の最新バージョンは4.3.5となっている。

またRed Hat Linux 7.0では、XFree86 4.0.1が採用されており、それに合わせてXconfiguratorも4.4.3となっている。

■xf86config

xf86configは、XFree86標準ともいえるテキストベースの設定ツールだ。とはいえ、このツール、XFree86が起動時に参照する設定ファイル「XF86Config」（たいてい/etc/X11/に作成される）の基本形を生成するだけというシンプルなものである。ほかのツールと異なり、Xの実行テストを行ったりすることはできない。設定を行い、Xを起動しては不具合を調整するといったトライアンドエラーが必要になる。ただし、XFree86 4.0.xに対応しているのは前出のRed Hat Linux 7.0に付属のXconfiguratorを除けば、唯一このxf86configだけである。したがって、XFree86 4.0.xをインストールする場合には利用する機会があるだろう。



画面2　英語版のXconfigurator
Xconfiguratorはディストリビューションによって英語版と日本語版にわかれる。ちなみに、日本語版を通常のコンソールから起動すると文字化けする。

■XF86Setup

XF86Setupも、XFree86標準のツールである。xf86configと違い、こちらはGUIになっており、マウスによる操作が可能だ。作業の手順がわかりにくい、製品リストが充実していないなど、ほかのGUIツールに比べると使い勝手がよくない。しかし、標準ツールということもあり、ほぼどのディストリビューションでも使うことができる。ただし、最近では別の設定ツールが付属するため、デフォルトでインストールされないこともある。どうしてもXF86Setupを利用したい場合は、rpmを使って別途インストールする必要がある。XF86Setupは、VGA16サーバを使って起動するのでこれもあわせてインストールする。Red Hat 6.2ならば以下の手順だ（スーパーユーザー権限が必要）。

```
# mount /mnt/cdrom
# cd /mnt/cdrom/RedHat/RPMS
# rpm -ihv XFree86-VGA16-3.3.6-20ji.i386.rpm
# rpm -ihv XFree86-XF86Setup-3.3.6-20ji.i386.rpm
```

■ツールを使うときの注意

ここで紹介している設定ツールはコンソールから起動する。Xが未設定なら当たり前の話だが、再設定時には注意したい。ブート時のLILO boot:プロンプトに「linux 3」と入力するなどして、テキストログインモードにしておこう。いずれのツールもX上で起動することはできるが、X上で起動した場合、自動検出や動作テストがうまく機能しないことがあるので、お勧めできない。

xf86config以外のツールは、英語環境では英語表示、日本語環境では日本語表示で動作するようになっている。ただでさえわかりにくい設定項目の説明をわざわざ英語で読むこともないので、ツールを実際に起動する前には、必ず日本語環境のコンソールkonを起動しておこう。

```
$ /usr/bin/kon
```

設定ツールは、/etc以下のいくつかのファイルを変更するため、スーパーユーザーで実行する必要がある。suコマンドを使ってrootになっておくのを忘れずに。

ツール内でXの動作テストを行った場合、正しく設定されていないと画面が真っ黒になったままだったり、画面表示が崩れて内容が読み取れなくなることがある。そんなときXを終了させるキーコンビネーション、Ctrl＋Alt＋Backspaceを覚えておこう。この組み合わせを押せば、Xは終了する。

turboxcfgを使う

turboxcfgの起動は以下のように入力する。

```
# /usr/sbin/turboxcfg
```

turboxcfgは設定項目を画面単位で順序よく表示してくれる。設定手順に

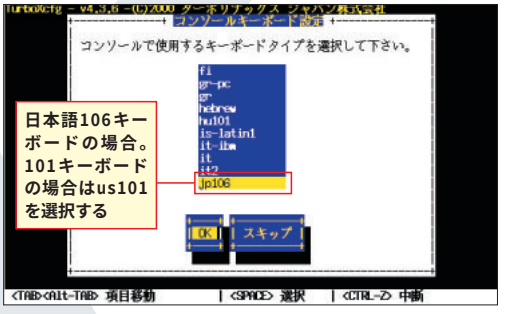
ついてユーザーが戸惑うことはないだろう。ツールの操作はキーボードで行う。画面内の各項目間はTABキーで移動し、リストボックス内は矢印キー、チェックボックスのチェックはスペースキー、ボタンを押す操作はTABキーでボタンを選んでEnterキーだ。

まずは、キーボードとマウスの設定から始まる（**画面T-1**～**画面T-3**）。それが終わると、ビデオカード（チップ）の自動検出に移る。検出されると**画面T-4**のように表示される。

ここで自動認識に失敗したり、**画面T-4**で［手動設定］を選択した場合は、**画面T-5**のようになるので、正しいビデオビデオカード（チップ）を指定して次へ進む。カードを設定するとビデオメモリ容量の確認に進む（**画面T-6**）。搭載メモリ容量を指定しよう。

次はモニタ（turboxcfgではディスプレイと表記される）の設定だ。turboxcfgではメーカーごとに階層構造になっているので、まずメーカーを選択する（**画面T-7**）。［OK］を押すとそのメーカーの製品を選択する画面になるので製品名を選択しよう。

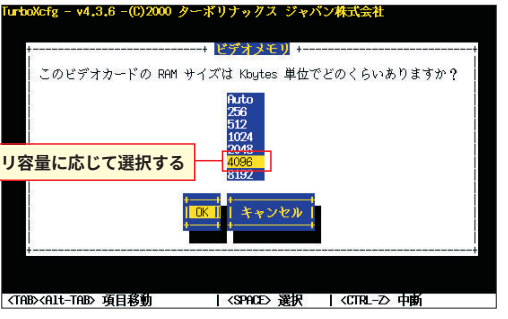
該当する製品名がない場合は、［キ



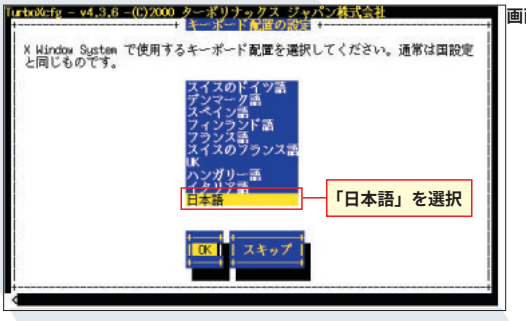
画面T-1　コンソールキーボードの設定

画面T-5　ビデオカードの選択

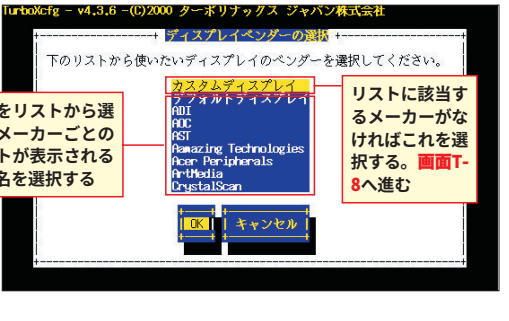
画面T-2　キーボードモデル設定



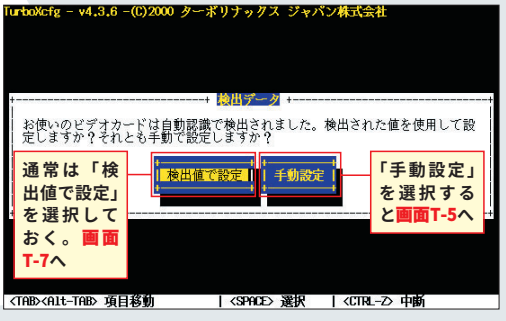
画面T-6　ビデオメモリ



画面T-3　キーボード配置の設定



画面T-7　ディスプレイベンダーの選択



画面T-4　検出データ
自動検出に成功するとこの画面が表示される。検出されないといきなり画面T-5が表示される。

画面T-8　カスタムディスプレイ設定

ャンセル］を押して**画面T-7**に戻り、「カスタム」を選択する。すると、**画面T-8**のようになる。ここでカスタムモニタに任意の名前を付けるのだ。［OK］を押すと**画面T-9**のようになり、モニタで利用できる最大解像度を入力する。［OK］ボタンを押して**画面T-10**に進み、同期周波数を選ぶ。もし周波数が不明な場合は、無茶をせず、リストからSVGAや1024×768を選択したほうがよい。むやみに高周波を選択するとモニタを壊すことがあるので注意しよう。モニタの周波数がわかっているなら、リストから「カスタムの水平同期幅」を選択し、「水平周波数帯のカスタム値」の欄に入力する。**画面T-11**に進み垂直同期周波数を選択する。不明な場合は最低値である「50-70」を選んでおこう。

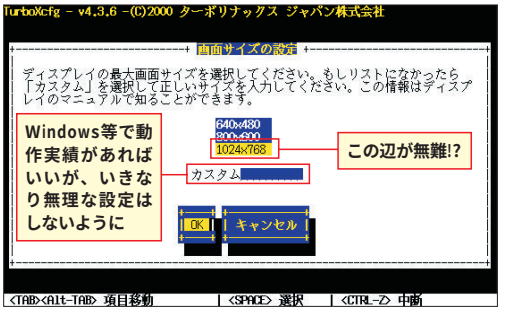
モニタの設定が終わるとデフォルトの色数の設定である（**画面T-12**）。8bpp～32bppの中から選択する。8bppは256色、16bppは6万5000色、24bppと32bppはともに1600万色である。通常、同じ1600万色を表示する場合でも32bppのほうが処理効率がよく動作も速い（反面メモリの利用効率が悪い）。

色の設定を終えると、一瞬画面が切り替わり、ビデオモードとクロックの自動検出を行い、画面サイズの設定に進む（**画面T-13**）。ここで複数のサイズを選択しておくと、X利用中に画面サイズを切り替えることができる。［OK］ボタンを押すと、デフォルトの画面サイズ選択に進む。複数選んだうちのどれをデフォルト値にするかを選択するのだ。

次にフォントパスの設定に進む。ここでは、フォントパスの変更については省略する。［デフォルトでよい］を選択し［100DPI］を選択しておこう。

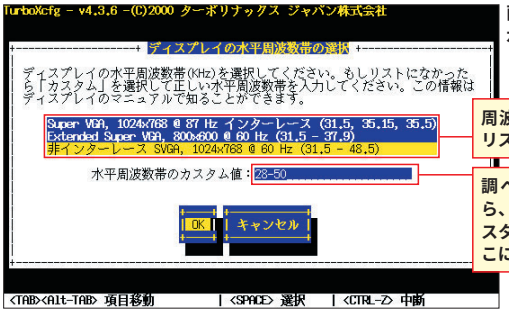
設定が終わると、Xの動作テストに入る。設定に問題なければ、複数の画面サイズを順に切り替えてテスト画面が表示される（**画面T-14**）。［Next］ボタンで別の画面モードに切り替わる（複数選択時）。［Quit］で終了する。

グラフィカルログインに切り替えるかどうかを設定して、最後に設定内容をXF86Configに書き込んで終了だ。



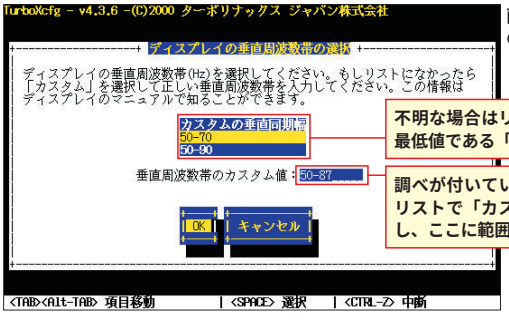
画面T-9　画面サイズの設定

画面T-12　デフォルト色数

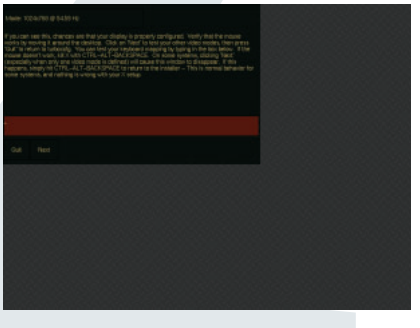


画面T-10　ディスプレイの水平周波数帯の選択

画面T-13　画面モードの設定



画面T-11　ディスプレイの垂直周波数帯の選択



画面T-14　Xの動作テスト

Xconfiguratorを使う

Xconfiguratorの起動は以下のように入力する。

```
# /usr/X11R6/bin/Xconfigurator
```

Xconfiguratorもturboxcfgと同様、設定項目を画面単位で順序よく表示してくれる。ただし、XconfiguratorではXで使用するキーボードやマウスの設定は行わないため、ビデオ周りの設定のみとなる。

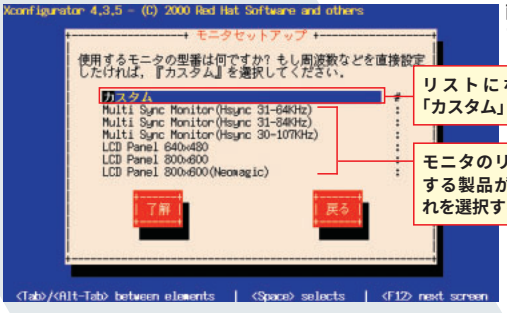
起動画面に［了解］と答えると、すぐにビデオカード（チップ）の自動検出が行われ、成功すると**画面R-1**が表示される。使用しているビデオカードがサポートされていれば、ここで検出に失敗することはほとんどない（たまに間違えることはあるようだが）。表示された情報を確認し、［了解］ボタンで次の画面に進もう。

次にモニタの設定だ（**画面R-2**）。モニタのメーカ名ごとの型番がリストになっているので、該当するものがあればそれを選択する。このリストは非常に長いのでメーカー名の頭文字を押してジャンプするといい。たとえばSony製のモニタを選択したければSキーを押す。Sで始まるメーカーの先頭にジャンプできる。

リストに該当する製品がなければ「カスタム」を選択して、同期周波数選択の画面に進む（**画面R-3**）。ここには画面解像度を基準に、いくつかの候補がリストされている。最も近いと思われるものを選択しよう。選択すると実際の垂直同期周波数を選択する画面に進む（**画面R-4**）。

モニタの設定が終了すると、ビデオチップやメモリ容量の自動検出を行うかどうか問い合わせてくる（**画面R-5**）。ここで利用可能なビデオモード（画面サイズ）を判断するのだ。［検出］を選択すると画面が一瞬切り替わり、検出結果をもとにビデオモードの選択画面に進む。検出できない場合や［検出しない］を選択すると、ビデオメモリの容量とクロックチップの設定画面が順に表示される（**画面R-6**、**R-7**）。ク

画面R-1　PCI機器の検出

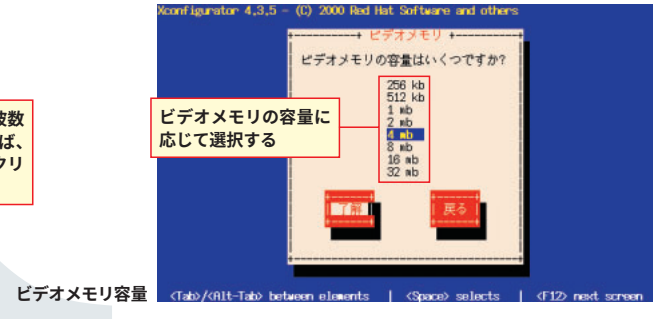


画面R-2　モニタのセットアップ

画面R-3　カスタムモニタセットアップ（水平同期）

画面R-4　カスタムモニタセットアップ（垂直同期）

画面R-5　解像度と色数の自動検出



画面R-6　ビデオメモリ容量

推奨値である「クロックチップの設定をしない」を選択しておけばいい

画面R-7　クロックチップの設定

利用したいビデオモードは複数チェックしておく

画面R-8　ビデオモードの選択
利用するビデオモードにチェックマーク（＊）を付ける。各項目間の移動はTABキーと矢印キー、チェックマークはスペースキーで付ける。

ロックチップについては、「クロックチップの設定はしない」を選択する。特別なケースを除いて指定する必要はない（画面からもわかるように「設定しない」が推奨値になっている）。

ビデオモード選択画面では、Xで利用したい画面サイズにすべて［＊］を付けて選択しておく（**画面R-8**）。選択したモードはX利用中に切り替えることができる。

最後に、Xの動作テストが行われる。問題なく動作すれば、設定を保存して終了だ。

xf86configを使う

xf86configの起動は以下のように入力する。

```
# /usr/X11R6/bin/xf86config
```

xf86configは日本語端末上でも全文英語表示となる。xf86configを使って一発で完璧な設定をこなせる人はなかなかいない。結局、あとから/etc/X11/XF86Configをエディタで修正することになるはずだ。したがって、ここではポイントだけを紹介する。

まず最初はマウスプロトコルの設定だ。選択肢がリストされるので、その番号を入力すればいい。一般的なPS/2マウスなら「4」と入力する。次にマウスのデバイスファイルを指定する。通常、マウスのポートは/dev/mouseにリンクされているので、これを指定するといい。

次はキーボードの設定である。最初のXKEYBORDを利用するかという問い合わせは「y」。keymapは日本語106キーが選択できないので、「US101」か「None of the above」を選ぼう。CountryもJapanがないので「USA」を選んでおく。

いよいよビデオ周りの設定だ（**画面C-1**）。まずはモニタから。水平同期周波数の一覧が表示される。番号で選択しよう。もし、周波数の調べがついているなら、「11」を選択して値を入力しよう。次に垂直同期周波数の一覧が表示されるのでこれも番号で選択する。

```
    ......
   hsync in kHz; monitor type with characteristic modes
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
 4  31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56 Hz
 5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
 8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
 9  31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
10  31.5 - 95.0; Monitor that can do 1280x1024 @ 85 Hz
11  Enter your own horizontal sync range

Enter your choice (1-11): 6

    ......


 1  50-70
 2  50-90
 3  50-100
 4  40-150
 5  Enter your own vertical sync range

Enter your choice: 1

    ......


The strings are free-form, spaces are allowed.
Enter an identifier for your monitor definition: MySony
Enter the vendor name of your monitor: Sony
Enter the model name of your monitor: CPD-1304
```

周波数がわかるなら「11」を選択して範囲を入力する

周波数がわかるなら「5」を選択して範囲を入力する

名前を付ける

メーカー名を入力

製品名を入力

画面C-1　ディスプレイモニタの設定

```
Do you want to look at the card database? y        ← データベースを見るので「y」

  0  2 the Max MAXColor S3 Trio64V+                 S3 Trio64V+
  1  3DLabs Oxygen GMX                              PERMEDIA 2
  2  3DVision-i740 AGP                              Intel 740

     ~~
     ~~

446  Miro Video 20SV                                S3 968
447  NVIDIA GeForce 256 (generic)                   GeForce 256
448  NVIDIA GeForce DDR (generic)                   GeForce DDR
449  NVIDIA Riva 128 (generic)                      Riva 128

Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.

447        ← リスト表示の途中でも、見つかった番号を入力していい

Your selected card definition:

Identifier: NVIDIA GeForce 256 (generic)
Chipset:    GeForce 256                  ← 選択したカード名、チップセット、使用されるXサーバの種類が表示される
Server:     XF86_SVGA
Do NOT probe clocks or use any Clocks line.
```

画面C-2　ビデオカードの設定

```
     ......

Which one of these screen types do you intend to run by default (1-5)? 5
                                                   ← デフォルトのサーバを選択
     ......

Do you want me to set the symbolic link? y         ← Xの実体であるXF86_SVGAとリンクを張る
     ......

How much video memory do you have on your video card:

 1  256K
 2  512K
 3  1024K
 4  2048K
 5  4096K
 6  Other

Enter your choice: 6        ← 最近のビデオカードなら確実に「6」を選択することになる

     ......

                                                   ← 名前を付ける
The strings are free-form, spaces are allowed.
Enter an identifier for your video card definition: gf256
You can simply press enter here if you have a generic card, or want to
describe your card with one string.
Enter the vendor name of your video card: nvidia        ← メーカー名を入力
Enter the model (board) name of your video card: gforce256   ← 製品（チップ）名を入力

     ......

Just press enter if you don't want a Clockchip setting.
What Clockchip setting do you want (1-12)?   ← クロックチップの設定はしないので、Enterキーのみ入力
```

画面C-3　Xサーバの選択

こちらも値がわかっている場合には、「5」を選択してそれを入力する。最後に、このモニタの設定に任意の名前を付ける（ここで付けた名前については**P.73を参照**）。

次はビデオカードの設定だ。カードのデータベースを参照するかどうかの問い合わせがあるので、「y」と入力すると、膨大なカードリストが表示される（**画面C-2**）。途中で目的のカードが見つかったら、その番号を入力する。選択したカードの定義が確認のため表示される。

ビデオカードの設定が終わったらデフォルトのサーバの選択だ（**画面C-3**）。ここでは、モノクロ、VGA16色、SVGA256色、アクセラレートサーバと、カード定義から選択されるSVGAサーバの5つから選べる。とりあえず「5」にしておけばいいだろう。続くシンボリックリンクを張るかという問い合わせは「y」。ビデオメモリの容量は4Mバイトまでならリストから選択、それ以外なら「Other」を選び値を入力する。ビデオカード（チップ）の定義にも名前を付ける。クロックチップの設定は行わないので、Enterキーのみ入力する。

最後の設定は、Xの画面サイズだ（**画面C-4**）。ここで一度Xを検出モードで起動する。それが済んだら、デフォルトの色数を選択する。そして、画面モード（解像度）を選択する。ここでは複数のモードを選択することができる。リストの番号をつなげて入力すればいい。仮想画面を利用するかどうかを問い合わせてくるので、利用するなら「y」と入力する。

以上で、設定は終了である。設定内容をファイルXF86Configに保存するかと問い合わせてくるので、もちろん「y」と入力する。

```
Note that 16, 24 and 32bpp are only supported on a few configurations.
Modes that cannot be supported due to monitor or clock constraints will
be automatically skipped by the server.

 1  Change the modes for 8pp (256 colors)
 2  Change the modes for 16bpp (32K/64K colors)
 3  Change the modes for 24bpp (24-bit color, packed pixel)
 4  Change the modes for 32bpp (24-bit color)
 5  The modes are OK, continue.

Enter your choice: 2          ← デフォルトで使用する色数を選択

    ......

Please type the digits corresponding to the modes that you want to select.
For example, 432 selects "1024x768" "800x600" "640x480", with a
default mode of 1024x768.

          Which modes? 43     ← 複数のモードを選ぶにはこのように。例は「4」と「3」

    ......

Do you want a virtual screen that is larger than the physical screen? n
                              ← 仮想画面の利用
```

画面C-4　画面サイズの設定

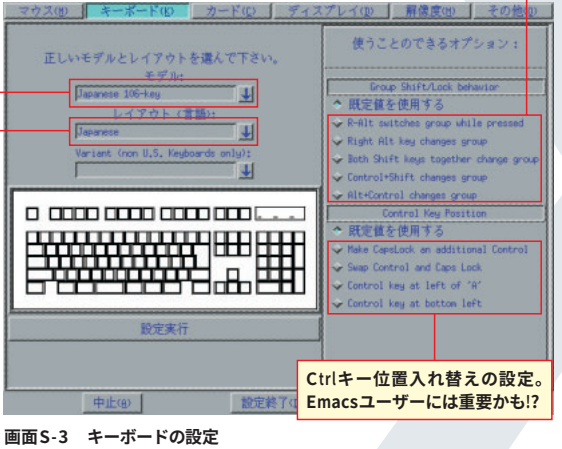
XF86Setupを使う

XF86Setupの起動は以下のように入力する。

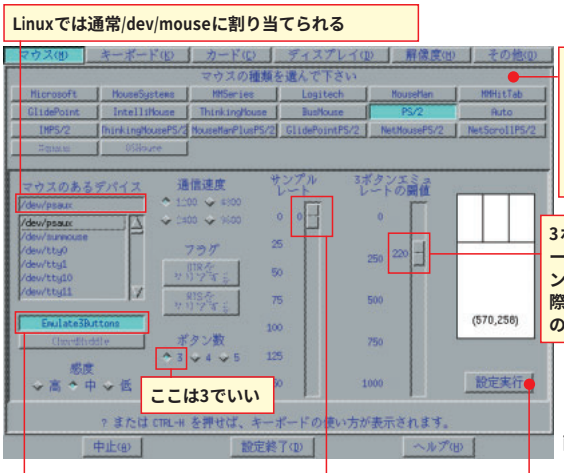
```
# /usr/X11R6/bin/XF86Setup
```

すでに設定ファイル/etc/X11/XF86Configが存在する場合、それをデフォルトとして使うか？　と英語で質問が表示される。「n」と答えると、VGAモードのXが起動し、XF86Setupの設定画面になる（**画面S-1**）。X上で動作する設定画面は、最初にマウスの設定さえしてしまえば、以降マウスで操作可能となる。

画面S-1　XF86Setupの起動画面

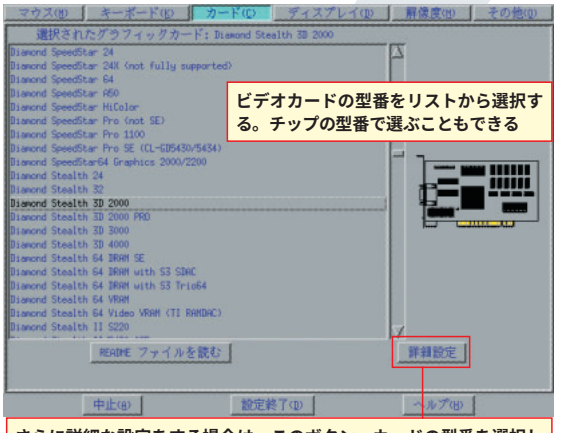


画面S-3　キーボードの設定



画面S-2　マウスの設定

画面S-4　ビデオカードの設定



XF86Setupは、ほかのツールのように順序よくナビゲートしてはくれないが、逆にどの順で設定してもかまわない。ここでは、画面上部に並ぶボタンを左から順にマウスでクリックして、設定していこう。

まずは、マウスの設定だ（**画面S-2**）。マウスが未設定なことも考慮して、起動画面で単にEnterキーを押せばマウス設定画面に移行できるようになっている。TABキーや矢印キー、スペースキーで設定しよう。マウスの設定は画面上の［設定実行］ボタンを押すことで、すぐに反映される。正しく設定できていれば、以降はマウスを使って設定できる。

次にキーボード（**画面S-3**）。キーボードの種類やCtrlキーの位置の入れ替えなどが設定できる。

そしてビデオカード（**画面S-4**）だ。ここでビデオカード（またはチップ）を選択する。ビデオカードの画面では、さらに［詳細］ボタンをクリックすると**画面S-5**のようになる。ここで、Xサーバや、ビデオチップ、ビデオRAMの容量等を指示することもできる。**画面S-4**でカードやチップを選択してあり、**画面S-5**で［Probed］ボタンが押された状態ならば、メモリ容量は検出値が使用される。

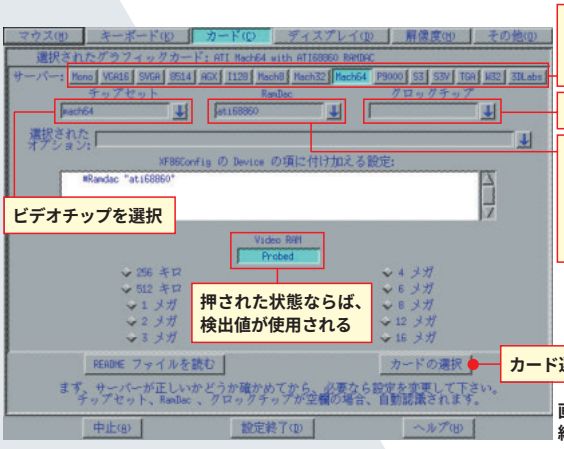
［モニタの設定］画面（**画面S-6**）では、ほかの設定ツールのような製品名リストがないので、同期周波数をリストから選択するか、数値を入力するしかない。

［解像度の設定］では解像度と色数を設定する（**画面S-7**）。

［その他の設定］では、Xの終了キーや、解像度切り替えの有効／無効等を設定する（**画面S-8**）。Ctrl＋Alt＋Backspaceを有効にするチェックは必ず有効にしておこう。Xの設定中は言うに及ばず、ふだんXを利用しているときにも、Xを強制終了させたい場面がかならず訪れるはずだ。

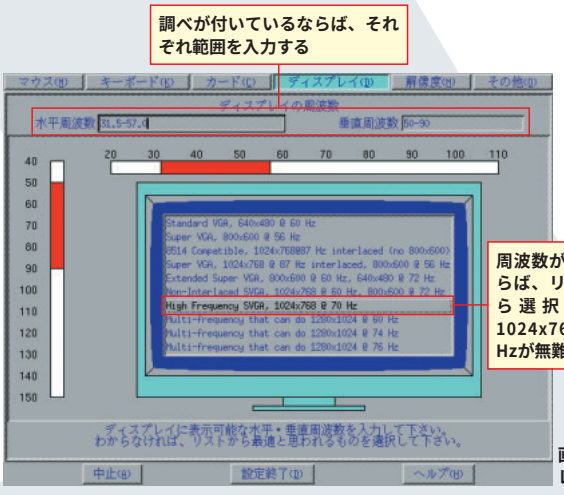
すべて設定したら画面下の［設定終了］ボタンをクリックする。

これで、ひと通り設定ツールの使い方を説明したわけだが、事前に設定する値がわかっていれば、操作自体はそれほど難しいものではない。ツールを使う前に必ず、最初に説明した項目をチェックしておくようにしよう。

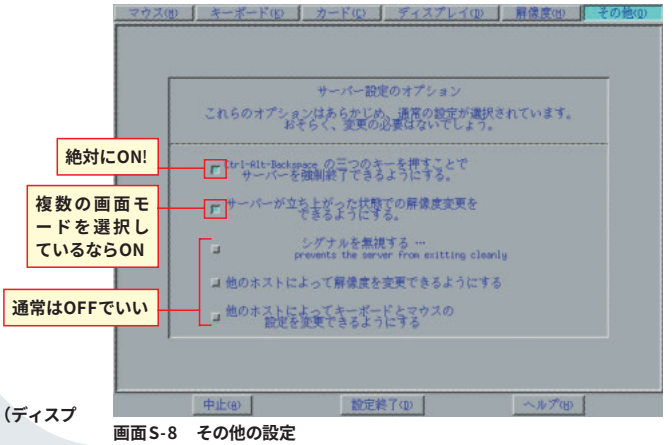


画面S-5　ビデオチップの詳細設定

画面S-7　解像度の設定



画面S-6　モニタ（ディスプレイ）の設定



画面S-8　その他の設定

XFree86の設定ファイル XF86Config徹底解剖

文：山岸典将
Text : Norimasa Yamagishi

XFree86の設定は、XF86Configというテキストファイルで行う。現在では、GUIプログラムのXF86Setupや、ディストリビューション付属のさまざまなソフトウェアを使い、簡単にXF86Configを作成できる。このため、何もないところからXF86Configを書き起こすという人はほとんどいないだろう。

しかし、GUIのプログラムでは設定できない項目もあるし、マシン環境によっては、GUIのプログラムが起動しないことさえある。そういった場合はXF86Configを自分の手で編集する必要がある。

XFree86 3.3系と4.0系とでは、XF86Configの書式が一部異なっている。ここでは、まず現在普及している3.3系について解説し、最後に4.0系統の変更点について解説しよう。

XF86Configはどこにある？

XF86Configを編集する前に、このファイルをどのディレクトリに置くべきなのかを知らなくてはならないだろう。実は、XF86Configを置くことのできるディレクトリは複数ある。これらのディレクトリには優先順位があり、Linuxでは以下の順になっていることが多い。

```
/etc/XF86Config
/usr/X11R6/lib/X11/XF86Config.<hostname>
/usr/X11R6/lib/X11/XF86Config
```

Linuxではほとんどの場合/usr/X11R6/lib/X11/XF86Configが使われている。実は、このファイルは/etc/X11/XF86Configへのリンクとなっているので、設定の際にはどちらを編集してもよい。

ただし、rootユーザーのの場合のみ、ホームディレクトリのXF86Configが最優先の設定ファイルとなる。通常の運用時に、/rootディレクトリにXF86Configを置いて使うことはまずないだろう。しかし、XF86Configをテストするときには、ホームディレクトリの直下にXF86Configを置いていろいろ試してみるというのは有効な手段だ。そうすれば、/etc/X11にあるXF86Configを変更せずに、一時的に設定を変更して、いろいろ試すことができる。

なお、通常XF86Configは一般ユーザーは書き込みできないパーミッションになっている。まず、スーパーユーザになろう。実験をする場合は、スーパーユーザー（root）になり、XF86Configを/rootディレクトリにコピーする。あとはコピーしたファイルを編集していろいろ試してみればよい。XF86Configが作成されていなければ、あらかじめ用意されているサンプルを流用しよう。次のようにしてホームディレクトリにコピーする。

```
# cp /usr/X11R6/lib/X11/XF86Config.eg ~/XF86Config
```

XF86Configの書式

XF86Configの内容は、いくつかのセクションに分かれている。1つのセクションは「Section "名前"」から「EndSection」で囲まれる範囲までだ。

なお、セクション内に指定できる項目は非常に多く、ここですべてについて解説することはできない。そのため、よく使われると思われる項目についてのみ解説する。

以下の解説記事中、［●］の項目については必須。［○］の項目は環境によっては必須。［▲］の項目は必要に応じて設定する（なくても動作する）項目となっている。

なお、XF86Configでは、行中の“＃”以降はコメントとみなされるので、一時的にオプションを変更する際などに活用してほしい。

また、オプションに対してパラメータを設定するときには、項目の前後を「"」で囲むものと、囲まないものがあることに注意してほしい。これに気が付かないと、いつまでたってもエラーを解消できないで悩むことになる。

さて、XF86Configの書き方を解説する前に、XF86Configのエラーの見つけ方を書いておこう。今までに、XF86Configにエラーがあり、Xがうまく立ち上がらなかった経験があるなら、画面にエラーメッセージらしきものが表示されたのを見たことがあるだろう。ただし、このメッセージは非常に大量に出力されるため、画面の外まで一気

に流れていってしまう。そこで、エラーメッセージを“x.log”というファイルに保存するには次のようにする。

```
$ startx -- -probeonly &> x.log
```

実はXFree86のXサーバには、probe機能という調査機能がある。“-probeonly”オプションは、Xを立ち上げずに調査のみを行うオプションだ。このオプションはXコマンドのオプションで、startxコマンドのオプションではないため、startxコマンドのオプションとして“--”を指定し、“-probeonly”オプションをXコマンドに渡してやる必要がある。また、検出結果は標準エラー出力に出力されるため、通常のリダイレクトの「>」ではなく「&>」を使っている。これでx.logファイルにすべての調査結果がレポートされる（**リスト1**）。この結果は実際にstartxコマンドでXを立ち上げたときに表示されているものと同じだ。XF86Configのどの部分でエラーが発生しているかだけでなく、ハードウェア情報も得ることができる。XF86Configの設定を自分で変更しようとするときには、必須のコマンドといってもいいので、ぜひ覚えておいてほしい。

なお、XFree86 4.0では、Xの起動メッセージが“/var/log/XFree86.0.log”などのファイルに保存されるので、エラーが起きたらこのファイルを参照しよう。

XF86Configの内容

では、セクションごとにXF86Configの内容を説明する。

■Filesセクション

Filesセクションは、Xが使用するフ

▼リスト1　エラーが起こったときのstartxコマンドの出力（XFree86 3.3.6の例）

```
                              :
                              :
Configured drivers:
  SVGA: server for SVGA graphics adaptors (Patchlevel 0):
      NV1, STG2000, RIVA 128, RIVA TNT, RIVA TNT2, RIVA ULTRA TNT2,
            RIVA VANTA, RIVA ULTRA VANTA, RIVA INTEGRATED, GeForce 256,
      GeForce DDR, Quadro, ET4000, ET4000W32, ET4000W32i, ET4000W32i_rev_b,
                              :
                              :
      ct64300, mediagx, V1000, V2100, V2200, p9100, spc8110, i740, i740_pci,
      Voodoo Banshee, Voodoo3, i810, i810-dc100, i810e, i815, smi, generic
(using VT number 7)

XF86Config: /root/XF86Config
(**) stands for supplied, (--) stands for probed/default values
(**) XKB: disabled
(**) XKB: rules: "xfree86"
(**) XKB: model: "jp106"
(**) XKB: layout: "jp"
(**) Mouse: type: PS/2, device: /dev/mouse, buttons: 3
(**) Mouse: 3 button emulation (timeout: 50ms)
(**) SVGA: Graphics device ID: "Matrox Millennium G200 4MB"
(**) SVGA: Monitor ID: "LCD Panel 1024x768"
(--) SVGA: Mode "640x480" needs hsync freq of 31.47 kHz. Deleted.
                              :
                              :
(--) SVGA: Mode "1280x1024" needs hsync freq of 91.15 kHz. Deleted.
(**) FontPath set to "unix/:-1"
(--) SVGA: PCI: Matrox MGA G200 AGP rev 3, Memory @ 0xe3000000, 0xe2000000
(--) SVGA: Linear framebuffer at 0xE3000000
(--) SVGA: MMIO registers at 0xE2000000
(--) SVGA: Video BIOS info block at 0x000c7540
(--) SVGA: Found and verified enhanced Video BIOS info block
(--) SVGA: detected an SGRAM card
(--) SVGA: chipset:  mgag200
(**) SVGA: videoram: 8192k
(**) SVGA: Option "dac_8_bit"
(**) SVGA: Using 8 bits per color component
(**) SVGA: Using 8 bpp, Depth 8, Color weight: 888
(--) SVGA: Maximum allowed dot-clock: 250.000 MHz
(--) SVGA: There is no mode definition named "1600x1200"
(--) SVGA: Removing mode "1280x1024" from list of valid modes.
(--) SVGA: There is no mode definition named "1152x864"
(--) SVGA: Removing mode "1024x768" from list of valid modes.
(--) SVGA: There is no mode definition named "1024x600"
(--) SVGA: Removing mode "800x600" from list of valid modes.
(--) SVGA: There is no mode definition named "640x480"

Fatal server error:
No valid modes found.

When reporting a problem related to a server crash, please send
the full server output, not just the last messages

X connection to :0.0 broken (explicit kill or server shutdown).
```

使用しているXサーバ
Xサーバが対応しているビデオチップ
使用した設定ファイル
キーボードの設定
マウスの設定
モニタで使用できるビデオモード
フォントファイルの設定
ビデオカードの調査結果
無効なモードが調査されている
有効なモードが1つもないので致命的なエラーとなる！

ファイルのパス（置き場所）を指定する（**リスト2**）。ここで指定されているファイルがまったく見つからないとXサーバは起動しない。

RgbPath［●］

Xのカラーデータベースのパスを指定する。通常は"/usr/X11R6/lib/X11/rgb"でよい。

FontPath［●］

フォントの検索パスを指定する。複数のディレクトリをカンマ区切りで書くことができる。また、FontPathという項目自体を複数指定してもよい。先に指定したものから順に検索される。フォントサーバ（xfs）を利用する場合は、PontPathに"trans/hostname:port_number"という形で指定する。たとえば、"unix/:7100"などはフォントサーバを利用する設定だ。

ModulePath［○］

ダイナミックモジュールの存在するパスを指定する。

■Moduleセクション

Moduleセクションは、ロードするダイナミックモジュールを定義するためのセクションだ。タブレットなどの拡張入力機器をサポートするモジュールを使う場合はここで指定する（**リスト3**）。モジュールを使わないのであればセクション自体必要ない。

Load［▲］

ロードするモジュールを指定する。

■ServerFlagsセクション

ServerFlagsセクションではXサーバのオプションを指定する（**リスト4**）。

NoTrapSignals［▲］

不当なシグナルを受けた場合コアダンプし異常終了する。デバッグ用のオプションなので通常は指定しない。

DontZap［▲］

XFree86はCtrl＋Alt＋BackSpaceにより終了するが、これを禁止する。通常、設定する必要はないが、アプリケーションソフトウェアでこのキーコンビネーションを利用したい場合に設定する。

DontZoom［▲］

XFree86はCtrl＋Alt＋（テンキーの）“＋”か“－”を同時に押すことにより解像度を変更できるが、これを禁止する。これも、通常は設定する必要はない。

■Keyboardセクション

Keyboardセクションは、文字通りキーボードの設定を記述する（**リスト5**）。

Protocol［●］

特殊なキーボードを使っていない限り、"Standard"で問題ないはずだ。

AutoRepeat［▲］

キーボードのオートリピートの設定を変更する。

LeftAlt、RightAlt、ScrollLock、RightCtl［▲］

これらのキーに対する機能の割り当てを指定する。

XkbRules、XkbModel、XkbLayout［▲］

これらはキーボードのレイアウトを指定する。XkbRulesは"xfree86"がデフォルトだが、ほかの2つはそれぞれ使

▼リスト2　Filesセクションの例

```
Section "Files"
   RgbPath       "/usr/X11R6/lib/X11/rgb"
   FontPath      "/usr/X11R6/lib/X11/fonts/misc:unscaled"
   FontPath      "/usr/X11R6/lib/X11/fonts/75dpi:unscaled
   FontPath      "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
   FontPath      "/usr/X11R6/lib/X11/fonts/Type1"
   FontPath      "/usr/X11R6/lib/X11/fonts/Speedo"
   FontPath      "/usr/X11R6/lib/X11/fonts/misc"
   FontPath      "/usr/X11R6/lib/X11/fonts/75dpi"
   FontPath      "/usr/X11R6/lib/X11/fonts/100dpi"
   ModulePath    "/usr/X11R6/lib/modules"
EndSection
```

▼リスト3　Moduleセクションの例

```
Section "Module"
   Load  "xf86Wacom.so"
EndSection
```

▼リスト4　ServerFlagsセクションの例

```
Section "ServerFlags"
    DontZoom
EndSection
```

▼リスト5　Keyboardセクションの例

```
Section "Keyboard"
   Protocol     "Standard"
   AutoRepeat   500 30
   XkbRules     "xfree86"
   XkbModel     "jp106"
   XkbLayout    "jp"
   XkbOptions   "ctrl:swapcaps"
EndSection
```

っているキーボードに合わせて変更しよう。

XkbOptions［▲］

"ctrl:swapcaps"を指定すると左CtrlとCaps Lockキーを入れ替える。

■Pointerセクション

マウスやタブレットといったポインティングデバイスを設定するためのセクションだ（**リスト6**）。PS/2マウスを動作させるだけならば、ProtocolとDeviceを指定するだけでよい。

Protocol［●］

マウスのプロトコルを指定する。現在使われているマウスのほとんどはPS/2プロトコルで動作するだろう。もし、USB接続のマウスを使っていたとしても"PS/2"で問題ないはずだ。MicrosoftのIntelliMouseを使う場合は"IMPS/2"を選ぶと、ホイールが有効になるが、他社製のホイール付きマウスで"IMPS/2"を選ぶと痛い目に合うこともある。ホイール動作に対応したアプリケーションもほとんどないので"PS/2"にしておくのが無難だろう。

シリアルマウスの場合は、よほど古いものでなければ、"Auto"で問題ないだろう。

なお、"Logitech"は、かつて販売されていたLogitech独自仕様のバスマウスを示す。現在販売されているPS/2接続のLogitechマウスは、"PS/2"を選んでおけばよい。

▼リスト6　Pointerセクションの例

```
Section "Pointer"
   Protocol      "PS/2"
   Device        "/dev/mouse"
   Emulate3Buttons
   Emulate3Timeout 50
   Resolution    100
EndSection
```

Device［●］

Linuxシステムに認識されているマウスのデバイス名を指定する。ほとんどの場合は"/dev/mouse"のはずだ。

BaudRate［○］

シリアルマウスを使う場合に通信速度を指定する。

Resolution［▲］

マウスを動かしたときの画面上でのカーソルの移動量を指定する。大きければ大きいほど、カーソルの移動量も大きくなる。

Emulate3Buttons［▲］

Xは3ボタンマウスでの使用を前提としている。2ボタンマウスを使用している場合、このオプションを指定すると、左右ボタンの同時押しで中央のボタンを押したことになる。

Emulate3Timeout［▲］

3ボタンマウスのエミュレーションにおいて、どれだけの時間内に2つのボタンを同時に押すと中央のボタンが押されたことにするかを設定する。指定の単位はミリ秒。指定がなければ50ミリ秒になる。

■Monitorセクション

Monitorセクションは、ディスプレイの設定をするためのセクションだ（**リスト7**）。XF86Configの中でも難解なセクションだといえる。このセクションは複数書くこともできる。すなわち、状況によって使用するディスプレイを変更する場合は、Identifierの異なるMonitorセクションを複数用意しておき、切り替えることが可能になる。

Identifier［●］

ディスプレイの識別名を指定する。この識別名は後述のScreenセクションで利用する。複数のMonitorセクションを記述する場合、同じIdentifierがあってはいけない。単に区別するためのものなので“ levin ”、“ trueno ”などすきな名前をつけてもかまわない。

VendorName［▲］

ディスプレイのメーカー名を指定する項目だ。

ModelName［▲］

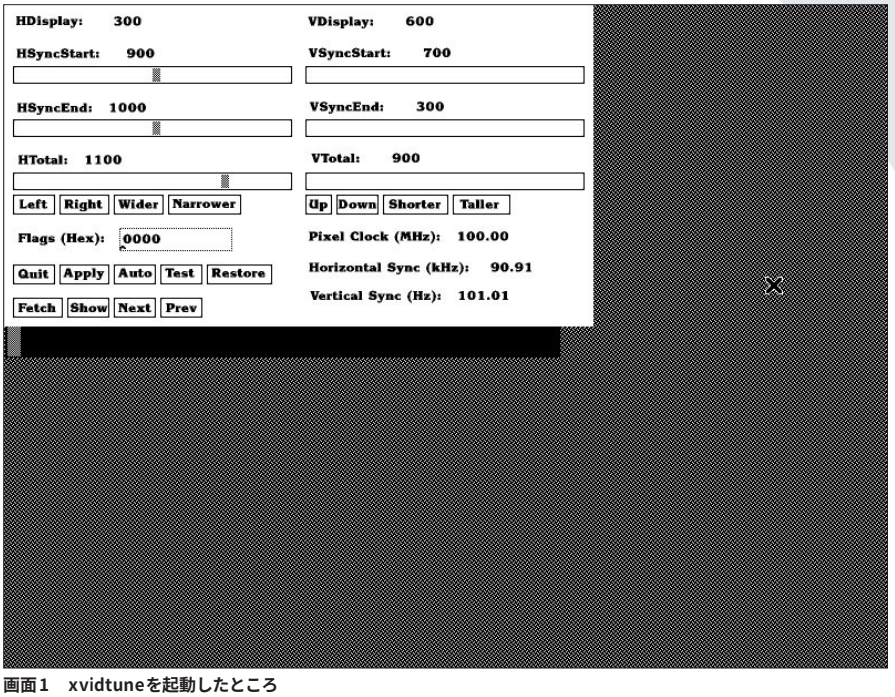
ディスプレイの型番を指定する。

HorizSync［●］

水平同期信号の周波数を指定する。単位はkHz。ディスプレイのマニュアルで仕様をチェックしよう。“ － ”（ハイフン）を使った範囲指定でも、複数の周波数を“ , ”（カンマ）区切りで並べて書いてもよい。

▼リスト7　Monitorセクションの例

```
Section "Monitor"
   Identifier      "RD17G3"
   VendorName      "MITSUBISHI"
   ModelName       "RD17G3"
   HorizSync       30-95.0
   VertRefresh     50-152
   Modeline  "1024x768" 75.00 1024 1048 1184 1328 768 771 777 806 -hsync -vsync
   Modeline  "800x600"  60.75 800 864 928 1088 600 616 621 657 -hsync -vsync
   Modeline  "640x480"  36.00 640 696 752 832 480 481 484 509 -hsync -vsync
EndSection
```



画面1　xvidtuneを起動したところ
xvidtuneで表示を調整した後、それぞれの値を確認し再度XF86Configに記述しよう。

VertRefresh［●］

垂直同期信号の周波数を指定する。単位はHz。指定方法はHorizSyncと同じだ。これもディスプレイのマニュアルで仕様を調べて設定しよう。

Modeline［●］

利用できる解像度を記述する。Monitorセクションを難解にしているのがこのModelineだ。1つめのパラメータでは解像度の識別名を指定する。これもIdentifierと同様にScreenセクションで参照するためのものなので、すきな名前をつけてかまわないが“1024x768”のように解像度を名前として指定しておくのが一般的でわかりやすいだろう。

2つめにくる数字が、その解像度で使用するドットクロック値。

そして、それに続く4つの数字が水平方向の周波数設定、そのあとの4つが垂直方向の周波数設定だ。これらの数値は、ディスプレイの仕様詳細からめんどうな計算をして求める必要がある。それぞれの組の1つめが解像度のドット数となるのだが、それ以外の値については、xvidtuneというソフトウェアを使用してModeline行を作成してほしい。そして、行末にはオプションをつけることができる。たとえば、“-hsync”や“-vsync”は同期信号の極性を指定するオプションだ。

ただし、xvidtuneはX上で動作するソフトだ。Xが起動さえしないという場合は、/usr/X11R6/lib/X11/doc/README.Config内に一般的な値の例が載っているので、それを参考にして暫定の値を設定してXを立ち上げてxvidtuneで調整をしよう（**画面1**）。xvidtuneでは、画面のサイズ（幅、高さ）、位置（上下左右）などを調整できる。調整が済んだら、画面に表示されているパラメータの値をメモし、Modeline行に記述する。xvidtuneの値とModeline行の値の対応は**リスト8**のとおりだ。

■Deviceセクション

Deviceセクションでグラフィックスカードの設定を行う（**リスト9**）。最近では、多くのグラフィックスカードを自動的に検出できるようになっている。そのため、特殊なカードや古いものでなければIdentifierを指定するだけで動作することも多い。

オプションは、グラフィックスカードごとに必要となるもの、書いてはいけないものがある。基本的なオプションに関しては以下で説明するが、個々のグラフィックスカード固有のオプションについては、ドキュメントを読んでほしい。現在では、ドキュメントも日本語化されているので、英語が苦手でも大丈夫だ。ドキュメントの中にはサーバのバグおよびバグの回避方法なども掲載されている。

このセクションも複数記述することができる。ただし実際に有効になるのは、Screenセクションで指定された

▼リスト9　Deviceセクションの例

```
Section "Device"
    Identifier  "Matrox G200"
    VendorName  "Matrox"
    BoardName   "G200 AGP"
    VideoRam    8192
EndSection
```

▼リスト8　Modelineオプションのxvidtuneとの対応

```
識別名 PixelClock HDisplay HsyncStart HsyncEnd HTotal VDisplay VsyncStart VsyncEnd VTotal Flags
```

Deviceだけだ。実際に複数のデバイスを記述することはほとんどないと思われるが、実験マシンなどで頻繁にグラフィックスカードを入れ替える場合には、毎回書き換える必要もなくなるので役に立つだろう。

Identifier［●］

グラフィックスカードの識別名を指定する。この識別名は後述のScreenセクションで利用する。

VendorName［▲］

グラフィックスカードのメーカー名を指定する。

BoardName［▲］

グラフィックスカードの名前を指定する。

Chipset［○］

グラフィックスカードが搭載しているグラフィックスチップを記述する。自動認識されない場合は必要になる。

VideoRam［○］

ビデオRAMの容量を指定する。自動認識される場合も多いが、中には誤認識されるカードもある。調べるまでもなくわかると思うので、書いておくのが間違いないだろう。Kバイト単位で指定する。

Clocks［○］

ドットクロックの値を列挙して指定する。単位はMHz。自動認識されない場合は必要になる。

Option "no_accel"［○］

ハードウェアアクセラレーションを使わずに、ソフトウェアのみによる描画を行う。通常は必要ないが、うまく動作しない場合は試してみるとよいだろう。

Ramdac［○］

グラフィックスカードに使われているRAMDACチップを指定する。ほとんどの場合は自動認識される。

DacSpeed［○］

RAMDACの周波数を指定する。単位はMHz。ほとんどの場合は自動認識されるが、うまく動作しない場合は設定してみるとよいだろう。グラフィックスカード上にあるRAMDACの表面に数字が印刷されていることが多い。

■Screenセクション

このセクションでは、MonitorセクションとDeviceセクションを関連づけて、実際にどのモードで起動するかを指定する（**リスト10**）。

Xサーバはまず、DefaultColorDepthで指定した色数のDisplayサブセクションを見る。そして、Modesに記述された解像度を行のはじめから順に調べ、MonitorセクションのModelineで設定されているいるものを探す。合致する解像度が見つかると、Deviceで指定されたグラフィックスカードの機能を使

▼リスト10　Screenセクションの例

```
Section "Screen"
    Driver     "svga"
    Device     "Matrox G200"
    Monitor    "RD17G3"
    DefaultColorDepth    24
    Subsection "Display"
        Depth     8
        Modes    "1024x768" "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth     16
        Modes    "1024x768" "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth     24
        Modes    "1024x768" "800x600" "640x480"
        Virtual 1024 768
    EndSubsection
EndSection
```

パラメータ	Xサーバ	説明
"Svga"	XF86_SVGA	現在一般的に使われる、高解像度、高色数のサーバ
"Vga16"	XF86_VGA16	基本的なVGA（640×480）16色のサーバ。ほとんどのマシンで利用できる
"Vga2"	XF86_Mono	基本的なVGA（640×480）モノクロのサーバ。ほとんどのマシンで利用できる
"Mono"	XF86_Mono、XF86_VGA16	基本的なVGA（640×480）モノクロのサーバ。ほとんどのマシンで利用できる
"Accel"	XF86_S3、XF86_Mach8、XF86_Mach32、XF86_Mach64など	それぞれのハードウェアのアクセラレーション機能に対応したサーバ

表1　XFree86 3.3.xのXサーバ

い表示しようとする。

ハードウェアの性能に比べて、Xの解像度が低いようであれば、ModesとMonitorセクションのModelineの記述を見直してみよう。

Driver［●］

使用するXサーバを指定する。最近は、SVGAサーバが多くの種類のグラフィックスカードをサポートしているので、たいていは“svga”を指定することになる。パラメータとして指定するXサーバには**表1**のようなものがある。

Device［●］

使用するグラフィックスカードを指定する。Deviceセクションで指定したIdentifierを指定する。

Monitor［●］

使用するディスプレイを指定する。Monitorセクションで指定したIdentifierを指定する。

DefaultColorDepth［▲］

デフォルトの色数を指定する。

Subsection "Display"［●］

色数別のビデオモードを指定する。Displayサブセクションは複数設定することができる。最優先になるのは、DefaultColorDepthで指定されたDepthのサブセクションとなり、一致するものがなければ、先に定義されたものから順に利用される。SubsectionはEndSubsectionで終了する。

Depth［●］

色数を指定する。

Modes［●］

解像度を指定する。MonitorセクションのModelineで指定した名前を記入する。前にあるものから順に、対応するディスプレイのModelineの識別名と一致するものを検索して、見つかったものを利用する。ServerFlagsセクションでDontZoomが指定されていなければ、Ctrl＋Alt＋（テンキーの）“＋”か“－”で指定されている解像度を変更することができる。

Virtual［▲］

仮想スクリーンの解像度（横と縦）を指定する。指定されている場合、ディスプレイが表示するのはModesで指定した解像度だが、ルートウィンドウはここで指定した解像度となる。すなわち、マウスを端に持っていくと画面がスクロールして、見えていない部分が表示されるようになる。画面の狭いノートパソコンなどで有効に使える機能だ（**図1**）。

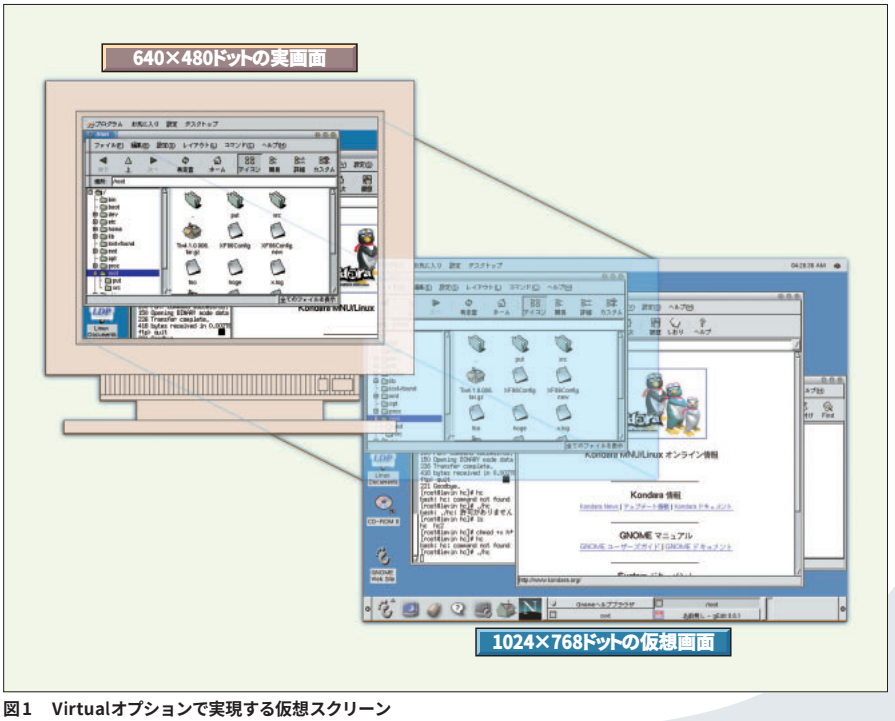


図1　Virtualオプションで実現する仮想スクリーン

XFree86 4.0での設定

XFree86 4.0ではいくつかの新機能が導入されたため、XF86Configの記述方法が変更された。XFree86 3.3の設定のままで問題ない部分も多いが、4.0の機能を活かすには一部の記述方法を変更する必要がある。注意すべき点は以下のとおりだ。

なお、/usr/X11R6/lib/doc/XF86Config.egというサンプルファイルが用意されているのでそちらも参考にしてほしい。

■Moduleセクション

XFree86 4.0では、モジュールを組み込んでさまざまな拡張機能を使うことができる。利用するモジュールはこ

▼リスト11　Moduleセクションの例

```
Section "Module"
    Load        "type1"
#   Load        "freetype"
    Load        "xtt"
    Load        "glx"
    Load        "dri"
EndSection
```

のセクションで指定する。たとえば、XFree86 3.3で日本語TrueTypeフォントを利用するには、XサーバにX-TTパッチをあてていたが、4.0ではX-TTモジュールの組み込みにより利用できるようになった（**リスト11**）。また、GLX、DRIなどもモジュールとして実現されているので、利用する場合はここでモジュールを指定する。

■InputDeviceセクション

キーボードやマウスなどの入力デバイスの設定は、InputDeviceセクションに統合された（**リスト12**）。これは、XFree86 4.0で導入された複数のディスプレイや入力デバイスをサポートするマルチヘッド機能に対応してのことだ。複数定義した場合は後述のServerLayoutセクションで組み合わせを定義する。

それぞれのデバイスについて別々のInputDeviceセクションを作成し、IdentifierとDriverを書くことになる。Driverはキーボードの場合は"keyboard"、マウスの場合は"mouse"だ。それぞれのデバイスのオプションについても若干変更がある。

なお、XFree86 3.3で使われていたKeyboardセクションやPointerセクションも互換性を維持するために残されている。

■Deviceセクション

Deviceセクションの変更は、BusIDというエントリーが加わったことだ。これはPCIやAGPのカードが複数あった場合、どのカードに対応するかを判別するために使われる。カードが1つの場合は特に必要ない。

■ServerLayoutセクション

XFree86 4.0では、マルチヘッドをサポートしたため、複数のディスプレイや入力デバイスに対応できる。これにより、どのデバイスを利用するかを決定するために、1つ以上の"ServerLayout"セクションが必要になる。このセクションでは、Screenセクション、Monitorセクション、InputDeviceセクションで定義したIdentifierを組み合わせてレイアウトを定義する（**リスト13**）。

使用するレイアウトは、Xの起動時に-layoutオプションをつけることにより決定できる。指定がなければ最初に定義されたセクションが利用される。

■DRIセクション

DRI（Direct Rendering Infrastructure）は、XFree86 4.0で導入された、3D描画を高速にするための機能である。このセクションにはDRIの設定を記述する。しかし、まだ実際にDRIを活用できることはあまりないので、記述しなくても問題ない。

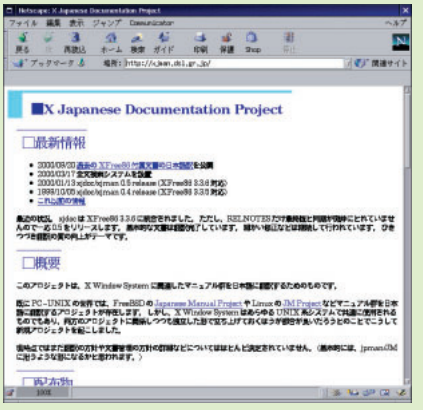
最後になるが、XF86Configの各項目は、man XF86Configを実行して調べることもできるので参考にしてほしい。

Column

XFree86に関する日本語ドキュメント

X Japanese Documentation Projectは、Xに関連したドキュメント、マニュアルなどを日本語に翻訳するプロジェクトだ。現在ではこのプロジェクトの成果は本家のXFree86 Projectに統合されているが、日本において、このプロジェクトがXFree86の普及に果たした役割は大きい。

英語が苦手だという方は、ぜひこれらの日本語ドキュメントを手に入れて、設定の参考にすることをお勧めしたい。なお、今回の記事の執筆においても参考にさせていただいている。感謝したい。



X Japanese Documentation ProjectのWebページ
URL： http://xjman.dsl.gr.jp/

▼リスト12　InputDeviceセクションでのキーボードの定義の例

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "keyboard"
    Option      "XkbRules"    "xfree86"
    Option      "XkbModel"    "jp106"
    Option      "XkbLayout"   "jp"
EndSection
```

▼リスト13　ServerLayoutセクションの例

```
Section "ServerLayout"
    Identifier   "Server1"
    Screen       "screen1"
        :
    InputDevice  "keyboard1"
    InputDevice  "Mouse1"
        :
    options
        :
EndSection
```

実践！ XFree86セットアップ

文：編集部
Text : Linux magazine

設定例その1－Intel 815編

ここでは、グラフィックス機能統合型チップセットであるIntel 815と、高性能でありながらリーズナブルな価格で人気の高いNVIDIA GeForce2 MXを搭載したビデオカードを例にXFree86のセットアップの実際を解説していくことにしよう。とは言うものの、どちらも新しいグラフィックチップなので、XFree86では標準でサポートされていない。さて結果やいかに……。

統合型チップセット i815

Intel 815チップセット（i815）は、Intel 810（i810）の後継チップセットとして登場したIntelの統合型チップセットの最新モデルである。「統合型」と呼ばれるのは、グラフィックス機能とサウンド機能がチップセット内に組み込まれているためで、i815ではGMCH（Graphics and AGP Memory Controller Hub）と呼ばれるホストコントローラ82815（**写真1**）にグラフィックス機能が、ICH （I/O Controller Hub）と呼ばれるI/Oコントローラ82801AAにサウンド機能が統合されている。 i815シリーズには、I/Oコントローラに82801BA（ICH2）を採用したハイエンドモデルi815E、GMCH 82815EMと VCH（ Video Controller Hub）82807AA、82801BAM（ICH2-M）をセットとしたモバイル用途向けのi815EMがラインナップされている。また、同シリーズながらi815EPは、82815からグラフィック機能を除いたMCH（ Memory Controller Hub）82815EPを採用した非統合型のチップセットである。

統合型チップセットを採用したマザーボードは、グラフィックスやサウンド専用のカード／チップを追加する必要がないことから、オールインワンタイプでコンパクトなPCの設計に向いている。実際にi810は、省スペース型のメーカー製PCやベアボーンキットを中心に普及している。i815の普及も、これらのカテゴリとi815EMを採用したノートPCがメインとなるはずだ。

Linuxとの相性は？

i81xシリーズのXFree86でのサポートは、比較的スムーズに行われた。開発元であるIntelがXFree 3.3.6に対応したi810用のXサーバとカーネルドライバモジュール（agpgart.o）を配布し、XF86_SVGAサーバ用のパッチも公開している。これらは、バージョンアップを経て現在ではi810e、i810-DC100、i815にも対応済みである。また、XFree86 4.0.xには標準でi810（i810e、i810-DC100を含む）のドライバが付属している。ただし、Intel、XFree86

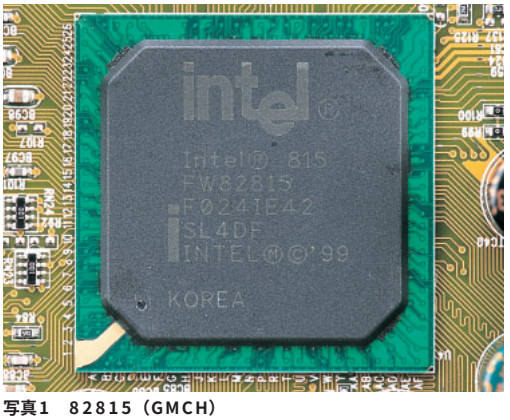


写真1　82815（GMCH）
i815チップセットのメインチップ。ビルドインされているグラフィックス処理機能は「Direct AGP」と呼ばれている。グラフィックス処理で使用されるメモリは、必要に応じてメインメモリから割り当てられる。外部バス用のAGPx4コントローラも搭載する。

写真2　テストで使用したベアボーンキット
マーザーボードはGigaByteのGA-60MM7E。FDDのみでCD-ROMドライブがないのは残念だが、日本語112キーボード、ホイールマウス、ネットワークカード（チップはRTL8139B）が付属して実売2万9800円と手ごろな価格だ。

4.0.xのどちらもLinuxのみでのサポートで、BSDなどほかのOSでXFree86を利用するケースには対応していない。

XFree 3.3.6を採用している各Linuxディストリビューションでも、Intelが提供するパッチ、または4.0系列からのバックポートパッチを適用することで、i810を（一部はi815も）サポートしている。主な日本語ディストリビューションでのサポート状況を**表1**にまとめておくので参考にしてほしい。表1からわかるように、i810に関しては、ほとんどのディストリビューションが何らかの形で対応済みだ。i810とi815をともに標準でサポートしているのは、Red Hat Linux 7JとKondara MNU/Linux 2000のみである。

ディストリビューション	i810サポート	i815サポート
Red Hat Linux 7J	標準でサポート	標準でサポート
Vine Linux 2.1	標準でサポート	未サポート
Kondara MNU/Linux 2000	標準でサポート	標準でサポート
LASER5 Linux 6.2	FTPサイトで対応パッケージを提供（http://www.laser5.co.jp/support/csm/learning/i810-000627.htmlを参照）	未サポート
Red Hat Linux 6.2J	FTPサイトで対応パッケージを提供（http://www.redhat.com/support/docs/tips/i810.htmlを参照）	未サポート
TurboLinux 6.0	標準でサポート（ただし/etc/modules.confの変更が必要）	FTPサイトで対応パッケージを提供

表1　主なディストリビューションにおけるi81xシリーズのサポート状況

提供元	FTPサイトのURL	ウンロードするファイル
ターボリナックスジャパン	ftp.turbolinux.co.jp/pub/TurboLinux/TurboLinux/ia32/Workstation/6.0/ja/updates/RPMS/	XFCom-i815-1.2-1.i386.rpm kernel-2.2.16-5.i386.rpm
Intel	download.intel.com/support/graphics/intel810	I810Gtt-0.2-4.src.rpm XFCom-i810-1.2-3.i386.rpm

表2　本文で紹介したサイトのURL

実際に試してみる

ここからはi815に絞って、表1に示したディストリビューション別にXFree86の設定方法を説明しよう。i815をサポートしているXサーバは、ほぼ例外なくi810もサポートしているので、i810ユーザーの方も以下で紹介する設定が参考になるはずだ。動作確認は、i815Eマザーを搭載したベアボーンキット（**写真2**）をベースにしたPCで行った。i815マザーをベースとしたマシンでは、オンボードのグラフィックス機能を無効にし、AGPスロットにビデオカードを装着して使用することも可能だが、今回はi815にビルドインされているグラフィックス機能のみについて検証している。

■Red Hat 7J、Kondara 2000

前述のとおり、Red Hat 7JとKondara 2000ではi815が標準でサポートされている。この2つのディストリビューションの場合、インストーラが自動認識してくれるはずだ。今回のテストでも、問題なくインストールできたし、そのままの状態でXを起動することができた。

■TurboLinux 6.0

ターボリナックスジャパンは、i815専用のXサーバのRPMパッケージ（XFCom-i815-1.2-1.i386.rpm）をFTPサイトを通じて配布している（URLは表2を参照）。これはIntelが配布しているXサーバにパッチを当てたもののようだ。まずは、このRPMパッケージを入手しよう。

agpgart.oについても、バージョンアップが必要になる。新しいagpgart.oは、同じURLにあるカーネルのアップグレードRPMパッケージ（kernel-2.2.16-5.i386.rpm）に含まれている。あわせて入手しておこう。

TurboLinuxのインストールでは、i815のグラフィックス機能はビデオカードとして検出されない。とりあえず、手動設定でi810を選択しておこう。「リストにないカード」を選択し、使用するXサーバにXF86_SVGAを指定してもよい。ビデオRAM容量は4096Kバイト、画面解像度は16bppの1024×768をとりあえず指定しておく。

カーネルのアップグレードとi815用Xサーバのインストールはrpmコマンドで行う。

```
# rpm -Uvh kernel-2.2.16-5.i386.rpm
# rpm -ivh XFCom-i815-1.2-1.i386.rpm
```

/usr/X11R6/binにXFCom_i815がコピーされる。このままではXの起動コマンドでサーバが正しく参照されないので、/etc/X11にシンボリックリンクを作成する必要がある。このとき、念のためオリジナルのXF86_SVGAのバックアップをmvコマンドで作成しておこう。

```
# cd /usr/X11R6/bin
# mv XF86_SVGA XF86_SVGA.SAVE
# ln -sf /usr/X11R6/bin/XFCom_i815
/usr/X11R6/bin/XF86_SVGA
# ln -sf /usr/X11R6/bin/XF86_SVGA
/etc/X11/X
```

さらに、/etc/modules.conf内の「agpgart」が指定されている行の先頭から「#」を削除して、この指定を有効にする。そして新しいカーネルから起動するために、システムをリブートしよう。

これで「startx」でXが起動するはずだ。なお、ターボリナックスジャパンのWebサイトに詳細なインストール手順を解説したページが用意されている（http://www.turbolinux.co.jp/knowledge/public/323.html）。このページでは、i815のサウンド機能を利用する方法についても解説しているので、一度目を通しておくことをお勧めする。

■ Vine 2.1、LASER5 6.2、Red Hat 6.2J

Vine Linux 2.1、LASER5 Linux 6.2、Red Hat Linux 6.2Jは、標準ではi815をサポートしていない。自動認識されないので、インストールの際にはXの設定をスキップしよう。当然のことながら、XFree86自体はインストールしておくこと。

インストール時の注意点がもうひとつある。グラフィカルインストーラはXの機能を使用しているため、正常に起動しない可能性がある。インストール時のブート画面でboot:プロンプトに「text」と入力して、テキストインストーラを使ったほうが確実だ。

Xを利用するには、Intelが配布しているi81xシリーズ用のXサーバとagpgart.oモジュールが必要になる。まずは、**表2**に示したURLからXFCom_i810-1.2-3.i386.rpmとI810Gtt-0.2-4.src.rpmを入手する。RPMパッケージなので、インストール自体は簡単だ。以下のコマンドを実行すればいい。

```
# rpm --recompile I810Gtt-0.2-4.src.rpm
# rpm -ivh XFCom_i810-1.2-3.i386.rpm
```

1行目のコマンドで、/lib/modules/＜カーネルバージョン＞/miscにagpgart.oが作成される。同時に、/etc/conf.modulesに必要な記述が追加され、/devにデバイスファイルagpgartが作成される。

2行目のコマンドでは、TurboLinuxの場合と同様に、/usr/X11R6/binにXサーバがコピーされる。ただし名前が違っていて、こちらはXFCom_i810となる。必要なシンボリックリンクも自動的に作成される。

あとはXconfiguratorで必要な設定を行えば、Xが起動するようになる。このときi815が自動認識されれば、そのまま設定を進めればいい。認識され

Column

そのほかのディストリビューションは？

今回テストしたディストリビューション以外を使用するケースについても簡単に触れておこう。

■サポートの有無を確認する

なにはともあれ、インストールされているディストリビューションで「XFree86＋i815」の組み合わせがサポートされているかを確認しなければならない。ディストリビューターのWebサイトや付属のドキュメントに明記されている場合はよいとして、それ以外の場合における確認方法を紹介しよう。

Xサーバでのサポートを確認するには、「XF86_SVGA -showconfig」とする。表示されるビデオチップの一覧に「i815」があればOKである。次にカーネルモジュールagpgart.oとデバイスファイルagpgartの有無を確認。ファイルパスは、ほとんどのディストリビューションで80ページで紹介したものと同じはずだ。最後に、/etc/conf.modules（またはmodules.conf）に以下の行が含まれているかlessコマンドなどで確認しよう。

```
alias char-major-10-175 agpgart
```

基本的には、Xサーバでサポートされていて、agpgart.oがあればXが動作するはずだ。デバイスファイルはコマンドで作成できるし、conf.moduleの記述はテキストエディタで追加すればいい。デバイスファイルを作成するコマンドは次のとおり。

```
# mknod -m 666 /dev/agpgart c 10 175
```

XサーバにX-TTパッチが適用されているかどうか確認するためには、lddコマンドが有効だ。このコマンドは指定されたプログラムまたはライブラリが動作するために必要とするライブラリ（そのプログラムが「依存」しているライブラリ）を表示する。たとえば次のようにして確認できる。

```
$ ldd /usr/X11R6/bin/XF86_SVGA
```

X-TTパッチが適用されていれば、「libfont.so.1」と「libttf.so.2」が表示されるはずだ。

■Intelのコンポーネントを利用する

使用しているディストリビューションでサポートされていない場合には、Intelが提供しているXサーバとカーネルモジュールを試してみよう。Intelのリリースノートによると、次の条件が満たされていれば動作するとされている。

- **2.2.X系のカーネルとそのソース**
- **XFree86-3.3.6**
- **glibc 2.1以上**
- **gccコンパイラ**

RPMパッケージで提供されているので、もちろんrpmコマンドが使えることも前提条件となる。

ない場合は、カードの一覧から「リストにないカード」（「Unlisted Card」）を選択する。ビデオメモリに指定した値は、実際には反映されないので任意の値でかまわない。解像度はモニタに合わせて設定すること。設定後のXF86ConfigのDeviceセクションとScreenセクションの例を**リスト1**に示すので参考にしてほしい。

▼リスト1　XF86Configの設定例

```
# Device configured by Xconfigurator:

Section "Device"
    Identifier  "i810"
EndSection

# **********************************************************************
# Screen sections
# **********************************************************************

# The Colour SVGA server

Section "Screen"
    Driver      "svga"
    Device      "i810"
    Monitor     "My Monitor"
    Subsection "Display"
        Depth       16
        Modes       "1280x1024"
        ViewPort    0 0
    EndSubsection
EndSection
```

TrueTypeに若干の問題

今回のテストで、Intelが配布しているXFCom_i810は日本語TrueTypeフォントの処理に問題があることがわかった。XF86Configの「FontPath」行やXフォントサーバの設定ファイル（/etc/X11/fs/config）で日本語TrueTypeフォントを含むパスを指定していると、エラーが発生してXが起動しなくなったり、起動しても日本語TrueTypeフォントが処理できないケースがあった。これは、X-TT（X-TrueType Server）といったTrueTypeフォントを扱うための機能がIntelのXサーバに組み込まれていないためだと思われる。

この問題を解決するには、何らかの方法でi815とX-TTの双方をサポートするXサーバをインストールする必要がある。TurboLinuxのところで紹介したXFCom_i815サーバは、日本語TrueTypeフォントを処理できる（X-TTパッチが適用されていると思われる）ので、このXサーバを他のディストリビューションでも試してみた。

その結果、Vine 2.1とLASER5 6.2では動作したが、Red Hat 6.2Jでは動作しなかった。原因としては、Red Hat 6.2ではTrueTypeの処理にX-TTを利用していないことが考えられる。i815サポートとX-TTのパッチを適用して、Xサーバをソースからコンパイルし直すことで対処できるが、手順が複雑なため残念ながらここでは紹介しきれない。適用するパッチについては、X-TrueType Serverの Webサイト（http://x-tt.dsl.gr.jp/）に掲載されている情報や、XFree86のソースRPMに含まれるスペックファイルなどが参考になる。チャレンジャブルな方は、挑戦してみよう。

▼リスト2　XFree86のフォントパス設定

```
[/etc/X11/XF86Configでのフォントパスの例]
                              ⋮
Section "Files"

    RgbPath      "/usr/X11R6/lib/X11/rgb"
    FontPath   "unix/:-1"
    FontPath     "/usr/X11R6/lib/X11/fonts/TrueType"

EndSection
                              ⋮
[/etc/X11/fs/configでのフォントパスの例]
                              ⋮
catalogue = /usr/X11R6/lib/X11/fonts/japanese:unscaled,
                              ⋮
        /usr/X11R6/lib/X11/fonts/TrueType,
        /usr/share/fonts/ISO8859-2/misc:unscaled,
                              ⋮
```

この行をコメントアウト

フォントパスの指定から日本語TrueTypeの指定をはずせば（**リスト2**を参照）、その他の処理は問題なく行える（TrueType以外の日本語フォントは扱える）ので、とりあえず「限定版」としてXFCom_810を使うことをお勧めしたい。

どうしても日本語TrueTypeフォントが必要なケース以外は、ディストリビューターの対応を待ったほうが賢明だ。Red Hatユーザーは、7Jへのバージョンアップというのもひとつの選択肢だろう。

設定例その2－NVIDIA GeForce2 MX編

NVIDIA GeForce2 MXは、圧倒的な3D描画性能で一大ブームを巻き起こしたGeForce256の次世代を担うGeForce2ファミリーに属するグラフィックスチップだ。GeForce2ファミリーには、Ultra、Pro、GTS、MXがあり、末弟のMXはローエンド・コンシューマをターゲットにした普及モデルという位置づけとなっている。

ここでは、XFree86でGeForce2 MXを利用できるのかどうか検証してみよう。ターゲットとして用意したのは、Leadtek社のグラフィックスカードWinFast GeForce2 MXだ（**写真3**）。購入価格は1万3980円だった。

そのままでは動かない！

残念ながら、XFree86の最新配布バージョンである3.3.6、4.0.1のどちらもこのチップをサポートしていない。これは、現在配布、あるいは販売されているほとんどのディストリビューションは、そのままではX Window Systemを利用できないことを意味する。では、どうすればXが使えるのだろうか。NVIDIAでは、XFree86 4.0.1に対応するドライバをWebページ（http://www.nvidia.com/）で配布している。従って、XFree86 4.0.1とNVIDIAのドライバをインストールすればよいのだ。また、XFree86プロジェクトのCVSツリーから開発版パッチを入手し、XFree86 4.0.1に適用してもGeForce2 MXを動作させることができる。

今回は、XFree86 4.0.1とNVIDIAドライバ0.95を組み合わせ、i815チップセットの場合と同じ6つのディストリビューションで検証を行った。セットアップの手順は次の通りだ。

1. **XFree86 4.0.1をインストール**
2. **NVIDIAドライバのカーネルモジュールをコンパイルし、インストール**
3. **NVIDIAドライバをインストール**
4. **設定ファイルのXF86Configを作成**

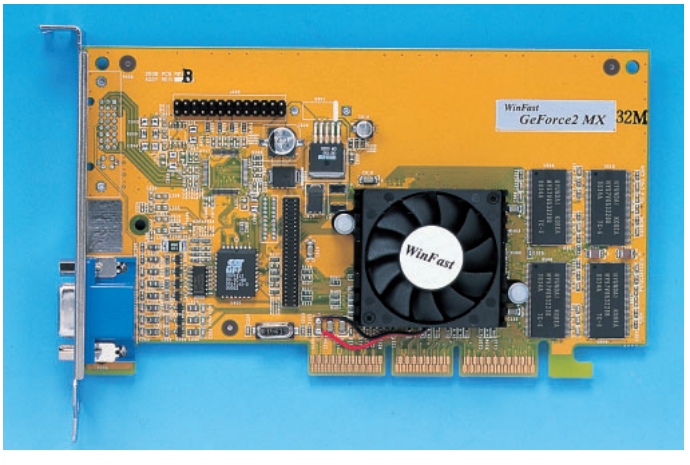


写真3　WinFast GeForce2 MX
32Mバイトのメモリを搭載し、TV出力アダプタは別売り。それにしてもストレートな名前だ。

```
# rpm --rebuild NVIDIA_kernel-0.9-5.src.rpm
# rpm -ivh /usr/src/redhat/RPMS/i386/NVIDIA_kernel-0.9-5.i386.rpm
# rpm -ivh NVIDIA_GLX-0.9-5.i386.rpm
```

画面1　Red Hat Linux 7.0JにNVIDIAドライバをインストールする

XFree86 3.3.xがインストールされている環境では、これらの作業を行う前にXFree86 3.3.xをアンインストールしなければならない。これは初心者にとってはやや難しい作業となるので、自信がなければXFree86 4.0.1以降を採用したディストリビューションをインストールしたほうがよいだろう。この数カ月で主要なディストリビューションはXFree86 4.xを標準装備するはずだ。

インストールの手順

それでは、インストールの手順を見てみよう。

■Red Hat Linux 7.0J

Red Hat Linux 7.0Jは、XFree86 4.0.1に開発版パッチをあてた4.0.1aを採用している（GeForce2 MXは未サポート）。そのため、作業はNVIDIAのドライバをインストールするだけで済む。　最初に、NVIDIAのWebサイトからカーネルモジュールとドライバのファイルを入手しよう。ファイル名は、それぞれNVIDIA_kernel-0.9-5.src.rpm、NVIDIA_GLX-0.9-5.i386.rpmだ。入手したら、**画面1**のようにしてインストールする。

次に、XFree86の設定ファイルを作る。Red Hat Linux 7.0Jには、XFree86 4.0.1に対応する設定ツールXconfiguretorが用意されているので、これでひな形を作る。「画面の設定」画面では、“検出を行わない”を選ぶことと、「Xを起動しています」画面で“スキップ”を選ぶのがポイントだ。これはXconfiguratorは、NVIDIAの配布するドライバを考慮して作られていないためだ。以上で、XFree86の設定ファイル/etc/X11/XF86Config-4が作られ

た。Red Hat Linux 7.0Jでは、XFree86 4.0.1用の設定ファイルはXF86Configではなく、XF86Config-4であることに気をつけたい。

XF86Config-4ができたら、このファイルをエディタで開き、Driver "nv"と書かれている部分を探して“ nv ”→“ nvidia ”と書き換えればよい。“ nv ”は、XFree86 4.0.1が標準で持っているNVIDIA用のドライバだ。このドライバはGeForce2 MXに対応していない。そこで、先ほどインストールしたNVIDIAドライバが組み込まれるよう書き換えるのだ。

以上で、Xが起動するようになる。

■XFree86 3.3.xからアップデート

こんどは、XFree86 3.3.xを採用しているディストリビューションではどうすればよいか解説しよう。

これらのディストリビューションでは、XFree86 3.3.xをアンインストールしてからXFree86 4.0.1をインストールしなくてはならない。**画面2**のようにしてXFree86 3.3.xをアンインストールする。念のため、アンインストールする前に/usr/X11R6ディレクトリと/etc/X11ディレクトリのバックアップを取っておくとよいだろう。

次はXFree86 4.0.1のインストールだ。Red Hat Linux 6.2J、LASER5 Linux 6.2、Vine Linux 2.1については、XFree86 4.0.1のソースをコンパイルしてインストールした。ソースコードは、XFree86 ProjectのWebページ（http://www.xfree86.org/）などから入手可能だが、付録CD-ROM Disc2にも収録したので利用してほしい。インストールの手順は**画面3**のようになる。make Worldには結構時間がかかるのでのんびりと待っていよう。ちなみに、AMD K6-III 400MHz、メモリ128Mバイトのマシンで50分弱かかった。make installをしたら、/etc/ld.so.confファイルをエディタで開き、/usr/X11R6/libという行を追加し、ldconfigを実行する。これは、共有ライブラリを検索するパスの設定だが、XFree86 3.3.xをアンインストールする際に削られてしまっているので、再び追加する必要があるからだ。

TurboLinux Workstation 日本語版 6.0では、同社のFTPサイトにあるunstableディレクトリからsrc.rpmファイルを入手し、Kondara MNU/Linux 2000については、次期バージョンに向けたファイルが置かれているFTPサイトのJiraiディレクトリからsrc.rpmを入手してそれぞれビルドした。どちらも、XFree86 4.0.1に開発版パッチがあてられており、標準で含まれる“ nv ”ドライバでもXが起動した。ただし、開発版のファイルは常に更新されているので、ここではこれ以上解説しない。

■NVIDIAドライバのインストール

XFree86 4.0.1の次は、NVIDIAドライバのインストールだ。手順はRed Hat Linux 7.0Jの場合と同じだが、RPMファイルのインストール時に依存関係のエラーが出るので、--nodepsオプションを付ける必要がある。また、LASER5 Linux 6.2、Vine Linux 2.1では、カーネルのバージョンの関係で、src.rpmからカーネルモジュールを作ることができない。カーネルのバージョンを上げるのが面倒なら、NVIDIA_kernel-0.9-5.tar.gzを展開し、os-interface.cを**リスト3**のように書き換えてmakeする。

■XF86Configの作成

XFree86 4.0.1にはXF86SetupのようなGUIの設定ツールは付属していな

```
# rpm -qa | grep ^XFree | grep 3.3.6 | xargs rpm -e --nodeps
```

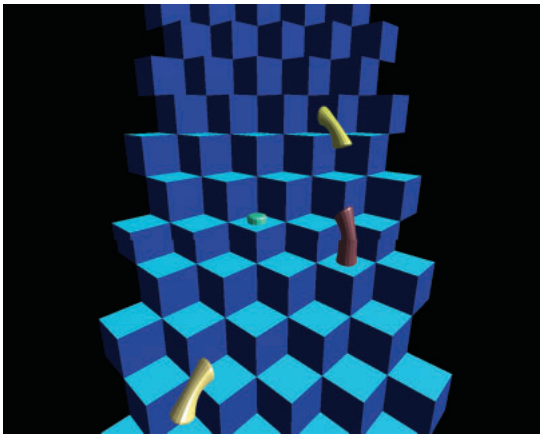
画面2　XFree86 3.3.xをアンインストール（3.3.6での例）

```
$ tar xvzf X401src-1.tgz
$ tar xvzf X401src-2.tgz
$ tar xvzf X401src-3.tgz
$ cd xc
$ make World
$ su
# make install
―― /etc/ld.so.confファイルに“/usr/X11R6/lib”を追加する ――
# ldconfig
```

画面3　XFree86 4.0.1のインストール

▼リスト3　os-interface.cを書き換える

```
                        :
                        :
/*      case INTEL_I840:        chipset = "Intel i840";       break; */
        case VIA_GENERIC:       chipset = "VIA";              break;
        case VIA_VP3:           chipset = "VIA VP3";          break;
        case VIA_MVP3:          chipset = "VIA MVP3";         break;
/*      case VIA_MVP4:          chipset = "VIA MVP4";         break; */
                        :
                        :
```



画面5　xscreensaver
OpenGLを利用した3Dスクリーンセーバの画面。ソフトウェア描画に比べ、数十倍は速くなった。

画面6　Tux Racer
ペンギンのTuxが雪山を滑り降りる3Dゲーム。なめらかな動きで暴走しまくるペンギンは一見の価値あり！

いので、対話式のCUIツールxf86configを使って設定ファイルを作成する。ビデオカードデータベースから“RIVA TNT2”を選び、ひな形としよう。

続いて、作成したばかりの設定ファイル/etc/X11/XF86Configをエディタで開き、Red Hat Linux 7.0Jの場合と同様にドライバ名を“nv”→“nvidia”と書き換える。さらに、“# Load "glx"”と書かれた行のコメントマーク“#”を外す。

以上でXが起動するようになった。

OpenGLを使う

ここまでで、Xが使えるようになったが、GeForce2 MXを使うからには3D性能を発揮させたい。Linuxでは、OpenGL互換のMesaライブラリを使って3D描画を行うことが多い。Mesaライブラリをインストールしていなければ、ディストリビューションのバイナリCD-ROMからインストールしよう。Mesaライブラリをそのまま使うとソフトウェアで描画処理を行うため、速度は期待できない。**画面4**のように設定して、GeForce2 MXのハードウェア3D描画機能を使えるようにしよう。“1.2.0”の部分は、“1.2.030300”などになっていることもあるので適宜読み替えてほしい。以上の設定で、OpenGLを利用する3Dアプリケーションが劇的に速くなったはずだ（**画面5、6**）。

日本語TrueTypeフォントを使う

XFree86 4.0.1には、日本語TrueTypeフォントを扱うことのできるX-TTモジュールが標準装備されている。このモジュールを有効にするには、/etc/X11/XF86Configファイルの“Module”セクションに“Load "xtt"”という行を追加すればよい。ただし、FreeTypeモジュールと競合するので、“Load "freetype"”という行の先頭に“#”を付けてコメントアウトしておこう。

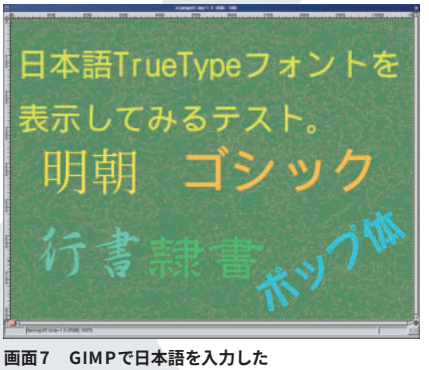
あとはFontPathの設定を行うだけで**画面7**のように日本語TrueTypeフォントが利用できるようになる。Red Hat Linux 7.0Jでは、Xフォントサーバ（xfs）を利用しているため、FontPathに各フォントのパスは指定されておらず、“FontPath "unix/:7100"”とだけ書かれている。xfsを使うと若干パフォーマンスが低下するので、ほかのマシンとフォントを共有しないならパスを直接指定してもよいだろう。

GeForce2 MXは買いか？

はじめにも書いたように、XFree86 3.3.xから4.0.1にアップデートするのはちょっと面倒だ。だが、この作業に自信がある、またはXFree86 4.xを標準装備したディストリビューションを使うなら、低価格で新世代3Dパフォーマンスを手に入れられるGeForce2 MXは“買い”だ（私はもう買った）。

```
# ln -sf /usr/lib/libGL.so.1.0.5 /usr/lib/libGL.so.1
# ln -sf /usr/lib/libGL.so.1 /usr/lib/libGL.so
# ln -sf /usr/lib/libGL.so /usr/X11R6/lib/libGL.so.1.2.0
# mv /usr/X11R6/lib/libGL.so.1.2.0 /usr/X11R6/lib/libGL.so.1.2.0.Mesa
# ln -s /usr/lib/libGL.so /usr/X11R6/lib/libGL.so.1.2.0
```

画面4　NVIDIAドライバでGeForce2 MXの3D描画機能を使う



画面7　GIMPで日本語を入力した
X-TTモジュールを有効にすれば日本語TrueTypeもバッチリ使える。

/etc/X11ディレクトリを探れ

文：編集部
Text : Linux magazine

LinuxやBSDなどUNIX系のOSは、好みに応じて、キーボードからコマンドを入力して使うことも、マウスなどのデバイスを用いたGUI(Graphical User Interface)で利用することもできる。

サーバ用途で使う場合や、キーボード入力に慣れ親しんでいるユーザーにとっては、GUIは不要かもしれない。しかしクライアントマシンとして使うなら、やはりGUI環境は便利なものだ。UNIX系のOSでは、X Window System（以下Xと表記）によるGUIが一般的に用いられている。中でも、もっともユーザーが多いのは、PC用のXFree86であろう。

Xは、それぞれXサーバ／Xクライアントと呼ばれる、2種類のプログラムが連携して動作している。ファイルマネージャ、エディタ、Webブラウザなど、一般にX上で使っている「プログラム」は、Xクライアントと思ってよい。Xサーバは、画面の描画と、キーボード、マウスなどの入力デバイスからの入力を、Xクライアントに伝えるのが仕事だ。本特集の前半で説明されている、

```
/etc/X11/XF86Config
```

は、XFree86のXサーバの設定ファイルである。

最近のディストリビューションに含まれているXFree86の設定ツールは、よくできており、最新のグラフィックスカードや一部のノートPCをのぞいては、容易に設定が行えるようになっている。以前、コマンドライン用のxf86configで苦労して設定した覚えのあるユーザーなら、思わず遠い目になってしまう。

XF86Config以外のファイルは？

ファイルマネージャやlsコマンドを用いて/etc/X11ディレクトリを見てみると、XF86Config以外にもいくつかのファイルやサブディレクトリが存在していることがわかるだろう（**画面1**）。

UNIX系のOSでは、各種の設定ファイルを/etcディレクトリ以下に配置するのが一般的だ。名前からも想像がつくように、そのサブディレクトリである/etc/X11には、Xに関連した設定ファイルが含まれている。これらのファイルを、ユーザーが書き換える機会はあまりないと思われる。だが、それぞれのファイルが、どのプログラムから呼び出されているか、デスクトップ画面のどこに反映されているかを知ることで、自分が使っているLinuxシステムへの理解も深まるだろう。

そこで、本特集のしめくくりとして、/etc/X11以下のファイルとサブディレクトリに迫ってみたいと思う。取り上げるのは、以下に示したRed Hat系の代表的なディストリビューション5種類である。

Red Hat Linux 6.2J

```
drwxr-xr-x  4 root  root   4096 Nov 16 07:16 AnotherLevel
drwxr-xr-x  2 root  root   4096 Nov 16 07:16 TheNextLevel
drwxr-xr-x  2 root  root   4096 Nov 16 07:16 WindowMaker
lrwxrwxrwx  1 root  root     29 Nov 15 22:42 X -> ../../usr/X11R6/bin/XF86_SVGA
-rw-r--r--  1 root  root  14054 Nov 15 22:42 XF86Config
drwxr-xr-x  9 root  root   4096 Nov 16 07:28 applnk
drwxr-xr-x  2 xfs   xfs    4096 Nov 16 07:34 fs
drwxr-xr-x  2 root  root   4096 Nov 16 07:20 fvwm2
drwxr-xr-x  6 root  root   4096 Nov 16 07:20 gdm
-rwxr-xr-x  1 root  root    803 Mar  9  2000 prefdm
drwxr-xr-x  2 root  root   4096 Nov 16 07:16 twm
drwxr-xr-x  2 root  root   4096 Nov 16 07:34 wmconfig
drwxr-xr-x  3 root  root   4096 Nov 16 07:16 xdm
drwxr-xr-x  3 root  root   4096 Nov 16 07:34 xinit
drwxr-xr-x  2 root  root   4096 Nov 16 07:16 xsm
```

画面1　/etc/X11の内容。Red Hat Linux 6.2Jの場合

Kondara MNU/Linux 2000
Vine Linux 2.1
LASER5 Linux 6.2
TurboLinux Workstation 6.0

それぞれインストール時には、パッケージ選択で「すべて」を選択している。なお、TurboLinuxについては、「開発ワークステーション」を選択してインストールしている。

X Window Systemの起動について

最初に、まずXがどのように起動しているのかを確認しておこう。インストール時に、「GUIによるログインを使用」を選択すると、起動時に**画面2**のようなログイン画面が現れる。選択しない場合には、黒い画面にテキストで「login:」などの文字が表示される。この2種類のログイン方法は、それぞれ異なるランレベルに対応している。

ランレベルとは、UNIX系OSの実行状態を表したもので、通常は0～6が用いられる。レベルと名が付いてはいるが、特に数字が大きいから良いとかいうものではない。0は停止（halt）、6は再起動（reboot）に割り当てられている。そのほかの1～5は、ディストリビューションによって異なるが、今回取り上げているRed Hat系では、1がシングルユーザーモード、3がキャラクタベースのマルチユーザーモード、5がGUIを用いたマルチユーザーモードに割り当てられている。

これらのランレベルは、/etcディレクトリ以下のinittabというファイルで定義されている（リスト1）。inittabの各行は、

```
id:runlevel:action:process
```

という書式になっている。actionの項目によって、各行の動作が決定される。actionは、全部で15種類ほどあるが、Xの起動に関連して知っておく必要があるのは、“initdefault”だ。

initdefaultが含まれた行のrunlevelに書かれているランレベルが、そのマシンのデフォルトのランレベルとなる。**リスト1**の例では、ランレベル5がデフォルトだ。この場合、inittabの各行のうち、runlevelの項目に5が指定されている行のprocessが、順次実行されていく。この規則に従い、**リスト1**の最後の行、

```
/etc/X11/prefdm -nodaemon
```

が実行されることで、GUIによるログイン画面が表示される。

ディスプレイマネージャの選択

GUIログイン用の画面を表示するプログラムは、ディスプレイマネージャと呼ばれる。Xの標準ディスプレイマネージャとしてはxdmがあるが、最近はxdmを元に独自の拡張を加えた、ディスプレイマネージャが多く用いられるようになっている。

prefdm（“preferred display manager”の略と思われる）は、名前が示すように、これらのディスプレイマネージャのどれを利用するのか選択するためのシェルスクリプトである。**リスト2**にRed Hat Linux 6.2Jのprefdmを示した。

いきなり暗号のようなアルファベットの羅列を見せられて、鬱な気分になってしまうかもしれないが、比較的短めのスクリプトなので、全体を調べてみよう。なお「シェルスクリプトのことをもっと知りたい！」というユーザーには、11月号より連載が開始された「**[超]入門シェルスクリプト**」の併読をお勧めする。

まず前半部、10～18行の“if”と“fi”で囲まれた部分では、/etc/sysconfig

画面2　GUIによるログイン画面

▼リスト1　Red Hat系の/etc/inittab（一部）

```
id:5:initdefault:
  :
  :
  :
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

ディレクトリ内の“ desktop ”というファイルをチェックしている。このファイルにGNOMEという文字列が含まれていれば、9行めで用意しておいた“ preferred ”という名のシェル変数にgdmという文字列を代入している。同様にKDEという文字列があれば、kdmが、AnotherLevelという文字列ならxdmが代入される。

次の19～27行の“ if ”～“ fi ”ブロックは、最初にシェル変数preferredの長さをチェックし、ゼロでなければ、言い換えれば10～18行ですでに指定されていれば、素通りされる。ゼロの場合は、内部でwhichコマンドを呼び出して、有効なディスプレイマネージャが存在するかどうかgdm、kdm、xdmの順に探し出している。

このように2段階でシェル変数preferredを指定し、29行めでようやく、実際にpreferredを用いて、ディスプレイマネージャが実行されていることがわかるだろう（**図1**）。

Red Hatのデフォルトはgdm

Red Hat Linux 6.2Jのデフォルトの状態では、/etc/sysconfigディレクトリ内にdesktopというファイルが存在するが、中身は何もない。また、ディスプレイマネージャgdmは、/usr/binディレクトリ以下に存在する。結果としてRed Hat Linux 6.2Jでは、デフォルトでgdmが用いられる。LASER5 Linux 6.2Jも同様だ。

もしディスプレイマネージャを変えたい場合は、/etc/sysconfigディレクトリ内のdesktopファイルに書き込めばいい。たとえば「私はKDE派だ！」という場合には、desktopファイルに、“ KDE ”の1行を加えればよい。次に起動したときは、kdmが表示されるはずだ。

Red Hat系ディストリビューションのなかでも、Kondara MNU/Linux 2000やVine Linux 2.1では、wdmという別のディスプレイマネージャが使用される。KondaraやVineのprefdmを説明することは省略するが、機会があればご自分で確認してみてほしい。

一覧表を見ればわかるように、TurboLinuxはprefdmがない。それではどうやってディスプレイマネージャ

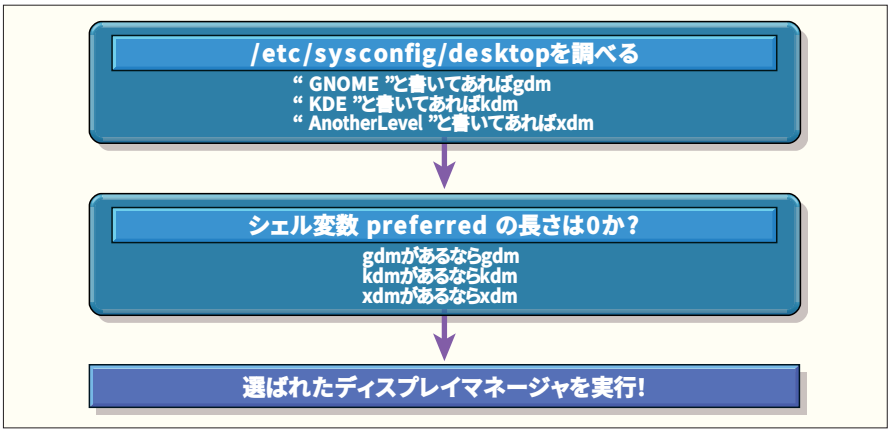


図1　ディスプレイマネージャ選択の流れ

▼リスト2　prefdm。ディスプレイマネージャを選択

```
#!/bin/sh

PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin

# We need to source this so that the login screens get translated
. /etc/profile.d/lang.sh

# Run preferred X display manager
preferred=
if [ -f /etc/sysconfig/desktop ]; then
    if grep -q GNOME /etc/sysconfig/desktop 2>/dev/null; then
        preferred=gdm
    elif grep -q KDE /etc/sysconfig/desktop 2> /dev/null; then
        preferred=kdm
    elif grep -q AnotherLevel /etc/sysconfig/desktop 2> /dev/null; then
        preferred=xdm
    fi
fi
if [ -z "$preferred" ]; then
    if which gdm >/dev/null 2>&1; then
        preferred=gdm
    elif which kdm >/dev/null 2>&1; then
        preferred=kdm
    elif which xdm >/dev/null 2>&1; then
        preferred=xdm
    fi
fi
if [ -n "$preferred" ] && which $preferred >/dev/null 2>&1; then
    exec `which $preferred` $*
fi
exit 1
```

ディストリビューション	デフォルト
Red Hat Linux 6.2J	gdm
Kondara MNU/Linux 2000	wdm
Vine Linux 2.1	wdm
LASER5 Linux 6.2	gdm
TurboLinux Workstation 6.0	gdm

表1　デフォルトのディスプレイマネージャ

を指定しているのか？　TurboLinuxのinittabを参照すると、prefdmの代わりに独自のlaunch_xdmというプログラムが呼び出されていることがわかる。また/etc/X11ディレクトリに、launch_xdm.confという、そのものズバリの名前の設定ファイルがあり、その中でディスプレイマネージャ（デフォルトではgdm）とロケールが指定されている。

以上、長々とディスプレイマネージャについて説明してきた。ディスプレイマネージャが表示するログイン画面でユーザー認証を行うと、ウィンドウマネージャが起動し、Xのデスクトップ画面に到達するというわけだ。PCを起動してからデスクトップが表示されるまでの短時間に、このようにいろいろなプログラムが動いている。

コマンドラインからのX起動

GUI環境のみでLinuxを使う場合は、Xサーバはディスプレイマネージャによって起動され、ログアウトするたびに、リセットされているが、Xサーバのプロセスは常に存在している。

それに対して、ふだんはコンソールで作業を行い、必要なときだけXを使うというユーザーもいるだろう。Red Hat系のディストリビューションなら、ランレベル3の状態だ。このような使い方をする場合には、Xサーバは必要に応じて起動され、コンソールに戻る際には終了されることになる。

このような使い方をする際に、一般的に用いられるコマンドは、

```
startx
```

である。だが、manコマンドや書籍で調べると、コンソールからXサーバを起動させるコマンドは、

```
xinit
```

と説明されているはずだ。これはどちらかが間違いというわけではない。startxは、たいてい/usr/X11R6/binディレクトリに入っているので、バイナリのプログラムのように見えるが、実はスクリプトなのだ。xinitコマンドはすべてのX Window Systemにあり、ハードウェアやXのインプリメントに依存した、細かい差異を吸収しきれない。そこでXFree86では、これらの差を埋めるためにstartxというスクリプトを用意したということだ。

名前	R	K	V	L	T	説明	ページ
ファイル							
X	○	○	○	○	○	Xサーバへのシンボリックリンク	70
XF86Config	○	○	○	○	○	Xサーバの設定ファイル	70
prefdm	○	○	○	○	−	ディスプレイマネージャを選択するスクリプト	92
launch_xdm.conf	−	−	−	−	○	ディスプレイマネージャを選択するプログラム用の設定ファイル（TurboLinux）	92
サブディレクトリ							
AnotherLevel	○	−	−	○	−	ディスプレイマネージャの「AnotherLevel」に関連したファイルを格納する	−
SwitchXIM	−	−	−	○	−	日本語入力ソフト切り替えプログラムに関連したファイルを格納する（LASER5 Linux）	−
TheNextLevel	○	−	−	○	−	特に利用されていない	−
WindowMaker	○	○	○	−	−	ウィンドウマネージャWindow Makerに関連したファイルを格納する	94
applink	−	−	−	−	○	KDEのメニュー項目用データ（TurboLinux）	94
applnk	○	○	○	○	○	KDEのメニュー項目用データ	94
ccdef	−	−	−	○	−	日本語入力ソフト切り替えプログラムに関連したファイルを格納する（Vine Linux）	−
fs	○	○	○	○	○	X フォントサーバに関連したファイルを格納する	−
fvwm2	○	○	○	○	−	ウィンドウマネージャfvwm2に関連したファイルを格納する	−
gdm	○	○	−	○	○	ディスプレイマネージャgdmに関連したファイルを格納する	92
im	−	−	○	−	−	日本語入力ソフト切り替えプログラムに関連したファイルを格納する（Vine Linux）	−
twm	○	○	○	○	○	ウィンドウマネージャtwmに関連したファイルを格納する	−
wdm	−	○	○	−	−	ディスプレイマネージャwdmに関連したファイルを格納する	92
wmconfig	○	○	○	○	○	wmconfigコマンドの設定ファイルを格納する	94
xdm	○	○	○	○	○	ディスプレイマネージャxdmに関連したファイルを格納する	92
xinit	○	○	○	○	○	xinitに関連したファイルを格納する	89
xsm	○	○	○	○	○	Xセッションマネージャ（xsm）に関連したファイルを格納する	−

表2　/etc/X11ディレクトリの内容

表中のディストリビューション略称
R　Red Hat Linux 6.2J
K　Kondara MNU/Linux 2000
V　Vine Linux 2.1
L　LASER5 Linux 6.2
T　TurboLinux Workstation 6.0

xinit

コマンドラインからXを起動するために、xinitというプログラムが用意されているが、XFree86ではstartxという名のスクリプトがあり、こちらを用いるのが一般的だ。startxは、内部でxinitを起動するのだが、その際ハードウェアやユーザー固有の設定ファイルを元にした引数を渡しているため、さまざまな環境で同じようにXを起動することが可能だ。

xinitが使用する設定ファイルのうち、全ユーザーに共通なものは、

```
/etc/X11/xinit/
```

以下にまとめて置かれている（**表1**）。また、ユーザーがカスタマイズした設定は、ホームディレクトリ以下に、ドットファイルとして置くのが一般的である。“ startx ”と入力してから、Xのデスクトップにたどり着くまでの流れは、**図1**のようになっている。

startx

/usr/X11R6/binというディレクトリに置かれているが、startxはシェルスクリプトである。Xクライアント、Xサーバそれぞれの設定ファイルを選択し、xinitに渡している。**リスト1**にその一部分を示した。

ここでは、xinitの設定ファイルとして、ユーザーのホームディレクトリにある.xinitrcと/etc/X11/xinit/xinitrcをそれぞれシェル変数に設定し、もしファイルが存在すれば、そのファイル名をシェル変数“ clientargs ”に入れている。ファイルが両方とも存在する場合は、ユーザーのホームディレクトリにある.xinitrcが優先的に指定される。clientargsは、startxの最後でxinitに引数として渡されていることがわかるだろう。

そのほか、Xサーバ用の設定ファイルなども指定できるが、デフォルトでは/etc/X11/xinit/xserverrcも存在しないので、一般的なPCでは、特に何もする必要はない。

xinitrc

startxから起動されたxinitは、XサーバとXクライアントプログラムを1つ起動する。Xクライアントは、上記のstartx内でclientargsとして指定されたファイルであり、ユーザーの設定ファイルが存在しない場合は、/etc/X11/xinit/xinitrc（**リスト2**）が相当する。xinitrc内で設定されるのは、以下の4項目である。

ファイル	説明
Xclients	セッションマネージャを選択する
Xmodmap	キーマップの追加設定ファイル
xinitrc	リソース、キーボード設定など
xinitrc.d/	クライアントごとの設定ファイルを置くディレクトリ
├─rxvt.xinitrc	rxvt
└─xinput	xinputc

表1　xinitディレクトリ以下のファイル

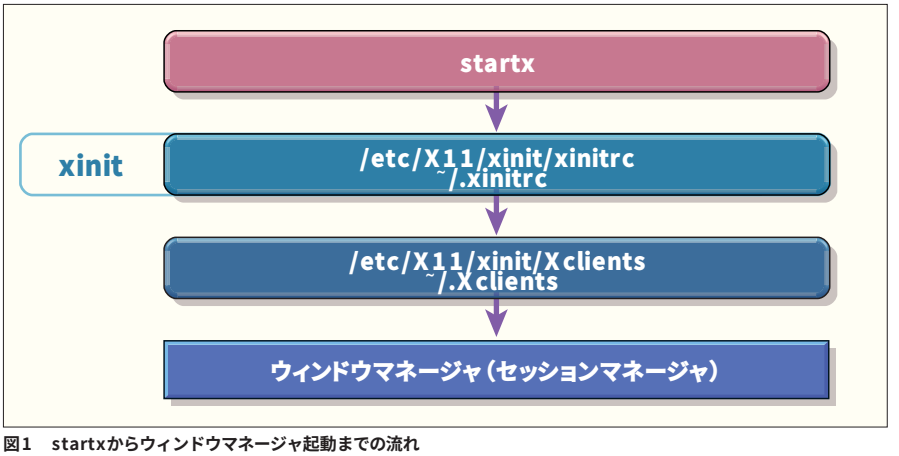


図1　startxからウィンドウマネージャ起動までの流れ

▼リスト1　Red Hat Linux 6.2Jのstartx（抜粋）

```
userclientrc=$HOME/.xinitrc
sysclientrc=/etc/X11/xinit/xinitrc
clientargs=""

if [ -f $userclientrc ]; then
    clientargs=$userclientrc
else if [ -f $sysclientrc ]; then
    clientargs=$sysclientrc
fi
fi
   :
   :
xinit $clientargs -- $server $display $serverargs
```

ホームディレクトリ内に設定ファイルがあれば、実行なければシステムの設定ファイルを実行

xinitに引数を渡して実行

- **リソースの設定**
- **キーマップの設定**
- **xinitrc.d/以下のスクリプトを実行**
- **Xclientsを実行**

●リソースの設定

xinitrcで最初に行われるのは、リソースの設定である。ユーザー固有の設定ファイル、システム全体の設定ファイルがそれぞれデータベースに登録される。

●キーマップの設定

Xでは、キーボードのキーと実際に入力される文字コードの組み合わせ（キーマップ）を自由に設定できる（キートップの文字とは無関係に、DVORAK配列にすることも可能）。キーマップの設定プログラムは、setxkbmapとxmodmapがあるが、同時に使うと問題が起きる可能性があるので、xinitrcでは、両方が使われることがないようになっている。優先的に使われるのは、setxkbmapだ。

●xinitrc.d/

/etc/X11/xinit/xinitrc.d/ディレクトリ以下のスクリプトが実行される。特定のプログラムだけで用いるリソースの登録などに用いられる。

▼リスト2　Red Hat Linux 6.2Jのxinitrc（抜粋）

```
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
userxkbmap=$HOME/.Xkbmap

sysresources=/etc/X11/xinit/Xresources
sysmodmap=/etc/X11/xinit/Xmodmap
sysxkbmap=/etc/X11/xinit/Xkbmap
    :
    :
# merge in defaults
if [ -f "$oldsysresources" ]; then
    xrdb -merge "$oldsysresources"
fi

if [ -f "$sysresources" ]; then
    xrdb -merge "$sysresources"
fi

if [ -f "$userresources" ]; then
    xrdb -merge "$userresources"
fi
    :
    :
if [ -f "$sysxkbmap" ]; then
    setxkbmap `cat "$sysxkbmap"`
    XKB_IN_USE=yes
fi
    :
    :
# xkb and xmodmap don't play nice together
if [ -z "$XKB_IN_USE" ]; then
    if [ -f "$oldsysmodmap" ]; then
        xmodmap "$oldsysmodmap"
    fi
```

リソース設定。システム、ユーザーリソースをデータベースに登録

setxkbmapの設定ファイルがあれば、実行
XKB_IN_USEをyesに

setxkbmapの設定をしておらず、かつxmodmapの設定ファイルがあれば、実行

●Xclientsを実行

最後にセッションマネージャの選択を行うXclientsを呼び出す。実際には、ユーザーがホームディレクトリに作成した~/.Xclientsが優先的に実行されるが、デフォルトの状態では~/.Xclientsは存在しない。

Xclients

X起動の最終段階として、Xclients（**リスト3**）が実行される。デフォルトのXclientsは80行もあるが、行っているのは、セッションマネージャの選択・起動のみである。サイズが大きい理由は、何かの原因で、ユーザーやシステムが指定したセッションマネージャが起動できない場合でも、代わりのセッションマネージャを起動できるようにするためである。

ここで言うセッションマネージャとは、GNOMEやKDEといったデスクトップ環境、またはウィンドウマネージャのことだと思ってよい。

最も優先されるのは、

```
/etc/sysconfig/desktop
```

というファイルでセッションマネージャを直接指定することである。このファイルに、GNOMEまたはKDEと書いておけば、gnome-sessionまたはstartkdeスクリプトが起動される。

desktopファイルで指定されなかった場合は、無条件にGNOME、続いてKDEが選択される。GNOMEやKDEのセッションマネージャとしての機能の高さを考えれば、現状では妥当な選択と言えるだろう。Red Hat Linux 6.2Jのデフォルトでは、ここでGNOMEが選択されている。

通常は、**リスト3**に示す範囲で、セ

▼リスト3　Red Hat Linux 6.2JのXclients（抜粋

```
if [ -f /etc/sysconfig/desktop ]; then
    if [ -n "`grep -i GNOME /etc/sysconfig/desktop`" ]; then
        PREFERRED=gnome-session
    elif [ -n "`grep -i KDE /etc/sysconfig/desktop`" ]; then
        PREFERRED=startkde
    elif [ -n "`grep -i AnotherLevel /etc/sysconfig/desktop`" ]; then
        PREFERRED=AnotherLevel
    fi
fi
if [ -n "$PREFERRED" -a "$PREFERRED" != "AnotherLevel" ] && \
        which $PREFERRED >/dev/null 2>&1; then
    PREFERRED=`which $PREFERRED`
    exec $PREFERRED
fi
   :
   :
if [ -z "$PREFERRED" ]; then
    GSESSION=gnome-session
    STARTKDE=startkde
    # by default, we run GNOME.
    if which $GSESSION >/dev/null 2>&1; then
        exec `which $GSESSION`
    fi
    # if GNOME isn't installed, try KDE.
    if which $STARTKDE >/dev/null 2>&1; then
        exec `which $STARTKDE`
    fi
fi
```

/etc/sysconfig/desktopにGNOMEの文字列があれば、GNOMEを起動。KDEがあれば、KDEを起動

/etc/sysconfig/desktopに何もなければ、GNOMEを起動

GNOMEがなければ、KDEを起動

ッションマネージャが選択・起動されるが、desktopファイルに、“ Another Level ”が指定されている場合に限り、別のセッションマネージャが選択される。

まず、ホームディレクトリの.wm_styleというファイルを参照して、After Step、WindowMaker、fvwmなどのウィンドウマネージャを起動しようと試みる。それでもうまくいかない場合は、fvwm2、またはtwmをを起動する。

独自構成のKondara MNU/Linux

上記の説明は、Red Hat Linux 6.2Jを元にしているが、Vine Linux 2.1、LASER5 Linux 6.2でもほぼ同じと考えてよい。

しかしKondara MNU/Linux 2000には、lang.d、session.d、xim.dという3つのサブディレクトリが存在している。これらはそれぞれ、Kondaraのwdmで設定できる“ language ”、“ Session ”、“ Input Method ”に対応しており、用意されたスクリプトにより、環境変数などが適切に設定される。これらの設定は、コマンドラインからsdrコマンドを用いて変更することも可能である。

Column

ユーザー固有の設定ファイル

Linuxディストリビューションをインストールすると、大多数のユーザーがそれほど違和感なく使えるような、設定ファイルが用意されている。最初のうちは、与えられた環境をそのまま使うのもいいが、慣れてきたら、徐々に自分の好みに合わせてカスタマイズしてみよう。使いやすい環境が得られるし、何よりカスタマイズを通じて、Linuxシステムへの理解が深まるからだ。

/etc/X11/xinit/xinitrcと~/.xinitrcの関係ように、UNIX系OSの設定ファイルは、全ユーザー共通のものと、ユーザー固有のものが分かれていることが多いので、他のユーザーに影響を与えずにカスタマイズが可能だ。

いきなりゼロから環境を構築するのは大変だし、無駄でもあるので、共通の設定を活かしたうえで、カスタマイズしていくのがお勧めだ。たとえば~/.xinitrcの先頭に、

```
exec /etc/X11/xinitr/xinitrc
```

という行を書いておけば、まず共通の設定、続いてオリジナルの設定が行える。

ディスプレイマネージャ

ディスプレイマネージャは、グラフィカルなログイン画面を表示して、ユーザーがログインしてくるのを待っているプログラムだ。X Window Systemの標準ディスプレイマネージャは、xdmだが、多くのディストリビューションでは、xdmを改良したgdm、wdm、kdmといったディスプレイマネージャも含んでいる。今回取り上げたディストリビューションでは、gdmまたはwdmがデフォルトのディスプレイマネージャである。

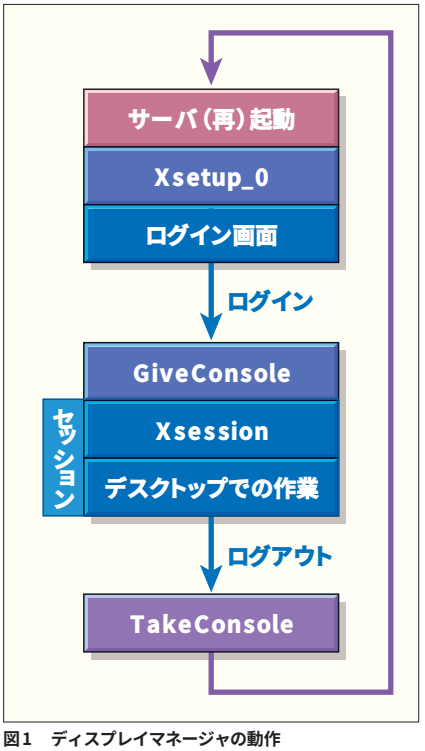


図1　ディスプレイマネージャの動作

ディスプレイの管理については、ローカルマシン上で動作するXサーバ以外に、ネットワーク上のほかのマシン上のXサーバと通信することも可能だ。この際に用いられるプロトコルを、XDMCP（X Display Manager Control Protocol）という。これは高価なワークステーションに、比較的安価なX端末と呼ばれる端末専用マシンでアクセスするときに、よく使われていた方式だ。それぞれのマシンが非常に高性能かつ安価になった現在では、それほど利用されていないようだ。実際、Linuxディストリビューションをデフォルトでインストールした状態では、XDMCPは使用しない設定になっている。

設定ファイル

xdm、gdm、wdmに関する設定ファイルは、それぞれ/etc/X11のサブディレクトリに格納されている。主なファイルを表1に示した。xdm、gdmについては、Red Hat Linux 6.2Jのもの、wdmはKondara MNU/Linux 2000のものだ。

外観や、ウィンドウマネージャの選択が可能といった改良点はあるが、機能的には同等だということがわかるだろう。表1を見ると、wdmにXsession相当のスクリプトはないように見えるが、実際はwdm-config内で、

```
/etc/X11/xdm/Xsession
```

が指定されている。また、gdmの各種スクリプトは、表中のサブディレクトリ内にある“Default”というファイルが用いられる。Red Hat Linux 6.2Jの場合は、/etc/X11/xdm以下の相当するファイルへのシンボリックリンクになっている。

これらの設定ファイル群は、ディストリビューション間の差が大きいので、同じディスプレイマネージャであっても、ファイル構成が異なることが多い。これらの差異は、各ディストリビューションのポリシーの違いによるものであり、比較することでそれぞれの開発者の考えが見えてくるようで、興味深い。

ディスプレイマネージャの動作

ディスプレイマネージャは、単にログイン画面を表示するだけでなく、**図1**のような流れでほかのプログラムと協調して機能している。以下、Red Hat Linux 6.2Jのxdmを例に、実際の動作を追ってみよう。

まず、/etc/inittabに指定された、prefdmというシェルスクリプトによって、ディスプレイマネージャ（この場

ディスプレイマネージャ	xdm	gdm	wdm
ディストリビューション	Red Hat Linux 6.2J	Red Hat Linux 6.2J	Kondara MNU/Linux 2000
設定ファイル	xdm-config	gdm.conf	wdm-config
Xサーバ起動後実行するスクリプト	Xsetup_0	Init/	Xsetup_0
セッションの最初で実行するスクリプト	Xsession	Sessions/	—
コンソールのオーナーをユーザーに渡す	GiveConsole	PreSession/	GiveConsole
コンソールのオーナーをrootに返す	TakeConsole	PostSession/	TakeConsole
XDMCPで使うリモートのXサーバを指定	Xaccess	—	Xaccess
ローカルマシンで動作するXサーバを指定	Xservers	—	Xservers
リソースファイル	Xresources	—	Xresources

表1　ディスプレイマネージャ関連ファイル
基本的な設定ファイルは、同じものが揃っている

合はxdm）が起動する。xdmは、

```
/etc/X11/X
```

で指定されたXサーバを起動する。このファイルは、グラフィックスアクセラレータによって異なるXサーバへのシンボリックリンクになっているが、たいていの場合、実体は、

```
/usr/X11R6/bin/XF86_SVGA
```

になっている。Xサーバ起動後には、Xsetup_0が実行されるが、デフォルトでは、背景の色と画像を指定しているだけだ。この段階で、ログイン画面が表示される。

ユーザー名とパスワードを正しく入力しログインに成功すると、セッションが開始される。セッションとは、ログインからログアウトするまでの期間と考えればよい。

いったんセッションを終了させても、次にログインした時には、前回のセッション終了時の状態が復活すると便利だ。この機能を提供するのが、セッションマネージャである。GNOME／KDEといったデスクトップ環境や、いくつかのウィンドウマネージャがこの機能を持つ。ディスプレイマネージャを用いた環境では、使用するセッションマネージャを選択できる。これらの設定は、セッション開始時に実行される、

```
/etc/X11/xdm/Xsession
```

というスクリプトで行っている。

Xsession

Xsessionは、多岐に渡った設定を行っているうえに、Xsessionから呼ばれるスクリプトも長くて複雑なものがあるため、全体を把握するのは大変だ。そこで、いくつかのセクションを抜き出して、どのような設定を行っているのか見てみよう（**リスト1**）。

最初の部分は、システム／ユーザーのリソースファイル指定を行っている。そのマシンに特有のリソース設定や、ユーザー個人で必要な設定を、リソースデータベースに統合している。ここで参照されているファイルは、初期状態ではどれも存在しないので、必要に応じて足せばいいだろう。

次の部分は、xinitが起動時に参照するスクリプトファイルを実行している。xinitは、コマンドラインからXを起動するためのプログラムなので、これらのスクリプトを実行することで、Xの起動方法にかかわらず同じデスクトップ環境が得られることになる。

最後の部分は、ユーザーのホームディレクトリにある“.xsession”、“.Xclients”、またはxinitサブディレクトリ以下にある“Xclients”のどれかひとつを実行している。“.xsession”、“.Xclients”とも初期状態では存在しないので、“Xclients”が実行されることになる。xinitのところで説明されているが、ここではセッションマネージャの設定を行っている。

▼リスト1　Red Hat Linux 6.2Jのstartx（抜粋）

```
userresources=$HOME/.Xresources
sysresources=/etc/X11/xinit/Xresources
oldsysresources=/etc/X11/xinit/.Xresources

if [ -f "$oldsysresources" ]; then
    xrdb -merge "$oldsysresources"
fi

if [ -f "$sysresources" ]; then
    xrdb -merge "$sysresources"
fi

if [ -f "$userresources" ]; then
    xrdb -merge "$userresources"
fi
  :
  :
for i in /etc/X11/xinit/xinitrc.d/* ; do
    if [ -x "$i" ]; then
        . "$i"
    fi
done
  :
  :
if [ -x "$HOME/.xsession" ]; then
    exec "$HOME/.xsession"
elif [ -x "$HOME/.Xclients" ]; then
    exec "$HOME/.Xclients"
elif [ -x /etc/X11/xinit/Xclients ]; then
    exec /etc/X11/xinit/Xclients
else
    # should never get here; failsafe fallback
    exec xsm
fi
```

リソース設定。システム、ユーザーリソースをデータベースに登録

/etc/X11/xinit/xinitrc.d/以下のスクリプトを実行

~/.xsession、~/Xclients、/etc/X11/xinit/Xclientsのどれか1つを実行

ウィンドウマネージャ

何段階もの設定を経て、X Window Systemが起動し、ウィンドウマネージャによって管理されるデスクトップに到達する。ウィンドウマネージャやデスクトップ環境は、ユーザーの好みに応じて、色、フォントの種類やサイズなど多彩なカスタマイズが可能である。これらの設定内容を、/etc/X11以下のディレクトリに置いているウィンドウマネージャもいくつかある。

UNIX系OSでは、設定ファイルはテキスト形式で、ユーザーが読んだり、エディタで編集したりできるようになっているのが一般的だ。最近では、GUI上からこれらの設定を行えるようになっている環境もある（**画面1**）。もっとも、以前からUNIXを使用しているユーザーには「GUI上のツールはどのファイルを変更しているかわかりにくい」、「エディタで修正した場合、整合性がとれなくなる気がする」といった理由で、あまり人気がないようだ。

だが、テキスト形式の設定ファイルをGUIツールで操作するというのは、UNIX系OSとWindowsの「良いとこ取り」であり、慣れてしまうと非常に快適ではある。

ふだんの設定はGUI上で行うにしても、一度くらいは設定ファイルをエディタで開いて、中身を見てみよう。システムへの理解を深まるはずだ。

画面1　KDEコントロールセンター
ほとんどの設定がGUIで行える。

fvwm2
fvwm95
mwm
AfterStep
icewm
WindowMaker
KDE

表1　wmconfigに対応したウィンドウマネージャ

wmconfig/

Red Hat Linuxには、wmconfigというプログラムが用意されている。wmconfigは、/etc/X11/wmconfigディレクトリ内のファイルと、ユーザーのホームディレクトリに置かれた“.wmconfig”ファイルを参照して、ウィンドウマネージャの設定ファイルを出力してくれるユーティリティだ。**表1**に示した7種類のウィンドウマネージャに対応している。

各設定ファイルは、「名前」「説明」「アイコンファイル」「プログラム名」「グループ」などを記述するようになっている。このファイルは、プログラムの作者または、rpmパッケージの作成者によって、rpmパッケージに同梱される。適切に設定しておけば、インストールしたプログラムをウィンドウマネージャのアプリケーションメニューに登録することができる。**リスト1**に元のファイルと、出力ファイルの例を示す。

▼リスト1　wmconfigの例（xosview）

元のデータ（xosview）

```
xosview name "xosview"
xosview description "OS Stats Viewer"
xosview group Administration
xosview exec "xosview &"

AfterStep
Popup   "Administration"
        Title   "Administration"
        # Added by the "xosview" package
        Exec    "xosview"       exec xosview &
EndPopup


WindowMaker
        "Administration" MENU
                "xosview" EXEC xosview &
        "Administration" END
```

WindowMaker/

美しい外観と軽快な動作で、根強い人気を持つウィンドウマネージャ、WindowMakerの設定ファイルは、/etc/X11/WindowMakerディレクトリに置かれている（表2）。いずれもテキストファイルなので、エディタを用いて変更することも可能だが、通常はWindowMakerの設定ツールを用いる。

これら以外にも、/usr/share/WindowMaker以下とホームディレクトリのGNUstepディレクトリ以下に、ロケール、アイコンファイル、テーマなどが用意されている。

applnk

applnk/ディレクトリ以下には、KDEやGNOMEのメニューに用いるための、各アプリケーションの設定ファイルが置かれている。ファイル形式は、**表2**のようになっており、アイコンファイル、グループ、アプリケーション名とロケール別の名前、コメントが記されている（**リスト2**）。

Red Hat Linux 6.2JのKDEのメインメニューには、“ Red Hat ”という項目があり、その中を見てみると、/etc/X11/applnk以下のディレクトリ構成がそのまま反映されていることがわかる（**画面2**）。これは、設定ファイル内でアイコンファイル名を指定している項目だけ、メニュー上に独自のアイコンが現れるようになっていることからも確認できる。

実際には、KDEのメインメニューは、/usr/share/applnkディレクトリ以下の構成がそのまま反映されている。**リスト3**に示すように、/usr/share/applnk内からシンボリックリンクが張られているので、/etc/X11/applnkの内容がメインメニューに現れるというしかけになっている。

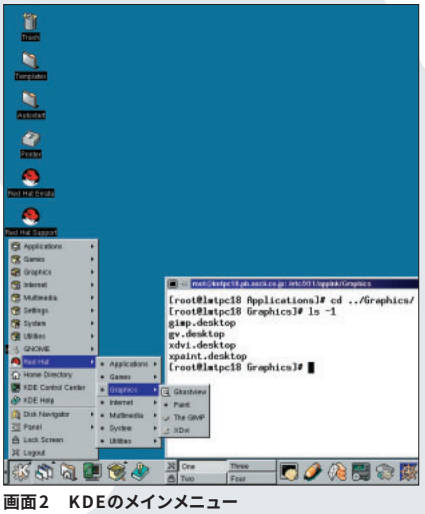
メインメニュー内には、同様に“ GNOME ”という項目も存在する。これも/usr/share/applnk内から/usr/share/gnome/appsに張られたシンボリックリンクである。/usr/share/gnome/appsは、GNOMEのメインメニューの項目に相当するファイルが置かれている。予想がつくと思うが、GNOMEとKDEのメニュー項目には、互換性があるのだ。試しに/usr/share/gnome/apps内から、/etc/X11/applnkにリンクを張ってGNOMEを起動してみると、ちゃんと“ Red Hat ”というメニュー項目が表示された（**画面3**）。

TurboLinux Workstation 6.0では、/etc/X11ディレクトリ以下に、applnkに加えて、applinkというサブディレクトリも存在するが、どちらもメニューには反映されておらず、中身のファイル数も少ない。Red Hat Linuxと同じような使い方はしていないようだ。

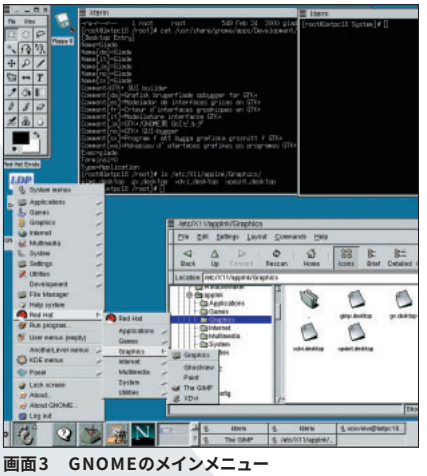
急ぎ足で/etc/X11ディレクトリの探索をしてみたが、思ったよりも奥が深いようだ。今回の特集を足掛かりに、X Window Systemを攻略してみよう。

WMGLOBAL	表示フォントなど、システム全体の共通設定項目
WMRootMenu	ルートウィンドウをクリックした時に表示されるメニューの項目
WMState	ルートウィンドウに表示されているドックの設定項目
WMWindowAttributes	アイコンファイルの指定
WindowMaker	全般的な設定項目

表2　/etc/X11/WindowMaker内の設定ファイル



画面2　KDEのメインメニュー
ディレクトリ構造がメニューに反映されている。



画面3　GNOMEのメインメニュー
メニューの構成方法は、KDEと同じだ。

▼リスト2　emacs.desktop（抜粋）

```
[Desktop Entry]
Name=Emacs
Name[ca]=Emacs
Name[da]=Emacs
Name[de]=Emacs
Name[es]=Emacs
Name[eu]=Emacs
Name[fi]=Emacs
Name[fr]=Emacs
Name[it]=Emacs
Name[ja]=Emacs
Comment=Emacs text editor
Comment[ca]=L'editor de text Emacs
Comment[da]=Emacs tekst editor
Comment[de]=Der Texteditor Emacs
Comment[es]=Editor de textos Emacs
Comment[eu]=Emacs testu editorea
Comment[fi]=Emacs-tekstieditori
Comment[fr]=Editeur de texte Emacs
Comment[hu]=Emacs sz♪egszerkeszt
―omment[it]=Editor di testo Emacs
Comment[ja]=Emacsテキストエディタ
TryExec=emacs
Exec=emacs
Icon=emacs.png
Terminal=0
Type=Application
```

▼リスト3　/usr/share/applnkの内容

```
Applications/
Games/
Graphics/
Help.kdelnk
Home.kdelnk
Internet/
KControl.kdelnk
Multimedia/
Settings/
System/
Utilities/
gnome -> ../gnome/apps
redhat
  -> ../../../etc/X11/applnk
```

連載特集

申し込みから独自ドメイン運用まで徹底解説！

フレッツ・ISDNで常時接続インターネットサーバ!

独自サーバで自分だけのインターネットサイトを構築しよう

フレッツ・ISDNの登場によって、独自ドメインによるサーバ運用が身近なものになってきた。さっそく、ドメインを取って、独自サーバを運用しよう！　もちろん、ホスティングサービスや、ドメイン名の転送サービスなどを利用するのではなく、すべてを自前で用意する。CGIだって、telnetだって、IMAP4だって自由に使いまくりのやりたい放題。メールアドレスだって好きなものが作れる。そんな自分だけの夢のサーバを構築しよう！

ACT. 1

フレッツ・ISDNとプロバイダの申し込み

今年6月からサービスが開始されたフレッツ・ISDNによって、誰でも気軽にインターネット常時接続環境を手に入れることができるようになった。もちろん、フレッツ・ISDNの本来の使い方はインターネットの定額利用にあるのだが、このサービスを利用することでインターネット常時接続環境が実現できることは間違いない。

今回の連載特集は、このフレッツ・ISDNを使って独自サーバを構築しようというものだ。

最終目的は独自ドメインの運用であり、DNSもセキュリティもメールサーバもすべて独自のものを構築するという主旨である。もちろん、これらのことはフレッツ・ISDNに限った話ではなく、ダイヤルアップ環境から、CATVやxDSLを使った常時接続、専用線まで、全般多岐にわたる内容になる。部内サーバなどのLANから、ホームサーバまで活用できる実践的な内容になる予定なので期待してほしい。

また、この連載特集では、編集部でも実際にフレッツ・ISDNを導入し、独自ドメインによる運営までを連載に合わせて構築していく。実際に申し込むことによって、読者と同じ視点に立って記事を掲載していきたい。構築したサーバは、実際に公開する予定だ。

大まかには、次のものを取り上げる予定だ。

- **フレッツ・ISDNの導入**
- **フレッツ・ISDNによるインターネット接続**
- **セキュリティ設定**
- **DNSサーバの構築**
- **ドメイン取得**
- **メールサーバの構築**

これ以外にも取り上げてほしい話題や、サーバを構築するうえで疑問点などがあったら、Web上のアンケートページを使って編集部まで送ってほしい。随時、記事に反映していきたい。

フレッツ・ISDN

フレッツ・ISDNは、NTT東日本、NTT西日本が提供する公衆回線を利用した接続料定額の回線サービスだ。インターネット接続サービスではないので、別途プロバイダとの契約が必要になる。今までのISDN回線を使ったプロバイダ経由の接続とほぼ同等の接続サービスで、電話回線使用料が定額になったと考えればよい。このため、新たに機器を用意したりすることなく、既存の機器をそのまま流用することができる。唯一違うのは、ISDN回線を

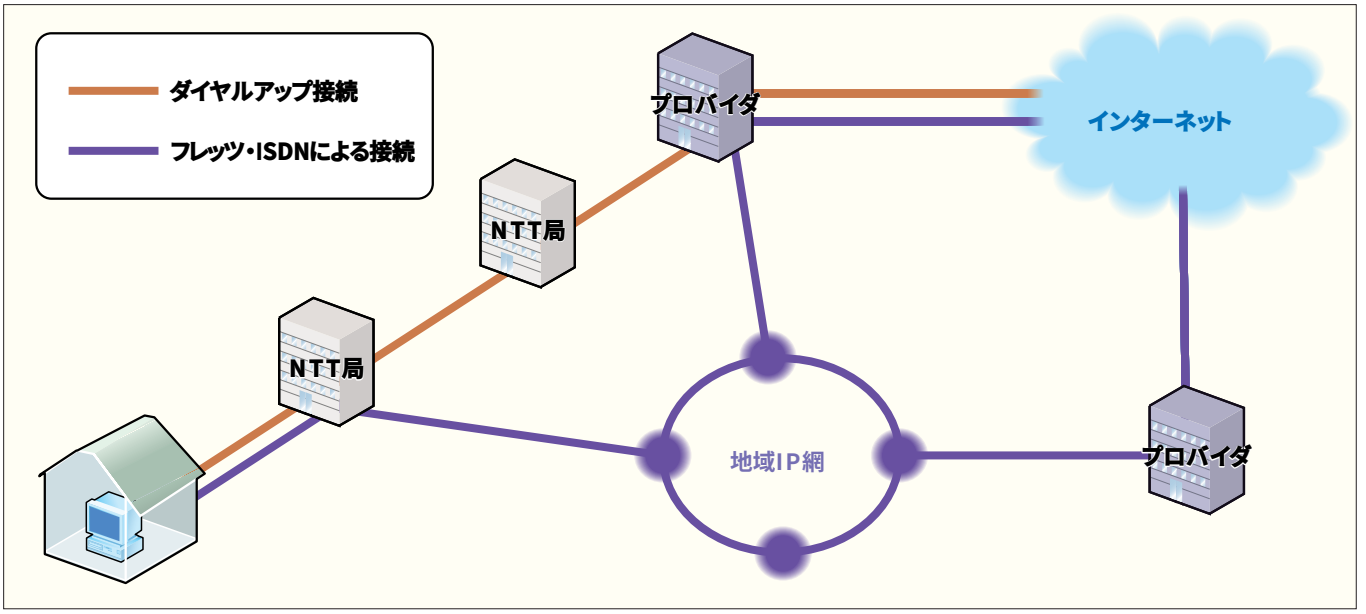


図1　フレッツ・ISDNと通常の接続の違い

使って直接プロバイダに接続するのではなく、NTTの地域IP網経由でプロバイダに接続する点だ（**図1**）。つまり、接続するのはプロバイダではなく、NTTの地域IP網ということになる。この地域IP網への接続には、専用の電話番号が与えられ、確実に接続できるようになっている。

この方式により、プロバイダはTAなどの新たな機器に投資することなく常時接続サービスを提供することができるのだ。また、ユーザーも特に新たな機器や回線を設置することなく利用できるというメリットもある。

回線の使用料は4500円と、OCNなどにくらべて安くなっている。今までの23時から翌8時までの時間帯のみ、接続料が固定となったテレホーダイの1800円（同一地区内）に比べるとまだまだ割高ではあるが、24時間つなぎっぱなしにできるという点が嬉しいサービスだ。

いいことずくめのフレッツ・ISDNだが、回線速度という点においては若干難がある。回線速度が64kbps（1Bチャンネル）のみの提供だということだ。テレホーダイでは、1Bチャンネルを2つ利用（MP）して128kbpsでの通信が可能だったが、フレッツ・ISDNでは64kbpsでしか利用できない。また、最大速度が64kbpsのベストエフォート型のサービスであるため、実際に64kbpsという帯域が保証されるものでもない。さらに、常時接続を保証するサービスでもない。実際、サービス開始から何度かつながらないなどの問題が発生している。このため、正確には常時接続環境とはいえず、クリティカルなインターネットサービスの提供には向いていない。

しかしながら、比較的サービスエリアが広く、低価格な定額インターネット向け回線サービスと考えるとフレッツ・ISDN以外の選択肢はないだろう。いままでCATVやxDSLによるサービスエリア外であったユーザーにとっては福音に違いない。

なお、フレッツ・ISDNで定額の対象となるのは、専用の電話番号へのデジタル通信接続のみとなる。これ以外の電話番号への発信や、モデムによるアナログ接続の場合は、フレッツ・ISDNの対象外となるので注意しよう。

複数のプロバイダを切り換えて接続することも、NTT地域IP網に接続する際のユーザー名とパスワードを変えることで（これはプロバイダから提供される）可能だ。これは、直接プロバイダに接続するのではなく、NTTの地域IP網に接続するために実現できる仕組みだ。こういった点は、接続先電話番号固定のテレホーダイに比べて使いやすいといえるだろう。

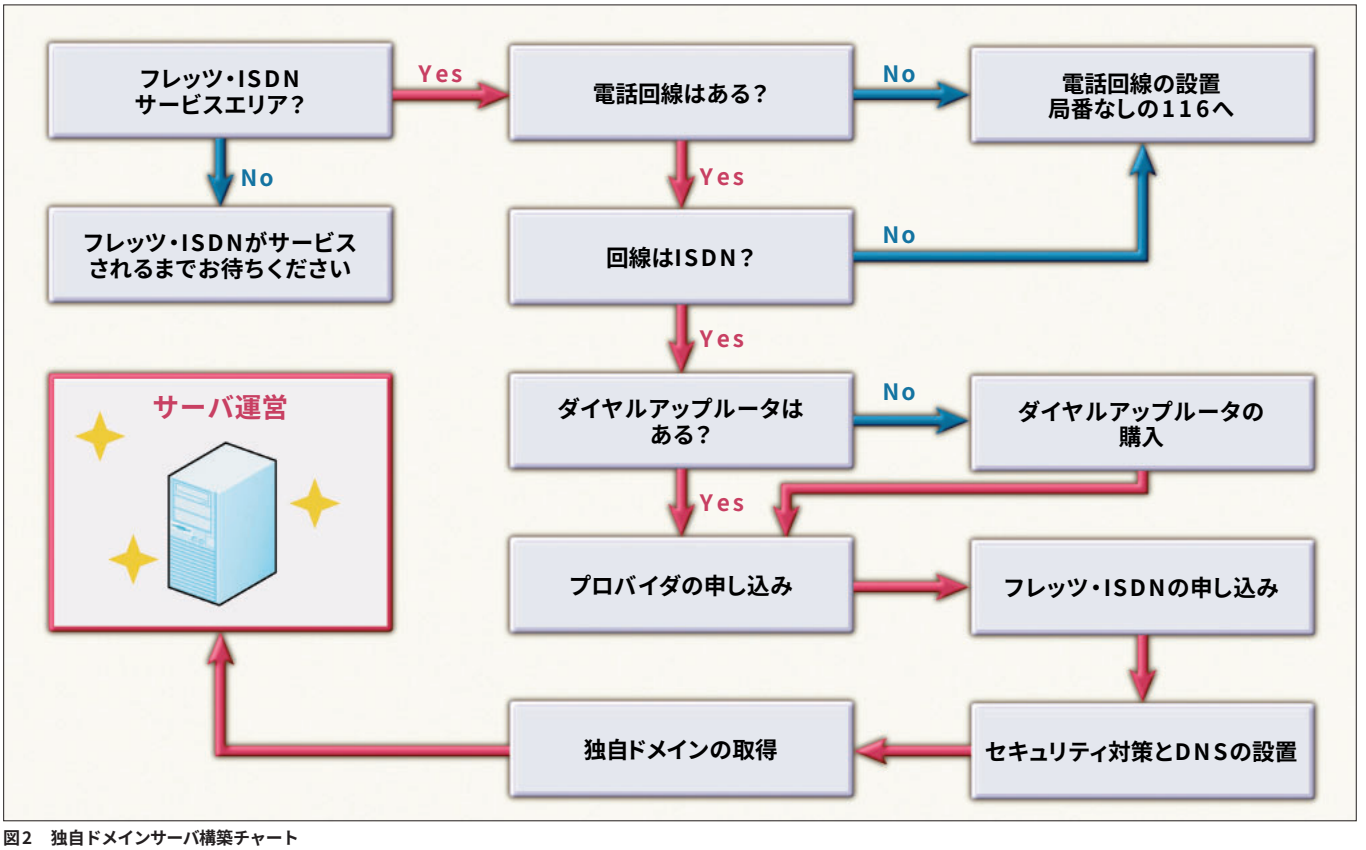


図2　独自ドメインサーバ構築チャート

固定IPアドレス

前述したように、フレッツ・ISDNは本来であればインターネットの公開サーバ用途に利用できるものではない。また、プロバイダにとってみれば接続先が地域IP網に変わったものの、実質的なサービスは変わっていない。サーバを構築する際に必須となるIPアドレスも、従来通り動的にグローバルIPアドレスを割り当てる。動的なグローバルIPアドレスの割り当てとは、プロバイダに接続した際に、プロバイダが所有する空いているIPアドレスを割り当てるもので、接続のたびにIPアドレスが変更される。つまり、接続するたびにIPアドレスが変わってしまうのである。これでは、独自ドメインの取得はおろか、サーバ構築すらままならない。毎回、IPアドレスが変わってしまうのであれば、外部からサーバを特定することができないからだ。

しかし、プロバイダの中には静的（固定）グローバルIPアドレスを割り当ててくれる付加サービスを提供しているところも出てきている。静的グローバルIPアドレスでは、IPアドレスが固定的に割り当てられ、常に同じIPアドレスが利用できるということになる。こういったプロバイダのサービスを利用し、静的グローバルIPアドレスの割り当てが保証されれば、サーバを構築することも可能だし、ドメインを取得することも可能になる。一気に独自ドメイン取得が可能になるということだ。

この連載でもこのようなプロバイダを利用してサーバを構築することにする。CATVやxDSLなどのエリア内であっても、プロバイダ内のプライベートIPアドレスや動的IPアドレスしか利用できなかった、あるいは静的グローバルIPアドレスサービスは高価で利用できなかったユーザーもサーバの構築が現実的なものとなるだろう。

フレッツ・ISDNの初期費用

フレッツ・ISDNは、すでにISDN回線が設置されていれば、NTT局内工事のみで開通することができる。工事費は、基本工事費と交換機等工事費を合わせて2000円だ。

申し込みから開通までは、最低2週間はかかる。

サーバ構築までの流れ

さて、独自サーバを構築するまでの手順を紹介しよう。

実際、独自ドメインの運用となると、かなり複雑な手続きを踏まなければならない。ダイヤルアップ接続とは桁違いに難しくなるのだ。また、独自サーバを構築するためには、静的グローバルIPアドレスの取得や、DNSの設置など、先に準備をしておかなければならないものがある。静的グローバルIPアドレスとDNSが設置されていないとドメインは取得できないからだ。なかなか大変なのだ。

それでは実際に必要な手順を順を追って説明していこう。

①ダイヤルアップルータの購入
②プロバイダの申し込み
③フレッツ・ISDNの申し込み
④DNSサーバの設置
⑤ドメインの取得

以上が、独自ドメインのサーバを構築する際に、最低限必要な手順だ。

ダイヤルアップルータはインターネットに接続する場合に必要となる接続機器だ。TAにルータが組み込まれている機器だと考えればよい。TAとLinux自身をルータにして運用することも可能だが、シリアル接続の煩雑さや、ルータとしての設定の難しさなどを考えると、ダイヤルアップルータを利用したほうがずっと簡単だ。ダイヤルアップルータはファイアウオールとしても利用できるので、常時接続の必須アイテムといえるだろう。

新規にISDNに移行するのであれば、DSU内蔵のダイヤルアップルータを購入するといいだろう。

なお、②と③は同時でもかまわない。しかし、一方だけ先に開通したとしても、もう一方が開通するまでは利用できないので、ムダな出費となってしまう。NTTとプロバイダに問い合わせを

Column

もうひとつのフレッツ・ISDN

フレッツ・ISDNには、4500円／月の定額サービスのほかに、2900円／月の定額サービス（収容局単位での接続）というのがある。これは、利用者が使用するISDN回線が収容されているNTT局舎内の加入者収容モジュール（ISM）から、直接プロバイダに接続するサービスだ。地域IP網を利用せずに、NTTの収容局から直接プロバイダに接続するという形式だ。このサービスを利用するには、プロバイダの回線と同じ収容局に利用するISDN回線が含まれていなければならない。簡単にいうと、「プロバイダが近くにあるのなら、お安くしときます」というサービスだ。

といっても、残念ながらこのサービスを提供しているプロバイダは存在しないようなので、事実上サービスは行われていない。NTTのホームページでも一応記載されている程度で、いつのまにか消えてしまいそうな扱いになっている。

して、同時に開通できるように手配しよう。

プロバイダの選択

すでに、ほとんどのプロバイダがフレッツ・ISDNに対応したサービスを提供している。すでにプロバイダに加入している方であれば、そのまま利用することができるかもしれない。

しかし、この連載の目的は独自ドメインによる運用にあるので、静的グローバルIPアドレスをサービスしているプロバイダを選択する必要がある。

残念ながら、現状では静的グローバルIPアドレスを提供しているプロバイダは少ない。

表1は、編集部で調査した、現在静的グローバルIPアドレス割り当てサービスを行っているプロバイダだ。このほか、現在検討中というプロバイダもいくつかあるので、順次こういったサービスを提供するプロバイダが増えてくると予想される。正式サポートでは

プロバイダ名	サービス名	月額料金（合算）	サービス提供会社	URL
CDS-Net	フレッツISDN対応サービス2	1800円	有限会社回路設計サービス	http://www.cds.ne.jp/
CISネット	ＩＰアドレス固定割当	1917円（換算）	中国情報システムサービス株式会社	http://www.cisnet.or.jp/
DCNインターネットサービス	フレッツＩＳＤＮ固定ＩＰプラン	2450円	ディーシーエヌ株式会社	http://www.dcn.ne.jp/
Inforyukyu	しっかりコース	4800円	株式会社沖縄富士通システムエンジニアリング	http://www.ryukyu.ad.jp/service/fletsi/
InfoSphere	Biz IP8	6800円	NTT PCコミュニケーションズ	http://www.sphere.ne.jp/ip_service/service/index.html
VC-net	タイプB	2500円	ビジュアルシステム株式会社	http://www.vc-net.ne.jp/ipmenu.html
ZOOT	MOOT	2400円	株式会社インターリンク	http://il24.net/
アンフィニー	Econo-mist	12800円	株式会社デルタインテグラル	http://www.infini.ne.jp/eco/
インターネットWIN	WIN IP接続サービス	3000円	株式会社ウインシステム	http://www.win.ne.jp/
かもめインターネット	IPアドレス固定型	3834円（換算）	株式会社ネットフォレスト	http://www.kamome.or.jp/
キャンパスネット	固定IPサービス	3300円	株式会社インターネットコミュニケーションサービス	http://www.campus.ne.jp/
ネスク	フレッツ固定IPアドレスサービス	4500円	株式会社ネスク	http://www.nsknet.or.jp/
多摩インターネット	FLET'S 対応 IP 固定型ダイヤルアップ(1IP)	4000円	有限会社多摩インターネットサービス	http://www.tama.or.jp/flets.html
ホッカイ・ネット	ホッカイ・ネットオンライン24	4800円	ホッカイ・ネット	http://www.hokkai.net/online24/

表1　静的グローバルIP割り当てサービスを行っているプロバイダー覧

Column

Biz IP8申し込み書郵送の怪

編集部で実際にBiz IP8の申し込み書とパンフレットの郵送をInfoSphereに電話でお願いしたのだが、なぜか郵送には時間がかかるらしい。

編集部「あの、御社でサービスしているBiz IP8のパンフレットと申込書をいただきたいのですが」
InfoSphere「お問い合わせありがとうございます。お届けまでに1週間から2週間ほどかかりますがよろしいでしょうか」
編集部「え？　いや、申し込みじゃなくって、申込書とパンフレットが欲しいのですが」
InfoSphere「はい。1週間から2週間ほどお時間をいただいております」
編集部「申込書の送付ってそんなに時間がかかるのですか？」
InfoSphere「はい。申し訳ありませんが、お時間をいただいております」
編集部「……（ん〜。審査でもされているのだろうか）。これ以上はどうしても早くならないのですか？」
InfoSphere「はい。申し訳ありません」
編集部「どうしてそんなに時間がかかるのですか？」
InfoSphere「申し訳ありませんが、お時間をいただいております。申し込み書でしたら、インターネットでダウンロードできますので、そちらをご利用していただくことも可能ですが」
編集部「はぁ。そうですか。そんなにかかるのですか……（ヤバイな、原稿間に合わねーぞ）。どこかに置いてあるのでしたら、取りにお伺いいたしますが」
InfoSphere「申し訳ありませんが、郵送とインターネットでのみ提供させていただいております」
編集部「そうですか……それでは、一応送っていただけますか……」
InfoSphere「それではお送りいたします。ありがとうございました」

というわけで、申込書とパンフレットの入手だけで1、2週間ほどかかってしまうらしい。インターネット時代になったとしても、事務手続きには時間がかかってしまうのが世の常なのだろうか。Webページからダウンロードすることは可能だが、もう少し早く入手できてもいいのではないかと思う。

しかし、どうしてそんなに時間がかかるんでしょうねぇ。

ないが、要交渉というプロバイダもあるので、一度、プロバイダに問い合わせを行ったほうがいい。

なお、フレッツ・ISDNを使う場合、地域ごとにサービスしているプロバイダが異なるので注意が必要だ。通常のプロバイダのサービスのように、アクセスポイントが提供される形式とは異なるからだ。

編集部では、NTT PCコミュニケーションズがサービスしている、InfoSphereのBiz IP8に申し込むことにした。InfoSphereのBiz IP8では、静的グローバルIPアドレスを8個（利用可能なIPアドレスはルータ分を含め6個）も割り当ててくれるという太っ腹なサービスなのである。IPアドレスを8つも割り当ててくれるサービスは、ほかのプロバイダではちょっと見あたらない。複数サーバの運用も可能になる点が嬉しいところだ。だたし、ほかのプロバイダに比べると、若干料金が高い。

申し込み時に、加入費用として5000円、IPアドレス新規登録料として1万5000円、計2万円ほど必要になる。月額の使用料は6800円となっている。フレッツ・ISDNの使用量の4500円と合わせると、月々1万1300円ほどかかる計算だ。

編集部ではBiz IP8を選択したが、ほかにも静的IPアドレスを1つだけ割り当てるBiz IP1（4800円）というサービスもある。

Biz IP8の申し込み

Biz IP8の申し込みは郵送のみである。申込書とサービスの概要が紹介されたパンフレットは郵送で入手することもできるが、申込書とパンフレットの郵送だけでも1、2週間かかるということなので、InfoSphereのWebサイトでサービスを確認し、申込書（PDF）をダウンロードするほうがいいだろう。実際、送られてくるパンフレットはWebで公開されている内容とさほど変わらない。

申し込み書は2枚綴りで、かなり複雑になっている。サンプル（**写真1・2**）を参照してほしい。申し込み受理からサービスの開始までは2週間から3週間ほどかかる。通常のプロバイダの申し込みに比べて開通までに時間がかかるのは、JPNICにIPアドレスを申請する時間があるので仕方のないところだ。

場合によっては、フレッツ・ISDNの申し込みから実際にインターネットに接続できるまで2カ月近くかかる可能

Column

フレッツ・ISDN申し込みのオカルト

フレッツ・ISDNの申し込み方法には、郵送による申し込み、Webサイトからの申し込み、電話での申し込みの3つがあるのだが、電話による申し込みが一番早いというウワサが流れている。インターネットといっても、やっぱり人と人のふれあいでしょう……というのもわからなくはないが、真相のほどは不明だ。このウワサのせいで、電話での申し込みが殺到してフリーダイヤルがつながらないという事態に陥っているらしい。ウワサに惑わされず、Webで申し込むのが賢明だ。

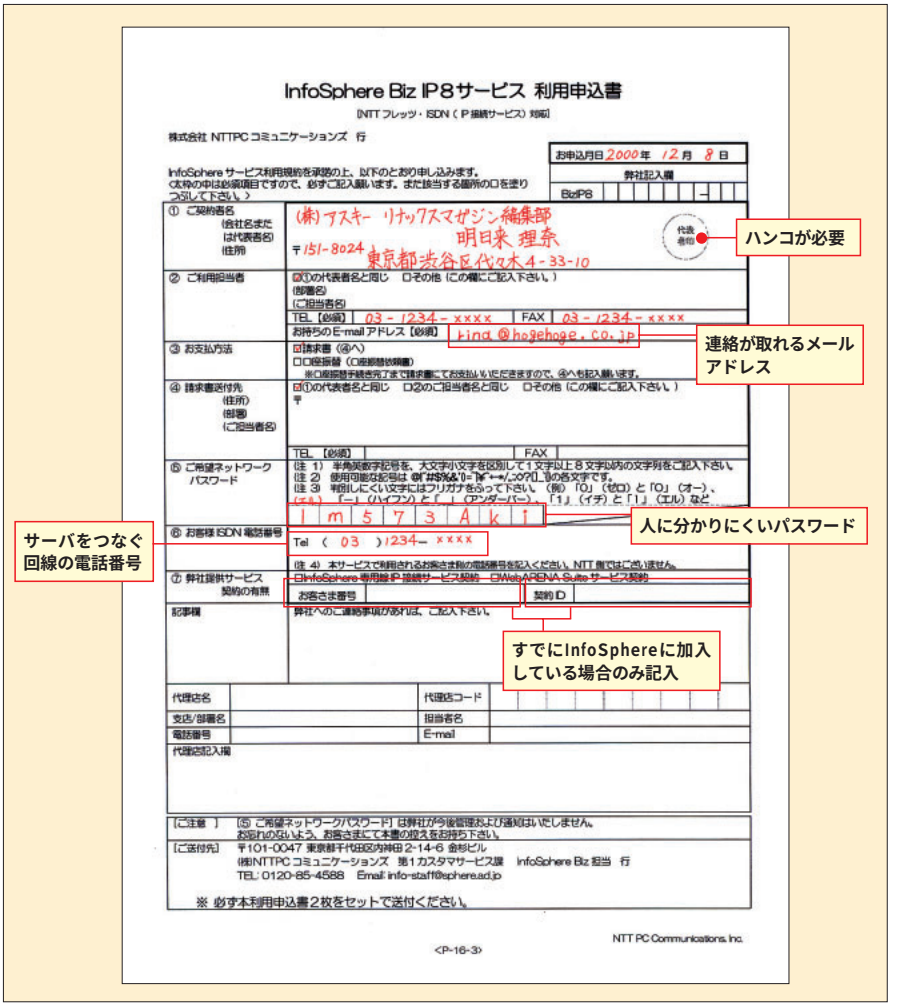


写真1　InfoSphere Biz IP8サービス利用申込書（1/2）

性もある。インターネット時代の今日でも、事務手続きには時間がかかってしまうようである。残念ながら仕方がないので開通を気長に待とう。

その間にドメイン名でも考えておくといいだろう。

フレッツ・ISDNの申し込み

フレッツ・ISDNは、順次サービスエリアを拡大しており、多くの地域で利用できるようになってはいるものの、現状で全ての地域でサービスを提供しているわけではない。**表3**のフレッツ・ISDN提供エリアを参照してほしい。この表に載っていない地域の場合でも、サービスが開始される可能性があるのでNTTに問い合わせをしてみるといいだろう。NTTのWebサイトでも利用地域の確認ができる。

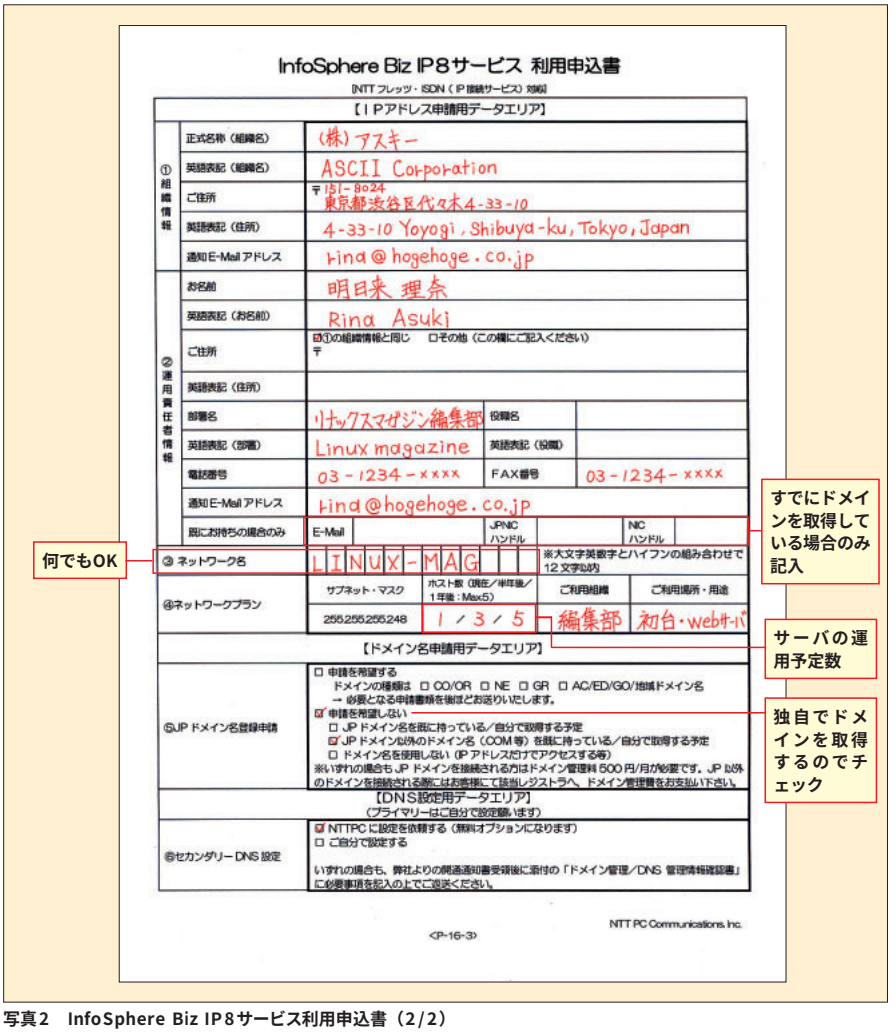


写真2　InfoSphere Biz IP8サービス利用申込書（2/2）

また、フレッツ・ISDNの申し込みページでは、電話番号を入力することで、フレッツ・ISDNが利用できるかどうかを知ることができる。あわせて利用してほしい。

申し込み自体は非常に簡単であり、Webページで申し込むことも、局番なしの116で申し込むこともできる。フレッツ・ISDNのサービスは、ISDN回線が前提となっているので、アナログ回線の場合は、同時にISDN回線への切り替えも必要となる。その際、テレホーダイなどの付加サービスを解約することもできる。

ただし、申し込みから工事日が決定するまで1カ月以上かかる場合もあるようだ。開通まで最短でも2週間はかかるということなので、残念ながらすぐに開通とはいかないようである。工事自体はNTT局内のみなので、ユーザー宅内の工事は必要ない。ただし、アナログ回線からISDN回線への変更を伴う場合は、宅内工事が必要になる場合もある。もっとも、よほど古い家でなければ「お客様工事」と呼ばれる、工事日の指定時間に自宅に居るだけでよいという工事ですますことができる。

次回は……

今回は、フレッツ・ISDNとプロバイダの申し込みだけで時間がかかってしまったが、次回はいよいよ独自ドメイン取得に向けて動き出す。次回は実際にインターネットに接続することになるので、マシンのご用意をお忘れなく。

InfoSphere Biz IP8サービス案内ページ	http://www.sphere.ne.jp/ip_service/service/
InfoSphere Biz IP8申し込み書ダウンロードページ	http://www.sphere.ne.jp/ip_service/join/
NTT東日本フレッツ・ISDNサイト	http://www.ntt-east.co.jp/flets/
NTT西日本フレッツ・ISDNサイト	http://www.ntt-west.co.jp/ipnet/ip/
NTT東日本フレッツ・ISDNサービス申し込みサイト	http://www.ntt-east.co.jp/teigaku/mskm.html
NTT西日本フレッツ・ISDNサービス申し込みサイト	http://www.ntt-west.co.jp/ipnet/ip/ipncheck.html

表2　URL一覧

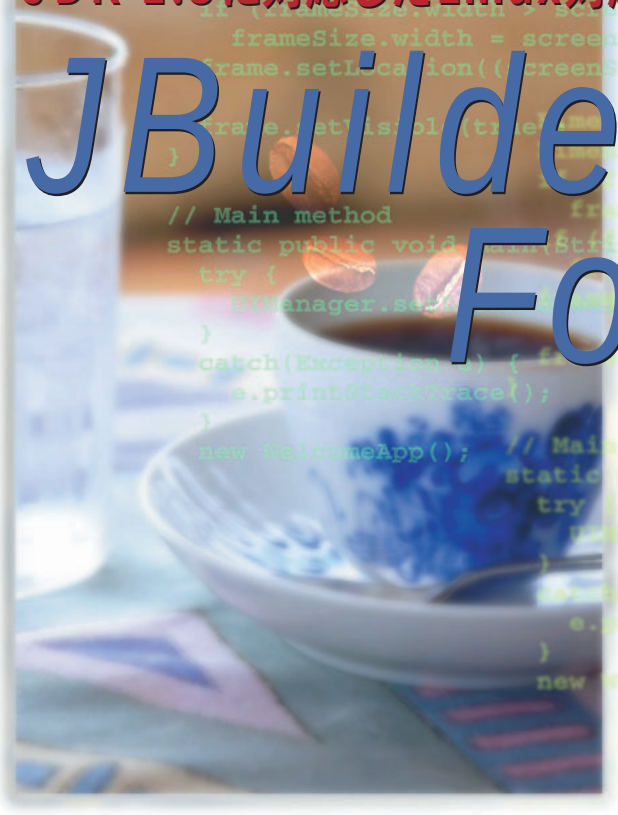
都道府県	サービス該当エリア
北海道	札幌市、北広島市、江別市、旭川市、函館市、釧路市、帯広市、北見市、岩見沢市、苫小牧市、小樽市、室蘭市、千歳市
青森県	青森市
岩手県	盛岡市
宮城県	仙台市、塩釜市、石巻市、多賀城市、名取市、古川市
秋田県	秋田市
山梨県	甲府市、富士吉田市
山形県	山形市
福島県	郡山市、福島市
茨城県	水戸市、つくば市、土浦市、牛久市、ヶ崎市
栃木県	宇都宮市、小山市、取手市、日立市、石岡市
群馬県	前橋市、高崎市、太田市
埼玉県	浦和市、大宮市、与野市、川口市、越谷市、久喜市、狭山市、坂戸市、志木市、新座市、和光市、富士見市、鶴ヶ島市、春日部市、鳩ヶ谷市、熊谷市、蕨市、上福岡市、八潮市、川越市、岩槻市、戸田市、所沢市、上尾市、草加市、朝霞市、入間市、桶川市、三郷市、行田市、鴻巣市、北本市、深谷市、秩父市、本庄市、加須市、飯能市、東松山市
千葉県	千葉市、浦安市、市川市、松戸市、船橋市、習志野市、柏市
東京都	東京23区、三鷹市、武蔵野市、調布市、狛江市、稲城市、小平市、国分寺市、小金井市、国立市、府中市、立川市、八王子市、日野市、多摩市、昭島市、武蔵村山市、保谷市、田無市、東久留米市、清瀬市、東村山市、東大和市、町田市
神奈川県	横浜市、川崎市、相模原市
新潟県	新潟市、長岡市、上越市
富山県	富山市、高岡市、砺波市、新湊市、小矢部市、氷見市、魚津市、黒部市、滑川市
石川県	金沢市、小松市、加賀市、松任市、羽咋市、七尾市、輪島市、珠洲市
福井県	福井市、武生市、鯖江市、敦賀市、大野市、小浜市、勝山市
長野県	長野市、松本市、上田市、諏訪市、塩尻市、飯田市
岐阜県	岐阜市、大垣市、各務原市、多治見市、可児市、関市、美濃加茂市、高山市、中津川市、羽島市、土岐市、瑞浪市、恵那市、美濃市
静岡県	浜松市、静岡市、清水市、富士市、沼津市、藤枝市、焼津市、島田市、富士宮市、三島市、磐田市、浜北市、御殿場市、掛川市、伊東市、袋井市、裾野市、熱海市、湖西市、下田市、天竜市
愛知県	名古屋市、豊橋市、豊田市、岡崎市、春日井市、一宮市、安城市、小牧市、瀬戸市、刈谷市、豊川市、半田市、稲沢市、東海市、江南市、蒲郡市、西尾市、知多市、大府市、犬山市、尾張旭市、碧南市、津島市、日進市、知立市、豊明市、尾西市、常滑市、岩倉市、新城市、高浜市
三重県	津市、四日市市、鈴鹿市、松阪市、桑名市、伊勢市、名張市、上野市、久居市、亀山市、鳥羽市、尾鷲市、熊野市
滋賀県	大津市、草津市、彦根市、近江八幡市、守山市、長浜市、八日市市
京都府	京都市、宇治市、亀岡市、舞鶴市、長岡京市、福知山市、向日市、城陽市、八幡市、京田辺市、綾部市、宮津市
大阪府	大阪市、堺市、東大阪市、枚方市、豊中市、高槻市、吹田市、八尾市、茨木市、寝屋川市、岸和田市、和泉市、守口市、門真市、松原市、大東市、富田林市、箕面市、河内長野市、羽曳野市、池田市、泉佐野市、貝塚市、摂津市、柏原市、交野市、泉大津市、藤井寺市、高石市、泉南市、阪南市、大阪狭山市、四條畷市
兵庫県	神戸市、姫路市、尼崎市、西宮市、明石市、加古川市、宝塚市、伊丹市、川西市※1、三田市、高砂市、芦屋市、三木市、赤穂市、加西市、小野市、豊岡市、洲本市、龍野市、相生市、西脇市、篠山市
奈良県	奈良市、生駒市、橿原市、大和郡山市、大和高田市、天理市、香芝市、桜井市、御所市、五條市
和歌山県	和歌山市、田辺市、橋本市、海南市、有田市、新宮市、御坊市
鳥取県	鳥取市、米子市※2、倉吉市、境港市
島根県	松江市、出雲市、益田市、浜田市、大田市、安来市、平田市、江津市
岡山県	岡山市、倉敷市、津山市※3、玉野市、笠岡市※4、総社市、井原市、新見市、高梁市、備前市
広島県	広島市、福山市、呉市、東広島市、尾道市、三原市、廿日市市、府中市、三次市、竹原市、大竹市、因島市、庄原市
山口県	山口市、下関市、宇部市、防府市、徳山市、下松市、光市、新南陽市、岩国市、柳井市、萩市、小野田市、長門市、美祢市
徳島県	徳島市、鳴門市、阿南市、小松島市
香川県	高松市、丸亀市、坂出市、観音寺市、善通寺市
愛媛県	松山市、新居浜市、今治市、宇和島市、西条市、大洲市、川之江市、伊予三島市、八幡浜市、東予市、伊予市、北条市
高知県	高知市、南国市、中村市、土佐市、須崎市、宿毛市、安芸市、室戸市、土佐清水市
福岡県	福岡市、北九州市、久留米市、大牟田市、春日市、筑紫野市、大野城市、飯塚市、宗像市、行橋市、大宰府市、直方市、前原市、田川市、古賀市、小郡市、中間市、筑後市、大川市、甘木市、柳川市、八女市、豊前市、山田市
佐賀県	佐賀市、唐津市、伊万里市、鳥栖市、武雄市、鹿島市、多久市
長崎県	長崎市、佐世保市、諫早市、大村市、島原市、福江市、平戸市、松浦市
熊本県	熊本市、八代市、荒尾市、玉名市、本渡市、人吉市、宇土市、山鹿市、水俣市、菊池市、牛深市
大分県	大分市、別府市、中津市、日田市、佐伯市、宇佐市、臼杵市、津久見市、杵築市、豊後高田市、竹田市
宮崎県	宮崎市、都城市、延岡市、日向市、日南市、小林市、西都市、えびの市、串間市
鹿児島県	鹿児島市、鹿屋市、川内市、国分市、名瀬市、出水市、指宿市、串木野市、枕崎市、阿久根市、加世田市、大口市、垂水市、西之表市
沖縄県	那覇市、名護市、浦添市、宜野湾市、沖縄市、具志川市、糸満市、石垣市、平良市、石川市

■受付開始中
■2001年1月25日サービス開始（受付開始1月11日）
■2001年1月下旬サービス開始
■2000年度中サービス開始
※1市外局番0727、 市内局番32、33、36、38、39地域を 除く
※2市外局番0859、市内局番22、23、26、27、31〜39
※3市外局番0868、市内局番22〜25、31、32
※4市外局番0865、市内局番62、63、65、68、69（69-0xxx〜69-2xxx、69-5xxx、69-8xxx）

短期集中連載

JDK 1.3に対応したLinux対応のJavaビジュアル開発ツール

JBuilder 4 Foundation



＜第1回＞JBuilder 4 Foundationとは

文：加藤大受
Text：Daiju Kato　daiju@ms.tokyo.jcom.ne.jp

今年4月に公開された無償のビジュアルJava開発ツールであるJBuilder Foundationが、最新のJDK 1.3に対応して新登場しました。そこで、今回から新しいJBuilder 4 Foundationについて徹底解説していきたいと思います。

JBuilder Foundationは、インプライズが開発・販売しているJava開発ツールであるJBuilderのエントリー版で、Webからダウンロードできる無償のバージョンです。ビジュアルなJava開発ツールというコンセプトで開発されている製品で、製品自体がすべてJavaで書かれています。このため、Linux、Solaris、Windowsの3つのプラットフォームで稼働するマルチプラットフォーム対応のJava開発ツールといえます。海外では、IBMのVisualAge for Java Entry Editionが、Linuxのディストリビューションパッケージにバンドルされているものもありますが、こちらの日本語版は用意されていません。

JBuilderは、これからLinux上でJavaを開発に利用しようとしている方や、すでにJDK（Java Development Kit）を利用してJavaを書かれている方は検討してみる価値があるでしょう。

JavaとLinuxの関係

LinuxとJavaの結びつきはまだそれほど深くないと思われる方も多いと思いますが、Web系のシステムを見てみた場合、多くの企業がすでにLinux上でJavaを使用し、動的なコンテンツを配信しています。

これらのシステムでは、サーブレットという技術が使われています。サーブレットが稼働するサーブレットエンジンは、Apache Software Foundationのプロジェクトのひとつである、Java Apache Projectによって開発されているApache JServや、jakarta ProjectのTomcatなどがあり、Linuxを形成するオープンソース文化は、Javaとの深いつながりがあります。

また昨年12月に、SunよりLinuxをサポートしたJDKが登場し、Linuxがオフィシャルプラットフォームのひとつになりました。これより、LinuxとJavaの結びつきがさらに強くなったとも言えるでしょう。現在、Linux版のJDKはblackdown、Sun、IBMの3カ所からリリースされており、ユーザーは自分の好みに合ったJDKを選択できるようになっています。

また、GNOMEにおけるJavaの実行環境であるJRE（Java Runtime Environment）の搭載、JDKの各ディストリビューションパッケージへのバンドルなどにより、今後、より多くのLinux上で使用できるJavaツールが増えていくことになるでしょう。

Javaは洗練された言語でもありますし、マルチプラットフォームという優れた特徴も持っています。そういった意味でも、Linux上でこれからプログラミングを始めようと思う方はJavaを選択することをお勧めします。

JBuilder 3.5 Foundationとの違い

今年4月にリリースされたJBuilder 3.5 Foundation（本誌2000年6月号に掲載）から、約半年で新しいバージョンであるJBuilder 4 Foundationがリリースされました。製品版であるProfessional Edition、Enterprise Editionに追加された機能については、インプライズのWebサイトでの情報を参照していただくことにして、Foundationで変更された機能について、まずは簡単にまとめてみましょう。

- **対応するJDKが1.2.2から1.3に変更され、IBMのJDKが搭載された**
- **JDK 1.3の搭載により、Javaアプリケーションのパフォーマンスを向上させるHotspot機能を搭載**
- **起動時にワンポイントの表示（画面1）**
- **ソースコードの履歴情報を表示するリビジョン・ブラウザ**
- **VisualStudio、Briefエディタ対応のキーマップ**
- **［ヘルプ］－［バージョン情報］で表示されるバージョン表示ダイアログボックスでJava環境の情報表示（画面2）**
- **そのほか統合開発環境（IDE）、ウィザードなどの改良**

今回のバージョンアップでは、Foundationの変更点は数少ないですが、最新のJDK 1.3が使用でき、ビジュアルな設計画面、プロジェクト単位でのソースコード管理、JPDA（Java Platform Debugger Architecture：Java 2対応のデバッグアーキテクチャ）対応のビジュアルデバッカなどが提供されています。無償で使用できるにもかかわらず、多機能で、非常に強力な開発ツールだといえるでしょう。

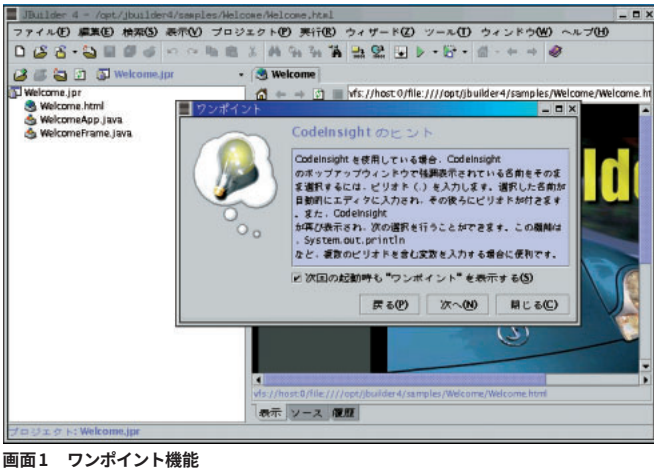
動作環境

JBuilder 4 FoundationのLinux版の動作環境は**表1**の通りです。JBuilder 3.5 Foundationのときはオフィシャルにサポートしているディストリビューションは LASER5 Linux 6.0と、Red Hat Linux 6.1日本語版の2種類だけでしたが、今回のJBuilder 4 Foundationからは、TurboLinux Workstation日本語版6.0とOpenLinux 2.3のサポートが追加されました。

JDK 1.3 SEのREADMEファイルによると、カーネル 2.2.12、glibc 2.1.2の環境でテストされています。これ以外のディスリビューションである、Debian GNU/Linux 2.2（potato）や、Vine Linux 2.1、Kondara/MNU Linux 1.2などでも動作するでしょう。

http://www.inprise.co.jp/jbuilder/foundation/

表3　ライセンスキーの入手先



画面1　ワンポイント機能

画面2　バージョン情報ダイアログボックス

CPU	Intel Pentium II 233MHz以上
メモリ	128Mバイト以上（256Mバイト以上推奨）
ハードディスク	150Mバイト以上の空き容量
モニタ	SVGA以上（256色以上）
OS	LASER5 Linux 6.2 Red Hat Linux 6.2 OpenLinux 2.3 TurboLinux 6.0

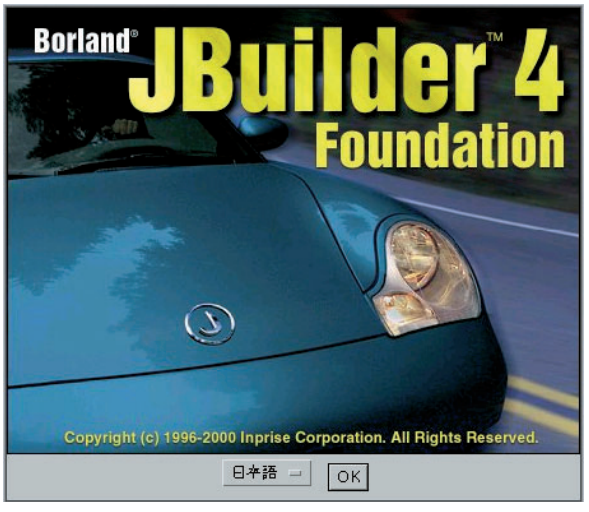
表1　JBuilder 4 Foundationの動作環境

IBM PC-350(6587-JBV)	
CPU	AMD K6-2 400MHz
メモリ	160Mバイト
ハードディスク	6Gバイト
OS	LASER5 Linux 6.2
DELL Dimension XPS T700r	
CPU	Intel Pentium III 750MHz
メモリ	256Mバイト
ハードディスク	30Gバイト
OS	TurboLinux Workstation日本語版6.0

表2　筆者の使用環境

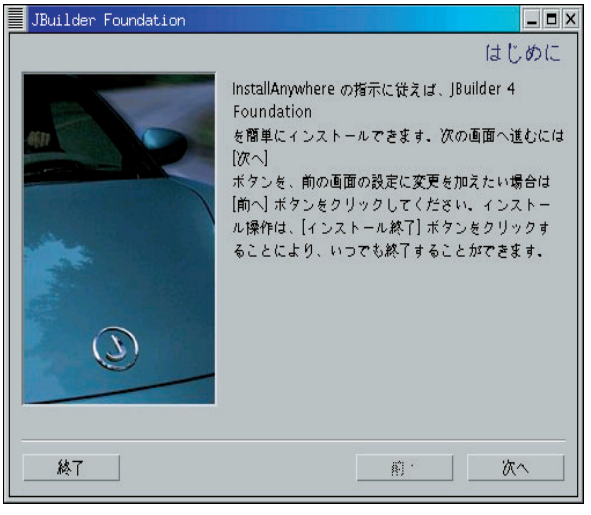
ライセンスキーの入手

JBuilder 4 Foundationを使用するためには、ライセンスキーが必要となります。このライセンスキーは、インプライズのWebサイトで登録を行うことで入手することができます（**表3**）。Webサイトに必要な情報を記入すると、記載したメールアドレスにインストールに必要なインストール番号とインストールキーが送付されます。



画面3　インストーラ画面（言語の選択）

画面4　インストーラの開始



画面5　GNOMEのデスクトップ上に作られたラウンチャ

```
# export PATH=/opt/jbuilder4/jdk1.3/bin:$PATH
# java -version
java version "1.3.0"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.0)
Classic VM (build 1.3.0, J2RE 1.3.0 IBM build cx130-20000815 (JIT enabled: jitc)
# ./doc_install.bin
# ./smp_install.bin
```

JDKにパスを通す

Java VMが起動するかどうかの確認

ドキュメントのインストールプログラムの起動

サンプルファイルのインストールプログラムの起動

画面6　ドキュメントおよびサンプルファイルのインストール

JBuilder 4 Foundationのインストール

JBuilder 3.5 Foundationでは、別途JDKをダウンロードし、自分でインストールしたあとにJBuilderをインストールする必要がありましたが、今回からJDKが同梱されているので、インストールが非常に簡単になりました。

JBuilder 4 Foundationのインストーラは次の3つに分かれています。

- **JBuilder 4 Foundation本体**
  fnd_linux_install.bin
- **サンプルファイル**
  smp_install.bin
- **ドキュメント**
  doc_install.bin

本誌の付録CD-ROMをドライブにセットし、root権限でマウントしてくだ

Column

ボーランド復活する

インプライズは、以前はボーランドという会社名で、Turbo Cなど、言語系ツールの開発元として知られていましたが、1998年に現インプライズに社名変更しました。

これにともない、ボーランドという名称は、ボーランドブランドへと変化しました。

しかし、ここにきて以前のボーランドへ社名を戻すことになりそうです（正確にはボーランド・ソフトウェア・コーポレーション）。以前からボーランドを知っているユーザーにとっては、こちらのほうがしっくりとくることでしょう。

さい。マウントはGNOME上であれば、CD-ROMのアイコンをダブルクリックするだけでマウントできます。ターミナルで行う場合は以下のコマンドを実行します。

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

CD-ROMをマウントしたら、ファイルブラウザでCD-ROM内のJBuilder4/linux_solディレクトリにある、fnd_linux_install.binをダブルクリックします。インストーラが起動すると、**画面3**のように表示されます。

JBuilderのインストーラはJavaで書かれており、インストールプログラムの中に含まれているJREを解凍し、自分自身を実行します。以前のバージョンでは、インストーラに実行権限が付いていませんでしたが、今回は実行権限が付いているのでファイルブラウザから簡単に実行することができるようになっています。

インストール手順は、**画面4**から始まるインストールの流れに従っていくだけです。

JBuilder 3.5 Foundationでは、/usr/local/jbuilder35にインストールされましたが、JBuilder 4 Foundationでは、/opt/jbuilder4（rootでインストールしたとき）、または/home/＜ユーザー名＞にインストールされることになります。もし、異なるディレクトリにインストールしたいときは、インストール時に変更することができます。

JBuilderがインストールされると、デスクトップ上にランチャが作成されます（**画面5**）。JBuilder 3.5 FoundationのときはKDEの場合のみ、自動的にランチャが作られましたが、今回はGNOMEでも自動的に作られるようになりました。

JBuilderのインストールが終了したら、続いてドキュメントおよび、サンプルプログラムのインストールを行います。ドキュメントとサンプルプログラムのインストールもJBuilderと同じように、ファイルブラウザからファイルを選択し、ダブルクリックすることでインストールすることができますが、これはJDKにパスが通っているときのみです。

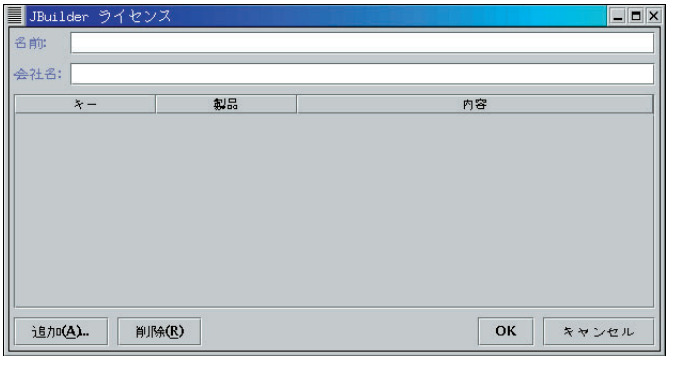
ドキュメントとサンプルファイルにはJREが含まれていませんので、基本的にターミナルからのインストールとなります。

ターミナルを起動し、exportコマンドを使用してJDKにパスを通します。JDKはJBuilder4ディレクトリ以下にインストールされています。パスを通したら正しくJava VMにパスが通っているかどうかを確認するといいでしょう。パスの確認が終了したら、ドキュメントのインストールプログラムおよび、サンプルのインストールプログラムを起動して、それぞれインストールを行います（**画面6**）。インストール先は必ず本体と同じディレクトリにしてください。

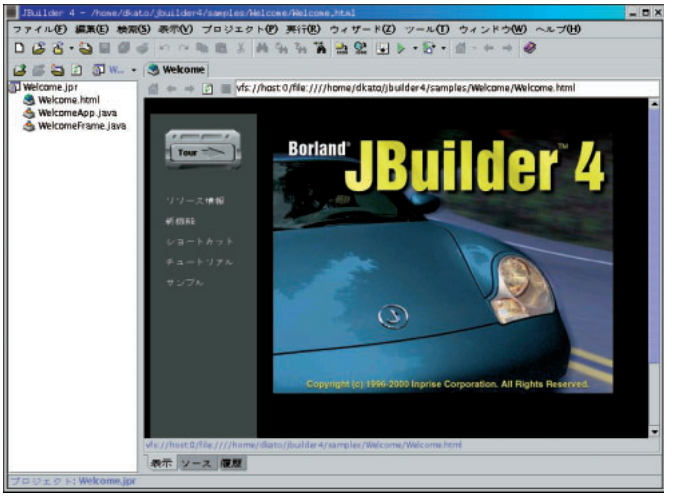
ライセンスキーの入力

デスクトップからJBuilderのアイコンをクリックすると、初回だけ**画面7**のようなライセンスキーの入力画面が表示されます。ここで、［追加］ボタンを押して、メールで受け取ったライセンスキーを入力します。すると、**画面8**のようにJBuilder 4 Foundationの開発環境が起動します。

最初に起動したときは、Welcomeプロジェクトが開かれています。Welcomeプロジェクトでは、JBuilderに関する



画面8　JBuilder 4 Foundationの開発環境



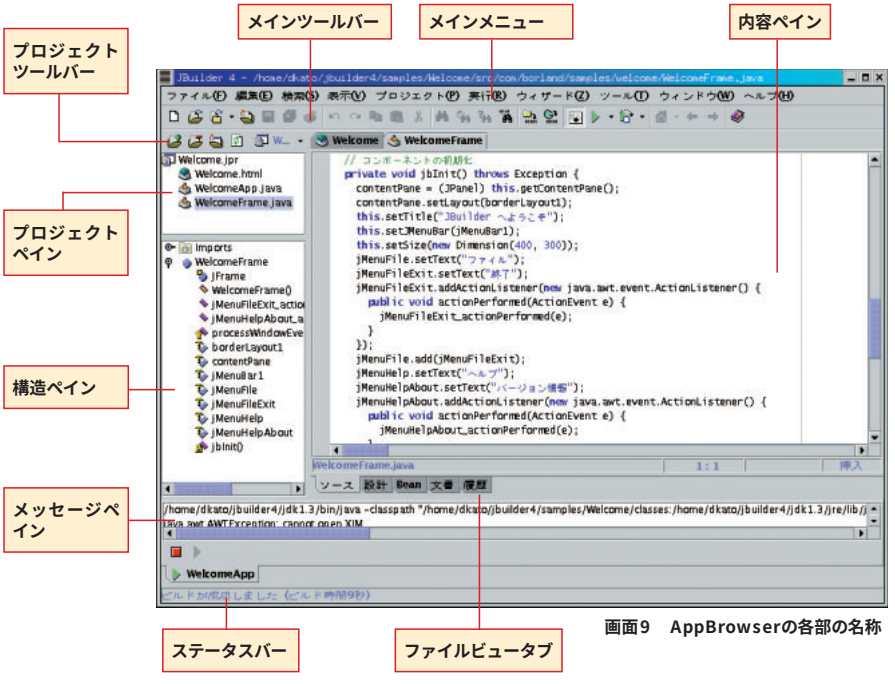
画面7　ライセンスキーの設定画面

基本的な情報を説明してくれるツアーを提供しています。初めてJBuilderを使用される方はこのツアーを利用して、JBuilderの提供する機能の概略を参照されることをお勧めします。

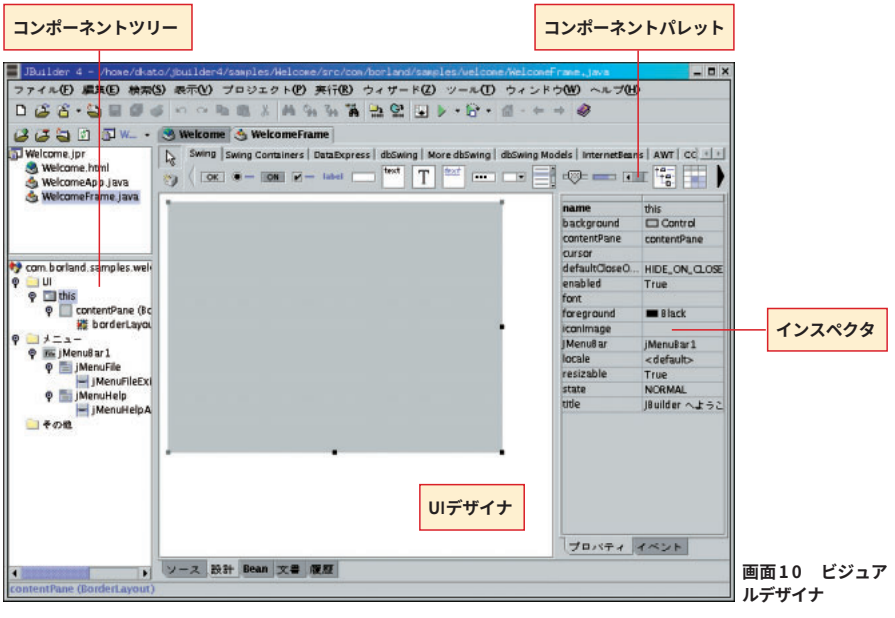
統合開発環境(IDE)について

JBuilder 4 Foundationの統合開発環境は非常に多彩な機能を持っており、プログラムの作成、設計、実行、デバックなどの一連の作業がすべてできるようになっています。JBuilderでは、この統合開発環境をAppBrowserと呼んでいます。**画面9**は、AppBrowerの各部の名称で、非常に多くの機能を1つのウィンドウで提供していることがわかると思います。

Javaファイルを表示しているとき、ファイルビューの［設計］タブをクリックすると、**画面10**のようなビジュアルデザイナが起動され、ユーザーインターフェイスを設計するデザイナ、コンポーネントの設定を行うインスペクタ、UIやさまざまな機能を提供するソフトウェア部品であるコンポーネントが並んでいるコンポーネントパレット、使用しているコンポーネントが一覧で表示されるコンポーネントツリーなどのウィンドウが表示されます。



画面9　AppBrowserの各部の名称



画面10　ビジュアルデザイナ

このビジュアルデザイナ画面を利用することで、Swingベース（JavaのGUIコンポーネント）のユーザーインターフェイスを簡単に作成することができます。

作成されたユーザーインターフェイスは、すべてJavaのプログラムコードとして生成され、ソースコードエディタ内で修正可能になります。ソースコードエディタとデザイナが一対一で対応していますので（この機能をインプライズでは2Way-tool機能と呼んでいます）、ソースコードからでも、ビジュアルな操作でも同じように使用することができるでしょう。

ソースコードエディタには、codeInsightと呼ばれるコード支援機能があります（**画面11**）。この機能は、選択されたオブジェクトが使用できるメソッドを表示してくれるコードの入力支援機能です。スペルミスなどを避けることができますので、プログラム初心者だけでなくても非常に便利な機能です。

このほか、入力している行にエラーがある場合に警告を出してくれる、ErrorInsightと呼ばれるバックグランドコンパイルを利用した機能（**画面12**）なども用意されており、コンパイル前にエラーを発見できるような機能が提供されています。

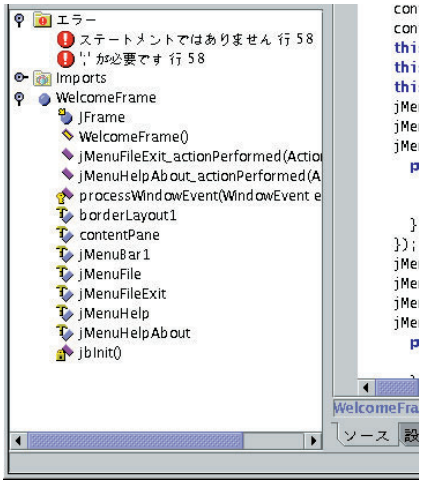
簡単なプログラムを作る

それでは実際にJBuilder 4 Foundationを使用して簡単なプログラムを作成しながら、JBuilder 4 Foundationの操作方法を説明していきます。GNOMEデスクトップ上のJBuilderのアイコンをダブルクリックしてJBuilder 4 Foundationを起動します。

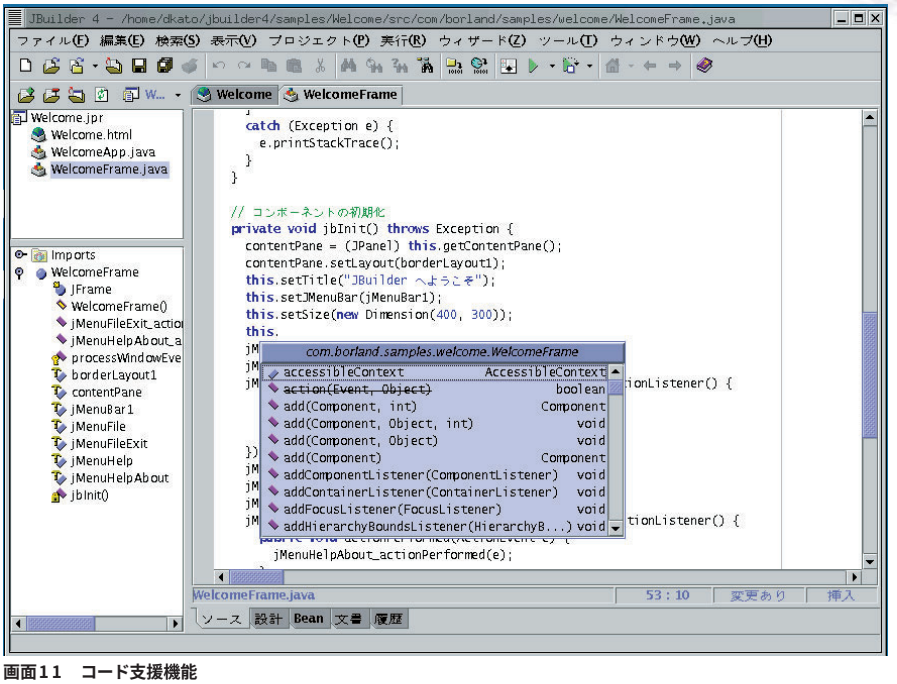
Welcomeプロジェクトが起動していますので、［ファイル］－［プロジェクトを閉じる］を選択して、Welcomeプロジェクトを閉じます。

続いて［ファイル］－［新規プロジェクト］を選択して、新規のプロジェクトを作成するためのプロジェクトウィザードを起動します（**画面13**）。プロジェクトとは、作成するJavaプログラムを管理するもので、クラスパスの設定、ソースファイルのディレクトリなどを管理します。プロジェクトウィザードに従うことで、パッケージ名、ソースの管理場所、プロジェクトメモなどを設定していくことができますが、ここでは［終了］をクリックして、プロジェクトウィザードを終了します。

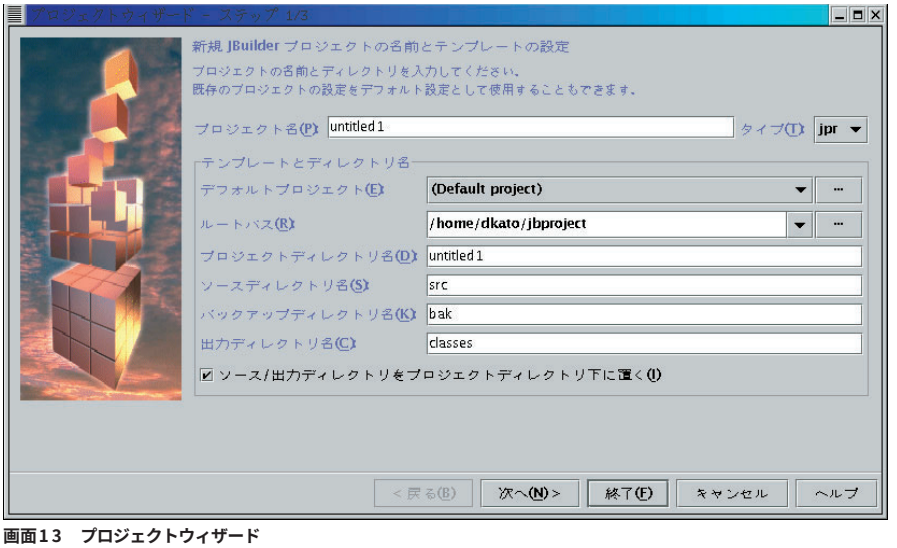
続いて、［ファイル］－［新規］を選択して、**画面14**のようなオブジェク



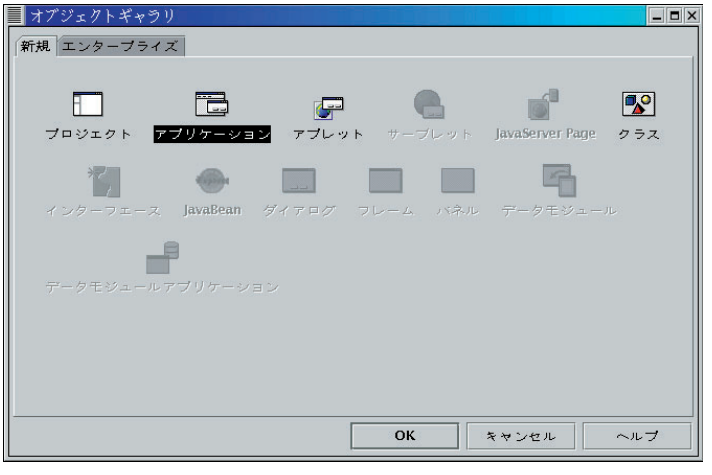
画面12　ErrorInsight



画面11　コード支援機能



画面13　プロジェクトウィザード

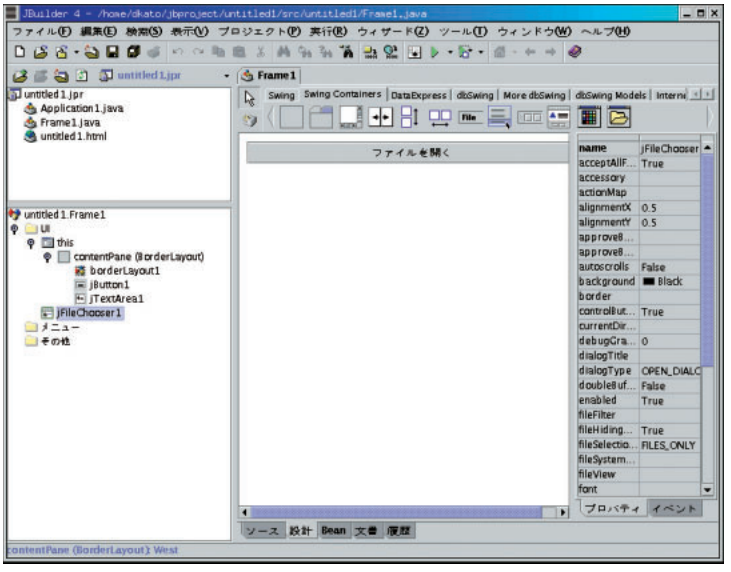


画面14　アプリケーションウィザード

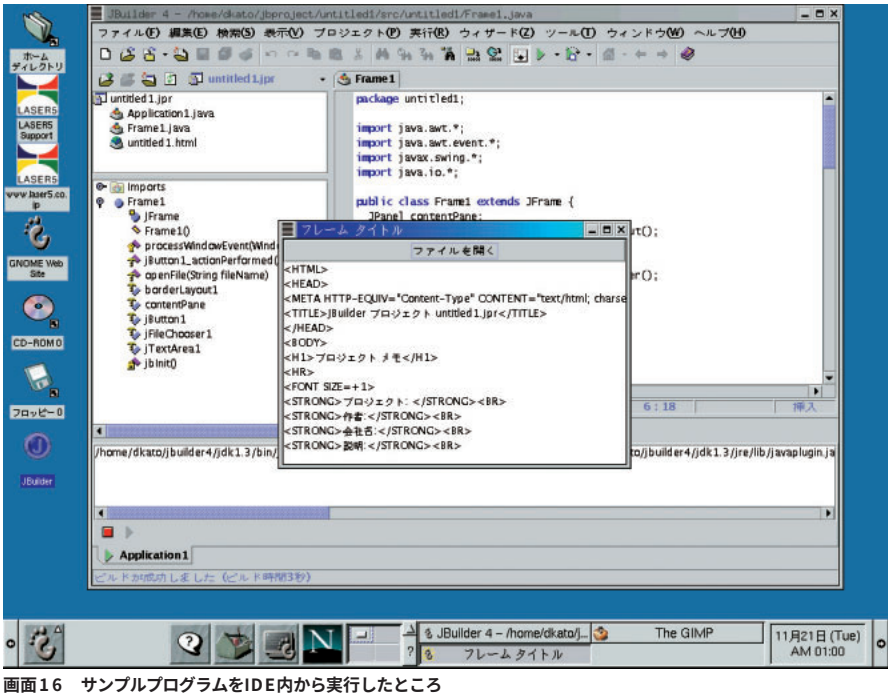
トギャラリを表示します。

オブジェクトギャラリは、作成できるオブジェクトの一覧を表示し、選択されたオブジェクトを作成するウィザードを呼び出します。ここでは［アプリケーション］をダブルクリックして、アプリケーションウィザードを起動します。アプリケーションウィザード、はJavaアプリケーションを作成するためのテンプレートを生成してくれるウィザードです。

このウィザードに従っていくことで、作成するアプリケーションクラス、フレームクラスの名前を設定したり、メニュー、ステータスバー、バージョン管理ダイアログのひな形を作成することができます。ここでは［終了］ボタンを押して、アプリケーションウィザードを終了します。これで非常に簡単なプロジェクト、アプリケーションクラス、フレームクラスを作成することができました。



画面15　サンプルプログラムのユーザーインタフェース



画面16　サンプルプログラムをIDE内から実行したところ

すでにソースエディタにはFrame1クラスが表示されています。［設計］タブをクリックしてビジュアルデザイナを起動し、ユーザーインターフェイスを設計してみましょう。

まず、コンポーネントパレットの［Swing］のページから、jButtonコンポーネントをクリックして選択し、デザイナ画面のパネル上で再度クリックします。これで、パネルの上にボタンが作成されました。このままボタンが選択されている状態で、インスペクタで［constraints］プロパティの値を「North」に変更し、［Text］プロパティの値を「ファイルを開く」に変更します。ボタンの位置が上部に変更され、ボタンのキャプションが設定された値に変更されます。修正された項目は、ただちにデザイナ画面に反映されます。

続いて、jTextAreaコンポーネントをパネルに貼り付け、［constraints］プロパティを「Center」に、Textプロパティを空白に変更します。上部にボタン、下部にテキストエリアが配置されているはずです。

続いて、［Swing Containers］のページから、jFileChooserコンポーネントを選択し、コンポーネントツリーの［UI］の位置にクリックします。ここにクリックすると、パネルに表示されませんので、非表示状態になります。設計された画面は**画面15**のようになります。

これでユーザーインターフェイスが完成しましたので、ボタンを押したときの処理であるイベント処理を作成しましょう。

再度ボタンを選択し、インスペクタの［イベント］ページをクリックし、一番上の［actionPerformed］の右の

空白部をダブルクリックします。これで、ビジュアルデザイナ画面からソースエディタ画面に切り替わり、イベントのコード入力を行うことができます。ここで、**リスト1**を参照してコードを追加してください。

リスト1では、まずボタンをクリックしたときに、ファイルを選択するダイアログを表示し、続いて選択したファイルを受け取ります。ここでは、選択したファイルを開くopenFileというメソッドを用意して処理しています。openFileメソッドでは、ファイルオブジェクトを作成し、選択したファイルを開いてバッファに格納し、最後にバッファの文字列をjTextArea1オブジェクトにセットして、選択したファイルをテキストエリアに表示しています。Java言語が初めての方でも、コメントを細かく書いておきましたので各行が何をやっているかを何となく理解できると思います。

リスト1を追加したら、Frame1クラスのソースの上部に**リスト2**のようにjava.ioパッケージを使用する行を追加します。このパッケージを追加しないとコンパイルが通りません。

パッケージを追加したら、［プロジェクト］－［プロジェクトのメイク］を選択して作成したプログラムをコンパイルします。コンパイルが終了したら、［実行］－［プロジェクトの実行］を選択して、作成したプログラムを起動してみましょう。

ボタンを押すと、ファイルを選択するダイアログボックスが表示され、ファイルを選択すると、**画面16**のようにテキストエリアにファイルの中身が表示されます。ここではプロジェクトファイルを表示しています。

このように、ほとんどコードを書かずにプログラムが作成できます。

次回に向けて

JBuilder 4 Foundationについて以前のバージョンとの違い、インストール方法、簡単な操作方法、そして簡単なサンプルプログラムの作成を説明しました。次回はもう少し高度なプログラムを題材としながらデバックの仕方などについて説明したいと思います。

JBuilder 4 Foundationは優れた開発環境を開発者に提供するツールです。プラットフォームの壁を超えるJavaという言語と、100% Pure Java開発ツールという組み合わせは、今後の開発環境を大きく変えることでしょう。

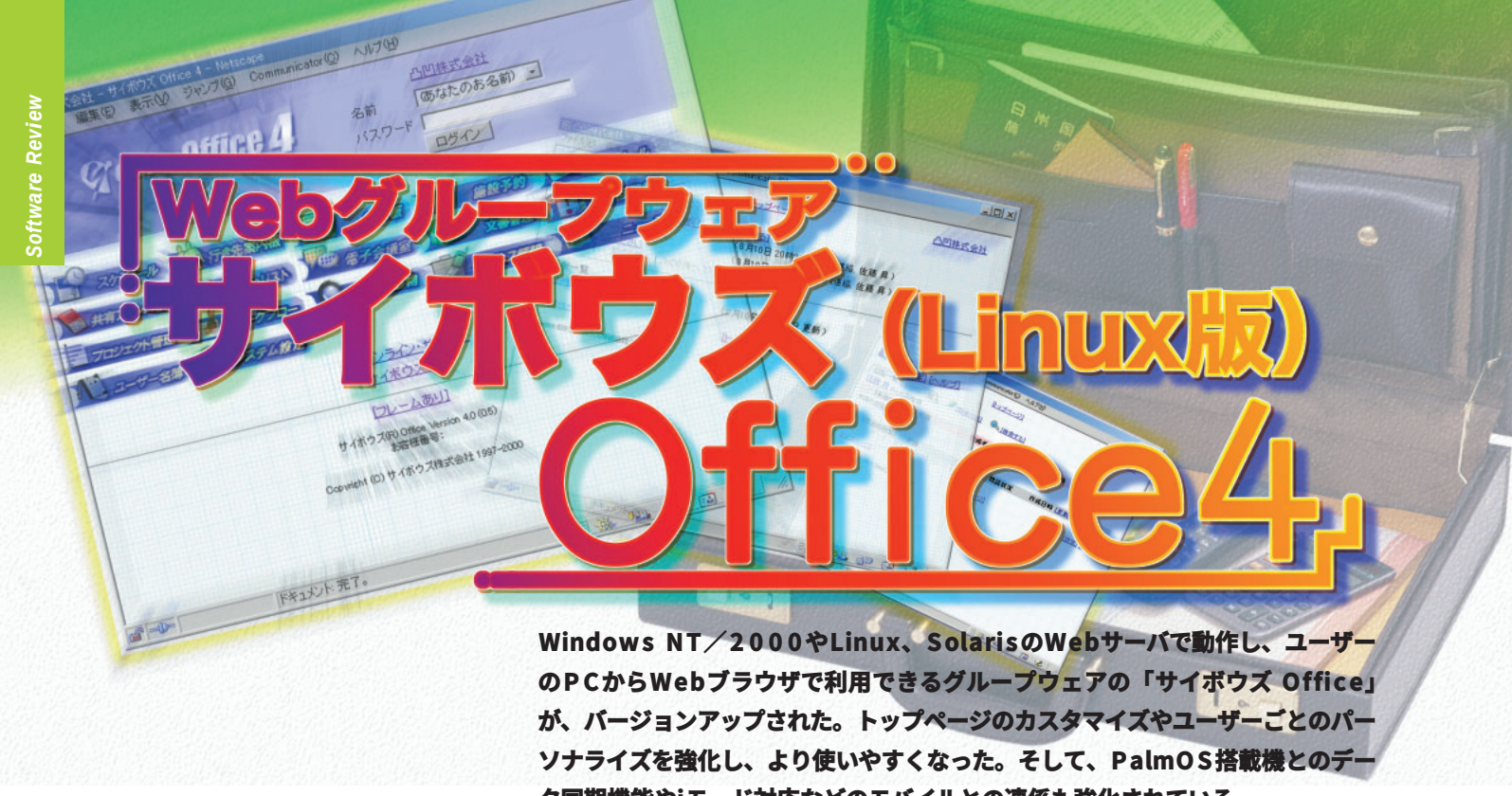
▼リスト1　サンプルプログラム

```
  void jButton1_actionPerformed(ActionEvent e) {
    //ファイル選択ダイアログの表示
    jFileChooser1.showOpenDialog(this);
    //選択したファイルを開く
    openFile(jFileChooser1.getSelectedFile().getPath());
  }

  //選択したファイルを開くメソッド
  void openFile(String fileName) {
    try
    {
      //ファイルオブジェクトの作成
      File file = new File(fileName);
      //オープンするファイルのサイズの取得
      int size = (int)file.length();
      int chars_read = 0;
      //ファイルリーダーオブジェクトの作成
      FileReader in = new FileReader(file);

      //読み込んだデータを格納する文字列配列の作成
      char[] data = new char[size];

      // すべての文字列をバッファに読み込む
      while(in.ready()) {
        chars_read += in.read(data, chars_read, size - chars_read);
      }
      in.close();
      //読み込んだデータをTextAreaに書き込む
      jTextArea1.setText(new String(data, 0, chars_read));
    }
    catch (IOException e)
    {
    }
  }
```

▼リスト2　java.ioパッケージの使用を宣言

```
package untitled1;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
```

追加



Webグループウェア サイボウズ(Linux版) Office4

Windows NT／2000やLinux、SolarisのWebサーバで動作し、ユーザーのPCからWebブラウザで利用できるグループウェアの「サイボウズ Office」が、バージョンアップされた。トップページのカスタマイズやユーザーごとのパーソナライズを強化し、より使いやすくなった。そして、PalmOS搭載機とのデータ同期機能やiモード対応などのモバイルとの連係も強化されている。

文：上間俊雄
Text : Toshio Uema

個人の生産性とグループ内の連携

オフィスでの情報の共有といえば、回覧版やホワイトボードを利用したものであった。しかし、出張などで長期にデスクを空けたままにすると、戻ってきたときには未読の資料が紙の山となって積み重なっているということがある。

その中にはすでに期限切れの情報になってしまったものや、未提出のまま締め切り日が過ぎてしまっていることも多々ある。これらが不定期に回ったり、書き忘れたりするとそこで情報の共有というのは崩壊してしまう。

また、風で飛んだり、シールがはがれてしまって電話メモがどこかに行ってしまい、お客さんに連絡が取れなくなれば信用を落としてしまうことにもなりかねない。

そこでオフィスでは、紙やホワイトボードに代わって、デジタルで情報交換を行うようになってきている。1人に1台のPCとインターネットへの常時接続の環境がオフィスに導入され、連絡には電子メールが使われる。

また、出張などで長期不在のユーザーのためにリモートアクセスサーバが構築されていたり、インターネット経由でアクセスできるようにもなってきた。ここでの情報の共有にはグループウェアが利用されるというように、オフィスでの情報共有のためのデジタル化は、ほぼ整備されてきている。

情報の共有に使われるグループウェアは、企業の部署内全員のスケジュール表や電話メモなどをネットワークを通じて共有することで、仕事を効率よくこなすためのツールである。オフィスの生産性を高めるためにグループウェアを導入する企業が増えたが、とりわけインターネットがすでに普及しているため、Webベースのグループウェアが使われるようになっているのだ。

Webベースのグループウェア

Webベースのグループウェアとは、その名のとおりWebサーバ上にグループウェアのアプリケーションがインストールされており、ユーザーはふだん使っているWebブラウザを利用してアクセスする。

Microsoft ExchangeやLotus Notesのようにグループウェア導入時にクライアントのソフトウェアを全PCにインストールしないといけないという面倒な作業もなく、サーバの構築も本を見て勉強してからという必要もない。サーバのインストールが終了すれば、インターネットのWebブラウジングと同様に、検索エンジンやWeb掲示板を利用したことのあるユーザーなら何も見ずに設定できてしまうのだ。

また、Webブラウザが使えれば、クライアントにもほとんど左右されない。Windowsはもちろんのこと、現在のLinuxのディストリビューションのほとんどが、GUI環境で利用できるようになったので、Linux＆Netscapeという組み合わせをクライアントとして使ったとしても、全然構わないのだ。

使い方もポータルサイトのようなインターフェイスなため、リンクをクリ

ックしたり、フォームを入力することで使えてしまう。

いくらWebグループウェアとはいえ、使わないユーザーがいるのではという心配もあるだろう。しかし、現在の電子メールの普及度を考えれば、電子メール環境が整っている企業でメールチェックをしない社員はまずいないはずだ。電子メールは、Webグループウェアに機能として統合されているので、WebブラウザでサーバにログインしたときにいやでもWebグループウェアを利用することになる。

サイボウズOffice 4

Webベースのグループウェアである「サイボウズ Office 4」が発売となった。昨年、サイボウズから発売された「サイボウズ Office 3」のバージョンアップ版で、LinuxのほかにもSolaris、FreeBSD、Windows 98／NT／2000に対応している。

サイボウズ Office 4は新しく追加された回覧板の機能や、これまでのスケジュール機能など、12種類のアプリケーションが利用できる。これらはすべてパーソナライズされたユーザーごとのトップページから利用する。どのような機能があるか見ていこう。

■進化したパーソナライズ機能

今回のバージョンアップは、ユーザーごとにオリジナルのトップページを作成できるパーソナライズ機能がより強化されている。その中でも、Myグループという機能が追加され、ユーザーごとに自分だけのグループ分けをできるようになった。

たとえば、担当しているプロジェクトのメンバーを自分だけのグループとして登録でき、スケジュール管理でこの機能を利用する場合、そのプロジェクトのメンバーのスケジュールを一覧で表示させることができる。

そのため、だれが出張に行く予定であるとか、休暇を取っているかなどをひと目で見ることができ、ミーティングの日時調整のためにメーリングリストなどで確認したり、ほかの部署のグループのスケジュールをチェックする必要もない（**画面2**）。

そこに施設予約のアプリケーションも組み込むことができ、会議室の予約状況と各ユーザーのスケジュールを確認しながら、ミーティングの日時を決めることもできる。

また、ページデザインを選ぶことができるようになっている。「熱帯魚」や「パステル（ピンク）」を選ぶと雰囲気が変わって見える（**画面3**）。

サイボウズOffice 4では、そのほかにも以下のように多くの機能が改良されている。

- **「スケジュール」**
  ダブルブッキングの警告、一度登録したスケジュールの再利用が可能
- **「Webメール」**
  フィルタリング機能、インターフェイスをツリー構造へと改善
- **「共有アドレス帳」**
  ユーザー名簿と連動
- **「ToDoリスト」**
  締め切り日をカレンダーからクリック1つで入力可能
- **「文書管理」**
  ファイルを削除しても復活が可能、ドキュメントのバージョン管理機能
- **「プロジェクト管理」**
  インターフェイスを変更し、大幅に機能改善
- **「ワークフロー」**
  アクセス権の設定、再申請が可能

また、サイボウズOffice 4は、独自に開発したオブジェクト指向データベ

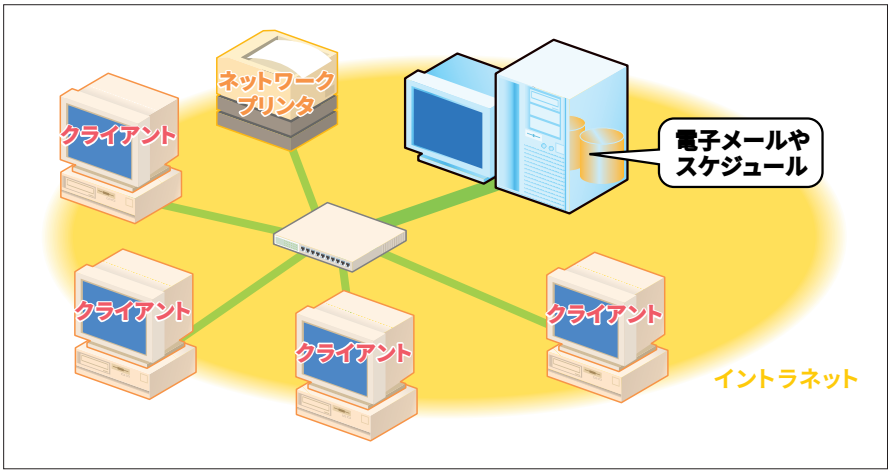


図1　イントラネット・インターネットでの情報共有は企業のIT化への第一歩だ

画面1　サイボウズ Office 4の起動画面

ースをバックエンドに標準搭載している。テキストファイル形式でデータを保管しているWebグループウエアに比べて、登録されているデータが増えてもレスポンスが低下することがなく、高速動作を維持することができる。

そして、データベースを利用することで、各機能間での連係がスムーズに行え、一覧表示での記事の並べ方を変更できたり、検索機能の充実が可能になっている。

モバイルとの連携強化

スケジュールなどを出先で確認したくなった場合、ノートPCでサイボウズサーバにアクセスするということもできるが、1kgもあるようなPCを毎日持ち歩くのは結構大変だ。さらに、外出の多い営業部員であれば、街中でノートPCを広げて、ことあるごとにスケジュールをチェックするのはなかなか勇気がいるものだ。

そこで、グループウェアとは別にスケジュール管理はPDAを使って行っているということがよくある。たとえば、最近人気の高いPalmは、スタイラスと呼ばれるペンを使って、アドレス帳や予定表を使えるので非常に便利だ。

ただし、この場合、問題になるのがデータの整合性である。Palm側でスケジュールを修正しても、グループウェアのサーバに登録し忘れたりすると、その時点でほかのユーザーが確認できるスケジュールとの整合性が取れなくなるということだ。サイボウズ Office 4では、「サイボウズ シンク4 for PalmOS」（**画面4**）というWindows用のソフトウェアを組み込むことで、サイボウズ Office 4とデータの同期を取ることができるようになった。

サイボウズOffice 4とPalm間でデータのリンクができるのは、スケジュール、共有アドレス帳、ToDoリスト、掲示板、Webメールで、たとえば外出先や電車の中で入力しておき、帰ってからそのデータをサイボウズOffice 4に転送することが簡単に行える。

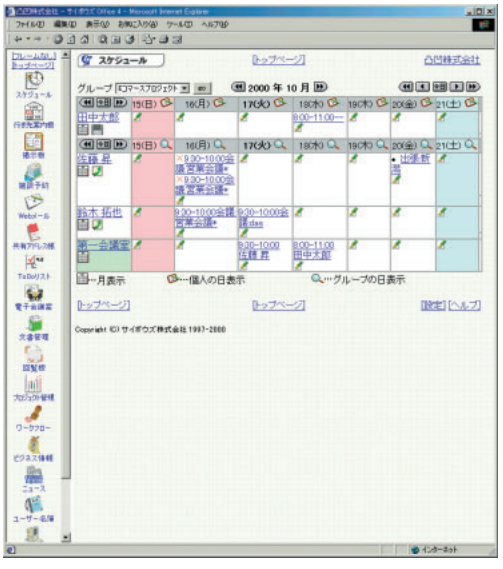
対応機種は、これまでのPalmシリーズやWorkPadのほか、HandspringのVisorや、SONYのCLIEにも対応している。

また、iモードでのアクセスに対応する「サイボウズ ケータイ4 for iモード」（**画面5**）を使用すると、外出先から直接サイボウズOffice 4のデータにアクセスできる。なお、現在はベータ版を公開中で、2001年第1四半期に発売を予定している。

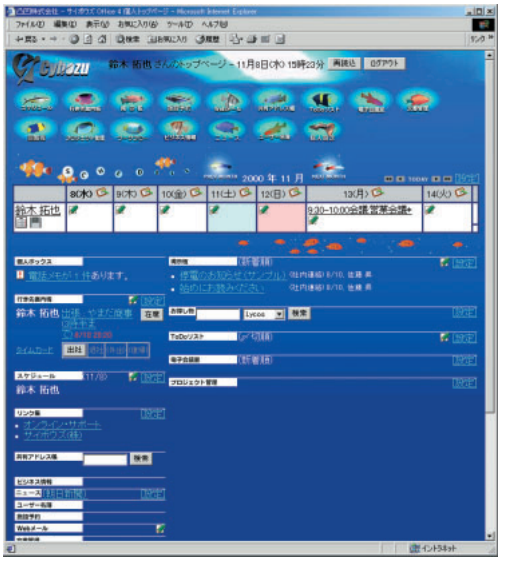
インストール

ここでは実際に「サイボウズ Office 4 Linux版」を、TurboLinux Workstation 日本語版 6.0にインストールしてみることにする。なお、LinuxマシンのIPアドレスは、固定で割り振られているものを利用しよう。実験的な運用では毎回IPアドレスを確認しながら利用すればよいが、本運用となれば、IPアドレスが毎回変わっていてはユーザーが混乱してしまうからだ。

次に、HTTPサーバソフトのApacheがインストール・起動されているかどうかも確認しておこう。「サイボウズ Office 4 Linux版」は、HTTPサーバのApache上で動作するCGIのため、単体では起動しない。Apacheについては、ほとんどのLinuxのディストリビューションで標準のHTTPサーバとして採用されているので、パッケージから組み込めばよい。TurboLinuxやRed Hat系Linuxでの確認方法は、コンソールからはrpmコマンドで確認できる。



画面2　便利なスケジュール管理
自分だけのオリジナルグループを作成して、部署以外のプロジェクトのメンバーのスケジュール状況もひと目で把握できる。



画面3　個人トップページを熱帯魚にデザイン変更
ユーザーごとにトップページを、標準、Office3準拠、ビジネス、熱帯魚、パステル（ピンク）、シック（ダークブルー）から選ぶことができる。

（rootでログインして）

```
# rpm -q apache
apache-1.3.12-1
```

上記のようにRPMパッケージが表示されればインストールされている。

「サイボウズ Office 4 Linux版」は、Apacheが動作していなければインストールができないようになっている。psコマンドでhttpd（Apacheプログラム）が動いているかどうか確認するか、実際にWebブラウザからアクセスしてみるのもよいだろう。

```
# ps ax | grep httpd
```

として、httpdの表示が「grep httpd」行だけならばApacheが動作していない。その場合には、Red Hat系Linuxなら下記コマンドで、httpdを起動させておく。

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

さらに、Linux起動時に自動的にhttpdを実行するように、ntsysv（Red Hat系）やturboservice（TurboLinux）コマンドで設定しておくとよい。

Apacheがインストール・動作していることを確認したら、Apacheのコンテンツを格納するドキュメントルートディレクトリと、CGIが実行されるディレクトリを確認しておこう。

TurboLinuxやRed Hat系Linuxで、ApacheをRPMパッケージからインストールした場合には/etc/httpd/confディレクトリに、それ以外の場合には/usr/local/apache/confディレクトリに、httpd.conf、access.conf、srm.confといった設定ファイルがあるはずだ。その3つのファイルの中にディレクトリ名が書かれている。ドキュメントルートディレクトリは「DocumentRoot」、CGIのディレクトリは「ScriptAlias /cgi-bin/」といった行の前後を確認しておこう。

TurboLinuxやRed Hat系の場合、ドキュメントルートディレクトリは/home/httpd/html/、CGIディレクトリは/home/httpd/cgi-bin/となっている。サイボウズOffice 4のインストール時には、これらの標準的なディレクトリがデフォルトになっているので、そのまま［Enter］キーを押していくだけだ。

ソースからApacheをインストールした場合や、Red Hat系でない他のLinuxディストリビューションでは、ディレクトリ構成が異なっていることがあるので、メモしておき、サイボウズOffice 4のインストール時にそのディレクトリ名を入力する。

「サイボウズ Office 4 Linux版」は、「http://cybozu.co.jp/」よりダウンロードして入手できる。ファイルは1Mバイト程度だ。本誌の付録CD-ROMにも収録している。ファイルは、tar+gzip形式で1つのファイルに固められているので、まずこの展開から始める。/tmpなど適当なディレクトリにおいて、tarコマンドで解凍する。

```
# tar zxvf cbof40jal.tar.gz
```

解凍されたインストーラ「cb4setup」はシェルスクリプトとなっている。インストーラは対話式で行い、日本語でも表示されるので、コンソールのkonコマンドやX Window Systemの漢字ターミナル（ktermなど）、Windowsに付属のtelnetでEUCの文字コードで実行する。なお、cb4INSTALL.txtというインストール説明ファイルが同じところに解凍されているので、これを参考にしながらインストールすればよい。インストーラはroot権限で実行する。

```
# ./cb4setup
```

まず、日本語で表示しているかどうか確認してくるので、表示されている場合には［Y］キーを押して、［Enter］キーで進む。次に、「ソフトウェアの使用許諾」をよく読んで、［Enter］キーを押して進もう。ドキュメントルートディレクトリとCGIのディレクトリの確認をしてくるので、Apacheの設定

画面4　サイボウズ シンク4 for PalmOS
スケジュール、ToDoリスト、共有アドレス帳、掲示板、Webメールのデータを、Palmに連係することができる。

画面5　サイボウズ ケータイ 4 for iモード
iモード搭載携帯電話から、サイボウズOffice 4のデータを閲覧、書き込むことができる。

と合っていればそのまま［Y］キーを押して、［Enter］キーを押す。

「製品の選択」では、パック製品か単体での導入ができるので、実際に試したい製品番号を入力する。「プロジェクト管理」と「ワークフロー」は単体での販売となっているので、導入する場合には個別に指定しよう。最後にインストールの確認をしたら、ファイルがコピーされインストールは終了する。

起動と初期設定

インストールが終了すれば、あとはWebブラウザからサイボウズのサーバにアクセスできる。Webブラウザから実際に動作を確認してみよう。“（IPアドレス）”の部分は、DNSなどで名前が解決されていればサーバ名でもよい。

```
http://（IPアドレス）/cgi-bin/cb4/office.cgi?
```

そうすると**画面1**のようなメニューが表示されるはずだ。最初にやることは、Webグループウェアで運営者の設定だ。［システム設定］のボタンをクリックして、「管理・運用」の「設定用パスワードの登録」で管理者のパスワードを設定する（**画面6**、**7**）。そして、「サイボウズ Office 共通設定メニュー」を設定していこう。

まず、「会社情報の変更」で会社情報をフォームに入力していく。入力が終了したら、［変更する］ボタンをクリックして保存しよう。さらに、「グループの設定」で部署ごとにグループの設定をし、あとは「ユーザーの設定」でユーザーを追加していけばよい（**画面8**、**画面9**）。

ほかにも、「ログイン前のトップページの設定」を変更することで、サイボウズOffice 4に最初に接続したときの画面をカスタマイズすることができる。

ユーザーログイン後の設定

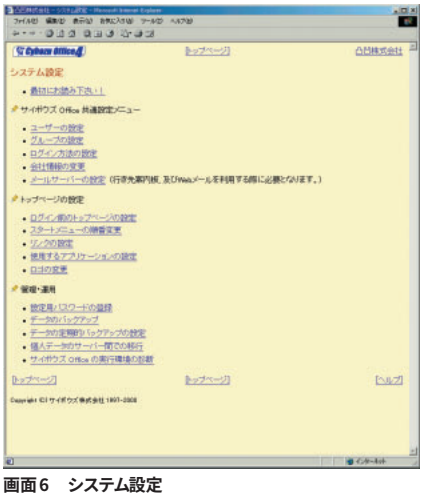
トップページに戻ると、先ほど登録したユーザーでログインすることができるようになっているはずだ。ログインしたら、ユーザーのトップページが表示される。もちろん、これですぐに使うこともできるが、上のメニューから「個人設定」を選択して、自分用に設定を変更していくことにしよう（**画面10**）。

最初はパスワードを設定する。あとは、ユーザー情報を入れ直したり、新機能の「Myグループの設定」や、レイアウトやデザインも変更できる。

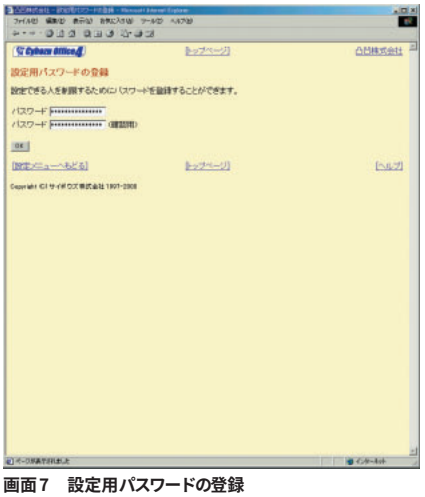
個人の設定が終われば、次に、行き先案内板の［在席］ボタンを押して、自分が現在、デスクにいることを他のユーザーに通知することにしよう（**画面11**）。

Myグループの設定

新機能のMyグループは自分がかかわっているプロジェクトのメンバーを登録できる。これで、グループが違っ

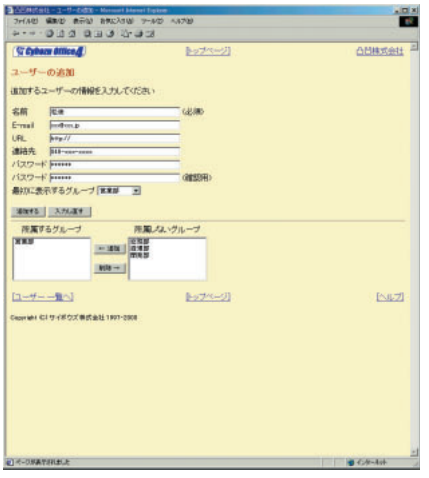


画面6　システム設定
インストールが終わったら、あとはWebブラウザから設定できる。まず、システム設定から始めよう。



画面7　設定用パスワードの登録
管理者の第一歩はパスワードの設定だ。同じパスワードを2回入力して、あとは［OK］ボタンを押すだけで、今日から管理者だ。

画面8　ユーザーの設定
ユーザーを追加するには、あらかじめExcelなどで作成し、CSV形式で保存しておいたファイルからコンバートすることもできる。



画面9　ユーザーの追加
通常のユーザー追加作業は、この画面で各ユーザーの情報を入力することで行う。

ていても、自分のプロジェクトのメンバーのスケジュールを確認したり、すぐに打ち合わせをしたくなった場合でも、行き先掲示板の在席状況で確認することができる。

設定は、ユーザーのトップページの「個人設定」から「Myグループ」のリンクをクリックして行う（**画面12**）。ここで［このMyグループを「最初に表示するグループ」に設定する］にチェックしておくことで、以後、スケジュールや行き先案内板に表示されるメインのグループになる。

サイボウズの購入方法

「サイボウズ Office 4」は、ライセンス販売を採用している。まず、サイボウズOffice 4はインストールしてから60日間は、すべての機能を試すことができる。この間にライセンスを購入すれば、そのまま継続して利用できる。もちろん、試用期間に登録したデータもそのまま移行される。60日を超えるとライセンスキーを登録するまではすべての機能が利用できなくなる。なお、ライセンス販売なので、ショップでパッケージを買うことはできない。

ソフトウェア本体はダウンロードするか、雑誌付録のCD-ROMなどから入手し、マニュアルもインターネット上のオンラインヘルプを参照する。ライセンスはどのプラットホームで利用してもいいことになっており、たとえばWindows NT／2000で運用することをやめて、Linux版に乗り換えても費用は発生しない。

サイボウズOffce 4は、12種類の機能をそれぞれ単体の製品として購入することができ、機能とユーザー数（10ユーザー、50ユーザー、100ユーザー、無制限）に応じて価格は異なる。各機能は50ユーザーで、2万9800円～9万9800円となっている。

また、機能をまとめたパッケージが3種類あり、「サイボウズOfficeパックEX 4」には、「スケジュール4」「行き先案内板4」「掲示板4」「施設予約4」「Webメール4」「共有アドレス帳4」「ToDoリスト4」「回覧板4」「電子会議室4」「文書管理4」が含まれていて、50ユーザーが19万8000円、100ユーザーが38万円、無制限が88万円となっている。「ワークフロー4」と「プロジェクト管理4」は、別途単体で購入する必要がある。

SOHO向けの「サイボウズOffice SOHO 4」は、EX 4と同機能で10ユーザーパックが7万9800円で、将来、10ユーザーを超えるようになったときには、EX 4へ差額のみでバージョンアップが可能だ。また、「サイボウズOfficeパック 4」は、「スケジュール4」「行き先案内板4」「掲示板4」「施設予約4」「Web メール 4」が含まれていて、50ユーザーが9万9800円、100ユーザーが19万8000円、無制限が48万円となっている。

なお、以前のバージョンからのバージョンアップや、ユーザー数の追加など細かく価格設定されているので、詳細はサイボウズのWebページ（http://cybozu.co.jp/）を参照していただきたい。

製品名	**サイボウズOffice 4**
価格	7万9800円（10ユーザー）～88万円（無制限）
問い合わせ先	サイボウズ株式会社 E-Mail　info@cybozu.co.jp Tel 06-4796-8668 http://cybozu.co.jp/cb4/

画面10　個人設定メニュー
ユーザーごとの設定を行う。新機能のMyグループの設定や、レイアウトの変更もここで設定できる。

画面11　ログインして表示される画面
ログインしたら、まずは画面の左側にある行き先案内板の［在席］ボタンを押すことから始めよう。

画面12 myグループの追加
Myグループでは、現在の仕事内容に応じてグループを追加したり、自分標準のグループとして登録できる。

スケジュール

各ユーザーのスケジュールを確認したり、共有したりできるスケジュール帳だ。グループ全員のスケジュールがひと目で確認できるので、ミーティングのスケジューリングに役立つ。また、新機能のMyグループ機能で、現在進めているプロジェクトのメンバーのスケジュールを一覧で見たいといったような個人的な事情にも対応できる。施設予約と連動して会議室も同時に予約可能だ。

行き先案内板

行き先案内板は、グループのメンバーの現在の行き先が確認できる機能だ。スケジュール4では大まかなその日の予定のみ書かれていて、現在どこにいるのかまでは書かれていないときに役立つ。会社の部署ごとによくあるホワイトボードのような役割をしてくれる。もし、打ち合わせに出ているようなら、メモを残しておくことができる。またメモをメールで送信することも可能だ。

掲示板

たとえば、辞令や停電のお知らせなど、全員に連絡しておきたい情報を掲示しておく。実際の掲示板のように、スペースが限られているわけではないので、バックナンバーを残しておくことができ、1階の掲示板、2階の掲示板というように複数の掲示板を作成できる。また、記事が検索できるので、古い情報や、似たような情報でも見つけやすい。

施設予約

会議室や打ち合わせスペースの予約ができる機能だ。会社全体の施設を登録しておけば、どこがいつ空いているかなどを、PCからチェックすることができ、取引先と電話しながら、その場で予約することができる。また、解除するのも自分の席から簡単にできる。使われていない場所の把握などもできるので、打ち合わせスペースの穴場や無駄なスペースを見つけるといったこともできる。

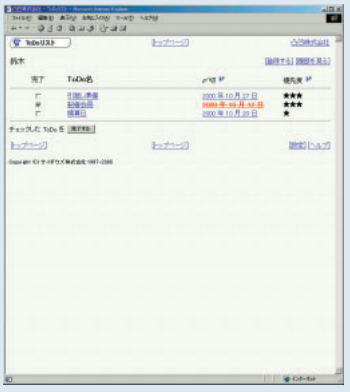
Webメール

グループウェアの基本ともいえる電子メールもWebブラウザから利用できる。POP3やSMTPサーバの機能ではなく、あくまでも電子メールのクライアントソフトといった位置付けなため、メールサーバを別途用意する必要がある。メール送信時には添付ファイルを送ることも可能だ。ただし、メール受信時にAPOPやIMAPといったプロトコルには対応していない。

共有アドレス帳

ふつうのアドレス帳と同じだが、グループで共有すると、ほかの人がすでに登録してくれていたりすれば登録の手間も省ける。名前順に並べられているが、会社名ごとにソートしたり、個人用の秘密のアドレス帳も作ることができる。また、顧客を管理する場合に、キーワードで検索することもでき、メールアドレスをクリックするだけでWeb メールの送信先にアドレスを挿入してくれる。

ToDoリスト

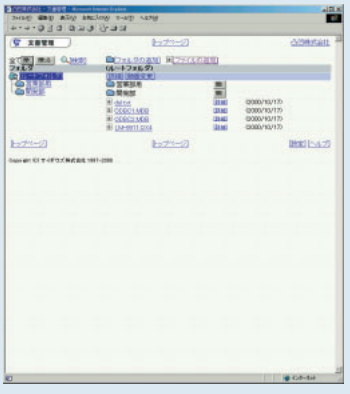


現在進行している自分の仕事の締め切りを設定したり、仕事の優先順位をつけることができる機能がTo Doリストだ。いくつも同時に平行して仕事をしている場合、頭の中での優先度をつけていても、うっかり忘れてしまうような案件も、ここで設定しておくことで、ミスを減らすことができる。また、入力時に設定する日付はカレンダーから日付をクリックするだけで登録できる。

電子会議室

電子会議室は出された議題に対して各ユーザーから意見を聞いたり、そのユーザー同士で意見のやり取りをする場所だ。ユーザーがそれぞれ時間の空いたときに発言できるので、ミーティング前にある程度のコンセンサスを取っておくこともできる。また、Excelで作成した見積書や、PowerPointで作成した企画書を添付して、それに対しての意見をもらうといったこともできる。

文書管理



社内で共有する文書は、ユーザーのローカルディスクにあると各自が個別に変更を行うことになり、すぐ独自のバージョンに変化してしまい社内全体の統一がとれなくなってしまう。文書管理は、書類などのファイルをサーバ上で管理するので、常に最新のデータを共有することができる。さらにファイルを間違えて消してしまっても復旧できるようになった。

回覧板

回覧板は今回のバージョンから新しく追加された機能だ。紙ベースの回覧板では、全員が読み終えるまでに日にちがかかったり、途中で書類の山に埋もれてしまったりするが、デジタル化することで一斉に読むことができるし、行方不明になることもない。もちろん誰が読んだかの確認もできる。回覧を検索したり、回覧書類にファイルを添付することや、コメントを書くことも可能だ。

プロジェクト管理

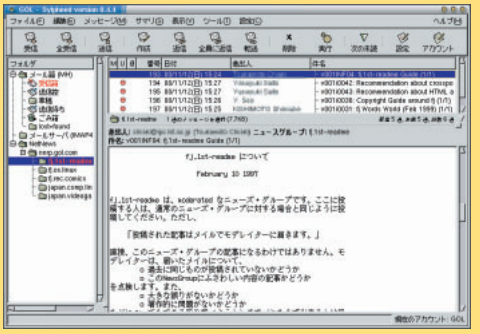
プロジェクト管理は仕事の進捗状況をユーザーごとに活動テーマと達成度を入力させることで、プロジェクト全体の状況を把握できるようにした機能だ。ここで達成度の悪いところがあった場合には、人員追加で対処するといったこともできる。また、進行状況の報告会議を開いたりする時間も省ける。なお、今回のバージョンから回覧板の機能と入れ替えでこの機能が別売りになった。

ワークフロー

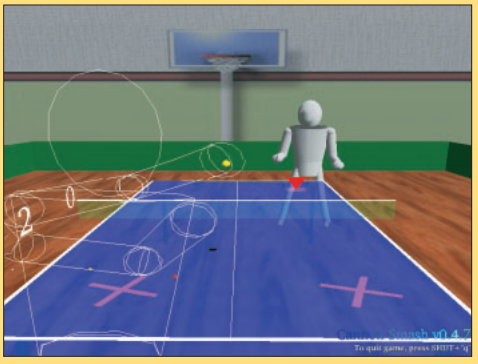
紙ベースで稟議や承認をもらう場合、金額に関係なく、忙しくなれば数週間もほったらかしになることがある。また、早急に承諾を取りたい場合には、催促のためにどこで止まっているのか問い合わせてもらうといったことも起きてくる。「ワークフロー」はこれらをデジタル化することで、たとえ出張中の上司に承諾を取りたい場合にも見てもらうことができるのだ。

Free Application Showcase

文：出井 一
Text : Hajime Dei



Sylpheed P.124



Cannon Smash P.130

TuxTyping P.133

※紹介したソフトは、すべて付録CD-ROMに収録されています。

EDICT形式に対応した辞書ソフト

Gjiten

バージョン：0.7　　ライセンス：GPL

http://gjiten.sourceforge.net/index.ja.php
http://www.gtk.org/~otaylor/kanjipad/（kanjipad）

Gjitenは、EDICT形式の和英辞書や漢字辞典などを利用するグラフィカルな辞書ソフトだ。指定した単語の意味を複数の辞書からまとめて引いたり、部首や画数から漢字を検索できる。さらに、マウスで描いた漢字を認識するkanjipadを併用することで、読みのわからない漢字の読みや意味を辞書で調べることも可能だ。地名やコンピュータ用語の辞書も別配布されている。動作にはGTK＋が必要だ。

●ビルドとインストール

Gjitenは、tarボールとRPMバイナリパッケージが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どちらか選択しよう。

なお、配布されたソースのままでは、起動時に環境変数LANGUAGEを「ja」に設定しないと、付属の日本語カタログが利用されない。この問題を修正するには、ビルドの前にsrc/gjiten.cの1376行目「setenv（"LANGUAGE","C",0）;」という1行をを削除する必要がある。

tarボールからのビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。また、tarボールに含まれるspecファイルを利用して「rpm -ta gjiten-0.7.tar.gz」とすれば、自分でRPMパッケージを作成できる。

このほか、Gjitenが利用する手書き漢字認識ソフトkanjipadもインストールしておこう。tarボールとRPMパッケージが配布されている。tarボールのビルドとインストールは、「make」→「make install」でOKだ。

●辞書ファイルのインストール

今回は、J. Breen氏作の和英辞書（EDICT）と漢字辞典（KANJIDIC）を収録している。RPMパッケージの場合は「/usr/share/gjiten」、tarボールの場合は「/usr/local/share/gjiten」にこれらをコピーし、「gunzip edict.gz kanjidic.gz」として展開しよう。

続いて、Gjiten付属のgenxjdxを利用して、検索時に使われるインデックスを作成する。辞書の展開先ディレクトリで「genxjdx 辞書ファイル名」とすればいい。

なお、これらの辞書は現在はモナシュ大学EDRDグループが管理しており、「モナシュ大学日本語アーカイブ」（国内ミラーはftp://ftp.u-aizu.ac.jp/pub/SciEng/nihongo/ftp.cc.monash.edu.au/00INDEX.html）で配布されている。このほか、日独辞書（JDDICT）、コンピュータ用語辞書（COMPDIC）、地名辞書（J_PLACES）なども配布されており、Gjitenで利用可能だ。

●起動と初期設定

「gjiten&」とするか、GNOMEメニューから[プログラム]－[アプリケーション]－[Gjiten]を選択すると、Gjitenのウィンドウが開く。ソースを修正していない場合は、コマンドラインから「LANGUAGE=ja gjiten&」として起動しよう（**画面1**）。

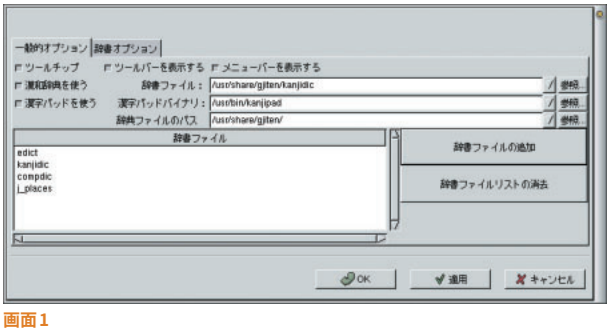
最初に起動した場合は、自動的に設定ダイアログが開くので（**画面2**）、利用する辞書の登録を行う。[辞書ファイルの追加]ボタンを押して、辞書ファイル（edit、kanjidic）を登録すればいい。その他の辞書を追加する場合も同様だ。

●単語を検索する

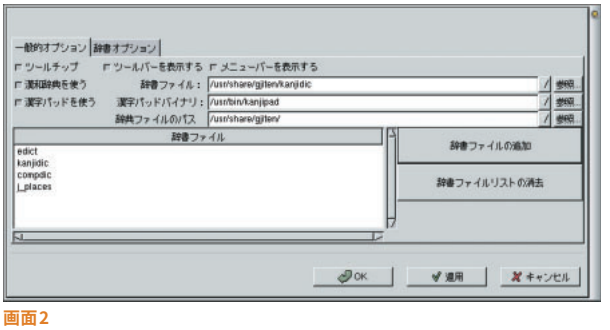
登録した辞書を使って検索を行うには、[表現を入力してください]のフィールドに日本語や英語の文字列を入力し、[検索]ボタンを押せばいい。検索された項目がその下のリストに表示される（**画面3**）。

なお、日本語の動詞については、語尾の活用形を自動的に終止形に修正してくれる。

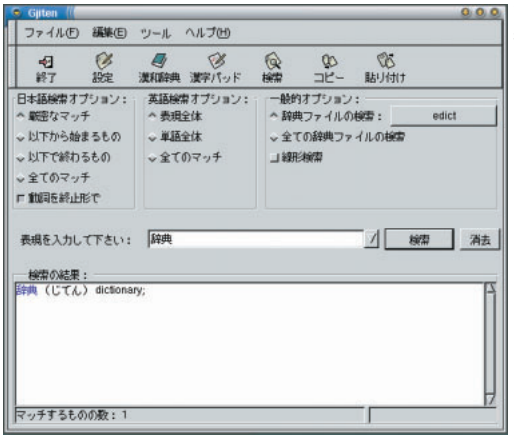
検索文字列に一致した部分は、青い文字で表示される。検索対象には、項



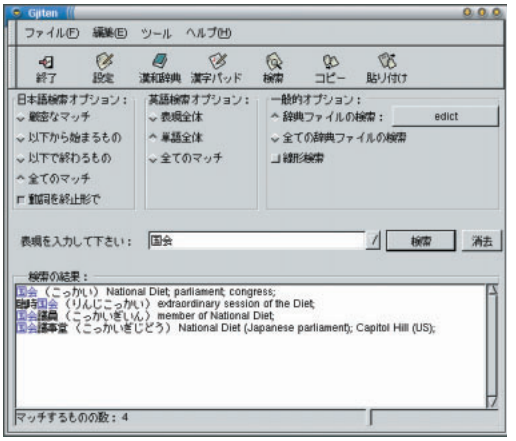
画面1
Gjitenのウィンドウ。付属の日本語カタログが使われる。



画面2
初めて起動した場合は、利用する辞書を登録する必要がある。



画面3
指定した文字列にマッチする項目がリストに表示される。



画面4
検索文字列が項目名の一部に含まれる場合も検索可能だ。

目名だけでなく説明部分も含まれるため、和英辞書（EDICT）を英和辞書として利用可能だ。

初期設定では、日本語・英語ともに検索文字列に厳密に一致するものだけが検索される。検索文字列を一部に含む単語や、検索文字列の単語を使った熟語などを検索するには、日本語・英語それぞれの検索オプションを変更する必要がある。

たとえば、日本語の検索オプションの[以下から始まるもの]にチェックを付けて「国会」を検索すると、「国会」「国会議員」「国会議事堂」のように、検索文字列で始まる項目がすべて検索される（**画面4**）。同様に、[以下で終わるもの]をチェックした場合は、「国会」「臨時国会」のように、検索文字列で終わる項目が検索され、[全てのマッチ]をチェックすると、検索文字列を含む項目がすべて検索される。

一方、英語の検索オプションは3種類用意されている。[表現全体]をチェックした場合は、説明部分（複数ある場合は「；」で区切られる）が検索文字列に一致する項目、[単語全体]をチェックした場合は、説明に含まれる単語が検索文字列に一致する項目、[全てのマッチ]をチェックした場合は、説明の一部でも検索文字列に一致する項目がすべて検索される。

●辞書の切り替えと串刺し検索

検索対象となる辞書のファイル名は、ウィンドウ右上の[辞典ファイルの検索]の右に表示されている。このファイル名をクリックすると、Gjitenに登録された辞書ファイルの一覧が表示され、再度クリックすると検索対象となる辞書を切り替えられる。

また、その下の[全ての辞典ファイルの検索]をチェックすると、登録した辞書ファイルすべてを検索対象とする、いわゆる「串刺し検索」が可能になる。このとき、検索結果には「マッチedict:」のように、検索された項目が属する辞書ファイル名が併せて表示される（**画面5**）。

なお、漢字辞典（KANJIDIC）については、後で説明するように専用のウィンドウが別に用意されているため、この機能は、地名辞書やコンピュータ用語辞書など、アドオン的な辞書をGjitenに追加登録した場合に利用するとよいだろう。

●漢字辞典を利用する

漢字辞典（KANJIDIC）の説明には、各国の漢字コードや辞書の検索番号などが含まれるため、メインウィンドウで検索しても結果がわかりにくい。そこで、漢字辞典の検索ウィンドウと結果表示が別に用意されている。

ツールバーの[漢和辞典]ボタンを押すとウィンドウが開く（**画面6**）。漢字の検索条件は、画数・部首・キーの3種類。それぞれをチェックすることで指定が有効になる。複数の検索条件を指定して、すべての条件に合う漢字だけを検索することも可能だ。

（1）画数による検索

検索したい漢字の画数を指定する。厳密な画数がわからない場合は、プラス／マイナスの誤差として許す画数をあわせて指定すればいい。

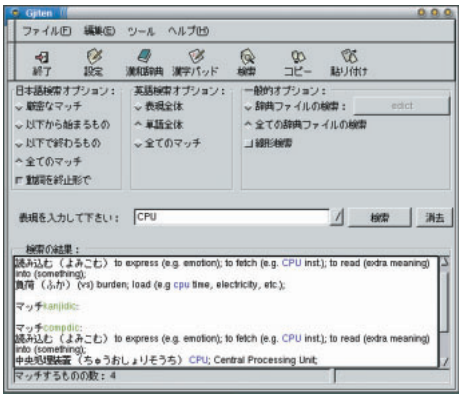
（2）部首による検索

検索したい漢字の部首を指定する。[部首一覧の表示]ボタンで、画数順に部首が表示されるので、その中から選択しよう（**画面7**）。過去に選択した部首は、入力部の[▽]ボタンのリストから呼び出せる。

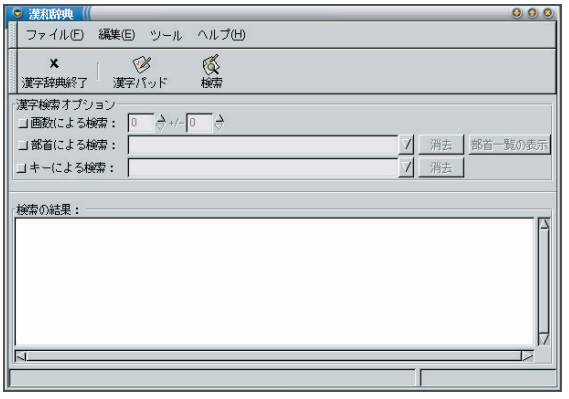
（3）キーによる検索

各漢字の説明に含まれる内容を指定する。たとえば、漢字そのもの（kanjipadで入力）、漢字の読み、英語の意味などだ。過去に入力した内容は、[▽]ボタンで呼び出せる。

これらのいずれか（あるいは複数）を指定した後、ツールバーの[検索]ボ



画面5
複数の辞書ファイルをまとめて検索する「串刺し検索」。



画面6
漢字辞典については検索ウィンドウが別に用意されている。

タンを押すと、指定した条件に合う漢字がウィンドウ下部に一覧表示される（**画面8**）。

それぞれの漢字をクリックすると、漢字の読みや英語の意味、文字コードや辞書の検索番号などの詳しい情報が別ウィンドウに表示される（**画面9**）。表示される内容については、設定ダイアログの[辞書オプション]ページで細かな設定が可能だ。

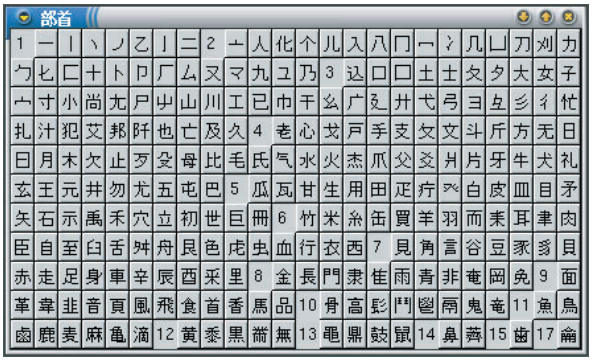
●kanjipadで手書き漢字認識

Gjitenのメインウィンドウや、漢字辞典ウィンドウで、ツールバーの[漢字パッド]ボタンを押すと、手書き漢字認識ソフトのkanjipadのウィンドウが開く。

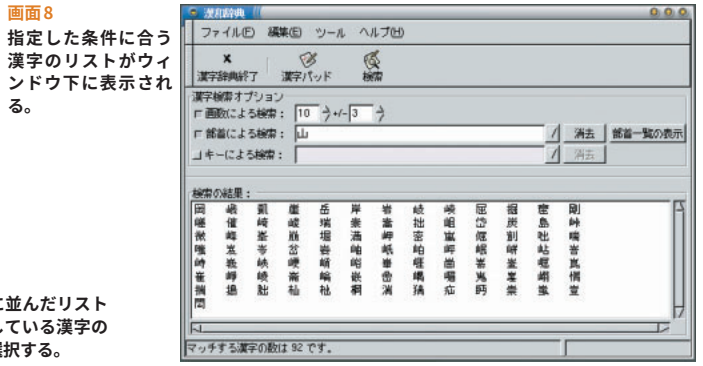
もし、起動に失敗するようなら、まずはkanjipadをインストールしたかどうかを、コマンドラインで「kanjipad &」として確認する。kanjipadがちゃんと起動する場合は、Gjiten側の設定に問題がある。設定ダイアログの[一般的オプション]ページで、[漢字パッドバイナリ]のパス設定を確認しよう。kanjipadが/usr/binにあるのに、「/usr/local/bin/kanjipad」と設定されているかもしれない。

kanjipadの使い方は簡単だ。マウスの左ボタンドラッグで線を引いてパッド上に漢字を描き、最後に[引]ボタンを押すと候補一覧のリストが右側に表示される（**画面10**）。探している漢字が候補にない場合は、[消]ボタンでパッドをクリアして再度挑戦してみよう。kanjipadの認識手法は書き順が占める割合が大きいので、正しい書き順で漢字が書くことが重要だ。

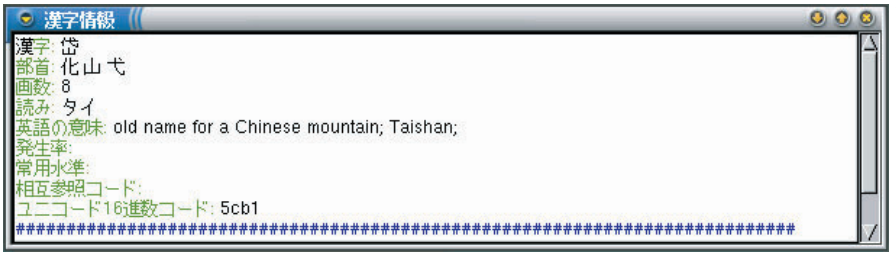
候補リストの漢字をクリックで選択すると、セレクションにコピーされ、中ボタンクリックでペーストされる。これを利用して、Gjitenのメインウィンドウの[表現を入力してください]や、漢字辞典ウィンドウの[キーによる検索]のフィールドにkanjipadの認識結果をペーストすれば、マウスで描いた漢字を調べられるわけだ。



画面7
画数順に並んだリストから探している漢字の部首を選択する。



画面8
指定した条件に合う漢字のリストがウィンドウ下に表示される。



画面9
クリックした漢字に関する詳しい情報を別ウィンドウに表示。

画面10
kanjipadでは、マウスで描いた漢字を認識できる。

安定して使いやすいメールクライアント

Sylpheed

バージョン：0.4.4　　ライセンス：GPL

http://sylpheed.good-day.net/
http://members.linuxstart.com/~sunnyone/mylinux/sylpheed.html（RPM）

Sylpheed（シルフィード）は国産の軽快で使いやすいメールクライアント兼ニュースリーダである。Windows系メールクライアントによく似た外見と、Emacs系メールクライアントの操作感をあわせ持つ。無制限の複数アカウント管理をはじめ、メッセージのスレッド表示や振り分け、ファイルの添付、ビルトイン画像表示など機能も充実している。動作にはGTK＋が必要だ。

●ビルドから初期設定まで

Sylpheedは、tarボールとRPMパッケージ（上記URLを参照）が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ボールからのビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。

「sylpheed&」として起動するとメインウィンドウが開く。画面構成はWindows系メールクライアントでおなじみの3ペイン方式で、フォルダツリー、サマリビュー、メッセージビューで構成される（**画面1**）。

まずは、アカウントの登録を行おう。ツールバーの[アカウント]ボタンを押し、「新規アカウントの設定」ダイアログの[基本]ページで名前やメールアドレス、POP／SMTPサーバ名などを設定する（**画面2**）。他の設定は後で行えばいい。

登録したアカウントは「アカウントの編集」ダイアログに表示される。このダイアログでは、別のアカウントの追加、既存のアカウントの編集・削除などの操作が可能だ。Sylpheedでは、1人のユーザーが複数のアカウントをいくつでも利用できる。

なお、現在のアカウント名はメインウィンドウの右下に表示される。別のアカウントに切り替えたい場合は、この部分をクリックして、リストから選択すればいい。

●メールの受信と閲覧

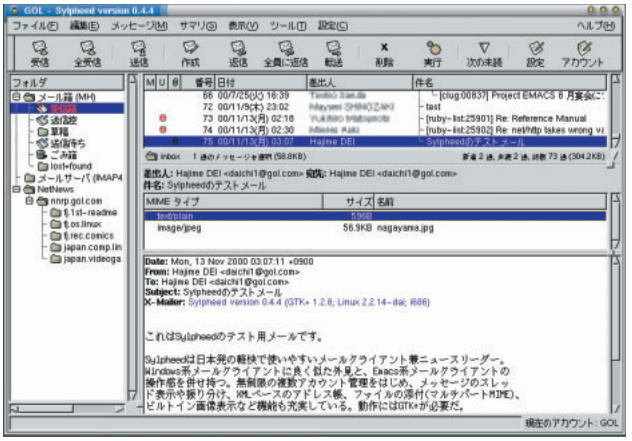
ツールバーの[受信]ボタンを押すと、現在のアカウントのメールサーバに接続して新着メールを受信する。登録したすべてのアカウントの新着メールを一度に受信する[全受信]ボタンも用意されている。

受信したメールは[受信箱]フォルダに格納される。このフォルダを選択すると、メールのタイトルなどがサマリビューにスレッド表示され、サマリビューで選択したメールの本文がメッセージビューに表示される。フォルダツリーやメッセージビューを別ウィンドウで表示することも可能だ。

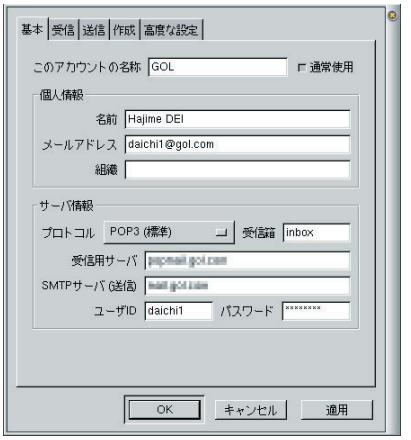
未読メールを閲覧するには、スペースキーを押せばいい。次の未読メールへの切り替え、画面に収まりきらない本文のスクロール、次のフォルダの切り替えなどをインテリジェントに処理してくれる。

マルチパートMIMEによる添付ファイルを持つメールは、サマリビューにクリップアイコン付きで表示され、メッセージビューの上部にファイル名やMIMEタイプのリストが表示される。テキストや画像の場合は、単にクリックするだけでメッセージビューで閲覧可能だ（**画面3**）。このほか、ファイルの保存やMIMEタイプに応じたアプリの起動を行える。

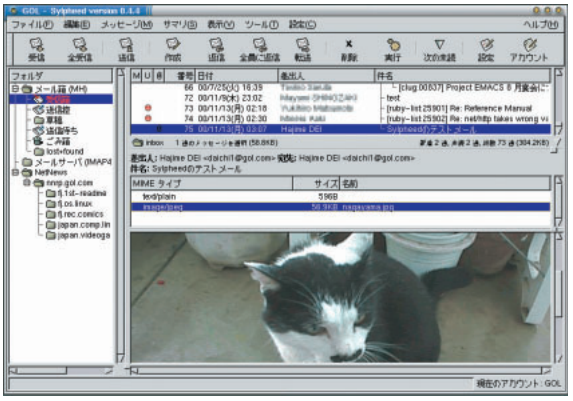
もちろん、整理用の仮想フォルダ（階層化可能）を作成し、メールを振り分ける機能も用意されている。設定ダイアログの[振り分け]ページで、振り分けに利用するヘッダとその内容、移動先フォルダを「ルール」として登録しよう。メール受信後に、[サマ



画面1
Outlook Expressなどと同じ3ペイン構成のウィンドウ。



画面2
アカウントの設定はこのダイアログで。複数アカウント対応だ。



画面3
メールに添付された画像を直接メッセージビューで閲覧できる。

画面4
メッセージ作成ウィンドウで送信するメールを書く。

リ]－[メッセージを振り分ける]を選択すると、設定したルールに基づいて振り分けが行われる。また、メールの受信と同時に振り分け処理を自動実行する設定も可能だ。

●メールの作成とアドレス帳

メールを新規作成するには、ツールバーの[作成]ボタンを押す。また、メールの返事を書く[返信]ボタンや、相手とは別の人に転送する[転送]ボタンも用意されている。

いずれにせよ、メッセージ作成ウィンドウが開くので（**画面4**）、件名や宛先、本文を入力しよう。返信などの場合は、件名や宛先はすでに入力済みだ。本文には自動的に署名（シグネチャ）が挿入される。

メールアドレスの入力は、[アドレス]ボタンで表示される「アドレス帳」（**画面5**）を利用するのが簡単だ。一覧中からメールアドレス（複数可）を選択して[宛先]や[Cc:]ボタンを押すだけで、対応するフィールドにそれらが挿入される。

また、本文の入力を外部エディタで行ったり、gmcなどのファイルマネージャからテキストファイルをドラッグ＆ドロップで本文に挿入することもできる。同様に、添付ファイルについても、リストへのドラッグ＆ドロップによる追加が可能だ。

●ニュースリーダ機能を使う

Sylpheedには、ネットニュースを購読するニュースリーダとしての機能も用意されている。メールを読むのと同じ感覚でニュースグループの記事を閲覧可能だ。ただし、現時点では記事の投稿には対応していない。

ニュースリーダとして使うには、利用するニュースサーバやニュースグループの名前をSylpheedに登録する必要がある。

まずは、フォルダツリーの[NetNews]フォルダ上で右クリックし、メニューから[ニュースサーバを追加]を選択してサーバ名を入力しよう。[NetNews]フォルダの下に、サーバ名のフォルダが作成される。

続いて、そのフォルダ上で右クリックし、メニューから[ニュースグループを購読]を選択して、グループ名（たとえば「fj.os.linux」など）を入力する。これを、購読したい各グループに対して行えば準備完了だ。

ニュースグループの記事を取得するには、そのグループ名のフォルダをクリックするだけでいい。自動的にニュースサーバに接続して、新しいニュースを取得する。

サマリビューには、ニュースの件名や投稿者の一覧がスレッド表示され、メッセージビューには選択したニュースの本文が表示される（**画面6**）。あとは、メールを読む場合と同じようにして閲覧すればいい。もちろん、スペースキーだけを使って未読のニュースを読み進められる。

画面5
XMLベースのアドレス帳。フォルダによる分類や検索も可能だ。

画面6
ネットニュースをメールと同じ感覚で読み進められる。

2ペインタイプのファイルマネージャ

emelFM

バージョン：0.8.2　　ライセンス：GPL

http://www.pitt.edu/~macst92/emelfm/

emelFMは、2つのディレクトリのファイル一覧が表示され、コピーや移動の結果を確認できる2ペイン構成のファイルマネージャだ。ボタンの内容やキーバインド、ファイルの拡張子に応じて起動するコマンドなど、すべてダイアログで設定できる。このほか、任意のコマンドを実行できるコマンドラインが用意されている。ソースの修正とフォント設定により日本語表示が可能。動作にはGTK＋が必要だ。

●ビルドと日本語への対応

emelFMはファイル一式をtar＋gzipしたtarボールのみ配布されている。configureスクリプトは用意されていないが、通常はそのまま「make」→「make install」とすればビルドとインストールが行われる。ただし、そのままだと日本語のファイル名やテキストを正常に表示できないので、ビルドする前にソースを修正しよう。

修正が必要なのは、ソースファイルemelfm.cのフォントファイルの指定と、window.cのフォントファイルの読み込みに関する部分だ。付属CD-ROMに収録したパッチファイル（emelfm-ja.patch）を、emelFMのtarボールの展開先にコピーした後、

```
$ patch -p1 < emelfm-ja.patch
```

とすれば修正が行われ、日本語を含むフォントセットがファイル一覧やビューアで使われるようになる。

●基本的な使い方

「emelfm&」として起動すると、左右2つのペインと中央にボタンが並んだウィンドウが開く（**画面1**）。各ペインには、現在注目しているディレクトリのファイル一覧が表示されている。なお、ドットファイル（先頭が「.」で始まるファイル）については、各ペイン左上の[H]ボタンで表示の有無を切り替えられる。

ディレクトリの変更は、ファイルリストのサブディレクトリをダブルクリックしたり、ディレクトリ表示部に直接キー入力すればいい。このほか、1つ上のディレクトリに移動するボタンや、以前表示したディレクトリを選択できるヒストリーリストも用意されている。

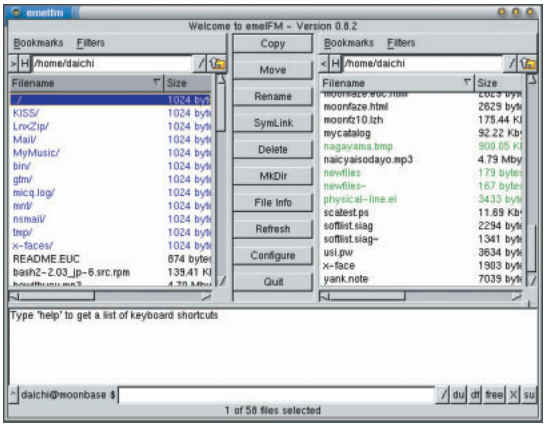
設定の変更は、[Configure]ボタンで開く設定ダイアログで行う。このダイアログはジャンル別に設定項目がページ分けされている。初めて起動した場合は、[General - Page2]ページの[Xterm Command]の設定を「kterm」に変更しておこう（**画面2**）。これで、端末画面で実行されるコマンドがktermを利用するようになる。

●ボタンを利用した操作

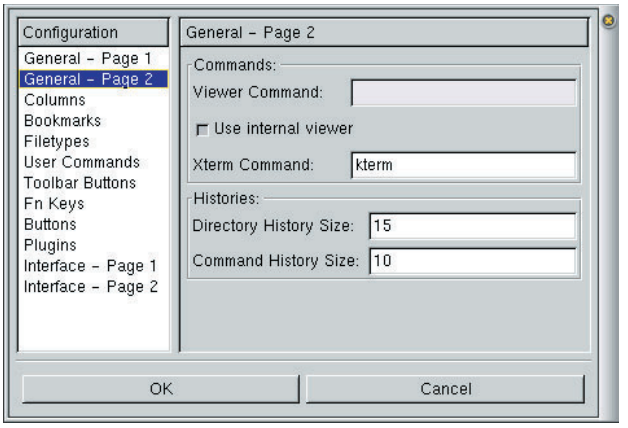
コピーや削除、閲覧などのファイル操作は、選択ファイル（反転表示）に対して行われる。選択するには単にファイルをクリックすればいい。ドラッグやShift－クリックによる範囲指定、Ctrl－クリックによる複数選択にも対応している。

ファイルを選択後、中央部のボタンを押すと、それぞれのボタンに応じた処理が行われる。なお、対象となるのはアクティブなペイン（インデックス部が濃い灰色）の選択ファイルだけだ。他方のペインの選択ファイルには影響しない。

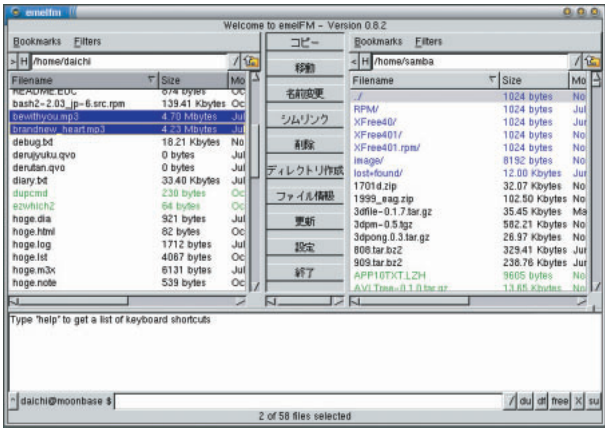
ボタンの多くは、処理の際に両方のペインの情報を利用する。たとえば[Copy]や[Move]ボタンでは、アクティブでないほうのペインのディレクトリが自動的にコピー・移動先となる。もちろん、[Delete]ボタンのように、片



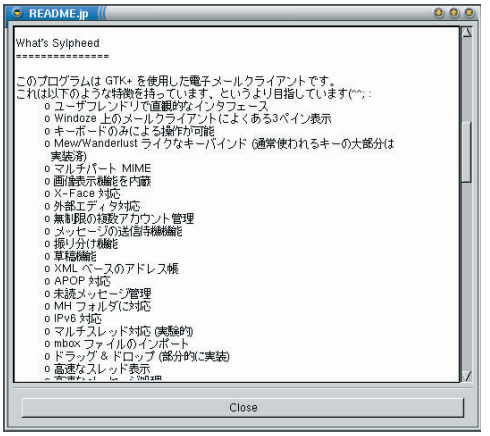
画面1
左右のファイル一覧と下のコマンドラインがemelFMの特長だ。



画面2
設定はすべてこの設定ダイアログで行われる。



画面3
ボタンの名前を日本語を使って変更してみた。



画面4
テキストファイルの内容は内蔵ビューアで閲覧できる。

側のペインの情報しか利用しないボタンもある。

設定ダイアログの[Buttons]ページでは、各ボタンの名前や処理内容の変更をはじめ、新たなボタンの追加、ボタンの削除などを行える。ボタンに日本語の名前（「コピー」など）を付けることも可能だ（**画面3**）。

●ファイルタイプの利用と設定

テキストファイル選択後に右クリックメニューの[View]を選択すると、ファイルごとに内蔵ビューアが起動されて、内容を閲覧できる（**画面4**）。パッチを当ててソースを修正していれば、日本語表示も問題ない。

また、ファイル名をダブルクリックするか、右クリックメニューの[Open]を選択すると、それぞれの拡張子に応じた「ファイルタイプ」の処理が行われる。たとえば、画像ファイルはxvで表示され、RPMファイルの場合はファイル一覧がkterm上に表示され、tarボールの場合はそのディレクトリに展開される、といった具合だ。

また、右クリックメニューの[Choose action]以下には、それ以外の処理を行う項目も用意されている。たとえば、tarボールの場合はファイル一覧の表示、RPMファイルではパッケージのインストールが行われる。

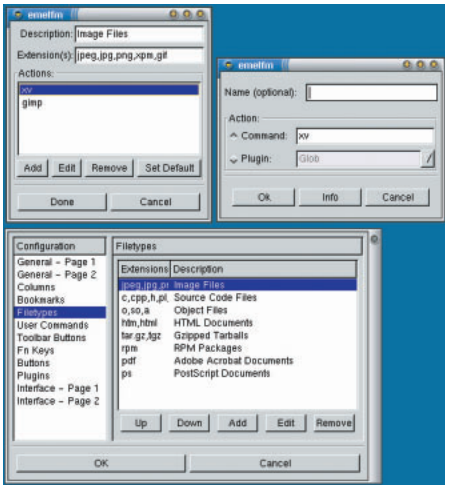
こうしたファイルタイプの設定は、設定ダイアログの[Filetypes]ページで行う（**画面5**）。ファイルタイプごとに、ファイルタイプ名、所属する拡張子（複数可）、アクション（複数可）を設定できる。アクションの内容としては、任意の外部コマンドとemelFMプラグインを利用できる。選択ファイルを表わすマクロ（%f）などについては、[Info]ボタンで表示されるヘルプを参照されたい。

なお、ファイルタイプが設定されていないファイルをダブルクリックすると、処理内容を尋ねるダイアログが表示される（**画面6**）。ここから、新しいファイルタイプを作成したり、既存のファイルタイプに拡張子を追加することも可能だ。

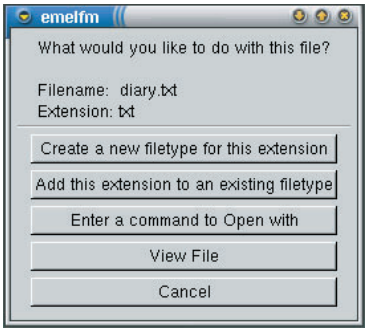
●コマンドラインによる操作

emelFMの特長のひとつに、ウィンドウ下部に用意されたコマンドラインがある。ここでは、ktermのコマンドラインと同じように、任意のコマンドをキー入力して実行できる。Tabキーによるコマンド・ファイル名の補完や、コマンドラインの編集、ヒストリー機能も利用可能だ。また、duやdfなど、一部のコマンドは右側のボタンで実行できる。

lsやcatのように端末内で実行されるコマンドでは、実行結果がenelFMのウィンドウに表示される。パッチを当ててソースを修正していれば、日本語表示も問題ない。一方、XEmacsなど別ウィンドウで開くコマンドを実行することも可能だ。このとき、末尾に「&」を付ける必要はない。



画面5
ファイルタイプの設定でダブルクリックのみの快適環境を実現。



画面6
ファイルタイプが設定されていない場合はこのダイアログが出る。

MP3やWAVEに対応したサウンドエディタ

spwave

バージョン：0.5.2　　ライセンス：LGPL

http://www.itakura.nuee.nagoya-u.ac.jp/people/banno/spLibs/spwave/index-j.html

spwaveは、WAVE／AIFF／MP3などさまざまな形式の音声ファイルに対応するマルチプラットフォームなサウンドエディタだ。波形の切り出しや削除、振幅の変更、サンプリング周波数やビット数の変換、スペクトル解析などが可能だ。編集後に、MP3形式でファイルを出力することもできる。日本語カタログが付属しており、日本語環境ではメニューなどがすべて日本語で表示される。動作にはGTK＋（またはMotif）とspLibsライブラリ群が必要だ。

●ビルドとインストール

spwaveは、tarボールとRPMパッケージの両方で配布されている。tarボールからインストールする場合、先にspLibsライブラリ（spBase、spLib、spAudio、spComponent、spPlugin、spMpeg）をインストールする。また、spMpegの作成にはFreeAmpとBladeEncのソースも別途必要だ。インストールの順番や手順（各ライブラリのREADME_SJIS.txtを参照）を考えるとかなり面倒な作業なので、RPMパッケージを利用したほうがいい。

実行に必要なRPMバイナリパッケージは、spwave本体（spwave-0.5.2-4.i386.rpm）とプラグイン集（spPlugin-0.8.1-2.i386.rpm）、MPEG入力プラグイン（spMpeg-iplugin-0.8.1-2.i386.rpm）の3つ。spwaveはspPluginに依存しているため、spPlugin→spwaveの順でインストールするか、一度にまとめてインストールしよう。

さらに、BladeEncのコードを利用している関係で今回のCD-ROMには収録していないMPEG出力プラグイン（spMpeg-oplugin-0.8.1-2.i386.rpm）を、Webサイト（上記URLを参照）から入手してインストールしておこう。

●起動と画面表示

「spwave&」として起動すると、白紙のウィンドウが開く（**画面1**）。環境変数LANGの設定が適切なら、日本語カタログを利用した日本語表示が行われるはずだ。

音声ファイルを開くには、ツールバー左端のボタンか、[ファイル]－[開く]を選択してオープンダイアログを開き、ファイル名を選択すればいい。gmcなどからのドラッグ＆ドロップにも対応している。ウィンドウを複数開き、複数の音声ファイルを同期させて編集することも可能だ。

WAVE／MP3などのファイル形式では、サンプリング周波数やビット数などのパラメータは自動的に判定される。AUのように自動判定できない形式の場合は、パラメータを設定するダイアログが開くので、周波数などを指定する必要がある（**画面2**）。

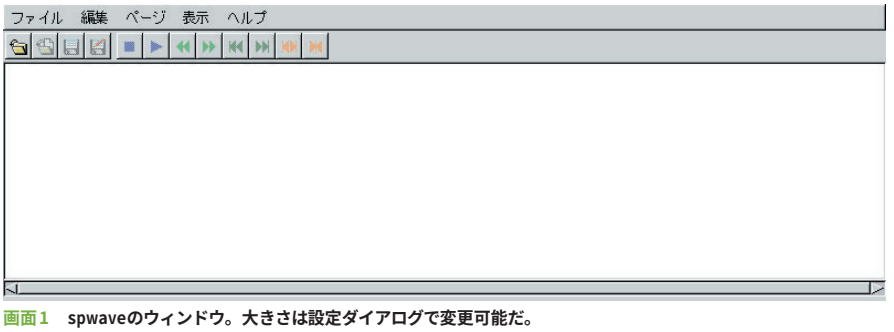
1曲分のMP3ファイルなど、記録時間が長い場合には、読み込みに時間がかかる。しばらく待って読み込みが終了すると、全波形がウィンドウ内に表示される（**画面3**）。ウィンドウのサイズは自由に変更可能だ。

●基本的な使い方

spwaveの操作は一般的なサウンドエディタと同じで、マウスのドラッグにより処理範囲を選択（反転表示）し、さまざまな処理を行う。ツールバーのボタンにより、選択部分の再生や停止、表示の拡大・縮小、表示位置の変更などが可能だ。また、Emacs風のキーバインドによるキー操作も行える。

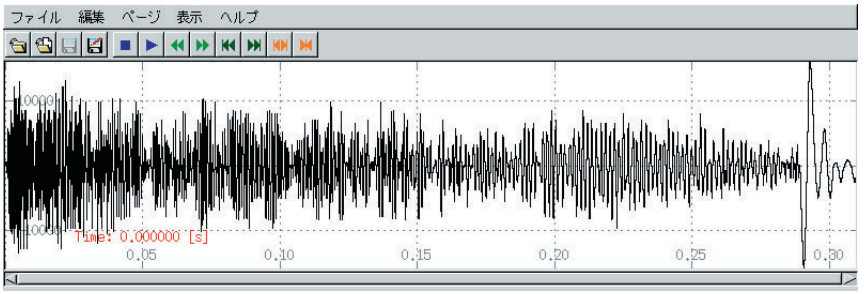
たとえば、波形の一部を削除するには、その部分を選択した後で、[編集]－[削除]を選択すればいい。何かまずい点があれば、[編集]－[取り消し]（またはAlt－Zキー）で無制限の操作の取り消し（アンドゥ）が可能だ。

同様に、[編集]－[イレース]では選択部分の波形がフラットになり、[編集]－[切り出し]（またはShift－Cキー）では選択部分以外が削除され、[編集]－[取り出し]（またはShift－Xキー）



画面1　spwaveのウィンドウ。大きさは設定ダイアログで変更可能だ。

画面2
ファイル形式を自動判定できない場合にはこのダイアログで設定。



画面3　読み込んだファイルの音声波形がウィンドウに表示される。

では選択部分を別ファイルとして編集できる。なお、選択範囲をウィンドウ外にドラッグ&ドロップすることでも取り出し処理が可能だ。

spwaveでは、スクロールや拡大・縮小の処理のたびにファイルを読み込むため、大きなファイルだと処理が重くなる。まずは、効果音のWAVEファイルなど短いファイルで、基本的な操作に慣れるといいだろう。

●さまざまな波形処理を行う

フェードイン・フェードアウトや、振幅の変更・反転などは、[編集]メニューに用意されている。これらは、選択範囲の波形が処理の対象になる。

たとえば、波形全体の振幅が小さい（あるいは大きい）場合は、[ページ]－[全て選択]（またはAlt－Aキー）で全波形を選択した後、[編集]－[振幅の変更]でダイアログを開く（**画面4**）。

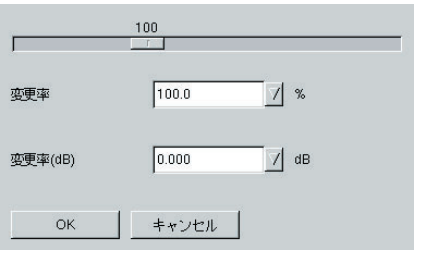
振幅の変更率は0～300%の範囲で指定できる。振幅が小さい場合は100%より大きな値、大きすぎる場合は100%より小さな値を指定すればいい。スライダで変更率を指定したり、dB単位で指定することも可能だ。[OK]ボタンで実際の処理が行われる。

また、フェードインやフェードアウトの場合は、ダイアログは表示されず、メニューを選択した時点で選択範囲の波形が即座に処理される。いずれの場合も、処理後の波形を再生してみて、まずい点があるようならアンドゥすればいい。

このほか、サンプリング周波数（8000～96000）やビット数（8～64）の変換、MP3用のID3タグ設定も可能だ。これらは波形全体に作用するため、いちいち範囲指定する必要はない。

●波形のスペクトル解析も可能

spwaveは研究用に開発されたソフトなので、初期設定のスペクトル解析

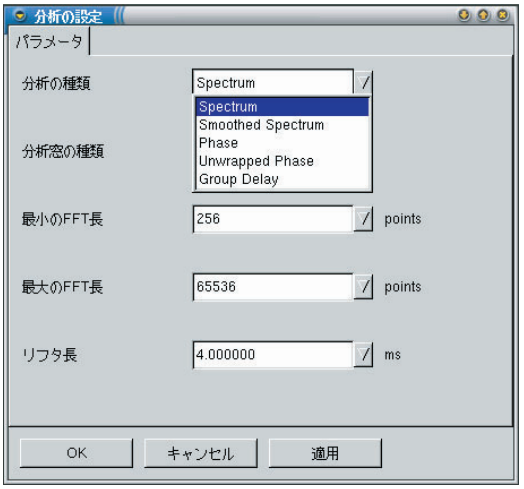


画面4
振幅の変更を設定するダイアログ。変更率を0～300%の範囲で設定。

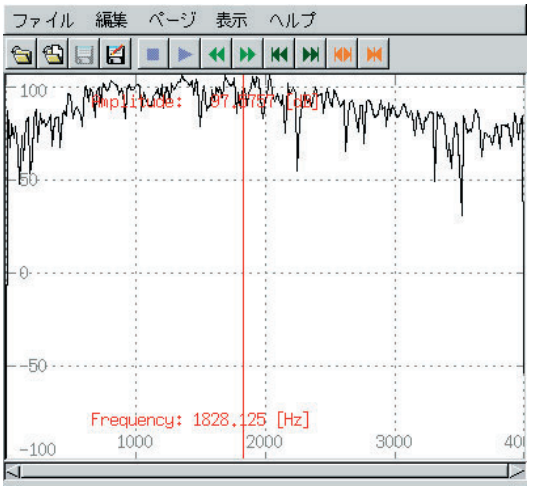
をはじめ、平滑化スペクトル、位相、アンラップド位相、群遅延などさまざまな分析方式をサポートしている。分析方式は、[編集]－[分析の設定]のダイアログで変更する（**画面5**）。パラメータの設定も可能だ。

分析を行うには、分析位置をマウスで指定して、右クリックメニューの[分析]－[広帯域分析]（あるいは[狭帯域分析]）を選択する。[分析]－[指定範囲の分析]で指定した範囲の分析も可能だ。分析結果は別ウィンドウに表示される（**画面6**）。

なお、spwaveは、ホームディレクトリの.spwave_tmpディレクトリに一時ファイルを作成する。通常はspwave終了時に自動的に削除されるが、異常終了したりすると残ってしまうことがある。その場合は、spwaveを実行していない状態で、これらの一時ファイルを削除すればいい。



画面5
スペクトル解析のほか、4種類の分析方式をサポート。定。



画面6
スペクトル解析の結果は別ウィンドウに表示される。

ネットワーク対戦可能な本格卓球ゲーム

Cannon Smash

バージョン：0.4.7　　ライセンス：GPL

http://cannonsmash.sourceforge.net/
http://www.utmc.or.jp/~nan/csmash/（日本語）

Cannon Smashは、ラリーが気持ちいい本格的な3D卓球ゲームだ。プレイヤーは3タイプ用意され、コンピュータ相手のソロプレイと人間同士のネットワーク対戦が可能だ。SourceForgeのダウンロードランキングでトップ10に入るなど、海外でも評価が高い。動作には3DライブラリのMesa3DとGLUTが必要だ。バージョン0.4.6で追加された簡易モードを使えば、3Dアクセラレータのない環境でもなんとか動作する。

●ビルドとインストール

Cannon Smashをビルドする前に、3DライブラリのMesa3DとGLUTをあらかじめインストールしおく必要がある。

Cannon Smashの最新版は、tarボールで配布されている。ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。なお、サウンドドライバEsounDのない環境では、ビルド時にエラーが表示される。あらかじめ、main.cppの87行目の「#elif」を「#else」に修正しておこう。

別サイトで配布されているRPMパッケージは、バージョン4.0のrpmコマンドに対応したものなので、バージョン3.0のrpmコマンドを採用しているRed Hat 6.x系ディストリビューションでは利用できない。tarボールに含まれるspecファイルを利用して、「rpm -ta csmash-0.4.7.tar.gz」とすると、バイナリのRPMパッケージがリビルドされるので、それをインストールするとよいだろう。

●ゲームの起動とプレイヤーの選択

「csmash&」として起動すると、白熱したラリーが展開されるメニュー画面が表示される（**画面1**）。

3Dアクセラレータがない（あるいはXでは有効にならない）場合は、装飾的な表示をできるだけ簡略化した簡易モードを指定する－S（大文字）オプションを付けて起動するといい。画面はさびしくなるが、CPUが速ければなんとかプレイできる。

もし、あなたがCannon Smashを初めてプレイするなら、メニューから「How to play」を選択して、プレイ中の操作を解説したデモを見よう。このほか、設定の変更（Config）や練習（Training）を行うことも可能だ。

「Start Game」を選択するとゲーム開始だ。まずはプレイヤーの戦型を選択する。テーブルに接近した状態で戦う「Pen Attack」、少し下がった状態で戦う「Pen Drive」、テーブルからかなり下がった状態で戦う「Shake Cut」の3種類が用意されている（**画面2**）。初めのうちは、攻撃しやすい最初の2つのどちらかを選ぶといい。対戦相手となるコンピュータの戦型は、毎回ランダムに決定される。

一方、人間同士のネットワーク対戦を行う場合は、片方が－s（小文字）オプションを付けてサーバとして起動した後、もう一方が－cオプションを付けてクライアントとして起動する。この場合はメニュー画面はスキップされ、プレイヤーの戦型を選択する画面から始まる。

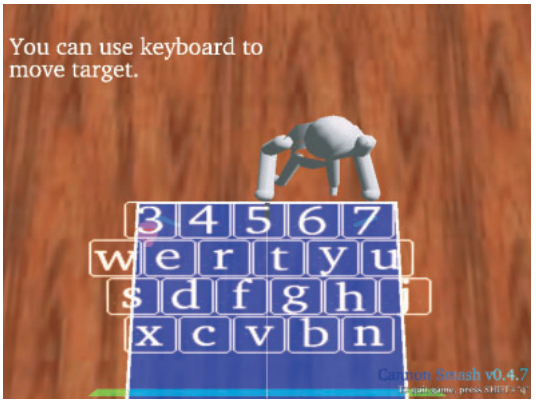
●マウスとキーボードで操作

プレイヤーの操作には、キーボード（左手）とマウス（右手）を併用する。キーボード左半分の3キーからnキーまで22個のキーを利用して、相手コートにボールを打ち込む位置（**画面3**）を指定すると、その位置にインジケータ（▼）が表示される。ラリーの最中にも位置を変更できるので、フォアとバックへの打ち分けや前後の揺さぶりといった戦術が可能だ。

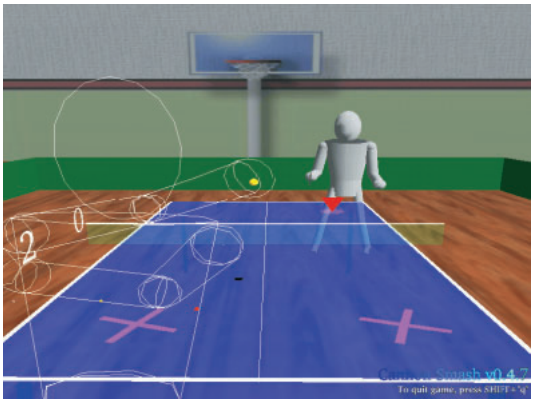
マウスを移動させると、そのままプレイヤーの移動に反映される。相手の返球の強さやコースを考慮して、最適な位置に移動しよう。たとえば、相手

画面1
メニュー画面のバックでは白熱したラリーが続く。

画面2
操作するプレイヤーを3種類のタイプから選択。



画面3
相手のコートに打ちこむ位置はキーボードで指定する。



画面4
返球の軌跡上に表示された赤い点が最適打点だ。

がネット際にサーブをしてきた場合は前進し、浮いたボールをスマッシュされそうなら後ろに大きく下がって待つといった具合だ。

スイング動作は、マウスの左右ボタンに割り当てられている。左ボタンがバックハンド、右ボタンがフォアハンド担当だ。返球時にはボールの軌跡と最適打点が表示されるので（**画面4**）、赤い点までボールが来るのを待って、左右いずれかのボタンをクリックしよう。タイミングが合っている場合は「Nice!」、ずれた場合は「Bad」と表示される。

ルールは実際の卓球に基づいており、5球ごとにサーブをチェンジし、先に21ポイントを取ったほうが勝ちだ（1セットマッチ）。

●練習モードでひたすら鍛錬

試合に勝てないようなら、まずは練習モードで訓練するという手もある。メニュー画面で「Training」を選択すると、高速ラリーとフットワーク練習を行える。

高速ラリー（Fast Rally）練習は、フォアハンドのラリーを何回続けられるかを競うものだ（**画面5**）。キーボードによる位置の指定は無効になっており、返球されるコースもほぼ一定なので、マウスクリックのタイミングを覚えるのに最適だ。

一方、フットワーク（Footwork）練習では、相手がこちらのフォアとバックに交互に返球してくるので、マウス移動の練習になる。左右ボタンをうまく使い分けられるように、バックの球をバックハンドで返す練習をしてもよいだろう。

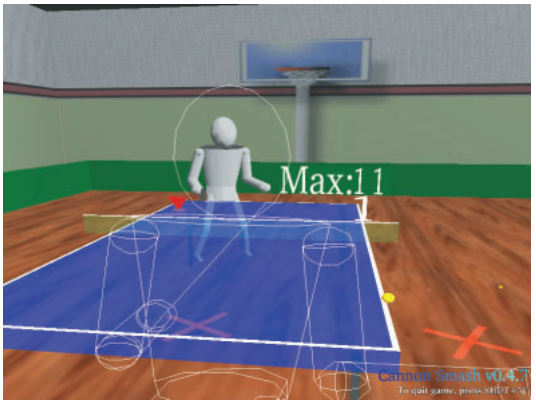
いずれの場合も、テーブル中央のネット付近に現在のラリー回数と過去最高のラリー回数が表示されるので、これを励みにして鍛錬すべし。

●ルールや設定のカスタマイズ

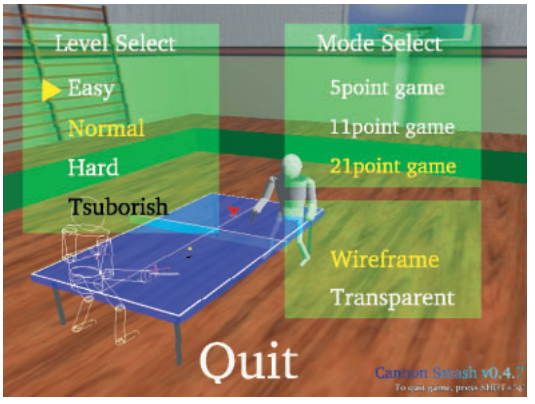
21ポイント先取ではゲームが長すぎるようなら、5ポイント先取や11ポイント先取に変更しよう。メニュー画面で「Config」を選択すると設定変更画面になる（**画面6**）。

ここでは、勝利ポイントの変更のほか、コンピュータの強さやサウンドドライバの切り替え、プレイヤー表示方法の切り替えなども可能だ。たとえば、コンピュータが強すぎて何度対戦しても勝てないようなら、[Level Select]を「Easy」に変更する。

これまでの卓球を題材としたゲームは「温泉でピンポン」レベルのものが多かったが、Connon Smashは球のスピードや軌道がスポーツとしての卓球にかなり近い。このため、ラリーを続けているだけで実に爽快感がある。球がテーブルやラケットに当たる音を聞いていると、卓球部に所属していた人はありありと記憶がよみがえってくるのではないだろうか。



画面5
マウスクリックのタイミングを掴むのに最適な高速ラリー練習。



画面6
勝利ポイントやコンピュータの強さはこの画面で変更する。

任意の画像を使えるジグゾーパズルゲーム

KPuzzle

バージョン：0.1　　ライセンス：GPL

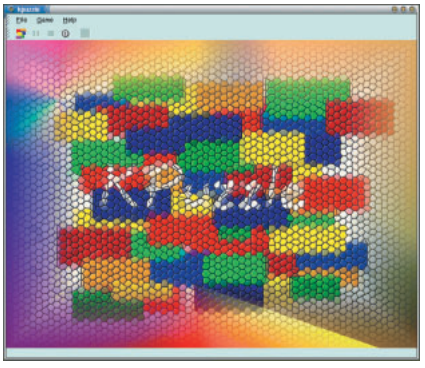
ftp://ftp.kde.org/pub/kde/unstable/apps/games

KPuzzleは、KDE 1.x環境で動作するジグゾーパズルゲームだ。付属する11種類の画像に加え、任意のBMPファイルをパズルの盤面として利用できる。ピースの大きさや難易度をカスタマイズでき、ピースごとに制限時間をつけるなど、さまざまなルールのバリエーションが用意されている。実際のジグゾーパズルと同様のピース形状のほか、より難易度の高い矩形ピースでプレイすることも可能だ。

●ビルドとインストール

KPuzzleは、ソース一式をtar＋bzip2したtarボールのみ配布されている。ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。

なお、Kondara MNU/Linux 1.2（あるいは2000）では、KDE 1.xを/usr/kde1x以下にインストールしている関係で、そのままだとconfigureスクリプトのKDEチェックではねられてしまう。この場合は、「./configure --x-includes=/usr/kde1x/include/kde」とすればOKだ。KDE環境以外でビルドする場合は、環境変数QTDIRとKDEDIRが設定されているかどうかも確認しよう。



画面1　美しいイメージが表示されるタイトル画面。

●ピースを適切な位置に置け

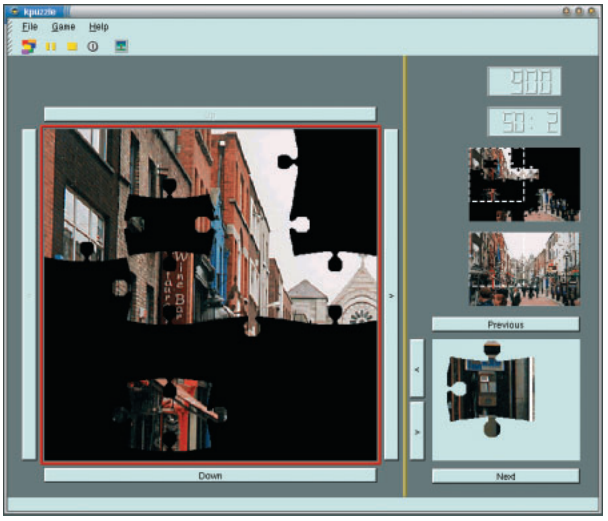
「kpuzzle&」とするか、パネルの[ゲーム]－[KPuzzle]を選択すると、ウィンドウが開いてタイトル画面が表示される（**画面1**）。

まずは、ツールバー左端のボタンを押して、盤面に利用する画像を選択する。KPuzzle付属のBMPファイルが一覧表示されるので、適当に選択しよう。その他のBMP画像を盤面として利用することも可能だ。

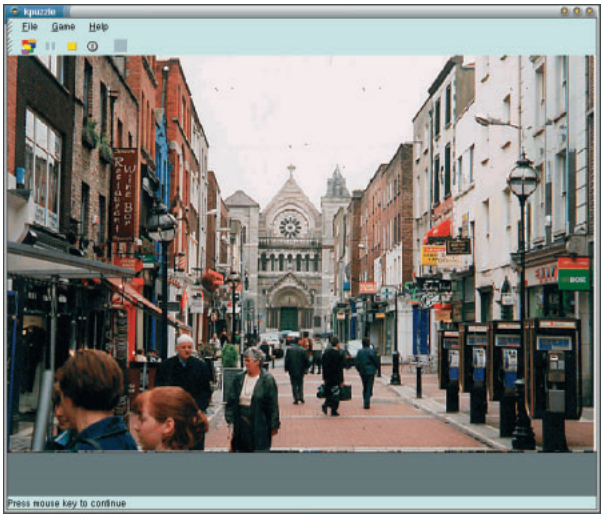
続いて、ゲームの種類（5種類）・ピースの大きさ（5種類）・難易度（3種類）を設定するダイアログが開く。はじめのうちは、初期設定のままプレイすればいいだろう。[OK]ボタンを押すとプレイ開始だ。

プレイ中のウィンドウには、左側に盤面の一部、右側には現在の盤面と完成図の縮小イメージ、現在のピースなどが表示される（**画面2**）。マウスを使ってピースを移動させ、左ボタンクリックで配置しよう。位置と向きが正しければ、そのまま盤面にピースが置かれて次のピースが現れる。ピースの向きは右ボタンと中ボタンのクリックで変更可能だ。

右下の[Previous]と[Next]ボタンを押すと、前後のピースへ切り替えられる。実際のジグゾーパズルと同様に、四隅や端のピースを先に配置すると解きやすい。すべてのピースを盤面に配置すると、「You win!」とメッセージが表示された後、完成図がウィンドウに表示される（**画面3**）。



画面2　ピースを盤面に配置する。わかりやすいピースから始めよう。



画面3　ジグゾーパズルが完成すると、画像全体が表示される。

ペンギンが活躍するタイピングゲーム

TuxTyping

バージョン：0.5　　ライセンス：GPL

http://www.geekcomix.com/dm/tuxtype/
http://www.libsdl.org/（SDL）

TuxTypingは、Linuxのマスコットであるペンギンのtux君が、落下する魚を食べるというコンセプトのタイピングゲームだ。タイピング習得のためというより娯楽性の高いゲームだ。面ごとに変化する美しい背景や、軽快なBGMも楽しい。動作には、SDL、SDL_mixer、SDL_image、SMPEGの各ライブラリが必要だ。

●ビルドから起動まで

初めに、SDLをはじめとするライブラリをインストールしておこう。上記URLから入手できるほか、本誌付録のCD-ROMにも収録されている。

TuxTypeは、tarボールとRPMバイナリパッケージの両方が配布されているので、使用しているディストリビューションに合わせて選択しよう。tarボールからのインストールは、「./configure」→「make」→「make intsall」という一般的な手順だ。

「tuxtype&」として起動すると、フルスクリーン画面に切り替わってタイトル画面が表示される（**画面1**）。ウィンドウ表示でゲームを行うには、「tuxtype -w&」と－wオプションを付けて実行すればいい。

●落下する魚を受け止めろ

まずは、タイトル画面のメニューで、「Key Cascade」（1文字ずつ落ちてくる）か、「Word Cascade」（単語が並んで落ちてくる）のいずれかを選択する。続いて、ゲーム難易度を選択するとプレイ開始だ。

画面には、われらがTux君が、文字が描かれた魚が落ちてくるのを待ちうけている。Tux君の操作は簡単で、それぞれの魚に書かれている文字を順番に入力すればいい。正しく入力すると、Tux君が移動してその魚を食べてくれる（**画面2**）。

文字を間違えたり、魚を食べ終わる前に次の文字を入力してはいけない。Tux君が別の位置に移動して、元の魚を下に落としてしまう（LIFEが1つ減る）。決められた数の魚を食べるとその面はクリアで、次の面に切り替わる。Tux君のLIFEが0になるとゲームオーバーだ。

「Word Cascade」では、単語の文字数分の魚が横に並んで落ちてくるので、それらの文字をまとめて入力する必要がある。最初は3文字の単語だけだが、面が進むに連れてもっと長い単語が現れる（**画面3**）。

このほか、難易度を「Hard」にすると、魚ごとに落下スピードが変わる。先に画面に登場した魚が、先に下に落ちるとは限らないので、素早い判断と高速なタイプ速度が必要だ。

画面1
軽快なBGMとともにタイトル画面が表示される。

画面2
画面上部から落ちてくる魚に書かれた文字をタイプしよう。

画面3
「Word Cascade」では単語を間違いなく入力しなければならない。

プロセス情報をツリーやリストで表示する

GProc

バージョン：0.6.0　　ライセンス：GPL

http://gproc.cjb.net/

GProcは、現在のプロセス情報をツリー構造（あるいはリスト）で表示し、マウスを使ってプロセスの停止や強制終了など行えるソフトだ。ウィンドウの外見をスキンで変更できるアプリ版と、GNOMEパネルに常駐するアプレット版が用意されている。ふだんは小さなウィンドウやパネルに収まっているため、他のアプリの邪魔にならない。アプレット版の実行にはGNOME 1.2以降が必要だ。

●ビルドとインストール

GProcは、アプリ版とアプレット版それぞれのtarボールとRPMパッケージが用意されている。ディストリビューションに合わせて選択しよう。tarボールからのインストールも、「make」→「make install」とするだけなので難しくはない。

続いて、アプリ版用スキンのtarボールを展開する。付属CD-ROMには、Brushed/BlueSteel/SuedE/Cyrus/Viridisの5つが収録されている。これらを、RPMパッケージの場合は/usr/gproc/skins、tarボールの場合は/usr/local/gproc/skinsに展開すればいい。

●プロセス情報の表示と制御

アプリ版の場合は「gproc&」とすると、小さなボタン状のウィンドウ（**画面1**）とヘルプが表示される。アプレット版の場合は「gproc_applet&」とするか、パネルメニューの[アプレット]－[ユーティリティ]－[GProc]を選択すると、GNOMEパネルに常駐する。

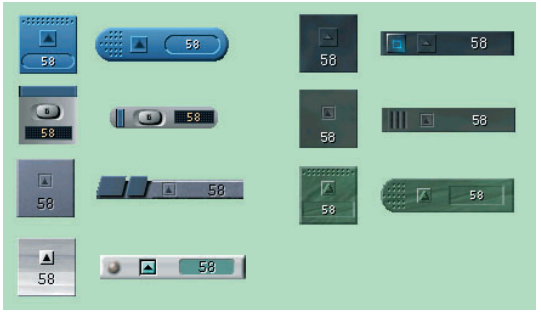
どちらの場合も、最初はプロセス数を示す数字が表示されているだけだ。アプリ版では中央の[▲]ボタン、アプレット版では上部の[!]ボタンをクリックすると、プロセス情報がツリー表示される（**画面2**）。

プロセスの停止や強制終了を行うには、対象となるプロセスをクリックで選択し、右クリックメニューで表示されるシグナル一覧から、目的のシグナルを選択する。たとえば、停止の場合は「SIGSTOP」、強制終了の場合は「SIGKILL」だ。一般ユーザーの場合は、自分が所有していないプロセスは制御できない。

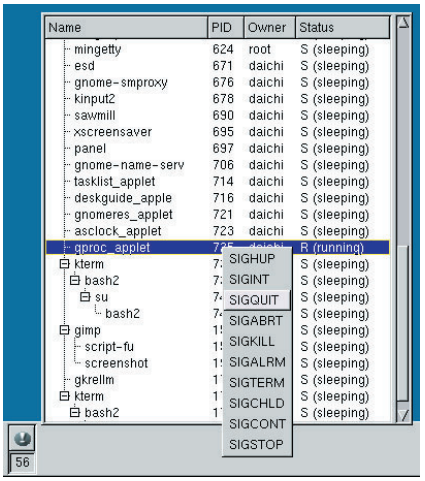
●設定ダイアログでスキンを変更

アプリ版では[▲]ボタンを右クリックして[GProc properties]を選択、アプレット版では右クリックメニューの[Properties]を選択すると、設定ダイアログが開く（**画面3**）。

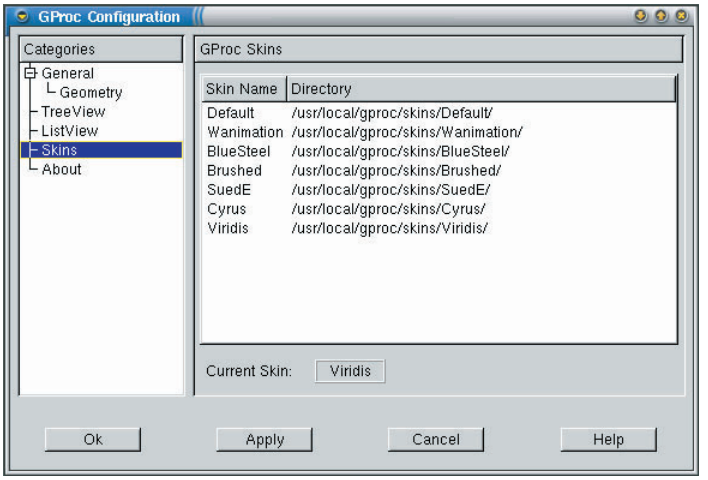
設定項目はジャンル別にページ分けされており、表示形式（リスト／ツリー）の切り替えや、プロセス情報の更新間隔（初期値は1秒）の変更、ウィンドウの形（縦型／横型）ウィンドウの位置やサイズの設定、スキンの変更（アプリ版のみ）などが可能だ。



画面1
アプリ版はスキンを利用して外見をさまざまに変更できる。



画面2
アプレット版を利用してプロセス情報をツリー表示。



画面3
さまざまな設定がジャンル別にページ分けされた設定ダイアログ。

DOS形式のフロッピーを簡単に扱える

MToolsFM

バージョン：1.6　　ライセンス：GPL

http://www.core-coutainville.org/MToolsFM/

MToolsFMは、MS-DOSファイルシステムで保存されたDOSやWindowsのフロッピーを、Linux上で簡単に扱えるファイルマネージャ（以前は「mfm」という名前だった）だ。mountコマンドでいちいちフロッピーをマウントすることなく、フロッピー上のファイル一覧を参照し、ファイルのコピーや削除を行える。実行にはmtools（mdir／mcopy／mdelなどからなるツール群）とGTK+が別途必要だ。

●ビルドとインストール

MToolsFMのビルドにはmtoolsが必要だ。もっとも、たいていのディストリビューションにはmtoolsが最初から含まれているので、新たにインストールする必要はないだろう。ただし、mtools 3.9.7より古いバージョンにはバグが残っており、MToolsFMが止まってしまう可能性がある。そのような場合には、最新版のmtools（3.9.7）をWebサイト（http://mtools.linux.lu/）より入手しよう。

MToolsFMはtarボールとRPMパッケージの両方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ボールの場合は、「./configure」→「make」→「make install」という手順でビルドとインストールが行われる。

以前のバージョン（mfm）をインストールしていた場合は、今回実行ファイル名が変更されたため、アンインストールやRPMパッケージの削除を手動で行う必要がある（設定ファイルはそのまま流用可能）。なお、ソフト名が変更されたのは、別ソフト（Motif File Manager）と実行ファイル名が衝突するためだ。

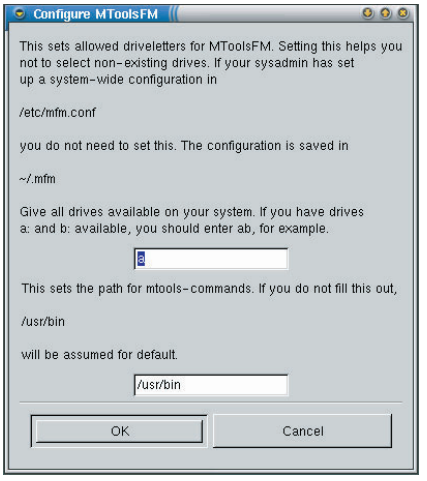
●フロッピーの内容を一覧表示

「MToolsFM&」として起動すると、左右2つの領域（ペイン）を持つウィンドウが開く（**画面1**）。最初はどちらもカレントディレクトリのファイル一覧が表示されている。ディレクトリを切り替えるには、リスト中のディレクトリをダブルクリックすればいい。

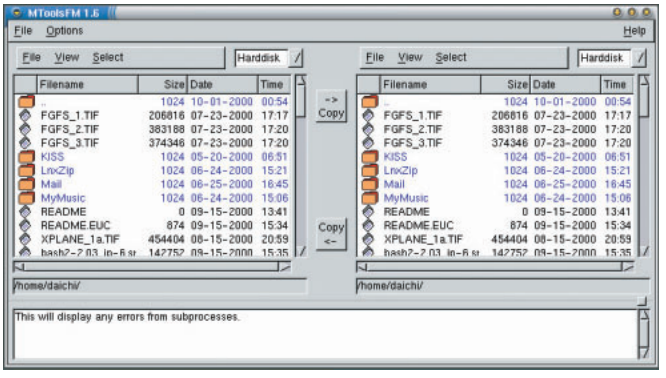
フロッピーをFDDに挿入し、どちらかのペインの左上のリストボックスを「Hardisk」から「a:」に切り替えると、フロッピーのファイル一覧が表示される。長いファイル名もそのまま表示可能だ（**画面2**）。

ファイル操作の際は、対象となるファイルをクリックで選択する。Shift／Ctrlキーによる範囲・複数選択や、メニューの[Select]－[All]による全ファイル選択も可能だ。その後、中央部の[Copy]ボタンを押すと、選択したファイルをまとめてコピーできる。このほか、[File]メニューや右クリックメニューから、ファイルの削除や名前の変更、印刷などを行える。

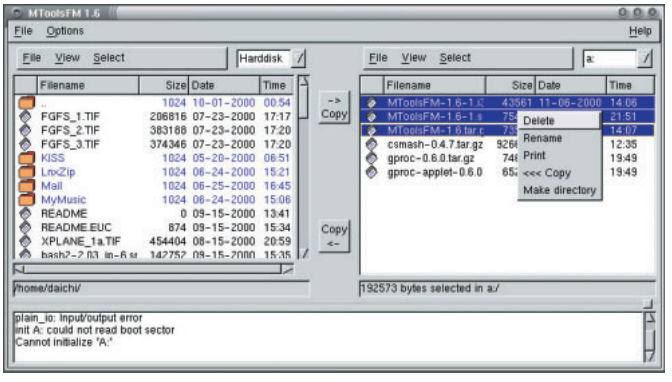
なお、FDDが複数あったり、mtoolsが/usr/bin以外に置かれている場合は、[Options]－[Configure MToolsFM]で設定ダイアログを開いて、設定を変更する必要がある（**画面3**）。



画面3
複数のFDDのドライブ名やmtoolsのディレクトリはここで設定。



画面1
2つのファイル表示領域を持ち、ファイルを簡単にコピーできる。



画面2
VFATに対応しているので、長いファイル名も正しく表示される。

Software Catch Up

インターネットと融合したオフィススイート

ThinkFree Office

ThinkFree OfficeはJavaで動くオフィススイートとして、Linuxをはじめ、Windowsなどで動作する。また、ハードディスクストレージをインターネット上に持つという、新しいスタイルとして注目されている。

文：塩田紳二
Text : Shinji Shioda

価格	未定
問い合わせ先	シンクフリー・ジャパン Tel:03-5777-6671 http://www.thinkfree.ne.jp/

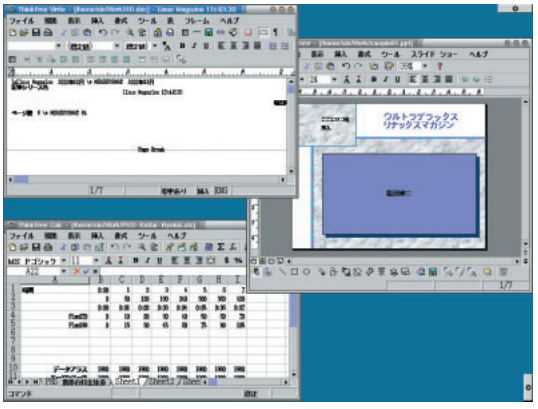
ThinkFree Office（**画面1**）は、Javaで記述されたマルチプラットフォーム対応のオフィススイートである。Javaを使っているため、原則的にはプラットフォームに依存することなく実行が可能だが、現状で対応しているのはWindowsとLinux環境のみである。それ以外の環境では、Macintoshとほかの UNIX環境への対応が予定されている。また、Javaが基本的に多国言語に対応しているため、リソースなどの変更により各国語対応が可能となる。現在のバージョンは、日本語、韓国語、中国語などのアジア圏言語と、英語などの欧米圏言語に対応している。

ThinkFree Officeは、無料のライセンスフリーバージョン（スポンサードエディション。**画面2**）とスタンダードエディションがあり、スタンダードエディションは広告の表示が行われないかわりに、初回年間登録料24.99USドルが必要になる。ただし、現在このサービスは米国でのみ行われており、日本国内ではスタンダードエディションのみの提供で、スポンサードエディションは提供されないようだ。日本国内向けのサービスは2000年第4四半期開始予定で、いまのところ料金を含めて未定となっている。

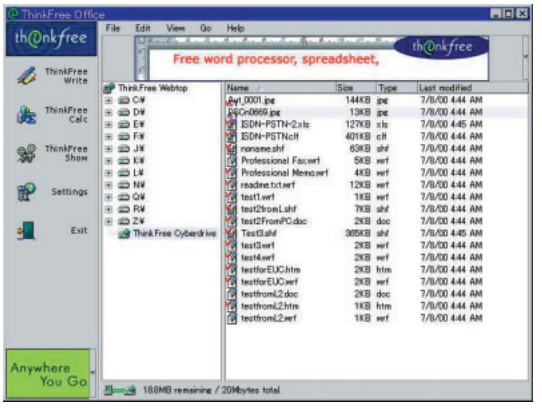
ちなみに英語版でもメニューなどが英語になっているだけで、日本語フォントを持つ日本語環境で動作させれば、日本語の入力、表示は可能である。これは、Javaが多国語対応機能を持つためだと思われる。

また、評価用としてオフラインで利用できるThinkFree Officeも提供されており、こちらは30回までしか起動できないが、ThinkFreeのWebサイトで登録を行うことで継続して利用が可能になる。

ThinkFree Officeのもうひとつの特徴は、インターネット上のサービスであるCyberDriveである。これは、ThinkFree Officeユーザーに提供されるインターネット上のファイルストレージスペースだ。スポンサードエディションでは20Mバイト、スタンダードエディションでは40Mバイトのストレージが提供される。また、このCyberDriveに接続した際に、ThinkFree Officeのバージョンなどがチェックされ、常に最新のバージョンをダウンロードして自動インストール



画面1
ThinkFree Officeは、ワープロ、表計算、プレゼンテーション作成の3つのアプリケーションから構成されている。



画面2
広告が表示されるスポンサード版のThinkFree Office英語版（画面はWindows用のもの）

できるようになっている。

今回は、RPM版のThinkFree Office Ver.1.5日本語版の評価バージョンを使った。評価バージョンであるため、配布されるバージョンなどとは一部相違があるかもしれない。また、動作環境としてはKondara MNU/Linux 1.1を使用した。

ThinkFree Officeの構成

ThinkFree Officeは、

①ThinkFree Folders（画面3）
②ThinkFree Write（画面4）
③ThinkFree Calc（画面5）
④ThinkFree Show（画面6）

の4つのプログラムで構成されている。ThinkFree Foldersは、Windowsでいうエクスプローラに相当し、ローカルホストのディレクトリのツリー表示とファイルの一覧の表示のほか、CyberDrive（前述のファイルストレージ）の表示も行うことができる。また、ThinkFree Officeのアプリケーションを起動するランチャとしての機能もある。

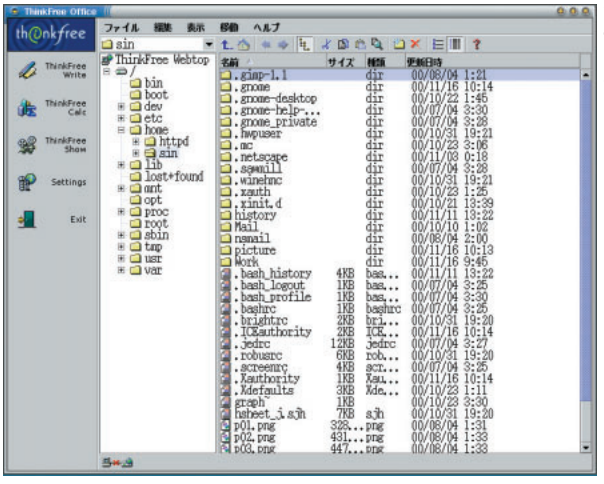
ThinkFree Writeはワードプロセッサで、Microsoft Wordファイルの読み書きにも対応している。また、HTMLファイルの編集も可能だ。

ThinkFree Calcは表計算ソフトで、グラフなどの作成も可能だ。Microsoft Excelのファイルの読み書きができる。

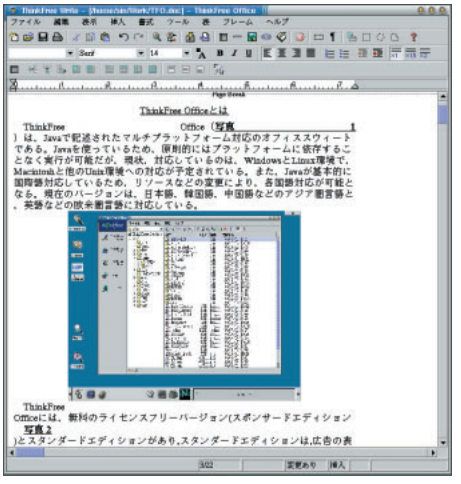
ThinkFree Showは、プレゼンテーション資料作成ソフトで、MicrosoftのPowerPointに相当するプログラムだ。やはりMicrosoft PowerPointのファイルの読み書きが可能になっている。

このほか、イメージビューアやThinkFree Write、Calc、Showのファイルビューア（ファイルを編集することなく表示だけを行う）、スペルチェッカ（各ソフト共有）、クリップアートなどが含まれている。

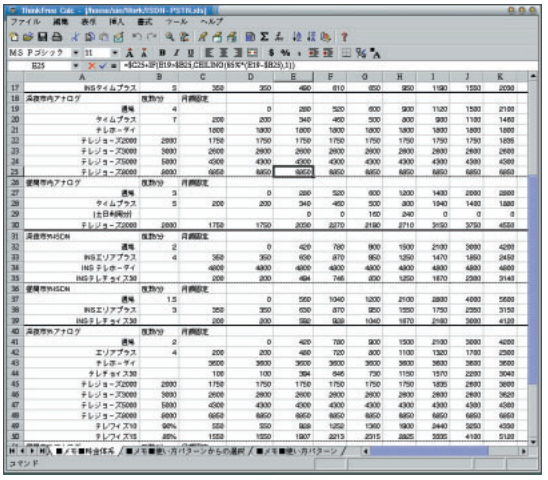
前述のようにThinkFree Officeは Javaで記述されているが、Java2にはまだ対応していない。Linux環境で推奨されているJavaは、IBMのJavaランタイム（IBMJava118-JRE-1.1.8-1.0）である。これは、Java 1.18相当のものだ。動作推奨がこれになっているので、利用時にはこれを使うほうが無難だろう。



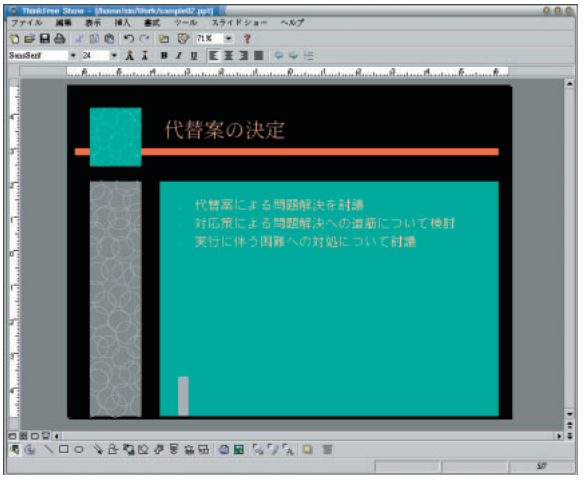
画面3
ThinkFree Officeのランチャ兼、ファイルブラウザであるThinkFree Folders。CyberDriveへのアクセス機能も持つ。



画面4
ワープロソフトであるThinkFree Write。図版や表の挿入など、HTMLファイルなみの表現力を持つ。



画面5
表計算ソフトThinkFree Calc。基本的な表計算機能を持つが、現在のものはβ版だという。



画面6
プレゼンテーション作成ソフトThinkFree Show。スライドショーを行うこともできる。

今回の評価用バージョンはRPMファイルとなっており、インストールは非常に簡単である。ただし、前述のようにIBMのJava 1.18を前提にしているため、これに依存したRPMも一緒にインストールする必要がある。

ThinkFree Officeの多くのファイルは、/usr/share/ThinkFree_Office内に置かれ、起動用のプログラム（tfoというリンクファイル。実体は同じディレクトリにあるTFOCmd.exec）は、/usr/binに配置される。

また、インストールと同時にGNOME、KDEのメニューにThinkFree Officeの項目が追加されるようになっている。

RPMファイルでなく、インターネットからダウンロードしてインストールした場合、初回にダウンロードするコンポーネントを指定するが、以後は必要に応じてインターネットからダウンロードするようになる。ただし、そのためにはインターネットに接続している必要があるため、ダイヤルアップ環境だと転送速度も含めて少し使いづらいのではないかと思われる。

ThinkFree OfficeのGUI

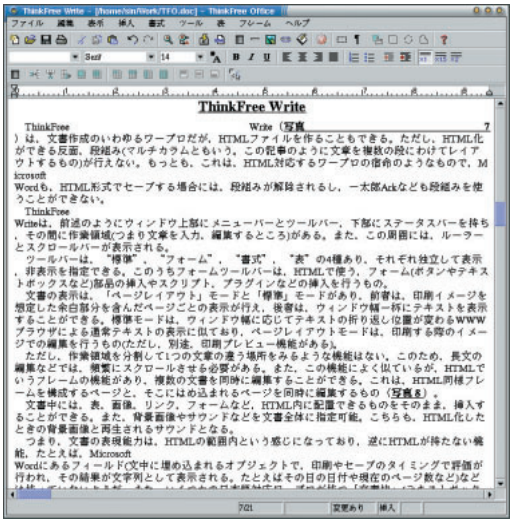
ThinkFree Officeの3つのアプリケーション（Write、Calc、Show）は、メインウィンドウ上部にメニューバー、ツールバーを持ち、下にステータスバーを持つ構造となっている。簡単にいうと、Microsoft WordやExcelなどと非常によく似た構造といってもよい。

各アプリケーションとも、ファイル1つに対してウィンドウ1つが作られ、同じソフトのウィンドウは複数開くことができる（SDI）。1つのウィンドウ内に複数の文書を開くような構造（MDI）にはなっていない。

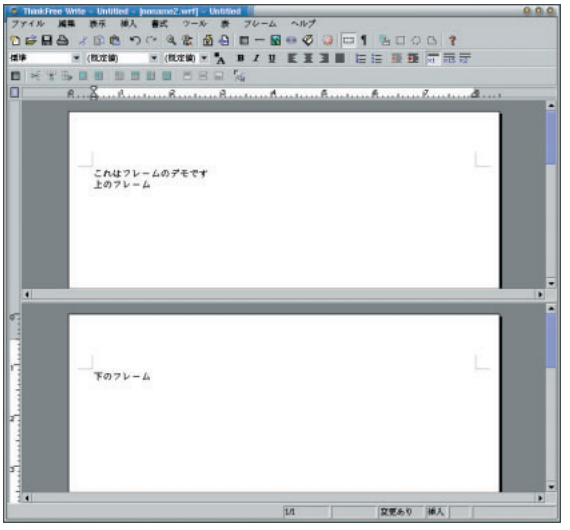
ただし、ツールバーについてはそのオン／オフが指定できるのみで、Microsoft Officeなどのようにツールバー自体を動かして位置を変えたり、フローティング状態で使うといった使い方はできない。また、ウィンドウの横幅よりも長いツールバーは、はみ出した部分が表示されないため、ツールバーを使うのであれば、ウィンドウサイズをある程度の大きさにしておかなければならない。なお、ツールバーのアイコンは、Microsoft Officeのものに似ており、Microsoft Officeを使ったことのあるユーザーなら、それほど迷うことはないだろう。さらに、各ボタンにマウスポインタを重ねるとツールヘルプ（ツールティップ）によりボタン名が表示される。

ダイアログボックスについては、Microsoft Officeと似ているわけではなく、ファイルオープンなどのダイアログボックスも独自のものである。また、複数のダイアログボックスを開く際に、あとから表示されるダイアログボックスが下になってしまうことがあった。ダイアログボックスやメインウィンドウも自由に動かすことができるので、作業ができないこともないのだが、もう1つダイアログボックスが出ることを知らないと、プログラムがハングアップしてしまったかのように勘違いしてしまうことがある。これは、ダイアログボックス同士のZ軸上の順序が違っているためだと思われる。このような場合には、モーダルなウィンドウとして、親ウィンドウや親になるダイアログボックスの操作を許さないようにしておくほうが、混乱がなく親切ではないかと思われる。

そのほか、マウスのダブルクリックが効きにくい場面がいくつかある。フォルダを開くつもりで、アイコンをダブルクリックしても、フォルダ名の編集になってしまうなど、ダブルクリックとして受け付けられないことがある。テストに使ったマシンは、Celeron



画面7
ThinkFree Writeは、HTMLの表現機能をベースにしたワードプロセッサだ。



画面8
ThinkFree Writeは、HTMLのフレームに対応しており、複数のファイルをフレームを使って1つのページに表示することができる。

400MHzと決して最高速のマシンとはいえないが、それほど遅い部類にも入らないものだ。

もう一点気になったのは、スクロールバーがマウスのホイールに対応していない点だ。一度ホイール付きのマウスに慣れてしまうと、スクロールしたいときに無意識的にホイールを動かしてしまうため、使えないとかなりストレスを感じる。メニューやツールバーなどの操作がまったく違っていても簡単に慣れることができるが、ホイール操作はなかなか抜け出すことができない。Microsoft Officeからの移行ということを考えるなら、この機能の実現が望まれる。

ThinkFree Write

ThinkFree Write（**画面7**）は、文書作成のいわゆるワープロだが、HTMLファイルを作ることもできる。ただし、HTML化ができる反面、段組み（マルチカラムともいう。この記事のように文章を複数の段に分けてレイアウトするもの）が行えない。もっとも、これはHTMLに対応するワープロの宿命のようなもので、Microsoft Wordも、HTML形式でセーブする場合には段組みが解除されるし、ジャストシステムの一太郎Arkなども段組みを使うことができない。

ThinkFree Writeは、前述のようにウィンドウ上部にメニューバーとツールバー、下部にステータスバーを持ち、その間に作業領域（つまり文章を入力、編集する領域）がある。また、この周囲にはルーラーとスクロールバーが表示される。

ツールバーは、「標準」「フォーム」「書式」「表」の4種があり、それぞれ独立して表示／非表示を指定できる。このうちフォームツールバーは、HTMLで使うフォーム部品（ボタンやテキストボックスなど）やスクリプト、プラグインなどの挿入を行うものだ。

文書の表示は、「ページレイアウト」モードと「標準」モードがあり、前者は、印刷イメージを想定した余白部分を含んだページごとの表示が行え、後者は、ウィンドウ幅一杯にテキストを表示することができる。標準モードは、ウィンドウ幅に応じてテキストの折り返し位置が変わるWebブラウザによる通常テキストの表示に似ており、ページレイアウトモードは、印刷する際のイメージでの編集を行うものだ（別途、印刷プレビュー機能もある）。

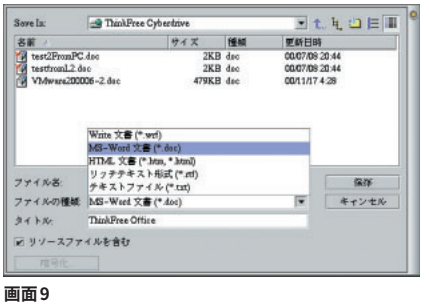
なお、作業領域を分割して1つの文章の違う場所を表示させるような機能はない。このため、長文の編集などでは頻繁にスクロールさせる必要があるが、この機能によく似ている、HTMLでいうフレームの機能があり、複数の文書を同時に編集することができる。これは、HTML同様フレームを構成するページと、そこにはめ込まれるページを同時に編集するものだ（**画面8**）。

文書中には、表、画像、リンク、フォームなど、HTML内に配置できるものをそのまま挿入することができる。また、背景画像やサウンドなどを文書全体に指定することも可能だ。これらは、HTML化したときの背景画像と再生されるサウンドとなる。

つまり、文書の表現能力はHTMLの範囲内という感じになっており、逆にHTMLが持たない機能、たとえば、Microsoft Wordにあるフィールド（文中に埋め込まれるオブジェクトで、印刷やセーブのタイミングで評価が行われ、その結果が文字列として表示される。たとえばその日の日付や現在のページ数などを表示させることができる）などは持っていないようだ。また、いくつかの日本語対応ワープロが持つ「文書枠」（テキストボックス）はサポートされない。

HTMLの表現機能に依存し、HTMLファイルを作成可能なものの、HTMLエディタとは多少違うものと思ったほうがよい。もっとも、ほかのHTML出力が可能なワープロも似たようなもので、ワープロとしての機能を優先するか、HTML出力を優先するかの違いであろう。ただし現状において、マルチプラットフォーム間でなんとか利用可能な書式付きのテキスト形式はHTMLぐらいしかないため、文書交換用の1形式と思えばいいだろう。

ThinkFree Writeで読み書きできるファイル形式は、オリジナルの文書形式であるWRF形式のほか、Microsoft WordのDOC形式、HTML、RTF（Rich Text Format。マイクロソフトの書式付きテキストファイル形式）、テキストファイルの5種（**画面9**）となっているほか、URLを直接指定してHTMLファイルを読み込むこともできる（**画面10**）。これらのファイルは、読み込んだ時点でThinkFree Writeが扱う形式に変換される。たとえば、DOC形式で段組みが設定されていればこれを解除し、フィールドなどは表現文字列に変換する。ただし、この時点で表示はできないものの、HTML化す



画面9
ThinkFree Writeは、Microsoft Wordの文書ファイルであるDOC形式の読み書きに対応しているほか、RTFファイルの読み書きも可能だ。

るときには残せる属性（たとえば、フォント指定）などはそのまま残る。

また、文書を編集してセーブする際には、欠落してしまう属性や要素についての警告が表示される（**画面11**）。たとえば、DOC形式でセーブしようとしたときに、フォーム要素や水平線などは欠落することになる。

実際、Microsoft Word 2000（SR1）で作成した文書を読み込んでみたところ、フィールドの変換におかしなところがあるものの、だいたいは変換できた。また、文字列という点でみれば、基本的にはそのまま変換される。

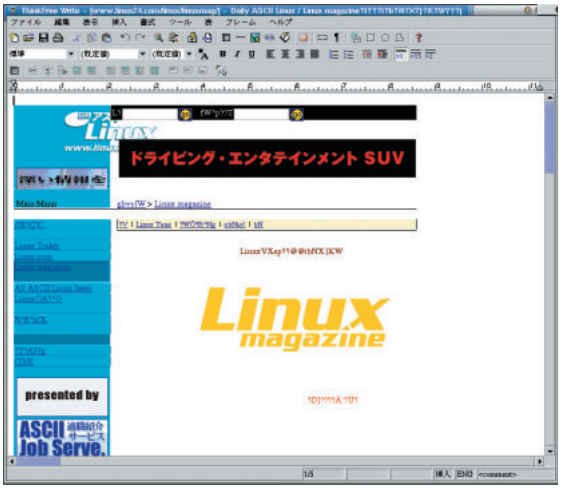
ThinkFree Officeでは、従来のワープロやHTMLでの表示と違って、画像の左右に文字を回り込ませることはできない。このあたりはちょっと物足りなさを感じる部分だ。HTMLでは、表や画像については左右に回り込みが自動で行われ、「<BR clear=xxx>」でそれを制御するか、CSSで制御するかのどちらかが可能なのだが、ThinkFree Writeは、HTMLソースの表示は可能なものの、HTMLやCSSを直接編集できるわけではないので、HTMLとしてセーブしたのち、手作業で作業する必要がある。

実際には、表や画像の挿入位置で文字列は切断され、自動的に「<P>」タグが挿入される。

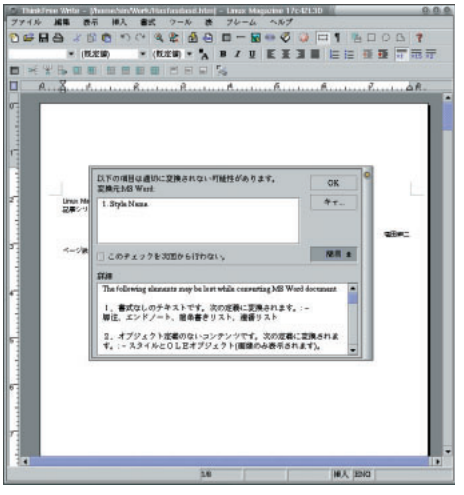
また、今回評価したバージョンでは、英文のワードラップが有効なままで、日本語の文章中にスペースを含む英単語があると、その位置で次の行へ送られてしまう（**画面12**）。この原稿のように英単語をいくつも含む文章では、かなり頻繁に改行が行われてしまう。これは、日本語（とういかアジア圏のスペースを持たない言語）の編集という点からみると、かなり問題だと思う。少なくとも、会社などで他人に見せる文書を印刷して作るには、このままでは不向きかもしれない。

ThinkFree Calc

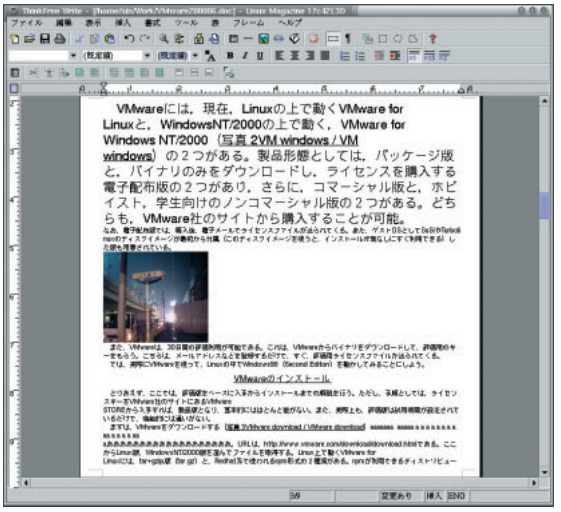
ThinkFree Calc（**画面13**）は、やはりMicrosoft Excelに使い勝手の似た表計算ソフトだ。Microsoft Excel同様、1つのファイルに複数のワークシ



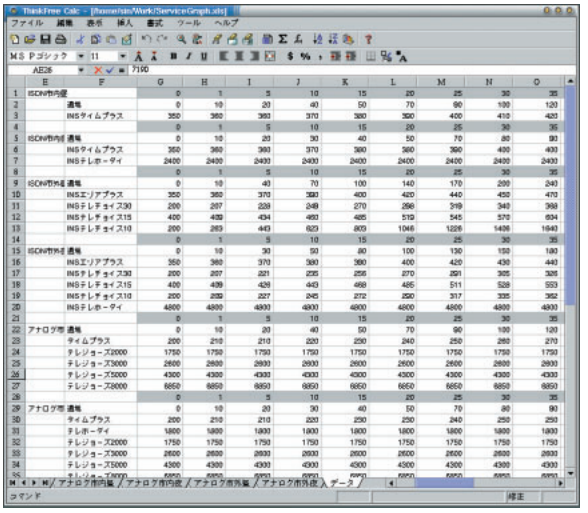
画面10
HTMLが編集可能なThinkFree Writeは、URLを指定してHTMLファイルを直接読み込み可能だ。また、FTPを使いファイルを転送することもできる。



画面11
ファイルの読み書きの時に欠落する書式などがあれば、その内容が警告として表示される。



画面12
ThinkFree Officeの現在のバージョンは、図版に対してテキストの回り込みができないほか、英文のワードラップが有効な状態になってしまっている。



画面13
表計算ソフトThinkFree Calc。やはりMicrosoft Excelのファイルの読み書きが可能。

ートやグラフシートを入れて扱うことができる。なお、ThinkFreeジャパンのサイトによると、ThinkFree Calcはまだβバージョンであるという。実際、Linux上の評価バージョンでは、グラフの作成は行えなかった。このため、このソフトの仕様や使い勝手などは、今後、大きく変わる可能性がある。

また、英語版のスポンサードエディションをWindowsで動かしてみたが、起動前にβ版でグラフ機能にバグがあるとの表示が行われたが、こちらはなんとかグラフの作成が行えた。このバージョンでは、Microsoft Excel同様、グラフをワークシート内に配置することも、独立したシートとして作ることもできる（**画面14**）。

関数は310個備えており、Microsoft Excel 2000の日本語版で定義されている、約330に比べると多少不足しているが、Microsoft Excel 2000には日本語関連の関数や、DLL呼び出しのための関数などもあるため、実際には問題になることはないと思われる。

ただし、サポートされていない機能もあり、たとえば条件付き書式（セルの値に対する条件に応じて書式を設定する機能）などはサポートされていないようだ。

ユーザーインターフェイスでは、選択範囲のドラッグ&ドロップによる移動や、ハンドルを掴んで、セル値のコピーや連続値の入力などもサポートされていない。ただし、画面上には選択範囲右下にハンドルが表示されるので、実装する予定はあるのかもしれない。

全体的な印象からすると、表計算ソフトの基本的な機能のみ実装されている感じで、メニュー体系などはMicrosoft Excelをお手本しており、機能的にはマクロのない初期バージョンのMicrosoft Excelと似ている感じである。

現状ではまだβ版であるため、今後機能が追加されていく可能性もある。

なお、このThinkFree Calcは、ネイティブのCLFファイルとMicrosoft ExcelのXLSファイルの読み書きが可能だ。Microsoft Excelにあって、ThinkFree Calcにない機能が使われていた場合には、読み込んだ時点で警告が表示されるようになっている。

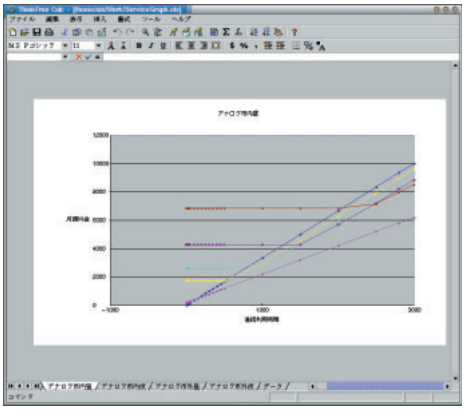
Microsoft Excel 2000で作成したファイルを読み込んでみたが、日本語を含め、セル内の数値、文字列、数式については、ほぼ問題なく読み込むことができた。あまり凝ったMicrosoft Excelファイルでなければ、少なくともデータを移す用途には使うことができるだろう。

ThinkFree Show

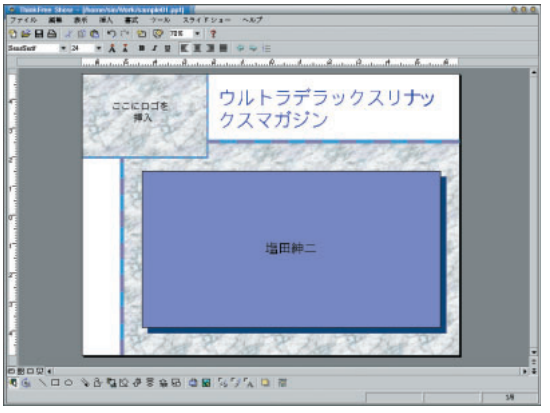
ThinkFree Show（**画面15**）は、プレゼンテーション資料作成ソフトで、簡単にいえば、スライドやOHPなどを作るためのものだ。Microsoft PowerPointのプレゼンテーションファイル（PPTファイル）と、ThinkFree Showネイティブの SHF形式ファイルの読み書きが可能なほか、プレゼンテーションのテンプレートとして、Microsoft PowerPointのテンプレートファイル（POTファイル）の読み込みが可能だ。また、HTMLファイルへの出力も行える。

ただしHTML出力は、評価版では少し動作がおかしかった。複数ページのプレゼンテーションは、どうやらサブディレクトリを作って、その下に各ページごとのHTMLと、そこに埋め込まれる画像ファイルを生成するようなのだが、各ファイルをセーブするときにパス区切り文字に「\」が指定され、すべてファイルとして作られてしまう。これとは別に、中身が空のサブディレクトリが作られるため、おそらく、パスの区切りがLinuxであるのにも関わらず、逆スラッシュ（円マーク）になったままなのではないかと思われる。

プレゼンテーションは複数のページから構成され、マスターと呼ばれるページですべてのページに共通な背景などの設定を行う（**画面16**）。また、そのそれぞれに対して、タイトルページや図版入りのページなどのひな形を使



画面14
ThinkFree Calcは、グラフ作成にも対応している。これは、Microsoft Excel 2000で作成したファイルシートを読み込んだもの。



画面15
ThinkFree Showは、プレゼンテーション作成ソフトだ。Microsoft PowerPointファイルの読み込みが可能。

って新規ページを作成する。このひな形は、テキストボックスや画像枠などがあらかじめ配置されたもので、白紙のひな形を使って、これらの要素を自由に配置することも可能になっている。

作成したプレゼンテーションは、スライドショーを行うことができる（**画面17**）。ただし、このThinkFree Showでは、次のスライドへ切り替えるときのアニメーション効果を付けることや、ページ内の項目を順次表示させるような、スライドショー向けの機能は持っていない。作成したページを前後に進めて表示させることは可能だ。

ページの背景などを指定するテンプレートファイルは、今回の評価版では読み込むことができなかった（Windows上のスポンサード版ではダイアログが表示され正しく動くようである）。前記のHTMLファイルのセーブの件を含め、まだ未完成な部分があるようだ。

ただし、Power Pointで作成したファイルのほうはちゃんと読み込めるようである。

雑感

Javaを使い、インターネット上でのファイルストレージとの組み合せで、機能を提供するというコンセプトは、たしかに便利なものがある。ファイルそのものを持ち運ぶことなく、まったく別の環境でインターネットにアクセスして作業の続きを行えるなどのメリットがあるし、サーバに接続することで、必要なファイルやアップデートされたファイルを随時持ってくることができるからだ。

しかし、日本語化という点でみるともう少し時間が必要だという感じがする。かつて米国のソフトハウスが国内へ進出してきた頃の日本語化された英語アプリケーションを見るようだ。JavaやLinuxが多国語対応したといっても、ワープロなどで使う組版の規則への対応はアプリケーションが対応しなければならない部分が多い。そのために、こうしたオフィススイートの移植はかなり大変な作業ではあるが、Windowsアプリケーションを移植するソフトハウスは、これをクリアしてきたのである。このあたりの改善を今後のバージョンアップなどで期待したいところだ。

フォントの表示について

今回のThinkFree OfficeはJavaを使っているため、表示フォントなどはJavaの設定に依存している。推奨されるIBMのJava VMのデフォルトのフォント設定では、serifフォントのみ指定されていて、そのほかのフォントはserifと同じ設定を使うようになってた。このため、ThinkFree Office側でフォントを指定しても、見かけ上はまったくフォントが変わらないといった問題があった。この問題を解決するには、JavaVMのフォント設定を変更する必要がある。ここでは、その手順を簡単に解説しておく。

```
$ rpm -ql IBMJava118-JRE | more
```

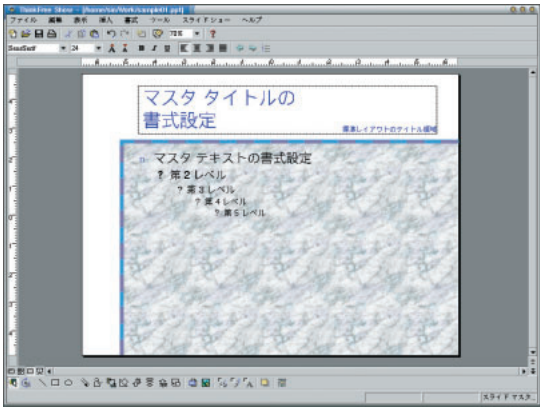
としてインストールされたファイルの一覧を得て、Java VMの場所を調べる。今回の場合には、「/usr/jre118」であった。

ここにある「/usr/jre118/lib」にフォントの設定を行うファイルがある。「font.properties.ja」というのが日本語のフォント設定ファイルである。

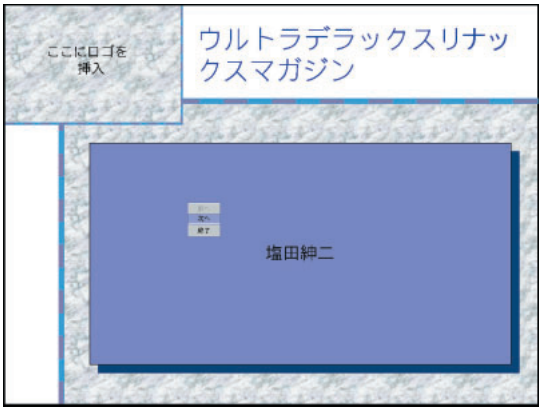
このファイルの先頭部分にある、

```
serif.0=-sony-fixed-medium-r-
normal-*-*-%d-*-*-*-*-iso8859-*
```

から4行で、serifフォントとX11のフォントを対応させている。また、このあとに、「alias.dialog=serif」からはじまる4行があり、これはほかのdialog、dialoginput、monospaced、sansserifをserifフォントの別名として定義している。また、さらにうしろの部分に、「fontcharset.serif.0=sun.io.CharToByteISO8859_1」から始まる4行があり、



画面16
ThinkFree Showでは、背景や文字スタイルなどの各ページ共通の基本設定をマスターと呼ばれるページで設定しておく。



画面17
ThinkFree Showでは、作成したプレゼンテーションをスライドショーとして表示することができる。

ここでは各フォントセット（この場合はserif）の文字セットの定義が行われている。

つまり、ほかのフォントを使えるようにするには、最初にJavaのフォント名とX11のフォント名の対応を定義し、aliasを取りやめ、文字セットを定義すればよい。

たとえば、sansserifを定義する場合、「alias.sansserif=serif」という行を削除し、

```
sansserif.0=……
      :
sansserif.3=……
```

というフォント定義と、

```
fontcharset.sansserif.0=……
      :
fontcharset.sansserif.3=……
```

という文字セット定義を追加すればいいわけである。なお、標準で付属するfont.propatiesはAIX用のものらしく、実際にはないフォントを指定したようなので、システムにインストールしてあるdynalabのフォントを使って、serifの定義を書き換え、さらにsansserifの定義を追加したのが**リスト1**である。このファイルでは、ASCIIコード部分に使われるフォントをさらに変更している。こうすると英数部分できれいなフォントが使えるのだが、欠点として半角カナが非常にきたなくなってしまう。半角カナを使わなければ特に問題はない。

この作業を行うだけで、ThinkFree Officeでちゃんと複数のフォントが表示できるようになる。なお、今回の記事で使った画面写真は、この作業を行ったうえで撮影したもので、font.propatiesを編集していない他のシステムでは、表示が異なる場合がある。また、インストールしてあるフォントによっても設定を変える必要があるので注意してほしい。

▼リスト1　font.properties.ja（抜粋。枠部分が変更点）

```
# Serif font definition
#
#serif.0=-sony-fixed-medium-r-normal-*-*-%d-*-*-*-*-iso8859-*
serif.0=-adobe-times-medium-r-normal-*-*-%d-*-*-*-*-iso8859-*
#serif.1=-dt-interface user-medium-r-normal-*-*-%d-*-*-*-*-JISX0208.1983-0
#serif.2=-dt-interface user-medium-r-normal-*-*-%d-*-*-*-*-JISX0201.1976-0
serif.1=-dynalab-mincho-medium-r-normal-*-*-%d-*-*-*-*-JISX0208.1983-0
serif.2=-dynalab-mincho-medium-r-normal-*-*-%d-*-*-*-*-JISX0201.1976-0
serif.3=-adobe-symbol-medium-r-normal--*-%d-*-*-p-*-adobe-fontspecific
serif.badsize=0-11:12,13-16:17,18-22:17,24-100:23

sansserif.0=-adobe-helvetica-medium-r-normal-*-*-%d-*-*-*-*-iso8859-*
sansserif.1=-dynalab-gothic-medium-r-normal-*-*-%d-*-*-*-*-JISX0208.1983-0
sansserif.2=-dynalab-gothic-medium-r-normal-*-*-%d-*-*-*-*-JISX0201.1976-0
sansserif.3=-adobe-symbol-medium-r-normal--*-%d-*-*-p-*-adobe-fontspecific

alias.dialog=serif
alias.dialoginput=serif
alias.monospaced=serif
#alias.sansserif=serif
        :
      途中省略
        :
fontcharset.serif.0=sun.io.CharToByteISO8859_1
fontcharset.serif.1=sun.awt.motif.CharToByteX11JIS0208
fontcharset.serif.2=sun.awt.motif.CharToByteX11JIS0201
fontcharset.serif.3=sun.awt.CharToByteSymbol

fontcharset.sansserif.0=sun.io.CharToByteISO8859_1
fontcharset.sansserif.1=sun.awt.motif.CharToByteX11JIS0208
fontcharset.sansserif.2=sun.awt.motif.CharToByteX11JIS0201
fontcharset.sansserif.3=sun.awt.CharToByteSymbol
        :
      以下省略
```

隠喩としてのコンピュータ

地域通貨LETSを立ち上げるには

文：豊福 剛
Text：Tsuyoshi Toyofuku

地域通貨は、まがりなりにも「通貨」と呼ばれるくらいだから、何か大袈裟なものを想像してしまいそうだが、規模や運用方式によってさまざまな種類があるらしい。西部忠『＜地域＞通貨LETS』（『可能なるコミュニズム』収録・太田出版）によると、LETSは、中央委員会などが集中して紙幣を発行するのではなく、参加者に対して口座を開設し、取引ごとに帳簿に記帳する方式で運営される。参加者の口座はすべてゼロから開始される。参加者は売りたいモノやサービスを目録に登録する。また買いたいモノやサービスを目録に登録することもできる。LETSでの取引は、売り手の帳簿の貸方に黒字を記録し、買い手の帳簿の借方に赤字を記録することで処理される。参加者は、他の参加者の口座残高や取引実績について照会することができる。口座残高に対して利子は課されないし、支払われない。

LETSのように帳簿式で無利子のものであれば、ちょっとした会員制BBSの応用システムとして実装することも可能だろう。BBSを応用して、売買したいモノやサービスを登録・検索できる仕組みを作り、そのバックエンドのデータベースに、参加者ごとの帳簿を保持すればよい。Aさんがリクエストしたサービスを Bさんが提供した場合、その取引トランザクションに対する処理は、Aさんの帳簿レコードにマイナスの値を記録し、Bさんの帳簿レコードにプラスの値を記録することとして表現される。これは一見、銀行口座への振込と似ている。

お金がなくても買える不思議

LETSの面白いところは、Aさんの口座残高がマイナス、すなわち赤字であっても、支払いが可能である点だ。国民通貨では、手持ちの現金がないときや銀行口座の残高がゼロのときには、モノやサービスを買うことはできない。お金やマネーは、それを持っていなければ、何も買えないのが本質であるとすれば、残高がマイナスであっても、買うことができるLETSは、通貨と呼ばれるものの、お金やマネーとはだいぶ異なる性質の「通貨」であるように思われるだろう。

赤字でも買えるというLETSの性質は、「ツケ」と呼ばれる後払いの行為とも違っている。ツケとは、モノやサービスを提供した人に対して、後日支払うことを前提に、支払いを延期してもらうことであり、売り手と買い手は債権

と債務の関係になる。債務者は債権者に借りを返さなければならない。

ところがLETSでは、残高が赤字である参加者は、誰かにモノやサービスを売って、赤字を減らしていくだけでよい。債権者に対して獲得したLETSで直接的に返済する必要はないのだ。

このことと関連して興味深いのは、LETSには利子がない点である。国民通貨では、もし手持ちの現金がないとしたら、誰かから借金をするしかない。通常、借りた金には利子が付く。借りている期間が長くなればなるだけ、利子は増えていく。しかし、LETSには利子がないので、LETSをどれだけ所有していても、価値は増えない。LETSは使わなければ意味がないのだ。

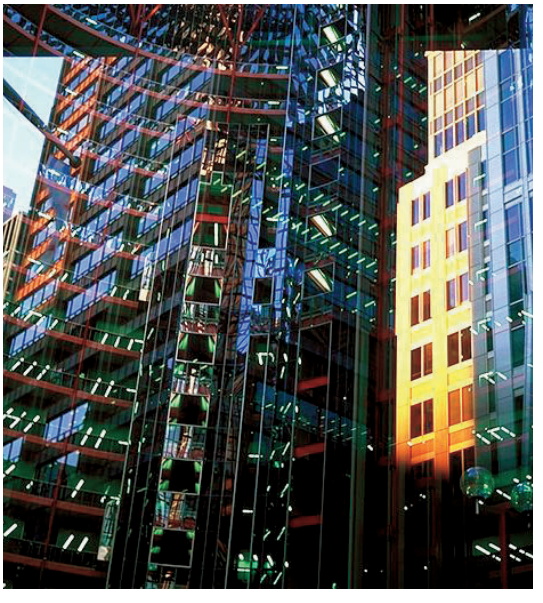
このように、赤字でもモノやサービスが買えるのであれば、モノやサービスを売ることをせず、一方的に買うだけで、どんどん赤字を膨らませる参加者が出てくることが予想される。しかし、LETSは参加者の口座残高や取引実績を照会することができる。そのため、ある限度額以上の赤字を計上している参加者に対しては、取引を制限するといった運用で対処することが可能である。

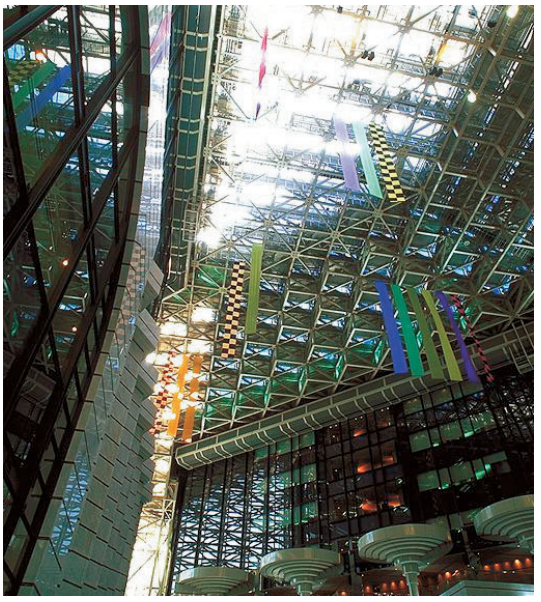
非匿名性とホスト集中管理

LETSでの取引は、すべてホストのデータベースに記録される。この点は、クレジットカードによる支払で、その内容がクレジットカード会社に知られてしまうのと似ている。クレジットカードと同様、LETSでの取引は、現金による取引と違って、匿名性はないのだ。

LETSの非匿名性は、大きな問題ではある。口座残高や取引実績だけでなく、すべての取引内容が照会できるような極端な運用方式も想定できるだろう。その場合には、誰が誰に何をいくらで売買したのか、すべてわかってしまうことになる。

このような非匿名性は、たとえばオークションサイトで参加者の出品履歴や落札履歴が照会できるのに、似たところがある。もちろん、オークションサイトの参加者は、ハンドル名で参加しているわけだが、売買が成立した場合には、売り手と買い手の双方が、相手の名前、住所、電話番号などを知ることになり、取引によって得られた個人情報を悪用される危険がないとはいえない。





LETSにおけるこのような非匿名性の扱いについては、どのような解を求めたらいいのだろうか。

ひとつの解としては、帳簿が保持されているホストの運用主体や監視主体だけが、個々の取引内容にアクセスできるようにして、LETSの参加者は他の参加者の口座残高と取引実績（取引の合計）だけしか照会できないようにする方法が考えられる。

この解は現実的なものに思えるが、同時に、LETSは必然的にホスト集中型のシステムとしてしか実装できないのではないか、という疑問もわいてくる。

ゼロサム経済としてのLETS

この問題は、LETSが（地域的な）コミュニティで流通することを前提としたものであり、しかも、コミュニティ内での取引の合計がゼロサムになることを初めから条件づけられていることと、密接に関係していると思われる。全参加者の口座残高の合計は、常にゼロでなければならず、そうでなければ、どこかで不正な取引が発生していることになるからだ。

ゼロサムを保証するためには、ホスト集中型になってしまうのだろうか。そうではなく、参加者が個別に帳簿を管理する、分散型のシステムは構想できないだろうか。

参加者が個別に帳簿を管理する方式の場合、帳簿の数字を不正に操作する可能性があり、仮に不正操作が行われた場合に、それを検出することが困難である点に問題がある。ならば、自分の帳簿はもちろん、他人の帳簿についても、不正操作ができないような仕組みがあれば、分散型のシステムであっても、ゼロサムを保証することができるのではないだろうか。

おそらくは、そのような技術として公開鍵暗号があるのだろう。とはいえ、公開鍵暗号の応用としては、電子マネーなどがあるわけだが、LETSのような地域通貨と電子マネーがどのような関係にあるのか。両者は同じものになるのか、それとも違うのか。いろいろ考えてみたが、いまのところ、よくわからない。たとえば帳簿の改ざんは検証できるだろうし、取引に対してデジタル署名を適用することもできるだろう。ただし、それによって、LETSの流通するコミュニティにおけるゼロサムが保証されることになるのかが、よくわからないのだ。

実際の取引は、どのようなものなのか？

公開鍵暗号技術については、またあらためて検討するとして、そもそもLETSで売買できそうなモノやサービスには、具体的に、どのようなものがあるのか気になるところである。Webで公開されているLETSを探したところ、gmLETSystem（http://www.gmlets.u-net.com/letsdir/letsdir.html）というのがあった。gmLETSytem（gmはGreater Manchesterの略）は、イギリスのマンチェスターを中心に実践されているLETSで、参加者がどのようなサービスやモノを売買しているのか、その雰囲気がわかるサンプルが公開されている。

gmLETSystemに登録された利用者には、gmLETS IDが発行される。そして売買したいモノやサービスを目録（ディレクトリ）に登録する際に、このIDと電子メールアドレス、そして実際に売りたい（offer）あるいは買いたい（request）サービスやモノをフォームに入力する。

目録に登録するサービスを分類するために、3桁のコードが設定されていて、**表1**のような内容になっている。

一方、国内の地域通貨運動については、Miguel Yasuyuki Hirotaさんのホームページ（http://www3.plala.or.jp/mig/japan-jp.html）が、わかりやすくまとめられている。ここで紹介されている地域通貨のひとつである「レインボーリンク」（http://www.geocities.co.jp/WallStreet-Bull/1964/）では、目録に登録する際のサービス分類として、**表2**のような項目が設定されている。

こうして見てみると、モノよりはやはりサービスが中心になるようだ。フェミニズムなどの文脈では、家事労働のようにお金が支払われない労働のことを、シャドウワークやアンペイド・ワークと呼ぶのだが、地域通貨を媒介にした取引の多くは、こうしたシャドウワークを中心としたものであるようだ。ただし、地域貨幣の適用範囲は、そうしたシャドウワークにのみ限定されるわけではない。実際、地域通貨の中には、国民通貨とリンクしているものもあり、潜在的には国民通貨と競合する可能性も秘めている。そういえば、地域振興券というものがばらまかれたことがあったが、あれもまた地域通貨的な性質を持っていたといえなくもないだろう。ただし、国家が発行する地域通貨というのは、奇妙な存在に思える。

000	時間労働
100	手作業
200	食品
300	工芸品
400	不動産
500	機材・施設
600	輸送関係
700	中古品
800	イベント
900	コマーシャル

表1　gmLETSystemの財・サービス分類

1	家事、家庭とコミュニティ支援
2	輸送
3	事務、オフィスサービス、電算機処理
4	スポーツと外交
5	修理、補修、建築、建設
6	間貸し、宿泊、旅行
7	絵画、工芸
8	庭仕事、園芸
9	被服、織物
10	教授、翻訳
11	料理配達と食べ物
12	新品・中古品販売及び貸し出し
13	健康、セラピー
14	精神的な事
15	上演、創造的芸術
16	雑務

表2　レインボーリンクにおける財・サービス分類

Profile

とよふく　つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

オープンソース
・
アクティビズム

社会にITを、そしてオープンソースを

文：安田幸弘
Text : Yukihiro Yasuda

警察庁版カーニボー

以前、アメリカでFBIのメール盗聴プログラム、カーニボー（http://www.fbi.gov/programs/carnivore/carnivore.htm）をオープンソースにしろ、という声があるという話を書いたのだけれど、結局オープンソースの要求はボツになったようだ。その代わりに、カーニボーに関するドキュメントが段階的に公表されている。

アメリカの人権団体のEPIC（Electronic Privacy Information Center、http://www.epic.org/）は、Security Forcusが分析した墨塗りだらけのドキュメントに関する情報をWebで公開しはじめたのだが、早くもカーニボーに対しては相当いろいろな問題点が指摘されている。特に、FBIは対象となる情報以外は一切収集しない、と言っていたのに、実はカーニボーにはフィルタされないパケットをそのまま記録する機能がある。つまり令状に書かれていないユーザー宛のトラフィックもそっくりそのまま記録できるわけだ。また、前のバージョンには関係のないパケットまで記録してしまう「バグ」があったという。さらに、非公開になったソースコードは第三者が監査することになったのだが、この第三者というのが政権とのコネクションのある団体らしい。

いずれにせよ、カーニボーの正体が明らかになるにつれて疑念は膨らむ一方で、カーニボーの放棄を求める意見が高まっている。EPICはカーニボー撲滅にはやっぱりソースの公開しかないとして、情報公開法の規定に従って、ソースの公開を求めているという。やはり、ソースに勝るドキュメントはない、ということだけど、墨塗りだらけのソースリストなんてのはイヤだなあ。

ところで、ここまでは海の向うの異国の話。しかし、外国の話だと思っていたら、まさかと思っていたことが本当になってしまいそうな勢いだ。日本でも、アメリカでこれだけ問題になっているカーニボーと同種のシステムが導入されようとしているのである。

当局が「仮のメールボックス」と名付けたこのシステムは、どこが「仮のメールボックス」なんだか、要するにカーニボーと同様のパケット横取りソフトだ。そもそも'99年の盗聴法の成立そのものが、FBIの指金だったらしいのだが、それ以来日本の警察はFBIの筋書きを忠実になぞっている。「仮のメールボックス」をめぐる国会でのやりとりでも、「無関係な通信まで捕捉するのはまずいんじゃないか」という質問に「令状に書かれた範囲で通信を捕捉することになっている」と、アメリカの国会でのFBIの答弁と全く同じだったりするのである。アメリカに言われた通りに導入を決めたとたん、元祖カーニボーが存亡の危機に立たされたというのも皮肉だが、いまさら引っ込めるわけにもいかないのだろう。当然、議員さんはそれに対して「ソースコードを公開しろ」と言ったらしいのだが、ここから先はどうなるんでしょうね。

日本の警察がFBIの言うなりというのもずいぶんな話だけれど、どうせアメリカのマネをするんだったら、「仮のメールボックス」に関する詳細なドキュメントの公開、ソースコードの公開までマネしてほしいものだ。ま、FBIが日本の警察にドキュメントの公開をしろと言ってくれるわけはないか。

IT＝インターネット通販？

ITといえば、日本も「IT先進国」になろうとしてか、IT基本法とやらが成立したそうだ。IT化を推進すること自体は賛成なのだが、条文を読んでみると結局単なる電子商取引のための環境整備でしかないのだから悲しい。確かに、これでIT関連の企業はそれなりに儲かるところがあるだろうし、ITをベースとする企業活動の活性化は重要だ。だけど、日本のIT政策は企業の利益の保護に熱心なあまり、普通の市民の存在感があまりにも薄いのが残念だ。たとえば、IT基本法などでも

知的所有権の保護には積極的でも、情報の公開や共有といった部分についてはほとんど触れられていない。そればかりか、市民のコミュニケーションを盗聴法なんかで規制することばっかり考えているのだから困ったものだ。極端な話、日本のIT政策のなかで、市民というのは単なる財布でしかない。来たるべき高度IT社会とやらの姿が、みんなでパソコン通販で買い物をする社会だというのではさびしいじゃありませんか。

ITというものの革命的なところは、無名の市民を結び付けることで、とんでもないことが可能になるということだろう。ITの奇跡の双璧は、ドット・コム企業とオープンソースだと思う。最近は落ち目だけれど、ドット・コム企業というやつは、つまり街角の名もない小さな店でも、うまくITを使えば巨万の富を得ることができるという奇跡だった。

ぼくが面白いと思うのは、ドット・コムのような営利活動とオープンソースという非営利活動が、ITの世界で確固たる地位を占めているというところだ。つまり、ITの世界では、ドット・コムにばかり目を奪われている日本の官僚は見向きもしない非営利活動の分野が、無視できないプレーヤーだということだ。とはいえ、ITに関わる限り、いかに日本の官僚でも否応なくオープンソースに接近せざるを得ない。特に、IT政策の重要な武器として使われることになったIPv6にしても、日本の政治家や官僚がIPv6と言うとき、それはKAMEプロジェクト（BSD用のIPv6スタックを作る国産プロジェクト、http://www.kame.net/）によるオープンソースの実装そのものを意味している。

ところが、IPv6、あるいはKAMEが民間のオープンソースプロジェクトだと言っても、官僚の目には成果が無料で公開される産学共同プロジェクト、ぐらいにしか見えていないようだ。KAMEの開発に、どれだけ多くの無名のプログラマーや技術者がかかわっていたのか、世界中のオープンソースコミュニティがどんなに貢献したのか、日本の政治家や官僚は理解してほしいものだよね。

みんながコミットすれば社会が変わる

しかし、民間のパワーと言っても営利と非営利、言い換えれば企業活動と市民活動の勢力が重要だなんて、実は何もITだけのものじゃない。欧米の民主主義国家なら、極めて当たり前の常識に属する話なんだと思う。最近になって、日本もようやくNPO（非営利法人）の設立が認められるようになったのだが、あまりその社会的な意味が正しく理解されているとは言えそうもない。

たとえば、先日あるNPOの集まりで、「市民活動を支援すると言って役所が予算をつけてくれたり企業からの寄付があるのは嬉しいけれど、本当に必要なのは行政や企業の情報なんだ」という話を聞いた。欧米のNPOにとって、そうした情報の「蛇口」のような役割を果たしているのがインターネットなのだが、これはオープンソースソフトウェアの世界とまったく同じ構造だ。

IT社会というやつは、情報の公開と共有、そして自由なコミュニケーションが確保されることを前提として、無名の市民が一方の主役になる社会だ。そして、オープンソースソフトウェアってやつは、IT社会での無名の市民の参加がどれだけ現実的なメリットをもたらすのかを知らせるきっかけになるだけの力を持っている。何しろ、日本の国運をかけたIT戦略の要になっているのがオープンソースのKAMEである。KAMEばかりじゃなく、われわれも、もっとオープンソースの開発にコミットし、ユーザーもオープンソースのメリットをもっと積極的に伝える努力をしていけば、それは自分たちの社会を変えていくことにもつながっていくんじゃないだろうかと思うのである。

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

11・10 JPNIC汎用JPドメインの実施を1カ月延期
11・1　S3社名をSonicblueに変更
10・31 MS次期OSのβテストを開始
10・30 AMD DDR対応チップセットとAthlon 1.2GHzを発表
10・30 ローソン1万5000台のLinuxサーバを導入
10・27 MS本社に侵入者
10・26 AT&T分割を発表

さて、そろそろ21世紀なのだが、なんとなく、子供の頃に感じていた21世紀とは違うような気が……。20年、30年前の予想って全然当たりませんね。まあ、この業界、来年を予想するのも難しいのですから。

マイクロソフトに動きあり

裁判も控訴して、ひとまず落ち着いたマイクロソフトだが、本社のコンピュータシステムに侵入されてしまったようだ。これは、QAZ Trojanと呼ばれるウイルスのようなソフトで、電子メールなどで配布され、これを開くと、WindowsのNotepadに置き換わる。その後、システムを検索して、IPアドレスを特定の電子メールアドレスに転送し、侵入口となるというもの。

原因は、従業員が持ち帰ったノートパソコンが自宅で感染、そのままマイクロソフト社内に持ち込まれたのだという。まあ、こういう感染経路もあるんですね。自宅で仕事する人は、これからもう少し気を使わないといけないのかも。マイクロソフトは、被害は小さく何も盗まれていないとしているが、業界トップ企業がこうした侵入を許したという点で、マイクロソフトにとっては損なニュースとなった。

ところで、多くのニュース記事が「マイクロソフトにハッカー侵入」という見出しをつけていたが、本誌の読者の方であれば、「ハッカー」を悪い意味で使わないで欲しいと思う方もいるであろう。でも、難しいよね。今回の場合には、システムを破壊したわけでもないので、「クラッカー」ともいえないし。まあ、悪者とされているマイクロソフトに侵入したのだから、「悪いことをしたわけではない」という見方もできるが……。

AMDはPentium4対策

AMDはついにDDR SDRAM対応のAthlon用チップセットであるAMD-760を発表した。DDR SDRAMは、現在広く使われているSDRAMの転送レートを倍にした高速なメモリ。インテルは当初SDRAMの次は、Rambusメモリ（DRDRAM）だとして、このDDRを無視してきた。しかし、インテルは820チップセットの問題など、Rambusメモリを使うシステムの立ち上げでトラブルを連発し、Rambusメモリ自体もなかなか価格が下がらない状態。このため、市場ではあまり人気がない状態が続いている。そこで、インテル以外のチップセットメーカーは、インテルへの対抗としてDDRを担ぐことになった。Athlonには速いメモリが必要なので、AMDは当初からDDRの採用を決めていたが、ようやく対応のチップセットが登場したわけだ。

また、AMDは同時に1.2GHzのAthlonを発表。また、最高クロックを更新した。インテルは、Athlon対抗のPentium4を11月中に発表する予定だが、これはそれに対する先制攻撃というわけ。特にPentium4は、Rambusでの動作をメインとしており、当分ほかのメモリに対応できない状態。ここで先に市場に製品を投入したいところだ。Pentium4は、クロック周波数を高くしやすく設計したため、同クロックのPentiumIIIよりは若干パフォーマンスが落ちる。しかし、最初から1.5GHzで登場するため、PentiumIIIに負けることはない。

というわけで、また来年から新たにクロック競争が始まるわけで、そうなると2GHzなんかもアッという間に達成されてしまいそう。でも、なんか競争

が激化するたびに、自分のマシンが遅くなっていくような気がするんですけど……。

インテルはCrusoe対策

さて、日本のメーカー3社（ソニー、日立、富士通）が、Crusoe搭載のノートパソコンを出荷した。あせったインテルは早速、低消費電力のCPUを開発するチームを編成したという。この前のIntel Developer Forumでは、「すでにインテルCPUの消費電力はかなり低いし、評価するなら平均使用電力を見て欲しい」と主張していたが、やはりメーカー採用がきいたようである。

日本のメーカーがなぜ、Crusoeを採用したのか？ これは、日本車とアメ車のような違いである。日本人は、たしかにハイスペックが好きだが、デカいエンジン、デカい車体といったアプローチは好きではないのである。SpeedStepはたしかに消費電力が小さくなるものの、デカいエンジンの低速での燃費を向上させたようなものなのだ。

これに対してCrusoeは、最高クロックは高くないかもしれないが、最大速度で動かしてもファンが不要な「小さなエンジン」なのである。

日本語ドメイン取りました？

近頃の話題は、日本語ドメインかもしれない。JPNICは、来年1月から予定していた日本語ドメインを含む汎用JPドメインの実施を1カ月延期した。というのは、この汎用JPドメインの実施とセットで考えていたドメイン登録を行う会社の設立が困難になったため。これは、11月のJPNIC総会でこの件がもめて、継続審議となったからである。しかし、COMドメインなどでは、先に日本語ドメインの受け付けが開始され、こちらはこちらで、サーバにアクセスが集中して大変だったようだ。

この日本語ドメイン、いわゆるDNSの多国語化として現在、検討が行われているが、ドメインの登録自体は先行して行われている。つまり、取ったとしても、すぐには使えないのである。

このDNSの多言語化は、iDNS社が見えないトップレベルドメイン（ゼロレベルドメイン）を使って、サービスを開始したが、従来のDNSとの相互運用性が低いとして、現在、互換性のある方法が検討されているところ。各国語のドメイン名をユニコード（UCS）化して、ASCIIコードにエンコードする方法が有力視されている。つまり、各国語の表現とASCIIコード化された2つの表現を持つことになる。これだと、少なくともすでに設置されている多数のDNSサーバを変更しなくてすむわけだが、問題がないわけでもない。

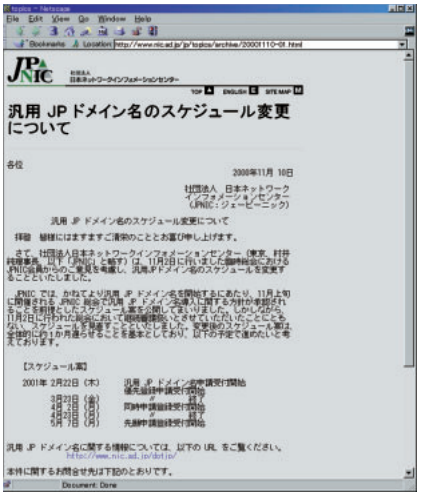
対応には、メールソフトやWebブラウザなどが直接これに対応するか、ライブラリの入れ換えやシステムへのフックを使ってOS側で対応する、Proxyを使うといった3つの方法がある。しかし、クライアント側のアプリケーションは多種多様で、さらに数多くのコピーが動いている。これらが対応するのにはかなり時間がかかる。また、OS側の対応もできないわけではないが、やはり多数のシステムがあり、これも時間が必要。Proxyはいったい誰が設置すべきなのかといった問題がある。

もうひとつは、プロトコルヘッダ内にホスト名を入れてしまっているようなもの、たとえば、HTTP/1.1のようなものへ対応である。HTTP/1.1ではマルチホームサーバ用にHTTPヘッダ内にホスト名を入れるようになっている。当面、HTTP/1.1サーバは、ユニコードとASCIIエンコードされた2つのホスト名を受け付けなければならなくなる。こっちは、アクセス量が多いだけにProxyでの対応が難しく、また、各国語によるドメインの多くは、Webで利用したいがためともいえるので、さすがにWebブラウザは従来のままというわけにもいかないだろう。

さて、インターネットは、こういう混乱状態で21世紀を迎えてしまっていいのでしょうかねぇ。

では、よいお年を。

Network Solutionsの日本語ドメイン検索
http://global.networksolutions.com/purchasing/welcome-j.jhtml;$sessionid$0TFPNVYAAADAHWFI3EHCFGI



汎用JPドメイン名のスケジュール変更について（JPNIC）
http://www.nic.ad.jp/jp/topics/archive/20001110-01.html

日刊アスキーLinux on Linux magazine

http://www.linux24.com/

日刊アスキー Linuxの舞台裏

～秋のLinux World緊急開催～

10月末から11月にかけて、Linux関連では大きなニュースが続いた。先月お伝えしたVA Linuxジャパンの設立、LinuxMLD 5の発表、デスクトップ関連ではNetscape 6の正式版出荷、KDE 2.0やGNOME用ファイルマネージャ「Nautilus」のPR2、HancomOfficeのリリースなどだ。だが、なんといってもLinuxWorld Demo＆Conference／Tokyo 2000がその筆頭だろう。イベントのレポート記事は本誌にお任せするとして、今月は同イベント取材中の裏話をお届けしよう。（日刊アスキー編集部・吉川大郎）

今回のLinuxWorld Demo＆Conference／Tokyo 2000の案内メールが主催のIDGジャパンから配信されてきたのは7月31日。Subjectは「LinuxWorld、10月に緊急開催のご案内と出展募集開始！」というもので、5月に行われたLinuxWorld Expo／Tokyo 2000での盛況を受け、“緊急開催”を決定したという内容だった。

そして、名称のとおり今回はデモンストレーションとカンファレンス中心で行うという。これは、年に2回も「Expo」を行うよりも、秋は春と違った趣旨のイベントを、という出展各社からの意向が反映されたものらしい。新製品のデモンストレーションやカンファレンスが主体で、展示場は春ほどは大きくしないという。Linux関連イベントが頻発した昨年から今年である。たとえ毎回盛況のLinuxイベントであろうとも、同じ内容のものを年に2回も行うのはあまり経済的とはいえないのだろう。

日刊アスキー Linuxの、Do Linux!インタビュー記事

会場はTFTホールと東京ファッションタウン。これらの会場は、春のLinuxWorldが開催された東京ビッグサイトとは、ゆりかもめの駅をはさんで反対側の位置にある。過去LinuxWorldやLinuxConferenceが開催されたので、日刊アスキー Linuxも何回も足を運んだおなじみの場所だ。TFTホールと東京ファッションタウンという2つの建物を行き来しなければならないのが面倒だが、ワークショップなどを聴講する場合は、部屋の中で落ち着いて聞くことができるのでありがたい。

展示会場内でワークショップが開かれると、臨時に間仕切りされただけの部屋に椅子を並べた場所もあり、外の雑音で相当聞きづらいのだ。

さて、肝心の内容はといえば、なかなかの盛況だった。来場者自体は1万8198名と、春の2万4193名に比べると下回ってはいるが、そもそも目標来場者数は1万8000人である。また、Expoというわけでもないのに、展示会場もかなりの人混みとなっていた。

印象に残るのがRed Hat Linux 7Jのワークショップだった。Red Hat Linux 7Jのワークショップは、製品が出荷された直後でもあり、あまり新しい発表はないだろうと思いつつ、やはりどのソフトウェアでもまず最初にサポートするディストリビューションの日本語版だし、Project Vineとのアライアンスのあととあって、絶対聞きに行かねばと思っていたのであった。このワークショップの講演者は、もはやレッドハットの顔ともいえる染谷邦裕氏で、Red Hat Linux関連では同氏の講演を聞くことがもっとも多い。

その染谷氏の講演が、これまた人気があった。ほかのワークショップでは、

開場前はせいぜい50人程度が並び、あとの人は開場してから三々五々部屋に入っていくのだが、このときはすでに100人以上（ちなみに定員は100名）並んでいたのではないだろうか。その中には業界関係者もいたようで（というか、そもそもLinux関連イベント自体、まったくLinux業界に関わっていない人がどれだけいることやら）、私の前に並んでいた2人組は、

A氏「いやー、すごく混んでいるねえ!」
B氏「でもまあ“そめちゃん”の講演じゃ、聞かないわけにもいかないよね」

といった会話を交わしていた。“そめちゃん”というのは、染谷氏のことであろう。

Linux関連イベントを取材していて面白いのは、さっきの講演では熱心に質問をしていた人が、次の講演では壇上に上っているという事態が起こったりすることで、講演者と聴講者の垣根はかなり低い。「コミュニティ」といった側面から考えると当然かもしれないが、ビジネスにフォーカスし、いわゆる「背広族」が大量に参加するLinuxWorldでもこれは同じなのである。

Do Linux!な人にビックリ

イベント取材の際は、必ず写真を撮りながら展示会場を回るのだが、ここで思わぬ人とお会いした。以前、Linuxの技術者になりたいということで、「Do Linux!」という人材教育プロジェクトに参加していた方だ。

日刊アスキー Linuxでは、IT系の人材サービス会社「パソナテック」が中心となって展開している「Do Linux！プロジェクト」関連の記事を10本弱掲載している。Do Linux！を簡単に説明すると、Linux技術者になりたい人に、資格取得から就業までを一貫してサポートするプロジェクトである。参加者は、パソナテックの行う試験に合格後、レッドハットの実施する「RHCE」を受講。RHCEに合格したあとはさらにテンアートニ内で実践研修を受け、最後にパソナテックの斡旋によって、Linux技術者として各企業に就業していくという流れをとる。

ちなみに、パソナテックの「人材サービス会社」というのは、同社が人材“派遣”だけではなく、正社員としての斡旋も行っているからだ。また同社は、就業のみならず教育（各種ベンダー資格のトレーニング企業でもある）事業も行っている。こうした事情で、人材「派遣」ではなく、人材「サービス」会社なのだそうだ。

話を元に戻して、筆者は数カ月前、このDo Linux!第1期生数人に対し、2回に渡ってインタビューを行った。1回目がRHCEの試験終了直後（RHCEは4日間の講習と、5日目に最終試験がある）で、2回目はテンアートニによる実践コース修了後である。

このインタビュー時に話を聞いた1人に、Demoの展示会場内でバッタリと再会したのだ。

コンピュータ関連の編集者の場合、メーカーのマーケティング担当や開発担当の方にお話を聞くのは、さして珍しいことではない。それは取材を受ける側も同じ場合が多く、いわば取材慣れしているわけだ。ところが、Do Linux！の取材では、ふだんまったく取材を受けていない方々に対してのインタビューであり、こちらもネタを拾うというよりは、世間話的な会話をし、その中から記事を構成するといった流れであった。感覚としては個人的に話をしたのに近い。また、取材当時彼は、慣れないLinuxの学習にかなり苦戦されていたようで（ちなみにその方は、他のIT資格を複数持つエンジニアである）、ハードな勉強をされていることが傍目でもわかった。私も一番印象に残っていた方だった。だから、会場でとあるメーカーのブースで説明スタッフになっている彼とお会いした時は、まさに「再会」といった心境になったのである。

彼も私のことを覚えていてくださり、ほんの数分だがお話をすることができて、とても嬉しかった。

●

LinuxWorldは来年5月31日、場所をまた東京ビッグサイトに移し、今度は「Expo」の名を冠して開催される。今から楽しみである。

IBMのスマートウォッチ
会場内でもかなり人目を引いた、Linux腕時計

大勢が詰めかけた、レッドハット染谷氏の講演

正しきタコへの道

初級Linuxer養成講座

第16回　シェルの小技(2)〜コマンドラインの再利用

前回は、ファイル名やコマンド名などの入力を補助してくれる、シェルの「自動補完機能」を紹介した。bashには、単なるファイル名やコマンド名の補完だけでなく、さらに便利な補完機能が備わっている。また、同じような操作によって、過去に入力したコマンドラインを自由に呼び出して、再利用することもできるのだ。

文：竹田善太郎
Text : Zentaro Takeda

先日、自宅にあるノートマシンのハードディスクが故障した。家族が使っているマシンだったので、自分の仕事には大きな支障はなかったのだが、家族とはいえ、**他人**が使っているマシンの回復作業は、**かなり神経を使わされる**ことを思い知らされた。自分が使っているマシンなら、どこにどのようなデータを保存しているかは大体わかっているので、必要なデータだけバックアップしておいて、あとはきれいさっぱり**再インストール**してしまえばいいのだが、他人のマシンの場合、使っている人間によって、データを保存する場所などがまちまちなのでやっかいなのだ。

とくに、Windowsの場合は頭が痛い。アプリケーションごとに、データを保存する場所や方法が違うからだ。通常の「ファイル」としてデータを保存しているのならまだいいのだが、たとえばメールソフトの設定（接続先ホスト名、ユーザーのアカウント情報、アドレス帳）などを**レジストリ**に保存していることがあって、単なるファイルのバックアップではもとの環境に戻すことが**不可能**なのである。このような場合は、起動ドライブの内容を、そのまま新しいハードディスクに移植するしかない。

故障したハードディスクは、いっぺんに使えなくなってしまったのではなく、**時間を追うごとに不良セクタが増大してゆく**という、ありがちだが不気味な故障の仕方だったため、**まともに動いているうち**に早期に新しいハードディスクの移植をしなければならなかった。しかも、故障したのは5年ほど前に購入した旧式のノートPCなので、CD-ROMドライブなどの大容量メディアは利用できず、起動できなくなるほど症状がひどくなる前に、なんとか新しいハードディスクを接続して、現在の環境をまるごと移さなければならなかったのだ。

結局は、手元にあったPCMCIA接続のIDEインターフェイスと余っていた2.5インチのハードディスクを使って、なんとか**移植手術**に成功した。IDEインターフェイスが旧式だったためか、**大容量のハードディスク**（といっても、今となっては貧相な**6Gバイト**なのだが）をうまく認識できず、あせらされたが、別のマシンでパーティション設定と論理フォーマットだけしたら、ファイルのコピーは問題なくできるようになったのだ。最終的に、Windows環境のコピーもうまくいき、不良セクタによって失われたファイルも、とりたてて必要なものではなかったので助かった。ハードディスクが新しくなったせいか、以前よりもマシンのレスポンスも良くなったようで、家人にも喜ばれて一件落着となったのだが、いろんな意味で考えさせられる「事件」だった。

なにより、Windowsにおけるデータ保存方法の不統一と、バックアップの不自由さにはまいってしまう。レジストリの保存や復元などは、もちろん可能になっているのだが、レジストリだけ保存しておいても、Windowsを再インストールしたあとで、そのレジストリ情報をそのまま利用できるわけではない。アプリケーションのインストール状況や設定が以前のものと違っていると、古いレジストリのデータは使えないことが多いのだ。

また、アプリケーションごとに、データ保存のポリシーがまちまちなのも困る。「マイ ドキュメント」フォルダに保存しようとするもの、アプリケーションのインストールされているディレクトリに保存しようとするものなど、

さまざまである。ユーザーが意識して、データファイルを保存するディレクトリを決めていればいいのだが、それとて、急いでいるときなどは、ついついアプリケーションがデフォルトで指定するディレクトリにファイルを置いてしまって、あとになって探し出すのに苦労したりする。

こうやって、広大なディスクのあちこちに、ドキュメントファイルが散らばっている状態になってしまうと、もはや**ファイルの定期的なバックアップ**でデータを保護することなど不可能である。**ドライブまるごとバックアップ**するしか手段はないのだが、数Gバイトから数十Gバイトにもなるドライブをまるごとバックアップする、安価でだれにでも実行できるような手段はない。結局、**その場で思いつく限りのファイル**だけを保存しておいて、メールソフトのアドレスなどの設定は**紙にメモ**したりするのが、一般のユーザーができる精一杯のバックアップ、というのが現状ではないだろうか。

Linuxの世界も安穏ではない。現状では、とりあえず**/homeディレクトリ以下**を保存しておけば、ユーザーが作成したファイルやアプリケーションの設定などは、ほぼ確保できるのだが、たとえば、メールスプール中にあるメールメッセージや、一部のかな漢字変換システムのユーザー辞書などは保存できない。これらのデータは別途保存しておかなければならない。今後、Linuxで利用できるデスクトップアプリケーションが増えてくると、アプリケーションやOSが**抱え込んでしまう**ユーザーデータの保護対策も重要になってくる気がする。

コマンド名の補完

さて、前回から、Linuxの「標準」シェルといえるbashの自動補完機能について説明しているわけだが、とりあえず**迷ったらTabキー**ということだけ覚えておけば十分、という点については、前回も強調したとおりである。コマンドラインを入力するとき、綴りに自身がなかったり、うろ覚えだったりするコマンドを使いたいときには、最初の数文字だけ入力してTabキーを押してみる。運がよければ、目的のコマンドがすぐに出てくるし、だめでももう一度Tabキーを押してみれば、その文字列で始まるコマンドの一覧が表示される。そして、コマンドに与えるファイルの名前についても、最初の数文字を入力してからTabキーを押せばよい。Tabキーを押しても、不意にコマンドが実行されたり、**ファイルが壊されたり**するようなことはないので、とにかく**難しい理屈は抜きにTabキーをたたいてみる**ことをクセにすることをお勧めする。しばらく使っていれば、補完機能の使い方のコツも自然に身についてくるだろう。

このように、Tabキーだけ覚えておけば必要十分な自動補完機能なのだが、これ以外の便利な操作方法があることを知っておいても損はないだろう。無理に覚える必要はまったくないが、**表1**をコピーしてディスプレイの横にでも貼っておけば、いつか役に立つこともあるかもしれない。

■コマンドラインの途中のコマンド名

Tabキーを使った補完機能では、コマンドラインの先頭の単語を補完するときはコマンド名として扱い、それ以降の単語についてはファイル名として補完を試みるようになっている（**図1**）。しかし、コマンドラインの途中にコマンド名を入力したいこともある。たとえば、

```
$ man yuvtoppm
```

というコマンドラインを入力したいときだ。ご存じのように、manコマンドはコマンドの使い方を表示するためのコマンドなので、「man」の後ろにはコ

キー操作	説明
Tab	一般的な補完を試みる（コマンドまたはファイル名）
ESC ?	補完の候補一覧を表示する（Tabを2回押すのと同等）
ESC /	ファイル名補完を行う
Ctrl-X	ファイル名補完の候補一覧を表示する
ESC ~	ユーザー名の補完を行う
Ctrl-X ~	ユーザー名補完の候補一覧を表示する
ESC $	シェル変数名や環境変数名の補完を行う（「$」を入力直後にTabキーを押すのと同等）
Ctrl-X $	変数名の補完候補一覧を表示する
ESC @	ホスト名の補完を行う（「@」の入力直後にTabキーを押すのと同等）
Ctrl-X @	ホスト名補完の候補一覧を表示する
ESC !	コマンド名補完を行う
Ctrl-X !	コマンド名補完の候補一覧を表示する
ESC Tab	ヒストリ（後述）一覧に含まれるコマンドを候補として補完を試みる

表1　bashの自動補完機能の主なキー操作一覧

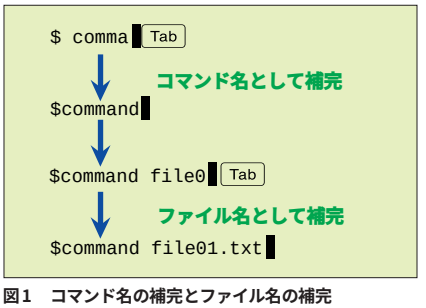


図1　コマンド名の補完とファイル名の補完
bashの補完機能では、コマンドラインの先頭（あるいは、コマンドラインの途中で「;」の直後）の「単語」はコマンド名として、それ以外はファイル名として補完の候補を探すようになっている。

マンド名を入力しなければならない。このときに、

```
$ man yuv
```

と入力してからTabキーを押しても、何も起こらない（あるいは、カレントディレクトリに「yuv」という文字列で始まるファイルがある場合は、そのファイル名が入力されてしまう）。このような場合には、**ESCキーを押してから！キーを押す**（以下「ESC !」と表記する）ことで、ファイル名ではなくコマンド名として補完することができる。また、Ctrl-X ！を入力すれば、候補の一覧が表示される（**画面1**）。

このようなコマンド名補完機能は、初心者のうちはmanコマンドぐらいしか使い道は思いつかないが、Linuxに慣れてくればくるほど、利用できる場面は増えてくる。たとえば、バックグラウンドで実行させるコマンドを実行するときに、そのコマンドの「優先度」を意図的に変えたい場合には、「nice」というコマンドを使うのだが、このときには、niceコマンドの引数としてコマンドラインを入力しなければならない。たとえば、次のようなコマンドラインである。

```
$ nice -n 5 wavtomp3 track01.mp3 &
```

この程度のコマンドラインなら、別に補完機能を使わなくても入力する労力はたかが知れているが、**塵も積もれば**……ということわざ（？）もあるわけだし、「そういえば、コマンド名の補完機能というのもあったっけ。ちょっと試してみるか」といった具合に使ってみると、いつか本当に役に立つときがくるかもしれないのだ。

■その他もろもろの補完

bashの補完機能としては、このほかに**ユーザー名の補完**と**ホスト名の補完**などがある。これらは、企業や学校など、規模の大きなLANに接続して使ったり、1台のLinuxマシンに多くのユーザーがログインして使ったりしている場合以外は、あまり利用する機会はないかもしれない。

ユーザー名の補完は、ほかのユーザーにメールを送ったりするときに利用できる。たとえば、同一のLinuxマシンを共有している「zentaro」というユーザーにmailコマンドでメールを送りたい場合、

```
$ mail ze
```

などと入力してから、ESC ~と入力すると、入力中の単語をユーザー名とみなして、使用中のLinuxマシンに登録されているユーザーのログインIDで補完される。

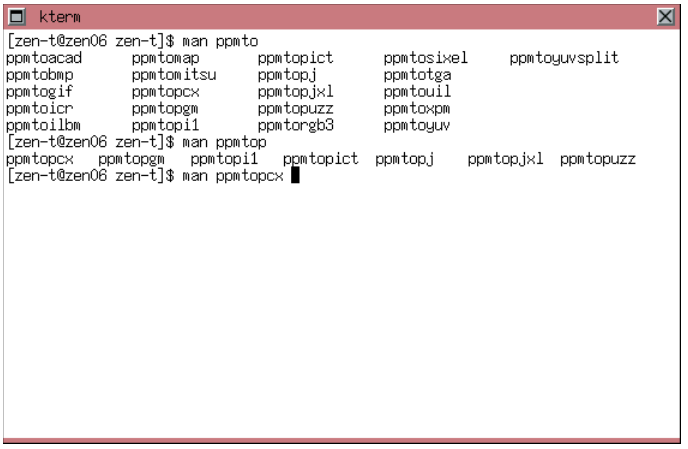
```
$ mail zentaro
```

LANなどで接続されているほかのマシンの名前を補完することもできる。同じように、メールアドレスなどを入力するときに、別のマシン宛てのメールにしたい場合には「zentaro@linux01」というように、「@」に続けてメールホスト名を入力するわけだが、このような場合は、

```
$ mail zentaro@li
```

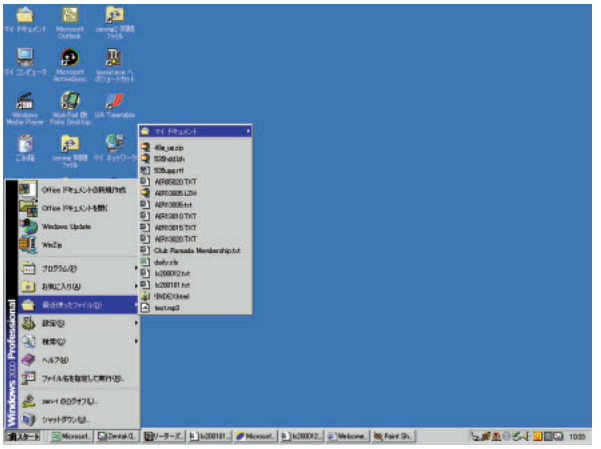
などと入力してからTabキー、あるいはESC @を押すと、ホストの部分が補完される。

```
$ mail zentaro@linux01
```



画面1　コマンド名補完の候補表示

コマンド名検索のキー操作（Ctrl-X !）を入力すると、変換候補が複数ある場合は、その候補一覧を表示してくれる。この候補一覧を見ながらコマンドラインにさらに文字を入力して、候補を1つに絞り込めばよい。



画面2　Windowsの「最近使ったファイル」

これはWindowsにおける「ヒストリ機能」といえるものだが、アプリケーションによっては対応していなかったり、使った覚えのないファイルが含まれていることがあるなど、ちょっと頼りないところがある。

ただし、ホスト名の補完が有効になるのは、/etc/hostsファイルに記載されているホスト名だけである。DNSなど、別の仕組みでホスト名を管理している場合には、ホスト名の補完は有効にならない。

このほかの補完機能について、**表1**にまとめておいたので、参考にしてもらいたい。

ヒストリ機能とは？

「ヒストリ」（history）という言葉は「歴史」とか「履歴」といった意味だが、コンピュータの世界では、**過去の操作の記録**というような意味で使われている。Windowsの世界なら、**「最近使ったファイル」**などがヒストリに相当する機能になるだろう（**画面2**）。直近の一定期間にどのファイルを使用したのかを記録しておいて、あとで同じファイルを利用したい場合に、簡単な操作で呼び出せるようにする仕組みである。

bashのヒストリ機能はWindowsの「最近使ったファイル」とはちょっと違って、使ったファイルではなく**実行したコマンドライン**そのものを保存して、適宜呼び出せるようになっている。bashに限らず、最近のシェルはほとんどがヒストリ機能をもっているのだが、bashのシェルは機能の多さや使いやすさの点で優れている。なにはともあれ、シェルのコマンドラインで、**上矢印のカーソルキー（↑）**を押してみてもらいたい。コマンドラインに、直前に入力したコマンド文字列が表示されるはずだ。

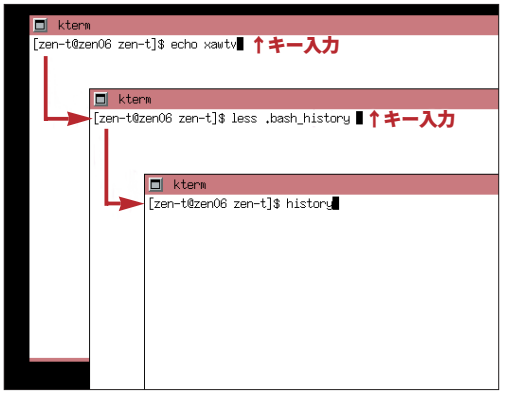
ディストリビューションの種類や設定によっては、カーソルキーを入力しても、意味不明の文字列が表示されることもあるが、このような場合は**Ctrl-P**を押してみる。

↑キー（あるいはCtrl-P）を何度も押してみると、入力したコマンドがどんどん過去にさかのぼって表示されるはずだ（**画面3**）。下矢印のカーソルキー（あるいはCtrl-N）を押すと、今度は逆にどんどん新しいコマンド文字列に戻っていくことができる。ちょうど、現在入力中のコマンド行が**窓**のようになっていて、過去に入力したコマンド文字列が記録された紙を、**上下にスクロール**させている様子を想像してみるとわかりやすい（**図2**）。

もう一度実行したいコマンドラインが表示された状態でEnterキーを押せば、ただちにそのコマンドラインが実行される。過去に実行したコマンドラインをそのままもう一度実行したいのなら、ワードプロセッサや表計算ソフトの画面をスクロールさせるような要領で、上下のカーソルキーを使って目的のコマンドラインを探し出し、Enterキーを押すだけでよいのだ。

コマンドラインの再編集

いったん入力したコマンドラインを再利用する場合、そのまま使う場合よりも、オプションを変えたりファイル名を変えたりするなど、コマンドラインの一部を変更したいことのほうが多いだろう。このような場合には、上下カーソルキーを使って呼び出したコマンドラインを**再編集**することができる。編集の操作は、通常のコマンドラインの入力時と変わらない。左右のカ



画面3　カーソルキーを使ったヒストリの表示
上矢印のカーソルキーを押すたびに、過去に実行したコマンドラインが1つずつさかのぼって表示される。

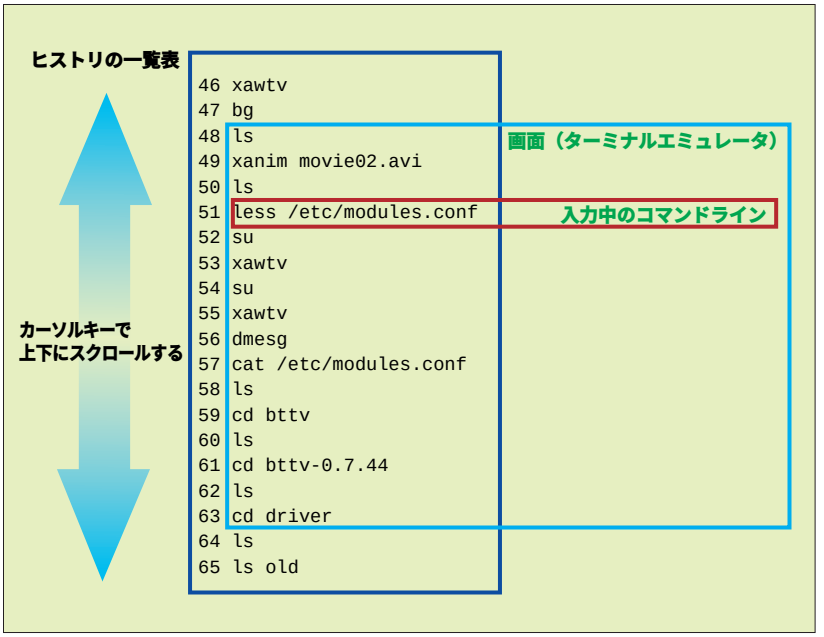


図2　bashのヒストリ機能の概念図
ディスプレイの裏側に、コマンドの履歴を記録した「紙」があって、それをカーソルキーで上下にスクロールさせていると考えるとわかりやすい。あるいは、表示画面が1行分しかない「テキストエディタ」（「ラインエディタ」と呼ぶこともある）のようなものと考えてもよいだろう。

ーソルキー（あるいは、Ctrl＋FとCtrl＋B）を使って、カーソルを左右に動かして、文字を挿入したり、Delキー（あるいはCtrl＋H）で1文字削除したりできるのだ。このようなコマンドラインの編集操作については、すでにマスターしている読者も多いとは思うが、念のため、bashのコマンドラインで利用できるキー操作の一覧を、**表2**にあげておこう。

コマンドラインの再編集のときでも、前回から説明している**自動補完機能**は有効である。たとえば、

```
$ wav2mp3 -b 32 track01.wav
```

というコマンドラインを再編集して、「track01.wav」の部分を「track02.wav」としたい場合には、ヒストリからコマンドラインを呼び出してから、

```
$ wav2mp3 -b 32 track02.
```

という状態にまで書き換えてTabキーを押すと、ほかにtrack02.で始まるファイルがないのなら、

```
$ wav2mp3 -b 32 track02.wav
```

まで入力された状態にしてくれる。

再編集が終了して目的のコマンドラインが完成したら、Enterキーを押せば、編集結果のコマンドラインが実行される。なお、Enterキーを押すときには、カーソルがどの位置にあってもよい。コマンドラインの途中にカーソルがあっても、**コマンドラインが途中で改行されるなんていうことはない**ので安心してほしい。

編集結果のコマンドラインは、あらためてヒストリの末尾に追加される。ヒストリ中のコマンドラインを呼び出して再編集しても、元となったコマンドラインの記録については、編集前の状態でそのまま残っている。したがって、あらためて、元のコマンドラインをそのまま呼び出したい場合でも大丈夫である。

ヒストリは保存される！

ヒストリの内容は保存されている。ヒストリとは履歴という意味だから、**あまりにもあたりまえ**に聞こえることなのだが、じつは、昔のシェルのヒストリ機能では、ログアウトしたり、ターミナルウィンドウを閉じたりしてシェルを終了してしまうと、それまでのヒストリは消えてしまっていたのだ。しかし、現在のbashのヒストリ機能では、**最大1000回分**のコマンドラインが、ホームディレクトリ上の**.bash_history**というファイルに保存されるようになっている。そして、ログアウトするなどしてシェルを終了しても、次にシェルを起動したときに、古いヒストリを呼び出せるようになっているのだ。

保存されているヒストリの内容は、~/.bash_historyファイルをlessコマンドなどで表示させても見ることができるが、その名もずばり**historyコマンド**を使って表示させることもできる（**画面4**）。historyコマンドだけでは、最大1000行ものコマンド行が一気に表示されるので、lessコマンドなどを併用するとよいだろう。

```
$ history | less
```

あるいは、historyコマンドに数字を引数として与えると、その数字で指定した数だけ、最新のヒストリの内容を表示してくれる。たとえば5行だけ表示したい場合は次のようにする。

```
kterm
[zen-t@zen02 zen-t]$ history 10
  992  history -5
  993  which history
  994  man history
  995  resize
  996  clear
  997  history
  998  history 10
  999  exit
 1000  clear
 1001  history 10
[zen-t@zen02 zen-t]$
```

画面4　historyコマンドの実行結果
historyコマンドはbashの「内部コマンド」で、その時点でのヒストリの一覧を表示してくれる。何行分のヒストリを表示するかを数字で指定できる。

```
kterm
[zen-t@zen06 zen-t]$ history 10
 1043  pw
 1044  pwd
 1045  history > zen01/hist.txt
 1046  clear
 1047  history
 1048  history 10
 1049  clear
 1050  telnet zen02
 1051  clear
 1052  history 10
(reverse-i-search)'tel': telnet zen02
```

画面5　ヒストリの「検索モード」
コマンドラインでCtrl-Rを押すと、ヒストリの検索を行うモードになる。ここで、検索したい文字を1文字ずつ入力していくと、入力されている文字列にマッチするコマンドラインが表示されていく。もう一度Ctrl-Rを押せば、同じ文字列にマッチする、さらに古いコマンドラインが表示される。Emacsのi-search機能と似ている。

```
$ history 5
 974  last
 975  w
 976  ps x
 977  kill 3023
 978  history 5
```

何桁かの数字に続いて、ヒストリ中のコマンドラインが表示される。先頭の数字は、そのコマンドラインの**ヒストリ番号**である。この数字を覚えておいて、その行のコマンドラインを直接呼び出すこともできる。新しいコマンドラインで、！（エクスクラメーションマーク）のあとに、ヒストリ番号を入力するだけでよい。

```
$ !976
```

この例では、先ほどのヒストリの一覧の976番の「ps x」というコマンドラインが実行される。「いまからX回前に実行したコマンド」という形で指定することもできる。先ほどの「！」のあとに、**負の数値**を指定すればよいのだ。たとえば、ヒストリの内容が、

```
1001  /usr/sbin/traceroute mail.hogehoge
1002  ping mail.hogehoge
1003  mail zen@mail.hogehoge
1004  history 5
```

となっている状態で、4回前のコマンドライン、すなわち「1001番」のコマンドラインを呼び出したい場合、

```
$ !-4
```

と入力してもよいのだ。

ヒストリ番号を直接指定する使い方は、あまり便利なように思えないかもしれない。実際、シェルを使い始めたばかりのころは、カーソルキーでヒストリをたどっていったほうが面倒がないだろう。しかし、次のような使い方ができると、**ずいぶん前にこんなコマンドを入力したはずだが**……と思ったときに便利である。

```
$ history | grep yuvsplit
342 yuvsplittoppm /home/lmuser/pictures/capture01 640 480 > /home/lmuser/pictures/capture01.ppm
```

このように、historyコマンドの出力をパイプでgrepコマンド（文字列を検索するコマンド）に渡せば、目的のコマンドラインのヒストリ番号を見つけることができる。あとは、

```
$ !342
```

と入力するだけで、ヒストリをカーソルキーでたどらなくても、大昔に実行したコマンドを呼び出すことができる。

ヒストリをもっとすばやく検索する

実は、bashのヒストリでは、上述のようにgrepコマンドを使わなくても、もっと簡単にヒストリの内容を検索できるようになっている。コマンドラインになにも入力されていない状態で**Ctrl＋Rキー**を押してみてほしい。**画面5**のようになるはずだ。

この状態で、キーボードから「history」と入力すると、画面上には、「history」という文字列を含む、最新のコマンドラインが表示される。これが目的のコマンドラインでない場合には、さらにCtrl＋Rキーを押せば、さらに古いヒストリにさかのぼって、「history」という文字列を含むコマンドラインが次々と表示される。

目的のコマンドラインが表示されたら、その時点でEnterキーを押せばそのコマンドラインが実行されるし、もちろん左右のカーソルキーなどを使ってそのコマンドラインを再編集することもできる。

●

ここで説明した機能の多くは、Emacsのキー入力に準拠している。**Emacsは嫌いとか大人ならvi**というユーザー向けに、実はviのキーに準拠したモードもあるのだが、それについては次号で説明することにしよう。

	キー操作	説明
カーソル移動	←またはCtrl-B	カーソルを左に1文字移動
	→またはCtrl-F	カーソルを右に1文字移動
	ESC B	カーソルを1単語分左に移動
	ESC F	カーソルを1単語分右に移動
	Ctrl-A	カーソルを行の先頭に移動
	Ctrl-E	カーソルを行の末尾に移動
文字削除	Delete、Backspace Ctrl-H	カーソルの左側の1文字を削除 （注意：Linuxの設定によっては動作が異なる場合がある）
	Ctrl-D	カーソル位置の1文字を削除
	ESC Delete ESC Ctrl-H	カーソルの左側の1単語を削除 （カーソルが単語の内部に位置している場合は カーソルの左側から単語の先頭までの文字を削除）
	ESC D	カーソル位置から右側の1単語を削除 （カーソルが単語の内部に位置している場合は カーソルの右側から単語の末尾までの文字を削除）
	Ctrl-K	カーソル位置からコマンド行の末尾までを削除
	Ctrl-Y	直前に削除した文字列をカーソル位置に挿入 （「カットアンドペースト」の「ペースト」操作に相当）

表2　bashのコマンドライン編集時のキー操作一覧（Emacs編集モード）

InterBase 6.0

今回は、スキーマ設計を行うときに必要な知識である制約と、インデックスについて説明します。これらは、メンテナンス効率が高く、クライアントアプリケーションの負荷を減らすスキーマ構造を作成するためには必要不可欠な機能です。

第4回　制約とインデックスについて

文：加藤大受
Text:Daiju Kato　kato@ms.tokyo.jcom.ne.jp

前回はInterBase 6から追加されたDialectについてと、サポートしているデータ型について説明しました。InterBase 6から、Dialectによってサポートするデータ型とサポートする範囲が異なっています。旧バージョンとの互換性を重視するか、最新機能を使用することを重視するかのどちらかを決めてからデータベースを作成することをお勧めします。

Dialectはデータベース作成後にも変更可能ですが、作成後の変更はデータの有効桁数の減少などを伴いますのでできれば避けたほうがよいでしょう。

今回は、テーブルを設計するときに同時に定義する整合性制約などについて説明します。制約というとすごく難しいと思われがちですが、言葉に惑わされずに本質を理解していただきたいと思います。

制約とは

これまで説明したように、リレーショナルデータベースには文字型、数値型、時間型、バイナリ型など、数多くのデータ型が用意されています。フィールドを定義する場合、必ずデータ型を指定します。データ型を指定することにより、そのフィールドに保存されるデータ型を特定することができます。もし、データ型がわからなかったら、クライアントはアクセスするたびに受け取ったデータの型を特定しなければなりません。これは非常にオーバーヘッドの大きい作業になるだけでなく、どの程度のメモリ領域が必要になるかがわからないため、プログラム的にも非常に複雑になってしまいます。データ型が特定されていることにより、クライアントは受け取るデータ型がわかり、プログラム言語との親和性も高くなるわけです。

では、制約はどんな意味を持つのでしょうか。制約とは言葉の通り、条件を付けて自由を制限することです。リレーショナルデータベースでは制約を使用することにより、フィールドに格納されるデータを制限することができます。たとえば、初期値としてある数値や文字列を入れたり、保存される数値の範囲を指定したりすることができます。論理型のないリレーショナルデータベースでも、数値型フィールドに0または1以外の値が入らないように制約を付け、論理型の代わりに使用することも可能です。

また、必ずデータが保存されるように、NOT NULL制約を付けることもできます。

画面1は、初期値とNOT NULL制約を指定したフィールドを持つテーブルに対して、データを格納したものです。初期値をfield2に指定していますので、データ追加時に保存されるデータを指定していないときは、この初期値が格納されます。

NOT NULL制約を持つフィールドにデータを入れずに保存した場合、**画面2**のようなエラーが発生します。このエラーは、Field3はNOT NULL制約なのにデータが指定されていないというものです。

制約にはこのほか、必ずほかと異なるデータが格納されるUNIQUE制約、妥当性検査を行うCHECK制約、プライマリキー（主キー）に代表される、整合性などの制約があります。

プライマリキーはスキーマを設計するときに必ずといっていいほど、よく使われる整合性制約です。また、各テーブルには必ずプライマリキーを付けることを勧めている方も多くいます。

プライマリキーとは、レコードを一意に識別するフィールド、またはフィールドの集まりで、ほかのテーブルを参

```
# ./isql -U SYSDBA -P masterkey ―――― isqlを起動
Use CONNECT or CREATE DATABASE to specify a database
SQL> create database '/tmp/mytest.gdb' default character set eucj; ―― データベースの作成（デフォルトキャラクタセットをEUCJISに設定）
SQL> create table table1 ( ―――― テーブルの作成
CON>        field1 char(10),
CON>        field2 char(10) default 'TEST',
CON>        field3 integer not null
CON>        );
SQL> insert into table1 ―――― データの入力
CON>        (field1, field3)
CON>        values
CON>        ('KATO',1);
SQL> select * from table1; ―――― データの参照

FIELD1     FIELD2           FIELD3
========== ========== ============

KATO       TEST                1

SQL> quit; ―――― isqlの終了
```

画面1　初期値の設定とNOT NULL制約を指定

```
SQL> insert into table1
CON>     (field1,field2)
CON>    values
CON>     ('DAIJU','KATO');
Statement failed, SQLCODE = -625

validation error for column FIELD3, value "*** null ***"
```

画面2　NOT NULL制約でデータを保存しない場合のエラー

▼リスト1　プライマリキーを指定したテーブルの作成

```
create table table2 (
       f1 char(10) not null,
       f2 char(10) not null,
       f3 integer not null,
       f4 date default 'now',
       primary key (f1));
```

```
SQL> create table table2 ( ―――― f1フィールドをプライマリキーに持つTable2を作成
CON>        f1 char(10) not null primary key,
CON>        f2 char(10) not null,
CON>        f3 integer not null,
CON>        f4 date default 'now'
CON>        );
SQL>
SQL> create table table3 ( ―――― f1、f2フィールドをプライマリキーに持つTable3を作成
CON>        f1 char(10) not null,
CON>        f2 char(10) not null,
CON>        f3 integer not null,
CON>        f4 date default 'now',
CON>        primary key (f1,f2)
CON>        );
SQL>
```

画面3　プライマリキーの設定

照することのできる外部キー（Foreign Key）を設定できる特別な整合性制約です。1つのテーブルには、プライマリキーは1つしか設定できません。ただし、必ず1つのフィールド単体である必要はなく、2つ以上のフィールドを連結してプライマリキーを設定することは可能です。

画面3は、f1フィールドをプライマリキーに持つTable2と、f1フィールドとf2フィールドをプライマリキーに持つTable3を作成するSQL文を、ISQLで実行しているところです。Table2では、f1フィールドの設定時にプライマリキーのオプションを指定していますが、**リスト1**のように書くことも可能です。プライマリキーに2つのフィールドを指定する場合は、Primary Key (＜フィールド1＞, ＜フィールド2＞……)という書き方となります。

外部キーについて

すでに何度か正規化について簡単に触れていますが、正規化とは、同じデータが2つ以上のテーブルに納められることなく、常に1つとして存在しているようにし、データの保存・検索、スキーマのメンテナンスを行いやすいようにすることです。

外部キー制約（Foreign Key Constraint）は、参照制約（Referential Integrity）とも呼ばれます。外部キーは、1つのテーブルのフィールド、またはフィールドの集まりで、ほかのテーブル（同一テーブルでも可能）のプライマリキーと一致するデータを持ちます。参照するデータを持ち、一致するかどうかの整合を行うので参照制約とも呼ばれるのです。

図1のように外部キーには必ず親子関係が成り立ちます。受注マスタの得意先コードは、得意先マスタの得意先コードを参照していますので、得意先マスタが親で、受注マスタが子という形になります。

このスキーマ構造を作成するSQL文は、**画面4**のようになります。実際にデータを入力し、参照元にデータがない場合にどのようなエラーが起きるかを示しています。参照データがない場合、つまり該当する顧客コードがないデータを入力しようとすると、「SQLCODE = -530」の参照制約のエラーが発生します。ここで、「INTEG_24」という名前が出てきますが、これはInterBaseが自動的に付けた参照制約の名前です。

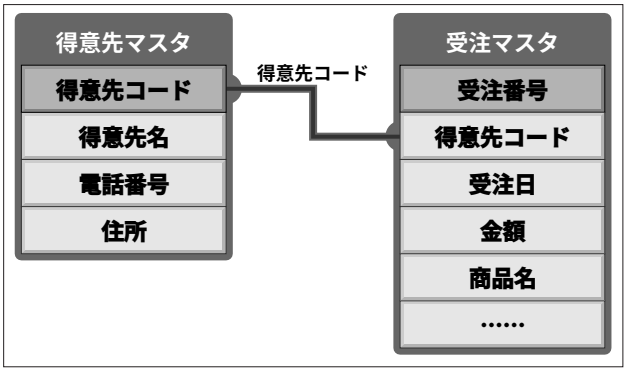


図1　受注マスタと得意先マスタの参照関係

どのような名前が付いているかは、SHOW TABLE ＜テーブル名＞を実行することで見ることができます。また、constraint ＜制約名＞と指定することで、**リスト2**のように自分で制約名を付けることもできます。

外部キーを利用することで、本来格納できないデータが保存されてしまうなどの問題を解決することができ、プログラム側でこれらの考慮を行う必要がなくなります。

CHECK制約について

CHECK制約とは、格納されるデータが妥当であるかどうか、つまり保存してもいいかどうかを検査するための制約です。たとえば、統計データの最小値と最大値を格納するようなテーブルの場合、最大値のフィールドに格納されるデータは、必ず最小値のデータよりも大きくなければならないというルールが成り立ちます。このルールとなるのがCHECK制約というわけです。ただし、フィールドは1つしかCHECK制約を持つことができませんので注意が必要です。

画面5はCHECK制約を使用するテーブルを作成し、データを格納しています。参照制約と同じように、CHECK制約に違反するデータを書き込もうとすると、「SQLCODE =-297」のCHECK制約のエラーが発生します。

ここでは、データが追加される際に、min_salaryがmax_salaryよりも小さいデータはエラーとなるように設定してあります（min_salary<max_salary）。

▼リスト2　制約名を付けてテーブルを作成

```
create table orders (
        order_no    char(10) not null,
        cust_no     char(10) not null,
        order_date  date default 'now',
        amount      integer not null,
        item        char(50) not null,
        primary key (order_no),
        constraint check_cust_no
                    foreign key (cust_no)
                    references customer (cust_no)
        );
```

ドメインについて

InterBaseには、ドメインというグローバルなフィールドの定義ができる機能があります。複数のテーブルで、同じデータ型、同じフィールドのサイズを持つ場合、ドメインを使用することでメンテナンス性が向上します。各テーブルの定義の時は、ドメイン名でフィールドのタイプを定義しておき、変更が生じたらドメインの定義を修正することでテーブル側の修正を行うことなく、スキーマの再構築を行うことができます。ドメインは参照制約は使用できませんが、CHECK制約の指定は可能です。

ドメインの作成はCREATE DOMAIN構文（**リスト3**）を使用します。

```
QL> create table customer (                                    顧客マスタを定義
CON>        cust_no char(10) not null,
CON>        cust_name char(50)  not null,
CON>        tel char(10),
CON>        address char(120),
CON>        primary key (cust_no)
CON>        );
SQL>
SQL> create table orders (                                     受注マスタを定義
CON>        order_no   char(10) not null,
CON>        cust_no    char(10) not null,
CON>        order_date date default 'now',
CON>        amount     integer not null,
CON>        item       char(50) not null,
CON>        primary key (order_no),
CON>        foreign key (cust_no) references customer (cust_no)  顧客マスタの顧客コードを外部キーとして設定
CON>        );
SQL>
SQL> insert into customer                                      顧客マスタにデータの入力
CON>        ( cust_no, cust_name, tel,address)
CON>        values
CON>        ('00001','DAIJU KATO', '03-5932-', 'Suginami-ku, Tokyo');
SQL>
SQL> insert into orders                                        受注マスタにデータを入力(顧客コードあり)
CON>        (order_no, cust_no, amount, item)
CON>        values
CON>        ('10001','00001',1980,'Linux Magazine');
SQL>
SQL> insert into orders                                        受注マスタにデータを入力（顧客コードなし）――エラー
CON>        (order_no, cust_no, amount, item)
CON>        values
CON>        ('10002','10000',1980, 'BSD Magazine');
Statement failed, SQLCODE = -530
violation of FOREIGN KEY constraint "INTEG_24" on table "ORDERS"   参照テーブルにデータがないのでエラー発生
SQL>
SQL> show table orders;
ORDER_NO                        CHAR(10) Not Null
ORDER_DATE                      DATE Nullable default 'now'
AMOUNT                          INTEGER Not Null
ITEM                            CHAR(50) Not Null
CUST_NO                         CHAR(10) Not Null
CONSTRAINT INTEG_24:                                           制約名を確認
  Foreign key (CUST_NO)    References CUSTOMER (CUST_NO)
CONSTRAINT INTEG_22:
  Primary key (ORDER_NO)
SQL>
```

画面4　外部キーを使うスキーマの作成

使用できるデータ型がフィールドで使用できるデータ型と同じです。ドメインにはデフォルト値を指定することができ、フィールド定義の時と同じように、文字列、数値、今日の日付（'NOW'）などを指定することができます。また、USERというキーワードを使用することで、ログインしているユーザー名を格納することができます。

画面6はドメイン定義を行い、ドメインを使用したテーブルを作成した例です。SHOW TABLEで作成したテーブルの構造を確認してみると、ドメイン定義の情報が活かされていることがわかります。

ドメイン情報の変更はALTER DOMAIN構文（**リスト4**）で行うことができますが、データ型、サイズ、NOT NULL制約については変更できません。CHECK制約やデフォルト値などについては変更が可能です。

インデックスとは

インデックスとは、その名前の通り索引です。リレーショナルデータベースでは、データの検索を高速化させるための技術として使用しています。インデックスを使用することで、データの検索を高速化することができます。しかし、複数フィールドからなるマルチインデックスを1つのテーブルに多く使用していたりすると、パフォーマンスが悪化してしまう場合もありますので、必要最低限のインデックスを作成するように心がけてください。

たとえば、顧客マスタなどでは顧客コードだけにインデックスを使用し、顧客名などについてはインデックスを作成しないようにします。

数値型のフィールドにもインデックスを付けることはできますが、筆者は文字型の、とりわけCHAR型のフィールドにのみインデックスを付けるようにしています。こういうルールを自分で決めておくことで、意味のないインデックスは作らないようにしているわけです。また、インデックスはあとから作ることができますので、必要になってから作成してもいいでしょう。

インデックスが必要となる条件は、

▼リスト3　CREATE DOMAIN構文

```
CREATE DOMAIN domain [AS] <データ型>
[DEFAULT { リテラル値  | NULL | USER}]
[NOT NULL] [CHECK (<CHECK制約の条件>)]
[COLLATE <コレーションオーダー>];
```

▼リスト4　ALTER DOMAIN構文

```
ALTER DOMAIN name {
[SET DEFAULT { リテラル値 | NULL | USER}]
| [DROP DEFAULT]
| [ADD [CONSTRAINT] CHECK (<CHECK制約の条件>)]
| [DROP CONSTRAINT]
};
```

```
SQL> create table job (                                    ← 職種テーブルの作成
CON>        jobcode char(10) not null,
CON>        jobname char(50) not null,
CON>        min_salary integer not null,
CON>        max_salary integer not null,
CON>        primary key (jobcode),
CON>        constraint check_salary
CON>           check (min_salary < max_salary)
CON>        );
SQL>
SQL> insert into job                                       ← 職種テーブルにデータを追加
CON>        (jobcode, jobname, min_salary, max_salary)
CON>        values
CON>        ('00001','ACCOUNT', 5000, 10000);
SQL>
SQL> insert into job                                       ← 職種テーブルにデータを追加――エラーケース
CON>     (jobcode, jobname, min_salary, max_salary)
CON>     values
CON>     ('0002','SALES', 8000,7000);
Statement failed, SQLCODE = -297

Operation violates CHECK constraint CHECK_SALARY on view or table JOB
SQL>
```

画面5　CHECK制約を使用

・検索条件が頻繁に列を参照する場合
・結合条件が頻繁に列を参照する場合
・ORDER BY文が頻繁に列を使いデータをソートする場合

です。

プライマリキーを設定すると、自動的にインデックスが作られます。InterBaseでは、最大64個まで1つのテーブルにインデックスを作成することができます。SHOW INDEX文を実行することで、プライマリキーがインデックスとして作成されることがわかります（**画面7**）。

InterBaseでは、検索時にどのインデックスを使用すればいいかを自動的に選択してくれる、クエリオプティマイザ機能が提供されています。つまり、有効なインデックスが設定されていれば、検索時にInterBaseサーバが有益なインデックスを選択してくれるというわけです。

インデックスの作成はCREATE INDEX構文を使用し、昇順、降順の両方を作成することが可能です。

```
CREATE [UNIQUE] [ASC|DESC] INDEX ...
```

ASCは昇順で、DESCは降順です。

インデックスが効果的に使われているかどうかは、SET PLAN文を使用することで確認することができます。InterBaseでは、クエリの方法をプランと呼んでいます。SET PLAN文を使用すると、検索時にどのインデックスを使用しているかを表示してくれます。**画面8**は、サンプルのemployee.gdbで、給料が10000以上で、かつ従業員番号は100以上の社員の名前と名字を検索した場合です。この場合では、NAMEXというインデックスを使用していることがわかります。NAMEXというインデックスがどんな構造をしているかは、SHOW INDEX ＜インデックス名＞で調べることができます。インデックスを使用していない場合、PLAN (NATURAL)と表示されます。

PLANの自動選択であるオプティマイザを使用したくないときは、SELECT文のPLAN句を指定することができます。

```
SQL> CREATE DOMAIN USERNAME AS VARCHAR(20)          ← デフォルトがユーザー名となるUSERNAMEのドメインの作成
     DEFAULT USER;
SQL>
SQL> CREATE DOMAIN CUST_NO AS CHAR(10) NOT NULL;    ← 顧客番号(CUST_NO)のドメインの作成
SQL>
SQL> CREATE TABLE ORDER_ENTRY (                     ← ドメインを使用したテーブルの作成
CON>         CUST_NO CUST_NO,
CON>         ENTRY_BY USERNAME,
CON>         ENTRY_DATE DATE DEFAULT 'NOW',
CON>         PRIMARY KEY (CUST_NO)
CON>       );
SQL>
SQL> show table ORDER_ENTRY;
CUST_NO                         (CUST_NO) CHAR(10) Not Null
ENTRY_BY                        (USERNAME) VARCHAR(20) Nullable DEFAULT USER
ENTRY_DATE                      DATE Nullable DEFAULT 'NOW'
CONSTRAINT INTEG_35:
  Primary key (CUST_NO)
SQL>
SQL> insert into ORDER_ENTRY                        ← データを追加
CON>     (CUST_NO)
CON>     VALUES
CON>     ('0001');
SQL>
SQL> select * from ORDER_ENTRY;

CUST_NO    ENTRY_BY              ENTRY_DATE
========== ==================== ===========

0001       SYSDBA                2000-11-19

SQL>
```

画面6　ドメインの定義

インデックスのメンテナンス

インデックスを持つテーブルにデータが保存されると、自動的にインデックスがメンテナンスされ、新しいデータや変更されたデータに合う形に修正されます。しかし、大量のデータの削除・追加が行われると、インデックスの更新が追いつかず、効率の悪いインデックスになってしまいます。InterBaseはガベージコレクションという機能を持っており、データの削除などで空いたスペースを一番後ろの領域に回して、再利用できるようにしていますが、時にはメンテナンスが必要です。メンテナンスの方法としては次のようなものがあります。

- **ALTER INDEX文を使用してインデックスを再作成する**
- **SET STATISTICS文でインデックスの選択性を再計算する**
- **DROP INDEX文を実行したあと、CREATE INDEXでインデックスを再作成する**
- **gbakユーティリティでデータベースのバックアップ・リストアを行う**

一番理想的なのはgbakユーティリティを利用してデータベースのバックアップ・リストアを行う方法ですが、誰かがアクセスしていると使用できないですし、24時間稼働しているようなシステムではこの方法は使えません。運用時間が長いシステムの場合は、SET STATISTICS文を使用するといいでしょう。

SET STATISTICS文を利用すると、インデックスの選択性を再計算します。インデックスの選択性とは、テーブルがアクセスされるときにInterBaseサーバによって行われるオプティマイジングのための計算で、テーブルの個々の行数を基に計算します。これらの情報はキャッシュメモリに格納され、オプティマイザはこのキャッシュメモリにアクセスして、一番最適なクエリの抽出プランを選択します。インデックス付きのフィールドのデータ量が急激

```
SQL> show index;
RDB$PRIMARY5 UNIQUE INDEX ON CUSTOMER(CUST_NO)
RDB$PRIMARY10 UNIQUE INDEX ON JOB(JOBCODE)
RDB$FOREIGN9 INDEX ON ORDERS(CUST_NO)
RDB$PRIMARY8 UNIQUE INDEX ON ORDERS(ORDER_NO)
RDB$PRIMARY11 UNIQUE INDEX ON ORDER_ENTRY(CUST_NO)
RDB$PRIMARY1 UNIQUE INDEX ON TABLE1(F1)
RDB$PRIMARY3 UNIQUE INDEX ON TABLE2(F1)
RDB$PRIMARY4 UNIQUE INDEX ON TABLE3(F1, F2)
SQL>
```

画面7　SHOW INDEXでインデックスを確認

```
SQL> connect '/opt/interbase/examples/employee.gdb';
WARNING: This database speaks SQL dialect 1 but Client SQL dialect was set to 3.
Database:  '/opt/interbase/examples/employee.gdb', User: SYSDBA

SQL>
SQL> select last_name, first_name from employee
CON>        where emp_no >100 and salary >100000
CON>        order by last_name, first_name;

PLAN (EMPLOYEE ORDER NAMEX)

LAST_NAME            FIRST_NAME
==================== ===============

Bender               Oliver H.
Cook                 Kevin
Ferrari              Roberto
Glon                 Jacques
Ichida               Yuki
Osborne              Pierre
Yamamoto             Takashi

SQL>
SQL> show index namex;
NAMEX INDEX ON EMPLOYEE(LAST_NAME, FIRST_NAME)
```

給料が10000以上でかつ従業員番号は100以上の社員の名前と名字を表示する

画面8　プランの表示

に増加、または減少すると、この抽出されたプランが最適ではなくなり、パフォーマンスが低下する可能性があるわけです。

SET STATISTICS構文は次の通りです。

```
SET STATISTICS INDEX <インデックス名>
```

この文を実行することで指定された再計算が行われます。ただし、SET STATISTICS文を実行できるのはインデックスを作成したユーザー、SYSDBAユーザー、オペレーションシステムでroot権限を持つユーザーだけとなっています。また、SET STATISTICS文はインデックスを再作成するわけではないので、再作成を行いたいときはALTER INDEX文、またはDROP INDEX文とCREATE INDEX文の組み合わせで行ってください。

大量のデータの追加または削除を行うときは、ALTER INDEX文を使用していったんインデックスを停止にしてしまう方法もあります。この方法を利用すると、データの追加、または削除で、1行ずつ行われるインデックスの修正が行われなくなり、最後にALTER INDEX文でインデックスを活動状態にしたときに、インデックスの再作成が行われます。

ALTER INDEX構文は次の通りです。

```
ALTER INDEX <インデックス名> [ACTIVE|INACTIVE]
```

データのインポート処理などを行うときはこちらの方法が向いていると思われますので、トランザクションの開始と最後にこのような処理を入れてパフォーマンスをあげてみるのもいいでしょう。

最後はインデックスを作り直す方法になります。再作成にはDROP INDEX文とCREATE INDEX文を使用することになります。DROP INDEX構文は次の通りです。

```
DROP INDEX <インデックス名>
```

ただし、こちらは誰もインデックスを使用していない場合にのみ有効な手段で、インデックスの作成が行われると、その間のCPUパワーがインデックスの作成に奪われてしまうので、システム全体のパフォーマンスが低下する可能性があります。作り直す方法を利用する場合は、クライアントからのアクセスのない時間にバッチ的に行うなどの方法を考えてみるのもいいでしょう。cronコマンドを利用して、メンテナンス用のSQLスクリプトをISQLで実行する方法などもひとつのやり方です。

次回に向けて

今回の制約は非常に難しい機能です。制約をうまく使いこなせるようになると、スキーマ設計に悩まなくなります。実際にISQLなどのツールを使用して簡単なスキーマを作りながら、実際に動かしてみると動作が見えてくると思います。

次回は、リレーショナルデータベースのプログラム言語であるトリガー、プロシージャについて説明します。それぞれのリレーショナルデータベースには、トリガー、プロシージャを作成するための言語が用意されています。InterBaseでは、DSQL(Dynamic SQL)と呼ばれています。この言語を説明しながら、実際にトリガー、プロシージャを作成していきます。

最新情報

10月29日にIBConsoleの最新ビルドであるBuild 322、Windows版のInterBaseのクライアントモジュールの一部であるgds32.dllのBuild 627が公開されました。また、FreeBSD 4.0へのポーティングについてもfirebird.sourceforget.netで始まっています。いよいよ、InterBase関連のプロジェクトが動き始めました。興味がある方はIBDIのWebサイトを参照してください。

また、11月2日にInterBase 6.01のLinux版が公開されました。

なお、米国のInterBaseの開発チームより、日本でもInterBase 6.0のメーリングリストを立ち上げたらどうかという話がきています。オープンソースの製品ですので、使っていく人々が育てていく必要があります。

どの程度の人がそれを望んでいるかによってメーリングリストの立ち上げを検討したいと思っています。参加したいと思われる方は、筆者までメールで連絡ください。

- IBConsoleの最新ビルド（community.borland.comに登録が必要）
  http://ww6.borland.com/codecentral/ccweb.exe/listing?id=15081
- InterBase 6.01(本体)のダウンロードサイト
  http://www.borland.com/interbase/downloads/
- InterBase 6.0(ツール)のダウンロードサイト
  http://www.interbase2000.org/binaries.htm

Javaプログラミング入門
Javaで3Dゲームを作ろう

いよいよ最終回を迎えるこの連載の最後は、いよいよ3Dブロックくずしの完成です。「GAME OVER」の表示や、音など、ちょっとした味付けを行いたいと思います。

最終回　3Dブロックくずしの完成

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote／Kazaguruma

この連載もとうとう最終回をむかえました。前回までに、ボールとラケットとブロック、そしてそれらが3D空間で動くようになりましたし、ブロックにボールが当たればブロックを消す、またはラケットにボールが当たらなければ空振りというような、このゲームの心臓部ともいえるグラフィックスと各オブジェクト間のやりとりがプログラミングできました。

まがりなりにも動きのある3Dグラフィックスのプログラミング、という大きな課題はクリアしたわけですが、前回までのアプレットを動かしてみた人はおわかりのように、「ゲーム」としてはまだちょっと物足りないのではないでしょうか。このままでは、いわば3Dグラフィックスの「実験」プログラムのようなもので、これが「ゲーム」といえるようになるにはルールや勝ち負けの判定といった味付けが必要でしょう。

ゲームの流れ

では、このアプレットを「ゲーム」と呼ぶことができるための、最低限のルールと流れを考えてみましょう。

図1が今回考えるゲームの流れです。あたりまえと言えばあたりまえの内容ですが、フローチャートなどと呼べるものでなくてもかまいませんから、できるだけこのような図を書いておくことをお勧めします。

実際にプログラムになってしまうと、全体の流れよりもそれぞれのオブジェクトがどのような処理をすればよいのかといった、部分ごとの構造になってしまうため、その流れをプログラムから読み取るのが困難になります。今回くらいの簡単な流れであれば、まだプログラムからでも読み取れるかもしれませんが、もうちょっと複雑な流れになると非常に困難になってしまいます。

たとえ自分で作ったプログラムであっても、少し時間がたつと「このプログラムは一体何をしているんだろう？」といったことになりかねないのです。

ですから、プログラミングの際にはまず、**図1**のような大きな流れを図に

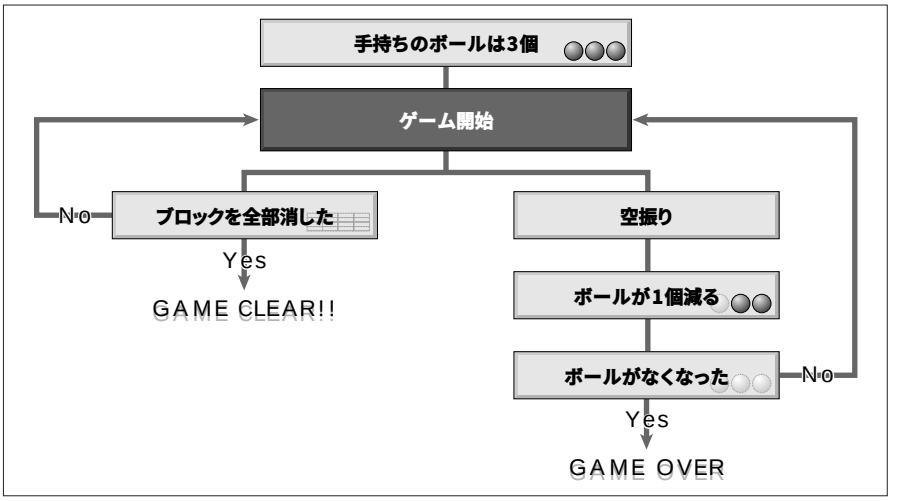


図1　ゲームの流れ

描きます。次に、この図の各部の処理をどのようにして行うかをメモしてみます（**図2**）。

ここでは、できるだけ具体的にどのオブジェクトのどのメソッドで処理するのか、また、ほかの（オブジェクトの）メソッドとの連携をどう行うのかと、そのときのパラメータなどを明記しておきます。そうすればプログラミングのときに、「えーと、どのようにプログラミングすればいいのかな」などと悩んだりすることもありませんし、あとでプログラムを見直したり修正するときにも、流れ図とこのメモを見ればプログラムの各部がどういう処理を行っているかが一目瞭然になります。

グラフィックスを表示したり、マウスやキーボード入力を受け付ける処理や、数式を使った処理などは、いわばコンピュータ処理と直結しているため、プログラムリストを眺めていけば比較的わかりやすいのですが、ゲームの流れやルールにあたる処理はあくまでも人間が決めたことであって、どちらかというとコンピュータの都合に合わせていない部分ですから、それを実現するプログラムは直感的にわかりにくいものになりがちです。ですから、このような図やメモがあとで役に立ちます。

音を鳴らしてみる

このゲームは、それほどきらびやかではありませんが、一応3Dということでグラフィックスが中心のゲームになっています。しかし、もうひとつゲームに彩りを添えるものとして「音」があります。音のないアクションゲームというのも、職場でコソコソ遊ぶ場合を除いては味気ないものです。そこで、このゲームにもささやかながらサウンドを付け加えてみます。

音を鳴らす場面は、ボールがブロックに当たってブロックが消えるときと、空振りをしたときです。もちろん、ゲームクリアのときのファンファーレなんていうのも良さそうですが、今回はとりあえず見送りますので皆さん自身でぜひ加えてみてください。

プログラム上で、音の処理をどのように行うかは**図2**のメモに記述がありました。

ボールがブロックに当たってブロックが消えたときと、空振りをしたときというのは、いずれもBallクラスのprocess()メソッドでの処理ですから、Ballクラスに音を鳴らす機能を付ければよいのですが、残念ながら音を鳴らすメソッドはアプレットクラスに用意されています。そのため、メモにもあるようにBallクラスのprocess()メソッドから、アプレットクラスに値を返して、アプレットクラス側で場面に応じた音を鳴らすようにしています。

Javaアプレットで使える音のデータは、au形式だけです。これは、UNIXで採用されてきた形式です。現在、インターネット上では色々な音データの形式がありますが、Javaアプレットで鳴らすためには、au形式の音データファイルを用意しなければなりません。

今回は、ボールがブロックに当たったときの音をhit.au、空振りをしたときの音をmiss.auというファイルとして用意しました。

音を鳴らすメソッドはアプレットクラスのplay()メソッドで、

```
play(getCodeBase(), "hit.au");
```

というように記述します。

getCodeBase()というのもアプレットクラスに用意されているメソッドで、これは現在実行しているアプレットが置かれているURL（サーバ内のディレクトリ位置）を返します。つまり、これでアプレットと同じディレクトリに

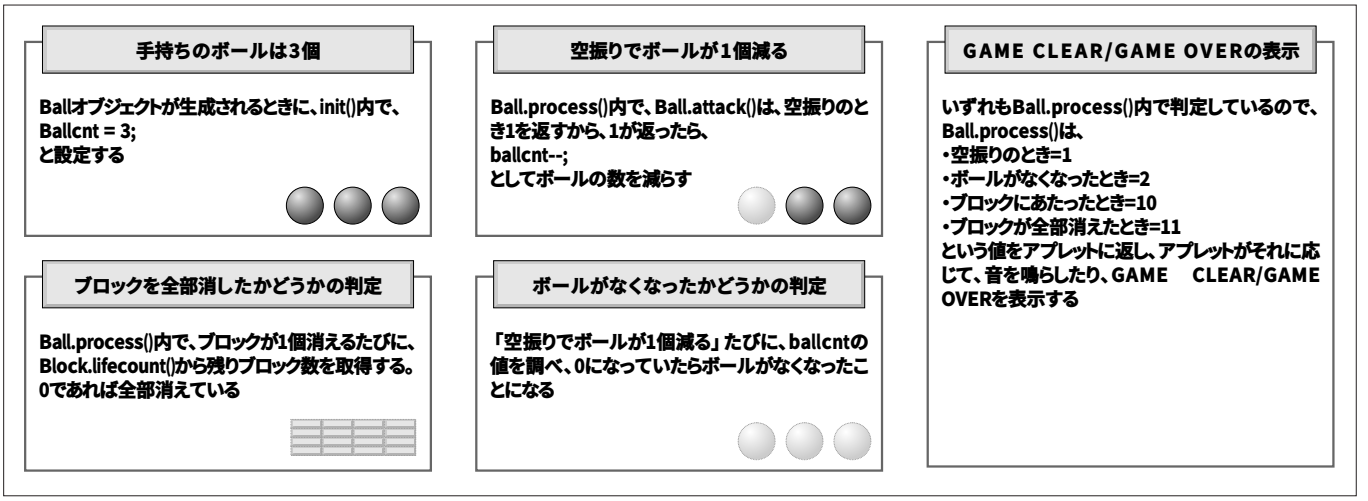


図2　各処理のためのメモ

あるhit.auファイルが再生されます。

音を鳴らすには、この1行の記述だけですみますから簡単なのですが、必ずアプレットクラスのメソッドを使わなければならないのが面倒です。

実際のプログラミング

以上のメモから、実際のプログラミングを考えてみましょう。

リスト1が新しいBallクラスです。例によって、網掛け部分が今回の追加箇所になります。

まず、ゲーム開始時にボールの個数を3個に設定します。コンストラクタ内にある、

```
ballcnt = 3;
```

がその設定です。ballcntという変数は、Ballクラスのメンバ変数として宣言されていますから、Ballクラス内のどのメソッドからでも参照／設定することができますが、Ballクラス外から参照することはできません。ですから、「残りボールがなくなった」という合図は、変数ballcntを使ってではなく、別の方法でアプレットクラスに通知することになります。

次に、**図2**のメモによるとBallクラスのprocess()メソッドで処理することとして、

- **空振りでボールが1個減る**
- **ボールがなくなったかどうか**
- **ブロックに当たった**
- **ブロックを全部消したかどうか**

の4つがあります。

最初の2つはボールをラケットで打つときの処理を行うattack()メソッドの返り値を、あとの2つは、ボールがブロッ

▼リスト1　Ballクラス

```
//---------------------------
// Ball Class
//---------------------------
class Ball extends Disp3D {
        int             vx,vy,vz;
        int     x,y,z;                                  // 3D座標
        int             gx,gy,gr;                       // 2D座標、直径
        int             sgx,sgy,sgr;                    // 影2D座標
        Racket  rc;                                     // ラケットへの参照
        Block   bl;                                     // ブロックへの参照
        int             ballcnt;            // 残りボール数

        Ball(int w, int h, Racket rc0, Block bl0) {
                super(w,h);
                rc = rc0;
                bl = bl0;
                init();

                ballcnt = 3;
        }

        ...
        (省略)
        ...

        int process() {
                int res = 0;

                x += vx;
                y += vy;
                z += vz;

                int lim = 500 - gr / 2;

                if ( x > lim )              vx = -vx;
                if ( x < -lim )             vx = -vx;
                if ( y > lim )              vy = -vy;
                if ( y < -lim )             vy = -vy;

                if ( z > lim )              res = hit();            // ブロック面
                if ( z < 150 )              res = attack();

                //if ( res == 1 )  x = y = z = -9999;   ← 削除

                gx = d2x(x,y,z);
                gy = d2y(x,y,z);
                gr = d2r(100,z);

                sgx = gx;
                sgy = d2y(x,500,z);
                sgr = gr / 2;

                if ( res == 1 ) {                       // 空振り
                        ballcnt--;
                        if ( ballcnt <= 0 ) {
                                vx = vy = vz = 0;
                                res = 2;            // ゲームオーバー
                        }
                        else init();
                }

                if ( res == 10 ) {                      // ブロックに当った
                        if ( bl.lifecount() == 0 ){
                                vx = vy = vz = 0;
                                res = 11;           // ゲームクリア
                        }
                }

                return res;
        }
        ...
        (省略)
        ...
}
```

クに当たったときの処理を行うhit()メソッドからの返り値をそれぞれチェックします。

これらのメソッドからの返り値は、いずれの場合も変数resに格納されています。attack()メソッドは、ボールがうまくラケットに当たったとき0を、空振りしたときは1を返すようになっていますから、

```
if ( res == 1 ) {     // 空振り
  ballcnt--;
  if ( ballcnt <= 0 ) {
    vx = vy = vz = 0;
    res = 2;
  }
  else init();
}
```

という部分で、変数resが1ならば空振りとしてballcntを1減らしています。このとき、もしballcntが0以下になってしまっていたら、それはボールがなくなったということですから、単に空振りしたときとは違ってゲームオーバーにならなければなりません。ゲームオーバーという文字の表示は、ゲーム全体にかかわることなので、Ballクラスが関与すべきことではなく、アプレットクラスで処理すべきです。このため、空振りのときとは異なった合図をアプレットクラスに送ります。ここではメモの「GAME CLEAR/GAME OVERの表示」にあるように、ボールがなくなったときには変数resを2に変えてやります。

その直前にある、

```
vx = vy = vz = 0;
```

は、ゲームオーバーの瞬間、ボールの動きを停止させるためのものです。

ボールがまだ残っているときは（elseの部分）、変数resの値は1のままでinit()メソッドを実行します。init()メソッドは、ボールの位置や速度を最初の状態に戻すために、ちょうど都合がよいからです。

ブロックに当たったときの処理は、同様にhit()メソッドの返り値が変数resに格納されていますから、その値に応じて行います。前回掲載しましたが、hit()メソッドは、

```
int hit() {
  int    res = 0;

  vz = -vz;
  res = bl.hitblock(x,y);
  return(10*res);
}
```

となっています。ここでもまたbl.hitblock()という下請けメソッドに処理をまかせていますが、bl.hitblock()は、ボールがブロックに当たってブロックが消えたとき（または黄色ブロックがグレーに変わったとき）に1を返します。hit()メソッド内の変数resは、そのbl.hitblock()からの返り値を格納していますので、その値をprocess()メソッドに返せばよいのですが、そのまま1を返してしまうと、先程のattack()メソッドからの返り値と区別が付かなくなってしまうので、10を掛けて、ブロックが消えたとき10を、そうでないときは0を返すようにしています。これで、process()メソッドでも、空振りしたときと（resが1）、ブロックが消えたとき（resが10）を区別して処理することができるようになります。

単にボールが消えただけなら、res=10をアプレットクラスに返せばよいのですが、ここでもうひとつ、「ブロックを全部消したかどうか」の判定処理をしておかなければなりません。

そのときの処理は次のようになっています。

```
if ( res == 10 ) {
  if ( bl.lifecount() == 0 ) {
    vx = vy = vz = 0;
    res = 11;
  }
}
```

ブロックが全部消えるのは最後に残ったブロックを消した瞬間ですから、いずれにせよhit()メソッドからの戻り値は10になっているはずです。ですから、まず変数resが10になっているかを判断します。

次に、前回用意だけしておいたbl.lifecount()というBlockクラスのメソッドを使って、残りブロック（life）数を取得します。当然、これが0であればブロックは全部消えたことになりますので、その場合は変数resを10のままではなく11にしてアプレットクラスに返します。あとはアプレットクラスにおまかせです。

最後に忘れてはならないのが、せっかく場合に応じてセットした変数resの値をアプレットクラスに返すために、

```
return res;
```

の1行を付け加えることです。ということは、このprocess()メソッドは、これまでと違って値を返すメソッドということになりますから、process()メソッドの宣言部分を、

```
int process() {
```

に変更しておきます。

アプレットクラスの処理

メモに書いたうちの各種判定処理はすべてBallクラスに用意できましたが、それらの場面によって音を鳴らしたり、文字を表示したりするのは、前述のようにアプレットクラスで行わなければなりません。アプレットクラスの追加／修正箇所が**リスト2-1・2-2**です。

run()メソッドを見てください。

ball.process()メソッドからの返り値を、ここでもまた変数resに格納します。ところで、これまでBallクラスのprocess()メソッド、hit()メソッド、そしてこのアプレットクラスのrun()メソッドに、resという同名の変数がでてきましたが、これらはいずれも各メソッド内部で、

```
int  res;
```

と宣言されていることに注意してください。つまり、それぞれの変数resは名前こそ同じですが、いずれも別のもので、それぞれ各メソッド内だけで有効なものなのです。仮にBallクラスのprocess()メソッド内の変数resの値を変更しても、アプレットクラスのrun()メソッドの変数resはまったく影響を受けません。

このように、メソッド内で宣言された変数（ローカル変数）は、お互い干渉することなく同じ名前を付けることができます。そうでないと、同じ役割の変数なのにそれぞれ違った名前を付けなくてはならず、混乱の元になってしまいます。

さて、run()メソッドの変数resは、bl.process()メソッドの返り値である、0、1、2、10、11のいずれかの値になっていることになります。それぞれの値の意味はメモに記述があります。

順に見て行くと、空振り（res=1）のときは、空振り用の音を鳴らして少しゲームを止めます。

```
play( getCodeBase(), "miss.au");
Thread.sleep(1000);
```

sleepは1000ミリ秒だから1秒です。

次に、変数resが2のときはボールがなくなったのですから、ゲームオーバーということになり、空振りの音を鳴らしたあと、画面に「GAME OVER」という文字を表示します。

```
if ( res == 2 ) {
```

以降が、そのときの処理です。ここで行う処理は、まず、アプレットの画面上のサイズを取得しています。

```
Dimension d = getSize();
```

これで、d.width、d.heightがそれぞれアプレットの画面上の高さ、幅になります。

次に、Stringオブジェクトsを生成して、「GAME OVER」という文字列をsに格納しておきます。

```
String s = "GAME OVER";
```

その次の行にある、

```
Font f = new Font("Serif",
    Font.BOLD, 48);
g.setFont(f);
```

は、文字を表示する際のフォントの設定を行っています。いったん、セリフ体、太字、48ポイントという属性で、fというフォントオブジェクトを生成して、Stringオブジェクトsのフォントを

▼リスト2-1　Appletクラス

```
//---------------------------
// Applet Class
//---------------------------
public class block3d extends Applet implements Runnable,MouseMotionListener  {

        Thread          t;
        Ball            ball;
        Background      bg;
        Racket          rc;
        Block           bl;
        Image           buffer;
        Graphics        bufferg;
        int             sleepval;

        public void init() {
                addMouseMotionListener(this);

                sleepval = Integer.valueOf(getParameter("sleep")).intValue();
                Dimension d = getSize();

                buffer = createImage(d.width, d.height);

                rc = new Racket(d.width, d.height);
                bl = new Block(d.width, d.height);
                ball = new Ball(d.width, d.height,rc,bl);
                bg = new Background(d.width, d.height);

                t = new Thread(this);
                t.start();
        }
```

画面サイズとsleep時間をパラメータとして取得

fの属性になるように設定しています。

さらに、「GAME OVER」という文字を画面の中央に表示するために、

```
FontMetrics fm =
    g.getFontMetrics();
int x = d.width/2 -
    fm.stringWidth(s)/2;
int y = d.height/2 +
    fm.getDescent();
```

という処理を行っています。文字を画面の中央に表示するには、先程のアプレットの画面サイズと文字列の画面上での長さが必要になります。

文字列の画面上の長さを計算するためには、FontMetricsというオブジェクトが必要になるので、変数fmに現在の画面gのFontMetricsへの参照を取得します。すると、fm.stringWidth()で文字列の長さ、fm.getDecent()で文字

▼リスト2-2　Appletクラス（続き）

```
        public void run() {
                int             res;
                Graphics        g = getGraphics();

                try {
                        while(true) {
                                res = ball.process();                    ball.process()の返り値による処理

                                if ( res == 1 ) {                                    // 空振り
                                        play( getCodeBase(), "miss.au");
                                        Thread.sleep(1000);
                                }

                                if ( res == 2 ) {                                    // ゲームオーバー
                                        play( getCodeBase(), "miss.au");
                                        Dimension d = getSize();
                                        String s = "GAME OVER";
                                        Font f = new Font("Serif", Font.BOLD, 48);
                                        g.setFont(f);
                                        FontMetrics fm = g.getFontMetrics();
                                        int x = d.width / 2 - fm.stringWidth(s) / 2;
                                        int y = d.height / 2 + fm.getDescent();

                                        g.setColor(Color.magenta);
                                        g.drawString(s,x,y);
                                        break;
                                }

                                if ( res == 10 ) {                                   // ブロックに当たる
                                        play( getCodeBase(), "hit.au");
                                }
                                if ( res == 11 ) {                                   // ゲームクリア
                                        play( getCodeBase(), "hit.au");
                                        Dimension d = getSize();
                                        String s = "GAME CLEAR";
                                        Font f = new Font("Serif", Font.BOLD, 48);
                                        g.setFont(f);
                                        FontMetrics fm = g.getFontMetrics();
                                        int x = d.width / 2 - fm.stringWidth(s) / 2;
                                        int y = d.height / 2 + fm.getDescent();

                                        g.setColor(Color.orange);
                                        g.drawString(s,x,y);
                                        break;
                                }

                                repaint();
                                Thread.sleep(sleepval);  ——  パラメータによる待ち時間設定
                        }
                }
                catch(Exception e) {}
        }
        ...
        (省略)
        ...
}
```

列の高さが取得できます。画面上のサイズは、先ほど説明したように、d.with、d.heightでわかりますので、それらの値を使って文字をアプレットの中心より少し上に配置するための座標を計算しています。

最後に、

```
g.setColor(Color.magenta);
g.drawString(s,x,y);
break;
```

で、文字の色設定をして文字を表示します。ゲームオーバーのときは、アプレットを終了させるため、break;と書くことによって、whileループから抜け出します。これにより、run()メソッドが終了し、そのままアプレットも終了します。

ボールがブロックに当たって、最終的にブロックが全部消えたとき（res=10、11のとき）も、ほとんど処理は同じです。音のファイル名と、表示する文字と色が変わるだけです。ただし、ボールがブロックに当たったときは、空振りのときとは違って、アプレットをsleepさせてしまうとゲームが一時停止してしまうので、sleepは入れないようにしてください。

このあたりはゲームの味付けですから、どんな文字（絵？）をどこに表示させるかなどは、皆さんの好みに合わせて変更してください。

外部から値を取り込む

今回のリストでは、もうひとつ新しい処理をしています。これまで、アプレットの画面上のサイズやスレッドがシステムに制御を返す時間、すなわちこれがゲームのスピードに関わってくるのですが、これらの値はプログラム中に数値として記述していました。この方法だと、アプレットの画面サイズを変更したり、ゲームのスピードを変えたりする場合に、その都度プログラムを書き換えてコンパイルし直さなければなりません。

そこで、プログラムを書き換えることなく、それらの値を外部から取り込むことができるようにしてみました。

Javaアプレットは通常、HTMLファイルに<applet>タグによって埋め込まれます。<applet>タグには、アプレットの画面上のサイズをパラメータとして記述することができます。また、それ以外の値も<param>タグを使うことによって、HTMLファイルに記述された数値や文字データをアプレットに渡すことができるのです。

<param>タグを使って、このアプレット用のHTMLファイルを作ってみます（**リスト3**）。

アプレットの画面上のサイズは、<applet>タグ内の、width、heightの2つのパラメータで指定します。そして、Thread.sleep()の値は、

```
<param name="sleep" value="40">
```

として、<param>タグを使って記述しています。もちろん、HTMLファイルにこう記述するだけで、アプレットがパラメータを取り込んでくれるわけではなく、プログラムにもそれなりの準備が必要です。

リスト2-1のinit()メソッドがそのからくりです。

```
sleepval =
  Integer.valueOf(getParameter(
  "sleep")).intValue();
```

という少し長い行が、<param>タグで記述されたパラメータを取り込む仕掛けです。この中の、getParameter()に与える引数が、<param>タグ内のnameの内容と一致していれば（ここでは"sleep"）、タグ内のvalueの値を取り込みます。そのままだと文字列扱いになってしまいますので、ここでは、Integer.valueOf()というメソッドを使って、整数に変換しています。

あとは、run()メソッド内の、

```
Thread.sleep(40);
```

であったところを、

```
Thread.sleep(sleepval);
```

に変更すればOKです。

これで、ゲームのスピードは固定ではなくなり、HTML内の<param>タグの値を書き換えれば、アプレットを変更することなくゲームスピードを変

▼リスト3　HTMLの例

```
<html>
<title></title>
<body>

<applet code="block3d" codebase="." width=400 height=400>
<param name="sleep" value="40">
</applet>

</body>
</html>
```

えられるようになりますので、ぜひ実験してみてください。

アプレットの画面上のサイズはもう少し簡単に、同じくinit()メソッドの、

```
Dimension d = getSize();
```

で、取り込むことができます。dはDimensionという型の変数として宣言していますので、d.widthで幅、d.heightで高さの値として使うことができます。

これにより、前回までは(400,400)と数値を直接記述していた部分を、(d.width,d.height)と変更することで、<applet>タグのwidth、heightの記述に対応することができます。先ほど、「GAME OVER」と表示する処理の部分で使用しました。ただし、このアプレットでは一応サイズ変更には対応しているのですが、奥行きの計算（z方向）で一部数値を直接使っているために、手前のほうでボールの動きが少し変になってしまいます。ここはぜひ皆さん自身で直してみてください。

あとは……

以上で、この連載のJavaアプレットプログラミングは終了です。Javaはなかなか奥の深いものですから、短い連載だけですべてを解説することはできませんでしたが、基本的なアプレットの動きや、プログラミングの方法、そしてなによりも初めて作るプログラムとしてはめずらしく、3Dゲームという題材に挑戦してみました。

筆者としてはゲーム自体も結構楽しめるもので、下手をするとハマってしまう大人の（？）ゲームだとひそかに自負しているのですが、この連載の最大の目的はなにもゲームで遊ぶことではありません。

どんなに面白いゲームや、興味深いソフトウェアがあったとしても、そのプログラムソースがなければ手を加えることはできませんし、そのプログラムを十分理解していなければ思うような改変はできません。

その点、もし皆さんがこの連載に辛抱強く付きあってくれていて、なおかつ毎回掲載しているプログラムを実際に作って実行していたとするなら、このゲームだけは、今までユーザーとして遊んできた数々のゲームとは違って、仕組みやからくりがすべて皆さんの頭の中にあることになります。

つまり、筆者の考え出したこのゲームにご不満がおありなら、ただちに自分の好みに合わせて、またクリアするのをもっと難しくしたり、やさしくしたりすることが自由自在になるはずです。もちろん、どこをどう書き換えればよいか、すぐに思いつかないときもあるでしょうが、そこはカットアンドトライ、まちがって爆発するものでもないのですから、書き換えては実行し、書き換えては……と、とにかく実験してみるのです。深く考え込んだり、参考書と首っ引きになるよりも、そのほうがずっと上達することまちがいなしです。そうしていくうちに、いつのまにかどんどんプログラミングをマスターしていくものです。プログラミングはまさに「習うより慣れろ」です。

Javaの将来

Javaは非常に洗練された高度なプログラミング言語です。現在、世の中で実用的に使われている言語の中でも最も新しく、ゆえに最近および将来のコンピュータシーンに合わせて開発された言語です。

これまでにも何度か触れたように、Javaはプラットフォーム非依存、ネットワークにネイティブに対応、ポータブル端末からPC、サーバシステムにいたるまでの対応力を持っています。まさに、これからのコンピュータシステムに最適な言語であることはまちがいないでしょう。

また、なによりも読者の皆さんにとっての恩恵とは、手軽にプログラミングが経験できることです。小さなアプレットから大きなアプリケーションまで、ホームページのちょっとした飾りから、本格的なネットワークシステムやサーバアプリケーションに至るまで対応できるのがJavaなのです。

本格的言語でありながら、初心者にとっても敷居の高いものではなく、高価な開発環境などがなくても、フリーの環境からはじめることができる。そんなプログラミング言語はほかにはちょっと見あたらないでしょう。

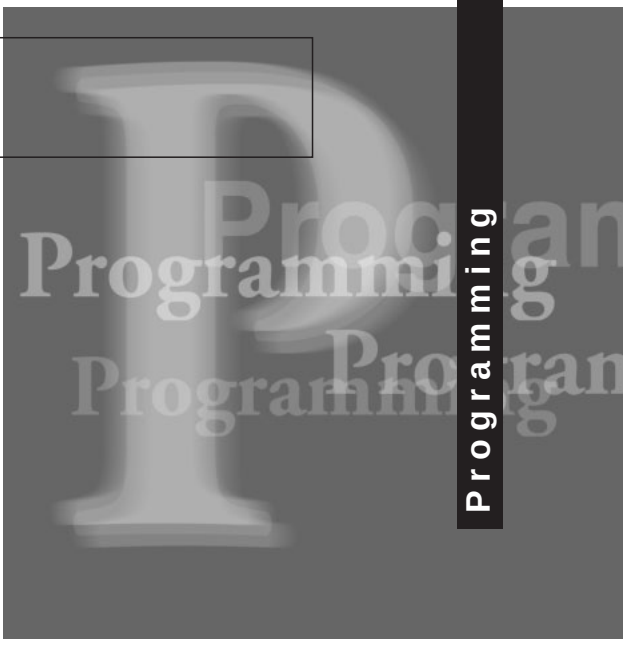
Webベースのアプリケーションや、ASP（Application Service Provider）の採用、XMLの普及やBluetooth[*1]の登場など、コンピュータシステムが今後、ネットワークベースで利用される世の中になっていくであろうことは疑う余地がありません。

このような将来を考えれば、きっとJavaのマスターが、皆さんそれぞれの目的のために役立つことと思います。

この連載が、そんなJavaの世界へのきっかけになればとてもうれしい限りです。

[*1]Bluetooth　低出力の無線を利用した通信技術で、室内など短距離にある機器同士で音声やデータを暗号化して通信するための規格。ケーブルレスでプリンタや、インターネットに接続したりするなどの目的に利用される。

プログラミング工房



プログラムを作成する場合、作成過程のバージョンを保存することは重要だ。ドキュメントやサーバの設定ファイルなどのことを考えても、履歴をきちんと管理できるツールの存在はありがたい。そこで、今月はバージョンを管理するシステムとして主流になっているCVSについて解説する。

第13回　開発過程のバージョン管理

文：藤沢敏喜
Text : Toshiki Fujisawa

バージョン管理の重要性

ソフトウェアを開発するためには、かなりの時間を必要とする。特に、大規模なソフトウェアになればなるほど時間がかかり、たくさんのバグが入り込むことになる。

大きなプログラムでは、開発をしていく過程で、プログラムが正常に動作するかどうかをチェックしていくのが普通である。その場合、前の日には動作していた機能が、今日追加したコードにより動作しなくなる、ということがよくある。このバグを修正するには、変更した部分を元に戻せばいいのであるが、普通の人間の記憶力はいい加減なものなので、前日の変更内容すら思い出すことができない場合が多い。

このようなときに古いバージョンに戻れないと、そのバグを直すのに多大な労力を要することになる。そして最悪の場合は、道に迷って遭難してしまう登山者のように、どんどん深みにはまっていって、最後には手がつけられなくなることも多い。

山の頂上を目指して道に迷った場合、迷いながら頂上を目指すことは極めて難しく危険である。確実な道であることをチェックしたポイントまで戻ってから、そこから再び頂上を目指すのが賢明な方法である。

山頂を目指すのと同じように、プログラムの開発でも確実に動作していたバージョンまで、正しく戻れるようにしておくことは重要である。この目的のためには、開発途中のバージョンのすべてのファイルを保存しておけばよいのだが、これは結構たいへんな作業だ。

たとえば、Linuxカーネルのように、ソースファイルが合計で20Mバイトもあるような場合、1時間あたり10回の変更をするとしても、10時間で2Gバイトが必要となる。これを30日続けると、1カ月で60Gバイトになる。80Gバイトのハードディスクが3万円で買える時代とはいえ、これだけの容量を保存しておき、きちんとバックアップすることは不可能に近い。たとえ保存できたとしても、どの時点で、だれが、どんな変更をしたかを確認することも難しい。

そのため、変更した部分の差分情報のみを記録して、統一的に管理するシステムが必要になってくる。

CVS（Current Version System）

バージョン管理をするためのソフトウェアとして、最も多く使われているのがCVS（Current Version System）である。1990年頃にはネットワーク対応などの機能拡張が行われ、現在も開発が続いている。

初期のCVSは、RCS（Revision Control System）と呼ばれる履歴管理システムを発展させたものである。RCSは、バージョン管理をするのに、当時としてはたいへん便利なシステムであった。筆者も、昔はMS-DOS上でRCS

を利用していた記憶がある。

しかし、RCSはあくまでもひとつのファイルに関する履歴を管理するツールであり、複数のファイルが多数のディレクトリに置かれているプロジェクトのバージョン管理をするにはあまり向いていない。また、ネットワークにも対応していなかったため、複数の開発者でRCSを使う場合は、NFSなどでディスク共有を行う必要があった。

また、RCSでは、ひとつのファイルを取り出すと、そのファイルに「ロック」がかかるようになっていて、そのファイルをしまわない限り、他人がファイルを編集することはできないという使いにくさがあった。

CVSは、このような欠点を解消すために、次のような特徴を備えている。

- **多数のファイルやディレクトリを一括管理可能**
- **ネットワークに対応**
- **「ロック」の必要がなく、多人数による同時編集が可能**

このように、CVSはRCSの弱い点を補強し、大規模なプロジェクトでの使用が可能なので、広く用いられるようになってきている。

CVSを使っている大規模な開発例としては、Apache（Webサーバ）やGNOME（デスクトップ環境）、そしてFreeBSDやNetBSDなどのオペレーティングシステムがあげられる。

CVSは、このような大規模なプロジェクトで利用できるだけの機能と安定性を備えたプログラムである。しかしながら、多人数での開発をサポートするための機能がたくさんあり、その機能を全部理解することは結構難しい。

そこで、今回は自分1人だけの履歴を管理するための、最小限必要な操作だけを説明してみる。

CVSを使うための準備

■CVSが使えるかどうかの確認

CVSは、最近のディストリビューションであれば、最初から入っていることが多い。CVSが使えるかどうかを確認するには、コマンドプロンプトから「cvs」というコマンドを実行してみればよい。CVSがインストールされている場合は、CVSの使い方が表示されるはずだ（**画面1**）。また、どのバージョンのCVSがインストールされているかは、「cvs --version」を実行すればよい。

```
$ cvs
Usage: cvs [cvs-options] command [command-options-and-arguments]
  where cvs-options are -q, -n, etc.
    (specify --help-options for a list of options)
  where command is add, admin, etc.
    (specify --help-commands for a list of commands
     or --help-synonyms for a list of command synonyms)
  where command-options-and-arguments depend on the specific command
    (specify -H followed by a command name for command-specific help)
  Specify --help to receive this message

The Concurrent Versions System (CVS) is a tool for version control.
For CVS updates and additional information, see
    Cyclic Software at http://www.cyclic.com/ or
    Pascal Molli's CVS site at http://www.loria.fr/~molli/cvs-index.html
$ cvs --version

Concurrent Versions System (CVS) 1.10 `Halibut' (client/server)

Copyright (c) 1989-1998 Brian Berliner, david d `zoo' zuhn,
                        Jeff Polk, and other authors

CVS may be copied only under the terms of the GNU General Public License,
a copy of which can be found with the CVS distribution kit.

Specify the --help option for further information about CVS
$
```

画面1　CVSの画面

■リポジトリの初期化と環境変数の設定

ファイルをCVSで管理するためには、ファイルの変更点情報を保存する場所が必要になる。この保存しておく場所を「リポジトリ」と呼ぶが、ここではテストのため、

```
/home/fujisawa/cvs-rep
```

に置くことにする。リポジトリの初期化を行うには、以下のコマンドを実行する。

```
$ cvs -d /home/fujisawa/cvs-rep init
```

これにより、指定したリポジトリの中に自動的にCVSROOTという名前のディレクトリが作成され、その中にCVSの管理ファイルが生成される。

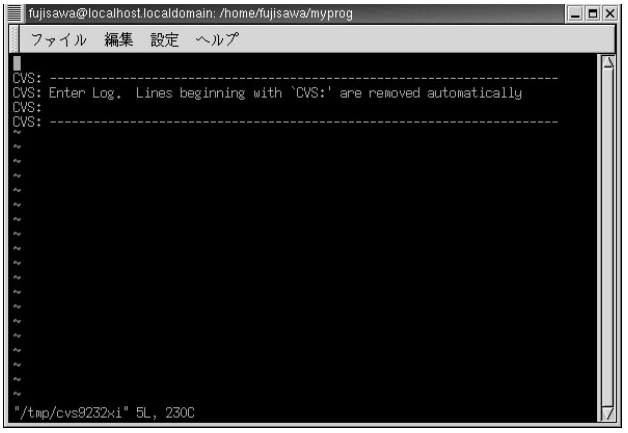
このリポジトリは、CVSROOTという名前の環境変数に登録する必要があるので、~/.bash_profile などに、

```
export CVSROOT="/home/fujisawa/cvs-rep"
```

と記述しておく。また、CVSでは、履歴コメントを記述するために外部のエディタを起動するため、EDITORまたはCVSEDITORという環境変数も定義しておく必要がある。もちろん自分の好きなエディタを指定してよいのであるが、ここでは説明の都合上viを使うことにするので、以下のようにしておく。

```
export EDITOR=vi
```

としておく（ただし、CVSEDITORを設定していると、そちらが優先される）。



画面2　cvsコマンドによってviが自動起動する

■CVSに格納するテストファイルの用意

それでは、CVSを使う練習をするためにテスト用のファイルを用意してみよう。

まず、自分のホームディレクトリにmyprogというディレクトリを作成し、その中にたとえば**リスト1**に示すような3つのファイルを用意する。

■管理するファイルの格納

ここまで用意ができたら、先ほどの3つのファイルを「/home/fujisawa/cvs-rep」に格納してみる。実行するコマンドは次の通りである。

```
$ cd /home/fujisawa/myprog
$ cvs import myprog v-tag r-tag
```

ここで、v-tag r-tagについては、とりあえずは適当な文字列を指定していても問題はない。

このコマンドを実行すると、自動的にviが起動して、**画面2**のような内容が表示される。ここでは最初の行が空になっているので、その行に適当なコメントを書いてエディタを終了する。なお、CVS: で始まる行は自動的に削除され、コメントとしては登録されないので、そのままにしておいてかまわない。

エディタを終了すると、以下のように、3つのファイルが正常に登録されたことを示す表示が行われる。最後の行は、このインポート作業では、競合（conflict）状態が発生しなかったことを意味する。

▼リスト1　CVS練習用の3ファイル

```
=== myprog/main.c ===
extern void disp(void);
int main(void)
{
        disp();
}

=== myprog/Makefile ====
myprog: main.c func/disp.c
        cc -o myprog main.c func/disp.c

=== myprog/func/disp.c ===
#include <stdio.h>

void disp(void)
{
        printf("Hello World\n:");
}
```

```
N myprog/main.c
N myprog/Makefile
cvs import: Importing /home/fujisawa/cvs-rep/myprog/func
N myprog/func/disp.c

No conflicts created by this import
```

ファイル名の左横に表示される“N”はNewの略で、そのファイルが新規にリポジトリに追加されたことを示す。

また、このインポート作業を行うときには、カレントディレクトリを、格納するファイルがあるディレクトリにしておかなければならないことに十分注意してほしい。

■ファイルの取り出し

CVSでは、格納したデータをだれでも取り出すことができる。取り出すときには、CVSの管理情報が出力される。追加や削除を行う場合は、この管理情報を基にするため、変更する前には、必ず取り出し（checkout）作業を行う必要がある。

先ほどのインポート作業では、単に「格納」しただけであるので、ソースコードを編集する前には必ず取り出し作業を行う。実験のため、インポートした元のディレクトリは、いったん消去する。なお、消すのが不安な場合は、「mv myprog myprog.old」などとしておくとよいだろう。

Column

diffとpatchコマンド

ファイルの差分を出力するツールとしては、diffが有名である。たとえば、次のような2つのファイルがあったとする。

```
(file-1.txt)
aaaaa
bbbbb
ccccc
ddddd
eeeee

(file-2.txt)
aaaaa
ccccc
DDDDD
eeeee
fffff
```

このとき、コンテキスト形式を指定するために、diffに「-c」オプションを付けて、

```
$ diff -c file-1.txt file-2.txt
```

というコマンドを実行すると、次のような出力が得られる。

```
*** file-1.txt  Mon Nov  6 15:26:23 2000
--- file-2.txt  Mon Nov  6 15:27:32 2000
***************
*** 1,5 ****
  aaaaa
- bbbbb
  ccccc
! ddddd
  eeeee
--- 1,5 ----
  aaaaa
  ccccc
! DDDDD
  eeeee
+ fffff
```

ここで、「＊＊＊」がfile-1.txtの、「－－－」がfile-2.txtの内容であることを示している。

「＊＊＊ 1,5 ＊＊＊＊」は、以下に続く内容が、file-1.txtの1行目から5行目であることを示している。

同様に、「－－－ 1,5 －－－－」から続く5行は、file-2.txtの1行目から5行目である。

また、行頭の「－」はその行が削除されたことを意味し、「+」はその行が追加されたことを示している。また「!」は、その行が置き換わったというしるしである。

したがって、この例ではfile-1.txtからfile-2.txtを作成するためには、bbbbbの行を削除し、dddddの行をDDDDDに置き換え、fffffの行を追加するということが示されていることになる。

さて、file-1.txtからfile-2.txtを作成するにはどうしたらよいだろうか? 今回の例では、手で直しても十分だが、数が多くなるとたいへんな作業になる。

そこで登場したのがpatchコマンドである。このコマンドは、Perlの作者としても有名なLarry Wall氏によって書かれたもので、きわめて賢い動作をする。

このコマンドを使うと、次のようにして、古いファイルと差分ファイルから新しいファイルを作成することが可能になる。

```
$ diff -c file-1.txt file-2.txt > file.diff
$ mv file-2.txt file-2.org
$ patch < file.diff
Hmm...  Looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
*** file-1.txt Mon Nov  6 15:26:23 2000
--- file-2.txt Mon Nov  6 15:27:32 2000

--------------------------
Patching file file-1.txt using Plan A...
Hunk #1 succeeded at 1.
done
```

なお、diffとpatchには、たくさんのオプションがあるので、マニュアルを見ながらいろいろと試してみるとよいだろう。

```
$ rm -r /home/fujisawa/myprog
```

次に、

```
$ cd /home/fujisawa
$ cvs checkout myprog
```

というコマンドを実行すると、次のようなメッセージが表示され、リポジトリに格納されているファイルを新しいmyprogディレクトリに取り出すことができる。

```
cvs checkout: Updating myprog
U myprog/Makefile
U myprog/main.c
cvs checkout: Updating myprog/func
U myprog/func/disp.c
```

ファイル名の左横に表示される“U”はUpdateの略で、そのファイルが正常にアップデートされたことを示す。

CVSを使ったバージョン管理作業

ここまでで、CVSを使うための準備はすべて整ったことになる。あとは、修正したファイルを格納する方法と、以前のバージョンのファイルの取り出し方法を覚えれば、CVSを活用することが可能になる。

■修正したファイルの格納方法

さて、いよいよプログラムソースコードの修正を行ってみる。まず、

```
$ cd /home/fujisawa/myprog
$ vi main.c
```

として、先ほど取り出したmyprog/main.cの4行目に、コメントを追加してみる。

```
extern void disp(void);
int main(void)
{
     disp(); /* display "Hello World" */
}
```

そして、このファイルを保存して、viを終了する。

この修正内容をリポジトリに格納するためには、修正したファイルがあるディレクトリで、下記のコマンドを用いるだけである。

```
$ cvs commit
```

このコマンドを実行すると、viが自動的に起動し、**画面3**のようなコメント入力画面になる。この画面では、main.cがモディファイされたことが示されているが、複数

Column

サーバ運用と原稿執筆におけるCVSの活用

CVSは、使い慣れると非常に便利なシステムで、手放せなくなるツールである。特に、プログラムソースの管理にCVSは必須であり、CVSなしでのソフトウェア開発はありえないと筆者は感じている。

しかしながら、CVSはプログラムのバージョン管理だけでなく、各種のドキュメント管理や、サーバの設定ファイルの履歴管理にも有効だ。

たとえば、DNSサーバを運用していくうえで、DNSデータベースファイルの履歴を保存しておくと、トラブル時に非常に役に立つ。さらに、数十台のサーバを管理しているような場合でも、それぞれのサーバの/etcにある各種の設定ファイルを、ひとつのCVSサーバで一括して履歴保存することも可能だ。

また、各種文書の履歴管理にも、CVSは欠かせない。もちろん筆者が書いているこの原稿も、毎号CVSでバージョン管理をしている。原稿を書く場合には、構成の都合上いったん書いた部分をバッサリと削ることがあるが、あとになってその部分を復活させなければならないこともよくある。こういった場合、CVSに入れておけば、いつでも昔のバージョンを取り出せるため、気軽に削除ができるのでたいへん気楽だ。

さらに、CVSで管理してあると、自動的にバックアップされるというメリットもある。この原稿は、muleを使い、EUC漢字コードで横26文字改行として書いているのだが、編集部に渡す際には自作のスクリプトで所定の書式に変換している。あるとき、このスクリプトにバグがあって、原稿ディレクトリのファイルをすべて消してしまった。lsコマンドを実行して、次の行にプロンプトだけが返ってきたときには、何が起こったのかわからず、頭の中が真っ白になってしまった。

しかしながら、ラッキーなことにそのスクリプトの先頭では、「cvs commit -m ...」というCVSにすべてのファイルを格納するコマンドが実行されていた。このため、原稿を1行も失うことなく、回復が可能だったのである。この日は締め切り前日だったこともあり、CVSのありがたさをしみじみと感じたのであった。

のファイルを編集した場合には編集したすべてのファイルが表示されることになる。

変更したファイルが正しいかどうかを確認したあとは、1行目に、

```
Add a comment for disp().
```

というようなコメントを記述して、viを終了する。viを終了すると、

```
Checking in main.c;
/home/fujisawa/cvs-rep/myprog/main.c,v <-- main.c
new revision: 1.2; previous revision: 1.1
done
```

というメッセージが出力され、現在のmain.cがリビジョン1.2として正常に登録されたことが示される。

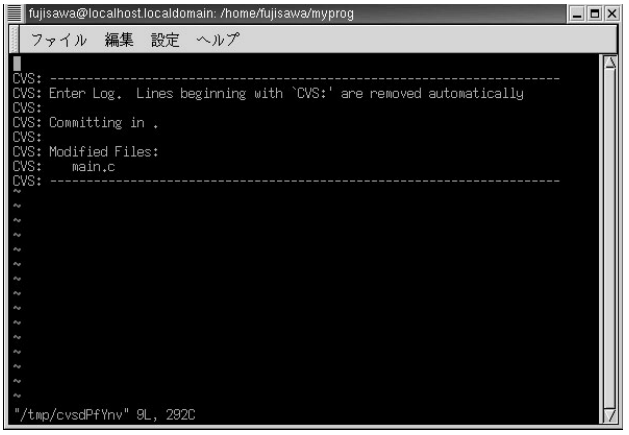
なお、コミットする際に、viがいちいち起動するのが面倒な場合は、-m オプションを使って、コメントを記述する方法もある。

```
$ cvs commit -m "Add a comment for disp()"
```

■新しいファイルやディレクトリの追加と削除

ここで、myprog/doc/readme.txtというファイルを追加してみる。まず最初に、

```
$ cd /home/fujisawa/myprog
$ mkdir doc
$ echo "Please read me" > doc/readme.txt
```



画面3　修正内容をリポジトリに格納する際のコメント入力画面

として新しいファイルを作成する。そして、

```
$ cvs add doc
```

とすると、

```
Directory /home/fujisawa/cvs-rep/myprog/doc added to
the repository
```

というメッセージが出力される。さらに、

```
$ cvs add doc/readme.txt
```

を実行すると、

```
cvs add: scheduling file `doc/readme.txt' for
addition
cvs add: use 'cvs commit' to add this file
permanently
```

というメッセージが出力され、doc/readme.txtというファイルが、追加スケジュールに加わったことがわかる。

次に、func/disp.cを消去してみよう。以下のコマンドを実行すると、カレントディレクトリで削除されたファイルが自動的に検出され、メッセージが表示される。

```
$ cd /home/fujisawa/myprog/func
$ rm disp.c
$ cvs remove
cvs remove: Removing .
cvs remove: scheduling `disp.c' for removal
cvs remove: use 'cvs commit' to remove this file
permanently
```

これで、doc/readme.txtの追加とfunc/disp.cの削除が、スケジューリングされたことになる。この段階で先ほどの-mオプションで、'add and del'というコメントを追加する。

```
$ cvs commit -m 'add and del'
```

このコマンドにより、上記のdoc/readme.txtがリポジトリに追加され、func/disp.cがリポジトリから削除され

る。ここで、リポジトリからファイルを削除しても、削除した以前の履歴は残っているため、いつでも元に戻すことが可能である。

なお、ファイル名やディレクトリ名を変更したいこともある。これにはいくつか方法があるが、削除と追加を行うのが最も簡単で安全な方法である。

ちなみに、CVSではリポジトリからディレクトリがあった痕跡を完全削除することはできないようになっている（そこにあったファイルを戻したいことがあるため）。

ディレクトリをワークディレクトリから削除したい場合は、中にあるすべてのファイルを削除したうえで、チェックアウト時に「-P」というオプションを付けて、空のディレクトリを作成しないようにするとよい。

■過去のバージョンとの比較

上記でリポジトリにコミットしたmain.cの過去の履歴を確認するには、

```
$ cd /home/fujisawa/myprog
$ cvs log main.c
```

を実行すればよい。このコマンドの結果は**画面4**のようになる。ここでは、revision 1.2とrevision1.1があり、revision1.2のコミット時には、

```
Add a comment for disp()
```

というコメントが書かれていることがわかる。なお、ここで表示されている時間はGMT（世界標準時）であることに注意が必要である。

このログを見てリビジョン番号を確認したら、次のようなコマンドを使って、その違いを調べることができる。

```
$ cvs diff -c -r 1.1 -r 1.2 main.c
```

ここで、「-c」はコンテキスト形式のdiff出力を行う指定である。また「-r」は、リビジョン番号を指定するオプションである。つまり、上記のコマンドでは、main.cのリビジョン1.1とリビジョン1.2の違いを出力することになる。この出力は、今回の例では**画面5**のようになる。

なお、この出力の読み方については、「diffとpatchコマンド」の**コラム**を参照してほしい。

```
$ cvs log main.c

RCS file: /home/fujisawa/cvs-rep/myprog/main.c,v
Working file: main.c
head: 1.2
branch:
locks: strict
access list:
symbolic names:
        r-tag: 1.1.1.1
        v-tag: 1.1.1
keyword substitution: kv
total revisions: 3;     selected revisions: 3
description:
----------------------------
revision 1.2
date: 2000/11/05 16:03:03;  author: fujisawa;  state: Exp;  lines: +1 -1
Add a comment for disp()
----------------------------
revision 1.1
date: 2000/11/05 14:11:54;  author: fujisawa;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2000/11/05 14:11:54;  author: fujisawa;  state: Exp;  lines: +0 -0
import-comment
```

画面4　main.cの過去の履歴表示

■タグ（バージョン名称）の設定

いま説明したように、各ファイルには自動的にリビジョン番号が割り当てられるが、この番号はファイルごとに違う。このため、ソースツリーのすべてに共通なバージョンを識別する文字列を設定できるようになっていて、これをCVS用語では「タグ」と読んでいる。

現在のワークディレクトリに「RELEASE-1_0」というタグ（識別文字列）を付けるのは簡単で、

```
$ cd /home/fujisawa/myprog
$ cvs tag RELEASE-1_0
```

というコマンドを用いればよい。このコマンドを実行したあとは、上で説明した「-r」オプションによるリビジョン番号の指定に、このタグを指定することが可能になる。

```
Index: main.c
===================================================
RCS file: /home/fujisawa/cvs-rep/myprog/main.c,v
retrieving revision 1.1
retrieving revision 1.2
diff -c -r1.1 -r1.2
*** main.c      2000/11/05 14:11:54     1.1
--- main.c      2000/11/05 16:03:03     1.2
***************
*** 1,5 ****
  extern void disp(void);
  int main(void)
  {
!       disp();
  }
--- 1,5 ----
  extern void disp(void);
  int main(void)
  {
!       disp(); /* disp "Hello World" */
  }
```

画面5　cvsによって表示された1.1と1.2の違い

■過去バージョンの取り出し

さて、最後に過去バージョンの取り出しについて説明しよう。main.cのリビジョン1.1を取り出し、main.c-rev.1.1というファイルを作成するためには、次のコマンドを用いればよい。

```
$ cvs -Q update -p -r 1.1 main.c > main.c-rev1.1
```

ここで、「-Q」は冗長なメッセージの出力を抑制するオプションで、「-p」は、過去のバージョンを標準出力に送ることを指示するオプションである。

なお、-pを付けないで「cvs update -r xxx」と行うと、古いバージョンを誤ってコミットしないように「sticky tag」と呼ばれるものが貼り付き、コミットできなくなることがあるので注意が必要である。もし貼り付いてしまった場合には、「update -A」オプションでタグを剥がす必要があるが、ここでは説明するスペースがないので、CVSマニュアルの第4章を参照してほしい。

おわりに

CVSはたいへん便利なツールではあるが、使い方に難しい面があるためか導入をためらう場合が多いようである。

しかしながら、CVSというツール自体が難しいというわけではなく、大規模に多人数で開発を行い、複雑に枝分かれしたバージョン管理をするという行為そのものがたいへん難しいのである。

今月号で具体的に説明したように、CVSを1人だけで用いる分には複雑なことはほとんどなく、簡単に用いることができる。一度使ってその便利さを知れば、CVSなしにプログラムや文章を書くことは考えられなくなるはずである。

Column

CVSの参考文献

CVSを使ううえで参考になるインターネット上の情報については、京都工芸繊維大学の西本さんのページ（http://www-vox.dj.kit.ac.jp/nishi/）にあるCVSのリンクが詳しい。ここからたどれるリンクには、日本語訳されたCVSのマニュアルも存在する。このマニュアルはたいへんよくできているので、印刷してじっくり読むことをお勧めする。

また、CVSに関しての日本語の書籍は少ないものの、

『CVS ―バージョン管理システム―』
Karl Fogel著 でびあんぐる監訳 竹内里佳訳
オーム社開発局　ISBN4-274-06372-0

という詳しい本が出版されている。この本は、使い方についての記述が優れているだけでなく、フリーソフトの文化的側面にもかなりのページが割かれていて、CVSを使ううえでたいへん参考になる。また、Debian Projectのメンバーが監訳していることもあって、記述も確かである。

Rubyで行こう

PerlやPythonに比べるとマイナー感の漂っていたRubyですが、いつの間にかかなり広く名が知られるようになってきたようです。今回は、ブレイクするまでの過程と現状をおさらいしてみましょう。

第13回　アウトブレイク

文：赤松智也
Text : Tomoya Akamatsu

初めの一歩

Rubyの誕生は1993年2月24日と言われています。この日、まつもとさんと石塚圭樹さん（つまりRuby本の作者ペアですね）の会話の中で、Rubyの開発と名前が決まったのだそうです。

少々意外なのですが、当初この言語は『改訂新版　オブジェクト指向プログラミング』（アスキー ISBN 4-7561-0276-X）に続く、石塚さんの次の本の例題プログラムと考えられていたようです。『言語を作りながら学ぶオブジェクト指向プログラミング』というようなねらいの本になるはずだったようです。なんだか、マニア向けな感じですね。結局、この「次の本」の企画は陽の目を見ることはありませんでした。やっぱりねえ（失礼）。

企画が形になることはありませんでしたが、きっかけを与えられた「言語おたく」のまつもとさんは、そのままRubyの開発を続けたということです。まつもとさんは、高校生のころから自分の言語が作りたいと思い続けていたそうですから、何かのきっかけさえあればよかったんでしょうね。

それから、FAQにあるRubyの名前の由来である「同僚の誕生石」の同僚とは、石塚さんのことなんだそうです。誕生石っていうので、なんとなく女性のような印象があるのですが、違ったのね。

ちなみに、石塚さんの実際の「次の本」は、『オブジェクト指向データベース』（アスキー ISBN 4-7561-1909-3）です。石塚さんの本のタイトルって、全部「オブジェクト指向」で始まるんですね。

コミュニティの形成

その後もRubyの開発は順調に進み、ある程度使いものになると感じたまつもとさんは、NetNewsで協力者を募集しました。これが1994年の12月です。開発が始まってから、2年近くが経っています。その後、すぐにruby-alpha-testという最初のメーリングリストが始まっています。このときのバージョンは0.60です。この時点で、ほぼ現在のRubyの原型ができています。1.0との大きな違いは、

- 定数のprefixは大文字でなく、%だった
- and、or、notがまだない
- beginではなく、protectだった
- 手続きオブジェクトがまだない
- デフォルト引数がまだない
- メソッド名に!や?が使えない
- nextでなく、continueだった
- 継承の指定が<でなく、:だった
- doによるブロック指定がまだない（{}のみ）

・例外が文字列ベースだった
・nilとfalseに区別がなかった

くらいでしょう。並べてみると結構ありますが、今のRubyの思想的な部分は、すでにかなり固まっていたのではないかと思われます。

1995年12月にはソースコードがNetNewsで一般公開されてます。そして、現在のメーリングリストであるruby-listが開設されています。この原稿の執筆時点で、ruby-listでは2万6000通近いメールがやり取りされています。

表1に、Rubyの初期の歴史年表を示します。

Perl Conference Japan

Rubyがブレイクした転機と言えば、おそらくは1998年11月のPerl Conference Japanではないでしょうか。オライリージャパンの主催による「第1回 Perl Conference Japan」では、なぜかまつもとさんによる基調講演が行われました。これはプログラム委員の歌代和正さん（IIJ）の発案によるものだそうです。歌代さんはこの後、『詳説 正規表現』の監訳者紹介で、

第1回 Perl Conference Japanのプログラム委員を依頼されるが、基調講演に国産スクリプト言語であるRubyの話題を推薦し、Perlファンからの顰蹙を買う

と書いておられます。

この「第1回 Perl Conference Japan」は、参加費が有料だったにもかかわらずかなり盛況だったようです。ここで、かなりの人が「ああRubyというものがあるのか」と感じたのではないでしょうか。

まつもとさんは、Perlの開発者Larry Wall氏に会えて感激していました。彼に憧れていたのだそうです。このときのことは『Ruby本』にも、

言語戦争
それぞれの言語信奉者の間でしばしば発生する激しい議論。しかし、周辺が激しく対立しがちなのに対して言語の

日付	内容	日付	内容
1993/02/24	作ろうと思った	1994/10/19	private method
1993/05/11	基本設計ができた	1994/12/08	alpha release — version 0.60
1993/07/05	宣言がなくなった	1994/12/08	ruby-alpha-test ML開設
1993/08/05	インタプリタが動いた	1994/12/15	手続きオブジェクト
1993/08/10	文字列・配列クラス	1994/12/19	and/orの導入
1993/10/15	GCができた	1995/02/09	protectなくなり、beginが導入
1993/11/05	ファイルIOができた	1995/02/20	定数の最初の文字を大文字に（それ以前は%を付けていた）
1994/01/25	ダイナミックリンク（a.outのみ）	1995/06/07	notの導入
1994/01/28	ruby-mode.el	1995/06/06	宣言されていない識別子はメソッド呼び出しに
1994/02/07	引数リスト末尾の＊	1995/07/04	デフォルト引数
1994/02/14	pack/unpack	1995/07/28	第1引数を賢く（?）判定するように
1994/02/17	trap	1995/08/09	dlopenによるダイナミックリンク
1994/03/07	自己代入（+=..）	1995/09/08	メソッド名の後ろに、!や?が付けられるようになった
1994/03/08	undef	1995/11/05	Tk I/F
1994/03/08	クラス定数	1995/12/16	retryでイテレータの再実行ができるようになった
1994/05/09	matzの転職	1995/12/21	fj.sourcesに公開 — version 0.95
1994/05/23	多重代入	1995/12/21	バグがあった。その日のうちに0.95aに
1994/05/30	autoconfの導入	1995/12/21	ruby-list ML開設
1994/06/13	Bignum	1996/01/09	nilとFALSEが分離
1994/06/24	-i optionによるin-place edit	1996/03/06	#{}で任意の式が展開できるようになった（これ以前は変数のみ）
1994/06/29	socket	1996/03/27	continueからnextに名前が変更
1994/07/11	private method	1996/05/22	継承の指定が、:から<になった
1994/07/14	配列式は[]、連想配列式は{}	1996/05/23	0.99-960523からバージョンに日付が入るようになった
1994/07/29	GCがスタックをスキャンするように	1996/06/25	break/next/redo/retryのメソッド化
1994/08/01	#{}による変数展開	1996/08/20	例外のオブジェクト化
1994/08/01	`cmd`	1996/08/29	「method do .. end」形式のイテレータ導入
1994/08/24	endの後ろのキーワードがなくなった	1996/09/30	djgppによるDOS版
1994/08/24	if/while修飾子	1996/11/29	%#..#形式の文字列導入
1994/08/26	メソッド呼び出しのかっこが省略できるようになった	1996/12/12	CYGWIN32版
1994/10/04	多重代入末尾の＊	1996/12/25	1.0リリース

表1　Ruby初期の歴史年表

設計者同士は意外に友好的である。事実、Perlの作者、Larry Wallはわたしのヒーローだし、サイン本も持っている。

と書かれています（付録C　Ruby用語集）。

Larry Wall氏のサインとともに、「To the Author of Ruby（Rubyの作者へ）」と書かれた『Programming Perl』は、まつもとさんの宝ものなのだそうです。「Perlファンに襲われそうになったら、これを見せたらいい」とLarryに言われたとか……。

Ruby本、集まれ

次の転機は、ちょうど1年後にやってきます。1999年11月、永らく出る出ると言われていた『オブジェクト指向スクリプト言語Ruby』（**コラム1**参照）がとうとう出版されたのです。最初に出ると聞かされてから1年以上待たされただけに、充実した内容でした。本連載の第1回目では、出版を記念してまつもとさんにインタビューを行っています。インタビューの内容は、

```
http://www.ruby-lang.org/ja/column/v0005.html
```

で公開されています。

本の出版はその後も続きます。1年近く間が空きましたが、2000年10月には、一気に『Rubyを256倍使うための本 ― 邪道編』（**コラム2**参照）と『Rubyプログラミング入門』（**コラム3**参照）の2冊が出版されました。

原稿執筆時点では、書泉グランデのコンピュータ関連書籍ランキングで『Rubyプログラミング入門』が1位、『Rubyを256倍使うための本 ― 邪道編』が3位にランクインしてます。コンピュータ関連書籍の話題独占という感じですね。初めてRubyに出会ったときの「これは使える」という印象は、間違いでなかったと思わせてくれます。

さて、国内だけでなく海外からも、『Programming Ruby: Pragmatic Programmer's guide』（**コラム4**参照）が出版されるというニュースが届いています。

Rubyの国際化もここまできました。海外では「Rubyって良さそうだけど、英語のドキュメントが少なくて」と言われていたようですが、この本はそんな声に応えるものになっています。

売れ行きも、マイナーな言語の解説書のわりには（失礼）好調のようで、発売直後のAmazon.comのランキングでは5173位です。そして、オープンしたばかりのAmazon.co.jpでは、瞬間最高ランキングが9位でした。しかも、これは洋書・和書合わせてのランキングなのですから大したものです。

私の知っている限り、日本製のフリーソフトウェアの解説書が、海外で出版されたことはありません。こんなにメジャーになるとは、数年前にはとても考えられなかったことです。

最後に、11月24日には、まつもとさんが執筆された『Ruby デスクトップリファレンス』（**コラム5**参照）が出版されます。すでに入手されている方もいらっしゃるでしょう。

Column 1

オブジェクト指向スクリプト言語Ruby
まつもとゆきひろ／石塚圭樹著
アスキー発行
本体価格4000円
ISBN4-7561-3254-5

言わずとしれたバイブルです。リファレンス的性質が弱いのと、間違いが多いのが玉に傷でしょう。しかし、2章の各機能の説明はそれなりにわかりやすいですし、4章から始まるオブジェクト指向プログラミングの説明は、Rubyの本としてだけでなく、純粋にオブジェクト指向プログラミングの本としても有用です。

Column 2

Rubyを256倍使うための本 ― 邪道編
arton著
アスキー発行
本体価格1200円
ISBN4-7561-3603-6

先月も紹介しましたが、WindowsでRubyを利用することに特化した珍しい本です。Rubyの紹介ではなく、「WindowsでのRubyの使い方」の紹介に的を絞っているところに好感が持てます。

一般誌への進出

朝日新聞社の出している週刊誌「AERA」の11月6日号をご覧になりましたか?

「IT天才を育てる」というタイトルで、通産省の「未踏ソフトウェア創造事業」について紹介する記事が掲載されました。

記事では、採択された56人のうちの5人が紹介されているのですが、そのうちの1人として、まつもとさんが登場しています。しかもカラー写真付きで。「オブジェクト指向スクリプト言語Ruby　次期バージョンの開発」というテーマが選ばれたのだそうです。メーリングリストでは、文章は大量に見かけるものの、なかなか表に出てこないまつもとさんを、一般誌で見かける日が来るとは思いませんでした。

本人は「天才なんてあんまりだ」と恥ずかしがっていましたが、これで知名度がますます上がりましたね。

勝とうなんて、考えていない。そもそも、ソフトは人間に奉仕するためのもの。出しゃばっちゃいけない。ぼくは、気持ちよくプログラムを書ければいいんですよ。
(AERA 2000年11月6日号 9ページより)

なんだかすごく「まつもとさんらしい」って思いませんか?

Column 3

Ruby プログラミング入門
原 信一郎著
まつもと ゆきひろ監修
オーム社
本体価格2800円
ISBN4-274-06385-2

「ハチドリ本」と呼ばれることになるのでしょうか。待望の入門書です。たしかに、全体の半分を占める1章と2章は、入門書的なテイストが感じられ、比較的わかりやすいと思います。しかし、3章以降は結構細かい点まで説明しているので、入門を卒業した人にも役立ちそうです。本書は、初めての1.6対応書籍になります。いや、もしかすると後述の『Programming Ruby』のほうが早かったかもしれません。

ところで、普通のAERAの読者はこの記事を読んでどう思ったんでしょうねえ。ちんぷんかんぷんだったのではないでしょうか。

Perl/Ruby Conference Japan

さて、11月29日から12月1日にかけて、京都国際会議場においてPerl/Ruby Conferenceが開かれます。というか、今月号が発売された時点では、もう終わっています。雑誌のタイムラグはいかんともしがたいですね。

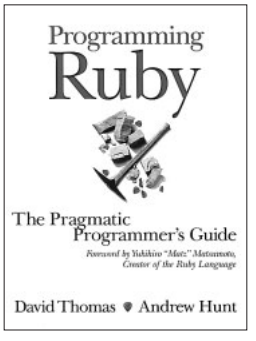
Perlの開発者Larry Wall氏も来るそうですし、なかなか楽しげなカンファレンスになりそうな予感がしています。

来月号では、ぜひこのPerl/Ruby Conference Japanをリポートしたいと思います。

Rubyの成功の秘密

世の中にフリーソフトウェアはたくさんあります。たとえば、http://www.freshmeat.net/を見れば、実に1万以上のプロジェクトが登録されています。ここに登録されていないものを含めれば、フリーソフトウェアがいったいい

Column 4



『Programming Ruby: Pragmatic Programmer's guide』
David Thomas & Andrew Hunt著
Addison-Wesley
ISBN0-201-71089-7

「Dave & Andy本」と呼ぶ人もいるようです。

I 言語の解説
II 応用編
III 言語リファレンス
IV ライブラリリファレンス
V 付録

という5部構成になっています。かなり良いリファレンスで、むしろ「Rubyは日本製なのに、なぜこの本が日本語でないんだ」と思わせるような内容です。なお、この本はピアソン・エデュケーションから春に発行される予定だそうです。

くつあるのか、想像もつきません。しかし、LinuxやPerlあるいは、われらのRubyのように広く使われるようになるものは、それほど多くはないようです。

その違いはどこにあるのでしょうか？ 個人的な考えですが、それは、以下の点にあるような気がします。

・ソフトウェアの質
・コミュニティの質
・開発の質

Rubyのソフトウェアとしての質については、本連載の第1回（「Rubyとの遭遇」）でも述べましたので、繰り返しては書きません。Rubyは、いろいろな意味で「良い言語」だと思います。

今回は、残りのコミュニティの質と開発の質について、少々考えてみましょう。

Rubyコミュニティ

Rubyコミュニティとは何かと考えると、やはりメーリングリストではないかと思います。Rubyについて語るチャンネルには、ほかにもネットニュースやIRCなどがありますが、開発者の登場頻度や有用な情報が流れる度合いから言って、メーリングリストは圧倒的に重要です。

Rubyのメーリングリストは5つあります。

・ruby-list@ruby-lang.org
メインのメーリングリストです。ほかのメーリングリストにあてはまらない話題すべてを受け入れます。

・ruby-dev@ruby-lang.org
開発者用メーリングリストです。Rubyのバグや仕様変更・追加の提案などについて話し合われます。Rubyの内部構造にかかわる難しい話題も流れます。

・ruby-ext@ruby-lang.org
拡張ライブラリ作成者用のメーリングリストです。拡張ライブラリのバグリポートが出ることもあります。

・ruby-math@ruby-lang.org
「Rubyと数学」についてのメーリングリストです。Rubyは数学的にも良い性質をもっている（らしい）ので、そのことについて話し合われたりするようです。

・ruby-talk@ruby-lang.org
英語のメーリングリストです。comp.lang.rubyニュースグループと相互乗り入れしています。

これらのメーリングリストに参加するためには、

```
ruby-xxx-ctl@ruby-lang.org
```

宛に、以下のように、本文に自分の名前を入れてメールします。xxxにはlist、dev、ext、math、talkが入ります。

```
subscribe Tomoya Akamatsu
```

しばらくすると確認メールが来ますので、それに返事をすると登録完了です。

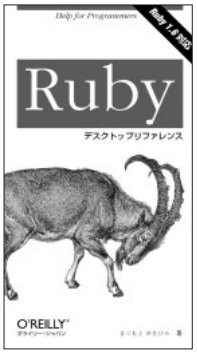
Rubyのメーリングリスト、特にruby-listでは、本当に初心者の説明からかなり高度な話題まで、さまざまなメールが流れています。メーリングリストのアーカイブは、長岡技科大の原先生の提供しているblade、

```
http://blade.nagaokaut.ac.jp/ruby/ruby-list/threads.html
```

から参照できます。

何かわからないことがあったら、まず検索してみるとよいでしょう。今回の原稿の執筆にもたいへん役立ちました。ありがとうございます。

Column 5



Ruby デスクトップリファレンス
まつもとゆきひろ著
オライリー・ジャパン
本体価格1000円
ISBN4-87311-023-8

オライリーのデスクトップシリーズの1冊です。1.6系をカバーしているということで、現時点では、（上記の「Dave & Andy本」を除いては）唯一の1.6系の唯一のちゃんとしたリファレンスだと思われます。値段も安いようですから、ぜひお手元に1冊どうぞ。

bladeの驚くべき点は、そのスレッド表示にあります。上記のURLから辿ってみると、各メールが1行ずつに要約されているのに、あたかも会話が成立しているように見える事実に驚愕するでしょう。

画面1にその一例を示します。

コミュニティの質

で、コミュニティの質という点で考えると、印象としてはRubyのコミュニティは「なんだか居心地がよい」というものがあります。多分、参加者の人柄（特に、まつもとさんを始めとする中心的なメンバーの）によるものだと思うのですが、今回は少し見方を変えて統計的に考えてみましょう。話を単純にするために、ruby-listだけを調べてみます。まず、メールの頻度について見てみました（**図1**）。

少々、意外な結果が出ました。1996年の秋ごろからメーリングリストのメール数は、平均するとほぼ一定なのです。折れ線グラフは、だいたい単調増加になっています。1996年というと、Rubyはあまり知られていないころだと思いますが、このころからすでに活発に議論されていたと考えるべきか、それとも最近話題になっているわりには、メール数には反映されていないと考えるべきなのか、私には判断できませんでした。

おそらくは前者で、それがこれまでのRubyを支えてきたのではないかと想像します。　次に参加者の数を見てみましょう（**図2**）。

グラフを見ると、参加者の方は単調増加以上のペースで伸びていることがわかります。あまり発言はしないが、メールを読んでいる人たちが支えていることも示しているようです。

新規参加者の山が1997年夏と1999年11月にあります。1999年11月のほうは、Ruby本の出版によるものだと思うのですが、1997年夏のほうはちょっとわかりません。なにかイベントがあったのでしょうか?

開発の質

Rubyの配布セットの中に、ChangeLogというファイルがあります。このファイルには、いつどのような修正を行ったかということが記録されています。このファイルから、いつ何件の修正があったかをグラフ化してみましょう（**図3**）。

これだけでは性質がわかりませんので、ほかのソフトウェアのChangeLogからのグラフと比較してみましょう。**図4-a**は全文検索のnamazuのChangeLog、**図4-b**はスケジュール管理ソフトのMHCのChangeLogです。この2

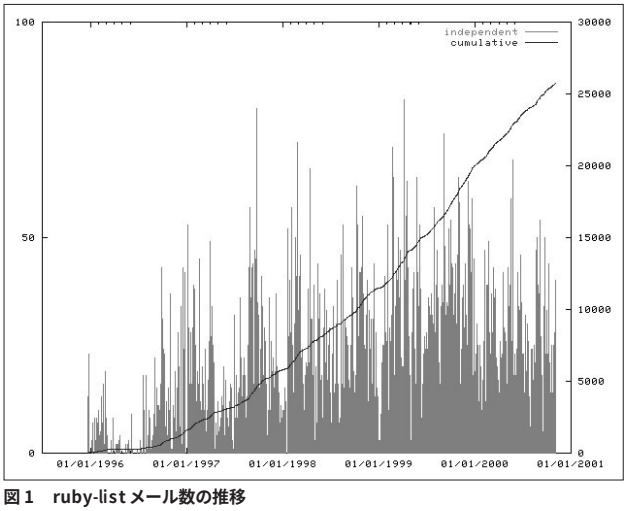


図1　ruby-listメール数の推移
棒グラフが日々のメール数、折れ線グラフがメール数の累計。

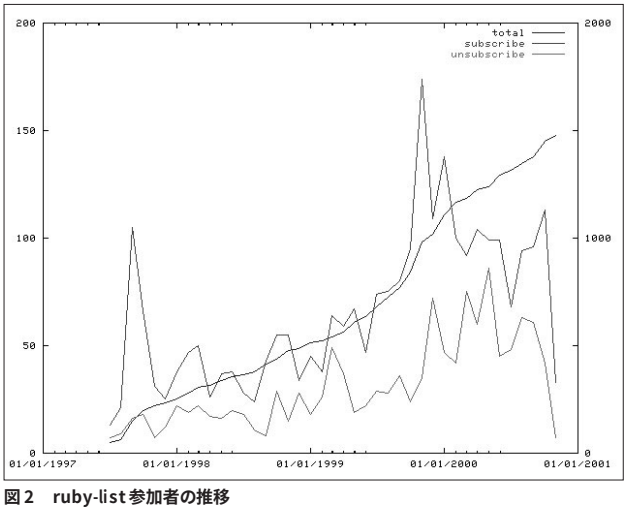


図2　ruby-list参加者の推移
色の濃い順に、参加者数累計、新規参加者数、脱退者数の月別グラフ。

```
* 半角カナ変換
25835 [tanaka@esd.s] cgi のプログラム中、半角カナを全角カナに変換したいのですが、
+ 25836 [sho@spc.gr.j]  NKFモジュールを使うといいですよ。
+ 25838 [kimura.koich] 漢字コードの指定をしていないからではないでしょうか? #!行に-Keを
+ 25839 [matz@zetabit] まずjcodeにバグがありました。パッチを添付します。
  25840 [tanaka@esd.s] ありがとうございます。
```

画面1　bladeのスレッド表示

つを選んだことに他意はありません。たまたま、ChangeLogがきちんと記録されていそうなソフトウェアだったということです。

分野が違うので直接の比較はできませんが、これらに比べるとRubyのグラフは更新がムラなく、こまめに行われているのがわかります。

このような継続的な改善（デバッグ?）が、Rubyの良さに貢献しているのではないかと思います。しかし、ここまで継続的に開発を続ける熱意はどこから来るのでしょうか? ずっと続けていて、飽きたりしないんでしょうか? よほどRubyが好きなんでしょうねえ。

グラフの舞台裏

今回のグラフは、もちろんすべてRubyを作って作りました。仕掛けは、以下のようになっています。

1. 元データからデータファイルをRubyで作る。
2. データファイルをgnuplotでグラフ化する。

データファイルは、以下のような形式です。

日付 数値

たとえば、以下のようになります。

```
1994-12-14 1
1994-12-15 6
  :        :
```

今回のデータファイルを作成するのに使ったスクリプトのうち、ChangeLogから更新頻度を生成するものを**リスト1**に示します。changes.rbはChangeLogを引数にして呼び出すと、カレントディレクトリにchanges.dat（日別更新数）とchages2.dat（累計更新数）の2つのファイルを作ります。

このデータファイルをグラフ化するには、gnuplotを使いました。**リスト2**に示すplotchange.pgというファイルを引数として、以下のようにgnuplotを呼び出すと、画面上にグラフが出力されます。

```
$ gnuplot plotchange.pg
```

現在コメントになっている、

```
#set term png color
```

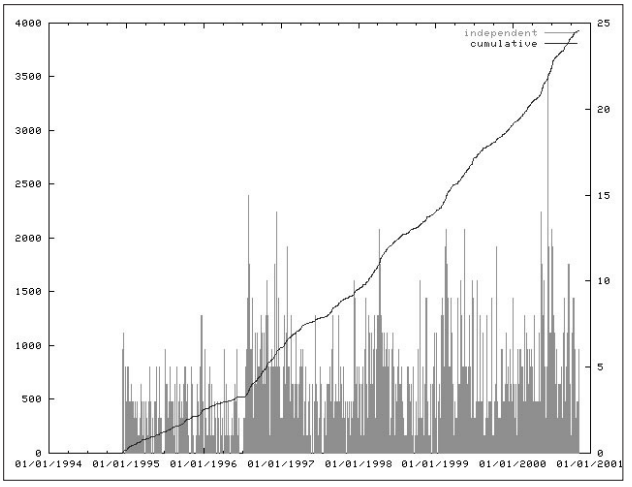


図3　RubyのChangeLog
棒グラフが日々の修正数、折れ線グラフが修正の累計。

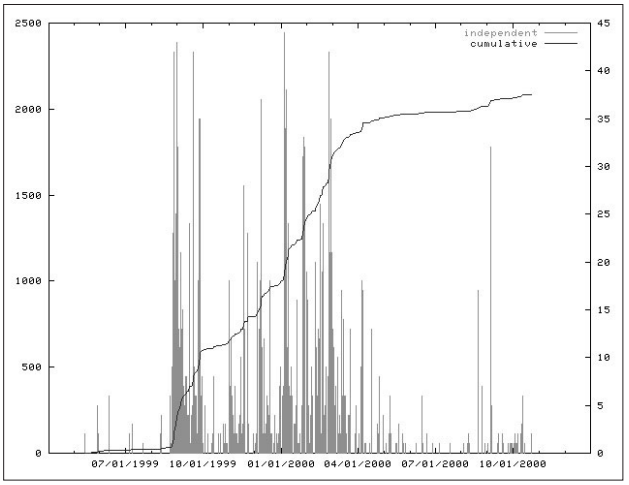


図4-a　namazuのChangeLog
棒グラフが日々の修正数、折れ線グラフが修正の累計。

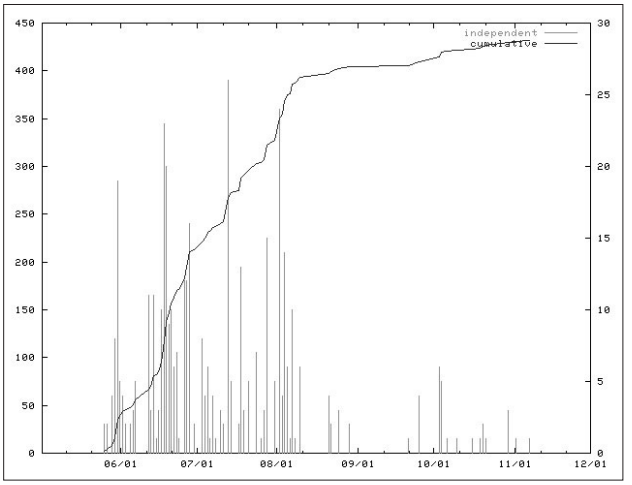


図4-b　MHCのChangeLog
棒グラフが日々の修正数、折れ線グラフが修正の累計。

の行のコメントをはずすと、標準出力にPNGファイルを出力します。

changes.rbとplotchange.pgを使えば、ソフトウェアの更新の傾向をひとめで見ることができます。いろいろなソフトウェアのChangeLogで試してみるとおもしろいでしょう。

本当はRuby/GDを使うなどして、グラフもRubyで書けば完璧だったのですが、時間の関係でgnuplotに頼ってしまいました。Ruby/GDについては、例によって読者への宿題にしましょう。Ruby/GDの情報は、以下のURLから得られます。

http://kirara.prec.kyoto-u.ac.jp/~tam/GD/

謝辞

今回、統計データを取るにあたり、メーリングリストのデータおよび古いChangeLogを、まつもとさんから分けていただきました。

また、ChangeLogを視覚化するというアイデアは、namazuプロジェクトからいただきました。この場をお借りしてお礼申し上げます。

▼リスト1　changes.rb

```
#! /usr/bin/env ruby

require 'parsedate'

data = Hash.new(0)
freq = 0
date = nil
while line = gets()
  if /^(?:Sun|Mon|Tue|Wed|Thu|Fri|Sat)\s+(\w+)\s+(\d+)\s+(?:[\d:]+)\s+(\d+)/x =~ line or
     /^(\d{4}-\d{2}-\d{2})\s/ =~ line then
    y, m, d = ParseDate.parsedate($&, true)
    if date
      data[date] += freq
      freq = 0
    end
    date = "%04d-%02d-%02d"%[y, m, d]
  end
  if /\s+\*/
    freq += 1
  end
end
f1 = open("changes.dat", "w")
f2 = open("changes2.dat", "w")
keys = data.keys
keys.sort!
sum = 0
for d in keys
  ent = data[d]
  f1.printf "%s %d\n", d, ent
  sum += ent
  f2.printf "%s %d\n", d, sum
end
```

▼リスト2　plotchange.pg

```
set xdata time
set timefmt "%Y-%m-%d"
#set x2mtics
#set y2tics 500
set ytics auto
set y2tics auto
#set term png color
plot "changes.dat" using 1:2  axes x1y2 title "independent" with impulses lw 2,\
     "changes2.dat" using 1:2 axes x1y1 title "cumulative" with lines lw 2
pause 5
```

[超]入門シェルスクリプト

Linuxの標準シェルであるbashのシェルスクリプトについて学ぶ本連載。今回は、シェルスクリプトで同じ処理を繰り返し（ループ）実行する際に利用するfor制御構造や、コマンドライン引数の利用方法などについて説明した後、複数のファイル名の拡張子をまとめて変更するシェルスクリプトを作成する。

第3回　for制御構造を使った繰り返し処理

文：大池浩一
Text:Koichi Oike

これまでの連載で紹介してきたシェルスクリプトは、スクリプトに書かれたコマンドを上から順に実行するだけだった。複雑なオプションなどの長いコマンドラインを入力しなくて済むという利点はあるものの、これだけではシェルスクリプトにする意味はほとんどない。たとえば、長いコマンドラインにわかりやすい別名をつけるだけなら、シェルの「エイリアス機能」を利用すればいい。

世の中で使われているシェルスクリプトのほとんどには、「指定したファイルが存在している場合に限ってこの処理を行う」といった条件判断や、「コマンドラインで指定したファイルに対して、1つずつ決められた処理を繰り返し行う」といった繰り返し（ループ）などが含まれている。処理の流れを変えるこうした「フロー制御構造」を利用することで、コマンドラインから直接実行するのは難しい複雑な処理を自動的に判断して行う「賢い」スクリプトを作ることができるのだ。

特に、同じような処理を繰り返し行いたい場合、繰り返しを実現する制御構造をマスターしているかどうかで大きく作業の効率が変わる。たとえば、100個あるJPEG画像のファイル名末尾の拡張子をすべて「.jpg」から「.jpeg」に変更することを考えてみよう。コマンドラインやGUIのファイルマネージャを使ってひとつひとつ手作業で変えていたのでは時間がかかって仕方ないが、今回紹介するfor制御構造を使ったシェルスクリプトにより、1回の実行でまとめて処理できるのだ。

繰り返し（ループ）を実現するfor制御構造

Linuxの標準シェルであるbashや、そのベースとなったBourneシェルでは、繰り返し（ループ）処理を行うfor、while、untilの3つの制御構造が用意されている。今回は、このうちのfor制御構造について取り上げる。

CやBASICなど他のプログラミング言語の経験がある場合、シェルスクリプトのforはそれらとはかなり毛色が異なるので気をつけてほしい。たいていのプログラミング言語では、forは「繰り返す回数」や「初期値・終了値・増加分」を指定して繰り返し処理を行う。

これに対し、シェルスクリプトのforは、繰り返し処理の対象となる値を「リスト」として並べ、それらを順番に処理するのだ。繰り返し回数はリストの値の個数によって自動的に決まる。なお、ある条件が成立している（あるは成立するまでの）間、繰り返し処理を行うには、whileやuntil制御構造を利用する。

以下では、

- **for制御構造の記述のしかた**
- **コマンドラインで指定したファイル名を扱う方法**

などの説明した後、実際にfor制御構造を利用したスクリプトの解説と作成を行う。

●for制御構造を記述する

for制御構造の書式は次のようになる。

```
for ループ変数名 in リスト
do
  繰り返す内容
done
```

以下の説明では、繰り返す内容の始まりを表わす「do」を、「for」と同じ行にまとめて書くことにする。それには、リストの後ろに文の区切りを表す「;」（セミコロン）をつけて、

```
for ループ変数名 in リスト; do
```

とすればいい。

それでは内容を説明していこう。まず、「リスト」には、空白文字（スペース・タブ・改行）で区切られた値（文字列や数値）を並べる。通常はスペースで区切って並べればいい。たとえば「foo bar baz」といった具合だ。ちなみに、「foo」や「bar」は、「特に意味のない文字列」を表わすときに慣習的に用いられる名前で、日本語なら「ほげ」や「ほげほげ」がこれに当たる。

このように、リストには任意の文字列や数値を指定できるのだが、実際にはファイル名が指定されることが多い。そのため、リストにはワイルドカード（＊、？など）を使用でき、シェルによりマッチするファイル名に変換された後でfor制御構造に渡される。たとえば、リストに「＊」と書くと、for制御構造にはカレントディレクトリの全ファイルのリスト（「.」で始まるファイルを除く）が渡される。

リストの内容は、先頭から1つずつ取り出されて、「ループ変数名」で指定したループ変数に格納される。ループ変数は、繰り返しのたびに値が変わる特殊なシェル変数で、変数名は自由に付けられる。たとえば、

```
for f in foo bar baz; do
```

とすると、ループ変数fには、最初の繰り返しでは「foo」、2回目の繰り返しでは「bar」、3回目の繰り返しでは「baz」が格納される。繰り返し処理が終了した後も、ループ変数にはリストの最後の値が格納されている。

実際に繰り返す内容は、「do」と「done」の間に記述する。複数のコマンドラインを記述可能だ。ここでは、「$変数名」とすることで、ループ変数の内容を参照（変数展開）できる。変数展開についての詳細は、先月号の本連載を参

図　for制御構造による繰り返し処理

照されたい。

たとえば、ループ変数fに先頭から順番に格納されるリスト「foo bar baz」の内容を、1行に1つずつ出力するには、doとdoneの間に、

```
echo $f
```

と書けばいい。最初の繰り返しでは「foo」、2回目の繰り返しでは「bar」、3回目の繰り返しでは「baz」がそれぞれ出力される（**図**）。

簡単な例題として、「カレントディレクトリにあるCとC++のソースファイル名を1行に1つずつ表示するスクリプト」を作ってみよう。一般に、Cのソースファイルは「.c」、C++のソースファイルは「.cpp」という拡張子を持つので、リストには「＊.c ＊.cpp」と書けばいい。そこで、スクリプトの内容は次のようになる。

```
#!/bin/sh
for f in *.c *.cpp; do
  echo $f
done
```

まず、「＊.c ＊.cpp」がシェルにより実際のファイル名に展開されたリストがfor制御構造に渡される。ループ変数fには、リストの内容が先頭から1つずつ順番に格納され、doとdoneの間の内容（ここでは「echo $f」）が繰り返し実行される。「$f」はループ変数fの内容に変数展開されるので、1行に1つずつCとC++のソースファイル名が表示されるわけだ。

ただし、「hoge␣␣␣hoge.c」のように、ファイル名に複数のスペースが含まれている場合、このスクリプトはうまく動かない。シェルは、空白文字（スペース・タブ・改行）を区切り文字として、与えられた引数を複数の「ワード」に分割した後でechoに渡す。echoは、与えられたワードを1個のスペースで区切って出力する。このため、ループ変数fの内容が「hoge␣␣␣hoge.c」だとすると、「hoge」と「hoge.c」という2つのワードに分割されてechoに渡され、echoはこれらを「hoge␣hoge.c」とスペース1個を挟んで出力することになる。

この問題を解決するには、先月号の本連載でも説明した「クォーティング」を利用する。復習を兼ねて、再度説明することにしよう。クォーティングとは、コマンドの引数（一部でも可）の周囲を「'」（引用符）か「"」（ニ重引用符）で囲むことだ。シェル変数の値を参照（変数展開）する場合には「"」を利用する。

具体的には、ループ変数fの変数展開部分を「"」で囲んで「"$f"」と書けばいい。修正後のスクリプトは以下のようになる。

```
#!/bin/sh
for f in *.c *.cpp; do
  echo "$f"
done
```

「"」の内部の空白文字は、シェルによるワード分割の対象外になる。このため、ループ変数fの内容が「hoge␣␣␣hoge.c」の場合、echoには「hoge␣␣␣hoge.c」という1つのワードが渡され、そのまま出力される。シェル変数の内容に空白文字が含まれる可能性がある場合は、必ず「"」でクォーティングするようにしよう。

なお、クォーティングに使われる記号には、もうひとつ「'」がある。こちらは、変数展開も行われない「強い」クォーティングだ。もし、上のスクリプトで誤って「echo '$f'」と書いてしまったとすると、ループ変数fの内容にかかわらず、echoにより「$f」という文字列そのものが繰り返し出力される。

この節での要点をまとめると、

- **複数の値のリストを順番に処理するにはfor制御構造を利用する**

Column

for制御構造で繰り返し回数を指定するには

シェルスクリプトで、「シェル変数の値を1から5まで1ずつ増やす」ような繰り返し処理は、「1 2 3 4 5」のようにそれぞれの数値をスペース区切りでリストに書かなければならない。回数が多いといちいち手で書くのは面倒になるので、数列を自動的に生成して出力するコマンドseqがGNUシェルユーティリティに用意されている。

使い方は簡単で「seq 5」のように終了値を指定すればいい。初期値や増加分を指定することも可能だ。前回説明したコマンド置換（コマンドを「`」で囲む）を利用して、seqの実行結果をfor制御構造のリストとして使う。たとえば、「1から100まで1ずつ増やして繰り返す」には、「for i in `seq 100`; do … done」と書けばいい。

- **リストにはスペースで区切って値を並べる（ワイルドカードも利用可能）**
- **繰り返すたびに、ループ変数にリストの値が1つずつ格納され、「$変数名」で値を参照できる**
- **空白文字を含む値を正確に出力するには、ループ変数の周囲を「"」で囲む（クォーティング）**

ということになる。

●コマンドライン引数で指定したファイル名を扱う

for制御構造がシェルスクリプトで使われる典型的なケースは、スクリプト実行時のコマンドラインで指定したファイル（複数可）をリストとして受け取り、1つずつ処理するというものだ。一度に1つのファイルしか受け付けないコマンドも、for制御構造を利用したスクリプトを被せる（ラッピングする）ことで、複数のファイルをまとめて処理できるようになる。こうしたスクリプトのことを「ラッパー」と呼ぶ。

スクリプト実行時にコマンドラインで指定した内容（これを「コマンドライン引数（ひきすう）」、あるいは単に「引数」と呼ぶ）を、for制御構造のリストとして利用するには、リストに「"$@"」と書けばいい。たとえば、コマンドライン引数を1行に1つずつ表示するスクリプトpargは次のように書ける。

```
#!/bin/sh
for f in "$@"; do
  echo "$f"
done
```

「"$@"」という表現は、本連載の第1回にも登場した。今回は、もう少し詳しく説明することにしよう。シェルスクリプト内で、実行時のコマンドライン引数を参照するには、「位置パラメータ」と呼ばれる特殊な組み込み変数を利用する（**表1**）。個別の引数に対応する位置パラメータのほか、すべての位置パラメータを並べて書いた内容が格納された「@」や「＊」も用意されている（2つの違いについてはコラムを参照）。

「@」に設定された内容を参照（変数展開）するには、通常のシェル変数と同様に「$」を付けて「$@」とすればいい。さらに、内容に空白文字が含まれている場合に備えて「"」でクォーティングすると「"$@"」という表現になるわけだ。なお、位置パラメータはすべて参照のみ可能で、「変数名＝値」として新しい値を設定することはできない。つまり、シェルスクリプト中では位置パラメータは常に「$」付きの形で書かれることになる。

「"$@"」を変数展開した結果は、コマンドライン引数の内容をそれぞれ「"」でクォーティングし、スペース1個で区切って並べたものになる。たとえば、上のスクリプトpargを、環境変数PATHに設定されたディレクトリ（~/binなど）にコピーした後、

```
$ parg foo bar baz
```

とすると、「"foo" "bar" "baz"」という3つの値のリストがfor制御構造に渡される。その結果、ループ変数fの値は、

パラメータ	動作
$0	スクリプトの名前に展開される
$1、$2、$3、…	対応する引数の内容に展開される
"$@"	全引数をそれぞれ「"」でクォーティングし、スペース1個で区切って並べたものに展開される
"$＊"	全引数を環境変数IFS（区切り文字を並べた文字列）の先頭文字（初期値スペース）で区切って並べ、全体を「"」でクォーティングしたものに展開される
$#	引数の個数に展開される

表1　位置パラメータの参照形

Column

"$@"と"$＊"の違い

コマンドライン引数すべてを表わす表現としては、本文中で取り上げた「$@」のほかに、もうひとつ「$＊」がある。両者の違いは、「"$@"」と「"$＊"」のように、クォーティングしなければわからない。

すでに説明したように、「"$@"」は全引数を「"」でクォーティングしたものをスペース1個で区切って並べた「"$1" "$2" "$3" …」と同じだ。これは、for制御構造のリストのように、それぞれの引数を別のワードとして再利用したい場合に都合がいい。

一方、「"$＊"」のほうは、全引数を区切り文字を表す環境変数IFSの先頭文字（初期値はスペース）で区切って並べ、それら全体を「"」でクォーティングした「"$1 $2 $3 …"」を意味する。全体が1つのワードとして扱われるため、for制御構造のリストで「"$＊"」を使うと、最初の繰り返しでループ変数に全引数が並んだ文字列が格納され、一度しか繰り返し処理が実行されない。

「"$＊"」が便利なのは、環境変数IFSを変更することで区切り文字を簡単に変えられる点だ。たとえば、全引数をカンマ区切りで出力するには、「IFS=','; echo "$＊"」とすればいい。trやsedなどのコマンドを使うことなく区切り文字を変更できるわけだ。

最初の繰り返しでは「foo」、2回目は「bar」、3回目は「baz」が設定され、「echo "$f"」によりそれらが1行ずつ表示されることになる。

なお、for制御構造の「in リスト」の部分を省略すると、シェルが「in "$@"」を自動的に補って実行する。今回の説明では、わかりやすくするために省略は避けたが、「in リスト」が省略されたスクリプトを読む機会があったら、コマンドライン引数を繰り返し処理しているのだと考えればいい。

echoを繰り返し実行しているだけではつまらないので、実際に役に立つシェルスクリプトの例として、指定した複数の「tarボール」（tarでアーカイブした後、さらにgzipで圧縮したファイルをこう呼ぶ）をまとめて展開するスクリプトを作成してみよう。

tarボールを展開するには、tarのオプションとして次の3つを並べて指定する。

z…圧縮ファイルを扱うことを指示する
x…アーカイブの展開を指示する
f…直後にファイル名を指定する

fオプションの直後には、展開対象となるファイル名を1つだけ書く。たとえば、

```
$ tar zxf foo.tar.gz
```

とすると、foo.tar.gzに含まれるファイルやディレクトリがカレントディレクトリに展開される。

tarは、一度に1つのアーカイブファイルしか扱えない。複数のファイル名を並べて書くと、fオプションの直後の1つだけがアーカイブファイル名として扱われ、残りはアーカイブに含まれるファイル名を指定したものと見なされる。たとえば、

```
$ tar zxf foo.tar.gz bar.tar.gz
```

と書くと、「foo.tar.gzに含まれるファイルのうち、bar.tar.gzだけを展開せよ」という意味になる。foo.tar.gzにbar.tar.gzが含まれていなければ、「ファイルが見つからない」旨のエラーメッセージが表示される。

そこで、指定した複数のtarボールを順番に展開するスクリプトuntarをfor制御構造を利用して作成しよう。スクリプトpargでは「echo "$f"」としていた繰り返し処理部分を、「tar zxf "$f"」に書き換えればいい。

```
#!/bin/sh
for f in "$@"; do
   tar zxf "$f"
done
```

このスクリプトuntarを、環境変数PATHに設定されたディレクトリ（~/binなど）にコピーすれば、

```
$ untar *.tar.gz
```

とするだけで、カレントディレクトリの「＊.tar.gz」にマッチするファイルをまとめて展開できる。一度に1つのアーカイブファイルしか展開できないという制限を、for制御構造を使ったスクリプトで解消したわけだ。

ところで、もし、ファイル名を1つも指定しないで、

```
$ untar
```

とすると、このスクリプトはどのように動作するだろうか。コマンドライン引数を1つも指定しなかった場合、「"$@"」は何にも展開されない。特に、for制御構造のリストに「"$@"」のみ書かれている場合は、一度も繰り返し処理が行われずに次の部分に移る。

untarをさらに実用的なスクリプトにするには、「ファイルが1つも指定されていない場合は使い方を表示する」とか、「指定されたファイルが実際に存在するか調べてからtarを実行する」といった処理を加えたい。こうした処理には、条件分岐を行うif制御構造を使う必要があるので、次回改めて取り上げることにしよう。

この節での要点をまとめると、

- **一度に1つのファイルしか受け付けないコマンドを、シェルスクリプトのfor制御構造と組み合わせることで、複数のファイルを処理できるようにする**
- **コマンドライン引数をfor制御構造のリストとして利用するには、リストに「"$@"」と書けばいい**
- **「"$@"」は、「"」でクォーティングされた引数をスペース1個で区切って並べたものに展開される**

ということになる。

今月のスクリプト

後半は、既存のスクリプトの内容を解説する「スクリプトを読む」と、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「スクリプトを書く」で構成する。今月の「スクリプトを読む」は、

- **デバイス名（「cdrom」や「fd0」など）を指定するだけでマウント動作を行う「gitmount」**

の内容のうち、for制御構造による繰り返し処理部分を抜粋して説明する。

一方、「スクリプトを書く」では、

- **指定した複数のファイルの拡張子をまとめて変更する「renext」**

を作成する。

●スクリプトを読む

gitmountは、GIT（GNU Interactive Tools）と呼ばれる対話的なファイル管理ツールに付属するシェルスクリプトで、ファイルシステムのマウントを簡単に行うために作られたものだ。

通常、mountを使ってファイルシステムのマウントを行う際には、-tオプションでファイルシステムの種類を指定し、その後の引数でデバイスファイルとマウントポイント（ディレクトリ）を指定する必要がある。

たとえば、CD-ROMをマウントする場合、スーパーユーザー（root）になった状態で、

```
# mount -t iso9660 /dev/cdrom /mnt/cdrom
```

としてマウントする。ファイルシステムは「iso9660」、デバイスファイルは「/dev/cdrom」（実際には/dev/hdcなどへのシンボリックリンク）、マウントポイントは「/mnt/cdrom」だ。これで、/mnt/cdrom経由でCD-ROMのファイルにアクセスできるようになる。

gitmountを使うと、まったく同じことを、

```
# gitmount cdrom
```

とするだけで実現できる。

gitmountは全部で90行あるスクリプトなので、ここではfor制御構造を使っている部分のみを抜き出してみた（**リスト1**）。この連載では説明していないif制御構造も含まれているが、複雑な処理はしていないので理解するのはそれほど難しくないはずだ。

1～2行目では、ループ変数fstypeに、ファイルシステムの種類（ext2、iso9660、vfat、…）を順番に格納して、繰り返し処理を行っている。1行目の行末の「＼」（バックスラッシュ）に注目しよう。

このように、スクリプトの行末に「＼」を書くと、次の行が現在の行と継続しているものとして扱われる。ここでは、ファイルシステムの種類が1行に書ききれないので、行末の「＼」を利用して次の行に続きを書いている。

シェルスクリプトの改行は、「;」と同様に文を区切るために使われるが、直前に「＼」を書くことでその意味を無視させるわけだ。この方法を「バックスラッシュエスケープ」という。改行だけでなく、ワイルドカードの「＊」「？」や、クォーティングに使う「"」「'」もバックスラッシュエスケープすることが可能だ。

続いて、実際に繰り返される部分について見ていこう。3行目では、ループ変数fstypeに格納されたファイルシステムでマウントを試みている。fstypeのほかに、deviceとmpというシェル変数が使われている。リスト1より前の部分で、deviceには最初のコマンドライン引数「$1」の内容、mpには「/mnt」がそれぞれ設定済みだ。

たとえば、

▼リスト1　スクリプトgitmount（一部抜粋）

```
1: for fstype in ext2 iso9660 vfat msdos ntfs minix ext xiafs\
2:               hpfs xenix sysv coherent ufs umsdos affs; do
3:   mount -t "$fstype" /dev/"$device" "$mp/$device"> /dev/null 2>&1
4:   if test $? = 0; then
5:     success=1
6:     break
7:   fi
8: done
```

```
# gitmount cdrom
```

として起動すると、それぞれの繰り返しでの3行目の実際の内容は次のようになる。

```
mount -t ext2 /dev/cdrom /mnt/cdrom
mount -t iso9660 /dev/cdrom /mnt/cdrom
mount -t vfat /dev/cdrom /mnt/cdrom
mount -t msdos /dev/cdrom /mnt/cdrom
  :
```

つまり、gitmountは、正しいファイルシステムを総当り方式で見付けるという、あまり洗練されていないスクリプトなのだ。間違ったファイルシステムでマウントしようとするとエラーメッセージが表示されるが、これは3行目末尾の「> /dev/null 2>&1」により、画面に表示させないようにしている。

4行目では、if制構造を使って、マウントが成功したかどうかを条件式「test $? = 0」で判断している（if制御構造や条件式については次回詳しく説明する）。マウントが成功した場合は、5行目でシェル変数successに1が設定され、6行目のbreakでfor制御構造から抜ける。つまり、一度マウントが成功したら、残りのファイルシステムは無視して繰り返し処理を終えるわけだ。

なお、リスト1より後ろの部分では、シェル変数successの値からマウントに成功したかどうかを判断して、それぞれの場合の後処理を行っている。

●スクリプトを書く

UNIX系OSでのファイル名の変更は、ファイルの移動を行うコマンドmvを利用する。mvの基本的な使い方は、

```
$ mv *.jpg ~/image
```

のように、「ファイル名（複数可）」と「移動先ディレクトリ」を指定して、ファイルを指定したディレクトリに移動させるというもの。ただし、

・ファイル名が2つだけ指定されている
・どちらもディレクトリではない

という条件を満たした場合は、ファイル名の変更を行う。たとえば、「foo.jpg」というファイル名の拡張子を「.jpeg」に変更するには、

```
$ mv foo.jpg foo.jpeg
```

とすればいい。

このような仕様のため、mvを使ったファイル名の変更は、一度に1つずつしか行えない。3つ以上のファイル名を並べると、ファイルの移動と見なされてしまうからだ（しかも、最後の引数がディレクトリでないため結局はエラーになる）。

しかし、多数のファイルの名前（特に拡張子の部分）をまとめて変更したいという場合は結構ある。そこで、for制御構造を使って、複数のファイルの拡張子をまとめて変更するスクリプトrenextを作成しよう。

処理を簡略化するため、「renext 拡張子 ファイル名…」のように、最初の引数で「変更後の拡張子（.は含まない）」を指定し、後の引数で指定したファイルの拡張子をすべて置換するという仕様にする。

たとえば、

```
$ renext jpeg *.jpg
```

とすると、「＊.jpg」にマッチするファイルすべての拡張子が「.jpeg」に置換される。

(1)プロトタイプの作成

初めに、指定した拡張子をファイル名の末尾に追加するだけのプロトタイプを作成しよう。最初のコマンドライン引数は「変更後の拡張子」であって、繰り返し処理の対象であるファイル名ではないということに注意する必要がある。これまでのように、そのままfor制御構造のリストで「"$@"」を使うと、ループ変数にファイル名以外の内容が設定されて都合が悪い。

こうした場合の定石は、「最初の引数の内容をシェル変数に格納した後で、引数の内容を1つずつ前にずらす」こ

▼リスト2　プロトタイプのスクリプトrenext-proto

```
1: #!/bin/sh
2: ext="$1"
3: shift
4: for f in "$@"; do
5:   mv "$f" "$f.$ext"
6: done
```

とだ。引数の内容を1つ前にずらすには、shiftを使う。実際のスクリプトはリスト2のようになる。2行目で、最初の引数の内容に展開される「$1」を、シェル変数extに格納した後、3行目のshiftにより、全引数を1つずつ前にずらしている。つまり、いままで2番目だった引数が「$1」で参照できるようになる（以下同様）。これで、「"$@"」にはファイル名しか含まれなくなるので、4行目のfor制御構造のリストで「"$@"」を指定できる。

ループ変数fには、処理対象となるファイル名が1つずつ格納される。繰り返す処理は5行目のmvのみで、変更後のファイル名として、元のファイル名「$f」の後ろに、「.」と「$ext」を追加している。これをすべてのファイルについて行うとスクリプトは終了する。

(2)ファイル名から拡張子を取り除く処理を考える

プロトタイプに不足しているのは、元のファイル名「$f」から、末尾の拡張子を取り除く処理だ。このような文字列の置換や削除といった処理には、ストリームエディタsedがよく使われる。

エディタといっても、sedはEmacsやviのような対話的なスクリーンエディタではない。処理の内容をスクリプトとして記述し、ファイルや標準入力の内容をスクリプトに従って変更して出力する「フィルタ」の一種だ。ファイルに書かれたスクリプトを読み込むことも可能だが、今回は処理が短いのでコマンドラインに直接記述する。

sedのスクリプトは、少々暗号じみている。たとえば、今回利用する文字列の置換は、「s/パターン/変換後文字列/」という形式で書く。パターンには、「正規表現」と呼ばれる、ワイルドカードよりさらに強力で柔軟な表現を利用できる。たとえば、ファイル名末尾の拡張子を表すパターンは「＼.[^.]＊$」だ。

正規表現について説明すると、それだけで1冊の本ができてしまうくらいなので、ここでは軽く触れるだけにする。「＼.」は「.」そのもの1文字、「[^.]＊」は「.」以外の任意の文字が0文字以上続く文字列、「$」は行末を表している。つまり、「.」で始まって、英数字などが行末まで続く文字列をこれで表現できてしまうわけだ。

今回はマッチした部分を取り除きたいので、変換後文字列にはなにも指定しない。よって、ファイル名の拡張子を取り除くスクリプトは「s/＼.[^.]＊$//」となる。あとは、ループ変数fに格納された元のファイル名をechoで表示し、それをパイプでsedに渡せばいい。具体的には次のようなコマンドラインになる。

```
$ echo "$f" | sed 's/\.[^.]*$//'
```

なお、sedのスクリプトには、シェルがワイルドカードと見なす「＊」などが含まれるので、「'」（引用符）でクォーティングする必要がある。

(3)スクリプトを完成させる

拡張子を取り除く処理をスクリプトに組み込むには、プロトタイプでは「"$f.$ext"」としていた2つめのファイル名の「$f」を、sedによる変換結果で置き換えればいい。具体的には、「`」（逆引用符）で囲むコマンド置換を利用して、

```
"`echo "$f" | sed 's/\.[^.]*$//'`.$ext"
```

とする。「"」によるクォーティングでは、このようにコマンド置換も利用可能だ。実際のスクリプトはリスト3のようになる。拡張子を取り除く処理が組み込まれた5行目以外はプロトタイプとまったく同じだ。

なお、bashのバージョン2.xを使っている場合は、シェル変数の末尾から、指定したパターン（ワイルドカード可）にマッチする最短部分を削除する演算子「${変数名%パターン}」が用意されている。これを利用すると、ループ変数fの内容から拡張子を取り除く処理は「${f%.*}」と書けるので、sedを利用する必要がなくなる。スクリプトは**リスト4**のようになる。

▼リスト3　完成版のスクリプトrenext

```
1: #!/bin/sh
2: ext="$1"
3: shift
4: for f in "$@"; do
5:   mv "$f" "`echo "$f" | sed 's/\.[^.]*$//'`.$ext"
6: done
```

▼リスト4　bash 2.x専用のスクリプトrenext2

```
1: #!/bin/bash2
2: ext="$1"
3: shift
4: for f in "$@"; do
5:   mv "$f" "${f%.*}.$ext"
6: done
```

一年の計に、TEXと人生について考えてみた

Emacsはじめました

第10回　理系大学（院）生のためのEmacs

TeX愛用のみなさまには、ひとつ朗報を。Emacsを使うとコマンドを入力する手間が省け、カッコの数が合わない苦労もなくなっていきます。おまけに、Emacsの窓のなかで原稿作成からタイプセット、印刷までワンタッチ。この快適な環境をぜひとも分かちあいたく、特に編集長に押してペンをとったしだい。

文：佐々木太良
Text : Taroh Sasaki

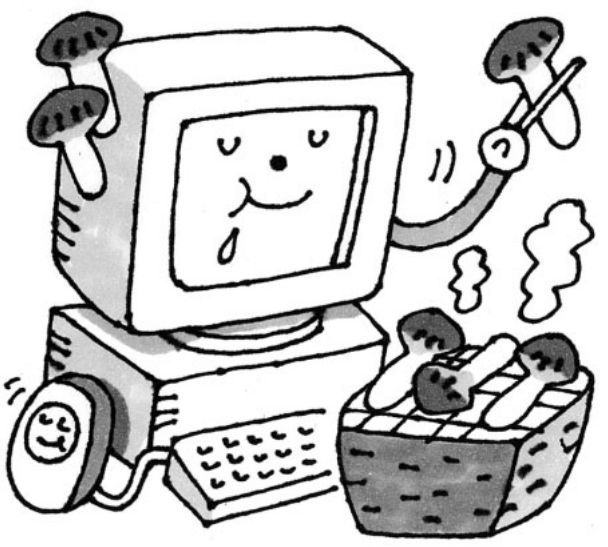


Illustration : Manami Kato

TeXで行こう

じゃーん！　今回のテーマは、TeXとプログラミング言語モードです、というと引いてしまう方がいるかもしれません。「えっ？ 俺プログラム書かないし……」「TeXってあの、出版に使う組版システムでしょ？」

しかーし。本誌読者の12.8%を占めるという（ほんとか）理系大学生・大学院生の諸兄が、研究室に配属されてから「うちはTeXだからね。勉強してきてね」といわれ「てふ？　それっておいしいんですか？」とボケたためにいきなり後ろからバール様のもので殴られる事件が後を絶たないと……（編集：先生、そろそろこのへんで）。

さてはて。C言語のプログラミング（こちらは小規模ながら、MS-DOS時代からパソコンでもできました）やTeXによる論文の執筆などが、大学の何百万もする高価なワークステーションではなく、9万9800円（税別・10%ポイント還元）で買えるPCとLinuxで可能になったのですから、今の若者は幸せじゃと言わずしてなんと言いましょうか。しかもこの2つのテーマは、Emacsから抜けずに実行すると実に幸せなお仕事なのです。

もともとUNIXは「研究的な開発環境」としての素地があったため、つまりあまり人様の役に立たないモノ（ま、おかげで今日のIT社会があるわけです←言い訳）を黙々とプログラミングしてみて、結果を文書にする、という作業が繰り返されていました。ちょっとしたプログラムを書いてみる、そのドキュメントを書いてみる、などという仕事をしてみると、ふだん使っているLinuxマシンががぜん光輝いて見えるかもしれません。プログラミングなんか関係ねーやという方も、ここはひとつ「モードって何？」という好奇の目でもって読んでみてください。

TeXってなに？

TeXは、Donald E. Knuthという数学者が作成した組版システムです。ほんとうは「TEX」と表記しますが、それが難しいときには「TeX」と書きます。ちなみに、読み方は「てふ」「てっく」などの宗教があるようですが、Knuth先生の解説書の中に「ドイツ語の『っは』の音と近い」という説明があるので、私は「てふ」派です。

このシステムは、アカデミック以外の場ではあまり知られていないかもしれませんが、理工系（特に数学やコンピュータサイエンス）の教科書や専門書のかなりがTeXで「書かれて」います。また、論文や学会の予稿集の多くは発表者が用意した出力品質でそのまま印刷されるので、標準的なテンプレートが揃い、組版の専門知識がなくてもそれなりに整った出力結果が得られるTeXを利用する人はかなり多いといえます。

TeXで「書かれる」といいましたが、この組版システムはずばりいうと植字工、つまり原稿を見て活字を拾ってく

れる職人さんのコンピュータ版です。TeXが開発された経緯は、コラム「Knuth先生のハッカー魂」をどうぞ。

TeXは、

```
\section{原稿の作成}
    まず最初に、
    例によってリスト1のようなYaTeXのスタートアップ
    を~/.emacsなどに書いておきましょう。
```

などと書いておくと、

1.3　原稿の作成

まず最初に、例によってリスト1のようなYaTeXのスタートアップを~/.emacsなどに書いておきましょう。

のように出力される、マークアップ言語型の組版ソフトです（**図1**）。これと対比されるのは、WYSIWYG（what you see is what you get＝見たものが得られる）型のワープロソフトなどです。

マークアップ型のものとしてはほかにSGMLなどが有名ですが、わりと特殊な用途が多いため、このまま絶滅するのかなぁ、などと思っていたのは数年前のこと。今では、意識していない方もあるでしょうが、かなりの人がWebページを組むためにマークアップ言語型のHTMLのお世話になっています。HTMLの場合、組版をしているのはWebブラウザということになります。

図2はTeXの原稿を書いてから出力を得るまでの手順です。まず、原稿となるソース（拡張子は.tex）を書かないといけません。どんなエディタでもよいのですが……もちろんこの連載ではEmacsですよね。

これをplatexというTeXの処理系を通すことで、タイプセット（組版）を行います。タイプセットの過程はちょうどコンパイルのようなもので、ページ上の文字の配置位置を決めて、その結果をいったんDVI（device independent）形式のファイル（拡張子は.dvi）に出力します。このファイルは、出力先がプリンタなのかブラウザなのか、また出力機器のコマンド体系や解像度には依存しません。

DVIファイルはその後、出力機器に合わせてフォントの字形の情報を与えられ、ビットマップなどに変換されて、出力されるわけです。プリンタによってビットマップ画像の表現の仕方は違いますが、それはドライバが解決します。

UNIX系の環境では、プリンタ駆動言語としてAdobeのPostScript（拡張子は.ps）というページ記述言語が好まれています。PostScriptはPDFの前身ともなったファ

Column

TeXの分家たち

本文中ではTeXとそのさまざまなマクロをひっくるめて「TeX」と呼んでいますが、正確にはTeXというのはタイプセッタープログラムのエンジン部分です。このエンジンと、Knuth先生（とその一味）が書いた組版（整形）の指定に使用する基本的なマクロセットを組み合わせたのが、plain TeXです。

現在、理工系の大学・研究所などで多く使われているのはLaTeX（ほんとうはLaTeXと表記します）というシステムですが、これは人工知能の権威であるLeslie Lamport先生の手になるマクロ一式で、さらに定型的な書籍や書類が作成しやすくなります。LaTeXは、TeXのように組版の細かい指定をたくさん覚えなくても、一定の様式で整形できるのが魅力です。

このほか、アメリカ数学会が「論文を投稿するならTeXのソースでね」という規定を作ったので、AMS TeXという数学の論文のためのマクロとフォントのパッケージがあります。また、日本の藤田新作氏（かなりのTeXHackerです）が化学用に書いたChemiTeXなどの分派があります。

TeXのエンジンやLaTeXは、その後2ε（2eとも表記します）というバージョンに進化し、強力かつ安定した実装になっています。これらの日本語対応はどうなっているかというと、以前はTeX/LaTeXを日本語化したアスキー版とNTT版のjTeX/jLaTeXが長らく使われていましたが、現在ではアスキーがエンジン部分から見直したpTeX2e/pLaTeX2eが定番環境になっています。これらは縦書き文書にも対応していて、さすがにビジュアル重視の仕事には使われないにしても、まだまだ「技術文書」を書く人のお友だちとして活躍していくものと思われます。

Linuxについていえば、RPMなどのパッケージがあるのはpTeX2e/pLaTeX2eでしょう。

日本語で使う場合は、原稿の文字コードにどれを選ぶかも重要です。ほかのソフトとの兼ね合いで、EUCかSJISコードを使う人が多いでしょう。Windowsなどと文書のやり取りがある場合はSJISが便利だよ、という声も聞かれますが、UNIX上のPerlなどのスクリプト言語で文書テキストを加工したい人にはEUCがお勧めです。また、TeXが対応しているわけではありませんが多国語を同時に取り扱ったり、あるいはネットワークでやり取りする文書をそのまま再利用することが多ければ、JISという選択肢もありでしょう。

ただし日本語Emacsを通してタイプセットをするかぎり、あまり意識する必要はないはずです。

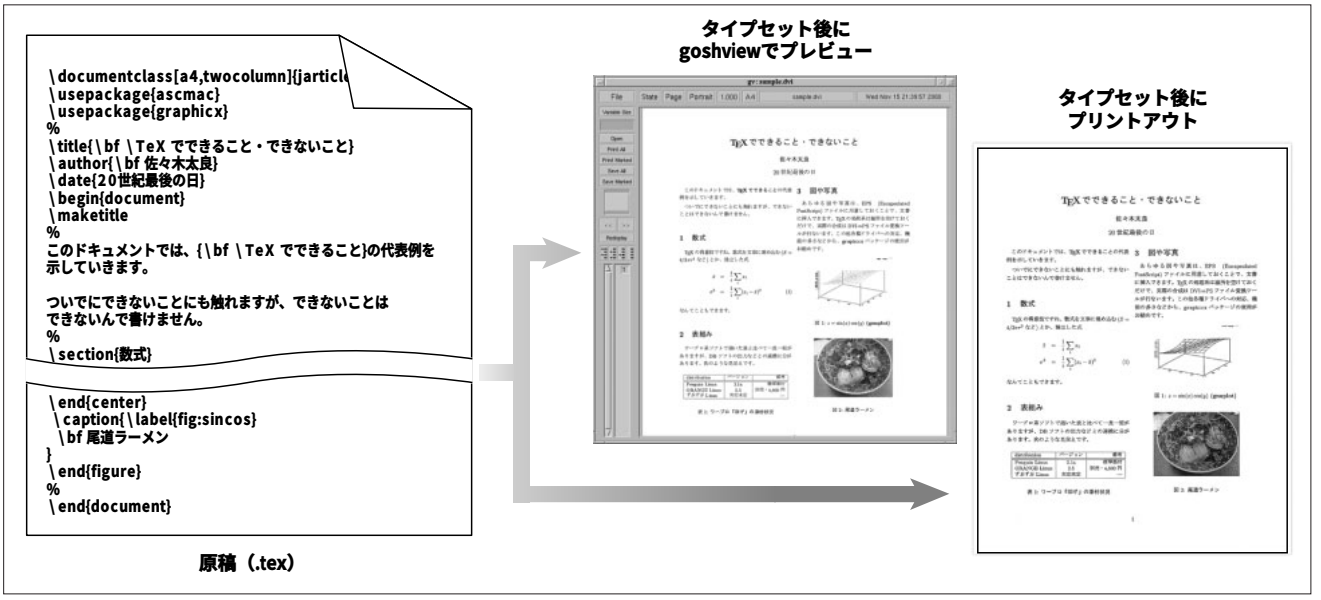


図1 TeXの原稿と出力

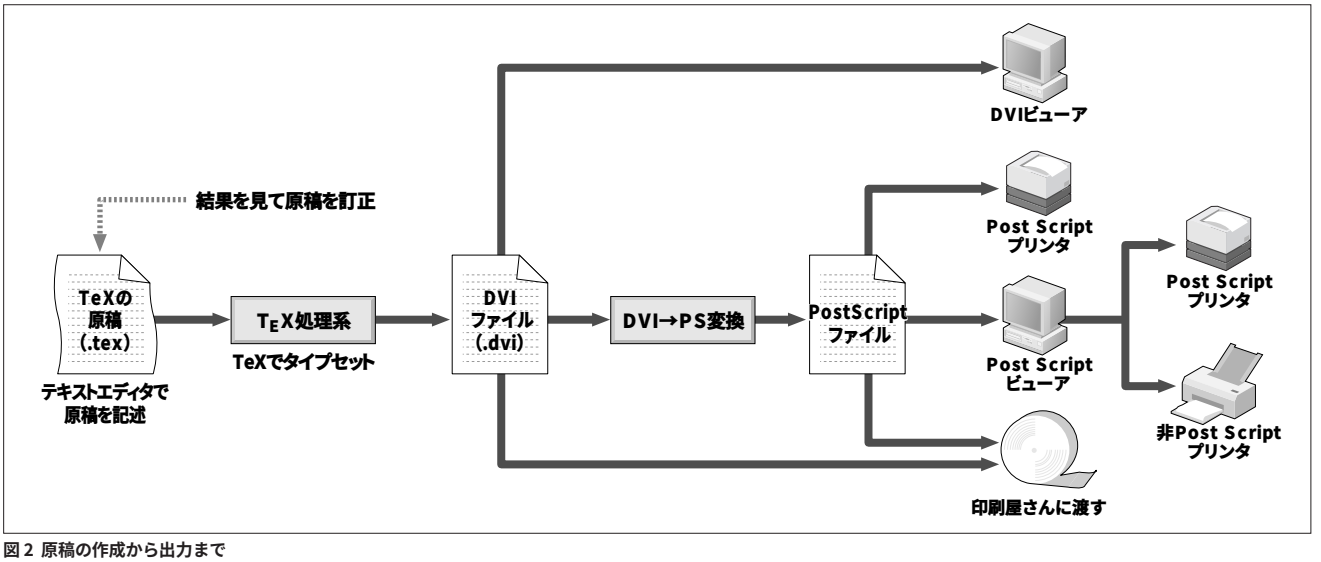
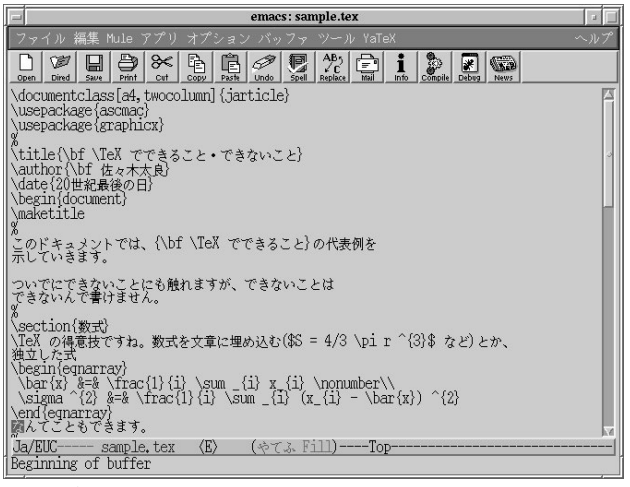


図2 原稿の作成から出力まで

イル形式ですが、単なる白黒の点々が含まれたドットマトリックスのデータだけではなく、「線を引け」「丸を描け」などという命令が書いてあって、PostScript対応のプリンタなどはそれを「よしきた」と言って（言いませんが）実行します。DVIからPostScriptへの変換には、おもにdvipsとdvi2psの2つのツールを使うのが主流ですが、これらを日本語化した日本語dvipskと日本語dvi2psも用意しましょう。

PostScriptを画面上でプレビューするには、Linux環境ではGNUのPostScript互換エンジンであるghostscriptと、その画面表示用ブラウザであるghostview（gv）が利用できるはずです。またghostscriptは、各種のプリンタの出力形式に合わせて変換する機能も持っています。というと「DVIから直接各プリンタのフォーマットに変換するツールはないの？」といわれそうですが、プリンタの種類だけDVI変換ツールを用意するより、いったん共通フォーマットのPostScript形式に変換して手持ちのプリンタに出力するほうが、用意するツールが少なくてすみます。ghostscriptはTeX以外にも、安価なプリンタでグラフィックを出力する場合、事実上唯一のプリンタドライバになりうるので、この際用意しておいて損はありません。なお、Xウィンドウシステムには、直接DVIファイルを表示できるプレビューアとして、xdviというツールなどが用意されています。

図や写真を挿入する場合は、PostScript形式の図を用意するのが便利です。これは、Tgifなどのドローイングツー



画面1　やてふモード

ルであれば、PostScript（EPS）ファイルへの出力ができるでしょうし、GIMPなどのフォトレタッチソフトもたいていは出力形式にPostScript形式を選べます。この時点で、寸法指定を気にする必要はありません。図の取り込みを原稿上で指定すると、TeXはタイプセット時にその場所を空けておいてくれるだけですが、dvipskやdvi2psなどでPostScript形式に変換するときに、その図を合成してくれます。このためには、PostScriptの図の大きさを自動判別して自動的にスペースを空けるマクロを利用します。

やめたくなっちゃいましたか？　しかし理工系の大学生で「卒論はTeXだからね」と先輩（先生）に言われれば、これはもう最後まで読まないといけません（笑）。

でも安心してください。今回特別に紹介するYaTeX（やてふ……って野鳥ですか？）を利用すると、マークアップに使うTeXのコマンド入力が大幅に楽になるうえ、Emacsの中で原稿書きからプレビューや印刷までほぼワンタッチでできてしまいます。

原稿の作成

まず最初に、例によって**リスト1**のようなYaTeXのスタートアップを~/.emacsなどに書いておきましょう。拡張子が.texのファイルをfind-fileしたときに、自動的に**画面1**のような、やてふモードに切り替わります。他の拡張子のファイルを編集する場合も、**M-x yatex-mode**とすれば強制的にやてふモードに突入可能です。

モードというのは、今までにもなにげに出てきてはいたのですが、編集するファイルの性質や今書いている内容に合わせて、Emacsが振る舞いを変えてくれることです。ステータス行のカッコ内に表示されているのが主モードで、そのほかに同じ性質のファイルを編集しているけどユーザーの指定によって「詰め込み」動作をオンにしているよ、といった副モードが表示されていることがあります。

これまでみてきたモードは、メールのタイトル一覧などという特殊なモードもありましたが、ほとんどはTextモードでした。しかし、プログラムを書くときに、CやPerlといった言語に応じた動作をしてくれることもEmacsを使っていてうれしい点です。〈TAB〉を押したらとたんに文字配置がガラッと変わるなど、モードを変更したときの効果は目立つものですが、Emacsは1文字入力されるごとに、「今はこのモードだからこういう動作をすればいいんだな」と判断し、マクロを実行しているのです。

文書執筆中にお世話になるのは、表1のようなショートカット集で、これにはTeXコマンドの強烈な補完機能があります。TeXの原稿を書いていて面倒なのは「うーん、たかがセンタリングのために“ \begin{center}…\end{center} ”なんてタイプするのはウザいなあ」ということなのですが、これなら**C-c b c**の3タッチでOK。キーのシーケンスはすべてC-cで始まりますが、あとは語呂合わせで覚えられるので、ちょっと面倒でも以下に説明するカテゴリと短縮形の由来を覚えてしまいましょう。たとえば、C-c b cはbegin centerの略です。C-cは好きなように変えられますが、ここでは一応デフォルトで説明することにします。

▼リスト1　~/.emacsの記述

```
....
(setq auto-mode-alist
    (cons (cons "\\.tex$" 'yatex-mode) auto-mode-alist))
(autoload 'yatex-mode "yatex" "Yet Another LaTeX mode" t)
(setq load-path (cons (expand-file-name "/usr/local/share/emacs/yatex")
    load-path))

(setq tex-command "platex")
(setq YaTeX-kanji-code 3)
....
```

適宜YaTeXのマクロがあるディレクトリをセット

platexを利用する場合

EUCの場合、SJIS = 1/JIS = 2

■begin…end型コマンド

これは最も重要なカテゴリでしょう。

```
\begin{……}
   ⋮
\end{……}
```

の形をしているコマンドです。手で直接タイプすると\beginと\endの対応が取れなくなりがちですが、それを防ぐためにも、やてふの機能を使うのは意味のあることです。

またTeXのコマンドは1バイト英数記号でなくてはならないのですが（自分で定義したものは別ですが）、やてふのショートカットを使うかぎり入力メソッドのオン／オフをいちいち切り替えずにすみます。日本人にとって、この効果は非常に大きいものです。

このカテゴリのコマンドは**C-c b**で始まります。C-c bの後ろ1文字で完結するコマンドを表1にまとめてあります。特別なショートカットがないコマンドでも、**C-c b〈SPC〉**とタイプすると、最初の何文字かをタイプして〈TAB〉などで補完することができます。

表中の＊2のコマンドのオプション引数なども、可能なかぎり補完が効くようになっています。\begin{thebibliography}などの死にそうに長い単語（しかも綴りがよくわから〜ん）を入力するときは、「補完があって本当によかった」となごんでしまいます。

\begin…\endは、プログラミング言語のようにインデント（段付け）をすると見やすくなる、とKnuth先生の本を始めとするTeXのチュートリアルにはよく書いてあります。やてふは、デフォルトで1文字下げのインデントをしてくれます。インデント幅を手動で変えてやると、中の部分も以後変わりますし、〈TAB〉によって適当な幅までインデントしてくれます。

＊1のコマンドのうちitemizeとenumerate環境では、最初に“\item”と表示されて、いきなり最初の項目を入力できるようになります。この項目を入力し終わったあとで〈RET〉ではなく〈ESC〉〈RET〉をタイプすれば、次の“\item”が自動的に表示されます。注目してほしいのは、現在どこの部分を編集しているのかを常にやてふが監視しているということです。すなわち、**C-c b i**の直後にだけ〈ESC〉〈RET〉が効くのではなく、ポイント（カーソル）移動中にやてふが「今\begin{itemize}…\end{itemize}の間に入ったな」と思うとこの機能をオンにしてくれるので、あとから付け足したい項目ができてもこの機能は使えるわけです。

tabbing環境、tabular環境などでは、“& & …… \\”まで必要な数だけ自動生成してくれるので、嬉しくて涙で前が見えなくなっても“&”の数が合わなくなって怒られることが少なくなります。同様なものに、各種の数式モードがありますが、数式モードでのやてふの動作は多少異なったものになりますので、あとでまとめて説明しましょう。

ほかにも細かいことでは、**C-b c〈SPC〉**のあとに表示されるデフォルトの候補（〈RET〉をタイプするだけで確定・挿入される）は、原則的に前回と同じものですが、たとえば\begin{figure}…\end{figure}の中では第一候補が\begin{caption}…\end{caption}になるなど、実際の使用局面で最も便利になるようにできています。

なお、\end{...}を誤って消してしまったとか、手動で\begin{...}を入力してタイプセット時に「対応がとれない」と怒られた場合などは、**C-c e**としてみましょう。

■section型コマンド

これも重要なカテゴリです。\section{...}のように、引数を1つ以上取るコマンドです。

C-sとタイプすると、ミニバッファに引数を入力できます。これだけでは、あまりありがたみがないように思えま

コマンド	説明	コマンド	説明
C-c b c	\begin{center}…… \end{center}	C-c b m	\begin{minipage}…… \end{minipage}＊2
C-c b r	\begin{flushright}…… \end{flushright}	C-c b v	\begin{verbatim}…… \end{verbatim}
C-c b l	\begin{flushleft}…… \end{flushleft}	C-c b d	\begin{document}…… \end{document}
C-c b e	\begin{enumerate}…… \end{enumerate} ＊1	C-c b q	\begin{quote}…… \end{quote}
C-c b i	\begin{itemize}…… \end{itemize}＊1	C-c b Q	\begin{quotation}…… \end{quotation}
C-c b t	\begin{tabbing}…… \end{tabbing}＊2	C-c b〈SPC〉	これ以外の省略形を入力＊2
C-c b T	\begin{table}…… \end{table}＊2	C-c e	対応する\end{……}を生成
C-c b C-t	\begin{tabular}…… \end{tabular}＊1＊2	C-c >	範囲内（\beginまたは\end上は環境内）をコメントアウト
C-c b E	\begin{equation}…… \end{equation}＊1＊2	C-c <	範囲内（同上）のコメント削除

表1　begin…end型コマンド
＊1: \begin…… \end間では特殊な動作をする
＊2: ミニバッファで指定するパラメータあり

すが、たとえば、

```
\documentclass[a4j,seminar]{jarticle}
```

を入力するときは、

C-s d o c 〈SPC〉c 〈SPC〉〈RET〉

```
\documentclass[▮]
```

a4j 〈RET〉sem〈SPC〉

```
\documentclass[a4j,seminar]▮
```

jar 〈SPC〉〈RET〉

```
\documentclass[a4j,seminar]{jarticle}▮
```

のように、カッコ内も補完してくれるので、タイプの手間やスペルミスを減らす効果があります。未知のコマンド名に対して学習機能（後述）を使う場合は、**C-u 3 C-s**（引数が3つ）などと指定して、TeXが間違った学習をしないようにしましょう。

■maketitle型コマンド

引数を取らない単独のコマンドです。これも**C-m**をタイプするとミニバッファで入力コマンドを補完してくれます。

■large型コマンド

{\large ...}のように、普通はカッコ内で使うコマンドを入力するために用意されています。**C-l**をタイプすると、ミニバッファで補完が可能です。

ちなみに、{\large\bf ...}とするときに、

C-l l a 〈SPC〉〈RET〉 C-l bf 〈RET〉

とすると、

```
{\large {\bf ...}}
```

となってしまうので（もちろんこれでも構いませんが）、ユーザ辞書に“large\bf”などという組み合わせを学習（後述）させるとよいでしょう。先頭の\のみを除去して学習させるのがミソです。

■学習機能

ミニバッファで補完できるような局面では、既存の候補

Column

Knuth先生のハッカー魂

TeXの作者のKnuth先生は、計算機科学、なかでも組み合わせ論などで有名な数学者です。

先生は、コンピュータ数学のシリーズ本を執筆していたのですが、植字工が数式などの記号の活字を正しく拾ってくれなかったり、なかなか意図した位置に配置してくれないのに業を煮やして、このシステムを作ってから執筆を再開したとかしないとか。まったく「仕事してるより道具を磨いている時間のほうが長い」ハッカー魂の権化のような人です。

著者が編集者やデザイナーを兼ねられるということは、著者の思いどおりにレイアウトができる反面、「センスのない著者が書いた本は見栄えが悪い」というデメリットがあります。つまり、本来なら文章の内容を練ってわかりやすく説明するのが著者の仕事で、それを構成したりページに配置したり、ときには目を引くレイアウトにするのが編集者やデザイナーの役割です。したがってTeXには、さまざまな字や行、図や表などの配置アルゴリズムが用意されていて、「格好悪く組もうとしたら特殊な技を使わないといけない」とまで言われるほど自動化が進んでいます。

Knuth先生は数学者だけあって、英文を自動的にハイフネーション（行にまたがる単語の分割）するアルゴリズム、英文文書の「川が流れる」（単語の間の空白が版面上で川のようにつながって見える）ことを防止するアルゴリズム、段落・ページ・章の間のテキストの分割位置を最適化して、「未亡人単語（行）」（単語が1個だけのさみしい行ができたり、1行だけのさみしいページができる）を防止するアルゴリズムなどなど、商用印刷物に必要とされる品質の高い組版作法をすべてアルゴリズム化して組み込んでしまいました。

また、理系の出版物を組んだとき美しく見えるComputer Modern Roman体という書体を開発してフリーで使用できるよう公開したり、このために好きな大きさ・デザインの書体を生成できるMetafontというシステムを「ついでに」開発しています。

TeXやMetafontはプログラミング言語として見ても非常にユニークなものなので、言語大好きな方は時間があったらドキュメントをひもといてみてください。

ところで最近は、ワープロソフトでもこのテの自動化は進んでいますが、印刷文化に理解のないソフト会社が開発していたり、リアルタイムで本の品質の編集をするのはソフトが重くなるという理由で、まだまだ行き届かないものもあります。そこを勘違いしてワープロソフトで出力したものをそのまま出版している例は、書籍ではあまり見かけませんが、アマチュアの出版物や学会予稿集などではまだまだ素人臭い組版の出版物が横行していて、日本の印刷文化はダメになってしまうのかなあ、などと思ってしまいます。

にないコマンドを使いたいことがあります。やてふの不備じゃないかって？　そんなことはありません。TeXは、\newcommandコマンドなどで新しいコマンドを定義できる（勝手に作れる）のも特徴なので、辞書にないコマンドが文書内で使われるのは当たり前なのです。

新しいコマンドを使うとき、最初の1回は手でフルスペルをタイプしないといけません（当たり前です）。このとき、やてふはこの新しいコマンドを辞書に登録して学習しようとします。ミニバッファに、

```
U）ユーザ辞書 L）ローカル辞書 N）メモリ D）しない
```

と表示されるので、どの辞書に学習させるか選びましょう。uを選ぶと~/.yatexrcに記録され、次回からまた使えます。lを選んだ場合は、そのディレクトリの~/.yatexrcに学習結果が記録されます。どういうことかというと、特定の論文でのみ使うようなコマンド、たとえば\ceil{}というコマンドを次のように定義したとします。

```
\newcommand{\ceil}[1]{\lceil #1\rceil}
```

このディレクトリ内のTeXのファイルでは、section型コマンドとして\ceilというコマンドを学習させておけば、より便利に使えるというわけです。

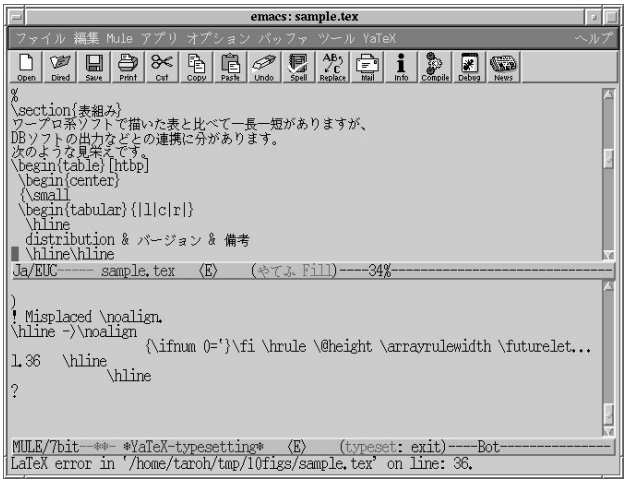
nのメモリ内の学習というのは、あまり使うことはないでしょう。一度Emacsを終了させてしまうと忘れ去られてしまいます。dは、うっかりスペルを間違ってしまい、そのまま学習されたくない場合などに選びます。ただし、辞書に学習されてしまったとしても、あわてることはありません。.yatexrcを覗いてみると、

```
(setq yahtml-user-typeface-table '(
("sebsectionmark")      ……正しくはsubsectionmark
    ⋮
```

のような記述があるので、問題の行を削除するだけです。

■コメント

大量の行を消す……ほどでもないが、原稿の一部を一時的にタイプセットの対象外にしたいときは、TeXでもプログラム同様、コメントを付けます。これには、マクロを使ってもいいのですが、**C-c >**を使うと非常に楽です。



画面2　タイプセットのエラーメッセージ

タイプセット

TeXで書かれた原稿は、TeXの処理系（プログラム）を通さないといけません。この操作は、もちろんC-x C-sで原稿をファイルにセーブして、シェルからコマンドラインでTeXを実行してもよいのですが、やてふではEmacs内からワンタッチで手軽にできるようになっています。

■タイプセット

タイプセットは**C-c t j**で実行します。このとき、コマンドが実行されたバッファはセーブされ、開いているやてふモードのバッファがあればセーブするかどうか聞かれます。また、タイプセットを起動したバッファに\documentstyleや\documentclassがなければ（ルートファイルでなければ）、たぶんルートファイルは別にあるだろうというのでルートファイル名の入力を求められます。\inputや\includeされている関連ファイルへの配慮も完璧、というわけです。

さてコラム「登場したツールのインストール」にしたがって、platexなどをインストールしている場合、TeX処理系のコマンド名（デフォルトではjlatex）が違っていますので、リスト2のように.emacsrcの設定を忘れないようにしましょう。

■エラーの発見と修正

タイプセットが始まると、新しいバッファ＊YaTeX-typesetting＊が開き、TeX処理系からのメッセージが表示されます（**画面2**）。エラーが出るとタイプセットは停止しますが、このとき**C-c '**でエラーとなった行に一発でジ

ャンプできます。

すでに述べたように、TeXのタイプセットの過程だけでは目に見える結果を得られません（なんのこっちゃ）。通常は、DVIファイルからPostScriptファイルを生成し、プレビューアを立ち上げて表示させ、必要があれば原稿を修正し、完成したらプリントアウトということになります。

プレビューと印刷にひと工夫

実は、YaTeXには**C-c t p**でプレビュー、**C-c t l**でプリントアウト、というショートカットが用意されているのですが、タイプセットの際に、ファイル変換からプレビューアの起動まで実行してしまう方法があります。

これは、タイプセットの一連のコマンドを実行するjlvというバッチファイルを作り、これを**C-c t j**で起動することで実現しています。**リスト2**は、dvipskとghostviewを組み合わせる場合の例ですので、参考にしてみてください。このバッチファイルのミソは、TeXの原稿に含まれている\documentclass行や\usepackage行を見て、ドキュメントのできあがりのサイズや縦横をdvipskやghostviewに与えているところです。

プレビューアで繰り返し見て間違いがなければ、あとは紙に出力するだけです。ghostviewなら〈P〉で全ページ出力とか、〈p〉で選択ページの出力ができてしまいます。

と、ここまでの操作をワンタッチにするには、PostScriptプリンタが必要ですが、lprのフィルタをちゃんと設定しておけば、安価なプリンタでPostScriptファイルがプリントアウトできます。ぜひチャレンジしてみてください。これについては、コラム「登場したツールのインストール」で紹介したLinux-HOWTOなどを参考にしてください)。

YaHTMLモードでWebページ作成

YaTeXに指が慣れてしまった読者のみなさんは、Webページも同じマークアップ型言語なのだから……ということで、あるいは同じ方法でHTMLを書いてみたくなるかもしれません。YaTeX作者の広瀬さんもそうだったようで、なんとYaTeXにはYaHTMLモードという、HTML執筆のためのマクロが付いています。もちろんこのネーミングには、オリジナルのHTMLモードに対するyet anotherの意味合いがあります。

とはいっても、YaTeXモードのキー操作がTeXのコマンド名からの連想だったのに対し、YaHTMLモードでは単にYaTeXとの操作統一が図られているだけなので、HTMLを書くだけのためにYaTeXパッケージを導入して使いこなすのはちょっと骨が折れるかもしれません。

表2に主要なキー操作を示します。表からわかるように、YaTeXと対応する動作が同じキー操作でできるようになっています。

また、\section{...}と〈h2〉……〈/h2〉のように、TeXでは単独のコマンドなのに、HTMLでは環境になっているものも多いため、C-c b型とC-c s型のどちらでも入力できるようになっています。ただし、C-c b型のものは（短縮形が用意されていることも利点ですが）開始タグ・空行・終了タグの計3行が挿入されて空行にポイントが移動するのに対し、C-c s型のものは開始タグと終了タグが1行に挿入され、タグの中間にポイントが移動します。

このほか、タイプセットの実行にあたる**C-c t p**で、Netscapeが立ち上がってできあがりをブラウズできますが、同じファイルを繰り返し修正するなら、セーブしてブラウザで「Reload（再読み込み）」したほうが手軽でしょう。

コマンド	説明	コマンド	説明
C-c b h	<html>……</html>	C-c b f	<form>……</form>
C-c b H	<head>……</head>	C-c b s	<select>……</select>
C-c b b	<body>……</body>	C-c B ……	範囲を上と同様のコマンドで囲む
C-c b t	<title>……</title>	C-c b 〈SPC〉	任意の<コマンド>……</コマンド>
C-c b 1	<h1>……</h>	C-c l コマンド	<コマンド>……</コマンド>
C-c b 2	……		（1行内完結型、……など）
C-c b 3	……	C-c s コマンド	<コマンド>
C-c b u	……		（環境になっていないタグ型、<img ……>など）
C-c b d	<dl> <dt>…… </dl>	C-c >	指定範囲をコメントアウト
〈ESC〉〈RET〉	改行後 や<dt>を挿入	C-c <	指定範囲のコメント削除
C-c b c	……	〈ESC〉 C-a	対応する開始タグ先頭へ
C-c b T	<table>……</table>	〈ESC〉 C-e	対応する終了タグの次へ
C-c b a	<a>……	C-c t p	プレビュー

表2 YaHTMLの代表コマンド

▼リスト2　一括プレビューのためのバッチファイル

■~/.emacs

```
....
(setq tex-command "jlv")
....
```

■/usr/local/bin/jlv

```
#! /bin/sh

SIZECOMMAND=/usr/local/bin/tex_documentsize
LATEX=platex
DVI2PS=dvips
#DVI2PS=dvi2ps

if [ $# != 1 ]; then
  echo 'usage: jlv FILENAME[.tex]'
  exit 1
fi
BASENAME=`echo $1 | sed -e 's:\.tex$::'`

if [ ! -f $BASENAME.tex ]; then
  echo $BASENAME.tex not found.
  exit 1
fi

DVI2PSOPT=`$SIZECOMMAND $BASENAME.tex dvips`
#DVI2PSOPT=`$SIZECOMMAND $BASENAME.tex dvi2ps`
GVOPT="-antialias `$SIZECOMMAND $BASENAME.tex
ghostview`"
$LATEX $BASENAME.tex || exit $?
echo $DVI2PS $DVI2PSOPT $BASENAME.dvi -o $BASENAME.ps
$DVI2PS $DVI2PSOPT $BASENAME.dvi -o $BASENAME.ps
#echo $DVI2PS $DVI2PSOPT $BASENAME.dvi \> $BASENAME.ps
#$DVI2PS $DVI2PSOPT $BASENAME.dvi > $BASENAME.ps
echo gv $GVOPT $BASENAME.ps
gv $GVOPT $BASENAME.ps || exit $?
exit 0
```

■/usr/local/bin/tex_documentsize

```
#! /bin/sh

if [ $# != 2 ]; then
  echo 'usage: '$0' FILENAME.tex
{dvi2ps,dvips,ghostview}'
  exit 1
fi
FILENAME=`echo $1 | sed -e 's/\\.ps$//' -e 's/\\.dvi$//'
-e 's/\\.tex$//'`.tex

if [ ! -f $FILENAME ]; then
  echo $FILENAME not found.
  exit 1
fi

head -100 $FILENAME |\
sed -e 's/\\%/-percent-/g' -e 's/%.*$//' |\
sed -n \
 -e '/\\document[a-z]*\[[^\]*\]{.*}/s/\\document[a-
z]*\[\([^\]*\)\].*$/\1/p' \
 -e
'/\\usepackage{.*}/s/\\usepackage{\([^}]*\)}.*$/\1/p' |\
gawk -F',' '
BEGIN {
    type = "a4";
    port = "portlait";
    opt  = "'$2'"
}
{
    for (i = 1; i <= NF; i++) {
    if ($i == "b4j" || $i == "b4" || $i == "b4paper") {
        type = "b4";
    } else if ($i == "a4j" || $i == "a4" || $i ==
"b4paper") {
        type = "a4";
    } else if ($i == "b5j" || $i == "b5" || $i ==
"b5paper") {
        type = "b5";
    } else if ($i == "a5j" || $i == "a5" || $i ==
"a5paper") {
        type = "a5";
    } else if ($i == "landscape" || $i == "seminar") {
        port = "landscape";
    }
    }
}
END {
    if (opt == "dvi2ps") {
        if (type != "a4") {
            printf("-o %s ", type);
        }
        if (port != "portlait") {
            printf("-o %s ", port);
        }
    } else if (opt == "dvips") {
        printf("-t %s ", type);
        if (port != "portlait") {
            printf("-t %s ", port);
        }
    } else if (opt == "ghostview") {
        if (type != "a4") {
            printf("-media -%s ", type);
        }
        if (port == "landscape") {
            printf("-seascape ");
        } else if (port != "portlait") {
            printf("-%s ", port);
        }
        printf("-magstep -2");
    }
    printf("\n");
}'

exit 0
```

Column

登場したツールのインストール

今回紹介したツールは以下の方法で入手できます。インストールには、http://www.linux.or.jp/JF/JFdocs/pLaTeX2e.htmlなどがよい参考になります。Vine Linuxなどではtexのカテゴリを選ぶと、以下にあげる関連ツールはほぼ整います。

■ptex2e/platex2e

これがないと始まりません。各ディストリビューションでは、最新版のpLaTeX2e-19990809とpTeX2.1.8がパッケージ化されていることが多いです。

■yatex

今回、TeXの原稿書きの目玉となるEmacsマクロです。Emacsにはもともと「TeXモード」があるのですが、いまいち賢くなくてお勧めできません。YaTeXの名前は、このTeXモードに対するyet another（もうひとつの）であることに由来しています。

Emacs19／XEmacs用のYaTeX 1.68がRPMで入手可能ですが、http://www.yatex.org/ を参考にインストールするのも難しくはありません。

■dvipskまたはdvi2ps

今回、自動タイプセットに使っています。dvipskは生成されるPostScriptファイルにビットマップのフォントが埋め込まれるため、dvipsk-vflibを用いてTrueTypeのきれいなフォントを購入すれば、プリンタに依存せずきれいなフォントで出力できます。また、ghostviewでも適切なフォントが表示されます。

dvi2psのほうはプリンタフォントを利用しているようで、PostScriptファイルが小さくなりますが、日本語フォントを持たないプリンタで出力したり、日本語化されていないghostviewを用いたり、PDFに変換して利用する場合などは注意が必要でしょう。

dvipskは5.78aの日本語版が、dvi2psは2.0の日本語版がRPMなどで入手することができます。

■JISビットマップフォント／その他のTrueTypeフォント

dvipsk/dvi2ps（+vflib）またはghostscript（+vflib）と組み合わせて使います。DVIファイルにはビットマップ情報がいっさい含まれていないため、PostScriptファイルを生成する際、またはプレビューする際に字形を与える必要があります。

このために最近では、各種形式のフォントを統合して利用できるvflibというライブラリが、dvipsk/dvi2ps、ghostscript、Xサーバなどで利用できるようになっています。この大きな利点は、Windows向けに発売されている安価なTrueTypeフォントが利用できることでしょう。筆者は個人的に、何十種類もあるような色物フォント（笑）のパックより、実用的なRyobiのフォントが大好きです。またフリーのフォントもRPMになっていたりします。ほかにもWindowsとデュアルブートにしているマシンなら、/mnt/msdos/windows/fonts/なんていうLinuxのファイルシステムの中にどういうわけかあまり綺麗でない日本語フォントが転がっている可能性があります。

■ghostscript／ghostview

Emacs内から一発でプレビューする際に必要です。またPostScriptプリンタではないプリンタに、ビットマップ出力する際にもフィルタとして（レンダリングエンジン）もっとも使われます。たとえばlprコマンドでPostScriptファイルを出力できるようにちゃんと設定しておけば、TeX以外の文書出力でも幸せになれるでしょう。

http://www.sat.t.u-tokyo.ac.jp/~hideyuki/Ghostscript/truetype.html によると、特にvflib対応していないghostscriptでもTrueTypeフォントが利用できるようです。最新版のghostscript5.50、ghostview（gv）3.5.8がRPMで入手可能です。

■以上のツールに関する参考文献

http://www.linux.or.jp/JF/JFdocs/JPprinting-mini-HOWTO.html
http://www.matsusaka-u.ac.jp/~okumura/texfaq/install-linux.html
http://www.protein.osaka-u.ac.jp/chemistry/matsuura/install-memo/linuxppc/rpm2html/Japanese_RPM_（JRPM）.html

■gnuplot／tgif／gimp／graphicxパッケージ

今回の記事には登場しませんでしたが、最終的にできあがった文書をPostScript形式に変換する環境なら、これらはぜひともインストールしたいものです。gnuplot、tgif（4.1.39が最新）とGIMP（1.1.27が最新）はRPMなどのパッケージから導入できるでしょう。graphicxパッケージは、pLaTeX2eのマクロ集のパッケージが用意されているディストリビューションもありますが、FTP慣れしている人は、TeXのマクロ群を集めたCTAN（http://www.ctan.org）を覗いてみるのもよいでしょう。

gnuplotでは出力形式としてset term postscriptを選ぶとPostScriptファイルを生成できますが、実用的なグラフを作るにはtgifで凡例や軸名称などをいじらないといけないので、set term tgifを指定してtgif形式のファイルを生成します。tgifで作成した図は、出力形式としてLATEXを選ぶとPostScript（EPS形式）でセーブされます。

GIMPで操作した写真や図などは、「ファイル名を指定して保存」でPostScriptを選ぶと、PostScriptファイルが生成されます。

これらで用意した図は、以下のリストのようなLaTeXの命令で文書に貼り込むことができます。ほかにもユニークなマクロが大量にありますので、http://www.linux.or.jp/JF/JFdocs/pLaTeX2e-3.htmlなどを参考にしてください。

▼リスト　PostScriptファイルの利用

```
\documentclass{...}
....
\usepackage{graphicx}
....
\begin{figure}[htbp]
\begin{center}
\resizebox{0.7\hsize}{!}{\includegraphics{hoge.eps}}
\end{center}
\caption{\label{fig:hoge}
 \bf 試しに入れてみた図}
\end{figure}
....
```

Linux日記

第16回　メール配送（3）

いよいよ、sendmailによるメール配信についての解説です。sendmailを直接実行したり、SMTPでsendmailと会話してメール配信の仕組みを学んでみましょう。

文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

DNSだとかsendmailを動かすといった作業は、慣れてしまえばどうということもないが、初めてやる時はけっこう大変なものだ。こういったことについて解説記事を書いているときには、また別の苦労がある。画面やリストの作成である。解説記事には、設定ファイルの例だとか、GUI設定ツールの画面などが不可欠である。これらの作成は、場合によっては原稿を書く以上の手間がかかる。なぜなら、例示するデータを捏造しなければならないことがあるからだ。

実際に実験、あるいは運用している環境のデータをそのまま使ってしまえば、話は簡単だ。単純なネットワーク設定などであれば、適当なプライベートIPアドレスの環境をでっちあげて、そこですべて動かせばそれで済む。ところが、このところ書いているDNSとかsendmailのような、複数のサーバ（それもインターネット上のもの）が連携するようなシステムだと、それだけでは済まない。たとえ簡単な実験でも、実際にインターネットに接続し、ちゃんとパラメータを設定しなければならない。かといって、これらのパラメータ、つまり、実験に使っているドメイン名だとかIPアドレスだとかをそのまま公表する気にはなれない。かくしてデータの捏造が始まる。

リストはまだ簡単だ。UNIXであればたいていは設定ファイルがあるし、対話セッションならscriptコマンドでファイルに落とせる。後はエディタを使って適当にIPアドレスだとかドメイン名を書き換えれば、捏造リストはできあがりである。

辛いのはGUIツールを使っている場合だ（この連載では、そんなものはもちろん出さないが、Windows関係の仕事をしていると、これがほとんどだ）。画面を画像ファイルとして落とすのは簡単だが、その中身の捏造をどうするか？　文字や数字を画像として確保しておいて、画面を直接書き換えるという方法もないではないが、あまりにも手間がかかりすぎる。そこで、最初から適当な捏造パラメータを使ってセットアップするのである。つながってもいないネットワークのIPアドレスを設定してみたり、架空のドメイン名を入力してみたり。とにかく必要な画面さえファイルに落とせれば、動かなくてもいいのだ。

たいていはこの方法でうまくいくのだが、まれにうまくいかないことがある。プログラムがご丁寧に入力データを検証し、「間違ってるよ」と指摘してくれるのだ。「そんなこたぁ百も承知でぃ」と怒鳴ったところでどうにもならない。持てる知識を総動員して、どうにかごまかすのである（場合によっては、画面撮りのためだけに、ダミーのサーバを運用したりしなければならないことなどもある）。

さんざ苦労して画面ファイルを作成したあげく、裏のウィンドウに本物のIPアドレスやドメイン名がさりげなく表示されていて、最初からやり直しなどということもある。「やっぱ、システム設定はテキストファイルが一番だな」

Column

Windowsの画面撮り職人

普通にやったら、どうしても画像ファイルを取得できない画面がある。たとえばシステムのインストール中の画面などだ。このような場合は、あきらめて写真撮影で済ますことも多い。だが、このような画面撮りをやってのけた編集者がかつて某編集部にいた。

Windowsの場合のやり方である。Windowsプログラムでは、ダイアログなどはリソースとして実行ファイル中に記録されている。これを、リソースエディタなどを使って抽出する。そして、このリソースを別に用意したVisual Basicなどのプログラムに組み込むのである。ダイアログ中に表示される文字などはVBの側から制御する。かくして、システムインストーラの画面が、通常の実行環境上で再現されるわけだ。よくやったもんだと思う。

と、こういう時は思うのである。

■sendmail

先月まではローカルホスト上でのメール配信についてを解説した。さて、そろそろsendmailの話に入ってもいい頃だろう。

では、sendmailなどのいわゆるメールサーバプログラムは、ローカルマシン宛のメールを配信するために必要なのだろうか？　歴史的経緯からいえば答えはイエスであるが、実際のシステム運用という面で見れば、答えはノーである。実際、sendmailのようなプログラムが必要になったのは、リモート配信を行うためである。ローカル配信しかしていなかった時代は、binMailだけでよかったからだ。

メールシステムを実際に運用する場合は、ローカル配信だろうとリモート配信だろうと、一貫した環境で行えることが望まれる。同じようにアドレスを指定してメールを送り、それがリモート配信されるのかローカル配信されるのかは、MTAまかせという方法である。したがって、現在のメールシステムは、たとえローカル配信であっても、sendmailが介在するようになっている。

まず、ユーザーがucbMailのようなMUAを使い、メールを作成する。To:やSubject:などのヘッダはユーザーが指定し、From:は、MUAかMTAが用意する。UNIX上で動作している場合は、ユーザー名とpasswdファイルの情報が使われる。

このようにしてMUAで作成されたメールは、sendmailプログラムに渡される。メールシステムが稼働しているUNIX上でpsコマンドを実行すると、デーモンとして動作しているsendmailのプロセスがあるが、MUAがやり取りするsendmailはこのプロセスではない。このsendmailは、リモートホストから送られてくるメールを待ち受けているデーモンだ。

MUAはsendmailを子プロセスとして起動する。メールを受け取った子プロセスのsendmailは、宛て先アドレスを調べ、ローカル配信であれば、今度はsendmailが子プロセスとしてbinMailを実行してメールを渡し、そのメールをローカルホスト上のシステムメールボックスに書き込む。リモート配信の場合は、どこかで待ち受けているメールサーバにどうにかして送ることになる（**図1**）。

このような形にすることで、リモート配信でもローカル配信でも同じ構成でメールを送ることができるようになる。どのように配信を行うかはすべてsendmailまかせで、MUAは宛て先アドレスを指定するだけだ。またsendmailは、どのように配信するかを判定したら、実際に配信を行う別のプログラムを呼び出すことができる。ローカル配信の場合は、mail.localやprocmailである。このような、実際の配信を行うプログラムを配信エージェントと呼ぶ。sendmail自体も配信エージェントであるが、ローカル配信用など、さらに特化した配信エージェントを内部から呼び出すのである（以下、mail.localやprocmailのようなローカル

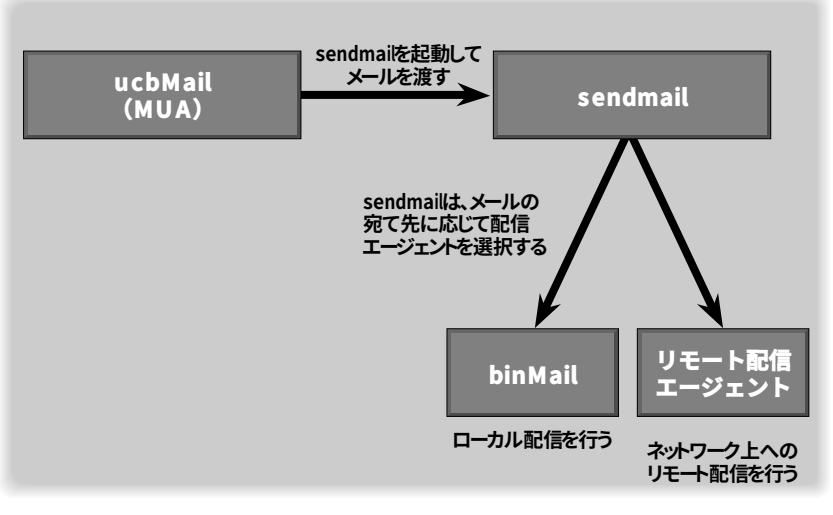


図1　sendmailを使った配信

配信エージェントを総称してbinMailとする)。

ここで重要なことは、電子メールサーバと一言でいっても、常時動作しているデーモンプロセスがすべてのメール処理を行っているわけではないということだ。MUAの子プロセスとして起動されるsendmailは、デーモンプロセスとして動作しているsendmailと同じプログラムであるが、サーバとして動作するわけではない。ユーザープロセスの一部として動作することになる。前に電子メールシステムが単純なサーバ／クライアント形式ではないと書いたのは、こういう面があるからだ。そして、メール配信に関与する裏方プログラムのことを、サーバではなくMTA(Mail Transport Agent)と呼ぶのもこういった理由による。配信処理を行うのは、サーバだけではないからだ。ここまではクライアント(MUA)に対して簡単にメールサーバという用語を使ってきたが、以後、より正確に使い分けていこう。具体的には、外部からの接続を待ち受けているsendmailデーモンはメールサーバと呼んでも間違いではないが、MUAから実行されるsendmail子プロセスはサーバとは呼べず、MTAとしか呼びようがないということだ。

■sendmailを使ったローカル配信

さて、UNIXのプログラムの中で、もっとも設定が難しいといわれるsendmailの話に入ろう。sendmailは、設定の難しさだけでなく、その動作モードの多さでもUNIX有数のものでなかろうか。とりあえず、psで表示される待ち受けモードのsendmailデーモンがある。そして、配信のためにMUAから実行されるsendmailがある。sendmailという名前で使われるのはおもにこの2種類だが、これとは別にメールキューの表示(mailq)、エイリアスデータベースの再構築(newaliases)、状態表示(hoststat)などのメール管理ツールがsendmailへのリンクになっている(つまり、実際に実行されるのはsendmailプログラムなのである)。

それぞれのsendmailの役割や動作についてはおいおい説明していくとして、ここではローカル配信におけるsendmailの役割を見てみよう。MUAでメールを送信すると、子プロセスとしてsendmailが実行される。これは次いでbinMailを実行してメールをローカル配信する。このsendmailの実行を、実際に手作業でやってみよう。こういった実験が簡単にできるのがUNIXのいいところだ。WindowsやMacではこうはいかない。

ここで使うsendmailは、mail.takobeya.comというホスト上で動作しており、user@takobeya.comという形式でメールを扱うようにすでに設定されている(user@mail.takobeya.comではない)。

この例に限ったことではないが、この種の実験はログにおかしな記録を残したり、システムメールボックスをおかしな状態にしてしまうことがあるので、自分が管理権限を持っているマシンで、実験用ユーザーアカウントを使って実験すること。

画面1のようにsendmailを直接実行した。acctonコマンドとlastcommコマンドを使って調べてみると、sendmailだけでなく、binMail(procmailまたはmail.local)も実行されているのがわかる。先頭のFromと末尾の空行はbinMailが付加したものだ。binMailはこれら以外は付加しないので、前々回に行ったmail.localの実験ではTo:やFrom:はユーザーが入力した。sendmailを使う場合は、いくつかのヘッダをsendmailが生成してくれる。実行例ではメールの本文しか入力していないが、スプールされたメールにはDate:、From:などのヘッダが付加されている。では、mail.localの実験の時のように、メールにあらかじめヘッダを入れておいてみよう(**画面2**)。

違いは明らかである。sendmailは、ヘッダがなければ自身で必要なヘッダを生成し、メール中に埋め込む。**画面1**の例では、システムから得た情報でFrom:ヘッダを作成していることがわかる(From: Test account <test@takobeya.com>)。すでにヘッダがあれば、既存のものを使う。ただし、多少の書き換え処理が行われている。**画面2**で入力したメールのTo:、From:に

```
[test ]$ /usr/sbin/sendmail test                    ← 宛て先はtest(自分自身)
test of sendmail                                    ← メールの本文
[test ]$ more /var/spool/mail/test                  ← スプールを見てみる
From test Mon Sep  4 19:00:12 2000                  ← この行はbinMailが付加したもの
Received: (from test@localhost)                     ← 以下のヘッダはsendmailが付加したもの
        by mail.takobeya.com (8.9.3/3.7W-2.8compat) id SAA36874
        for test; Mon, 4 Sep 2000 18:59:39 +0900 (JST)
Date: Mon, 4 Sep 2000 18:59:39 +0900 (JST)
From: Test account <test@takobeya.com>
Message-Id: <200009040959.SAA36874@mail.takobeya.com>

test of sendmail                                    ← 入力したメールの本文

[test ]$
```

画面1　sendmailを直接実行してみる

は、you、meというユーザー名しか指定していなかったが、これにドメイン名が付加され、正式なメールアドレスに変わっている点に注意（ドメイン名が省略されている場合は、デフォルトドメインとして自身のドメイン名を使う）。また、To:ヘッダは、sendmailによって生成されていないこともわかる（**画面1**の例にはTo:ヘッダがない）。To:を使うにしろCc:を使うにしろ、宛て先を指定しないメールというのは通常はないからだ。

ここで重要なのは、mail.localの実験の時と同じように、配信にはsendmailに引数で渡されたエンベロープ情報を使っているということだ。必要なヘッダがなければ作成するが、実際の配信はsendmailに渡された引数と、プログラムを実行したユーザー名に基づいている。メール中のTo:に関らず、メールはtestに送られており、そしてメールボックス中のFrom行は、sendmailを実行したユーザーtestの名前が記されている。別の見方をすれば、To:ヘッダはメールにおいて必須ではないということでもある。エンベロープ情報で宛て先が指定されていれば、メールはTo:やCc:といった宛て先ヘッダがなくても相手に届くのである。

宛て先をBcc:だけで指定し、受信者はほかの誰に同じメールが送られているかがわからなくする方法を前回紹介した。Bcc:で指定した宛て先は、エンベロープ情報としてメールを宛て先に送信するが、Bcc:ヘッダはメール中に残されないので、届いたメールには宛て先情報が記述されていないのである。

エンベロープ情報がどのように指定されるかは、配信エージェントの構成に大きく依存しており、メール中に現れることはあまりないが（binMailが追加したFrom行は例外的なものだ）、メール配信においてエンベロープ情報が重要であるということだけは覚えておいてほしい。

さて、ここでsendmailが生成したほかのヘッダも見てみよう。

・Received:ヘッダ

sendmailがメールを受け取り、何らかの配信処理を行うたびに、メールの先頭にReceived:ヘッダが付加される（先頭のFrom行は、sendmailの処理後にbinMailが付加したものだ）。このヘッダの形式はsendmailの設定に依存するが、基本的にはsendmailがいつどこから誰宛のメールを受け取ったかを示している。この情報は長くなるので、数行に分割されている。インデントされている行は継続行である。

ここではローカルホスト上でやり取りしているので、このホストの情報しか表示されていないが、ネットワークなどを介したリモート配信を行っている場合は、途中で中継されるごとにReceived:ヘッダが増えていく。受信したメールのReceived:ヘッダを見れば、そのメールがどのような経路で配信されてきたかがわかる。また、メールに付けられたIDも含まれているので、中継したMTAのログと照らし合わせることも可能だ。

Received:ヘッダの数を数えることで、そのメールが何度中継されたかがわかる。これをホップ数という。ホップ数が異常に多い場合、メールがループしている可能性がある。Received:ヘッダは、このようなエラーメールの検出にも使える。

・Date:ヘッダ

メールを発信した日時である。メール中にこのヘッダがなければsendmailが作成して付加し、あればそのまま残すので、配信に要した時間に関らず、発信側MUAか最初のMTAが設定した日時が最後まで維持されることになる。

・Message-Id:ヘッダ

メールを識別する一意なIDである。これは、インターネット上で一意に定まるドメイン名と、そのホスト上で一

```
[test ]$ /usr/sbin/sendmail test
To: you                                          ヘッダを手で入力
From: me
Subject: test #3

test mail                                        メールの本文
[test ]$ more /var/spool/mail/test
From test Mon Sep  4 19:05:09 2000
Received: (from test@localhost)
        by mail.takobeya.com (8.9.3/3.7W-2.8compat) id TAA36888
        for test; Mon, 4 Sep 2000 19:04:41 +0900 (JST)
Date: Mon, 4 Sep 2000 19:04:41 +0900 (JST)
Message-Id: <200009041004.TAA36888@mail.takobeya.com>
To: you@takobeya.com
From: me@takobeya.com
Subject: test #3

test mail

[test ]$
```

画面2　ヘッダを含んだメールをsendmailに渡す

意に定まるIDを組み合わせたものである。これもDate:と同じく、なければsendmailが作成し、あればそのまま残す。従って、最初に付けられたIDが配信途中で変わることはない。

ユーザーがメールをやり取りする上で、Message-Idはさほど重要なものではないが、何らかの理由で中継記録などを調べなければならない場合は、このIDをログ中から検索することになる。

■リモート配信

さて、sendmailを使ったローカル配信がどのように行われているかを見たので、今度はリモート配信について考えてみよう。

リモート配信の形態としては、常時接続のIPネットワーク（インターネットなど）での配信に使われるSMTP接続、ダイヤルアップ接続で配信するUUCP接続などがある。UUCP接続については、そのうち解説するかもしれないUUCPの話題のときに触れるとして、ここではインターネットで広く使われているSMTP接続について解説する。

SMTPを使った配信には、発信側のMTAと受信側のMTAが関与する。ホスト上で常時稼働しているsendmailデーモンは、外部からのSMTP接続を待っている受信側MTAである。発信側のMTAは、MUAから実行されたsendmailだったり、中継を行うように設定されたsendmailデーモンになるだろう。

MUAでリモートホスト上のアドレスに送るメールを作成し、送信すると、ローカル配信の時と同じように発信用sendmailが起動される。sendmailはリモート配信であることを判定すると、binMailを呼び出す代わりに、SMTP配信を行う。sendmailは自身でSMTP接続を行うことができるので、binMailのような配信サブプログラムを呼び出すことなく、自身がIPコネクションを確立し、メールを送ることができる。SMTP配信の詳細については後で解説しよう。

このようなSMTP接続リクエストを受ける側では、sendmail（別にsendmailである必要はないのだが、ここではsendmailとしておこう）がデーモンモードで動作している。このsendmailは、外部からのSMTP接続リクエストに対して待機しており、接続リクエストを確認すると、SMTP接続を確立し、発信側MTAから送られてくるメールを受信する。この時、発信側のsendmailが指定するエンベロープ情報は、基本的にメール中のTo:、From:に基づいたものだろう。

受信側のsendmailは、受信したメールを調べる。このような形で配信された場合、通常は、To:の宛て先、エンベロープ情報とも、自身が管轄するユーザーのアドレスを指定しているだろう。つまりこのメールは、このホスト上でユーザーのシステムメールボックスにローカル配信しなければならないということだ。そこでsendmailはbinMailを呼び出し、受信したメールをユーザーのスプールファイルに書き込む（メールをbinMailに渡す前に、自身のReceived:ヘッダも追加する）。これで、メール配信が完了したことになる（**図2**）。

■SMTP配信

IP接続された環境では、MTA間の

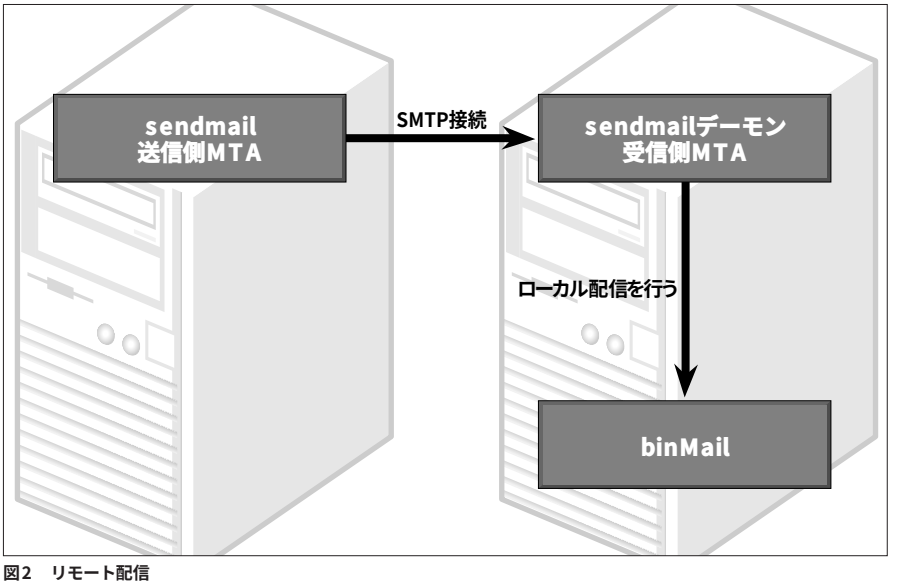


図2　リモート配信

コマンド	説明
HELO	SMTP通信を開始する。クライアント（接続側）は自身のドメイン名を表明する。これに対して、接続されたサーバ側も自身のドメイン名を表明する
MAIL	メールの発信者を示す
RCPT	メールの受信者を示す
DATA	メールメッセージの転送を開始する
RSET	転送中止
VRFY	相手側にユーザー情報を問い合わせる
NOOP	何もしない
QUIT	セッションを終了する

表1　SMTPの主なコマンド

配信にはSMTP（Simple Mail Transfer Protocol）というプロトコルが使われれる。これはその名の通り単純なプロトコルで、発信者、受信者を指定するエンベロープ情報と、メールメッセージ本体を送るだけである。SMTPでサポートされているコマンド（抜粋）を**表1**に示す。

SMTPは、TCPポート25番を使う。それでは、telnetを使い、実際にローカルホストの25番ポートでsendmailとおしゃべりしてみよう。通常はこの接続はリモートホストに対して行われるものだが、ここでは実験としてローカルホストに接続している（**画面3**）。各行の先頭の数字は、記事中から参照するための行番号だ。

実行例を順番に見ていこう。まず、telnetでローカルホストの25番ポートに接続している。このポートでは、デーモンモードのsendmailが待機している。4行目まではtelnetが表示している情報で、5行目以降がsendmailとのセッションである。デーモン側の応答行の先頭の数値は、応答の意味を識別するための数字で、プログラム間で対話するときは、この数値で相手の応答を解釈する。

接続すると、sendmailデーモンがメッセージ（5行目）を表示し、入力待ちになる。接続側（telnet）で「HELO localhost」とタイプする（6行目）。HELOは通信を開始するコマンドで、そのあとには接続側のドメイン名（ここではlocalhost）を指定している。これに対して、デーモンも自身のドメイン名を返してくる（7行目）。

接続側は、8行目でメールの発信者を指定し、10行目で受信者を指定している。これがSMTP配信におけるエンベロープ情報である。デーモンはそれぞれに対して確認応答（9行目と11行目）を返してくる。

ここまでくれば、メール本文を送ることができる。まず、DATAコマンドを送る（12行目）。これに対して肯定の応答が返ってくれば、メールメッセージを送ることができる。メッセージの終了は、行頭のピリオド1文字である（18行目）。あとはQUITを送り、セッションを終了する。

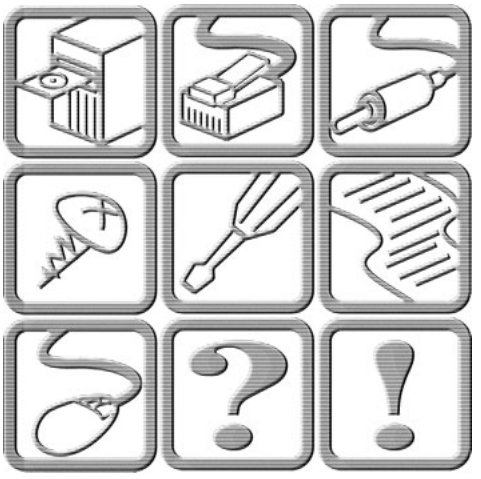
sendmailデーモンは、このようにして受け取ったメールをbinMailで配信する。スプールを見てみると、今タイプしたメールが届いているのがわかる。付加されたヘッダ類については、sendmailを手で実行して行ったローカル配信の場合と同様である。

この例は、ローカルホスト上のsendmailデーモンにメールを送ったが、通常は外部の別のホストから接続され、送られてくるメールも、もっときちんとヘッダ類が付けられたものである。しかし原理は同じだ。sendmailデーモンはSMTPプロトコルでメールを受け取り、それをローカルホスト上で配信するのである。

```
 1: [test ]$ telnet localhost 25
 2: Trying 127.0.0.1...
 3: Connected to localhost.takobeya.com.
 4: Escape character is '^]'.
 5: 220 mail.takobeya.com ESMTP sendmail 8.9.3/3.7W-2.8compat; Fri, 8 Sep 2000
    17:28:05 +0900 (JST)
 6: HELO localhost
 7: 250 mail.takobeya.com Hello localhost.takobeya.com [127.0.0.1], pleased to
    meet you
 8: MAIL FROM: <test@takobeya.com>
 9: 250 <test@takobeya.com>... Sender ok
10: RCPT TO: <test@takobeya.com>
11: 250 <test@takobeya.com>... Recipient ok
12: DATA
13: 354 Enter mail, end with "." on a line by itself
14: To: test@takobeya.com
15: From: test@takobeya.com
16: Subject: SMTP test
17:
18: This is test mail message.
19: .
20: 250 RAA42627 Message accepted for delivery
21: QUIT
22: 221 mail.takobeya.com closing connection
23: Connection closed by foreign host.
24: [test ]$ more /var/spool/mail/test
25: From test@takobeya.com Fri Sep  8 17:30:13 2000
26: Received: from localhost (localhost.takobeya.com [127.0.0.1])
27:         by mail.takobeya.com (8.9.3/3.7W-2.8compat) with SMTP id RAA42627
28:         for <test@takobeya.com>; Fri, 8 Sep 2000 17:29:00 +0900 (JST)
29: Date: Fri, 8 Sep 2000 17:29:00 +0900 (JST)
30: Message-Id: <200009080829.RAA42627@mail.takobeya.com>
31: To: test@takobeya.com
32: From: test@takobeya.com
33: Subject: SMTP test
34:
35: This is test mail message.
36:
37: [test ]$
```

画面3　デーモンとして動作しているsendmailと対話

Try & [trái] Try



なんだかんだ言ってみても Unicode化は進んでる？

文：政久忠由
Text : Tadayoshi Masahisa

世の中、対決という図式が氾濫している。なんでもかんでも、とりあえず対決である。

僕は、いつもおいおいと思いながら眺めていることが多い。ほとんど、どうしようもなく、しょうもないものばかりなんだけど、実際その図式にするとシンプルでわかりやすく、見ていて面白いという面もあり、悪いこととは思わない。どんなに意味のない対決、たとえば、こんにゃくとLinuxであっても、そのディベートのやり方、論点の置き方次第では、面白くなるだろうし、当事者は勝ち負け、見ている側、第三者は、それに関係なく、やじ馬根性で駆け引きを楽しめると思う。結構、視野が広がるような意見もあったりするしね。

コンピュータ業界もたぶんに漏れず、対決の嵐なわけで、Windows、Linux、MacなどのOS、サーバソフトウェア、CPUやメモリ、バスアーキテクチャなど、さまざまな事柄がいろいろな角度から議論、討議、そして罵倒されている。企業間のものもあれば、ネット上のユーザー間の宗教じみたものまで、いろいろなんだけど、嗜好がたぶんに影響するものや、規格に関する問題は、長い間くすぶり続ける傾向にあるようだ。

最近では、僕は、書き込み可能なDVD規格でいったいどれが本命になるのか、わからないでいる。これにはさまざまな意見があるが、歴史をさかのぼると、技術的によいものが残るわけでもない。ではサポートメーカーの数かというと、それだけでもない。結局、利用するユーザー数が多いものが、最も生き残る可能性があるわけだけど、この分野は立ち上がったばかりで、市場もまだ小さく、人気を予想するためのファクターも少ない。そういうわけで、優劣がつくまでには、まだまだ時間がかかりそうだ。ついこの間、ビデオデッキが壊れたので、書き込み可能なDVDも視野に入れて考えてみたのだけれど、どう考えても、今、手を出すという強い意志は働かなかった。DVDが10時間程度録画できるメディアだとしたら、違ったとは思うが、つまるところ、ごく普通のビデオデッキを選択してしまった（本当はHi8で残しているものもあるので、8mm付きのダブルデッキにしようと思っていたのだが、それは売れ筋ではないこともあって、あまりにも値段が高かった。ここでも市場原理を痛感させられたわけだ。だから泣く泣くごく普通のVHSを選択してしまった。うむ、Hi8はどうしたものだろう）。たぶん僕が書き込み可能なDVDを購入するのは、なるようになってからだと思う。でもまぁ、気分次第なんだけど。

さてともかく、ほとんどの対決は、利用者の好みで片付けられるわけなんだけど、いくつかの問題は、そうとばかりは言っていられない。そのひとつが文字コードの問題だ。これも長い間、喧喧諤諤している問題である。でも僕自身は、1ユーザーとして、現状にそれほど問題意識はない。僕が有するデータは、ほとんどシフトJISだが、他の人とやり取りする場合もそのままで問題なくできているし、外字を使用した場面も記憶の片隅に残っている程度だ。たま

に変な文字コードのメールが送られてくるものの、メールソフトがカバーしてくれるので、本当に困ったという記憶がなく、問題意識は薄らいでしまっている。

僕がふと、文字コードのことを思い起こすきっかけになったのは、日本語ドメイン名が登録できるようになった、というニュースだ。元来DNSでは、基本的にASCII文字（7ビット）しか利用しない設計で、多国語化というか、UTF-8を利用できるようにしようと本格的にRFCの策定や実装が始まったのは、2000年に入ってからだったと思う。だから日本語ドメイン名って本当に大丈夫？というのが率直な感想で、システム的に少々心配してしまう部分もあったりするわけだ。まぁDNS自体に関しては、また別の機会にするとして、ここでは文字コードについて考えてみることにしたい。

文字コードって？

文字コード自体やそれから派生する問題には、実にさまざまなものがあるが、僕がそのすべてを把握しているわけではないし、またそれらを調べて、ここで列挙して論じる気も毛頭ない。すなわち、ここで少しだけ触れる事柄は、僕の偏っているかもしれない知識の上に成り立っていることをあらかじめ断っておく。

一般に文字コードの大きな問題としては、文字セットとエンコード方法の2つがある。

まず文字セットは、コードを割り当てるために選択した文字の集合で、日本だと、歴史上使用され、残されている文字の中から、優先順位をつけて抽出するわけだ。知ってのとおりコードの数には制限があるため、コードを割り当てられる文字には取捨選択が生じる。つまり、切り捨てられる文字が必要な人にとっては、これが切実な問題となる。現状、一般の社会生活において使用する漢字の目安の常用漢字さえ含まれていれば、ある程度何とかなるわけだが、既存の人名や地名には尊厳が絡み、常用漢字以内にしてくれと強く指導できるものではないし、同様にちょっと古い文献をそのまま電子化しようとする場合にも、困るわけだ。前者の人名に関しては、画数占いの影響もあったりして本当に厄介だ。後者に関しては、僕は特に古典が嫌いなので、まったく必要性を感じないし、その手のものは、原文をスキャナで取り込んで、現代国語でいくつかの解釈を付加すればいいだろ、とそのことを聞くたびに思ってしまう（少なくとも中学、高校でやることじゃなくて、生涯教育でやることだよなぁ、古典は）。

次にエンコードは、文字セットを実際に利用できるよう実装する手法だ。これもいろんな事情があって、一般に利用されているものに、JIS、シフトJIS、EUCなどがある。完全に閉じた世界であれば、エンコード方法は特別問題にならないが、データを交換するとなるとこれが問題になる。幸い実装する文字セットに大きな違いはない。

特にインターネットでのデータ交換で問題となるので、国際規格として決まったのがISO 2022、で、この規格の日本語パートの部分iso-2022-jpには、JISが採用されている。だからメールやWebページも日本語が含まれる場合は、基本的にJISエンコードを使用するわけだ。でもデータのやり取りを行う両者にコンセンサスがあれば、何だって構わない。

見えないところで利用が進むUnicode

しかし、上記のエンコードの場合、8ビットのASCIIコードをベースに拡張していて、複数の国や地域の言語を扱うには、その切り替えの手段が別途必要となる。その煩雑さを回避する目的もあり、業界の一部から内部処理に適した複数言語を包含するコードとして生まれたUnicodeがここに来て方々で利用され始めている。ただし、文字コードをめぐる問題がUnicodeですべて解決されるわけではない。各国、各団体、各企業、そして各ユーザーをすべて満足させる規格は、残念ながら実現不可能だ。それなりに実用的なレベルのものが使用される。これが精一杯であろう。

当初のUnicodeは、16ビット固定長で世界中の主要な言語の文字のサブセットを含む点で、国際化ソフトウェアの内部処理には適していた。しかし、日本、中国、韓国の漢字をはじめ、文字を見た目でまとめてしまったことに違和感を持っている人も多く、また8ビット長で十分な英語圏では、不要にデータサイズが大きくなるだけなので、積極的に使用されるような動きは少なかった。

現在、Unicodeをベースにした文字セットは、国際標準ISO 10646に取り入れられ、そのUCS-2と呼ばれる16ビット固定長の部分は、今のところUnicodeと同じと考えてよい。ただし、ISO 10646もUnicodeもコード化の空間を32ビットに拡大するオプション（ISO 10646のUCS-4やUnicodeのVer.3の拡張部分が同じであるかどうかは調べていないけど）を用意しているので、当初のUnicode＝16ビット固定長というのは今では正確ではない。

今までUnicodeは、固定長であることを活かし、WindowsやJava言語の内部コードには採用されてきた

が、アプリケーションレベルでは、ほとんど実装は進んでいなかった。しかし、世界的にインターネットが普及し、拡大し続けていることもあって、インターネット関連のサーバ／クライアントアプリケーションへの実装が加速している。理由は簡単、ISO 2022をベースに各国の文字エンコードを複数実装するより、Unicodeをベースに1つのエンコード方法を実装したほうが、簡単に国際化をはたせるからだ。また、アプリケーションなどのレベルで容易にUnicodeを利用しやすくするための、特別なエンコード方法が規定されたことも大きな要因だ。

UTFというエンコード方法

従来Unicodeの処理を実装するには、専用の16ビットのインターフェイスを用意するなどの必要があり、またASCIIコードベースだったもので、単純にUnicodeを利用すると、データサイズは2倍になる。これらの問題を回避して、Unicodeの利用促進、そのメリットを活かせるようにしたのが、UTFというエンコード方法だ。

UTFには、基準バイトによって、UTF-8、UTF-16、UTF-32の3種類がある。Unicodeの32ビットコードスペースが有効に活用される状況になれば、UTF-32が有望視されるのだが、それはかなり先の話だ。現在は、インターネットにおいて、UTF-8が積極的に活用されている。UTF-8では、UnicodeをMBCSのようにシリアライズエンコードすることで、現状のアプリケーションが有する8ビット文字用のインターフェイスをほとんど変更することなくUnicode文字列を使用できるようにする。さらに、ASCIIコードを最優先に扱えるようにし、エンコードサイズも抑えている。UTFの詳細に関しては、http://www.unicode.org/やRFC 2279などを参照するとよいだろう。

文字エンコードの実際

実際にデータがどのようになるか見てみよう。例として、www.日本語.comが各種エンコードでどのようになるかを示す。

●シフトJIS

```
7777 772E 93FA 967B 8CEA 2E63 6F6D
```

0x77が“w”、0x2Eが“.”、0x93FAが“日”、というようになる。日本語の部分は、リードバイトとそれに続くトレールバイトの2バイトで構成される。

●EUC

```
7777 772E C6FC CBDC B8EC 2E63 6F6D
```

シフトJISと似ているが、リードバイトとそれに続くトレールバイトに使用する範囲が異なり、0x00～0x7FのASCII文字セットとの重なりを回避している。

●JIS

```
7777 772E 1B24 4246 7C4B 5C38 6C1B 2842 2E63 6F6D
```

JISでは、日本語文字コードの始まりと終わりに0x1B2442と0x1B2842のシーケンスを挿入して表している。

●Unicode（リトルエンディアン）

```
FFFE 7700 7700 7700 2E00 E565 2C67 9E8A 2E00 6300
6F00 6D00
```

先頭のFFFEはリトルエンディアンUnicodeを表すシグネチャで、各文字は、16ビット固定長ですべて表されている。文字コードはバイトオーダーの影響で入れ替わっている。“w”の文字コードは0x0077など。

●Unicode（ビッグエンディアン）

```
FEFF 0077 0077 0077 002E 65E5 672C 8A9E 002E 0063
006F 006D
```

先頭のFEFFはビッグエンディアンを表すシグネチャ。各文字コードは、実際のコードと同じ見え方になっている。

●UTF-8

```
EFBB BF77 7777 2EE6 97A5 E69C ACE8 AA9E 2E63 6F6D
```

先頭の0xEFBBBFはUTF-8エンコードを表すシグネチャ。Unicodeに対し、UTF-8では、ASCII文字範囲は、0x00を取り除いた状態、漢字が収められた範囲は、その文字コード16ビットを4／6／6ビットに分割した状態になる。実際に変換には、次のような手法が取られる（以下はUCS-2の部分を抜粋、実際にはUCS-4にも対応しているようだ）。

```
    UCS-2 (Unicode)       UTF-8 octet sequence
1:2 0000 0000-0000 007F   0xxxxxxx
2:2 0000 0080-0000 07FF   110XXXXx 10xxxxxx
3:2 0000 0800-0000 FFFF   1110XXXX 10Xxxxxx 10xxxxxx
#4:4 0001 0000-001F FFFF   11110zXX 10XXxxxx 10xxxxxx
10xxxxxx
```

たとえば、“日”のUnicodeは0x65E5である。これを2進表記すると次のようになり、

```
0110 010111 100101
```

これに上記のシーケンスに従ってプレフィックスビットを付加すると以下のようになる。

```
1110-0110
10-010011
10-100101
```

これは0xE693A5である。

UTF-8では、ASCII文字範囲をASCIIコード同様8ビットで表し、それ以外の部分は多少表現サイズが大きくなるものの、オクテット単位で見たときに0x80以上に必ずなるように区別することでロールオーバーを回避し、従来のASCII文字のみのサポート時と比べて、性能や効率性の低下を最低限に抑え、他国語も扱えるという機能性の両立を図っているのだ。またオクテット単位で扱うため、通常のUnicodeのようなバイトオーダーの問題もなく、EUCなど、今までのMBCS同様の扱いが可能だ。そのため、インターネットアプリケーションレベルでの実装が加速しているというわけだ。

バイトオーダーの影響

いい機会なので、少しバイトオーダーについて説明しておこう。一般にリトルエンディアンとビックエンディアンの2種類が取り出さされているが、実は、一口にエンディアンと言っても、これには実に複雑な事情が含まれている。このうち大きな要因は、ビットオーダーとバイトオーダーという2つ違いだ。

■ビットオーダー

まず、ビットオーダーというのはビット列の扱いの違いだ。コンピュータにおいては、処理の効率化のために、個々のビット単位で操作を行ったりはせず、あるまとまりでもって操作を行う。CPUではレジスタ長がその単位だ。ビットオーダーというのは、そのビットの並びの上位、下位をどのように位置付けるかということになる。これを人に見えるかたちで表現すると、たとえば32ビットデータでは、右に31ビット目から配置するか0ビット目から配置するかというものになるが、あくまで内部的な問題で回路設計に携わる人々にしか影響がない。ハードディスクケーブルなどの1番ピンが左右どちら側に配置されるか程度の問題でしかない。ただ伝送経路がシリアルの場合は、ビットオーダーが大いに問題となるわけだが、このことを考慮しなければならない人は、本当に一部の開発者だけだ。

■バイトオーダー

次にバイトオーダー。これは、それぞれのコンピュータにおけるデータのやり取りを行う単位が影響した問題だ。

コンピュータの中央演算回路は、8、16、32、64、128と演算で一度に扱えるビット列を長くしてきた（僕たちが普通に使っているのは32ビットだけど）。しかし、少なくとも多くのアーキテクチャでは、I/Oの単位は変えなかったのだ。それがバイトである。バイトというのは、ビットのまとまりという意味で、バイトが8ビットであるというのは、intが32ビットであると思い込んでいる人がいるのと同じ間違いということになる。本来バイトはアーキテクチャによってそのビット数は異なるのである。アーキテクチャによっては、ワードと呼ぶこともあるが、このまとまりは、8であったり、16であったり、32であったり、もっと違った数であったりすることもある。たまたま僕たちのまわりのコンピュータのアーキテクチャでは、バイト＝8ビットであっただけなのだ。そしてそれはCPUの基本レジスタ長が拡張されても、バイト単位は変わらずにきたのである。このまとまりの単位は、コンピュータの記憶域であるメモリのアドレスに対応する格納領域のビット長でもある。x86をはじめ、僕たちの身近なコンピュータでは、32ビットマシンであっても、1つのアドレス、たとえば0x00000000に格納されるデータのサイズは8ビットで、32ビットのデータを格納するには、通常4つの連続するアドレスが必要となる（僕は昔、16ビットCPUが出てきたとき、これらのことを納得するまでにかなり時間を要したことを覚えている）。

通常それぞれのアーキテクチャにおいて、バイトであるまとまりの単位を決定する場合、どのようなデータを扱うか、そしてそれに最適な単位は何か、という判定基準が適用される。PCの場合は、文字列操作が大きなウェイトを占めるので、文字の単位として使用される8ビット単位がバイトとして、記憶域の単位サイズにもなっているのだ。もし8ビットデータが希にしか使用されないシステムであれば、16ビットや32ビットを記憶域の単位サイズとしてもよかったのだけれど、そうはならなかった、現状なっていないというわけだ。なお、すでに8ビット＝バイトというのは定着してしまっているので、厳密には違ったとしても、今さらそれは違うと言っても、逆に変なやつと思われるので注意したほうがいいだろう。

すなわちバイトオーダーは、演算の処理単位であるレジスタ長と、そのデータをやり取りする単位、格納の単位が異なることによって生じる格納順の問題なのである。一般にPCでは、8ビット、16ビット、32ビットのデータが扱われる。8ビットデータは何の問題もなくやり取りできるが、16ビットと32ビットは、これを8ビットに区切り、格納していくことになる。ここで問題となるのが、その順番というわけだ。結論からいうと、リトルエンディアンと呼ばれる手法では、アドレスの下位から順にデータの下位バイトを収めていく。逆にビックエンディアンでは、アドレスの下位から順にデータの上位バイトを収めていく。たとえば、0x11223344という32ビットデータと0x5566の16ビットデータは、次のように格納される。

アドレス

・リトルエンディアン

```
0x0000  バイト1  0x44  0x66
0x0001  バイト2  0x33  0x55
0x0002  バイト3  0x22
0x0003  バイト4 0x11
```

・ビッグエンディアン

```
0x0000  バイト1  0x11  0x55
0x0001  バイト2  0x22  0x66
0x0002  バイト3  0x33
0x0003  バイト4  0x44
```

このようにアドレスの下位を上にして、上位を下に表記すると、リトルエンディアンでは、データは逆に見える。しかしビックエンディアンでは、ユーザーが目にするデータの並び順に表現される。回路設計的には、リトルエンディアンの手法のほうが素直な表現なのだが、ビッグエンディアンは、これらのデータを直接、目で見ることの多い開発ユーザーのことを考慮した結果、生まれた手法である。最終的な整形処理後の結果しか見ないエンドユーザーにとっては、これらの手法はまったく影響がなく、どうでもよいことなのだが、実際には、ファイルやネットワーク上でのデータのやり取りにもバイトオーダーは影響している。メモリに対するI/O処理のバイトオーダーが、ファイルやネットワークにも同様に影響するかどうかは、実装次第なのだが、通常同じと考えてよいだろう。

特にネットワークのやり取りでは、別のまとまりの単位が利用されている。それはオクテットという単位だ。オクテットは8を表し、8ビット単位を表現している。先ほど述べたように必ずしもバイト＝8ビットではないため、このように呼んでいるのだ。もちろんそのビットオーダーの問題もあるが、これは下位ビットからやり取りすることが根付いているのでもはや誰も問題にしない。バイト＝オクテットが適用できる場合は、バイトオーダーも先ほどと同じだが、それが異なる場合は、33441122という順番でやり取りされることもある。そこでネットワークでは、データの送信に先立ってバイトオーダー情報を付加するようになっている。

UTF-8の利用促進の影響

現実問題として、インターネットアプリケーション、つまりクライアント／サーバ形式で処理を要求し、そのサービスを受ける用途では、UTF-8がスタンダードになることは間違いない。実装も簡単で、最低限度の多国語化がそのエンコードのサポートだけでできるのだから当然だ。

しかし、だからといって、ユーザーアプリケーションでUTF-8が使われるようになるかどうかはわからない。多国語サポートは、世界中からの要求を受けるサーバ側には必須であるが、多くの一般ユーザーのローカル部分では、その機能は不要だからだ。近々に内部コードにUnicodeをサポートするとは思えないが、Linuxカーネルもglibcも、nlsのサポートでは、Unicode／UTF-8を基準に実装していることだし、今後の流れとして、Unicodeというのは自然のような気もする。個人的にUnicodeが嫌いじゃないからそう思うのかもしれないけど。

Athlon話のそれから

さて、前回Athlonの3D Now!を利用したメモリ、ページコピー話の続きを少しだけしておくと、やはりそれほど速くない。CPUの仕様書に速くなる可能性があるとは、書いてあるが、実際の結果は面白いものではなかった。一応、手元の結果は、4Kバイト程度までは整数演算回路のほうが効率的で、8Kバイト程度から逆転するのだが、1次キャッシュ、2次キャッシュのサイズやアルゴリズム、そしてメインメモリ性能の制限があるため、128Kバイト以降は頭打ちで、それ以降差が大きく広がることもなく、確かに速いけど、積極的に利用するほどでもないし、頻繁に利用するページサイズの4Kバイトでは効果が薄いので、なんだかなぁといった感想でしかない。

なお、Athlonの3D Now!はMMXに拡張命令を追加し

た機能で、レジスタ長は64ビットであった。で、IntelのMMXとSSEは、それぞれ64ビットと128ビット長のレジスタを使用できるのだが、整数データをまとめてセットして移動する命令はない。この命令が実装されるのは、SSE2からでこれはPentium4以降の搭載となる。すでにPentium4は発売されているようなので、その結果もそのうちわかると思う。

興味を抱いた人は、ソースコードを参照したと思うが、Linuxのそれらの実装は、GCCのインラインアセンブラで記述されているので、少々面食らった人もいると思う。

そこでその部分を少し説明しておくとしよう。まずは簡単な例としてCPUIDを取得するプログラムを見てみる。

```
inline void cpuid(int op, int *eax, int *ebx, int
*ecx, int *edx)
{ __asm__ ("cpuid"
: "=a" (*eax), "=b" (*ebx), "=c" (*ecx), "=d" (*edx)
: "a"(op));}
```

これは実際にLinuxカーネルの初期化時にCPUをチェックするために使用されているコードを抜き出したものだ。__asm__の中がアセンブラコードなわけだが、GCCでは、:で区切ったいくつかのパートを利用して実際のコードを埋め込む手間を省くことができる。上記では、命令、出力設定、入力設定の記述パートなる。つまり、最初のcpuidはCPUに対して情報を要求する命令で、次の"=a"(＊eax)は、レジスタEAXの結果データを引数で指定したeax変数に代入するという指定だ。最後の"a"(op)は、引数opのデータを命令の実行に先立ってレジスタEAXにセットするという指定になる。これがどのように展開されるかは、gcc -S ファイル名でチェックしてみるとよいだろう。

ちなみに、CPUIDのオペランドの指定は、IntelのPentiumIIIまでは、0x0から3、Athlonでは、0x0と0x1でx86互換としての情報、0x80000000から0x800000005でAthlonの拡張情報を取得できる。Pentium4では、0x80000000からのオペランドでいくつかの情報を取得できるように拡張されているハズだ。取得するには先ほどの関数を利用して次のようにするだけなので調べてみるとよいだろう。その際、各CPUの仕様書を入手してどのオペランドでどの情報がわかるのかを理解しておく必要があるけども。

```
int v[4];
cpuid(0x0, &v[0], &v[1], &v[2], &v[3]);
```

これらを踏まえて、各memcpyの処理内容をみると、多少は見通しもきくと思う。

mmx_memcpyでは、1:といった表記は、命令セクションのラベル、最後の“ memory ”というのは、メモリ内容の操作を行うので展開する際、それを考慮するようにという指示をGCCに明示している。まぁ実際には、__volatile__を指定する、しないのほうが、展開アセンブラコードに与える影響が大きいので、こちらの違いを見たほうが面白いと思う。

複数コンパイラの同居

新しいCPUを効果的に利用しようとすると、コンパイラもそれに対応したものが必要になるわけだけど、安定版GCCのリリースは、Linuxカーネル同様、遅い。両者ともシステムの土台となる部分のコンポーネントのため、そう簡単に安定版としてリリースというわけにもいかないという事情もあるのだが、開発版は随時公開されているので、使用しようと思えばできる環境にある。

Linuxカーネル2.4のテスト版は、すでにほとんどバグのない状態なので、実際に利用している人も多いと思う。でもGCCの開発版は、今のところ怪しい部分が多く残っている。前回述べたように、Linuxカーネルを開発版GCCでコンパイルするとさまざまな不具合が生じる状態であった。だからこういったテストを行う場合、コンパイラは、従来の安定版を含め、複数同居させておくことが重要となる。

GCCの場合、同居させるのは簡単だ。コンパイラの本体は、/usr/lib/gcc-lib/に、アーキテクチャ名、さらにバージョンというディレクトリ構成でそれぞれ格納されるようになっている。/usr/bin/のgccなどはこれらコンパイラ本体を起動するフロントエンドとして機能する。つまり複数のバージョンのコンパイラを同居させるには、gccのインストール先を別々にするか、同じディレクトリでも名前を変えておくだけで大丈夫だ。たとえば、gcc-2.97、gcc-.2.95.2、gcc-2.91といった具合に、コンパイラをインストールするごとにきちんと名前をつけておけばいい。makeでは環境変数CCを参照することがほとんどなので、CC=gcc-2.91といった感じに登録しておくとよいだろう。

なお、カーネルの場合は、ソースルートのMakefileに次のように設定されているので、このgccを適当な名前に変更するだけだ。ちなみにHOSTCCは、コンフィギュレーション用プログラムの設定なので注意したい。

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9回
【ウィンドウマネージャ】（うぃんどう・まねーじゃ）
【twm】（てぃー・だぶりゅ・えむ）
【Enlightenment】（えんらいとめんと）
【Sawfish】（そーふぃっしゅ）
【デスクトップ環境】（ですくとっぷ・かんきょう）
【ディスクトップ環境】（でぃすくとっぷ・かんきょう）
【KDE】（けー・でぃー・いー）
【GNOME】（ぐのーむ）

ウィンドウマネージャ
【うぃんどう・まねーじゃ】

単体だと見られたものでないウィンドウ環境“X”で、グラフィカルインターフェイス部分をまあまあ見られるようにするべく皮をかぶせるもの。はやい話が、そのままではさえない女性の顔に、化粧品、ブランドものを駆使して見られるようにするのと同じである。ユーザーとしては、なにも使わなくてもはじめから美しい環境が欲しいものだが、経済的・身体的・容貌的に折り合いがつかないと「しかたないからここらへんで手を打っとくか」と妥協して身を固めがちである。しかし、街に連れ出すときには男も見栄を張りたい。そこで化粧品やブランド品を買い与えて着飾らせようとする。彼女のほうは「この人、わたしのことをこんなに愛してくれてるんだわ」と勘違いしてくれるので、なおさらしめたものだ。3年もするとやがて男のほうがムダな努力だったことに気づき、コンソールに落ちて作業するようになるか、Linux自体を捨ててMac OSに走るのも、やはり女性との関係に似ている。こんなことを書いていると編者の身が危険にさらされるような気がしてきた。

筆者環境

twm
【てぃー・だぶりゅ・えむ】

シンプルなウィンドウマネージャ。女性にたとえるなら、田舎から出てきたてで三つ編みおさげにめがねの女子大生といったところである。「キミもウィンドウ枠を3Dにしてみたらどうだい？」などと言うと、うつむいて「わたし、そんな……」と恥ずかしがったりするところがマニアにはたまらないらしく、いまだにユーザーが多い。見た目素朴な少女を調教して美しくはばたかせるのは『源氏物語』をひもとかずとも、男性Linuxユーザーのロマン。あなたも都や県の条例に違反しない範囲でtwmを楽しもう。

数年前の筆者環境

Enlightenment
【えんらいとめんと】

まさに渋谷系女子高生といえるウィンドウマネージャ。ウィンドウはもちろん、日焼けサロンに通うなどして地肌までケバケバしく飾り立てるさまは、まさに世紀末の観あり。ちょっとメモリが足りないくらいでも平気で街なかにへたりこみ、「ちょっとぉ〜。超ムカツク〜。あと128メガぐらいよこせって感じぃ〜？」。enlightenment＝“啓蒙”という名前を持つわりには、知性が感じられないのである。それでいて多数のオヤジユーザーの人気を集めており、援助交際の見返りにハデなテーマをもらっては、また豪奢に着飾るのだから悪循環はとまらない。誰かが断固とした態度で彼女らに思い知らせてやるべきではないか。このあいだ、電車の中で「みてみて。あのオヤジぃ。ユニクロのシャツなんか着てる

よ。カッコわるぅ〜」と陰口をたたかれたうらみ、編者は忘れない。ユニクロのどこが悪いんだ。全品XLまであるのはユニクロだけなんだ。だいたい、三十路にもなっていない青年をつかまえてオヤジっていうな。

Sawfish

【そーふぃっしゅ】

“ノコギリエイ”という名をもつウィンドウマネージャ。エラの張った女性？　そろそろ、この比喩も苦しくなってきた。いずれにせよ、内蔵スクリプト言語として搭載されているのがLispであるところから、“カッコばかり”のいけすかないヤツであることは疑いの余地がない。なぜLinuxユーザーはLispばかり使うのか。たまにはFORTRAN内蔵のエディタくらい作ってはどうか。

デスクトップ環境

【ですくとっぷ・かんきょう】

Linuxユーザーのデスクトップ環境を調査した統計（雷通調べ）によると、

コタツトップ（死語？）

1位　乱雑で汚い（87%）
2位　パソコンに占領されモノが置けない（21%）
3位　そもそも机がなく、年中こたつを使用する（18%）
4位　GNOME、またはKDEを使用（2%）
5位　アクティブデスクトップを使用（0.1%）

であった。

ディスクトップ環境

【でぃすくとっぷ・かんきょう】

おそらく、フロッピーディスクベースで動作するGUI環境のこと。多くの人々がウワサしているが（google調べで約150件もの関連サイトが存在する）、実物を見たものは誰もいない。編者のもとにも、「ディスクトップパソコンにインストールして使う」、「ハードデスクの上に置かないと動かないらしい」など、断片的な情報しか届いていない。今後いっそうの研究が必要だろう。

KDE

【けー・でぃー・いー】

“Kのデスクトップ環境”。バドーの犯罪をくい止めるためつくられたロボット刑事や、一族全員が医療フェチという異常な家系に生まれついたスーパードクターなどとかなり関係が深いと思われる。さらに使用ライブラリは「キューティィ」と命名されており、空中元素固定装置つきヒロインが活躍するハレンチマンガへのオマージュであることはもはや疑う余地がない。ここから推察するに、開発チームは重度のマンガ・特撮オタクだ。もしかしたらアラーム設定時間になると「でじこだにょ」というセリフとともに、マスコットが現れたりするギミックくらい隠されているかもしれない。ユーザーになるなら、エラー発生時に「認めたくないものだな、自分自身の若さゆえの過ちというものを」などというメッセージが出てきても、ヒいたりせず「坊やだからさ」くらいつぶやいて返す余裕が必要だ。使ったことがないのでよくわからないが。

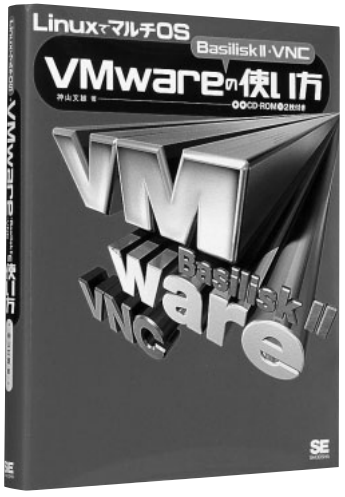
GNOME

【ぐのーむ】

深夜のテレショッピングにて？

GNUによるデスクトップ環境。家に小さい子供を飼っていると、クレヨンで机に落書きをしてしまったりすることが多く困りものである。そこで登場するのが、どこのご家庭にもあるGNU。汚れたところにGNUをひたひたとかけ、雑巾で軽〜く拭き取ると、あら不思議。クレヨンで塗ったあとが落ちている。なお、このウラワザを使う際には、子供の見ていないところで。汚しても落ちるとわかるとガキが図に乗るので教育上よくないのである……。GNUでのデスクトップ環境整備というのは、たぶんこんなところだと思う。えっ？　ネットワーク対応とか国際化の要素が説明されていない？　そういう場合は、書店店頭にてLinux magazine 12月号を注文のうえ読破すること。最近の読者は要求ばかり多くて困る。編者にばかり仕事させないで、読者もこのページ存続のために原稿を書いて送ってくるぐらいの協力をなぜ惜しむのか。かのケネディは言った。国が諸君らに何をしてくれるかではなく、諸君らが国に何をできるかを考えろ、と。だいたいGNOMEを使ったことがない編者にそんなこと期待するのが間違っている。そもそもウチにはLinux boxなんて（編集部注：以下、誌面の性質上問題があるので削除）。

Books



LinuxでマルチOS VMware・Basilisk II・VNCの使い方

神山文雄 著
翔泳社
B5変形判／280ページ／CD-ROM2枚付き　　本体価格 2800円

本書では、LinuxやWindows NT/2000で仮想的なPC環境を実現するVMware、Linuxに68k Macintosh環境を提供するBasilisk IIという2つのエミュレータのほか、あるマシンから別のマシンへネットワーク経由でGUI画面を表示させるVNCという3つのツールが紹介されている。

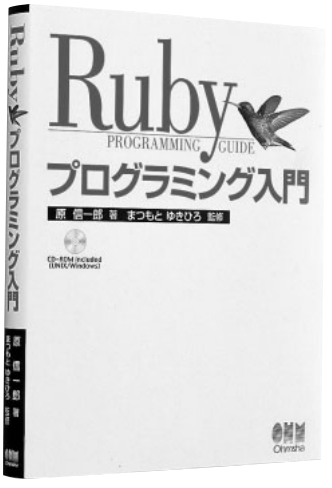
この中で本書のメインターゲットはVMwareである。VMwareをホストOSのLinuxへ導入し、VMware上の仮想PC環境にゲストOSのWindows 98をLinuxへインストールする方法、さらにはVMwareの拡張ツールVMware Toolsの設定方法や、Linux上のSambaを使って、LinuxとWindows 98の間でファイルを共有するという一歩踏み込んだ使い方までが詳細に解説されている。付録CD-ROMにはVMware（別途ライセンス取得が必要）、Basilisk、VNCのほかに、Turbo Linux 日本語版6.0（FTP版）が収録されており、読者は本書の内容を体験できるようになっている。1台のマシンで手軽に複数のOSを使いたい、という読者にお勧めの1冊だ。

Rubyプログラミング入門

原 信一郎 著／まつもとゆきひろ 監修
オーム社
B5変形判／376ページ／CD-ROM1枚付き　　本体価格 2800円

本誌連載「Rubyで行こう」でもすっかりおなじみ、オブジェクト指向スクリプト言語Rubyの入門書が登場した。すでにご存じの読者も多いと思うが、Rubyは本書の監修にもあたられている、まつもとゆきひろ氏が開発した国産のプログラミング言語で、ピュアなオブジェクト指向による簡潔なプログラミング手法が高く評価されている。

オブジェクト指向というと、その概念や実装の理論ばかりが頭に浮かんで「難しいもの」と捉えられがちだが、もともとはコンピュータのプログラムをより人間が理解しやすい形で簡潔に表現することを目的としていて、開発効率の向上や言語習得の簡易化を目指しているものだ。現在では多くのプログラミング言語で、その考え方が取り入れられている。ある程度の素養がある方だけでなく、これからはじめようという初級者にとっても、Rubyはプログラミングを学ぶ優れた教材となるはずだ。

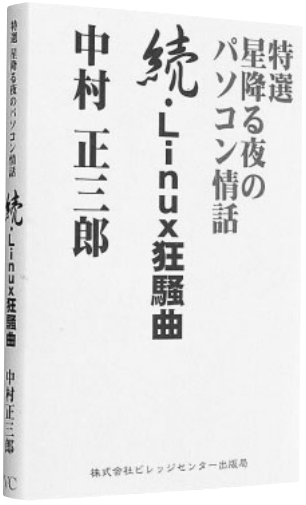


使えるエディタを作りながら学ぶ Tcl/Tk実践的プログラミング入門

若谷 純 著
セレンディップ／小学館
B5変形判／320ページ／CD-ROM1枚付き　　本体価格 3200円

Tcl/TkはLinuxではおなじみの開発環境だ。Tclがコマンドをベースとしたインタープリタ言語（Tool Command Languageの略）で、Tkがウィジェットと呼ばれるウィンドウやボタンなどのGUI部品のセットである。簡潔なプログラム記述によるGUIアプリケーションの構築を特徴としており、Linuxカーネルのコンパイル時に使われる、「make xconfig」で表示されるグラフィカルなインターフェイスもTcl/Tkで記述されている。

サブタイトルにもあるように、本書ではテキストエディタを開発しながらTcl/Tkを使ったプログラミングを解説していく。ひと通り読み終われば、エディタが（たぶん）完成するので、初心者でもちょっとしたプログラマー気分を味わえる。Tcl/TkはLinux（UNIX系のOS）だけでなくWindowsとMac OSもサポートしており、それぞれのOSに対応した解説もされている。CD-ROMには、各OSバージョンのTcl/Tkとライブラリ、サンプルコードを収録。



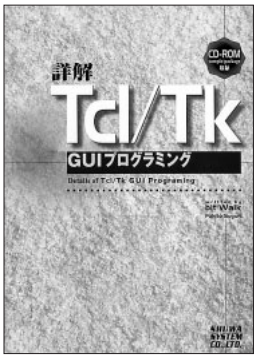
特選 星降る夜のパソコン情話 続・Linux狂騒曲

中村 正三郎 著
ビレッジセンター出版局
四六判／272ページ **本体価格 1200円**

中村正三郎氏をご存じの方は、反マイクロソフト論客の急先鋒というイメージが強いかもしれないが、著作やWebページでの論説を追っていくと、コンピュータ業界の健全な発展を願う熱血漢であることをうかがい知ることができる。氏はまた、パソコンをターゲットとした国内マスメディア（弊社を含む）の在り方にも、常に苦言を呈されている。にもかかわらず業界内（弊社を含む）に「正三郎ファン」や「正三郎ウォッチャー」が意外に多いのは、それが「正論」であるがゆえか？　実際、身につまされる指摘も多い。

本書はLinux Japanに連載中の「Bravo！ Linux」をまとめた書籍の第2弾（「パソコン情話」シリーズとしては通算5冊目になる）。Linuxにとどまらず、コンピュータ業界をとりまく豊富な話題が満載のエッセイ集だ。ややシニカルで偽悪的な表現もあって好き嫌いが分かれるかもしれないが、それも「正三郎節」のうち。一読して損はないはず。

詳解　Tcl/Tk GUIプログラミング



須栗歩人 著

秀和システム

B5判／474ページ／CD-ROM1枚付き

本体価格 3200円

Red Hat Linux 7J オフィシャルマニュアル

Red Hat, Inc 著
レッドハット株式会社 訳

インプレス

B5変形判／800ページ／CD-ROM4枚付き

本体価格 5200円

Linuxシステム管理

Jochen Hein 著
吉川邦夫 訳

アスキー

B5判／560ページ

本体価格 5800円

Red Hat系対応 Linuxサーバ入門

清水正人 著

アスキー

A5判／208ページ

本体価格 1900円

読者の声

俺にもいわせろ！

突然ですが、今月から読者アンケートハガキがなくなりました。今後は、アンケート、プレゼント、ご意見はWebにて承ります。Linux magazineのページ（http://www.ascii.co.jp/linuxmag/）からのリンクをご活用ください。

12月号特集1へのお便り

「GNOME 1.2＆KDE 2.0」特集は役に立ちました。ふだんお世話になっていながら、GNOMEを使っていることをまったく意識していませんでした。というわけで、GNOMEの設定すら知りませんでした。どうも、Linuxはサーバ OSという先入観があったのですが、身近なデスクトップ分野にももうちょっと関心を持とうと思います。

（東京都　丹羽直樹さん）

KDE 2.0に興味津々。完全日本語対応を待つ日々です。メーリングリストなどから、開発者、翻訳者諸氏のご苦労がうかがえますが、利用者がますます増えるよう祈ります。

（石川県　中山篤哉さん）

GNOME＆KDEの特集、すごくよかったです。これで、バリバリ設定して、サクサクとX Window Systemを使えそうです。ところで、GNOMEってなんて読むのでしょう？

（兵庫県　矢倉 武さん）

GNOMEとKDEの登場以来、Xのデスクトップ環境は飛躍的な進歩を遂げています。Linuxディストリビューションでは、デフォルトがGNOME環境というものも多くなりました。先月号でも紹介したように、KDEも非常に高機能なデスクトップ環境ですので、まだ試していないかたはぜひお試しください。

ただし、これらの統合環境は、少々動作が重いので、Windowsを使うときと同等のマシンパワーがほしいところです。非力なマシンでは統合環境ではないウィンドウマネージャを使ったほうが快適でしょう。ウィンドウマネージャなどを柔軟に選べるのもXのよいところですね。

GNOMEは、GNU Network Object Model Environmentの頭文字をつないだもので、グノームと読みます。地底で宝を守っている妖精もGNOMEですが、こちらはノームです。

12月号特集2へのお便り

「IDE RAIDカードで作る　使えるパーソナルRAIDシステム」はとても参考になりました。ぜひやりたいです。お金があれば……。

（愛知県　國枝信吾さん）

2000年1月号をはじめて購入し、その月末にマシンを組んではや1年。これまで毎号欠かさず読み、いろいろ試したり、悩んだりで、あっという間の1年でした。その間、インターネットの接続や、メールもmewに一本化したり、思い切ってkatmaiを買い足してデュアルCPUにしたり。ソフトウェアでは、XFree86の4.0を悩みつつもインストールして、Quake III DEMOのLinux版が動いたのもうれしかったです。そして、今年のシメはやはりRAIDでしょう！　冬休みにこれを成功させてLinux 1年生を終えたいと思っています。

P.S.　Tux Racerは、タイム表示されるようになり、いいですね。また、新年号より役に立つ記事を期待しています！

（埼玉県　小川恭生さん）

担当はSCSI派だったのですが、IDE機器との価格差がどんどん広がっていくのを見て、IDE派に転向しつつあります。口の悪い編集部員なんか、「SCSIはレガシーデバイス」なんていいますし……。今の3倍くらいの収入があれば、すべてのマシンをオールSCSI化したいんですけど、それは分不相応な願いなのでしょう。

型落ちになったUltraDMA/66の拡張IDEカードを安く手に入れられれば、LinuxのソフトウェアRAIDも悪くない選択肢でしょう。先月の特集では紹介していませんが、Linux magazine Labでは、PromiseのUltra66という拡張カードでソフトウェアRAIDを試していました。最高のパフォーマンスとはいえませんが、それなりの結果を出していたそうです。

Linuxのマスコット、ペンギンのTuxくんが雪山を滑り降りる3Dゲーム「Tux Racer」は、Linux版のほか、Windows版やMacintosh版も配布されています。担当はWindowsマシンにもインストールして、ダウンヒルしています。ダウンロードは、http://www.tuxracer.com/からどうぞ。

行楽シーズンでしたので

編集部のみなさまお久しぶりです。先月号は、秋の行楽シーズンにつき、多忙のためお休みしました。山形の一大イベント“芋煮会”を知っていますか？　たまにはパソコンを離れて、毎週のようにアウトドアをエンジョイしました。おかげで、毎週月曜日は二日酔いの嵐でした。

話は変わって、また自作パソコンの“虫”が騒ぎだし、とうとうインターネット通販でたくさんのパーツを買い集めてしまいました。当然、女房（大蔵省）からは非難の嵐。家族サービスそっちのけで“芋煮会”や“自作”では当然（？）かもしれませんね。編集部のみなさんも気をつけましょう。(*_*)

（山形県　ホゲホゲ遠藤さん）

月刊アスキー編集部には秋になると「芋煮会やるべさ」と大騒ぎをする編集者がいます。芋煮会とは、大勢で河原にあつまり、大きな鍋を囲む収穫祭で、東北を中心によく行われる行事だそうです。地域によって味噌仕立てだったり、しょうゆ仕立てだったりと、作り方が違うそうですね。件の編集者を中心に、芋煮会を開いたことがありますが、秋空の元、温かい鍋を食しながら飲む酒は最高でした。

山形市で毎年開かれている「日本一の芋煮会フェスティバル」では、直径6メートルの鍋で、なんと3万食を配ったというからすごいものです。鍋をかき混ぜるのに、パワーショベル2台を使い、材料は水6トン、芋3トン、牛肉1.2トン、コンニャク3500枚、ネギ3500本、酒50升、しょうゆ700リットル、砂糖200kgというビッグイベントです。テレビでも紹介されていましたが、パワーショベルのアームは、潤滑油としてサラダオイルを使っているとのことです。

小型車はいいですよ

「初級Linuxer養成講座」で、竹田善太郎さんは“リッターカー”に乗っていることが書かれていましたが、車種は何でしょうか。私も元祖リッターカー“ダイハツ　シャレード1000cc”に乗っています。

（山梨県　田辺保志さん）

竹田さんにコメントをいただきました。

「厳密にいうと『リッターカー』の区分に入るかどうか怪しいですが、1300ccのスターレット（EP-82、平成4年式）です。バブル期に設計された車なので、1300ccで自然吸気のくせに100psも出力があるという、けったいなエンジンを積んでいるやつです。『自動車評論家』の先生方はぼろくそにけなしていた車ですが、丈夫で取り回しも楽で、よく走るいい車だと思います。シャレードもリッターカーの鑑のような、合理的な車だと思います。今の車を買うときは、『スターレット』、『マーチ』、『シャレード』のどれにするかで、悩みました。結局は、歩いて行ける所にディーラーがあるかどうかという、なんとも現実的な基準で今の車に決めました」。

担当もスターレットターボ（EP-82、平成元年式）に乗っていました。8年間乗りましたが故障もなく、よく走ってくれたものです。

ユーザーフレンドリー！

久しぶりにLinuxを入れてみた（Vine Linux 2.0）。最近のLinuxはSlackware全盛の時代に比べて簡単になってしまったと思った。

（香川県　片山浩二さん）

片山さんのおっしゃるとおり、当時に比べるとLinuxのインストールは簡単になりましたね。そのぶん、バリバリと使う時間が増えるのはよいことです。ぜひ活用してください。

次期主力デスクトップOSの選定

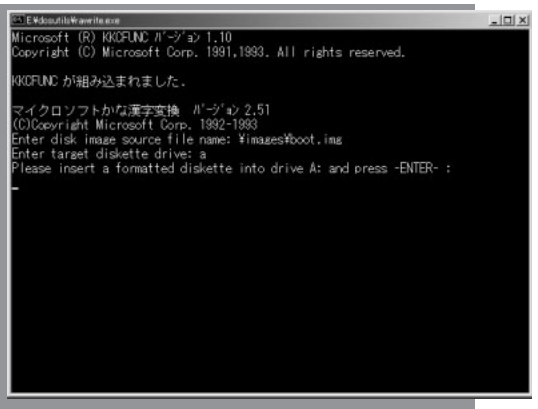
Linux magazineをはじめて買った。これは、貴誌をはじめて買ったというだけでなく、Linux関連の雑誌全体ではじめてということ。個人的にはOS/2が好きで、某M社のOSは安定性の点からGAME OPERATING SYSTEM＋αくらいの価値しか見いだしていない。もっとも、OS/2の状況はかなりブが悪いので、安定した次期OSとしてLinuxの世界をのぞいてみようか、というのが購入の動機。それにしても明るいですねぇ。Linuxワールドは。これなら次期OSとして期待できるかも！？

（北海道　高橋英一さん）

OS/2は安定したいいOSです。WindowsのcygwinのようにUNIXライクな環境を提供するEMXというフリーソフトウェアを使うと、Squid、bind、sendmailといったネットワークサーバはもちろん、XFree86も動作します。しかし、OS/2は、最近のハードウェアに対応できないなど、将来性が不安ですね。ファンとしては、もう少しがんばってほしいところです。

付録CD-ROMに収録した
Vine Linux 2.1のインストール

本誌付録CD-ROM収録のVine Linux 2.1はFTP（インテルアーキテクチャ）版です。非商用ソフトだけが含まれおり、製品版を販売しているレッドハット株式会社からサポートを受けることはできません。
また、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 ブート用フロッピーディスクの作成

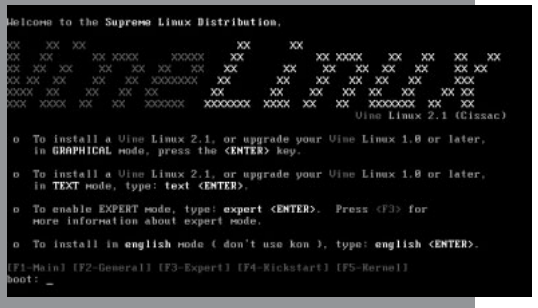
インストールするマシンがCD-ROMから起動できる場合は、CD-ROMからブートしてインストーラを起動します。CD-ROMから起動できない場合は、以下の手順で、インストーラ起動用のフロッピーディスクを作成します（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

（1）Windowsのエクスプローラで、CD-ROMの「dosutils」というフォルダを開き、その中にある「rawrite」をダブルクリックします。

（2）DOS窓が開き、ファイル名の入力を促してくるので、

\images\boot.img

と正確にタイプして[Enter]を押します。さらに、フロッピードライブ名を求めてくるので、Aをタイプして[Enter]を押します。最後にフロッピーがセットされているかを確認して、[Enter]を押すと、フロッピーの作成が始まります。



2 インストーラの起動

作成したフロッピーディスクや、CD-ROMをドライブにセットして、マシンを再起動します。インストーラが起動し、「boot:」というプロンプトが表示されたら[Enter]を押します。

しばらくすると、Xを使ったグラフィカルな画面が表示されます。インストール画面がうまく表示されない場合は、「boot:」の箇所で「text」とタイプして[Enter]を押します。こうすると、テキスト画面のインストーラが起動します。

3 キーボードとマウスの設定

デフォルトでは「モデル」に「Japanese 106-key」が、「レイアウト」に「Japanese」が選択されています。日本語キーボードを使用する場合は、このままの状態で「次」を押します。

次はマウスの設定です。デフォルトでは「3 Button Mouse(PS/2)」が選択されています。PS/2タイプの2ボタンマウスを使う場合は「2 Button Mouse(PS/2)」をチェックします。「3ボタンマウスのエミュレーションを設定する」は、2ボタンマウスの左右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代用する機能です。これは便利な機能なので、チェックしたまま「次」を押します。

インストールタイプの選択

「GNOMEワークステーション」と「全パッケージインストール」を選択すると、Windows領域はそのままの状態で、空き領域にLinux用のパーティションが自動で作成され、ブートローダのLILOがMBRに書き込まれます（画面5へ）。

「サーバ」を選択した場合は、ディスク上のすべてのパーティションが削除されます。

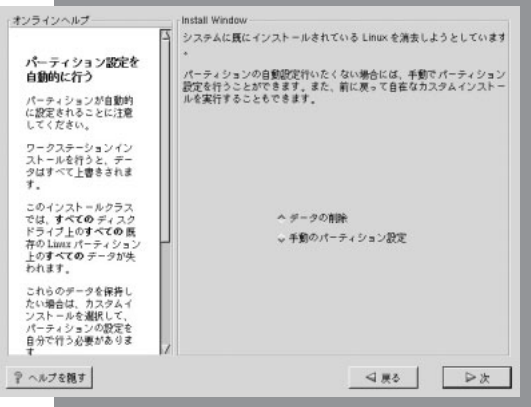
「カスタム」では、ハードディスクのパーティションを手動で設定したり、インストールするパッケージを選ぶことができます。

Windows NT／2000やSystem Commanderのブートローダを使用する場合は、LILOのインストール先が選べる「カスタム」を選択します（画面6へ）。

ここでは「全パッケージインストール」を選択します（未使用のディスク領域が1.3Gバイト程度必要です）。

Linux用パーティション削除の確認

既存のLinux用パーティションを削除してもよいかどうかの確認です。ここではLinux用のパーティションを自動で作成しようとしているので、このまま「次」を押します。



Linux用パーティションの作成

画面4の「インストールタイプの選択」で「カスタム」を選択した場合は、画面（奥）のようなパーティションの設定画面が表示されます。

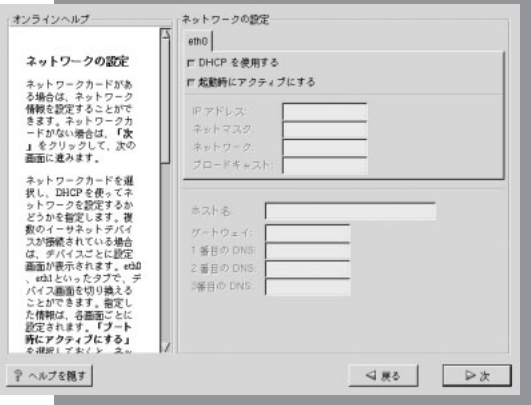
パーティションの設定画面では、Linuxの使用に必要なLinuxシステム用とSwap用の2つの領域を作成します。「次」を押してフォーマットするパーティションを選択し、さらに次の画面（手前）でブートローダLILOのインストール先を選択します。すでにほかのブートマネージャがMBRへインストールされている場合や、Windows NT／2000とデュアルブートする場合はLILOを「ブートパーティションの最初のセクタ」へ、そのほかの場合はLILOを「マスターブートレコード」へインストールします。

ネットワークの設定

家庭内でISDNルータやほかのサーバマシンでDHCPサーバを稼動させている場合は、ここで「DHCPを使用する」を選択します。DHCPサーバがない環境でLinuxを使う場合は、

IPアドレス	192.168.1.2～192.168.1.254
ネットマスク	255.255.255.0
ゲートウェイ	家庭内サーバマシンのIPアドレス
1番目のDNS	プロバイダのDNSサーバのIPアドレスなど

というようなネットワークアドレスを入力するとよいでしょう。



タイムゾーンの設定

デフォルトで「Asia/Tokyo」が選択されています。マシンを日本で使うときは、このままの状態で「次」を押します。

8

ユーザーアカウントの設定

Linuxを使用するユーザーを設定します。まず、Linuxシステム管理者のパスワードとして、「Root パスワード」と「確認」に同じものを入力します。この管理者のログイン名は「root」です。

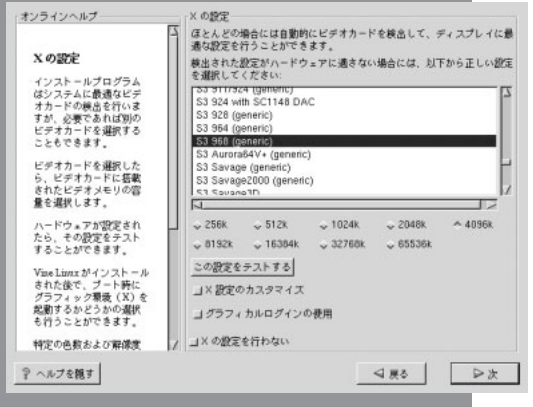
次に一般ユーザーのアカウントを設定します。「アカウント名」、パスワードとして「パスワード」と「パスワード（確認）」に同じものを、「フルネーム」にユーザーのフルネームを入力して、「追加」を押します。この要領で複数の一般ユーザーを作成できます。ユーザー設定を終えたら「次」を押します。

9

モニタの設定

この画面はモニタが自動検出された場合は表示されません。モニタが自動で検出されない場合はリストの中から選択して、リストにない場合は、モニタのマニュアルを参考にしつつ「Generic」から無理のない解像度のものを選択します。

10



Xの設定

多くのビデオカードは自動で検出されます（画面の例ではS3 968と認識されています）。

まず「この設定をテストする」を押して、Xの表示テストを行います。うまくXが表示されない場合は、インストール後にrootでログインしコンソール上で、

```
# kon
# Xconfigurator
```

とタイプしてXを設定するとよいでしょう。

11

パッケージのインストール

手前の画面で「次」を押すとパッケージのインストールが始まります。この作業にはしばらく時間がかかるので、お茶でも飲みながら休憩しましょう。

12

ブートフロッピーの作成

ハードディスクからLinuxを起動できなくなったときに備えて、緊急時用のフロッピーディスクを作成します。空のフロッピーをドライブにセットして「次」を押すとフロッピーの作成が始まります。

13

インストール終了

お疲れさまでした。これでLinuxのインストールは終了です。フロッピーディスクをドライブから抜いて「終了」を押します。

14

起動OSの選択

マシンを起動するとグラフィカルなブートローダLILOが立ち上がります。キーボードの矢印キー（↑↓）でOS名を選択し[Enter]を押すと、選択されたOSが起動します。

LILOを削除する場合は、Windows 95/98の起動フロッピーからマシンを起動して、fdisk /mbrというコマンドを実行します。

15