サン、サーバアプライアンス市場に本格参入 Sun Cobalt Qube3J

サン・マイクロシステムズから、新しいサーバアプライアンス製品「Sun Cobalt Qube3J」（以下Qube3J）が1月25日より発売された。

Qube3Jは、Webサーバに必要なソフトウェアを統合した製品で、ホームページの公開やファイル共有、電子メール、ファイアウォール、Webデータのキャッシング機能を持っている。PCのWebブラウザからQube3Jに接続することで、サーバの管理を行うことができる。

Qube2まではMIPS互換CPUだったが、Qube3JではIntel x86互換CPUのK6-2に変更された。Qube3Jには、3モデルが用意されている。標準モデル（PAJ）は、K6-2 300MHz、32Mバイトメモリ、10.2Gバイトハードディスク、10/100BASE-TXネットワーク×2、PCIスロット×1、USBポートを備えている。サイズは、190.5（W）×193.8（D）×196.9（H）mmで、Qube2よりわずかに大きくなった。

ビジネスエディション（T19J）は、メモリが64Mバイト、ハードディスクが20.4Gバイトになる。プロフェッショナルエディション（U20J）は、CPUがK6-2 450MHz、メモリが128Mバイト、20.4Gバイトのハードディスクを2台搭載し、ソフトウェアRAIDによってミラーリングが可能になっている。

Qube3Jの新機能としては、Webメール機能、DSLルータ／CATVモデムサポート、LDAP対応、新規サービスやアップデート情報を自動的に通知し、更新を行うCobalt BlueLinQ機能などがある。

発売日	2001年1月25日
発売	サン・マイクロシステムズ株式会社
TEL	03-3599-0722
価格	オープン価格
URL	http://japan.cobalt.com/

Hardware　　発売日　2001年2月5日

外部からの不正アクセス・内部ネットワーク監査に役立つパケットブラックホール

URL　http://www.laser5.co.jp/

発売　レーザーファイブ株式会社
TEL　03-5818-6626
価格

レーザーファイブは、ネットワークパケットを完全に記録するアプライアンス製品「パケットブラックホール」を2月5日より発売した。パケットブラックホールは、Debian GNU/Linuxをベースにした製品で、設計／開発をネットエージェントが、製造をナカガワメタルが、販売／プロモーションをレーザーファイブが行う。

既存のネットワークに接続し、パケットが物理的にネットワークからパケットブラックホールへ一方向にしか流れないようになっているため、悪質なクラッカーや一般ユーザーからは検知不可能であり、不正ユーザーが気づかないうちにネットワーク上のすべての行動を記録することができる。

不正アクセスデータのみを抽出する証拠資料作成機能、運用をより簡便にする簡易プライバシー機能、遠隔監視機能、メール／Webの利用に関しての自動分析機能、汎用パケット解析機能、万一の故障時にもネットワークに障害を与えないフェイルセーフ機能を備えている。

19インチ4Uラックマウント、またはスタンドアロンサーバ型で、CPUはIntel IA32アーキテクチャ、メモリ128Mバイト、ハードディスク40Gバイト（増設可能）、10BASE-Tネットワーク、10BASE-4ポートハブ機能を内蔵している。

Software　　発売日　2001年1月31日

複数のLinuxシステムをWebブラウザで集中管理ができるCaldera Volution

URL　http://www.nihonsco.co.jp/

発売　日本SCO株式会社
TEL　03-5486-3906
価格　50万円（10ノード管理）～

日本SCOは、Linuxシステムの導入および管理コストを低減するための総合Linuxシステム管理ソリューション「Caldera Volution」を1月31日より発売した。価格は、ノベルのeDirectory、OpenLDAP、セキュアWebサーバ、10ノード管理までのライセンスを含み50万円。

Caldera Systems, Inc.が開発したCaldera Volutionは、Webブラウザから複数のLinuxシステムを集中管理するためのソフト。

各マシンのハードウェア／ソフトウェア情報の取得、ソフトウェアのインストール／削除、システムモニタリング、システム負荷が基準値を超える場合にSNMPで警報する機能などを備えている。

複数システムの管理には、ノベルのeDirectory、OpenLDAP、iPlanetの主要LDAPディレクトリサービスをサポートする。

管理コンソールとの接続にSSL、LDAP情報はSSLと認証、SQLはネイティブSQL暗号化を利用してセキュア管理コンソールを実現している。

ポリシーやプロファイルと呼ばれる機能を使用して、数千台のLinuxシステムを集中管理することが可能になっている。また、すべての主要Linuxディストリビューションに対応する。

Software　　発売日　2001年2月27日

e-ビジネスを支援するJ2EEアプリケーションサーバBorland AppServer 4.5

URL　http://www.borland.co.jp/

発売　ボーランド株式会社
TEL　03-5350-9358
価格　27万円（開発キット）

ボーランドは、J2EEアプリケーションサーバの最新バージョン「Borland AppServer 4.5」を2月27日に発売する。「Borland AppServer 4.5 開発キット」が27万円、同「運用ライセンス」が150万円。最初に出荷されるのはWindows NT／2000、Solaris 2.6／7／8版で、Linux版の出荷は夏ごろの予定。

Borland AppServer 4.5は、Java開発ツールBorland JBuilder 4との統合により、開発／テスト／配布までをシームレスにサポートする。今回のバージョンアップでは、ERP、CRM、TPモニタなどと接続可能なJ2EE Connector APIを実装したVisiConnectが搭載されている。J2EEのエンタープライズアプリケーションパッケージ技術に対応し、ビジュアル操作によって、すばやくアプリケーションを配布、更新することが可能になる。

また、Web／EJB／CORBAオブジェクトレベルで、アクセスコントロール、SSL暗号化、X.509認証機能を提供するセキュリティ強化アドインソフト「Borland Security Service 4.5」がオプションで提供される。

なお、2001年2月27日～3月23日の期間限定で「Borland AppServer 4.5導入支援キャンペーン」が行われる。小規模ワークグループ向け運用ライセンスで、1サーバのみの分散トランザクション、メッセージングサービス、フォールトトレランス機能などを用いないコンパクト構成版「Borland AppServer 4.5コンパクトサーバー運用ライセンス」が通常50万円のところを37万5000円で提供する。

Software　　発売日　2001年2月9日

異機種混在ネットワーク上の資源をツリー構造で管理する NDS eDirectory 8.5

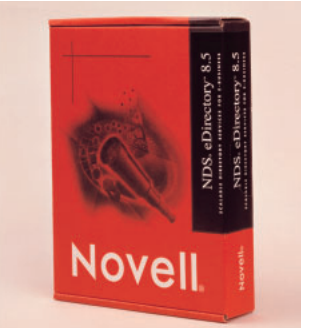
URL　http://www.novell.co.jp/

発売	ノベル株式会社
TEL	03-5481-1294
価格	オープンプライス

ノベルは、ユーザーや組織と、ネットワーク上のアプリケーションやサービスを関連付け、きめ細かく管理できるディレクトリサービス「NDS eDirectory 8.5」を2月9日より発売する。NetWare、Windows NT／2000、Solaris、Linux、Tru64（予定）に対応する。

特定の情報だけをまとめるフィルタード・レプリカ、DNSツリーフェデレーション、ブラウザを使ったモニタリングツールiMonitor、インデックスのカスタマイズ、問い合わせ（LDAPクエリー）の統計を取るプリディケートといった新機能が追加された。

また、NDS eDirectoryをベースに、異種混在のシステム環境でユーザーアカウントの集中管理を提供する「Novell Account Management 2.1」と、XMLを利用して、企業内のさまざまなディレクトリサービスやアプリケーションに散在する各種ユーザー情報やビジネスプロファイル情報の共有／同期化を実現するメタディレクトリソリューション製品「Novell DirXML 1.0」も同日より発売する。



Software　　発売日　2001年1月22日

グループ内で知識DBを共有するナレッジマネージメントシステム Kacis Cabinet Ver.1.0

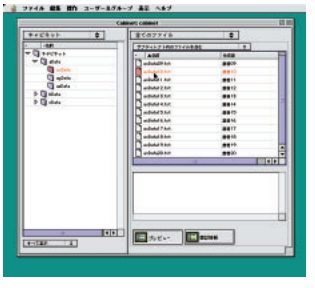
URL　http://www.mvi.co.jp/

発売	株式会社メディアビジョン
TEL	03-3222-4368
価格	120万円〜

メディアビジョンは、カシスと共同で企画／開発したナレッジマネージメントシステム「Kacis Cabinet Ver.1.0」を1月22日より発売した。Kacis Cabinetは、サーバ／クライアント型のシステムで、サーバOSは、Red Hat Linux 6.1J以上、クライアントには、Windows 95／98／Me／NT／2000、Mac OS 8.5以上が対応する。

クライアントから、各種ファイルをドラッグ＆ドロップでKacis Cabinetに格納、閲覧、ダウンロードできる。格納されたファイルにプロフィールやジャンルなどの書誌情報を登録し、ファイルの分類や検索に利用できる。

RDBによりインデックス化された高速な全文検索機能を持っていて、Kacis Book書類、Kacis書類、Word、一太郎、クラリスワークス、テキスト、RTF、PDF、PowerPointファイルに対応する。なお、専用クライアント以外にWebブラウザを利用することも可能になっている。



Software　　発売日　2001年1月15日

安全な掲示板機能を標準搭載したECサイト構築ツール @Trade for Linux

URL　http://www.twincom.co.jp/

発売	株式会社トゥインコミュニケーション
TEL	03-5643-1177
価格	100万円〜

トゥインコミュニケーションは、インターネットショップを立ち上げるためのECサイト構築ツール「@Trade for Linux」を1月15日より発売した。ショッピングモール運営者向けの「@Trade／Mall Manager」と、本格的インターネットショップシステムの立ち上げを目指す企業向け単独ECサイト構築機構「@Trade／EC」がある。

在庫状況がひと目でわかる商品管理システム、商品の追加や削除を容易にする売り場管理システム、お客様の支払い方法や商品の発送状態を確認できる注文情報管理システム、顧客情報管理システムを備えている。@Tradeには、ノーブレイクテクノロジージャパンの掲示板機能「CrazyWWWBoard2000」が標準搭載される。

Software　　発売日　2001年1月31日

複数のマシンへ同時接続可能なPC Xサーバ Exceed v7.0J

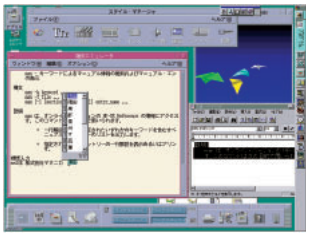
URL　http://www.macnica.co.jp/

発売	株式会社マクニカ
TEL	045-476-1960
価格	7万8000円（1ユーザー）

マクニカは、加HummingbirdのPC Xサーバソフトの最新バージョン「Exceed v7.0J」を1月31日より発売した。Exceed v7.0Jは、Windows95／98／NT／2000上でUNIX（Linux）のX Windows System画面を開くことを可能にするソフト。

新バージョンでは、Microsoft Windows Installerに対応したことにより、自動的にアプリケーションを自己修復する機能や、オンデマンドインストール機能などがサポートされた。また、もっとも低負荷のUNIXホストを自動選択するロードバランス機能の適用範囲を拡張し、より大規模なUNIX使用環境への対応が可能となった。

そのほか、Exceed稼動中の電源管理を行うパワーマネジメント機能や、仮想カラー（Pseudo Color）モードの強化、フォントサーバデータのローカルキャッシング、マウスホイールの設定がより簡単になり、自由な機能割り付けができるなど、数多くの新機能を搭載した。



PENGUINS@Work!

Turbolinuxソフトウェアコンテスト
第1回ターボリナックス・ソフトウェアコンテスト

PENGUINS@Work! 結果発表

ターボリナックス ジャパンが開催したソフトウェアコンテスト「PENGUINS@Work!」で、フリーソフトウェア2作品が最優秀賞に選ばれた。Linux用フリーソフトウェアは、これまで日本人の作者が少なかったが、両ソフトとも完成度は非常に高く、国際的にも通用するレベルに仕上がっている。

ターボリナックス ジャパンが開催した、第1回Turbolinuxソフトウェアコンテスト「PENGUINS@Work!」の受賞作品が決定した。

2000年8月より10月末までに応募された総数35作品の中から審査され、個人部門と団体部門のそれぞれ最優秀作品が選ばれ、その授賞式が2000年12月25日に行われた（**写真1**）。

審査は、操作性やドキュメント、国際化や移植性などの作品の完成度のほかに、Linuxでの有効性や、実用性といった項目で採点が行われた。個人部門最優秀賞には、平林氏のウィンドウアプリケーション統合開発環境「WideStudio」が、団体部門最優秀賞には、株式会社JCraftのピュアJava Xサーバ「WeirdX」が選ばれた（**写真2**）。

写真1　授賞式の参加者
審査員と各メディアの編集者が出席した授賞式。

写真2　最優秀賞の受賞者
WideStudioの作者、平林 俊一氏（左側）と、WeirdXの作者、JCraftの山中 淳彦氏（右側）。

応募結果をまとめると、エントリー総数は57作品だったが、実際に郵送で応募されたのは35作品で、応募作品の95%が個人部門だ。カテゴリーとしては、デスクトップ用が16作品と一番多く、開発用10作品、ゲーム関連3作品、教育・研究用途2作品の順であった。

なお、ターボリナックス ジャパンでは、引き続き第2回Turbolinuxソフトウェアコンテスト「PENGUINS@Work! 2001」を開催する予定だ。詳細は同社のWebサイト（http://www.turbolinux.co.jp/）に掲載される。

個人部門最優秀作品 WideStudio

WideStudioは、Windows 95／98／NT／2000、Linux、FreeBSD、Solaris上で動作するウィンドウアプリケーション統合開発環境である。

Linuxの動作環境として、X Window System、gcc／g++コンパイラ、gdbデバッガが必要だが、ほぼすべてのディストリビューションに含まれているので、そのまま動くはずだ。Windows版には、GNU gcc／Mingw32コンパイラ、gdbデバッガが含まれている。

WideStudioを使うとVisual BasicやC++Builderのように手軽にGUIアプリケーションを構築でき、ユーザーの作ったアプリケーションは、C++のソースコードで出力される。

WideStudioは、MITのX Window Systemと同様のライセンス形態のフリーソフトウェアで、作者のWebサイト（http://www.asahi-net.or.jp/~uj3s-hrby/index.html）からダウンロードできる。原稿執筆時の最新バージョンは1.20で、付録CD-ROMにも収録している（**表1**）。

審査員が評価した点として、そのまま商品化可能な完成度を持ち、実用性も非常に高く、そしてクラスライブラリから自作していてWindowsまで視野に入れているなどが挙げられた。

■WideStudioのインストール

Red Hat系のディストリビューションなら、RPMパッケージを使用してインストールするのが簡単だ。

```
# rpm -Uvh ws-runtime-v1.20.i386.rpm
```

ファイル名	内容
ws-v1.20.tar.gz	ソースコード（ドキュメント含む）
ws-runtime-v1.20.i386.rpm	Linux用ランタイムRPMパッケージ（ドキュメント含む）
wswin120.lzh	Windows用ランタイムパッケージ（ドキュメント含む）

表1　付録CD-ROMに収録したWideStudio v1.20

インストールが終了すると、/usr/local/wsディレクトリ以下に展開され、WideStudioの実行プログラムであるwsbuilderは、/usr/local/binにシンボリックリンクされる。

画面1　WideStudio起動時に表示される画面
アニメーションで「WideStudio」の文字が表示される。wstitleコマンドでこの画面だけを見ることもできる。

画面2　用意されているオブジェクトの例
サンプルプログラム（sample.prj）を実行した。このほか通常のアプリで必要なものが揃えられている。

■WideStudioの使い方

WideStudioの起動は、Xのコンソールからwsbuilderを実行する。最初にタイトルが表示され（**画面1**）、すぐにアプリケーションビルダWS BUILDERが起動する。基本的な操作は、マウスでメニューを選んでいくだけだが、どのようにしてアプリケーションを作成するのか手順を簡単に説明しよう。

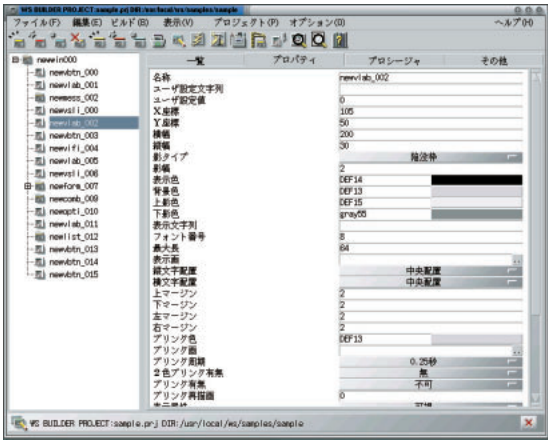
まず、プロジェクトメニューから新規プロジェクトを選び、プロジェクト名を入力する。そして、ファイルメニューから新規ウィンドウを選び、アプリケーションウィンドウを作る。

次に、部品となるオブジェクトをアプリケーションウィンドウに配置する。オブジェクトには、フォームやラベル／ボタン、プルダウン／ポップアップ／オプションメニュー、テキスト入力ボックス、円弧／矩形／折れ線／多角形などが用意されている（**画面2**）。

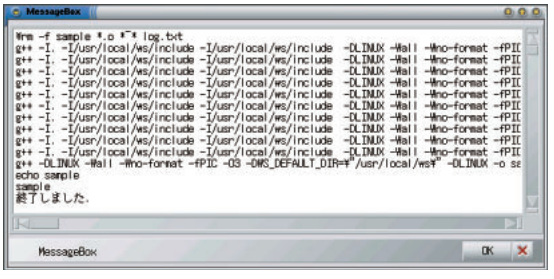
オブジェクトにはプロパティという属性があり、それを変更することで、表示や動作を定義していく（**画面3**）。また、用意されている機能でできない複雑なことを行う場合には、イベントプロシージャを使って、C++で記述することで対応可能だ。関数の受け渡し部分の雛形はWS BUILDERに作らせ、エディタが起動したところで実際の動作部分をプログラムすればよい。

完成したら、ビルドメニューにあるビルドオールを選んで、コンパイル&リンクを行う（**画面4**）。できあがったプログラムを、実行するのも、ビルドメニューから行える。

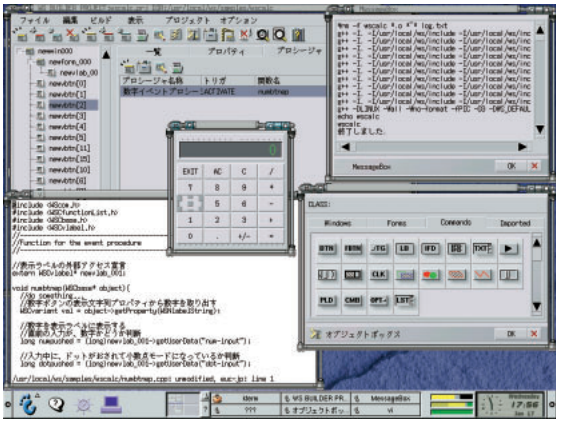
WideStudioは、見てのとおりGUIのアプリケーション作成を助けるツールである。Windowsなど他のプラットフォームにも移植されているため、開発したソフトは、リコンパイルするだけで異機種間で動作するアプリケーションとなる。GUIによる簡単なインターフェイス作成に加え、Javaのようにランタイム環境がないと動作しないといったことがない点と、最終的なオブジェクトがネイティブなバイナリコードのため高速に動作することがメリットとして挙げられるだろう。



画面3　sample.prjの中のプロパティの例
それぞれのオブジェクトには、このようなプロパティが含まれている。表示内容や動作の定義はプロパティで変更する。



画面4　ビルド中のメッセージ表示
コンパイルやリンクの途中経過はこのメッセージウィンドウに表示される。エラーメッセージに注意しよう。

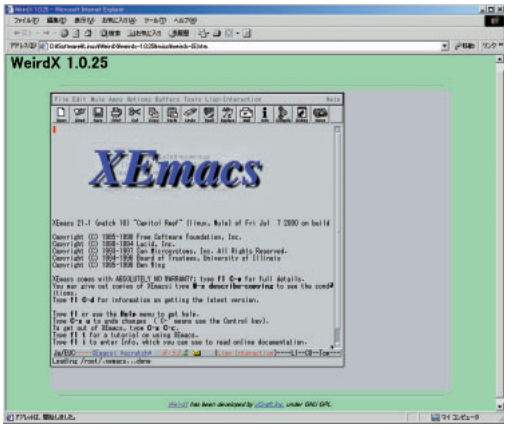


画面5　WideStudioの実行画面
アプリケーション開発中のデスクトップ画面はこんな感じになる。真ん中に表示されているのはWideStudioで作った簡易電卓。

団体部門最優秀作品
WeirdX

ピュアJavaで書かれているWeirdX

画面6　WeirdXの実行画面
X Window System上で、WeirdXを起動して別のマシンのデスクトップを表示した。外側はGNOME、中側はKDEだ。



画面7　Internet ExplorerでWeirdXを実行
Webブラウザ上でJavaアプレットとしても実行できる。IEの画面上でLinuxマシンのXEmacsの操作が行える。

は、JDK 1.1.*以上が利用できる環境で実行可能なXサーバである。WeirdXは、GPLライセンスのフリーソフトウェアで、JCraftのWebサイト（http://www.jcraft.com/weirdx/）からダウンロードできる。原稿執筆時の最新バージョンは1.0.25で、付録CD-ROMにも収録している（**表2**）。

審査員講評として、コンセプトが斬新、Javaを利用しているのでプラットフォームを問わない、クライアントマシンそれぞれにインストールする手間が省けるので、教育機関などの現場で将来的に用途が拡大すると思われるといった点が挙げられた。

■WeirdXのインストール

WeirdXを動かすためには、Javaのランタイム環境が必要である。Sunが運営しているJavaのサイト（http://java.sun.com/）から、JRE（Java 2 Runtime Environment Standard Edition v1.3）またはJDK（Java 2 SDK Standard Edition v1.3）をダウンロードすることができる。Solaris／Windows／Linuxの各プラットフォームに対応するJRE／JDKが用意されている。

Java環境のインストール方法は、先ほどのWebサイトにある「インストール手順」に従って行う。

WeirdX自体は、Javaで記述されており、実行用のJARアーカイブファイルであるweirdx-1.0.25.jarがあるので、コンパイルし直したりする作業は必要ないが、weirdx-1.0.25.tar.gzファイルを展開しておいたほうがよいだろう。ソース以外にドキュメントや実行のためのサンプルなどが参照できる。INSTALLというファイルにインストール上の注意や設定方法が記述されている。

■WeirdXの使い方

LinuxでWeirdXをアプリケーションとして実行するには、シェル上で「java -jar weirdx.jar」とする（**画面6**）。画面の大きさやディスプレイ番号などの設定は、weirdx-1.0.25/configディレクトリにあるpropsファイルを変更するか、「java -Dweirdx.display.width=1024 -Dweirdx.display.height=768 -jar weirdx.jar」のように、コマンドラインに－Dオプションを付けて実行する。

また、WeirdXは、Javaアプレットとして実行することもできるようになっている。Internet Explorer（IE）で実行するには、weirdx-IE.htmとweirdx-1.0.25-rsa.cab（CABファイル）を同じディレクトリに置いて、weirdx-IE.htmをダブルクリックするとIEが起動され、CABファイルが読み込まれる。CABファイルはJCraftで電子署名されているためにセキュリティ警告画面が表示されるが「はい」を押す。

次に、Linuxマシン上で、「xemacs -display ＜WeirdXのマシン名＞:2」のように実行すると、**画面7**のように、IE上でXEmacsを動かすことができる。この2は、ディスプレイ番号で、weirdx-IE.htmに書かれているデフォルト値だ。

なお、XDMCP機能を使って、WeirdX上にLinuxのグラフィカルログイン画面を表示し、完全にリモートで使用することもできるようになっている。その場合にはLinux上で、/etc/X11/gdm/gdm.confファイルのxdmcpセクションにある「Enable＝0」を1に変更し、グラフィカルログインが起動している必要がある。WeirdXの設定方法は添付のドキュメントを参照してほしい。

ファイル名	内容
weirdx-1.0.25-tar.gz	ソースコード
weirdx-1.0.25.jar	実行用のJARアーカイブファイル
weirdx-IE.htm	Internet ExplorerでJavaアプレットとして起動するためのHTMLファイル
weirdx-1.0.25-rsa.cab	電子署名された実行用ファイル。weirdx-IE.htmで必要

表2　付録CD-ROMに収録したWeirdX v1.25

Distribution

新着ディストリビューション

Miracle Linux with Oracle8i Workgroup Server

オラクルデータベースを利用するために開発されたディストリビューションであるMiracle Linuxが、Oracle8i Workgroup Server R8.1.6をバンドルした。これさえあれば、すぐにオラクルサーバを構築できる。

Do Office

日本語環境の使いやすさで多くの人に支持されているVine Linuxに、ハンコムリナックスが開発した複数言語対応のオフィスソフト、HancomOfficeをセットしたデスクトップセットだ。Vine LinuxもHancomOfficeも改良された最新バージョンになっているのがうれしい。

Turbolinux Officeパック

1年前までLinux用オフィスソフトといえば、Applixware Officeしかないという状況だったが、多くのライバル出現に待望の最新バージョン5.0が登場した。機能の豊富さとカスタマイズが可能な点が特徴だ。Turbolinux Workstation日本語版6.0と一緒に使用することで安定した動作が期待できる。

Miracle Linux with Oracle8i Workgroup Server

ミラクル・リナックスから、Linuxとデータベースをパックにした「Miracle Linux with Oracle8i Workgroup Server」（以下Miracle with Oracle8i）が2月1日より発売された。

これは製品名のとおり、オラクルデータベースを使用するために最適化されているディストリビューションである Miracle Linux Standard Edition V1.0（以下 Standard Edition）に、Oracle8i Workgroup Server Release 8.1.6（以下Oracle8i WGS）をバンドルしたもので、このパッケージを利用すれば、IntelベースのPCサーバで容易にOracleを稼働させることできる。

Miracle with Oracle8iにバンドルされるOracle8i WGSは、5指名ユーザー・ライセンスとなっている。本パッケージの価格はオープンプライスだが、Oracle8i WGSとStandard Editionを、別々に購入するよりお得な価格に設定されている。

5指名ユーザーというのは、聞き慣れないと思うが、2001年1月1日の日本オラクルの価格体系の変更に伴って導入されたものだ。従来は、最大何名のユーザーが同時に使用可能という「同時ユーザー・ライセンス」だったが、使用可能なユーザーが何名という「指名ユーザー・ライセンス」に変更された。ついでながら説明しておくと、「特定ユーザー無制限アクセス・ライセンス」と「不特定ユーザー インターネットアクセスライセンス」は、CPU数に応じた「プロセッサ・ライセンス」に変更された。エントリーレベルの価格が下がり、特にインターネットでサービスする場合には従来より導入しやすくなった。

Oracle8iがすぐに使える

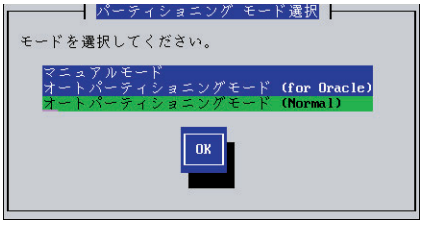
Standard Editionは、カーネル2.2.16をベースに、Oracle8iに適したカーネルパラメータにチューニングされている。1Gバイトを超えるメモリの搭載、OSのファイルシステムを経由せずにハードディスクにアクセスするためのRaw I/Oサポートなども加えられている。Raw I/Oを利用することでオラクルのパフォーマンスを高めることが可能だ。

Miracle Linuxのインストール画面では、オラクル用のメニュー（**画面1**、**2**）が追加されている。OSをインストールしたあとに、グラフィカルインストーラ「Install Navigator for Oracle」（**画面3**）を使って、ウィザード形式でOracleのインストールが行える。

PostgreSQLやSambaに特化した製品も

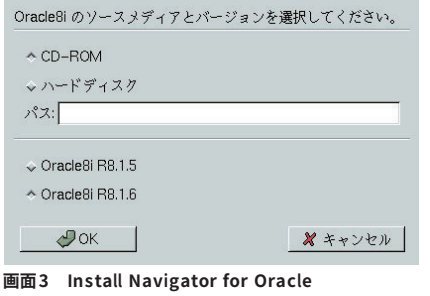
Miracle Linuxは、オラクル専用のディストリビューションといったイメージがあるが、ミラクル・リナックスでは、それ以外のエントリーレベルの製品も準備している。

フリーのデータベースであるPostgreSQLを使った「Miracle Linux for PostgreSQL」と、ファイルサーバ／プリンタサーバに特化した「Miracle Linux for Samba」を、3月1日に発売する予定だ。価格はどちらも1万9800円で、インストール直後からすぐに使えるように、インストーラが工夫されていて、それぞれの機能に必要なツール類もバンドルされる。



画面1　Miracle Linuxのインストール画面
オートパーティショニングモード（for Oracle）を選ぶと、/u01というオラクル用のパーティションが作られる。

画面2　インストールタイプの選択
オラクル用のメニューが追加されている。用途に合わせたパッケージを選べばよい。



画面3　Install Navigator for Oracle
Oracle8iのインストールも、ウィザードに従って進めていくだけで完了する。

製品名	Miracle Linux with Oracle8i Workgroup Server
価格	オープンプライス
問い合わせ先	ミラクル・リナックス株式会社 03-5562-8300 http://www.miraclelinux.com/

Do Office

レッドハットから、Vine Linux 2.1CRとオフィスアプリケーションHancomOffice 1.2をセットにした、Linux版オフィスパッケージ「Do Office」が2月9日より発売される。

Do Officeに含まれるVine Linux 2.1CRは、2000年11月に発売されたVine Linux 2.1CR Official製品版より、いくつかのパッケージが最新のものに変更されているようで、起動すると「Vine Linux 2.1.4CR (Gazin)」と表示される。基本的な部分に変わりはなく、カーネル2.2.17、glibc 2.1.3、XFree86 3.3.6が採用されている。

今回はIntel版だけで、PPC／Alpha／SPARC版は含まれていない。かな漢字変換ソフトにWnn Ver.3、日本語TrueTypeフォントにDynaFont 5書体が含まれている。

インストール直後から日本語環境が使いやすく整っているVine Linuxと、オフィスアプリケーションとして十分な機能を備えたHancomOfficeとの組み合わせは、デスクトップ分野でのLinux利用を促進させるための一歩となるであろう。

HancomOffice 1.2

ハンコムリナックスが開発したHancomOffice 1.2は、2000年10月より発売されたHancomOffice 1.0のバージョンアップ版で、ワープロ、表計算、プレゼンテーション、ペイントツールの4つのアプリケーションで構成されている。

■HancomWord R5.2

MS Wordライクなワープロソフトで、文書中に画像や罫線、表、テキストボックス、数式などを挿入することができる。HTML、MS Word、一太郎、RTFファイルの読み込み、編集、保存が可能になっている。

■HancomSheet 1.2

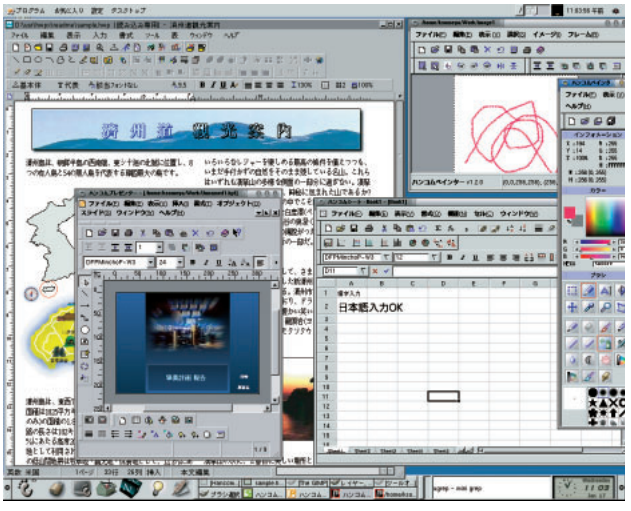
Excelファイルとの互換性を備えている表計算ソフト。複数のシートをサポートし、ソート／自動フィルタ／統計分析／行列計算などの機能を持っている。ウィザードを利用して簡単な操作で11種類の2次元／3次元のグラフを作成できる。

■HancomPresenter 1.2

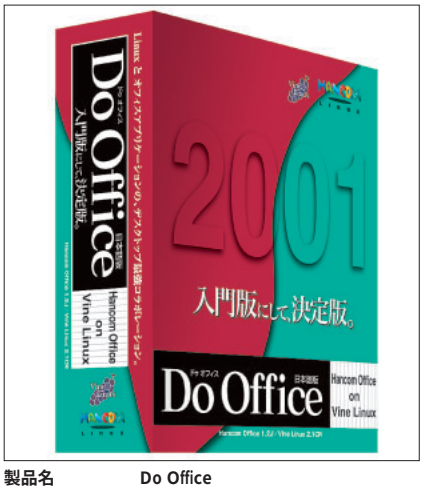
プレゼンテーション資料作成用ソフト。便利に使える700以上のクリップアート、オートシェイプ、サンプルイメージが用意されている。1枚の用紙に複数のページを印刷することや、指定した順序でスライドショーを行うことが可能である。PowerPointファイルとのインポート／エクスポート機能を持っている。

■HancomPainter 1.2

ビットマップ画像の編集、作成ツール。マルチフレームやアルファチャンネルの機能を備えていて、アニメーションGIFの作成や部分的に透明な絵などの描画が行える。GIF、JPEG、BMP、XBM、XPM、PNG、PNMフォーマットに対応する。



画面1　HancomOffice 1.2の実行画面
HancomWord、HancomSheet、HancomPresenter、HancomPainterの4つのアプリケーションを起動している。



製品名　Do Office
価格　1万9800円
問い合わせ先　レッドハット株式会社
03-3257-0411
http://www.jp.redhat.com/

Turbolinux Officeパック

ターボリナックス ジャパンから、オフィスアプリケーションソフト「Applixware Office for Linux 5.0日本語版」（以下Applixware）と、同ソフトとTurbolinux Workstation日本語版6.0（以下Turbolinux WS 6.0）をセットにした「Turbolinux Officeパック」（以下Officeパック）が、1月26日より発売された。

Officeパックは、専用のパッケージではなくApplixwareとTurbolinux WS 6.0の2つのパッケージをシュリンクした形で出荷される。Officeパックの価格は1万9800円で、Applixware単体での価格1万4800円とTurbolinux WS 6.0の1万2800円を足したものより、だいぶお買い得になっている。

Officeパックに含まれるTurbolinux WS 6.0には、**表1**のようなソフトウェアが含まれている。System Commander Liteを使用してWindows 95／98とのデュアルブートが可能なほか、VMware Expressを用いることで、Linux上の仮想マシンでWindows 95／98を動作させることも可能だ。

Applixware Office

Applixwareは、Linux用として歴史のあるオフィスアプリケーションで、1998年からTurbolinuxにバンドルして販売されてきた。今回のバージョン5.0はその最新版で、Microsoft Office 2000ファイルのサポートや多くの改良が行われている。

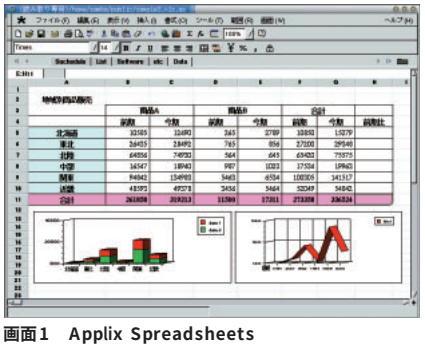
ワープロ「Applix Words」、表計算ソフト「Applix Spreadsheets」、プレゼンテーション「Applix Presents」、グラフィックス「Applix Graphics」のほか、メールソフト、HTMLエディタ、ビットマップ編集ツール、マクロ作成ツールなどのアプリケーションも付属している。

これらのアプリケーション間で、データの埋め込みや参照、相互リンクが可能なため、たとえばWords上に、Spreadsheetsの表やグラフ、Graphicsの画像を挿入するといったことができる。

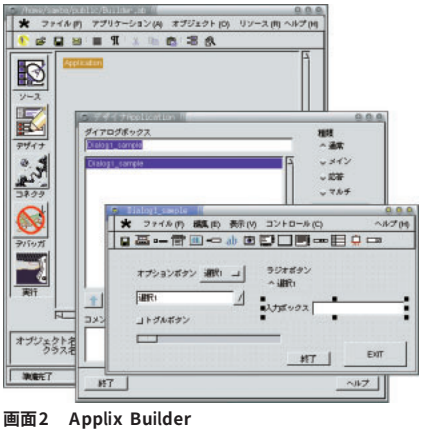
また、Applixwareは、ELF言語で記述されており、ユーザーがカスタマイズ可能になっている。簡単に機能追加が行えるように、グラフィカルなアプリケーション開発環境「Applix Builder」が用意されている。

データベースアクセスツール「Applix Data」を用いて、オラクルなどのデータベースに接続し、データ操作を行ったり、それと連係したアプリケーションを作成するなど、使い込んでいくことで便利になるソフトだろう。

Applixwareの詳細は、122ページからのSoftware Catch UPをご覧ください。



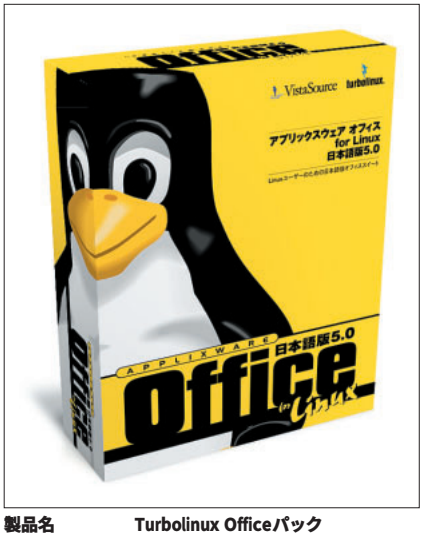
画面1　Applix Spreadsheets
Excelファイルを読み込み、グラフ表示なども可能。



画面2　Applix Builder
GUIでカスタムアプリケーションを開発することができる。

収録物	概要
Applixware Office for Linux 5.0日本語版	オフィスアプリケーション統合ソフト
Wnn6 Ver.3	日本語入力プログラム
ATOK12 SE for Linux R.2	日本語入力プログラム
RYOBI日本語TrueTypeフォント	明朝、ゴシック、羽衣、シリウス、サンセリフの5書体
翻訳魂	和英・英和翻訳ソフト
VMware Express	Linux上でWindows 95／98の仮想PC環境を実現する
Sysrem Commander Lite	マルチブートマネージャ

表1　Turbolinux Officeパックにバンドルされる主な商用ソフトウェア



製品名　Turbolinux Officeパック
価格　1万9800円
問い合わせ先　ターボリナックス ジャパン株式会社
03-5766-1188
http://www.turbolinux.co.jp/

※パッケージ写真は、Applixware Office for Linux 5.0日本語版

Distribution ▶▶▶

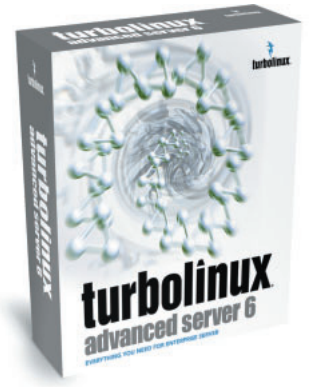
サーバ用Turbolinux Advanced Server 6リリース

ターボリナックスジャパンは、ハイエンドビジネス向けサーバ用OS「Turbolinux Advanced Server 6」を2月16日より発売する。Turbolinux Server日本語版6.1をベースに最新機能を搭載したものに、ホライズン・デジタル・エンタープライズのHDE Linux Controller Turbolinux Edition（以下Turbo Edition）をバンドルした。

安定版カーネル2.2.18ながら、カーネル2.4の機能を先取りして搭載した。2Gバイト超のファイルが扱えるLFSや、複数ディスクにまたがるパーティションを1つのパーティションとして扱えるLVM、ジャーナルファイルシステムであるExt3とReiserFSといったサーバOS向けの機能を備えている。

Turbo Editionは、Webブラウザからサーバの各種設定・管理が行えるツールで、HDE Linux Controller 2.0 Professional Editionから、バックアップ機能と権限委譲機能を除いたものである。Turbolinux Server日本語版6.1にバンドルしていたHDE Linux Controller 2.0 Express Editionに比べ、SSL対応、Proxy、Webバーチャルホスト、ディスク使用量制限、ユーザーアカウント一括登録などの機能が加えられている。

複数のLinuxサーバを管理できるHDE Linux Controller 2.0 Enterprise Editionのクライアントになることが可能で、ASP／ISP／iDCやDB／基幹業務などの大規模サーバシステムでの利用に適している。



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

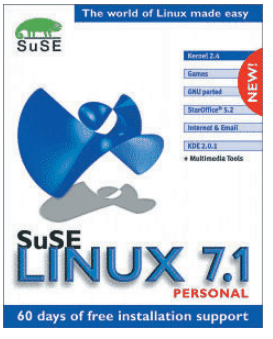
カーネル2.4を採用したSuSE Linux 7.1リリース

ドイツのLinuxディストリビューターSuSE Inc.は、2001年1月4日にリリースされた最新カーネル2.4とglibc 2.2を採用した、「SuSE Linux 7.1」を2月12日に出荷開始する。

64Gバイトまでのメモリ、64ビットファイルシステム、IPv6に対応し、最新デスクトップ環境KDE 2.0やStarOffice 5.2など多くのアプリケーションが含まれる。

なお、USBサポート、NFS V3、Pentium 4に対応したカーネル2.2.18の提供も引き続き行われる。

デスクトップ向けのSuSE Linux 7.1 Personal Editionは、3枚のCD-ROMと3冊のマニュアルで構成され、60日間のインストールサポートが付属し、29.95USドル。サーバ用のSuSE Linux 7.1 Professional Editionは、7枚のCD-ROMと1枚のDVD-ROM、4冊のマニュアルで構成され、90日間のインストールサポートが付属し、69.95USドルとなっている。



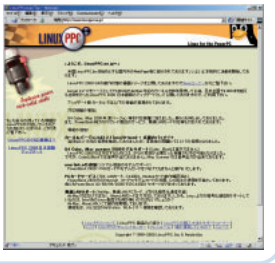
SuSE Inc.（http://www.suse.com/）

日本語LinuxPPCホームページ開設

1月18日から、米LinuxPPC, Inc.のオフィシャルWebサイト（http://www.linuxppc.com/）の日本語版Webサイトとして、日本語LinuxPPCホームページ（http://www.linuxppc.ne.jp/）が開設された。

同Webサイトでは、LinuxPPCの最新情報やアップデートファイルが提供される。

運営は、LinuxPPC 2000日本語版を販売しているアミュレットが行う。LinuxPPCは、CPUにPowerPCを搭載した機種用のLinuxディストリビューションで、Apple社のPowerMacなどで動作する。



アミュレット株式会社
（http://www.amulet.co.jp/）

CorelがLinux部門の一部をスピンオフ

カナダのCorel Corporationは1月23日に、Corel LINUXを開発している部門を分離・独立する計画を発表した。Corel LINUXは、Debian/GNU Linuxをベースに開発されたディストリビューションで、1999年11月に最初のリリースが発売された。WordPerfect Office for LinuxやCorelDraw for Linuxといった、Linux用アプリケーションの開発は継続する。

Corelは、深刻な資金不足に陥ったため、2000年10月にはMicrosoftと戦略的提携を結び、1億3500万USドルの出資を受けている。

Corel Corporation（http://www.corel.com/）

Products

ホットスワップ可能なハードディスクを採用した1U薄型ラックマウントサーバ

DUAXES AS-701R

放熱性に優れたオールアルミシャーシに、工業用のマザーボードを採用し、ハードウェアRAIDコントローラとホットスワップ対応のハードディスクを搭載したことで、高い信頼性とメインテナンス性を備えた、1Uサイズの薄型ラックマウントサーバである。

製品名	DUAXES AS-701R
価格	44万8000円
問い合わせ先	デュアキシズ株式会社 TEL 03-3523-6933 http://www.duaxes.com/jp/

Linuxに対応したフォールト・トレラントのPCシステムを開発・販売しているデュアキシズから、高い信頼性を備えた1Uラックマウントサーバが発売されている。

CPUにPentiumIII1GHzを採用した「AS-1001AR」と、PentiumIII 700MHzを採用した「AS-701R」の2機種があり、どちらも30Gバイトの IDEハードディスクを2台搭載し、ホットスワップに対応したRAID 1（ミラーリング）機能を備えている。

マザーボードには、PCIベースの産業用組み込みボードシステムの規格である、PICMG（PCI Industrial Computer Manufacturers Group）のメインボードを採用している。ケースは、日本製オールアルミシャーシで、厚板アルミを削出加工してあり、優れた放熱性と高い剛性を得ている。

ハードディスクはラックに設置したまま交換可能

今回は、AS-701Rを試用した。マザーボードとして実際に使用されているのは、IBASE TechnologyのIB700というボードで、チップセットにIntel 440BX、CPUはSocket370タイプのPentiumIII700MHzを搭載している。メモリは256MバイトのPC100 SDRAMを標準搭載し、3スロットのDIMMスロットを使用して最大768Mバイトまで拡張可能だ。

写真1　DUAXES AS-701Rの本体内部
日本製アルミシャーシを採用し、1Uサイズながら十分な放熱性を確保している。電源は大容量300Wを採用。

写真2　本体前面と背面コネクタ
フロントパネルのネジを外すと、前面部分からハードディスクを引き出して、簡単に交換できる。背面には、ネットワーク×2、VGA出力、PS/2×2ポートが用意されている。

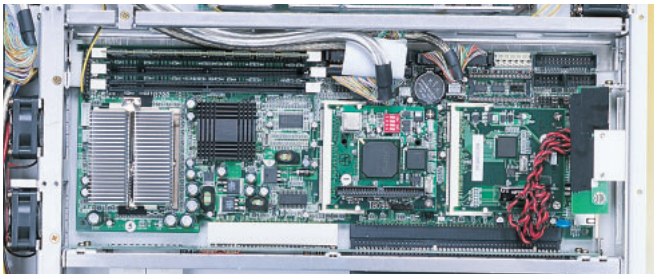


写真3　メインボード
通常のマザーボードではなく、工業用PICMGタイプを採用し、高い信頼性とメインテナンス性を確保している。

グラフィックスは、MicroPCIカードに搭載されたC&T69000（2Mバイト内蔵）を採用している。ネットワークインターフェイスは、10/100BASE-TX対応のi82559を2個搭載している。

マザーボード上には、RS232C×2やパラレルポート、USBも用意されているが、背面コネクタには接続されていないため、もし利用するには別途ケーブルで接続する必要がある。また、拡張スロットには、PCIまたはISAバスのカードを1枚装着可能で、SCSIカードを増設し、テープデバイスなどのバックアップ装置や大容量のRAIDハードディスクなどを接続するといった用途が考えられる。

ハードディスクは、30GバイトのIDEタイプを2台搭載している。向かって左側のハードディスクの上のスペースに、IDE-RAIDカードが装着されていて、ハードウェアでRAID 1（ミラーリング）機能を行う。CPUからは1台のIDEドライブに見えるため、OSの種類によらず利用できる。

ハードディスクはケースに直接ネジ止めされているわけではなく、前面から着脱可能なドライブベイに装着されている。ドライブはIDEインターフェイスだが、ホットスワップが可能なので、もしドライブが故障した場合には、本体の電源を切らずにフロントパネルを外して簡単に交換できる。故障したドライブを正常なものに交換すると、インジケータLEDがオレンジ色に変わり、自動的にデータのリビルドが行われる。リビルド中にはアクセスが遅くなるが、通常どおり使用することが可能だ。

電源ユニットは300Wと大容量であり、本機の構成ではかなり余裕を持たせている。**写真1**では2個搭載しているようにも見えるが、二重化電源ではなく5Vと12Vの2系統に分けているようだ。

本機には、OSは付属していない。対応するOSは、Turbolinux、Red Hat Linux、Windows 2000／NTとなっている。実際に、Turbolinux Workstation日本語版6.0、Red Hat Linux 7J、Kondara MNU/Linux 2000の3種類のディストリビューションをインストールしてみたところ、グラフィックスやイーサネットコントローラなども含めて自動認識され、問題なくインストールできた。

ミラーリングを行うと、2台のハードディスクに同時にアクセスするために、シングルドライブに比べてどうしても性能が低下するが、本機ではハードウェアRAIDのためか、ほとんど気にならなかった。同社によればWinBench99ベンチマークテストで、約98%の性能を得ているという。

AS-701Rは、わずか44mmという薄型ながら放熱と通気性を重視した設計がなされており、ラックマウントタイプのサーバを中心に導入されるASP／ISP／iDCといった、省スペースで高信頼性の機器が要求される用途に向いているだろう。

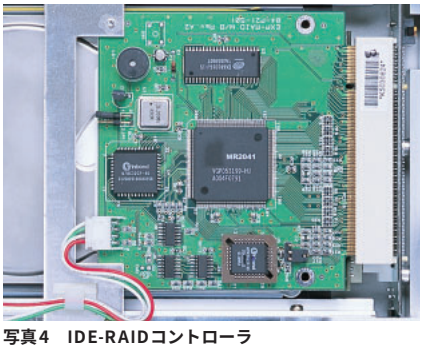
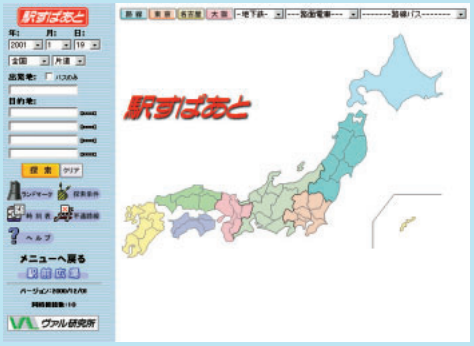


写真4　IDE-RAIDコントローラ
IDEハードディスク2台を、ハードウェアでミラーリング、自動リカバリを行う。ホットスワップにも対応。

CPU	PentiumIII700MHz
RAM	256MバイトSDRAM（PC100、最大768Mバイト）
ハードディスク	30GバイトIDE×2（ホットスワップ対応）
RAIDコントローラ	EXP-RAID（ハードウェアRAID 1ミラーリング）
CD-ROM	24倍速CD-ROM（スリムタイプ）
フロッピードライブ	3.5インチ1.44Mバイト（スリムタイプ）
グラフィックス	C&T69000
ネットワーク	10/100BASE-T×2ポート（i82559）
インターフェイス	PS/2×2ポート
拡張スロット	1スロット（PCIまたはISA）
サイズ（mm）、重量	482（W）×645（D）×44.4（H）、約8kg
電源	300W

表1　DUAXES AS-701Rの主な仕様



駅すぱあと・イントラネット版

出発地と目的地の駅名を入力すると、最適な経路と時間／運賃を計算してくれる駅すぱあとのイントラネット版が、Linuxに対応した。CGIインターフェイスを介して、ユーザーアプリケーションから直接駅すぱあとの機能を利用できる。

製品名	駅すぱあと・イントラネット版
価格	30万円（同時10ユーザー）～
問い合わせ先	株式会社ヴァル研究所 TEL 03-5373-3511 http://www.val.co.jp/

ヴァル研究所から、出発地と目的地を設定して探索すると、その経路と時間、運賃を表示するソフト「駅すぱあと」のLinux対応版の「駅すぱあと・イントラネット版」（以下イントラネット版）が発売されている。初めて行く場所に、どの電車に乗って、どこで乗り換えればよいのか、時間はどれくらいかかるのか、運賃はいくらかがわかるので、外出／出張などで下調べするときや、交通費の精算にとても便利なソフトである。

駅すぱあとには、PCにインストールするスタンドアローン版や、ファイルサーバにインストールしておくネットワーク対応版が用意されているが、イントラネット版は、Webサーバにインストールし、PCのWebブラウザから利用するものだ。従来は、Windows NT用だけだったが、12月12日よりLinux（Turbolinux Server 6.1）に対応した。

全国のJR、私鉄、地下鉄、新交通、路面電車といった鉄道（約160社9600駅1020路線）の時刻表データを元に、正確な探索結果を表示する機能を持っている。そのほかにも、航空路線（約16社85空港300路線）、主要な路線バス（12社約13740バス停2610路線）の情報を内蔵していて、最適な経路を探索できる。

これらの情報は、路線の追加／廃止や、運賃改定、時刻表改正によって変わってしまうため、ヴァル研究所では、1年間に6回最新版を提供している。

イントラネット版の価格は、初年度の最新版提供を含めて、10ユーザー（同時アクセス数）で30万円からとなっている。次年度以降の最新版提供には年間サポート料が必要で、10ユーザーで9万円である。

インストールは手作業で行う

それでは、Linux対応のイントラネット版をインストールしてみよう。Turbolinux Server日本語版6.1をインストールしたマシンには、メモリ64Mバイト、空きディスク容量500Mバイト以上が必要だ。イントラネット版は、CGIとして動作するため、Webサーバ（Apache）が動作することを確認しておく。

まず、イントラネット版のCD-ROMから、expwww.tgzというファイルを/tmpにコピーし、tar xvfz expwww.tgzと行って、/tmp/expwwwに全ファイルを展開する。そして、expwwwディレクトリを、/（ルート）に移動してから、/expwwwディレクトリにある環境設定ファイル（exp.conf）をエディタで修正する。

exp.confに、WebサーバのURLと、

画面1　Webブラウザで地図から選択
東京／名古屋／大阪近郊や、地下鉄／路面電車／路線バスを簡単に選べる。都営バスを選び、バス路線図を表示したところ。

画面2　探索結果の経路一覧
出発地と目的地を入れて探索すると、経路、所要時間、運賃が一覧表で表示される。

CD-ROMに書かれている登録番号、CDキーを記録することで、イントラネット版が実行できるようになる。そのほか、探索結果の回答数の最大値（1～20）や、運賃順や時間順といった並び順、クッキーを使用してブラウザに探索条件や不通路線の設定を覚えさせることの有無などを設定することができる。

次に、Apacheの設定ファイル（/etc/httpd/conf/access.conf）に、/expwwwディレクトリをアクセスできるように**リスト1**の内容を追加して、Apacheを再起動（/etc/rc.d/init.d/httpd restart）するとインストール終了である。

クライアントPCからWebブラウザで、http://＜サーバ名＞/expwww/exp.cgiに接続すると、タイトルにあるような画面が表示される。右側の地図をクリックして詳細図を表示し、駅（バス停）を選択する（**画面1**）。左側のメニュー上の出発地と目的地に駅名を直接入力することも可能だ。そして、探索ボタンを押せば、結果が表示される（**画面2**、**画面3**）。

メニューからは選べないが、管理者用画面（admin_cgi.html）が用意されていて、ランドマーク登録と不通路線の設定が行える。ランドマークに、自分の会社やよく行く施設を登録しておくと便利だろう（**画面4**）。

駅名のすぐ上にある地図のアイコンを押すと、その駅周辺の地図を表示してくれる（**画面5**）。この機能は、MapFan Web、MapionのWebサイトへアクセスして表示するため、実際にはインターネットに接続している環境が必要だ。その上にあるアイコンで、出口情報（**画面6**）、福祉設備（**画面7**）を表示することもできる。

CGIで駅すぱあとを利用できる

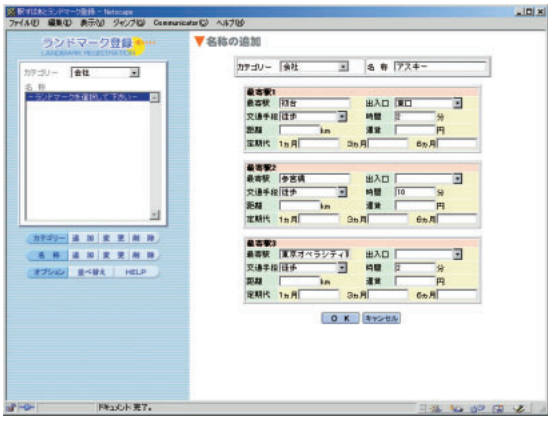
イントラネット版では、CGIインターフェイスが公開されている。たとえば、出張申請・精算といった専用のアプリケーションで交通費精算業務を行う部分を、駅すぱあとのCGIを利用して計算することができる。

このCGI機能を利用して社内システムとの連動ができる点が、イントラネット版の最大のメリットといえよう。

▼リスト1　access.confに追加するパラメータ

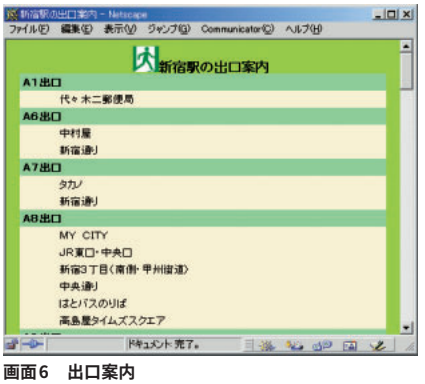
```
Alias   /expwww "/expwww"
AddHandler      cgi-script
.cgi
<Directory      /expwww>
allow from all
Options ExecCGI
</Directory>
```

画面3　探索結果のルート表示部分
経路一覧の下にある各ルートの詳細情報。駅名のアイコンで各種情報、路線の上の時計アイコンで時刻表を見ることができる。

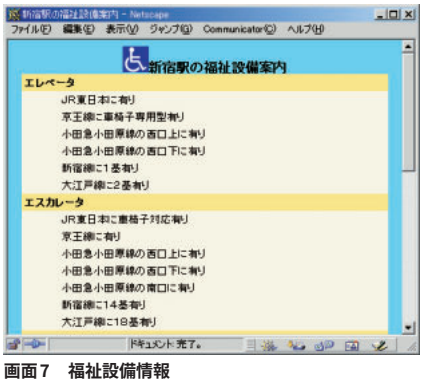


画面4　ランドマーク登録
管理者ページからランドマークの登録が行える。最寄り駅を複数指定しておくと、意外な最短ルートを発見するかも。

画面5　駅周辺の地図を表示
画面3のルート情報にある地図アイコンを押すと、Web上の地図情報（Mapfan Web／Mapionを選択可能）にアクセスする。



画面6　出口案内
駅の出口番号ごとに、建物や方面の情報を見ることができる。地下鉄など複数の出口がある場合に便利。



画面7　福祉設備情報
エレベータ、車椅子用エスカレータ、スロープ、トイレなどの設備情報を表示する。

Linuxで実現するクラスタリングWebサーバ
ワイノット流　Webサーバの作り方

グリーティングカードをインターネット上で提供しているサイトYnot（ワイノット）。ここでは数十台のLinuxマシンを使い、多い日で1000万ページビューあるアクセスをこなしているという。おそらくLinuxのビジネス利用例としては、国内トップクラスだろう。同社が運用しているシステムとはいったいどのようなものなのだろうか？　www.ynot.co.jpを運営するYnotのソフトウェアエンジニアの方々に話を伺った。

Interview

www.ynot.co.jp/

――まず初めにYnotが提供しているサービスについて教えてください。

Ynot：Ynotでは、年賀状や暑中見舞いなどのグリーティングカードをインターネットを通じて配信するサービスを提供しています。まずユーザー登録をしていただきます。そのあと、さまざまな種類のカードの中からお好きなものをを選んでいただき、メッセージとともに相手に送ることができます。またYnotの特徴として「アニメーションeカード」というのがあります。これは、Macromedia社のFlashを利用してアニメーションするカードです。ほかにもFlashを使ったカードはありますが、ダイナミックにジェネレートしているところは、もしかするとうちだけかもしれません。

――ダイナミックにジェネレートするというのはどういうことですか？

Ynot：普通は、Flashのファイルがすでにあって、それに送る人がメッセージをHTMLなどで後付けするので、アニメーションとメッセージは分離したものになります。これをダイナミックにジェネレートすることでFlashのファイル（アニメーション）の中にメッセージを添付することができます。このカードを見に行くと、そこでジェネレートされてストリーミングされます。

画面
Ynotのホームページ。アニメーションeカード、eポストカード、eレターなどのグリーティングカードを送ることができる。デザインはシンプルだが使いやすいページなので、ぜひ訪ねてみてほしい。

1000万ページビューを処理する複雑なサーバ構成

――次に利用者数を教えてください。

Ynot：今現在、登録ユーザー数は135万件です。送信実績でいいますと、昨年の暑中見舞いと残暑見舞いカードの送信数が400万通。クリスマスカードと年賀状の合計送信数が800万通です。アクセス数は、これらのイベントのないときは平均100万pv/d（ページビュー／日）くらいになります。

――年賀状になるとすごい数ですね。元日にはどれくらいのアクセス数がありましたか？

Ynot：1000万pv/dありました。

――1000万ですか。これを全部さばかなくてはならないとすると、かなり大変ですよね。そのサーバの構成について教えてください。

Ynot：サーバは大きく分けると、Webサーバと画像専用サーバの2つの構成になります。

――画像専用サーバとは？

Ynot：うちの場合ですと、1ページの中に画像が20個とか30個とかあります。

仮に20個だとすると、HTMLが1個と画像が20個の計21個のリクエストが来るので、遅くなってしまいます。これは画像が多いからなんですが、サーバを分けると、画像がなかなか表示されないことはあっても、HTMLによるページはポンと出てくるので、とりあえずは使える状態にはなります。ユーザーからするとこの状態ならまだ我慢できます。これが1つのサーバ構成になっているとHTMLも画像も関係なく送られてくるので、結果としてHTML自体も遅くなってしまうというわけです。

――それは画像が大きいからとかは関係ないですか？

Ynot：よく言われるのですが、実は画像のほうがサイズは小さいです。今はHTML自体が大きくなってしまって、実際に画像サーバとWebサーバのグラフを比べてみると、HTTPのリクエストに対する転送量は画像のほうが圧倒的に少ないです。

――それは意外ですね。

Ynot：あとは、個々のHTTPのパラメータを調整して、それぞれのコンテンツの特性に合った状態にしてあげれば、パフォーマンスはかなり期待できます。うちのようにCGIを使っているサイトでは、サーバを分けているところは多いのではないでしょうか。ただ、画像専用に設定を煮詰めているところは少ないようですけど。うちもそれに気がつくまでに時間がかかりましたから（笑）。

サーバOSはLinux

――OSは、すべてLinuxですか？

Ynot：だいたいがLinuxです。

――Windows NTやSolarisという選択もあったと思うのですが、Linuxにした理由を教えてください。

Ynot：最初、FreeBSDにしようかと思っていたのですが、オラクルのクライアントがサポートされていないのでやめました。データベースはオラクルを使いたかったので。そういった条件の中ではIntel版のSolarisという選択はありました。これだとオラクルのクライアントが動くということだったので。ただ、当時はライセンスが不透明な気もしたのでやめました。これと平行して実験的にLinuxを使っていたので、最終的に、だったらLinuxでやろうということになったのです。

――Linuxだとサポートも受けづらく、大変だと思いますが？

Ynot：フリーソフトとか、ドキュメント類が整備されていないソフトを使いこなす必要があったり、サポートがないところに不安を感じる人たちが、NTやSunを使っているのだと思います。

――NTは不安定だという話をたまに聞きますが、実際はどうですか？

Ynot：実はうちでも、Flashのジェネレータ用にNTを使っています。これはNT用しかないので使っているのですが、NTだからといって特に問題が起こったりはしてません。ジェネレータのバージョンが古かったときはよくリブートしてましたが、最新バージョンにしてからなくなりました。このような問題が起こるのは、NTを使いこなしてないからだと思います。もちろんNT上で動いているソフトの安定性とか、NT自体の設定とか理由はさまざまだと思いますが、設定次第である程度カバーできることだと思います。

――Linuxサーバが不安定になるのは、どんな場合ですか？

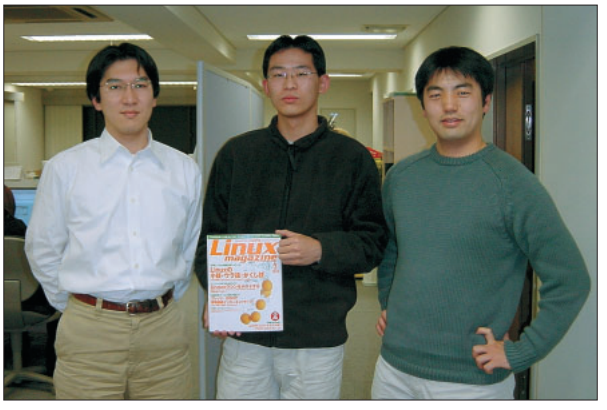
Ynot：Linuxは、OS的にはそれなりの出来ですが、ハードウェアに関するドライバの問題で、不安定になることが多いです。

――Linuxを使ううえでのリスクがあるとすれば、ハードウェアだと？

Ynot：そうですね。リスクというか、初めからサポートを受けられないのは承知してました。メーカーがドライバを出していないものがあるのも容易に想像できました。そこで、自分たちで徹底的に検証を行いました。多少でも不安要素は減らしたかったので、CPUはIntel製、チップセットもIntelの440BXを条件にハードウェアも選びました。ネットワークに関しては、何がいいかよくわかりませんでした。たまたま購入したカードのチップセットがIntelでした。これ以外のチップセットを使用したカードも使ってみましたが、致命的な不具合があったのでやめました。

――致命的な不具合というのは？

Ynot：そのNICが原因で落ちた瞬間に、ネットワーク上のすべてのNICに変なパケットを流すようで、同時にほかのマシンも落ちてしまうのです。実はIntelのチップセットにも同じ症状があって、初め



今回お話していただいたエンジニア方々。左から福田さん、板倉さん、藤田さん。

はまったく原因がわからなかったので苦労しました。これは、このとき使っていたTurbolinux Server 6.0に内蔵してたドライバが原因でした（その問題も過負荷をかけたときにしか現れないものだったのですが）。この件に関してメーカーに聞いてみてもわからず、メーリングリストも見ましたが、たまに「おかしい」という人がいても具体的な解決策は見つかりませんでした。結局、アメリカのサイトで見つけたドライバを試してみたらうまくいったので、問題は無事解決しました。苦労してやっと解決した直後にIntelからドライバが出ましたけど（笑）。

――Linuxを導入するには、それなりの勇気と努力が伴うと？

Ynot：先ほどの話にもありましたが、導入前に必ずテストをすることにしています。このとき、問題に気づくかどうかですね。これは経験とかノウハウとかではないと思うのですけど……。それをおかしいと思うか思わないかは「センス」ではないかと思っています。ただおかしいことはそうはありませんけど。高負荷をかけなければ問題は起きませんから、ほとんどの人は全然問題ないと思います。

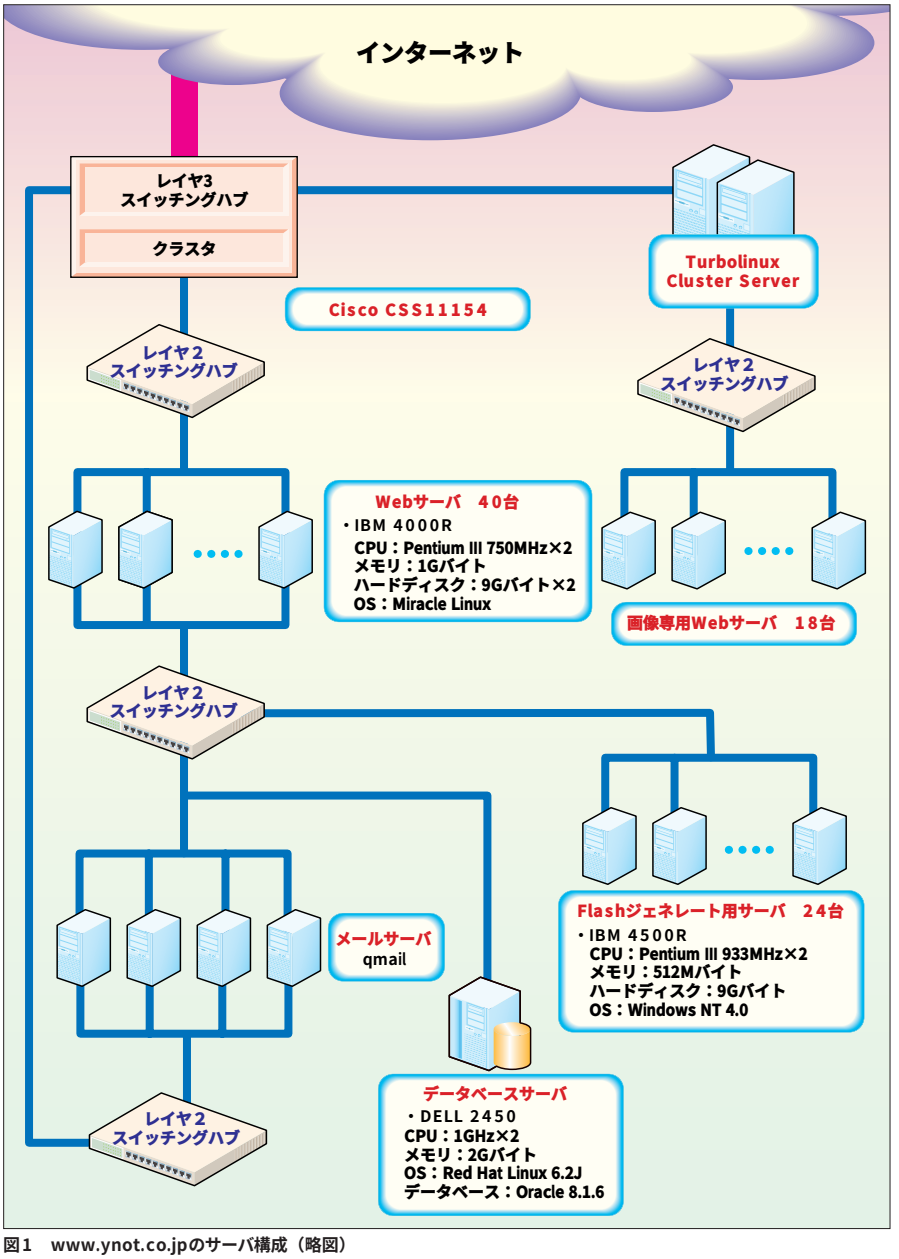


図1　www.ynot.co.jpのサーバ構成（略図）

クラスタにおまかせ

――次にWebサーバについてお聞かせください。

Ynot：うちではCiscoのクラスタマシンを導入しています。その下にWebサーバが40台、Flashをジェネレートするサーバが24台、ユーザー管理などをしているサーバ、そしてメールサーバがあります。あとは先にお話した画像用サーバが18台あります（**図1**）。

――Ciscoのマシンですか？

Ynot：そうです。これはレイヤ3のスイッチングハブとクラスタが一体化した製品です。

――具体的にどのようなことをしているのですか？

Ynot：そのマシンにWebサーバが複数台繋がっていて、管理というかロードバランシングさせてます。マシン内部にあるクラスタで、ユーザーからのリクエストをWebサーバに渡します。このとき、繋がっているマシンの状況や、各サーバのパフォーマンスの状態に合わせてIPを割り振ります。少し前だと、Webサーバを並列で置くときによく利用されているのにDNSラウンドロビンがありました。これは1つのwww.ynot.co.jpに対して複数のIPを書くことができることを利用して、リクエストを複数あるマシンに分散するわけですが、ばらつきなどがでてしまうことがあります。

――特定のマシンに集中するような「偏り」ですね。

Ynot：そうです。でも、ラウンドロビ

ンを使ううえでの一番の問題はほかにあります。たとえば1～4台のマシンのIPアドレスをWWWに割り当てたとして、そのうちの1台が止まってしまったときには1/4の確率でエラーのページが表示されてしまいます。これでは実際は全部はダウンしていないのにエラーになってしまって都合が悪いわけです。そこでうちではCiscoを使って、これより高度なロードバランシングを使用しています。

――Turbolinux Clusterも使っていると伺ったのですが。

Ynot：はい。画像サーバ側もクラスタになっているのですが、そちらがTurbolinux Cluster Serverです。Turbolinux Clusterは4.0から使ってます。

――両方同じマシンでないのは、何か理由があるのですか？

Ynot：一番の理由は金額的な問題ですね（笑）。もともとクラスタサーバっていうものを、私たち自身がよくわからなかったので、何か使ってみようかというような感じで始めました。世の中にはCiscoなどの製品があるのですが、あまり高価なものをいきなり導入してコケたときが痛いというのもあったし、そのときたまたま発売されたばかりのTurbolinux Cluster Server 4.0が目についたので、とりあえずやってみることにしました。

――そのときのマシン構成は何ですか？

Ynot：ぷらっとホームのVTrus-1です。これを2台で使っていましたが、この構成でかなりいけました。

――クラスタで使うマシンは、ハイスペックが必要ですか？

Ynot：実はあまり必要ではありません。というのも、クラスタサーバはルータと同じようなものだからです。ただルータと違うのはIPアドレスの付け替えをしたり、ライブチェックしたりといった付加的な作業をするところなのです。実際ルータ機能を拡張してクラスタを付加した製品もあります。現在使用中のCiscoのCSS11154もレイヤ3のスイッチを拡張した製品ですし、見た目も普通のスイッチみたいです。

Apacheのカスタマイズが肝

――Webサーバのソフトは何を使っていますか？

Ynot：Apacheをカスタマイズして使っています。極限までいらないものを削って軽いWebサーバを作りました。ほかにもtuxとか、phttpdとかいろいろ使ってみました。tuxはRed Hatが出している、たぶん、世界一速いWebサーバだと思うんですよ。

――でもApacheなんですよね。

Ynot：最終的にApacheを選んだ理由は設定が細かくできるからです。一番やりたかったのは、画像をキャッシュさせる時間の設定だったのですが、これがApacheだと簡単に設定できて便利です。うちの場合、1つのWebサーバでApacheが200個くらい走っています。つまり200個の受け付け窓口があることになりますが、これだけあっても、転送時のタイミングによっては全てが埋まってしまうことがあります。そういった状態になった瞬間に、ユーザから見てwebサイトが落ちているように見えるわけです。これはApacheの設定を見直すことで改善できたりするので、カスタマイズがキーポイントですね。

メールサーバはqmail

――Ynotさんでは、メールを大量送信することになると思うのですが、その仕組みについて教えてください。

Ynot：ユーザーがカードを送ったときは、Webサーバが処理して、メールサーバにメールを送るように指示します。そしてメールサーバが送信することになります。日付指定されているメールは、各メールサーバが毎日データベースに問い合わせをして送っています。

――たとえば年賀状などは大量に送信しなくてはならないわけですが、元日はどれくらいメールを送りましたか？

Ynot：うちは送り主にもメールが届く仕組みになっているので、だいたい200万通くらいにはなると思います。

――200万通ですか。それをデータベースに問い合わせているわけですよね？データベースにもかなりの負荷がかかると思うのですが。

Ynot：まったくもってそのとおりです。これと同時にユーザー情報などもやり取りをしているので、かなりの負荷になります。うちのシステムのボトルネックはデータベースかもしれません。データベースはオラクルを使っていますが、チューニングポイントがたくさんあるので、まだまだやりようがあると思います。実際、年末から年始にかけて設定を変更して改善されたところもあります。いずれにしても、これは今後の課題ですね。

――ところでメールサーバは何を使用していますか？

Ynot：qmailです。

――なぜsendmailではなく、qmailなのですか？

Ynot：sendmailより速かったからです。実はメールサーバは複数使っていて、Webからメールサーバまでの配信はeximを使っています。いくつかテストしましたが、相手先ホストがわかっていて連続で送るときはeximのほうがqmailよりも速かったのです。相手先ホストがいくつもあって、並列に送るような場合はqmailのほうが速いので、ユーザーに送るのにはqmailを使っています。それに、qmailhは設定が簡単なのに対し、

sendmailは難しかったので（笑）。

落ちないWebサーバ

――アクセスが一度に集中するとシステムダウンの危険があると思いますが、主な原因はどこらへんにありますか？

Ynot：いくつか考えられますが、まずはクラスタサーバが原因となる場合です。たとえば、ユーザからのリクエストがきたときにAさんは必ず1番のサーバに、Bさんは必ず2番のサーバで処理したいとします。この振り分け方法がいくつかあるのですが、一番ポピュラーなのがユーザーのIPアドレスで割り当てるサーバを変える方法です。この方法では、ソースIPごとにどのサーバに割り振られているかという管理テーブルをクラスタサーバ内で持っています。これが1000や2000ならいいのですが、100万とか200万とかになってくると、かなりのメモリを必要とするのでこれが原因で落ちます。ただクラスタが原因で落ちるようなことはほとんどありません。やはり、主な原因はWebサーバにあります。

――具体的に教えてください。

Ynot：クラスタがWebサーバのステータスをチェックしているとき、Webサーバの反応が鈍いと、このサーバは落ちていると誤認される場合があります。このようなときは、だいたい過負荷状態なので、1台が落ちると負荷がほかのマシンに振り分けられて、加速度的に全部のマシンが落ちてしまうのです。クラスタ側でサーバが過負荷で落ちていると認識したときには、全部のサーバが繋がらなくなってしまうのです。

――元日のアクセスでは平気でしたか？

Ynot：はい、大丈夫でした。こうなるまでにはかなりの時間がかかってますけど（笑）。ひとつ設定を直してよくなると、その次に前よりも多くの負荷がかかって落ちる。そしてまた直して……の繰り返しでしたね。うちもさらに融資を受けることになって、この際マシンをSunにしようかという話も出ました。ただ、この先サーバを今よりも増やすことになるでしょうし、そうなるとお金も今よりかかることになりますよね。そうするとコストパフォーマンスからいってSunというのはなくなりました（笑）。今現在、きちんと動いているのだから、この先もLinuxでいいかなと。なんだかんだいってLinuxを使い続けてきたのは、Linuxが良かったからですからね。　**（聞き手：編集部　菅野）**

募集

あなたもYnotで働きませんか

Ynot（ワイノット株式会社）では社員を募集しています。今回の記事を読んで興味を持った方は下記の募集要項を参照の上当社まで御連絡ください。なお、詳細は当社Webページwww.ynot.co.jp/**を御覧ください。**

●募集職種
プログラマ、コンテンツプログラマ、DB管理者、サーバー管理者
●仕事内容
「http://www.ynot.co.jp/」サイトおよびOEMサイトの運営及び管理。
必要条件
以下のいずれかに該当する方
・プログラマ
perl・CでCGIが作れる方
(掲示板、チャット等を自作できる方、排他制御を正しくできる方)
Javaでプログラムが作れる方
(Servlet、Applet等を自作できる方)
・コンテンツプログラマ
JavaScriptを熟知している方
(半年以上使用経験のある方)
・DB管理者
PostgreSQL、ORACLEの管理・運用ができる方、SQLを理解している方
(外部結合や効率的なインデックスの作成等ができるレベルの方)
・サーバ管理者
Linux、Solaris等の管理ができる方
(Unixの基本的なコマンドを理解している方で、Apache等フリーソフトウェアのインストールのできる方)
・とにかくLinuxに興味のある方
(Linuxに興味があれば技術力は問いません)
●給料
別途応談させていただきます
(経験・能力を考慮し優遇)
●待遇
交通費全額支給、社会保険（健康保険・ 厚生年金・雇用保険・労災保険）
勤務時間
9：30～18：30
休日休暇
土日、祝、年末年始、慶弔休暇、有給休暇
勤務地
東京都千代田区九段南3-4-5フタバビル5F
交通機関
JR市ヶ谷駅 徒歩10分
都営新宿線市ヶ谷駅 徒歩8分
営団地下鉄九段下駅 徒歩10分
なお、この件に関するお問い合わせはメールでお願いいたします
e-mail：recruit@ynot.co.jp

緊急リポート

カーネル2.4徹底大解剖

新世紀のLinuxカーネル

2001年1月4日、21世紀の始まりを待っていたかのように、カーネル2.4.0が正式リリースされた。最初の開発バージョンである2.3.0のリリースから数えること約2年、2000年5月の2.4.0-test1から約8カ月の時を経て、ついに最新安定版Linuxカーネルが登場したのだ。

まずは、Linus Torvalds氏をはじめとするカーネル開発に携わったすべてのプログラマーに敬意を込めて感謝の意を表わしておきたい。

さて、多くの時間と多くの開発者の寄与によって生まれ変わったLinuxカーネルは、どのような機能を備えているのだろうか？　限られた誌面ですべてを詳細に解説することはできないが、本リポートでは新機能／強化ポイントを俯瞰的にとらえることでカーネル2.4の全容を明らかにしていこうと思う。なお以降の解説は、x86（IA32）プラットフォームをベースにしていることをご了承いただきたい。

スケーラビリティの大幅な向上

2.4.0では、プロセス管理やメモリ管理、システムリソースの上限など、カーネルのコア部分において多くの変更がなされている。以下にあげる強化ポイントは、主にハイエンドなサーバ用途で有用なものだ。

■プロセス／スレッド管理

最も明らかな変更は、同時に実行可能なプロセス数の上限が引き上げられたことだろう。カーネル2.2では、ハードコーディングされた値で上限（デフォルト512／最大4096プロセス）が設定されていたが、2.4.0からは、メモリの使用状況に合わせて動的に上限値が変更されるようになった（値は/proc/sys/kernel/threads-maxで管理される）。

また、プロセスのスケジューリングアルゴリズムも変更され、システム上のあらゆるレベルでのパフォーマンスの向上が期待できる。

■マルチプロセッサ対応

SMP（Symmetric MultiProcessor）環境におけるスケーラビリティが改善された。2.4.0では、複数のCPUからのリクエストを処理する際のカーネルの排他制御に関するメカニズムが変更され、1つのCPUのリクエストによってカーネルがロックされる機会が減少している。このため、別々のCPUで実行されている複数のプロセスからのリクエストに対して、カーネルが同時に応答できる可能性が高まっているのだ。

このほかにも、CPUに対するハードウェア割り込みを処理するメカニズム（割り込みハンドラ）の改良、ネットワークサブシステムのマルチプロセッサ対応など、複数のCPUを効率的に利用するための変更がほどこされている。

■仮想ファイルシステム

仮想ファイルシステム（VFS）も大幅に改良された。これにより、標準的なext2ファイルシステムにおいても2Gバイトを超えるサイズのファイルを扱えるようになっている（ブロックサイズ4Kバイトの場合、上限値はテラバイトオーダーに達している）。データベースや動画データなど、巨大なサイズのファイルを扱う必要のあるユーザーにとっては朗報だろう。

ファイルシステムのマウントについても大きな変更があった。従来は、任意のファイルシステムでフォーマット

Column

サポートアーキテクチャについて

もともとはx86アーキテクチャ用のOSカーネルとして開発が始まったLinuxだが、これまでに、PowerPC、Alpha、SPARCといった多くのCPU／ハードウェアアーキテクチャ向けに移植されてきた。カーネル2.4.0では、新たにIntelの64ビットアーキテクチャIA64（Itanium）、IBMのメインフレームS/390、Windows CEマシンで多く採用されている日立のプロセッサシリーズSuperH（SH3およびSH4）、MIPSの64ビットプロセッサMIPS64が追加されている。

それぞれの移植作業プロジェクトのWebサイトは以下のとおり。興味のある方は一度のぞいてみては？

●IA-64 Linux Project
http://www.linuxia64.org/
●LINUX for S/390
http://oss.software.ibm.com/developerworks/opensource/linux390/
●LinuxSH
http://linuxsh.sourceforge.net/
●SGI Linux Scalability Project
http://oss.sgi.com/projects/LinuxScalability/

された1つのパーティションを1つのマウントポイントだけにしかマウントできなかったが、2.4.0では、1つのパーティションを複数のマウントポイントにマウント可能となっている。

■ファイルキャッシュ

従来は、ファイルアクセスの際に読み出しと書き込みを別々のメモリ領域を使ってキャッシュしていた。これが共通のキャッシュで処理されるようになった。2つのキャッシュ間で同期をとる必要がなくなり、メモリ領域も節約できることから、システムパフォーマンスの向上につながるはずである。

■システムリソース

同時に実行可能なプロセス数や最大ファイルサイズ以外にも、利用可能なシステムリソースの上限が大幅に引き上げられている。

システムに搭載可能な最大メモリは64Gバイト、同じくIDEコントローラが10個、ネットワークカードは16枚まで搭載可能になっている。

システムに登録できるユーザーとグループの数も65,536から約42億へと増加した。アカウントに関しては、実用上の制限はなくなったと考えていい。

ハードウェアサポートの強化

サポートハードウェアの充実は、ある意味で今回のバージョンアップにおける最大の目玉だといえる。特にコンシューマ分野でのユーザーにとっては、即効性のある魅力的な強化ポイントだ。

■USB

Linux USB Project（http://www.linux-usb.org/）を中心に進められてきたUSB（Universal Serial Bus）サポートの成果がカーネルソースに統合された。これまでにも、2.3系からのバックポートパッチによって、一部のデバイス（キーボード、マウスなど）の利用は可能であったが、カーネル2.4では、オーディオ、ネットワーク機器、プリンタ、スキャナ、デジタルカメラ、ストレージデバイスなど、サポートされるUSBデバイスが大幅に増えている（**表1**を参照）。

HandspringのVisor（PDA）やDiamond Multimedeiaの Rio500（ MP3プレーヤ）といった、ユーザーに人気の高い携帯デバイスとの通信インターフェイスもサポートされている。

■PCMCIA

USBと同様に、これまでカーネルとは別に開発・配布が行われてきたPCMCIA（PCカード）のサポートがカーネルに統合された。コアのコントロール部分だけでなく、ネットワークカード、無線LAN、IDE／SCSIアダプタなどの各種デバイスもカーネルレベルでサポートされている。

ただし、今回サポート外となったデバイスやcardmgrといった管理ツールを含めたPCMCIAの完全なサポートには、別途配布されるPCカードサービス（pcmcia-cs）のインストールが必要となる。詳細は、pcmcia-csプロジェクトのWebサイト（http://pcmcia-cs.sourceforge.net/）を参照のこと。

■IEEE1394

一般に「FireWire」または「i.Link」として知られているIEEE1394もサポートされた。「EXPERIMENTAL（試験的なサポート）」とはなっているものの、コントローラとインターフェイスのコア部分のサポートが組み込まれている。

■Raw I/Oデバイス

Raw I/Oデバイスとは、カーネル内部のバッファを使用せずに、アプリケーションから直接アクセスできるデバイスだ。ディスクアクセスのタイミングをアプリケーション側で制御できるので、分散したデータに交互にアクセスするデータベースシステムなどにおいて、パフォーマンス向上のために利用されることが多い。Raw I/Oデバイスのサポートは、Linuxのスケーラビリティとシステム構築の柔軟性の向上につながる機能強化だ。

■I2O

I2O（ Intelligent Input/Output）

ドライバの種別※	サポートされるデバイス
USBクラスドライバ	
ハブ	USBハブ全般
オーディオデバイス	スピーカ、音源、DAコンバータなど
マスストレージデバイス	フロッピー、テープ、ZIP、フラッシュメモリデバイスなど
プリンティングデバイス	プリンタおよびプリンタケーブル全般
ACMコミュニケーションデバイス	モデム、TA全般
HID（Human Interface Devices）	キーボード、マウス、ジョイスティックなど
ベンダーデバイスドライバ	
イメージデバイス	デジタルカメラ（Kodac DC-2xxシリーズ）、スキャナなど
マルチメディアデバイス	ビデオカメラ（IBM C-It Cameraなど）、FMラジオ、DABレシーバ
ネットワークアダプタ	USB-to-USBネットワークケーブル（Prolific PL-2302チップなど）、USBイーサネットデバイス（ADMtek AN986チップなど）
ポートデバイス	USBシリアルコンバータ（含むVisor）、USBパラレルポートアダプタ（Lucent USS-720チップのみ）
その他のデバイス	Diamond Multimedia Rio500

表1　カーネル2.4.0でサポートされるUSBデバイス　※　ドライバの種別はLinux USB Projectによる分類に基づく

はPCIのスーパーセットにあたるバス規格だ。I2Oでは、デバイスドライバをOSに依存する部分（OSM）とハードウェア固有の部分（HWM）に分割することが可能になっている。HWMはOSに依存しないため、ハードウェアベンダーが用意する各OSに共通のデバイスドライバを利用できる。カーネル2.4.0では、ブロックデバイス、ネットワークカード、SCSI用のOSMドライバがサポートされている。

■パラレルポート

パラレルポートの機能が強化され、ブート時にI/Oベースアドレス／IRQ／DMAチャネルを自動的に判断するSuper-IOチップセット、DMAを利用した高速なデータ転送をサポートしている。また、パラレルデバイスの情報の取得などに利用されるIEEE1284で規定された拡張モード（EPPとECP）にも完全に対応した。

■ACPI

パワーマネージメント機構としてACPI（Advanced Configuration and Power Interface）が追加された。ただし、カーネル側で用意されているのは、インターフェイス部分のサポートのみで、ACPIの利用には別途管理ツールが必要になる。ACPIの詳細とLinux用の管理ツールに関しては、IntelのWebサイトを参照してほしい（http://developer.intel.com/technology/iapc/acpi/downloads.htm）。

ファイルシステム

期待の高かったReiserFSのサポートこそ見送られたが（2.4.1では採用される見込み）、新たにいくつかのファイルシステムがサポートされた。

■UDF

UDF（Universal Disk Format）は光ディスク向けの汎用ファイルフォーマットだが、現在ではDVD-ROMのファイルシステムとして知られている。加速度的に普及が進んでいるDVDビデオの再生にはUDFのサポートが不可欠となるので、非常にタイムリーな強化ポイントだといえるだろう。

■JFFS

JFFS（Jounaling Flash Filesystem）は、フラッシュメモリに対応したファイルシステムで、名前のとおりジャーナリング機能を備えている。主に組み込み系のシステムで利用されているファイルシステムだ。詳細は、開発元であるAXIS CommunicationsのWebサイトを参照してほしい（http://www.developer.axis.com/software/jffs/）

■Ramfs

Ramfsはメモリを仮想的なディスク装置として使うという点では、従来のRAMディスクと似ている。RAMディスクと異なるのは、独自のファイルシステムである点と、サイズが可変であること。ファイルシステムのサイズは配置されたファイルのサイズに合わせて自動的に調整されるので、メモリを効率的に利用できる。

■ネットワークファイルシステム

ネットワークファイルシステム関連では、NFSがバージョン2から3へとアップデートされた。NFS v3は商用UNIXでも広く採用されており、NFS v2よりもデータ転送効率が高く、スケーラビリティもアップしている。

また、NFSに似たリモートファイルシステムであるCodaのクライアント機能が新たにサポートされた。Codaの詳細については、インフォサイエンスのWebサイト（http://www.infoscience.co.jp/technical/coda/）に日本語の情報が掲載されている。

Windowsの標準ネットワークファイルシステムであるSMB（Server Message Block）についても、オリジナルの拡張に合わせてサポートが強化されている。

■その他の強化ポイント

上記以外にも、SCO UnixWareのBFS、SGI IRIXの1世代前のファイルシステムであるEFSが追加された。また、リードのみサポートされていたOS/2のHPFSが完全にサポートされた。

パーティションタイプについては、これまで各プラットフォーム用の移植版ごとにサポート対象が分かれていたが、この区別を廃して各プラットフォームで使用されているパーティションタイプを認識できるようになった。

NLS（Native Language Support）には、日本語（コードページ932）サポートが追加された。これにより、FATファイルシステム上の日本語ファイル名が扱えるようになっている。NLSでは、Unicode（UTF8）も新たにサポート対象となった。

ネットワークサブシステム

前述したように、マルチプロセッサ対応のための変更に伴いネットワークサブシステムが改良され、NFSやSMBなどのネットワークファイルシステムのサポートも強化された。

このほかにも、パケットフィルタリング機能の大幅な変更（詳細は42ページの**「カーネル2.4実験室・netfilter編」**を参照）、さまざまなネットワークデバイスのサポート強化、ATM（Asyn

chronous Transfer Mode）やxDSLで利用されるPPPoE（PPP over Ethernet）のサポート、さらに後述のkHTTPdなど、あらゆるレイヤでネットワーク機能が全面的に強化されている。

注目の新機能

ここではカーネル2.4.0で新たに導入された注目の新機能を紹介しよう。

■Devfs

Devfs（Device File System）は、デバイスファイルを管理・制御するための新機能だ。従来の方式では、デバイスファイルをディストリビューターが用意するか、ユーザー（システム管理者）がmknodコマンドなどで作成する必要があった。また、ネームスペースが階層化されておらずデバイス名が静的であるため、デバイスドライバ間での名前の競合といった問題に柔軟に対処できなかった。

Devfsは動的なデバイスファイルの生成を実現しており（ドライバのロード時にデバイスファイルがメモリ上のDevfsファイルシステムにエントリとして登録される）、ネーミング規則も変更されて階層化されたネームスペースが採用されている（**図1**を参照）。これにより、上記のような問題を解決し、より柔軟で効率的な管理を行うことが可能となっている。

ネーミング規則の変更によって発生する従来方式との互換性の問題を回避するには、devfsdデーモンが必要になるが、このデーモンはカーネルには含まれていない。Devfsの主開発者であるRichard Gooch氏のWebサイト（http://www.atnf.csiro.au/~rgooch/linux/）からダウンロードできるので、Devfsを利用する場合には、必ず入手しておくようにしたい。

■DRM

XFree86 4.0で導入されたDRI（Direct Rendering Infrastructure）に関連する新しい機能が追加された。DRIはXFree86での3Dグラフィックスのレンダリングを高速化するためのメカニズムで、そのカーネル側でのサポート機能がDRM（Direct Rendering Manager）と呼ばれている。

DRMは、複数のプロセスからのグラフィックデバイスへのアクセスやグラフィック処理用の共有メモリを制御し、高速な3Dレンダリングを安定して継続できるように動作する（DRI／DRMに関する詳細は45ページの**「カーネル2.4実験室・DRM編」**を参照）。

■LVM

LVM（Logical Volume Manager）は、複数のディスクデバイス上のパーティションを1つの仮想的なボリューム（論理ボリューム）として扱うことを可能にする新機能だ。論理ボリュームへのパーティションの追加や削除、ボリュームサイズの変更が可能で、柔軟なディスク管理を実現できる。

LVMの機能を利用するためには、カーネルとは別に配布されているツールが必要になる（詳細は44ページの**「カーネル2.4実験室・LVM編」**を参照）。

■kHTTPd

kHTTPはカーネルに統合されたHTTP（Web）サーバ機能だ。静的なページに対するリクエストをカーネルモードで高速に処理することで、ユーザーモードで動作する通常のWebサーバ（Apacheなど）の補助的な役割を果たす。「HTTPアクセラレータ」といった表現が妥当かもしれない。ただし、CGI、PHPなどによる動的なコンテンツの処理には対応していない。

●

2.4.0へのバージョンアップでは、さまざまな面でスケーラビリティの向上を目指しているとの印象が強く残る。このことは、今後さらに進展していくであろうLinuxのビジネス分野への浸透を後押しするはずだ。

また、個人ユーザーレベルにおいても、システムのパフォーマンスや安定性の向上、ハードウェアの選択肢の増加といった多くのメリットをもたらすはずである。

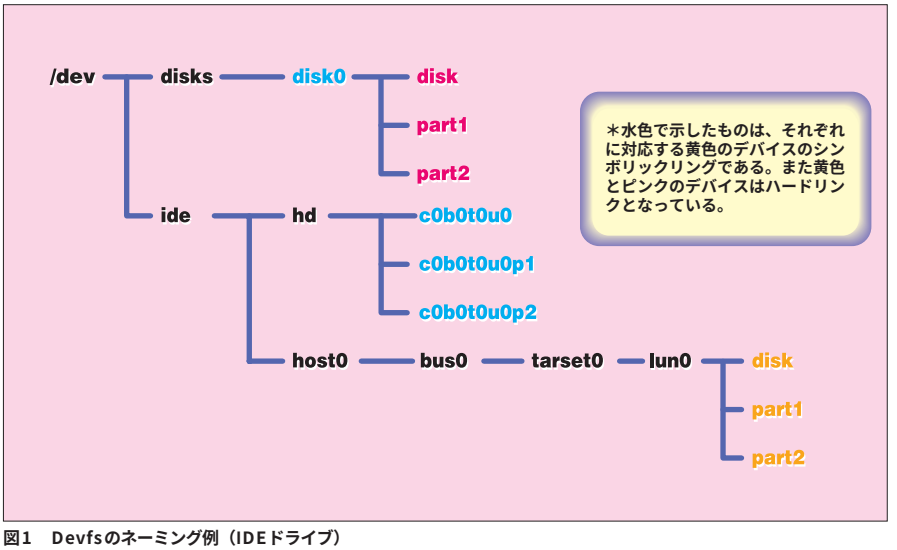


図1　Devfsのネーミング例（IDEドライブ）

カーネル2.4実験室

netfilter編
～パケットフィルタとNATの設定～

カーネル2.4には、パケットフィルタやNAT（Network Address Translation）を行うための基盤として、netfilterという仕組みが導入された。この機能を利用するには、iptablesというパッケージを使用して設定を行う。カーネル2.2で標準のipchainsを利用するためのモジュールも用意されているが、iptablesのほうがより細かく設定できる。そこで、ここではiptablesによるパケットフィルタ、NAT（IPマスカレードを含む）の設定について簡単に解説する。

iptablesを使うための準備

iptablesを利用するには、カーネルにnetfilter機能を組み込んでおく必要がある。カーネル構築の際に、make xconfigやmake menuconfigで、「Networking options」→「Network packet filtering」を有効にし、「IP:Netfilter Configuration」メニューで利用する機能を選んでおく。カーネルの構築方法は、今月号の特集1で解説しているので、そちらを参照してほしい。

カーネルの準備ができたら、iptablesをインストールする（ソースアーカイブを付録CD-ROM Disc2に収録した）。ソースを展開したら、make→make installでインストールできる。デフォルトでは、/usr/local以下にインストールされるので、ほかのディレクトリにインストールしたい場合は、Makefileの書き換えが必要だ。

netfilterの仕組み

カーネルにはパケットの扱いを定めた3つのルールセットがあり、このルールセットをチェーンという。3つのチェーンには、INPUT、OUTPUT、FORWARDという名前が付けられている。Linuxマシンにパケットが入ってくると、カーネルはそのパケットの行き先を調べる（これをルーティングという）。行き先がLinuxマシン自身であればINPUTチェーンに入り、そこに設定されたルールに従って処理される。行き先がLinuxマシン自身ではなく、別のネットワークインターフェイスであればFORWARDチェーンに入る。ただし、カーネルのパケット転送機能が無効の場合や、転送先がわからなければパケットは破棄される。Linuxマシンが発信するパケットはOUTPUTチェーンに入る。

iptableを使えば、これらのチェーンに対してルールを設定したり、新しいチェーンを作成できる。適切なルールを設定することにより、不必要なパケットを遮断して、より安全なネットワーク運用が可能になる。これがパケットフィルタである。

NATは、パケット転送を行うときにIPアドレスやポート番号を書き換えることで実現している。この設定もiptablesを使って行う。

それでは、パケットフィルタ、NATの設定方法を解説しよう。

パケットフィルタの設定

初期状態では、INPUT、OUTPUT、FORWARDいずれのチェーンにもルールは設定されておらず、ポリシーはすべてACCEPT（受け入れる）だ。従って、すべてのパケットが素通しとなっている。

例として、IPアドレスが192.168.10.0/24というネットワークからインターフェイスppp0を介してLinuxマシン宛てで入ってくるパケットを破棄してみよう。rootユーザーになって、**画面1**のようにiptablesを実行する。

-Aは、チェーンにルールを追加するコマンドだ。ここでは、INPUTチェーン（Linuxマシン宛て）にルールを追加することになる。続く-sは、送信元のIPアドレスを指定するオプションで、この例では送信元が192.168.10.0/24というネットワークであることを指定している。-iは、パケットが入ってくるインターフェイスの指定だ。ここでは、ppp0を指定している。-jはジャンプオプションといい、条件にマッチしたパケットをどうするかを決める。この例

```
# iptables -A INPUT -s 192.168.10.0/24 -i ppp0 -j DROP
```

画面1　iptablesでルールを設定する例

では、DROPというターゲットに送られる。DROPターゲットは破棄することを意味する。

iptablesには、**表1**のようなコマンドが用意されている。たとえば、先ほどの例の-Aを-Dに変えて実行すればルールが削除される。パケットフィルタを行う場合は、-Aコマンドを繰り返し使って、各チェーンにルールを追加していくことになる。

ルールの設定には、**表2**にあるオプションを使う。IPアドレスやプロトコル、インターフェイスを指定することで厳密なマッチングが可能だ。

プロトコルには、TCP、UDP、ICMPを指定でき、TCP、UDPの場合は、ポート番号も指定できる。たとえば、“ -p tcp --dport 23 ”とすれば、TCPでポート番号23あてのパケットがマッチする。--dportの代わりに--sportを使えば、送信元のポート番号の指定になる。

インターフェイスの指定オプションは、INPUTチェーンには-i、OUTPUTチェーンには-oしか使わない点に注意しよう。FORWARDチェーンだけが-i、-oの両方を使うことができる。

これらのオプションは、指定しなければすべてにマッチする。**画面1**の例では、宛先のIPアドレスやプロトコルは指定していないので、すべての宛先とプロトコルを指定したことになる。

条件にマッチしたパケットは、-jオプションで指定したターゲットに送られる。主なターゲットは**表3**のとおりだ。通常は、ACCEPTとDROPを使うことが多い。DROPの代わりにREJECTを指定すると、接続を拒否した相手にそれを伝えることになるが、クラッカーかもしれない相手に情報を与えることになるので通常はDROPを使おう。また、-Nコマンドで新しいチェーンを作り、それをターゲットに指定することも可能だ。

NATの設定

今度はNAT機能を設定する。netfilterには、送信元NAT、宛先NAT、IPマスカレードという機能がある。ここでは、PPPやDHCPなど、IPアドレスが固定されていない環境に適したIPマスカレードを使ってみよう。**画面2**のようにすれば、ppp0というインターフェイスから出ていくパケットがIPマスカレードされる。“ -t nat ”は、NATのルールを設定するnatというテーブルを使うことを意味している。POSTROUTINGというのは、NATのために用意された特別なチェーンで、ルーティングを経たパケットが入る。MASQUERADEは、IPマスカレードを行うターゲットだ。

外部インターフェイスに固定IPアドレスが割り振られている場合は、IPマスカレードの代わりに送信元NATを使う。たとえば、外部インターフェイスに1.2.3.4というIPアドレスが割り当てられているなら**画面3**のようにする。

設定後、カーネルのパケット転送機能を有効にすればNATが動作する。

```
# echo 1 > /proc/sys/net/ipv4/ip_forward（実際は1行）
```

今回は、サンプルとしてパケットフィルタと、IPマスカレードの設定を行うシェルスクリプトを付録CD-ROM Disc2に収録しているので、具体的な設定についてはこちらを参考にしてほしい。netfilterには、ここで紹介した以外にもさまざまな機能がある。Linux JF（Japanese FAQ）Projectにより、「Linux 2.4 Packet Filtering HOWTO」、「Linux 2.4 NAT HOWTO」の日本語訳が準備されているので、ぜひ目を通しておこう（http://www.linux.or.jp/JF/）。

コマンド	意味
-P	デフォルトポリシーを変更する
-A	チェーンにルールを追加する
-D	チェーンから指定したルールを削除する
-N	新しいチェーンを作る
-F	チェーンからすべてのルールを削除する
-X	空のチェーンを削除する
-L	チェーンに設定されたルールを表示する

表1　iptablesの主なコマンド

オプション	意味
-s	送信元のIPアドレスを指定する
-d	宛先のIPアドレスを指定する
-p	プロトコルを指定する
-i	パケットが入ってくるインターフェイスを指定する
-o	パケットが出ていくインターフェイスを指定する
-j	ジャンプするターゲットを指定する
-t	テーブルを指定する

表2　iptablesの主なオプション

ターゲット	意味
ACCEPT	パケットを受け入れる
DROP	パケットを破棄する
REJECT	パケットを破棄し、ICMPの「ポート未到達」エラーを送り返す
LOG	カーネルログにログを記録する

表3　iptablesで指定する主なターゲット

```
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

画面2　iptablesでIPマスカレードを設定する例

```
# iptables -t nat -A POSTROUTING -o ppp0 -j SNAT --to 1.2.3.4
```

画面3　iptablesで送信元NATを設定する例

LVM編
～論理ボリュームによるマルチディスク管理～

今回のテストではLVM（Logical Volume Manager）の基本的な機能であるディスクの統合とボリュームのサイズ変更を試みた。設定手順の解説に移る前に、LVMで使われる用語を解説して、その概念をおおまかにつかんでおくことにしよう。

●物理メディア：ディスク装置またはディスク上のパーティション
●物理ボリューム（PV）：物理メディアにLVMの管理用データを関連づけた状態。つまりLVMで管理できるディスク装置／パーティション
●物理エクステント（PE）：LVMにおける最少構成単位。PV上に配置される
●ボリュームグループ（VG）：複数のPV上のPEから構成される管理単位。
●論理ボューム（LV）：VG上の複数のPEから構成される仮想的なディスク

言葉だけでは理解しづらいかもしれない。**図2**に概念図を示してあるので参考にしてほしい。

論理ボリュームの作成と管理

LVMの管理ツールはカーネルとは別に入手する必要がある。配布元のURLはhttp://www.sistina.com/lvm/だ。なお、カーネル2.4.0に対応した0.9.1-Beta2を付録CD-ROMに収録してある。コンパイルは付属の文書を参照してもらえば簡単にできるはずだ。

カーネルのLVMオプションはすでに有効になっているものとして、手順を説明しよう（カーネルのインストールについては46ページからの特集1を参照）。ちなみにモジュール化した場合のモジュール名はlvm-mod.oとなる。

まずLVMで利用するパーティションを決定し、fdiskコマンドでパーティョンタイプを「8e」に変更する。続いて以下に示すコマンドを実行して、PV、VG、LVの順に作成していく。

```
# vgscan
# pvcreate /dev/hda2 /dev/hda3
# vgcreate lvm01 /dev/hda2 /dev/hda3
# lvcreate -L 600M -n lv01 lvm01
```

この例ではVG名を「lvm01」、LV名を「lv01」としているが、もちろん任意の名前でかまわない。lvcreateコマンドの引数は、「－L ＜LVサイズ＞ －n ＜LV名＞ ＜VG名＞」となる。これで、LVのデバイスファイル「/dev/lvm01/lv01」が作成される。

あとは、通常のパーティションを扱うように/dev/lvm01/lv01にファイルシステムを作成してマウントすればOK。論理ボリュームへの読み書きが可能になる。標準のext2だけでなく、そのシステムでサポートされる任意のファイルシステムが利用できる。

「gvdisplay -v ＜GV名＞」コマンドを実行すると、詳細なLVM情報が表示される。これを見れば、LVMの構成がよりよく理解できるはずだ。

VGに空きがあり、ext2を利用していれば、e2fsadmコマンドを使って記録されたデータを損なうことなくLVを拡張できる。ただし、この操作にはresize2fsコマンドも必要で、このコマンドはLVMツールではなくext2用のツールを集めたe2fsprogsのバージョン1.19以降に含まれているので事前に確認しておこう（e2fsprogs 1.19もCD-ROMに収録してある）。コマンドライン自体は簡単で「e2fsadm ＜LVパス＞ －L＋＜増分サイズ＞」となる。－Lオプションにマイナス値を指定してサイズを縮小することも可能だ。

最後にVGを拡張する方法を確認しておこう。新たに別のPVを作成し、それをVGに追加すればよい。コマンドは次のようになる。

```
# pvcreate /dev/hda4
# vgextent lvm01 /dev/hda4
```

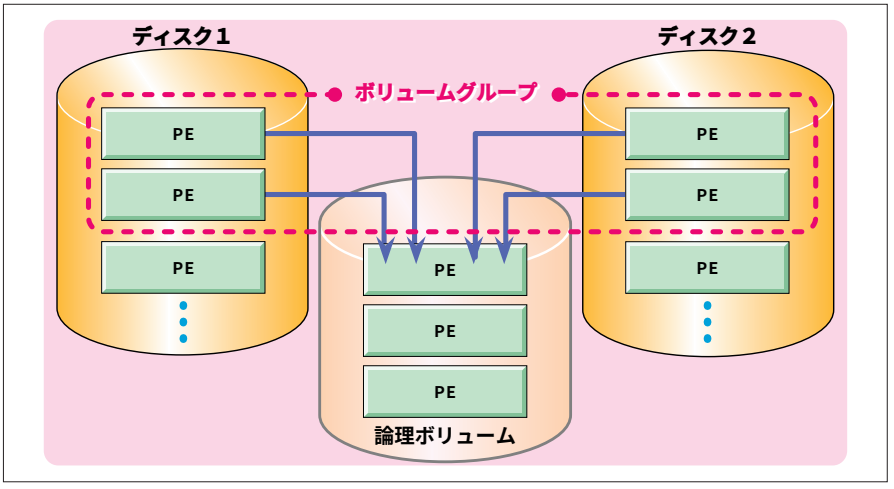


図2　LVMの概念図

DRM編
～ハードウェア3Dアクセラレーション～

前述したようにDRM（Direct Rendering Manager）は、XFree86 4.0.xでサポートされているDRI（Direct Rendering Interface）のカーネル側でのサポート機能である。DRMの役割は、DRIと協調して動作することで、3Dグラフィックスの高速な描画を安定して実行することにある。現在のところ、DRMのサポートは**表2**に示したグラフィックチップのみに限られている。

カーネルコンフィグレーションでは、［Character devices］パネルの最下部にある［Direct Rendering Manager］オプションを“y”にし、ビデオカードに搭載されているチップに合わせて追加のオプションを選択する（モジュール化しておこう）。また、DRIを有効にするためには、DRMオプションの上にある［/dev/agpgart］オプションを“y”にしておく必要がある。こちらもやはり、ハードウェア環境（マザーボードのチップセット）に合わせて追加オプションを選択する。

Rage 128でのテスト

今回のテストには、ATIのRage 128 Proを搭載したビデオカードを使用した。先ほどのカーネルコンフィグレーションでインストールを行うと、/lib/modules/2.4.0/kernel/drivers/char/drmにr128.oというカーネルモジュールが作成される。これがRage 128用のDRMモジュールだ。

DRIのモジュールも確認してみよう。なおXFree86はバージョン4.0.2を使用した。Rage 128用モジュールは/usr/X11R6/lib/modules/driにあるr128_dri.soだ。このほかにもDRI関連には、/usr/X11R6/lib/modules/extensions/にあるlibglx.a、libdri.a、libGLcore.aなどのモジュールがある。また、XFree86側のDRMモジュールは/usr/X11R6/lib/modules/linux/libdrm.aだ。

DRM／DRIモジュールとも、DRIが有効な状態でXが起動する際に自動的にロードされる。XF86ConfigでのDRI設定は“Modules”セクションに2行と、簡単な“DRI”セクションを追加するだけでよい。

```
Section  "Module"
        ⋮
    Load  "glx"
    Load  "dri"
EndSection

Section  "DRI"
    Mode  0666
EndSection
```

「Mode 0666」はすべてのユーザーにDRIの利用を許可する設定だ。

■動作の確認

設定が完了してXが無事起動したら、まず各モジュールが正しくロードされているかどうかを確認してみよう。ロードされていれば、Xのログファイル（/var/log/XFree86.0.log）に記録されているはずだ。モジュールのロードメッセージに続いて、**リスト1**に示すようなログが残っていれば、DRI／DRMによるハードウェアアクセラレーションが有効になっている。

ダイレクトレンダリングの有効／無効は、glxinfoコマンドでも確認できる。引数なしで実行して「direct rendering: Yes」、「OpenGL renderer string: Mesa DRI Rage128」といった情報が表示されれば有効になっている。

■簡易ベンチマーク

厳密なものではないが3Dアクションゲーム「Quake III Arena」のデモ版で利用できるベンチマーク機能を使って計測したところ、色深度24bppで約14～15fps、16bppで約32～35fpsという値が得られた。DRI／DRMなしのソフトウェアレンダリングでは計測不能なほど遅い（途中で固まってしまう）ことから、ハードウェアアクセラレーションによる一定の効果がはっきりと確認できる結果だといえる。

メーカー	グラフィックチップ（ビデオカード）
3dfx	Banshee、Voodoo3以降
3DLabs	Oxygen GMX 2000
ATI	Rage 128、RADEON DDR/SDR
Intel	i810、i810-dc100、i810e
Matrox	G200、G400

表2　DRI／DRMでサポートされるグラフィックチップ

▼リスト1　XFree86のログファイルの例

```
(==) R128(0): Write-combining range (0xd4000000,0x2000000)
(0): [drm] loaded kernel module "r128"
                        ⋮
(II) R128(0): Using XFree86 Acceleration Architecture (XAA)
                        ⋮
(0): [drm] installed DRM signal handler
(0): [DRI] installation complete
(II) R128(0): [drm] Added 128 16384 byte vertex/indirect buffers
(II) R128(0): [drm] Mapped 128 vertex/indirect buffers
(II) R128(0): Direct rendering enabled
```

失敗しないカーネル

2.4のインストール

文：編集部
Text : Linux magazine
Photo: Shuichi Mito(Dee)

速報でお伝えしたように、カーネル2.4には数々の新機能が盛り込まれ、ハードウェアへの対応も一層強化された。しかし、その恩恵に浴するためには、カーネルのコンフィグレーションとインストールという作業が必要だ。そこで、本特集では、カーネル2.4を円滑に導入するための手順を解説しよう。

はじめに、カーネルのインストール心得、導入の際に必要となるソフトウェア環境を伝授する。次いでインストール作業の手順を説明しよう。さらに今回は、コンフィグレーションメニューの中から、初心者が迷いやすい項目を選んだ簡単ガイドもお届けする。慣れてしまえば、カーネルの再構築は恐くない。

これだけはやっておこう！
カーネルバージョンアップの傾向と対策

カーネルバージョンアップ五箇条の心得

『緊急リポート』でもお伝えしたように、待望の最新Linuxカーネル2.4.0がついにリリースされた。新機能が盛りだくさんで、さっそくにでもインストールしてみたいところだ。しかし、カーネルといえばOSの中核、バージョンアップの影響はシステム全体に及ぶ。インストールに際しては、気を引き締めて取り組まなければならない。そこで、わが編集部としては以下のような「五箇条の心得」を提案したい。

■バージョンアップは自己責任のもとで行うべし

いきなり脅かすようで恐縮だが、これはしっかりと認識してもらいたいことである。バージョンアップによって起こる問題については、すべてユーザーがその責を負わなければならない。「オン・ユア・オウン・リスク」というヤツである。

■バージョンアップの必要性を徹底検討すべし

特にビジネス環境や常時接続環境のサーバとしてLinuxを利用している場合には、十分な検討が必要だ。新しくサポートされた機能の必要性とその効果、バージョンアップによって問題が発生する可能性とそれへの対応策などを考慮し、バージョンアップのメリットとデメリットをきちんと把握したうえで結論を出すようにしたい。

もちろん個人ユーザーでほかへの影響がなければ「早くアノ新機能を試してみたい」とか、「ソフトはすべて最新版でなきゃ夜も眠れません」とか、「とにかくカーネル2.4が見てみたい」といった理由でもOKだ（と思う）。

■スキルに合わせてインストール方法を選択すべし

カーネルをバージョンアップする方法は、なにもソースからのコンパイルに限られているわけではない。ディストリビューターが配布するカーネルアップデート用のバイナリパッケージを使ってもいいし、ディストリビューションで新しいカーネルが採用されるのを待ってもいい。コンパイルに自信がないなら、これらの方法が安全・確実でお勧めだ。

執筆時点で2.4系カーネルを採用しているディストリビューションには、CalderaのLinux 2.4 Technology PreviewとメディアラボのLinux MLD5がある。Red Hat 7JやLaser5 Linux 6.4などのように、バイナリパッケージを提供しているものもある。ただし残念ながら、どれもテスト版の2.4だ。

カーネル単体でのパッケージ配布を行わないディストリビューターもあるだろうから、やはりソースからコンパイルする方法が最も一般的だ。自分のシステムに合わせてカーネルを設定できるというメリットもある。バージョンアップを機会に、システム構成に最適化した「マイ・カーネル」を構築してみよう（というわけで、以下の説明はソースからのコンパイル／インストールを前提に話を進めていく）。

■付属ドキュメントには必ず目を通すべし

何はともあれカーネルソースツリーのルートにあるREADMEファイルを読もう。いきなり「WHAT IS LINUX？」で始まって気が抜けそうになるかもしれないが、インストールの方法や注意点、パッチの適用法などが詳しく解説されているので、しっかりチェックしておこう。

README以外にも、カーネルソースツリーには機能やハードウェアごとに、開発者がユーザーに知らせておきたい情報をまとめたドキュメントが数多く含まれている。そのほとんどはソースツリーのルートの直下にあるDocumentationディレクトリに収められている。最低でもChangesファイルには目を通しておくこと。インストールに際して必要となるコンパイラやツールといった「ソフトウェア要件」がリストアップされている。そのほかのドキュメントについては、00-INDEXファイルに記載されている各ドキュメントの説明を読んで、インストール環境に関連するものをピックアップして目を通しておけばいいだろう。

■インストールの前準備は慎重に行うべし

ここまで説明してきたことも準備といえば準備なのだが、ここでの「前準備」とはコンパイルに必要なソフトウェアのインストールやハードウェア環境の再確認といった作業のこと。何ごともしっかりした準備は欠かせないもの。これについては、次ページにて詳しく解説することにしよう。

インストールの前準備

■ソースファイルの入手

カーネルのソースファイルがなければ始まらないので、何らかの方法で入手してほしい。とか言いつつ付録CD-ROMに収録してあるので、それを活用しよう。

今後のバージョンアップに備えて、以下に主なダウンロード元のURLを載せておく。現に、次のカーネル2.4.1のリリースに向けて着々と開発が進行中だ。

```
ftp://ftp.kernel.org/pub/linux/kernel/v2.4/
ftp://ftp.kddlabs.co.jp/pub/linux/kernel/v2.4/
http://www.ring.gr.jp/archives/linux/kernel.org/kernel/v2.4/
```

■必要なソフトウェアのアップデート

カーネルソースを入手してドキュメントを熟読（忘れずに！）したら、次はコンパイラやツールのアップデート。前述したように、カーネル付属のChangesファイルにソフトウェア要件が記載されている。**表1**はそれをまとめたものだ。それぞれのバージョンをチェックして、必要なものがあればアップデートしよう（表1で※マークの付いているものは、付録CD-ROMに収録している）。pcmcia-csとPPPについては、環境に合わせて判断してほしい。使用しない場合は、カーネルのコンパイルのためだけにアップデートする必要はない。

インストール方法は、RPMパッケージのものは次のようにコマンドを実行する。

```
# rpm  -Uvh  <RPMパッケージ名>
```

ソースファイル（tarボール）の場合も、ほとんどのものがtarボールの展開先ディレクトリで、“./configure”、“make”、“make install”とすればいいはずだ。ただし、コンパイルの前に必ず付属のドキュメントを読んでおこう。

Changesに記載されている以外にも、カーネルのバージョンアップで影響を受けるシステム関連のソフトウェアがある。代表的なものは、カーネルで標準的にサポートされて「いない」デバイスのドライバ（カーネルローダブルモジュール）だ。デバイスドライバは、完全にカーネルのバージョンに依存する。ハードウェアベンダーが独自に配布しているデバイスドライバを使っている場合などには、サポート状況を確認する必要がある。たとえば、ALSA（Advanced Linux Sound Architecture）はバージョン0.5.10aでカーネル2.4.0のプレビューリリースに対応した（最終リリース版は正式にはサポートされていない）。

ワープロやWebブラウザといったアプリケーションは通常、影響を受けないが、中にはカーネルのバージョンに依存するものがあるかもしれないので、念のためにマニュアルなどに注意書きがないか確認したほうがよい。

■ハードウェア環境の再確認

カーネルのインストール時に指定するオプションには、システムのハードウェア環境に関するものが多い。インストールを行う前に必ず確認しておこう。必要になる情報は、CPUの種類、マザーボードのチップセットとビデオカードで使われているチップの型番、それにネットワークカード、SCSIアダプタなどの拡張インターフェイスの型番といったところだ。もちろん環境によっては、これ以外にもたとえばPCカードやサウンドカードなどの情報が必要になることもある。

チェック方法は、ハードウェアのマニュアルを見るのが手っ取り早くて確実。なくしてしまっている人は、dmesgコマンドを使ってチェックしよう。起動時にカーネルが認識したハードウェア情報のログを見ることができる。また、lspciコマンドがシステムにインストールされていれば、マザーボードのI/Oコントローラチップと PCIバスに装着されているカードの情報を確認することが可能だ。

	必要バージョン	バージョン確認コマンド	配布元のURL
GNU C	2.91.66（このバージョンを強く推奨）	gcc --version	ftp://ftp.valinux.com/pub/support/hjl/gcc/
GNU make	3.77以上	make --version	ftp://ftp.gnu.org/gnu/make/
binutils※	2.9.1.0.22以上（2.9.5以上を推奨）	ld -v	ftp://ftp.valinux.com/pub/support/hjl/binutils/
util-linux※	2.10o以上	fdformat --version	ftp://ftp.win.tue.nl/pub/linux-local/utils/util-linux/
modutils※	2.4.0	insmod -V	ftp://ftp.kernel.org/pub/linux/utils/kernel/modutils/v2.4/
e2fsprogs※	1.19	tune2fs --version	ftp://download.sourceforge.net/pub/sourceforge/e2fsprogs/
pcmcia-cs※	3.1.21以上	cardmgr -V	ftp://pcmcia-cs.sourceforge.net/pub/pcmcia-cs/
PPP※	2.4.0b1	pppd --version	ftp://linuxcare.com.au/pub/ppp/

表1　カーネル2.4.0のインストールに必要なソフトウェア

カーネルのコンフィグレーション&コンパイル

カーネル2.4の構築に必要なパッケージがそろったら、いよいよインストールだ。インストールの手順は今までのカーネルと変わるところはない。大まかな流れとしては、ソースの展開→コンフィグレーション→コンパイル→インストールとなる。ここではまず、カーネルインストールの手順を解説し、次のセクションでインストール作業の最難関「カーネルコンフィグレーション」の内容を説明しよう。

カーネルインストール作業の手順

バージョンに関わらず、カーネルの入れ替えには危険がともなう。新しくビルドしたカーネルがうまく動かないとLinuxが起動しなくなるのだ。そこで、現在動作しているカーネルでも起動できるようにしておこう。ここでは、ほとんどのディストリビューションで標準的に採用されているLILOでの設定を説明する。

複数のカーネルを選んで起動できるようにするには、LILOの設定ファイル/etc/lilo.confをエディタで開き、**リスト1**のように書き換える。

▼リスト1　/etc/lilo.confを書き換える

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
lba32
prompt
timeout=50
default=linux

# 元のカーネル
image=/boot/vmlinuz-2.2.18-1
        # ラベルをlinux-oldにする
        label=linux-old
        read-only
        root=/dev/hda1

# 新しいカーネル
image=/boot/vmlinuz
        label=linux
        read-only
        root=/dev/hda1
```

ハードディスクのデバイス名やパーティションの名前は環境によって異なる。また、元のカーネルの名前もディストリビューションによって違うだろう。ポイントは、新しいカーネルのためにエントリを追加し、元のカーネルのエントリにあるラベル名を書き換える点だ。ほかの部分はそのままにしておく。

カーネルのインストール

では、インストール作業を始めよう。最初にカーネルソースを展開する。通常、カーネルのソースは、/usr/src/linuxディレクトリに置かれる。多くの場合、このlinuxディレクトリは、linux-2.x.xといった名前のディレクトリへのシンボリックリンクとなっている。そこで、まず、このリンクを削除して、展開したソースディレクトリへリンクを張り直す。

```
# cd /usr/src/
# rm linux
# bzcat ~/linux-2.4.0.tar.bz2 | tar xvf -
# mv linux linux-2.4.0
# ln -s linux-2.4.0 linux
```

ソースがgzipで圧縮されている場合は、bzcatではなくzcatを使う。展開できたら、次の手順でカーネルコンフィグレーションを行う。

```
# cd linux
# make xconfig
```

2番目に実行する“ make xconfig ”でカーネルのコンフィグレーションを行う。コンフィグレーションとは、自分が必要とする機能をカーネルに組み込んだり、モジュールとして用意するためのカスタマイズ作業だ。例としてあげた“ make xconfig ”を実行すると、X Window Systemで動作するGUIの設定ツールが起動する（**画面1**）。コンソールで作業を行う場合は、“ make menuconfig ”とすれば、キャラクタベースの設定ツールが実行される（**画面2**）。今回は、X上の

GUIツールを例にして解説するが、どちらを利用しても設定できる内容は同じだ。このほかにも、画面制御ができない低機能な端末用に“ make config ”も用意されている。この場合、対話的に設定することになるが、使い勝手がよくないので説明は省略する。

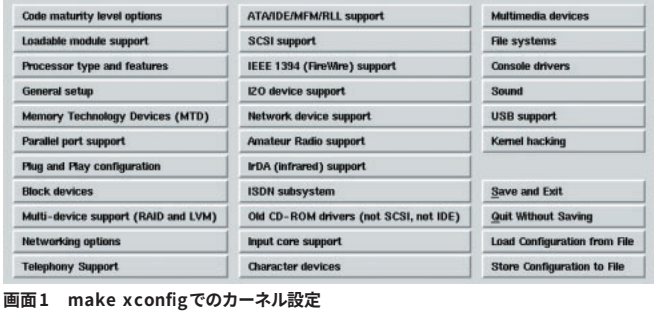
これらのツールで設定する項目は非常に多く、その内容も多岐にわたる。そのため、具体的な設定については、このあとのセクションで解説しよう。ここでは、インストール作業全体の流れを理解してほしい。

カーネルのコンフィグレーションが済むと、設定した内容は.configというテキストファイルに保存される（ドットで始まるファイル名を持つため、lsに-aオプションを付けないと表示されない）。このファイルをlessなどでのぞいてみると、項目ごとに「CONFIG_MODULES=y」のような形式で設定内容が書かれているのがわかるだろう。人間には少々わかりにくいが、カーネルのコンフィグレーションを説明する際には、単純で間違いを起こしにくいこの形式が用いられることもある。“ # ”で始まるコメント行を見れば、だいたい何を意味するかはわかるが、GUIツールに用意されたヘルプには、この形式も記載されているので参考にしてほしい。

残る作業はコンパイルとインストールだ。カーネル本体のコンパイルとインストールは次のようにして行う。

```
# make dep
# make clean
# make install
```

作業を行っているマシンの速度に依存するが、コンパイルには数分～数時間かかる。コンパイルが済めば、できあがったカーネルの圧縮イメージ（vmlinuz-2.4.0）が/bootなどのディレクトリにコピーされ、vmlinuzからリンクが張られる。同じディレクトリのSystem.mapもSystem.map-2.4.0へのリンクになる。ついで、先ほど設定したlilo.confにしたがってLILOが実行される。



画面1　make xconfigでのカーネル設定

ここまでうまくいったら、あとはモジュールのコンパイルとインストールだ。次のようにする。

```
# make modules
# make modules_install
```

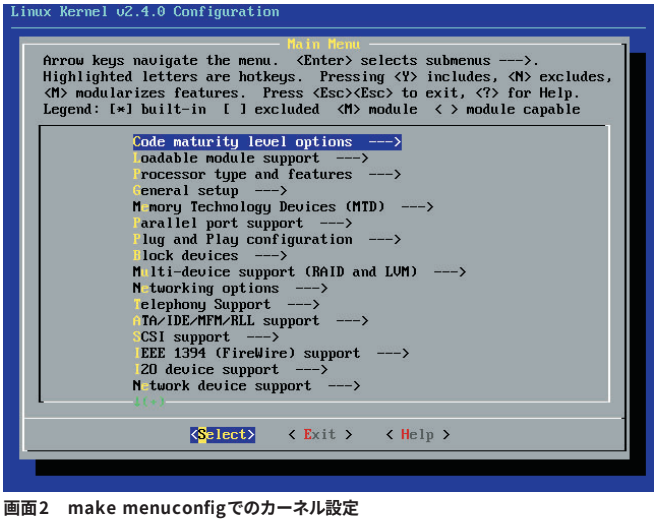
これで、/lib/modules/2.4.0ディレクトリ以下にモジュールがインストールされる。もちろん、コンフィグレーションでモジュールをいっさい利用しない設定にしていれば何もインストールされない。

以上でカーネル2.4のインストールが終了した。リブートすれば新しいカーネルでLinuxが起動するはずだ。なお、再びカーネルを構築する場合は、コンパイル済みの実行ファイルやオブジェクトファイル、設定ファイルを削除するために、次のようにしたあとで“ make xconfig ”からやり直す。

```
# make mrproper
```

コンフィグレーション内容が大きく変わらなければ、この作業を行わなくてもうまくいくかもしれないが、お勧めはしない。また、これを実行すると.configファイルも削除されてしまうので、ホームディレクトリなどにコピーしておき、“ make mrproper ”を実行してから書き戻すとよいだろう。コンフィグレーション以降は、各コマンドを“ && ”でつないで一気に実行することもできる。

```
# make dep && make clean && make install && make modules && make modules_install
```



画面2　make menuconfigでのカーネル設定

カーネルコンフィグレーションのメインメニュー

カーネルのコンフィグレーションは、前ページの画面に示したメインメニューから始まる。ここから各メニューを開いて個々の設定を行うのだ。“ make xconfig ”の場合は、項目ごとに、その機能を利用するなら“ y ”、モジュールにするなら“ m ”、利用しないなら“ n ”をマウスで選ぶ。“ make menuconfig ”では、カーソルキーで項目を移動し、Enterキーでメニューを選択する。設定はスペースキーで行う。利用するなら“ ＊ ”、モジュールにするのは“ M ”、利用しないは空白だ。

では、メインメニューから解説を始めよう。

●Code maturity level options

以降のカーネルコンフィグレーション設定で、開発中だったり、まだ完成していないコードを表示するかどうかの設定。

設定すれば、テスト段階のコードやドライバを選べるようになるが、テスト段階の機能は十分に安定しているものではない。実用に供しているマシンでは「n」と答えるべきである。

●Loadable module support

ローダブルモジュールを利用するための設定。

●Processor type and features

CPUの種類と、利用するCPUの機能を設定する。

●General setup

ネットワーク機能、PCカードや電源管理機能の利用可否、外部拡張バスの種類、実行可能なバイナリ形式などを設定する。

●Memory Technology Devices (MTD)

コンパクトフラッシュメモリや、M-SystemsのDiskOnChipなどのサポート機能を利用するかどうかを設定する。

●Parallel port support

パラレルポートを利用するかどうかを設定する。パラレルポートに接続するプリンタ、ZIPドライブなどを利用する場合は、このメニューにある「Parallel port support」、「PC-style hardware」を「y」または「m」にする。

「Use FIFO/DMA if available」を「y」にすると、パラレルポートコントローラが対応していれば高速な転送が可能になる。

「IEEE 1284 transfer modes」を「y」にすると、IEEE 1284に準拠したプリンタの状態報告機能を利用できる。

●Plug and Play configuration

プラグ＆プレイ機能を利用するかどうかを設定する。プラグ＆プレイ機能を持つISAカードをサポートするかどうかもここから設定する。

●Block devices

フロッピーディスクやRAMディスク、ループバックデバイスなどのブロックデバイスを利用するかどうかを設定する。パラレルポート接続のディスクドライブを利用する場合もここで設定するが、ATA・IDE接続、SCSI接続のブロックデバイスはそれぞれ個別のメニューが用意されている。

●Multi-device support (RAID and LVM)

ソフトウェアRAID機能、LVM（Logical Volume Manager）機能を利用するかどうかを設定する。RAIDモードは0（ストライピング）、1（ミラーリング）、4／5のほか、あるパーティションを別のパーティションに追加して容量を増やすリニアモードに対応している。

●Networking options

ネットワークに関する機能の設定を行う。

●Telephony Support

インターネット電話のための拡張カードを使うかどうかを設定する。現状ではQuicknet TechnologiesのPhoneJACKカード、Internet LineJACKカードのみがサポートされている。Webブラウジングなど、通常のインターネットアプリケーションを利用する場合は設定しなくてよい。

●ATA/IDE/MFM/RLL support

これらの方式で接続されるブロックデバイス（ハードディスクなど）のサポート機能を設定する。接続する機器やIDE

コントローラの種類などを細かく設定可能だ。

●SCSI support

利用する機器の種類など、SCSI機器のサポート機能を設定する。SCSIコントローラの種類は、サブメニューから選択する。また、ATAPI接続のCD-RWドライブをSCSIエミュレーションで使う場合はSCSI supportを有効にしておく必要がある。

●IEEE 1394 (FireWire) support

IEEE 1394インターフェイスを利用するならここでコントローラの種類を選ぶ。

●I2O device support

I2O（Intelligent Input/Output）対応機器のサポート。

I2Oは、CPUの代わりにI/Oプロセッサ（IOP）がデバイスからの割り込みやデータを監視／制御することでCPU負荷の軽減とスループットの向上を図るデバイスインターフェイス規格だ。OS側にOSM（Operating system Service Module）というドライバを、デバイス側にHDM（Hardware Device Module）というドライバを持たせ、これらの間を策定したプロトコルで結ぶ。

●Network device support

ネットワーク機器のサポート。

イーサネットカード、PPP、無線LANカードなどのネットワークインターフェイスのドライバを選ぶ。

●Amateur Radio support

アマチュア無線でのパケット通信への対応機能を使うかどうか（無線LANには関係ない）。

●IrDA (infrared) support

赤外線通信機能のサポート。

●ISDN subsystem

ISDN通信カードのサポート。日本国内で利用できるカードはないので通常は設定の必要はない。また、ISDNルータやターミナルアダプタを利用する場合もここでの設定は不要だ。

●Old CD-ROM drivers (not SCSI, not IDE)

SCSIやIDE（ATAPI）以外の独自インターフェイスで接続するCD-ROMドライブのサポート。数年前までは、独自のCD-ROMインターフェイスに専用のCD-ROMドライブを接続することが珍しくなかった。このようなCD-ROMドライブを使うならここで設定する。

●Input core support

キーボード、マウス、ジョイスティックなどの入力機器のサポート。これらの機器をUSB接続して使う場合はここで設定する。タブレットやデジタイザを使う場合は、X Window Systemで使う画面解像度も設定しよう。

●Character devices

コンソールデバイス、パラレルポートプリンタ、マウス、ジョイスティックなどの設定。ジョイスティックを使用する場合は、「Input core support」も有効にする必要がある。

●Multimedia devices

オーディオ・ビデオキャプチャ機器とラジオカードのサポート。これらの機器を利用するならここで機種を選択する。

●File systems

サポートするファイルシステムの選択。

Linuxから利用するファイルシステムやパーティションタイプの選択、ファイル名として利用する言語の選択を行う。

●Console drivers

コンソールドライバの選択。通常のVGAテキストコンソールのほか、高解像度で利用可能なフレームバッファコンソール機能を使うことも可能。

●Sound

サウンドカードの種類を選択する。

●USB support

USB接続機器のサポート。利用する機器のドライバのほか、USBコントローラの種類もここで選択する。

●Kernel hacking

SysRqキー（Alt＋PrintScreenキー）とほかのキーを組み合わせることで、強制リブートやレジスタのダンプなどの機能を有効にする（詳細はDocumentation/sysrq.txtを参照のこと）。主にカーネルのデバッグに利用する。通常は「n」を選ぶ。

主なメニューの設定項目

カーネルコンフィグレーションの設定項目は膨大な数にのぼるため、多くの人が設定しなければならないと思われるメニューを選び、その内容を解説する。今回取り上げるのは次のメニューだ。

- Loadable module support
- Processor type and features
- General setup
- Block devices
- Networking options
- ATA／IDE／MFM／RLL support
- SCSI support
- Network device support
- Character devices
- File systems
- USB support

Loadable module support メニュー

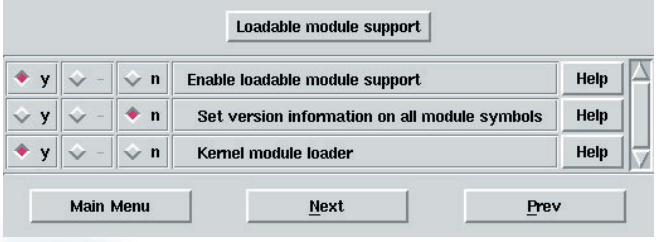
モジュールを利用するならここで設定をする。

●Enable loadable module support

モジュールを利用する場合は“ y ”にする。

●Set version information on all module symbols

通常は、カーネルを新しくしたらモジュールもコンパイルし直す必要がある。ここを“ y ”にすると、古いカーネルでコンパイルしたモジュールを安全に利用できるようになる。たいていの環境では“ n ”でよい。



画面3　Loadable module supportメニュー

●Kernel module loader

ここで“ y ”を選ぶと、カーネルが必要とするモジュールを自動的にロードするようになる。通常は“ y ”を選ぼう。

Processor type and features メニュー

CPUの種類や、CPU固有の機能、SMPの利用可否などを設定する。

●Processor family

カーネルのコンパイル時に、ここで選んだCPUに合わせた最適化が行われる。また、CPUによっては、パフォーマンスを向上させるための機能を有効にする。たとえば、Pentium IIIやCoppermineコアのCeleronは、“ Pentium-III ”を選ぶことで“ fast FPU save and restore ”機能が有効になる。CPUの種類を間違えると、最悪の場合起動しなくなるので気をつけよう。

●Toshiba Laptop support

ここで“ y ”か“ m ”を選ぶと、東芝製のポータブルPC用のSystem Management Modeドライバが組み込まれる。

●/dev/cpu/microcode — Intel IA32 CPU microcode support

IntelのP6コアを搭載したCPU（Pentium Pro、Pentium II、Pentium III、Pentium 4、Celeron、Pentium Xeonなど）にはマイクロコードを更新する機能がある。この機能を利用する場合はここを“ y ”か“ m ”にする。実際に利用するには、Intel Microcode Update Utility for Linuxが別途必要だ（CD-ROM Disc2に収録）。他社製CPUや、Pentium以前のIntel製CPUではこの機能は利用できない。

●/dev/cpu/*/msr — Model-specific register support

ここで“ y ”または“ m ”を指定すると、指定したCPUのMSRs（Model Specific Registers）をキャラクタデバイスとしてアクセスできるようになる。MSRsは、そのCPUに特

有の拡張機能を設定するためのレジスタだ。

●/dev/cpu/*/cpuid — CPU information support

ここで“y”または“m”を指定すると、CPUID命令の結果をキャラクタデバイスとしてアクセスできるようになる。

●High Memory Support

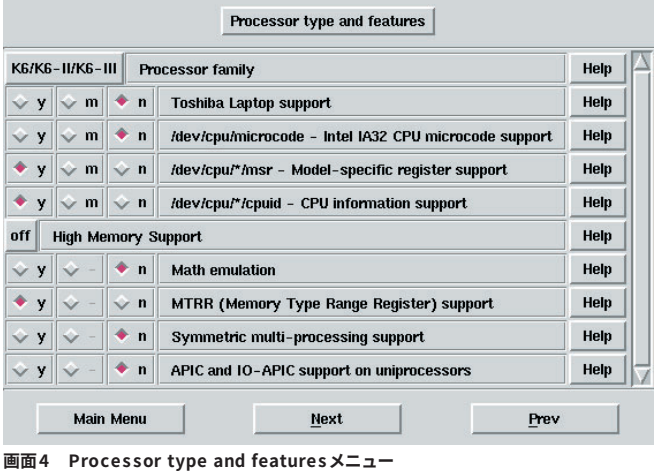
カーネル2.4では、最大64Gバイトのメモリ空間を扱うことができる。Linuxマシンに搭載している物理メモリ量が1Gバイト未満なら“off”、1G〜4Gバイトであれば“4GB”を選ぶ。Intel PAE（Physical Address Extension）に対応するCPUでは、“64GB”を選ぶことで、仮想的に64Gバイトメモリ空間を使うことができる。

●Math emulation

“y”にすると、CPU内蔵の浮動小数点演算ユニット（あるいは数値演算コプロセッサ）の代わりにソフトウェアで浮動小数点演算を行う。CPUに浮動小数点演算ユニットが内蔵されておらず、数値演算コプロセッサも搭載していない場合は“y”にする。386、486SX、486SLC、486DLCといった旧式のCPUを使っていなければ“n”でよい。

●MTRR（Memory Type Range Register）support

IntelのP6コアCPUやAMD K6以降には、MTRRという特殊なレジスタがあり、メモリアクセスの方法を制御して、グラフィックなどの性能を向上させることができる。Cyrixの6x86以降のCPU、WinChipシリーズにも似たような機構がある。“y”にするとこれらの機能が有効になる。XFree86 4.xや、一部のフレームバッファコンソールドライバはこの機能を使う。迷ったら“y”にしておこう。



画面4　Processor type and featuresメニュー

●Symmetric multi-processing support

SMP（対称型マルチプロセッシング）環境で利用するときは“y”にする。

●APIC and IO-APIC support on uniprocessors

APIC（Advanced Programmable Interrupt Controller）は、ハードウェア割り込み要求をCPUに伝えるための機構で、主に複数のCPUを持つシステムで使われている。シングルCPUのマシンでも、APICを搭載したシステムを使っているなら、ここを“y”にするとAPICを利用できる。APICがないマシンでこのオプションを有効にしても、速度の低下は起きない。

General setupメニュー

このメニューには、雑多な設定項目が集められている。順に説明しよう。

●Networking support

Linuxのプログラムには、内部でネットワーク機能を利用しているものもある。たとえ、スタンドアローン環境で使う場合でも“y”にしておこう。

●SGI Visual Workstation support

SGIのx86マシン「Visual Workstation」では“y”、一般的なPCでは“n”にする。

●PCI support

PCIバスを持たない旧式なマシンでは“n”にしてもよい。ほとんどの場合は“y”でよいだろう。

●PCI access mode

PCI接続のデバイスを見つけるためのアクセス方法を指定する。“Direct”を選ぶと、カーネルがPCIデバイスに直接アクセスする。“BIOS”を選ぶとシステムBIOSを利用してデバイスを検知する。“Any”にすると、まず、直接アクセスし、失敗するとBIOSを利用する。通常は“Any”でよい。

●PCI device name database

PCIデバイスはデバイスIDという番号を持っており、この番号からメーカーやデバイスの種類が判別できる。Linuxは

カーネル内にIDと機種名のデータベースを持っているが、“ n ”にするとデータベースを放棄する。その結果、カーネルの占めるメモリ容量を80Kバイトほど小さくできる。通常は“ y ”にしておこう。

●EISA support

ISAバスを32ビットに拡張したEISAバスを使うなら“ y ”にする。現在、このバスをサポートしているPCはほとんどないので、たいていの場合“ n ”でよい。

●MCA support

IBMのPS/2で採用されているMCA拡張バスを使うなら“ y ”にする。現在ではMCAバスを持つマシンは滅多にないので、ほとんどの場合は“ n ”でよい。

●Support for hot-pluggable devices

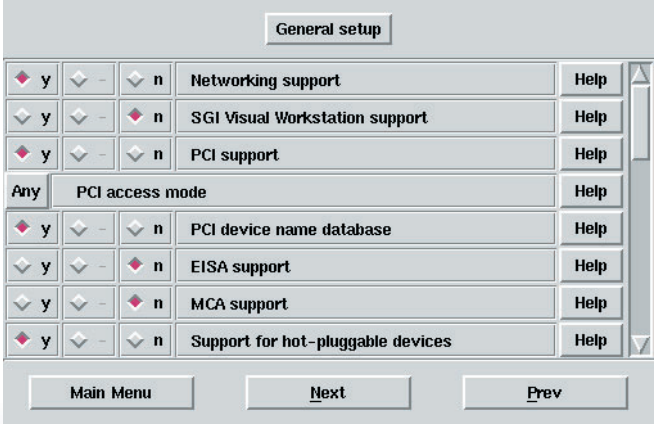
PCカード（PCMCIAやCardBus）やUSBなど、ホットプラグ（活線挿抜）可能なデバイスを利用するなら“ y ”にしよう。

●PCMCIA／CardBus support

PCカードを利用する場合はこのサブメニューで設定を行う。PCMCIAは16ビット、CardBusは32ビットのバスで接続されるデバイスだ。

●System V IPC

IPC（Inter Process Communication）とは、プログラム同士がデータをやりとりするための仕組みで、日本語ではプロセス間通信と呼ばれる。この機能を必須とするプログラムは数多くあるので、“ y ”を選ぶ。



画面5　General setupメニュー

●BSD Process Accounting

“ y ”を選ぶと、ユーザープロセスが終了するときに、プロセスの生成時刻、所有者、コマンド名、メモリの使い方や制御端末などの情報をカーネルが記録するようになる（ただし、ユーザープロセスがカーネルに情報を記録するよう指示した場合に限る）。“ y ”にしておこう。

●Sysctl support

sysctlインターフェイスは、カーネルの再構築やシステムの再起動をせずに、カーネルのパラメータや変数を動的に変更するための仕組みだ。/procファイルシステムとあわせて使うと便利なので“ y ”にしておく。

●Kernel core（/proc/kcore）format

/procファイルシステムが有効になっていれば、/proc/kcoreを通じてカーネルコアのイメージが見えるようになる（カーネルハッカー以外はまず使わない機能だ）。ここでは、このイメージがELFとa.outのうち、どちらのバイナリ形式で見えるようにするかを設定する。デフォルトの“ ELF ”のままにしておけばよいだろう。

●Kernel support for a.out binaries

a.out（Assembler.OUTput）は、昔のUNIXやLinuxで標準的に使われていたバイナリフォーマットだ。現在ではあまり使われないが、古いLinuxプログラムをバイナリで入手した場合には必要になるかもしれない。“ m ”を選び、モジュールにしておくことをお勧めする。

●Kernel support for ELF binaries

ELF（Executable and Linkable Format）は、現在のUNIXやLinuxで主流のバイナリフォーマットだ（Linux以外のUNIXプログラムが動作するという意味ではないので注意）。ここは必ず“ y ”にしなければならない。

●Kernel support for MISC binaries

この機能を有効にすると、Java、Pythonなどのプログラムを直接実行できるようになる（実行に必要なインタプリタなどが自動的に起動され、プログラムファイルが渡される）。便利な機能なので“ y ”にしておこう。

●Power Management support

電源管理機能を使う場合はここで“ y ”を選び、これ以降

の各項目で必要な機能を選ぶ。ACPIサポートはまだ実験段階なので気をつけよう。

Block devicesメニュー

ATA／IDE、SCSI以外の接続方式のブロックデバイスの設定を行う。

●Normal PC floppy disk support

通常のフロッピーディスクドライブを使うなら“y”にする。フロッピーレスマシンでは“n”でよい。

●XT hard disk support

IBM PC/XTで使われていた古いハードディスクコントローラを使うかどうか。現在のPCには使われていないので“n”でよい。

●Parallel port IDE device support

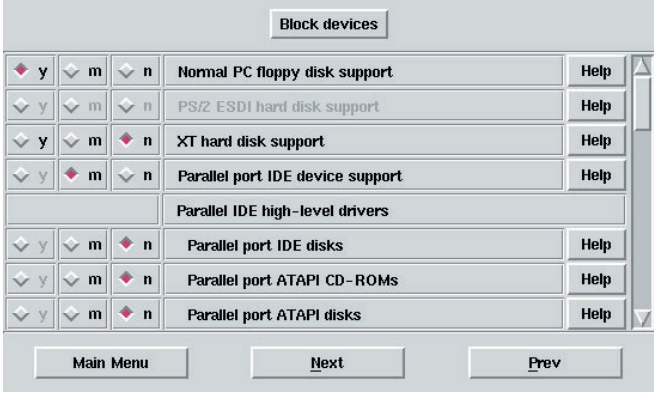
パラレルポートに接続するIDEデバイスを使うなら有効にし、以下の項目からデバイスの種類、プロトコルを選択する。「Parallel port support」を有効にしておかないと設定できないので注意。

●Compaq SMART2 support

●Compaq CISS Array support

● Mylex DAC960／DAC1100 PCI RAID Controller support

これらのRAIDコントローラを使うなら“y”または“m”にする。



画面6　Block devicesメニュー

●Loopback device support

ループバックデバイスを使うと、CD-ROMのISOイメージファイルをマウントして、仮想CD-ROMドライブのように使うこともできる。モジュールがお勧めだ。

●Network block device support

この機能を使うとネットワークの先にあるブロックデバイスを利用できる。通常は必要ないので“n”でよい。

●RAM disk support

メインメモリの一部をディスクとして利用する機能だ。あまり使うことはないだろう。

Networking optionsメニュー

ネットワークに関する設定項目はたくさんあるが、通常のサーバやデスクトップマシンとして利用する分には、必要な機能はそう多くない。

●Packet socket

カーネル内部に実装された中間プロトコルを介さずに、パケットプロトコルで直接ネットワークデバイスとやりとりするプログラムを使う場合は有効にする。たとえば、tcpdumpを使うならこの機能が必要だ。よくわからなければ“y”にしておこう。

●Packet socket：mmapped IO

有効にすると、パケットプロトコルを高速に処理する機能が組み込まれる。“n”にするのが安全だ。

●Kernel／User netlink socket

このドライバは、ユーザープロセスがカーネルの一部、あるいはモジュールと相互にやりとりできるようにする。通常は“y”にする。

●Routing messages

ルーティング情報を読み出すキャラクタデバイスを有効にする。通常は不要なので“n”でよい。

●Netlink device emulation

この項目は互換性を維持するために設けられており、近い

将来削除される。現状では“y”にしておく。

●Network packet filtering

カーネル2.4から用意されたnetfilter機能を使うかどうか。パケットフィルタやNAT（IPマスカレードを含む）の機能を使うなら“y”する。そのうえで、IP：Netfilter Configurationサブメニューで必要な機能を設定する。インターネットに直接つながないなら“n”でよいだろう。

netfilter機能を使う場合は、メニューの下のほうにある「Fast switching」を“n”にすること。さもないと、netfilter機能が迂回され機能しなくなる。

●Network packet filtering debugging

netfilter機能のデバッグをする場合は“y”にする。通常は“n”でよい

●Socket Filtering

ユーザプロセスがソケットフィルタを使えるようにするかどうか。通常は“n”でよい。

●Unix domain sockets

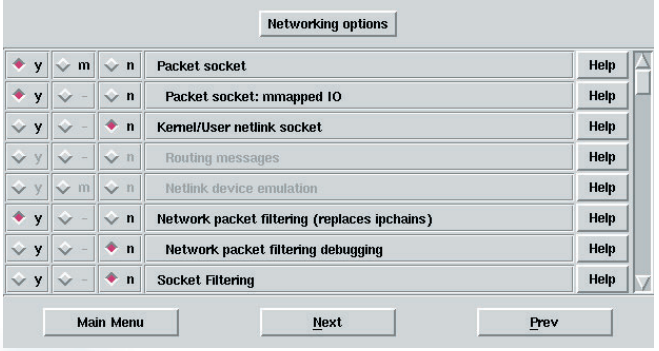
UNIXドメインソケットを使うかどうか。たとえ、ネットワークにつながっていなくても、X Window Systemやsyslogなど、重要なプログラムがソケットを利用するので、必ず“y”にする。

●TCP/IP networking

TCP/IPを使うかどうか。迷うことなく“y”にしよう。

●IP：multicasting

IPマルチキャストはインターネットなどで複数の相手に対してストリーミングデータを配信するのに使われる。多くの人にとっては“n”を選ぶのが安全だ。



画面7　Networking optionsメニュー

●IP：advanced router

“y”を選ぶと、細かなルーティング設定ができる。Linuxマシンをルータ専用機として使う場合以外は“n”でよい。

●IP：kernel level autoconfiguration

ネットワークブートが必要なディスクレスマシン以外は“n”にする。

●IP：tunneling

IP中にIPをカプセル化するための特殊なドライバ。“n”にする。

●IP：GRE tunnels over IP

GRE（Generic Routing Encapsulation）を使うとIPv6（IPv4も）をIPv4にカプセル化して、既存のIPv4環境に流すことができる。この機能が必要なければ“n”にする。

●IP：ARP daemon support（EXPERIMENTAL）

IPアドレスとMACアドレスの対応データの処理をARPデーモンを介して行うなら“y”にする。この場合、ARPデーモンが別途必要になる。通常は“n”でよい。

●IP：TCP Explicit Congestion Notification support

Explicit Congestion Notification（ECN）機能を使うかどうか。通常は“n”でよい。

●IP：TCP syncookie support

SYN flood攻撃（DoS攻撃の一種）の防御手段であるTCP syncookie機能を使う。この機能を有効にするには、General setupメニューの「Sysctl support」と、File systemsメニューの「/proc file system support」を有効にしたうえで、起動後に“echo 1 >/proc/sys/net/ipv4/tcp_syncookies”を実行する。“y”を選んでおこう。

●IP：Netfilter Configuration

「Network packet filtering」を有効にしている場合、netfilterの機能はこのサブメニューで設定する。パケットフィルタに必要な項目や、NATに必要な項目が含まれているので、使う機能を有効にする。よくわからない場合は、すべてモジュールにして、利用する際に適宜組み込むとよいだろう。

●The IPv6 protocol（EXPERIMENTAL）

IPv6を利用する場合は有効にする（ただし実験的な実装）。現在、IPv6はまだ普及しているとはいえない状態だ。たいていの環境では“n”でよいだろう。

●Kernel httpd acceleration（EXPERIMENTAL）

これも実験的な実装だが、カーネル内に簡単なWebサーバ機能を組み込む。Apacheなど、ユーザーモードで動作するhttpdに比べるとオーバーヘッドが小さいので高いパフォーマンスが期待できる。また、通常のhttpdと協調した動作をさせることも可能だ。ほとんどの方は“n”を選んでおけばよいが、実験してみたいなら“y”を選ぶ。

●Asynchronous Transfer Mode（ATM）（EXPERIMENTAL）

ATM（非同期転送モード）への対応。ほとんどの場合“n”でよい。

●The IPX protocol

NetWareで使われているプロトコルだ。NetWareサーバにつながないなら“n”でよい。

●Appletalk protocol support

Macintoshで使われているプロトコル。netatalkを使ってMacintoshとつなぐなら有効にする。

●DECnet Support

DEC（現Compaq）のDECnet対応機器につなぐ場合は有効にする。

●802.1d Ethernet Bridging

Linuxマシンをイーサネットブリッジにするなら有効にする。通常は“n”でよい。

●CCITT X.25 Packet Layer（EXPERIMENTAL）

X.25のパケット層を使うなら有効にする。通常は“n”でよい。

●LAPB Data Link Driver（EXPERIMENTAL）

X.25のデータリンク層を使うなら有効にする。通常は“n”でよい。

●802.2 LLC（EXPERIMENTAL）

イーサネット上でX.25を使うための論理リンク層を使うなら有効にする。通常は“n”でよい。

●Frame Diverter（EXPERIMENTAL）

ネットワークに流れるパケットをLinuxマシンに経由させる機能。ブリッジとして構成したマシンでこの機能を使うと、透過プロキシが実現可能だ。ほとんどの方はこの機能を使うことはないだろう。“n”でよい。

●Acorn Econet／AUN protocols（EXPERIMENTAL）

ずいぶん昔に使われていたAcornのEconetを使う場合は有効にする。“n”にする。

●WAN router

LinuxマシンをWANルータにする場合は有効にする。まず間違いなく“n”でよい。

●Fast switching

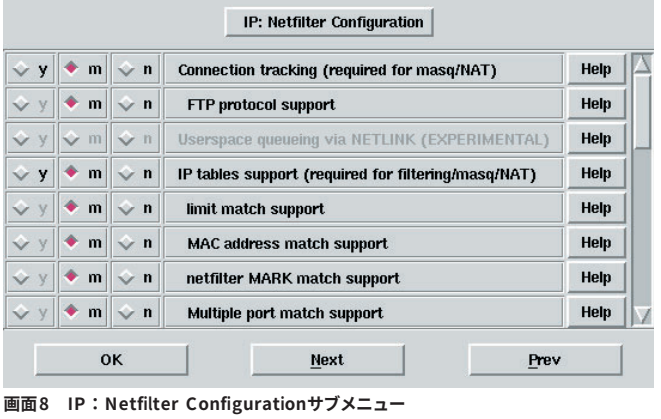
複数のネットワークインターフェイス間で高速にデータを転送する機能だが、わずかな種類のカードしかサポートしないうえに、netfilter機能と併用できない。“n”にしておこう。

●Forwarding between high speed interfaces

ネットワークカードのハードフロー機能を有効にする。通常は“n”でかまわない。

●QoS and／or fair queueing（EXPERIMENTAL）

カーネルから送り出されるパケットが複数ある時に、その順序を決めるアルゴリズムを選ぶことができる。実験のために設けられているサブメニューなので、“n”にしておく。



画面8　IP：Netfilter Configurationサブメニュー

ATA／IDE／MFM／RLL supportメニュー

マシンの構成をオールSCSIにしているのでもなければ、このメニューで設定をしなければならない。IDEコントローラチップごとの設定を行えば、ディスクパフォーマンスの向上も期待できる。

●ATA／IDE／MFM／RLL support

ATA／IDE接続のハードディスク、CD-ROMドライブなどを使うなら有効にする。オールSCSIマシンでなければ“y”にしよう。

●IDE, ATA and ATAPI Block devicesサブメニュー

以下の設定はこのサブメニューから行う。今回は主要な項目について解説する。

● Enhanced IDE／ MFM／ RLL disk／ cdrom／ tape／ floppy support

通常は“y”にする。

●Use old disk-only driver on primary interface

1番目のコントローラに古いドライバを使うかどうか。通常は“n”にする。

●Include IDE／ATA-2 DISK support

通常は“y”にする。

●Use multi-mode by default

通常は“n”にする。起動時に、この項目のヘルプにあるようなエラーが起きるなら“y”にする。

●Include IDE／ATAPI CDROM support
●Include IDE／ATAPI TAPE support
●Include IDE／ATAPI FLOPPY support

これらの機器を使うなら有効にする。ATAPI接続のCD-ROMドライブを使っている方は多いだろう。

●SCSI emulation support

ATA／IDE接続した機器をSCSI機器に見せかける機能。ATAPI接続のCD-R／RWドライブを使うときなどに使用する。利用するためには、SCSI supportメニューの「SCSI support」、「SCSI generic support」も有効にする。

●Generic PCI IDE chipset support

古いマシンでなければ“y”にする。

●Sharing PCI IDE interrupts support

IRQの共有を許すなら“y”にする。“n”にしたほうが安全だが、“y”でも動作することが多い。

●Generic PCI bus-master DMA support

“y”を選ぶとバスマスタDMA転送が使えるようになる。起動後にhdparmコマンドで設定するか、このあとにある「Use PCI DMA by default when available」も“y”にするとバスマスタDMA転送が有効になる。たいていの場合は“y”でよい。

●Boot off-board chipsets first support

拡張IDEカードにつないだディスクから起動する場合は“y”にする。

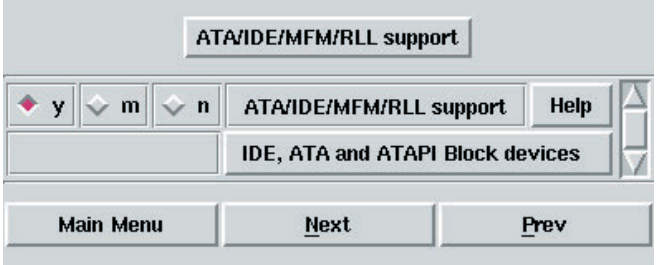
●Use PCI DMA by default when available

起動時にバスマスタDMA転送を使うようにする。ほとんどの場合、“y”で問題ないと思われる。

これ以降は、IDEコントローラの種類を選ぶ項目だ。使用するLinuxマシンに搭載されているものを選ぼう。

SCSI supportメニュー

SCSIを利用するためには、このメニューで接続機器のドライバや、SCSIホストコントローラのドライバを選ぶ。



画面9　ATA／IDE／MFM／RLL supportメニュー

●SCSI support

SCSI接続機器、パラレルポート接続のZIPドライブ、3wareのIDE RAIDカードを使う場合は有効にする。

●SCSI disk support

SCSIハードディスクを使う場合は有効にする。特に、起動ディスクとして使うなら“y”を選ぼう。

●Maximum number of SCSI disks that can be loaded as modules

Linuxの起動後にモジュールとして組み込まれるドライバのためのテーブル数を設定する。よほど多くのディスクをつながない限り、デフォルト値のままでよい。

●SCSI tape support

●SCSI OnStream SC-x0 tape support

●SCSI CD-ROM support

これらの機器を使うなら該当するものを有効にする。

●SCSI generic support

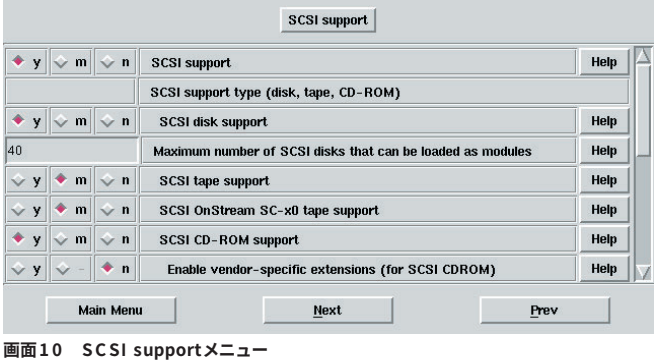
スキャナなど、ディスクやテープドライブ以外のSCSI機器を使うにはこの項目を有効にする。また、ATA／IDE／MFM／RLL supportメニューで設定する「SCSI emulation support」機能を使う場合も、この項目を有効にする。

●Enable extra checks in new queueing code

SCSIキューイングの新しいコードをチェックする機構を使うかどうか。このコードのデバッグの役に立つが、パフォーマンスは低下する。通常は“n”にする。

●Probe all LUNs on each SCSI device

多連装CD-ROMドライブのように、LUN（Logical Unit Number）を使うデバイスがあれば“y”にする。



画面10　SCSI supportメニュー

●Verbose SCSI error reporting

SCSIに関するエラーを詳しく表示する。トラブルが起きない限り“n”でよい。

●SCSI logging facility

SCSI関連プログラムのデバッグに便利なログ機能を使うかどうか。これもトラブルが起きない限り“n”でよい。

●SCSI low-level drivers

このサブメニューでSCSIホストアダプタに搭載されたSCSIコントローラチップの種類を選ぶ。

Network device supportメニュー

イーサネットカードやPPPドライバなど、ネットワークデバイスのドライバを設定するメニューだ。

●ARCnet devices

ARCnetのネットワークカードを使う場合はこのサブメニューで設定する（まずないだろうが）。

●Dummy net driver support

送られたデータをすべて破棄するダミーのネットワークドライバ。このドライバを組み込んでもカーネルの大きさは変わらないので“y”にしておこう。

●Bonding driver support

イーサネットによる複数の接続経路を束ねて、高速な接続経路とする機能。CiscoはEtherchannelと呼び、SunはTrunkingと呼んでいるが、LinuxではBondingと呼ぶ。

●EQL（serial line load balancing）support

複数のシリアル回線接続を束ねて、高速な接続経路として使うための機能。

●Universal TUN／TAP device driver support

パケットをユーザーレベルプログラムが送受信できるようにするためのドライバ。ユーザーレベルで動作するPPPドライバなどで使うことがある。

●General Instruments Surfboard 1000

General Instrumentの内蔵用ケーブルモデムSurfboard 1000のドライバ。日本でこのケーブルモデムを使うことはないと思われる。したがって“n”にする。

●Ethernet（10 or 100Mbit）

10BASE／100BASEのイーサネットカードを使う場合はこのサブメニューでドライバを選ぶ。カーネル2.4では、多くのドライバが書き直され、安定度が増したものも多い。

●Ethernet（1000 Mbit）

ギガビットイーサネットカードのドライバを選ぶサブメニュー。

●FDDI driver support

FDDIカードを使う場合は“y”にする。

●HIPPI driver support（EXPERIMENTAL）

特殊な用途で使われるHIPPI（HIgh Performance Parallel Interface）カードを使う場合は“y”にする。

●PLIP（parallel port）support

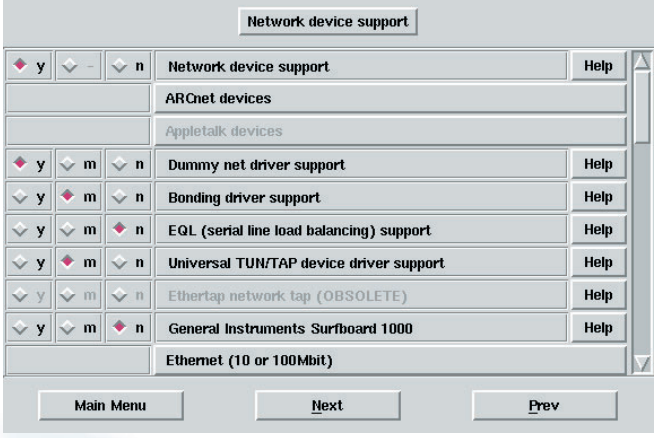
パラレルポートで通信するPLIPを使う場合は有効にする。

●PPP（point-to-point protocol）support

PPPを使う場合は有効にする。

●SLIP（serial line）support

SLIPを使う場合は有効にする。インターネットプロバイダの大部分はPPPによる接続を採用しており、SLIPを使うことは稀になった。



画面11　Network device supportメニュー

●Wireless LAN（non-hamradio）

無線LANカードのドライバを選ぶサブメニュー。日本で使われている無線LAN機器はほとんどがPCカードなので、このサブメニューは関係しない。“n”でよい。

●Token Ring devices

トークンリングカードを利用する場合はこのサブメニューでドライバを選ぶ。

●Fibre Channel driver support

ファイバチャネルカードを使う場合は“y”にする。

●Red Creek Hardware VPN（EXPERIMENTAL）

Red CreekのVPN（Virtual Private Networking）機器を使うなら“y”にする。

●Traffic Shaper（EXPERIMENTAL）

ネットワークデバイスから送信されるデータの速度を制限する仮想デバイス。まだ実験中なので、利用する場合はDocumentation/networking/shaper.txtをよく読もう。

●Wan interfaces

Wan（Wide Area Network）インターフェイスカードを利用する場合はこのサブメニューからカードを選ぶ。

Character devicesメニュー

コンソールデバイス、パラレルポートプリンタ、マウス、ジョイスティックなどのキャラクタデバイスの設定を行う。

●Virtual terminal

仮想コンソールと呼ばれる機能で、Alt＋ファンクションキー（X環境からはCtrl＋Alt＋ファンクションキー）で複数の端末画面を切り替えて利用できるようになる。“y”にしよう。

●Support for console on virtual terminal

カーネルメッセージや警告メッセージを表示するコンソール（システムコンソール）を仮想コンソールで使えるようにする。必ず“y”にする。

●Standard／generic（8250／16550 and compatible UARTs）serial support

標準的なシリアルポートのサポート。シリアルポートを利用しない場合のみ“n”にする。

●Non-standard serial port support

拡張シリアルカードを利用する場合は“y”にして、使用する機種を選択する。

●Unix98 PTY support

疑似端末名にUnix98で制定された名前を使えるようにする。通常は“y”にする。

●Maximum number of Unix98 PTYs in use（0-2048）

デフォルト値のままにしておく。

●Parallel printer support

パラレルポートに接続するプリンタを使うなら有効にする。

●Support for user-space parallel port device drivers

パラレルポートを自分で制御するユーザーレベルプログラムを利用する場合は有効にする。通常は“n”でよい。

●I2C support

Philipsが提唱したシリアルバスプロトコルI2C（Inter-Integrated Circuit）を利用するなら有効にする。使うことはほとんどないだろう。

●Mice

使用するマウスの種類を選ぶ。PS/2マウスを使うならここでの設定が必要だ。

●Joysticks

ジョイスティックを利用するならこのサブメニューで種類を選ぶ。Input core supportメニューで「Input core support」、「Joystick support」も有効にすること。

●QIC-02 tape support

SCSI接続でないテープドライブを使うなら有効にする。

●Watchdog Cards

システムのハングアップを監視するウォッチドッグ（番犬）カードや、ソフトウェアによる監視機能を使う場合はこのサブメニューで設定する。

●Intel i8x0 Random Number Generator support

Intelのチップセットi8xxシリーズが持つ乱数発生機能を利用する場合は有効にする。

●/dev/nvram support

BIOSの設定を保存しているCMOSメモリの内容を読み書きする/dev/nvramデバイスを利用する。CMOSメモリを書き換えるのは非常に危険なので、通常は“n”にする。

●Enhanced Real Time Clock Support

マザーボード上にある時計モジュール（RTC）にアクセスするデバイス/dev/rtcを使うかどうか。SMPマシンでは、ここを“y”にして、/dev/rtcデバイスを作成するとよい。

●Double Talk PC internal speech card support

RC Systemsの音声合成カードDoubleTalk PCを使う場合は有効にする。

●Siemens R3964 line discipline

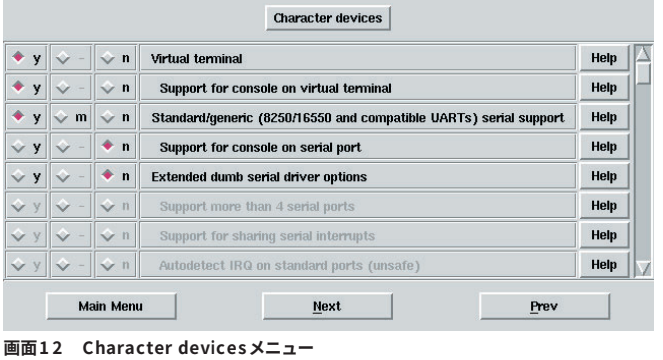
Siemens R3964による同期通信を行う特殊なデバイスを使うなら有効にする。

●Applicom intelligent fieldbus card support

intelligent fieldbusカードを使うなら有効にする。

●Ftape, the floppy tape device driver

フロッピーディスクインターフェイスに接続するテープドライブを使うなら、このサブメニューから機種を選ぶ。



画面12　Character devicesメニュー

●/dev/agpgart（AGP Support）

AGPバス用ドライバを使うなら有効にして、チップセットを選択する。

●Direct Rendering Manager（XFree86 DRI support）

XFree86 4.xで標準サポートされたDRI（Direct Rendering Infrastructure）をサポートするカーネル側の仕組みDRMを組み込むかどうか。有効にする場合は、グラフィックチップを選択する。

File systemsメニュー

Linuxは、さまざまなファイルシステムに対応している。ファイルシステムには物理的なファイルシステムと仮想的なファイルシステムがある。

●Quota support

ユーザーごとにディスクの使用量を制限する機能をクォータ、あるいはディスククォータという。この機能を利用するなら“y”にする。現状では、Linuxで標準となっているext2ファイルシステムでしか機能しない（Reiser FSにはクォータを使うためのパッチがあるようだ）。

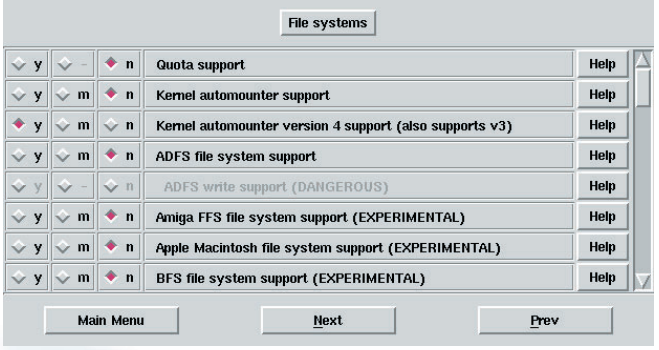
●Kernel automounter support

●Kernel automounter version 4 support

リモートファイルシステムを自動マウントする機能。これを利用するには、Network File Systemsサブメニューにある「NFS file system support」も有効にする。

●ADFS file system support

●Amiga FFS file system support（EXPERIMENTAL）



画面13　File systemsメニュー

●Apple Macintosh file system support（EXPERIMENTAL）

●BFS file system support（EXPERIMENTAL）

使われる場面はあまりないだろうが、これらのファイルシステムをマウントして読み書きしたい場合は有効にする。

●DOS FAT fs support

DOSやWindowsで使われるFATファイルシステムをマウントして読み書きするなら、ここを有効にして、この下にある「MSDOS fs support」、もしくは「VFAT（Windows-95） fs support」も有効にする。

●MSDOS fs support

MS-DOSで使われているファイルシステムのサポート（ファイル名は8.3形式）。

●UMSDOS：Unix-like file system on top of standard MSDOS fs

MS-DOSファイルシステム上で長いファイル名を利用できるように拡張したファイルシステムのサポート。現在ではほとんど使わない。

●VFAT（Windows-95）fs support

Windows 95で導入された、長いファイル名を使えるFATファイルシステムのサポート。

●EFS file system support（read only）（EXPERIMENTAL）

SGIの古いマシン（IRISなど）で使われていたファイルシステムのサポート。

●Compressed ROM file system support

組み込み用途のために作られたCramFSのサポート。

●Simple RAM-based file system support

必要に応じて使用するメモリ量を変化させるRAMベースのファイルシステム。

●ISO 9660 CDROM file system support

CD-ROMで標準的に使われているISO 9660フォーマットのサポート。RockRidge拡張にも対応している。CD-ROMドライブのあるPCでは必ず“y”にしよう。

●Microsoft Joliet CDROM extensions

CD-ROMで長いファイル名を使うためのJoliet拡張のサポート。主にWindowsで使われているファイルシステムだ。

●Minix fs support

MINIXのファイルシステムのサポート。現在ではほとんど使われない。

●NTFS file system support（read only）

Windows NTのファイルシステムNTFSのサポート。現状では書き込みはサポートしていない。

●OS/2 HPFS file system support

OS/2で使われているファイルシステムHPFS（High Performance File System）のサポート。カーネル2.4から書き込みもできるようになった。

●/proc file system support

Linuxシステムの状態を知るのに便利な仮想ファイルシステムのサポート。仮想ファイルシステムは、ファイルに見えるが、ディスクにファイルを作るわけではない。便利な機能なので“y”にしよう。

●/dev file system support（EXPERIMENTAL）

カーネル2.4で新たに用意されたのがこのDevfsだ。機能については、37ページからの速報記事を参照してほしい。

●/dev/pts file system for Unix98 PTYs

Unix98 PTYsのための仮想ファイルシステム。Character devicesメニューの「Unix98 PTY support」を“y”にしたなら、ここも“y”にする必要がある。

● QNX4 file system support（read only）（EXPERIMENTAL）

オペレーティングシステムQNX 4で使われているファイルシステムのサポート。

●ROM file system support

読み出し専用の小さなファイルシステム。起動用RAMディスクイメージ（initrd）を使ってシステムを起動する場合は必要になる。

●Second extended fs support

Linuxで標準的に使われているファイルシステム。必ず“y”にする。

●System V and Coherent file system support（read only）

SCO UNIX、Xenix、Coherentなどの商用UNIXで使われるファイルシステムの読み出しサポート。

●UDF file system support（read only）

DVDなどで使われるUDFファイルシステムの読み出しサポート。

●UFS file system support（read only）

BSDやSunOSなどで標準的に使われているファイルシステムの読み出しサポート。

●Network File Systems

NFSサーバ、あるいはクライアントとして使うにはこのサブメニューで設定する。また、Codaサポート、Windowsのディスクをマウントする SMBサポート、NetWareのボリュームをマウントするNCPサポートを利用する場合もここで選ぶ。Sambaを使ってWindowsネットワークにつなぐ場合はSMBサポートはなくてかまわない。

●Partition Types

PCではないプラットフォームのマシンで利用しているハードディスクをPCにつないで使う場合は、このサブメニューで必要なパーティションタイプを選択する。

●Native Language Support

ファイル名の各国語サポートを有効にするなら、このサブメニューからコード体系を選択する。

USB supportメニュー

カーネル2.4の目玉のひとつがUSBデバイスの標準サポートだ。すべてのUSBデバイスが利用できるわけではないが、動作するデバイスも徐々に増えていくだろう。ここでは、設定する際に迷いやすい項目を選んで解説する。

●Support for USB

USBデバイスを使う場合は有効にする。モジュールにしておくことも可能だ。

●USB verbose debug messages

USB関係のデバッグ用メッセージを出すようにする。

●Preliminary USB device filesystem

ここを“y”にすると、接続しているUSBデバイスの情報や、使用されているドライバの種類を/proc/usb/ディレクトリから参照できるようになる。USB機器の認識状況を調べる際に便利な機能だ。

● Enforce USB bandwidth allocation （EXPERIMENTAL）

帯域幅割り当てを行う。“y”にすると、USBサブシステムは、USBの帯域が飽和しないようにUSBデバイスの利用帯域を調整する。“n”にすると、うまく動かないデバイスが出てくるかもしれない。

●UHCI（Intel PIIX4, VIA, ...）support

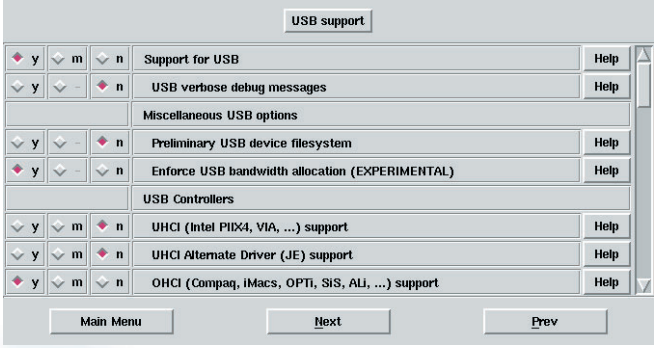
IntelやVIAのチップセットに内蔵されたUSBコントローラUHCIのドライバ。これらのメーカー製チップセットを採用したマザーボードを使っている場合は“y”にする。

●UHCI Alternate Driver（JE）support

これもUHCIのドライバだが、上記のものとは別に作られている。どちらか一方を組み込めばよい。

●OHCI（Compaq, iMacs, OPTi, SiS, ALi, ...）support

Compaq、SiS、ALiなどのチップセットではこのドライバを使う。



画面14　USB supportメニュー

●USB Device Class driversのパート

ここでは、サウンドデバイス、Bluetooth機器、ハードディスクなどのストレージデバイス、モデム、プリンタのサポートドライバを選ぶ。現状では、ストレージデバイスは動かないものが多いようだ。

●USB Human Interface Device（full HID）support

マウス、キーボード、ジョイスティック、タブレットなど、HID（Human Interface Device）全般をサポートするドライバ。

●USB HIDBP Keyboard（basic）support

キーボードのみをサポートするドライバ。full HIDドライバのサブセットで、こちらのほうがサイズが小さい。

●USB HIDBP Mouse（basic）support

マウスのみをサポートする小さなドライバ。

●Wacom Intuos／Graphire tablet support

WacomのIntuosまたはGraphireタブレットのドライバ。

●USB Imaging devicesのパート

デジタルカメラやスキャナのドライバを選択する。

●DABUSB driver

Digital Audio Broadcasting（DAB）受信機のサポート。DABは、FMラジオ放送を置き換えるデジタル放送とのことだが、日本では放送されていない（たぶん）。

●USB Network adaptorsのメニュー

ここでは、PCのUSBポート同士をつないでネットワークを構築するネットワークアダプタのドライバを選択する。

●USS720 parport driver

Lucent TechnologiesのUSS-720チップを使ったUSB－パラレルポートアダプタのドライバ。

●USB Serial Converter support

USB－シリアルコンバータのドライバを選択する。

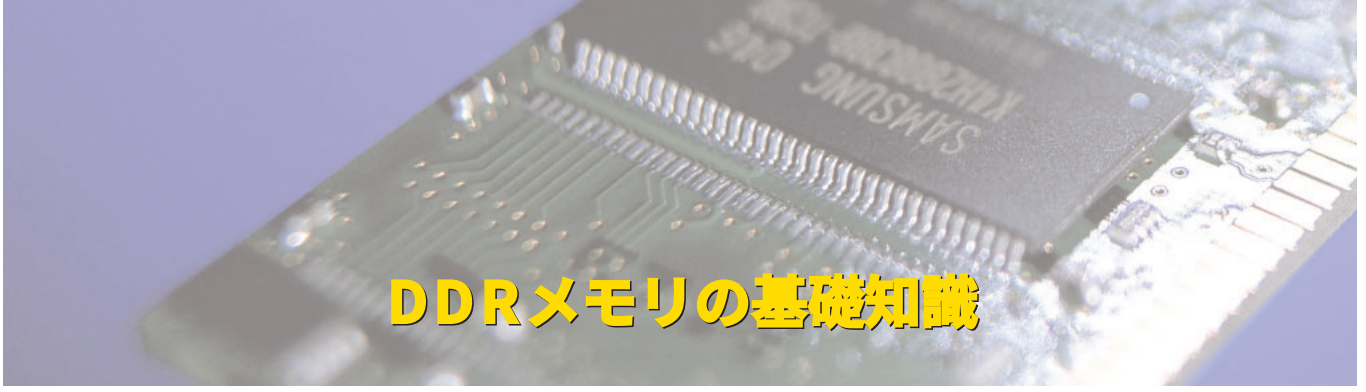
●USB Diamond Rio500 support（EXPERIMENTAL）

DiamondのMP3プレーヤRio500のドライバ。

新メモリ DDR-SDRAM の実力を探る

次世代メインメモリの大本命、DDR-SDRAMが登場した。1GHzオーバーのCPUを支える高性能にせまってみよう

文：Linux magazine ラボ
Text : Linux magazine Lab.
Photo : Junko Kitade (Dee)



DDRメモリの基礎知識

「3月号のハードウェア記事は、DDRでどうよ？」

年明けの企画会議での、副編集長のお言葉である。Linux magazine ラボメンバー各人は、机の下で「よっしゃあ」と拳を握りしめていた。

昨年末から、DDR（Double Data Rate）SDRAMと、対応マザーボードは、秋葉原界隈に少しずつ出回っていた。しかし、一般的なマザーボードやメモリと比較すると、まだまだ値段が高く、自前で買うにはちょっと勇気が必要なだったのだ。特にメモリは、PC100／PC133といった一般的なSDRAMが急激に価格を下げたせいもあって、割高感が強かった。

だが、仕事で買えるとなれば、話は別だ。そのうえ評価が終わったら、丸ごと会社での常用マシンとして使える可能性まであるのだ。これで燃えなきゃ、嘘だろう。早速翌日には、メンバーの一人が秋葉原に突撃していったのは、言うまでもない。

PCについてちょっと復習

買物に行ったメンバーが帰ってくるまでの間に、

「なぜDDR-SDRAMが必要なのか？」

という根本的な疑問に答えるために、ちょっとだけPCについて復習しておこう。PCに限らずコンピュータには、ディスク、ビデオカード、ネットワークデバイスなどさまざまな周辺機器が接続されている。CPUは、これらの周辺機器とデータをやりとりして、さまざまに加工、変換するためのプログラムを実行する。そしてメモリは、このデータを一時的に保存したり、プログラムを置くための領域だ。

実際には、CPUが周辺機器と直接データをやりとりするわけではない。間にチップセットと呼ばれるチップが置かれ、データの流れを制御している。チップセットは、数社から発売されているが、CPUメーカーでもあるIntelと、台湾のVIA Technologiesの2社が特に大きなシェアを持っている。

■440BX

図1に示したのは、'98年に発売された、Intelの440BXチップセットを中心にしたシステム構成図だ（USBやオーディオなどの周辺機器は省略してある）。440BXは、製品の寿命が短いこの業界としては、異常なほど長く使われているチップセットで、今でも440BXを使ったマザーボードは、簡単に手に入れることができる。

440BXは、主にPentium II／IIIと組み合わせて使用される。CPUやメモリ、AGPと接続される「ホストコントローラ」と、周辺機器と接続される「I/Oコントローラ」の2つからなる。これら2つのチップは、133Mバイト／秒の帯域を持つPCIバスを介して接続されている。

CPUとチップセットを接続しているバスは、FSB（Front Side Bus）と呼ばれる。440BXの場合は、64ビットのバス幅を100MHzで駆動しているので、FSBの帯域幅は、

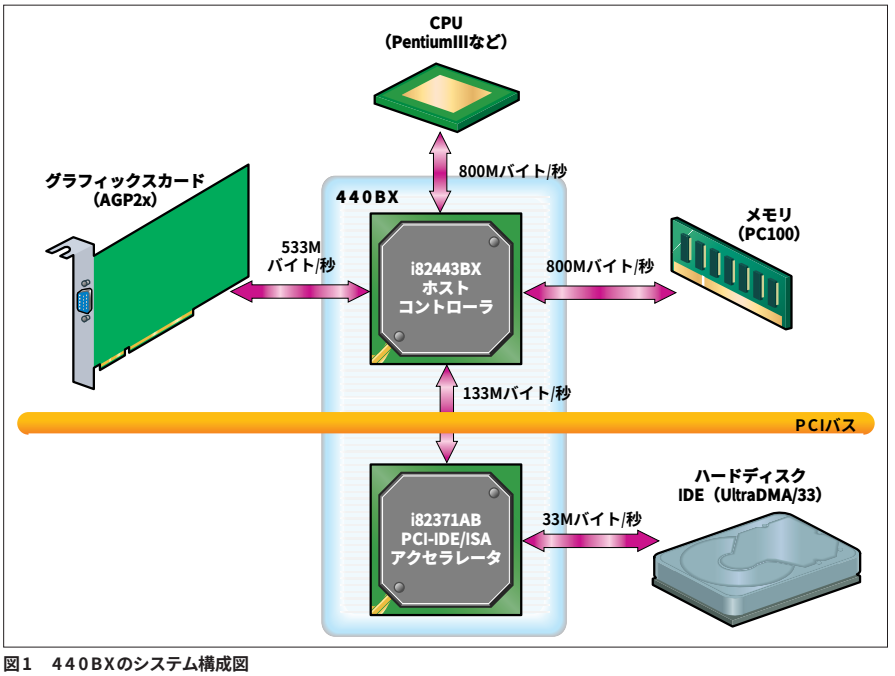


図1　440BXのシステム構成図
古さを感じさせるスペックだが、バランスの良さは今でもトップクラスだ。

100×（64÷8）＝800Mバイト／秒

となる。

「フロント」サイドバスがあるということは、「バック」もあるわけで、CPUに統合された2次キャッシュとCPUコア間をバックサイドバスと呼ぶことがあるようだ。

440BXは、100MHzで駆動されるSDRAM（PC100）をメインメモリとして利用する。SDRAMは、64ビット幅のバスを持った、DIMM（Dual Inline Memory Module）をマザーボード上のDIMMスロットに挿して利用するのが一般的だ。

■ノース／サウスブリッジ

PCのシステム構成図は、**図1**のようにPCIバスを中心として、上にCPUやメモリとホストコントローラ、下に各種周辺機器とI/Oコントローラを描くのが一般的だ。そこで、ホストコントローラをNorth Bridge（ノースブリッジ）、I/OコントローラをSouth Bridge（サウスブリッジ）と呼ぶ。

「ホストコントローラ」や「I/Oコントローラ」に相当する言葉は、メーカーによって異なるのだが、「ノースブリッジ」、「サウスブリッジ」と言い換えてしまえば、だいたい意味は通じるようだ。また、漢字を使用している国、たとえば台湾のPC情報サイトに行くと、「北橋」「南橋」という表記があったりするので、妙におかしい。

440BX以降のチップセット

440BXは、400MHzクラスのPentium IIと同時にデビューした製品であり、CPUのクロックが1GHzをオーバーする今となっては、IDEインターフェイスがUltraDMA/33しか使えない点や、AGP 2xまでしかサポートされていない点などに、古さを感じてしまう。

また、メモリの帯域不足も問題だ。元々CPUコアのクロックは、FSB（100MHz）の数倍に達している。キャッシュメモリの性能を上げることでカバーしているものの、PC100 SDRAMの800Mバイト／秒の帯域は、CPUの速度に追いついていなかった。

そこでIntelは、440BXの後継として、メインメモリにRambusのDirect RDRAMを使用するi820チップセットをリリースした。Direct RDRAMは、PC100 SDRAMの2倍の帯域（1.6Gバイト／秒）を持つ、まったく新しい構造のメモリシステムだ。

しかしi820は、Intelの歴史に残るような大失敗に終わってしまった。設計上の問題で、何度もリリースが延期になり、リリース後も安価なSDRAMも使えるようにするためのチップが原因で、マザーボードのリコール騒ぎまで起こしてしまったのだ。またDirect RDRAMそのものも、なかなか値段が下がらず、その結果普及が進まないという悪循環に陥ってしまった。

この騒動の間隙をついて、VIA Technologiesの、PC133 SDRAMを利用できるチップセットが急激にシェアを拡大し、ついにVIAは2000年のチップセットのシェアNo. 1の座についたのは、記憶に新しいところだ。

■i850

Pentium III用にDirect RDRAMを普及させることに失敗したIntelだが、次世代のCPUであるPentium 4で、再びDirect RDRAMをメインメモリとして選択した。**図2**は、Pentium 4と唯一の対応チップセットであるi850のシステム構成図である。

Pentium 4は、徹底的に高クロック化することを目指したCPUで、昨年11月に1.5／1.4GHzでデビューした（後に1.3GHzも追加）。CPUのクロックが上がっても、それに見合った速度でデータが入ってこなければ、クロックを

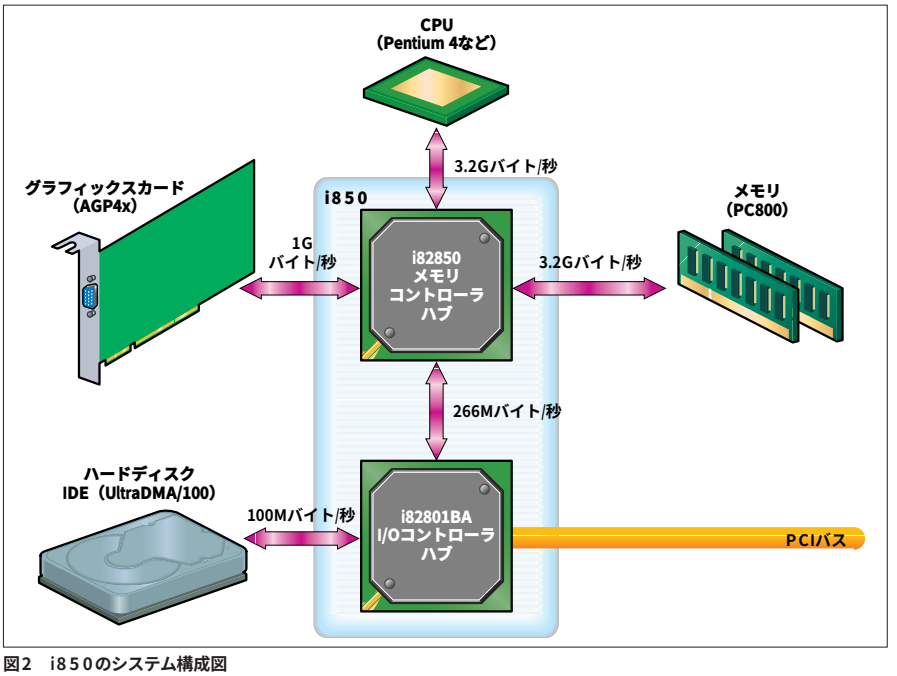


図2　i850のシステム構成図
超高クロックのPentium 4に合わせて、高価なDirect RDRAMを2本単位で使用する。

上げても意味がないが、Pentium 4ではコアの高速化に合わせてFSBは400MHzに強化されており、その帯域幅は、

400×（64÷8）＝3200Mバイト／秒

となっている。また、メインメモリには、デュアルチャネルのDirect RDRAMを用いており、FSBと同じ3.2Gバイト／秒の帯域を持つ。

i850は、AGP 4x、UltraDMA/100など440BXと比較して、より高速な規格に対応しており、最新の周辺機器を利用可能だ。また、ノースブリッジ／サウスブリッジ間は、従来のようにPCIバスを介するのではなく、専用のバス（266Mバイト／秒）で接続されており、増大するデータ量に対応している。この方式は、Intelの800番台のチップセットに共通している。

今後は、ほかのチップセットメーカーも、これと同様の専用バスを採用していくようだ。

AMDのCPUとチップセット

今のところ、Direct RDRAMをメインメモリとして採用しているのは、世界中でただ1社、Intelだけだ。もちろん、ほかのCPU／チップセットメーカーもSDRAM以上の高帯域が必要なことは認識しているはずだ。

そのうちの1社、AMDは同社のAthlon用の次世代メインメモリとして、DDR-SDRAMを選択した。Athlonは、Pentium Ⅲと互角以上の性能を持つ高性能CPUで、価格性能比が優れているために、PC自作マニアから強い支持を受けている。また最近では、国内外の大手PCベンダーもAthlon搭載機種を揃えている。

■AMD-760

昨年末、AMDからAthlon／Duron用のDDR-SDRAM対応チップセット、AMD-760が発表された（**図3**）。AMD-760は、DDR-SDRAMへの対応以外に、FSBを従来の200MHzから266MHzに高速化しているのが特徴だ。そのほかのインターフェイスについては、AGP 4x、UltraDMA/100など、最新の規格に対応しており、この点はIntelのi850と同等だ。

AthlonのFSBは、Intel製CPUとはまったく異なる、EV6と呼ばれるアーキテクチャを採用している。EV6は、元々コンパック（旧DEC）のAlphaプロセッサに用いられていたもので、供給されるクロックに対して、立ち上がり／立ち下がりの両方に同期してデータの入出力が行える。このFSB帯域の広さが、同じクロックで比較すると、AthlonがPentium Ⅲ以上の性能を示す理由のひとつだ。

今までのAthlon対応チップセットは、PC100やPC133 SDRAMのみに対応しており、FSBに比較するとメモリの帯域が見劣りしていた。DDR-SDRAMに対応したAMD-760の登場によって、AthlonはFSBに見合ったメモリ帯域を手に入れたことになる。

次世代のメモリ

上記のように、2大CPUメーカーの最新チップセットが採用した「ポストSDRAM」は、Direct RDRAMとDDR-SDRAMの2種類である。これ以外にも、サードパーティから数多くの新しいチップセットが発表され、すでに生産、販売が開始されているものもあるが、そのすべてがDDR-SDRAMのみ、またはDDR-SDRAMとSDRAMの両方

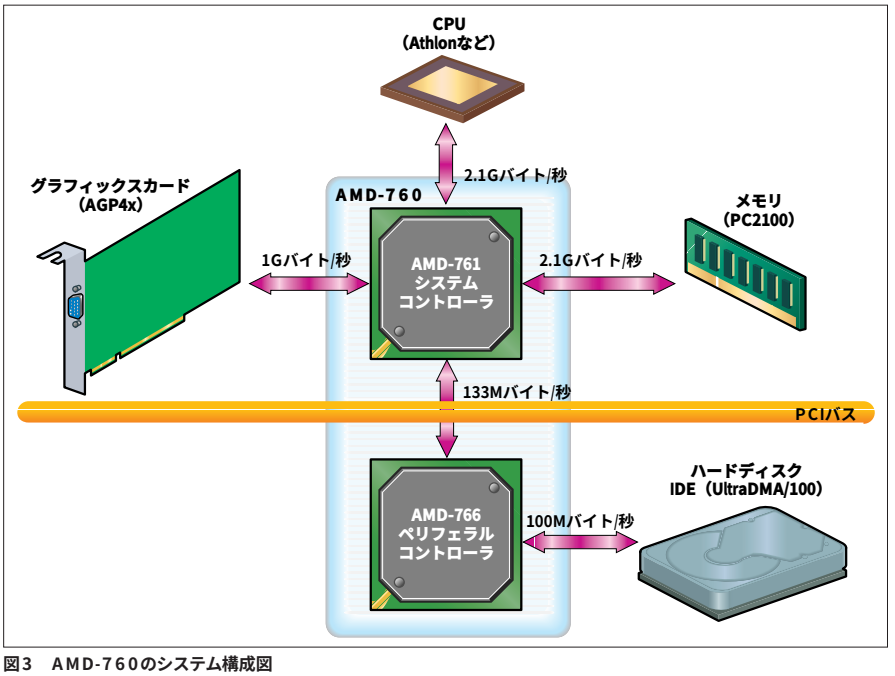


図3　AMD-760のシステム構成図
DDR-SDRAMを利用することで、AthlonのFSBに見合ったメモリ帯域が利用可能になる。

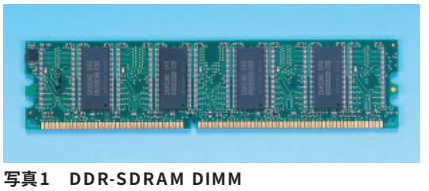


写真1　DDR-SDRAM DIMM
昨年末から出回りはじめたDDR-SDRAM DIMM。価格も徐々に下がっているようだ。

（同時使用は不可）をメインメモリとして使用する。

■DDR-SDRAM

名前に“SDRAM”が入っていることからもわかるように、DDR-SDRAMは今まで使われてきたSDRAM技術の延長として開発されてきたものだ。

SDRAMの“S”は、“Synchronous”（同期式の）の頭文字であり、外部から供給されるクロックの立ち上がりに同期して、データの入出力を行うことができる（**図4上**）。クロックサイクルは、PC100（100MHz）では10ナノ秒、PC133（133MHz）では、7.5ナノ秒になる。

これに対して、DDR-SDRAMは、**図4下**のように、供給されるクロックの立ち上がりと立ち下がりに同期してデータの入出力を行うため、ピーク時のメモリ帯域は、SDRAMの2倍になる。“Double Data Rate”の名はダテではないのだ。広帯域のメモリが必要なハイエンドのグラフィックスカードでは、すでにDDR-SDRAMが広く用いられており、その性能は実証済みといえる。

■PC1600、PC2100

現在メインメモリ用として利用されているDDR-SDRAMには、100MHz、133MHzの2種類があり、それぞれ200MHz、266MHz相当のデータ入出力が可能なため、「DDR200」、「DDR266」と名付けられている。また、これらのメモリを搭載したDDR-SDRAMのDIMMは、64ビットのバス幅を持つので、それぞれ、

```
200×（64÷8）＝1600Mバイト／秒
266×（64÷8）＝2100Mバイト／秒
```

のピーク転送能力を持つ。これらのDIMMは、それぞれPC1600、PC2100と呼ばれる。メモリ単体とDIMMの名前が異なるので、注意が必要だ。

DDR-SDRAM DIMMの名称は、ピーク転送能力を表しているのに対して、ライバルであるDirect RDRAMの名称（PC800など）は、実効クロック（400×2＝800MHz）を表している。おそらく営業上の理由（数字が大きいほうが、良さそうに聞こえる）から、命名方法が決まったのだろう。ユーザーにしてみれば、紛らわしくて迷惑な話だ。

メモリやチップセットの復習をしている間に、秋葉原買い出し部隊が帰ってきたようだ。早速、性能をチェックしてみるとしよう。

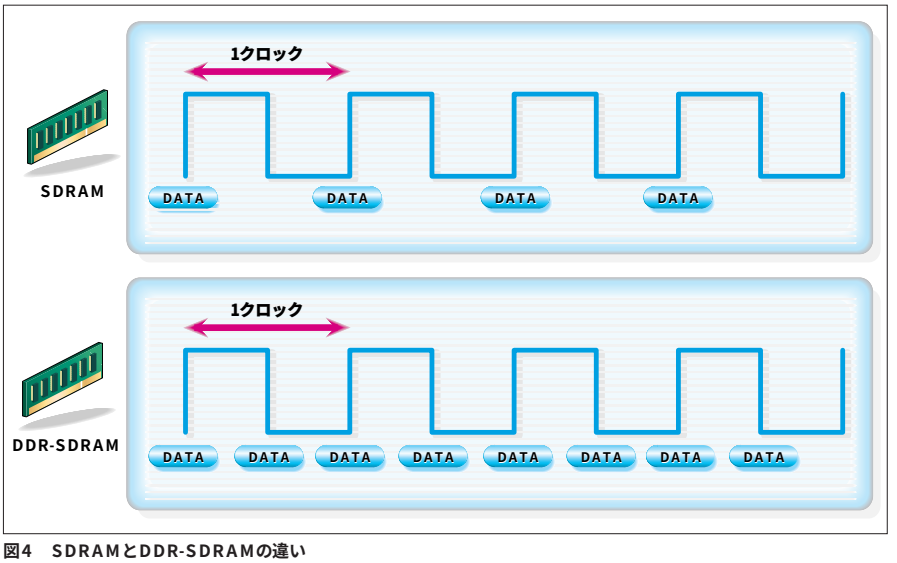
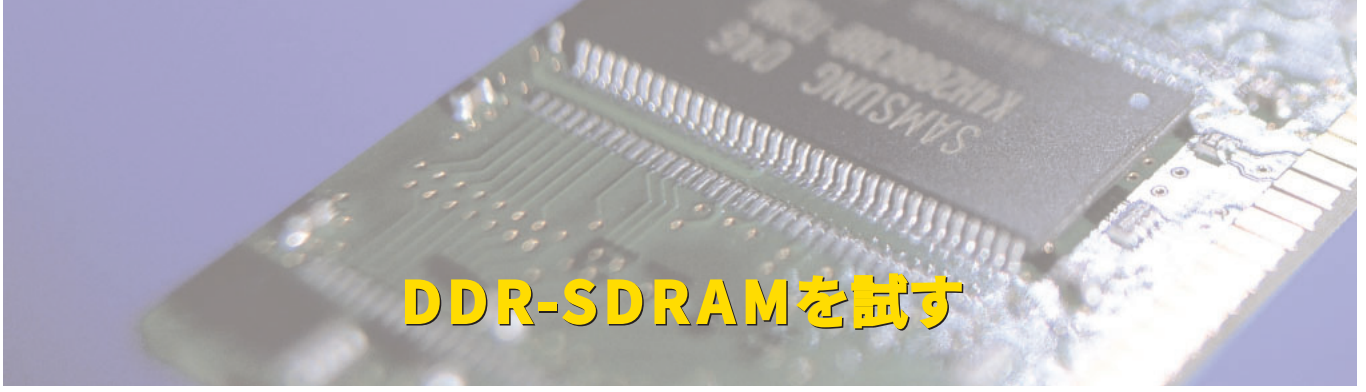


図4　SDRAMとDDR-SDRAMの違い
DDR-SDRAMは、クロックの立ち上がりと立ち下がりの両方に同期したデータ入出力が可能。

メーカー名	AMD	VIA Technologies	Intel	VIA Technologies	Intel
名称	AMD-760	Apollo KT133A	440BX	Apollo Pro133A	i850
ノースブリッジ	AMD-761	VT-8363A	82243BX	VT82C694X	i82850
サウスブリッジ	AMD-766	VT82C686B	82371AB	VT82C686B	i82801BA
ノース／サウス間	PCIバス（133Mバイト／秒）	PCIバス（133Mバイト／秒）	PCIバス（133Mバイト／秒）	PCIバス（133Mバイト／秒）	専用バス（266Mバイト／秒）
対応CPU	Athlon／Duron	Athlon／Duron	Celeron／Pentium II／III	Celeron／Pentium II／III	Pentium 4
FSB	200／266MHz	200／266MHz	66／100MHz	66／100／133MHz	400MHz
メモリ	DDR-SDRAM	SDRAM（PC100／PC133）、VC SDRAM（PC100／PC133）	SDRAM（PC100）	SDRAM（PC100／PC133）、VC SDRAM（PC100／PC133）	Direct RDRAM×2（PC600、PC800）
AGP	4x	4x	2x	4x	4x
PCIバス	Rev. 2.2	Rev. 2.1	Rev. 2.1	Rev. 2.1	Rev. 2.2
IDE	UltraDMA/100	UltraDMA/100	UltraDMA/33	UltraDMA/100	UltraDMA/100
USB	4ポート／OHCI	4ポート／UHCI	2ポート／UHCI	4ポート／UHCI	4ポート／UHCI

表1　主要チップセットの仕様



DDR-SDRAMを試す

前半でも述べたように、DDR-SDRAMへの対応は、AMD製のCPUが先行している。そこで今回は、AMD AthlonをDDR-SDRAM、SDRAMマザーボードとそれぞれ組み合わせて、性能の評価を行うこととした。

■ASUS A7M266

2001年1月現在、入手可能なDDR-SDRAM対応マザーボードは、チップセットにAMD-760を採用したものと、ALi（Acer Laboratories inc.）のMAGiK1を採用したものの2種類がある。今回は、AMD-760を搭載したASUS A7M266（**写真2**）を選択した。購入価格は、2万3800円だった。

実際には、A7M266のチップセットは純粋なAMD-760ではない。ノースブリッジにはAMD-761（**写真3**）を用いているが、サウスブリッジは、VIAのVT82C686B（以下、686B）を用いている。686Bは同社のVT82C686A（以下、686A）の後継の製品で、686Aと同様の4ポートUSB、オーディオ（AC'97）、モデム機能を備えたうえに、IDEインターフェイスもUltraDMA/100に強化されている。

686A／686Bは多くのマザーボードに採用されている実績があり、豊富な機能を取り込んでいるため、AMD製のサウスブリッジの代わりに用いられていると思われる。ASUS以外のメーカーも同じ構成のマザーボードを販売、または計画している。

マザーボード上のDIMMスロットは、2本だが、あと2本分のパターンが残っており、当初はもっと多くのDIMMスロットをサポートする予定だったと考えられる。なおマニュアルには、メモリチップの数が最大で18個までという記述がある。現在入手できるDDR-SDRAM DIMMは、128Mビットのメモリチップを、8個搭載した128Mバイトと16個搭載した256Mバイトの2種類だけなので、当面A7M266の最大メモリ容量は256Mバイトということになる。

■Athlon

AthlonはAMDが開発したx86 CPUであり、'99年に発表された。初代Athlonは、Pentium IIと同様のカートリッジのような形状で、マザーボード上のスロット（Slot A）に挿して利用するものだった。これはCPUのダイ（チップそのもの）と2次キャッシュが別になっていたためだ。2次キャッシュに用いるSRAMチップは、CPUほど高速化できないので、初代Athlonの2次キャッシュは、CPUコアの2分の1から3分の1の速度で動作していた。

一方、当時Intelの主力CPUであったPentium III（コードネーム Coppermine）の2次キャッシュは、ダイ上に統合されており、コアと同じ速度で動作可能だった。同じクロックで比較すると、初代AthlonよりPentium IIIのほうが若干速かったことの一因である。

AMDは2000年の6月に、2次キャッシュをダイに統合し、Pentium IIIと同等以上の性能を実現した新しいAthlon（コードネーム Thunderbird）をリリースした。2次キャッシュを統合したため、かさばるカートリッジのような形にする必要がなくなり、FC-PGA（Flip Chip Pin Grid Array）タイプがメインになった（**写真4**）。

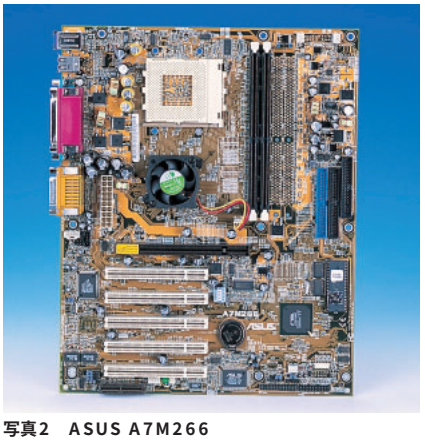


写真2　ASUS A7M266
ノースブリッジにもファンが装備されている。DIMMスロットは2本。

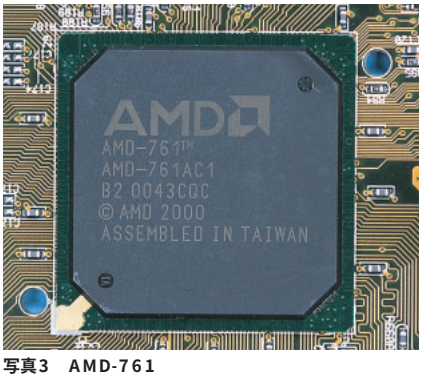


写真3　AMD-761
AMD純正のDDR-SDRAM対応ノースブリッジ。

写真4　Athlon 1000MHz
FSBは200（100×2）MHz。FSB 266MHz対応版は未発売（2001年1月現在）

今回購入したのは、Athlon 1000MHzのリテール版だ。FSBは200（100×2）MHzで、コアは供給される100MHzの10倍で動作する。

AMDエンブレムとヒートシンクファンが付属して、2万4800円だった。1GHzのCPUが2万円台なかばで手に入るとは、21世紀おそるべしだ。同梱のヒートシンクは、かなり巨大で、高性能の代償として猛烈に発熱することが予想できる。

なお“Thunderbird”を「直訳」して「雷鳥」と呼んでいる方も多いようだが、Thunderbirdと日本で言うところの雷鳥はまったく別の鳥なので、気をつけよう。

■PC2100 DIMM（128Mバイト）

現在入手可能なDDR-SDRAM DIMMは、200（100×2）MHzで動作するPC1600と、266（133×2）MHzで動作するPC2100がある。PC2100のDIMMをPC1600として利用することも可能なので、PC2100を選択した。2万800円という価格は、同容量のPC133 SDRAM（実勢価格5～6000円）と比較すると割高感がある。

購入したDIMMには「CL＝2.5」と表記されていた。CLは“CAS Latency”、CASは“Column Access Strobe”の略号だ。CLは、メモリにデータを要求してから、実際に読み出されるまでの遅延をクロック単位で表したもので、もちろん数値が小さいものほど高性能ということになる。DDR-SDRAMの規格には、PC2100にCL＝2.5とCL＝2があるが、市場に出回っているのはCL＝2.5のみである。

DDR-SDRAMとSDRAMは、DIMMの接点部分の形状が異なるため、互換性はない（**写真5**）。そもそも動作電圧が異なるので、仮に同じ形状であったとしても、スロットの共用などは不可能だ。

■ASUS A7V/WOA

比較対象のSDRAMマザーボードとして、ASUS A7V/WOA（**写真6**）を選択した。Athlon／Duron用のチップセットとして、最も人気の高いVIA Apollo KT133（**写真7**）を採用しており、SDRAMのほかにVC SDRAMも利用可能だ。3本のDIMMスロットを持ち、512MバイトのDIMMを利用することで最大1.5Gバイトまでメモリを搭載できる。

サウスブリッジには686Aを用いているので、IDEはUltraDMA/66までの対応になるが、Promise Technology製のIDEコントローラPDC20265を搭載しているので、UltraDMA/100に対応したハードディスクも利用できる。そのほか4ポートUSB、オーディオ（AC'97）、モデム機能などはA7M266と同等だ。

テスト条件

以上の新規に購入した部品以外は、Linux magazineラボにあるものを利用してテスト機を組み立てた。

マザーボードとメモリを交換することで、PC100 SDRAMと、PC1600 DDR-SDRAMの比較が行える。「PC2100のDIMMを買ったはずでは?」と思われるかもしれない。これは、AMD-761がFSBとメモリのクロックは同期、つまり同じ値で使うようになっているためだ。入手したAthlon 1000

製品名	詳細	購入価格
ASUS A7M266	DDR-SDRAM対応マザーボード	2万3800円
ASUS A7V/WOA	SDRAM対応マザーボード	1万7800円
Athlon 1000MHz	Socket A CPU（リテール版）	2万4800円
Athlon 750MHz	Socket A CPU（バルク）	1万3000円
DDR-SDRAM 128Mバイト	PC2100仕様DIMM（CL＝2.5）	2万800円

表2　今回購入した部品

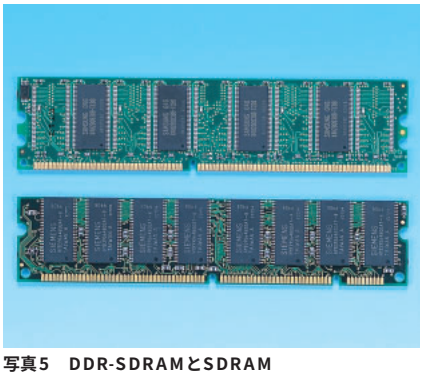


写真5　DDR-SDRAMとSDRAM
DIMMの形状が異なるので、古いマザーボードでDDR-SDRAMを利用することはできない。

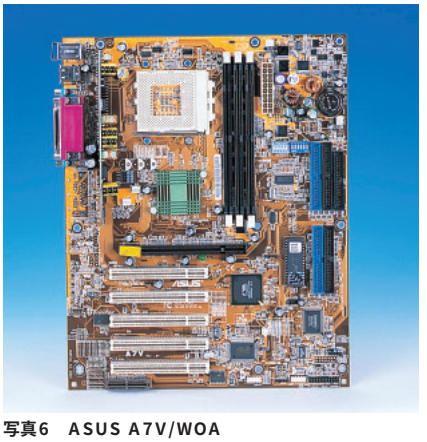


写真6　ASUS A7V/WOA
Athlon／Duron用チップセットで最も人気の高いKT133を採用している。

写真7　VT-8363
Apollo KT133のノースブリッジ。

MHzはFSBが200MHzで、供給されているクロックは100MHzだ。そのためメモリも100 MHzのDDRであるPC1600に固定される。PC2100として利用したければ、FSBが133MHzのAthlonが必要になる。

これに対してVIAのApolloシリーズチップセットなどでは、FSBとメモリのクロックを非同期に設定できるため、FSBが100MHzのCPUに、より高速なPC133 SDRAMを組み合わせるというような柔軟なシステム構成が可能になっている。

もっとも、FSBとメモリを非同期に設定すると、データ読み出しの際に、1クロック余分にウェイトをかける必要が出てくる。そのためメモリを高速化しても実効性能はほとんど上がらないことになる。

このことが原因なのか、それともDDR-SDRAMでは非同期の設計が困難なのかは不明だが、発売または発表されているDDR-SDRAM対応チップセットは、ほとんどが同期モードのみで利用するようになっている。

どうせならPC2100もテストしてみたいが、そのために必要になるFSBが266（133×2）MHzのAthlon 1000MHzは、残念ながら未発売だ。

■FSB 266MHzのAthlon

最近のマザーボードには、FSBクロックやコア／FSBクロックの比を変更できるものがある。A7M266もそのためのBIOS設定画面やジャンパーピンが用意されていると、説明書に記されている。そこで、

```
100×10＝1000（MHz）
```

の代わりに、

```
133×7.5＝1000（MHz）
```

になるように設定することで、同じコアのクロックで、PC2100を利用することができる。

ただし最近のAthlonでは、簡単に倍率の変更ができないようになっている。写真8に見られるように、Athlonの表面には、いくつかの回路パターンのようなものがある。このうち「L1」と書かれている4ヵ所のパターンが接続されていない製品は、倍率が固定化されているのだ。買ってきたAthlonは、写真8のように切断されていた。

もちろん「切断されているものは、つなげばいい」わけで、対策がWebサイトや雑誌で紹介されている。コンダクティブ・ペンや、シャープペンシル（！）で接続するというのが一般的なようだ。

ラボでもこの処置を行い、再度A7M266で倍率変更を行おうとしたのだが、それでもBIOS設定画面で倍率変更が有効にならなかった。さらに調査してみると、どうやらA7M266の現バージョンでは、倍率の変更ができないことがわかった。

Athlonもう1個！

しかしPC2100 DDR-SDRAMで測定したデータがないのでは、画竜点睛を欠いてしまう。そこで再び買物部隊が秋葉原に走り、今度は750MHzのAthlonを追加購入してきた。100×7.5＝750MHzのCPUを、

```
133×7.5＝1000（MHz）
```

で使おうという作戦である。いわゆるオーバークロックだ。750MHzという低い（！）クロックのAthlonはすでに品薄でリテール版は皆無だったが、なんとかバルク品を入手できた。オーバークロック時の発熱にも耐えられるように、大きめのヒートシンクファンも一緒に購入した。

コア電圧を上げるなどして、ベンチマーク測定や、CPU負荷100%で1日以上動作可能なレベルまで安定させることができたので、これでPC2100 DDR-SDRAMの測定を行った。

CPUを規定以外のクロック、電圧で使用すると、CPUやマザーボードなどに復旧不可能なダメージが与えられる可能性があります。その結果によるいかなる損害についても、Linux magazine編集部は責任を負いません。
また、この記事についての個別のご質問・お問い合わせはお受けできませんので、ご了承ください。

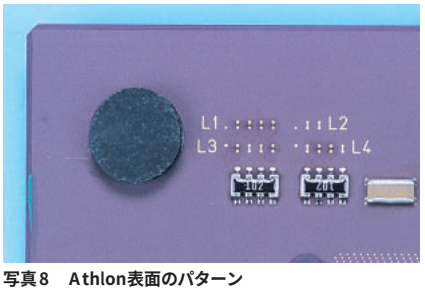


写真8　Athlon表面のパターン
L1と書かれた4ヵ所を接続することで、クロック倍率が可変になる。

メモリ	PC100 SDRAM	PC1600 DDR-	PC2100 DDR-
マザーボード	ASUS A7V/WOA	SDRAMASUS A7M266	SDRAMASUS A7M266
チップセット	VIA Apollo KT133	AMD AMD-760	AMD AMD-760
メモリサイズ	128Mバイト		
CPU	AMD Athlon 1000MHz		
ハードディスク	Maxtor 96147H6（60Gバイト）		
グラフィックスカード	WinFast GeForce2 MX		
OS	LASER5 Linux 6.4		
カーネル	2.2.18		

表3　テストに使用したPCのスペック

ベンチマーク

最新のチップセットを利用したマザーボードではあるが、Linuxのインストールや使用時に特に問題は起きなかった。LASER5 Linux 6.4はカーネル2.2.16を採用しているが、テスト時の最新の安定版であるカーネル2.2.18に入れ替えて、測定を行った。

今回ベンチマークプログラムとして使用したのは、HDBENCH Clone、LMBENCH、Quake III Arena（デモ版）の3本だ。またアプリケーションベンチマークとして、Linuxカーネルのコンパイルを行った

。

HDBENCH Clone（**画面1**）は、二之宮祐樹氏が作成したベンチマークソフトで、Windows用のHDBENCHと同等の外観を持っている。今回使用したのは、最新のバージョン0.14.1である。HDBENCH Cloneは、演算、メモリアクセス、描画速度、ディスクI/Oの測定が行える。今回は演算とメモリアクセス速度の項目を使用する。

LMBENCHは、BitMoverが公開しているベンチマークソフトで、データ転送時の帯域幅（連続でデータを転送する際の速度）やレイテンシ（データが要求されてから転送が始まるまでの遅延）を計測できる。メモリ操作のほかにネットワーク、ファイルシステムなどを対象にした測定も行える。今回はメモリの帯域幅測定に利用した。

カーネルコンパイルは、/usr/src/linuxディレクトリに移動後、コマンドラインから、

```
# make menuconfig
```

として、何も設定を変えずに保存・終了して“.config”ファイルを作成したのち、

```
# make dep;
# make clean;
# time make bzImage
```

としてカーネルコンパイルの所要時間を測定した。

カーネルのバージョンは、2.2.18である。各設定ごとに測定を3回行い、平均値を測定値としている。

Quake III Arenaは、非常に負荷の高い3Dアプリケーションである。プログラムに含まれている描画速度測定の機能を用いた。

方法は、「~」（チルダ）キーを押してコンソールを表示させ、

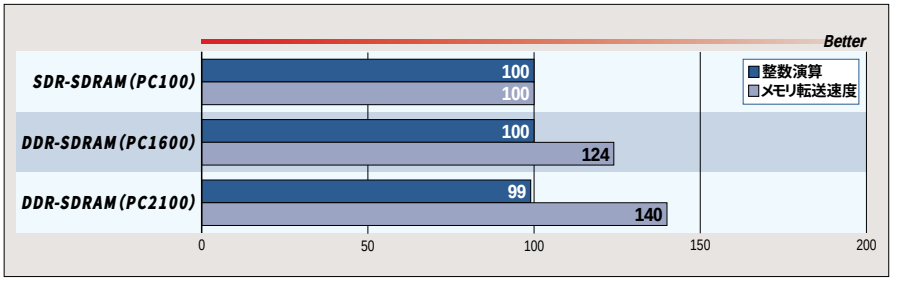
```
/timedemo 1
```

と入力し、用意されている2種類のデモをそれぞれ実行して、1秒あたりの表示フレーム数（フレームレート）を確認することで行った。テクスチャデータの品質、光源処理などはデフォルトのまま、色数は16ビットカラーで測定を行っている。

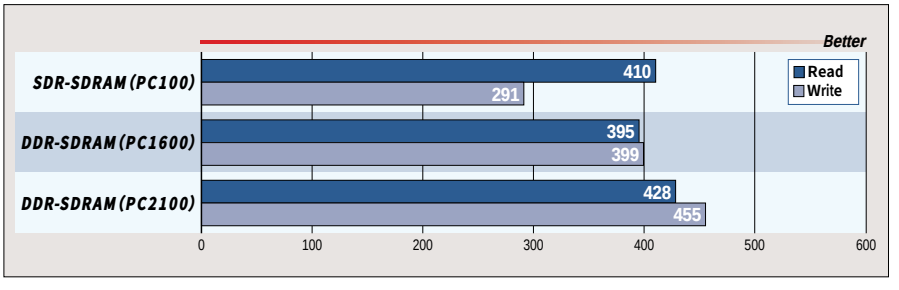
■HDBENCH Clone

グラフ1は、整数演算とメモリ転送速度の結果を、PC100 SDRAMの結果を100とした相対値で示している。CPUコアの速度が同じなので、整数演算の速度はまったく同じレベルであった。グラフには表示していないが、浮動小数点演算も同様である。

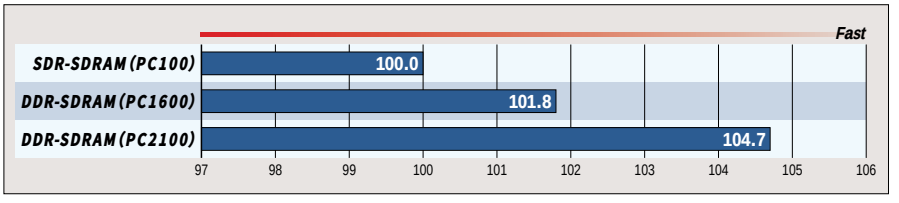
グラフ1　HDBENCH Clone（対PC100比）



グラフ2　LMBench（Mバイト／秒）



グラフ3　カーネルコンパイル（対PC100比）



それに対して、メモリ転送速度は、PC1600で20%以上、PC2100では50%近く高いスコアを示しており、メモリシステムだけを取り上げてみると、DDR-SDRAMのSDRAMに対する優位ははっきりしている。

■LMBENCH

LMBENCHの測定結果から、メモリシステムの読み出し・書き込みの帯域幅を示す（**グラフ2**）。読み出しについてはKT133＋PC100の性能が高く、わずかとはいえAMD-760＋PC1600の結果を上回った。これはメモリの差というよりも、チップセットやマザーボードの設計による差と考えるべきだろう。一方、書き込みでは、DDR-SDRAMがSDRAMのスコアを大きく上回っている。

メインメモリへの書き込みは、キャッシュシステムのおかげで、実使用時の性能に影響を及ぼしにくい。write back機能を備えたキャッシュなら、メインメモリへの書き込みを待つ必要がないからだ。反対にメインメモリからデータを読み出す場合は、実際に読み出されるまで待たされることになる。以上のことから、実際にPCを利用している時には、**グラフ1**、**2**で見られるように50%も速くなったとは感じられないだろう。

■カーネルコンパイル

カーネルに限らず、コンパイル作業を高速化するには、ハードディスクの性能を上げるべきだと言われている。しかし、単体のハードディスク性能が向上した今では、高速なディスクシステムを用いても、コンパイルの時間は短縮されない（Linux magazine 2000年12月号「使えるパーソナルRAIDシステム」参照）。

ハードディスクが30Mバイト／秒以上、CPUが1GHz前後と十分に速いPCなら、メモリの性能が効いてくる（**グラフ3**）。PC2100 DDR-SDRAMは、PC100と比較して、5%弱速度が向上した。

カーネルコンパイルでは108秒が103秒に向上した程度で、それほどありがたみを感じないかもしれない。だが、より大規模なソフトウェアを開発する際には、5%の差は意味があると言えるだろう。

■Quake III Arena

画面サイズを640×480ピクセルにした場合は、DDR-SDRAMにしたことでフレームレートの改善が見られる（**グラフ4**）。ただ、ディスプレイのリフレッシュレート以上のフレームレートは、実際に遊ぶときには意味がない。

そこで、徐々に画面サイズを大きくして測定を繰り返したが、大きくなるにつれてDDR-SDRAMとSDRAMの差は縮小し、1280×1024ピクセルモードではついに3つの条件でまったく同じスコアになってしまった（**グラフ5**）。

グラフィックスカードに搭載されているGeForce2 MXの3D描画性能は高く、グラフィックスカードのせいで、スコアが頭打ちになることは考えにくい。かなり不可解な結果だが、解像度が高いほど必要なデータが増えるため、1280×1024では1GHzのAthlonで生成できるデータの限界に達しているのではないだろうか。今後、さらに高クロックのCPUが登場してくれば、DDR-SDRAMの帯域を有効に活かせるということだ。

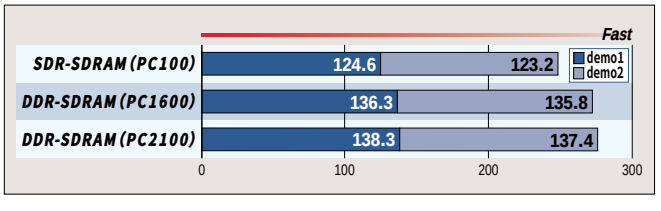
DDR-SDRAMは買いか？

PCの性能を向上させるには、各部の性能をバランス良く高めることが必要だ。上記の結果から大規模なソフトウェア開発では、CPUとディスクシステムが十分に速ければ、メモリ性能を向上させることでパフォーマンス改善が期待できる。DDR-SDRAMを導入する価値はあるだろう。

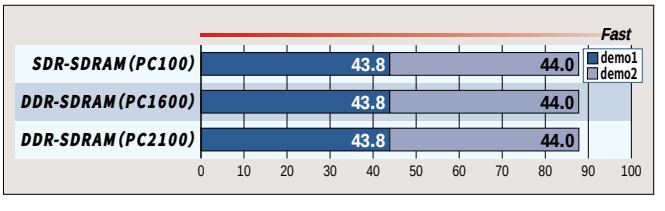
一方、3Dゲームを高解像度でプレイするなら、1GHz程度の遅い（！）CPUにはSDRAMで十分とも言える。幸い、CPUのクロックは今後も順調に上がっていくようなので、ゲーマー諸氏はDDR-SDRAMを先物買いしておくのもいいだろう。

評価のために長時間DDR-SDRAM PCを使用したわけだが、1GHz Athlonの効果もあり、とても快適だった。またLinuxへの対応の点でもまったく問題はなかった。1GHz以上の高性能CPUを用意できるなら、DDR-SDRAMは「買い」と評価しておこう。

グラフ4　Quake III Arena 640×480（FPS）



グラフ5　Quake III Arena 1280×1024（FPS）



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ユーザーに朗報がある。1月18日に、NTT東日本とNTT西日本から、3月1日より月額利用料が、今までの4500円から、3600円に値下げされると発表があった。ますますインターネットサーバが身近になる。
さて、今回はメールサーバの構築だ。メール配信を理解して、安全なメールサーバを構築しよう。

ACT. 3

ISDN フレッツ 常時接続

回線切断への対処

フレッツ・ISDNは、常時接続を保証するサービスではないということを、第1回で説明した。実際、フレッツ・ISDNでサーバを運用していると、頻繁ではないが1週間に1度ぐらいの割合で回線が切断されていることがあった。残念ながら、フレッツ・ISDNでは回線が切断されたとしても、プロバイダ側から自動的に再接続が行われないため、サーバ側から再接続を行わなければならない。もちろん、サーバ側からインターネット上のWebページを参照するなど、外部への接続要求があれば、ルータによって自動的に接続が行われる。しかし、夜間など、誰も外部への接続要求をしていない場合などは回線切断状態が続いてしまう可能性もある。

特に、メールサーバとして運用するのであれば、長時間の回線切断はメールが届かないなどの重大な問題となってしまう。

しかし、フレッツ・ISDNのサービスの性格上、意図しない切断を防ぐことができない。そこで、回線が切断されても自動的に再接続が行われるようにしておこう。

回線の再接続の最も簡単な方法は、定期的に外部へ接続することだ。たとえば、pingコマンドで、「ping www.ascii.co.jp」などとするだけでも、再接続することができる。このような要求を、一定間隔で出すことによって、長時間の回線切断を防ぐことができる。

画面1　ntsysvコマンドによるcrondデーモンの起動

そこで、編集部のサーバでは、10分間隔で外部への接続を行うように設定することにした。ただし、定期的にpingを発信すると、相手側サーバから不審に思われる可能性もあるので、pingコマンドではなく、自分自身の名前の解決を行うようする。

名前の解決であれば、定期的に実行しても不審に思われることはないだろう。まして、自分のサーバの名前であればなおさらだ。なお、自分自身のサーバの名前の解決であっても外部への接続要求が発生するのは、前回解説したCIDRによるサブネットの副作用だ。クラスC以上など、ビットクオテット境界でサブネット化されたネットワークのDNSでは、内部で名前の解決が行われてしまうために、外部への接続要求は発生しない。

定期的なコマンドの実行

一定間隔で、コマンドを実行するには、cronを使う。cronは指定した時間に、指定したコマンドを自動的に実行する。

まず、crondデーモンが起動しているかを確認する。

```
$ /etc/inet.d/crond status
```

または、

```
$ /etc/rc.d/init.d/crond status
```

「running...」と表示されていればcrondデーモンが起動している。もし、起動していないようであれば、ntsysvコマンド（Turbolinuxでは、turboserviceコマンド）でcrondデーモンを起動するように設定しておく（**画面1**）。

次に、rootユーザーになって、/etc/crontabファイルの最後に**リスト1**にある2行を追加する。

リスト1は、10分ごとにroot権限で「nslookup ns1.ascii-linux.com」を実行するというものだ。

/dev/nullにリダイレクトしているのは、nslookupコマンドの出力結果を捨てるためだ。これを行わないと、出力がroot宛にメールで届いてしまう。

この設定により、回線が切断された場合でも10分以内に再接続が行われるようになる。

名前の解決を行う間隔を短くすれば、回線切断の時間を短くすることができるが、そのぶんムダなトラフィックが増大してしまう。利用するサーバの運用状態に合わせて、タイミングを調整してほしい。

▼リスト1　crondデーモンに実行させるコマンド

```
#flet's keep connect
*/10 * * * * root nslookup ns1.ascii-linux.com >/dev/null
```

ISDN 常時接続 メールサーバを構築しよう！

DNSの設置も完了したし、回線の長期切断にも対応できた。

次に行わなければならないのがメールサーバの構築だ。特に、DNSに問題があった場合など、外部からサーバ管理者にコンタクトを取る重要な手段となるので必ず設定しよう。

メール配信について

メールの送受信（メール配信）は、メールサーバ上で稼働するMTA（Mail Transport Agent）と呼ばれるプログラムが行う。MTAがお互いのMTAと通信しながら、適切にメールを配信することによって、あたかも郵便物のように特定のユーザーのメールスプールにメールが届けられる（**図1**）。

Linux上で稼働するMTAはいくつも開発されており、代表的なものは、sendmail、postfix、qmailなどがある。その中でもsendmailは、インターネット上の80%のメールサーバで利用されているといわれているMTAだ。これは、UNIX上のMTAとしての長い歴史に加え、最も機能が豊富だという理由からであるが、そのぶん巨大なシステムで、設定が「異常」に複雑だ。

MTAは、SMTP(Simple Mail Transport Protocol）というプロトコルを使用する。このため、メールサーバをSMTPサーバと呼ぶことも多い。これに対し、POP（Post Office Protocol）サーバと呼ばれるものがある。

よく、「SMTPサーバはメールを送信するサーバで、POPサーバはメールを受信するサーバ」だと誤解されるのだが、SMTPサーバがメールの「送受信」を行うのであって、POPサーバはメールを受信しない。POPサーバは、ユーザーのメールスプールからメールを取り出すためのものであり、サーバマシン以外でメールを読むためのものだ。つまり、メールの送受信を行うだけなら、SMTPサーバだけでよいということになる。しかし、SMTPサーバ以外のマシンからでもメールを読みたい場合が大半なので、POPサーバを用意して、ユーザーのメールスプールに保存されたメールをメーラが取り出せるようにしているのである。なお、現在のPOPサーバはPOP3というプロトコルが標準となっている。

また、POPと同様の仕組みとして、IMAP（ Internet Message Access Protocol）がある。POPは受信したメールをすべてユーザーのローカルマシンのメールボックスにダウンロードする。しかしIMAPは、サーバのメールボックスを利用し、必要なメールだけを取り出せるようになっている。このため、POPを使用した際のように、複数のマシンでメールを読んでもメールが分散するようなことがないし、モバイル環境などで必要なメールだけを読みたい場合などに特に適している。

なお、IMAPはIMAP4というプロトコルが標準となっている。

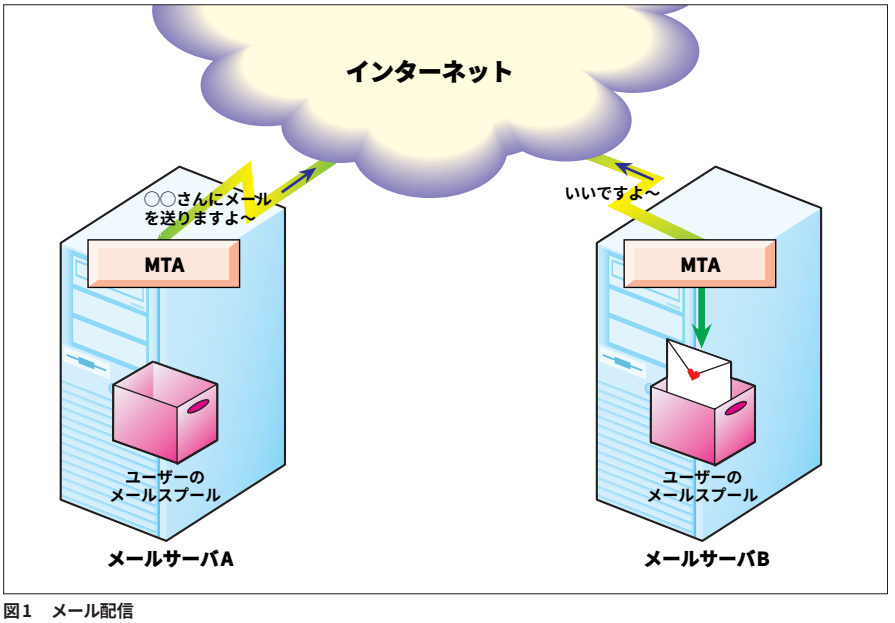


図1　メール配信

メールの不正中継とは

少し前までは、誰でもインターネット上のSMTPサーバを「勝手」に使ってメールを送信することができた。いわば、セキュリティに関してまったく関知していなかったのだ。インターネ

ットが平穏だった時代はこれでも良かったのだが、この仕組みを悪用した、勝手に大量のメールを送る者が登場するようになった。いわゆる、SPAMメールや、匿名メールである（**図2**）。

このため、現在ではSMTPサーバはユーザーアカウントを持つ正規の利用者以外はメールを送信できないように設定するのが「常識」になった。

この、正規の利用者かどうかをチェックする方法として一番簡単なのは、LAN内からの送信要求かどうかをチェックする方法だ。しかし、この方法ではたとえ正規の利用者であっても、LAN外（たとえば、インターネット上）からメールが送信できなくなってしまう。これでは、外出先からプロバイダ経由で接続した場合などはメールを送信することができない。

これと同様の問題はプロバイダの海外ローミングなどでも起こる。そこで、多くのプロバイダではSMTP before POPという認証方法を利用している。

メールを取り出すためのPOPには当然、以前から認証システムが備わっていた。認証システムがなければ、他人のメールを勝手に読めてしまうことになるからだ。

この認証システムを利用して、「POPサーバで認証されれば、メールの送信を許可しましょう」というのがSMTP before POPの発想だ。しかし、この認証方式ではメールを送信する前に、必ずメールの受信確認（POPサーバの認証）を行わなければならない。これでは少し不便だ。実際、POPとIMAPを切り換えてメールを読む場合は都合が悪い。

本来ならSMTPサーバがメールを送信する際に、正規の利用者かどうかを認証するシステムがあれば一番いいのだ。実は、sendmail 8.11.0には、SMTP AUTHという、メールの送信時に正規の利用者かどうかをチェックする仕組みが用意されている。これを利用すれば、不正中継を禁止しつつ、正規の利用者はインターネット上のどこからでもメールを送信することができる。

今回、編集部のSMTPサーバでもこれを利用することにした。ただし、利用できるメーラはSMTP AUTHに対応しているものに限られる。もっとも、Windows上のメーラとして多くのユーザーを持つBeckey!（バージョン2.03）や、Outlook Express 5.5など、最新のメーラは対応しているし、今後さらに対応メーラは増えていくだろう。

ファイアウォールの設定

前回、ダイヤルアップルータのIPパケットフィルタリング機能を利用してファイアウォールを構築した。前回設定したIPパケットフィルタリングでは、HTTP（WWW）と、DNSが利用する、ポート80番と53番だけが通過できるように設定しておいた。この設定のままでは、今回構築するSMTPサーバ、POPサーバ、IMAPサーバが使用するプロトコルを通過させることができない。そこで、これらのサーバが使用するプロトコルが通過できるようにIPパケットフィルタを追加しよう。

今回構築するSMTPサーバ、POPサーバ、IMAPサーバが使用するプロトコルとポート番号はそれぞれ、25番、110番、143番だ。ただし、sendmailなどがユーザー認証を行う場合に、ポート113番を使用する場合がある。ポート113番が通過できなくてもメールの送受信は可能なのだが、非常に時間がかかってしまうことがある。このため、ポート113番も開けておくほうがいいだろう。

それでは、前回の設定に加え、前述した4つのポートがプロトコルを通すように設定しよう（**画面2**）。

なお、編集部で使用しているダイヤルアップルータ（ヤマハ RTA50i）では、ポート番号の別名がRFC 1700で定義されているウェルノウンポート番号（/etc/servicesに定義されている）と若干違うので注意してほしい（**表1**）。

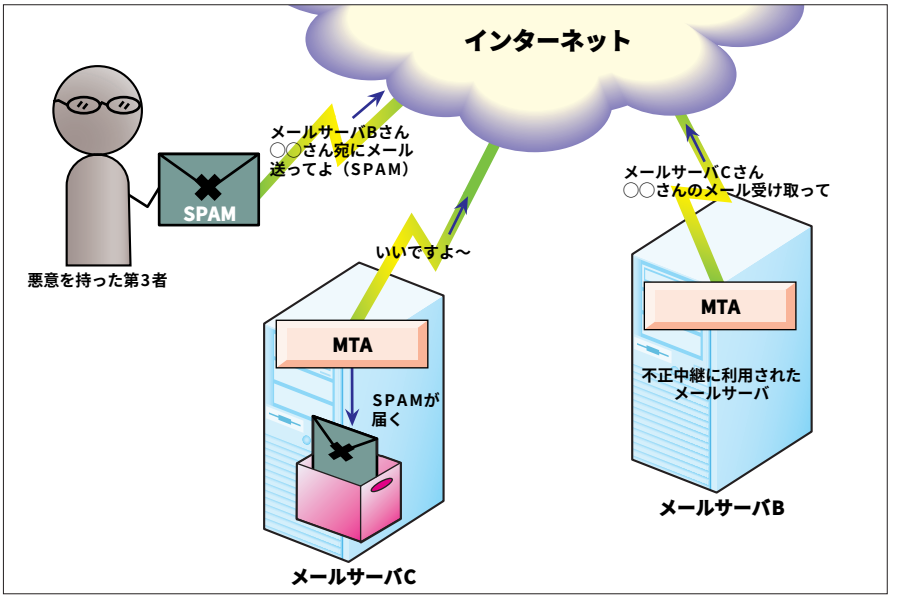


図2　メールの不正中継

別名が不明であれば、番号で定義したほうがいいだろう。

新しくIPパケットフィルタを設定したら、フィルタを適用するように設定する（**画面3**）。これで今回構築するサーバが利用するプロトコルがファイアウォールを通過できるようになった。

DNSの設定

SMTPサーバを構築するには、DNSの設定も必要となる。前回DNSの設定を行ったが、前回の設定だけではSMTPサーバの設定としては不十分だ。外部のサーバからメールを受信するために、SMTPサーバの名前を外部に公開する必要がある。

SMTPサーバをDNSに登録するには、MXレコードを正引き用のDNSのゾーンファイル（DNSデータベースファイル）に登録する。MXレコードを設定した新しいゾーンファイルは**リスト2**のようになる。

今回追加したのは、「IN　MX」で始まる行だ。これが外部にSMTPサーバの名前を知らせるための設定になる。「10」という数値は、複数のSMTPサーバが存在する場合の優先度を数値で指定するためのものだ。今回は1つのSMTPサーバしかないので意味はない。なお、指定した数値が小さいSMTPサーバがメールの受信時に優先される。あとでSMTPサーバを追加した場合に登録が簡単なように、10の倍数で指定するのが一般的だ（優先度の高いSMTPサーバを新たに設置した場合に、「5」などと指定することができる）。

リスト2では、SMTPサーバとして、「mail.ascii-linux.com」を指定しているが、実際にはそんなサーバは存在しない。そこで、「ns1.ascii-linux.com」と同じIPアドレスを「mail.ascii-linux.com」と定義している。

DNSゾーンファイルを変更した場合は、必ずDNSゾーンファイルのシリアル番号を変更するのを忘れないようにしよう。

設定が完了したら、DNSを再起動しよう。以上でDNSの設定は完了だ。

sendmail

編集部のサーバでは前述したように、MTAとしてsendmailを使うことにした。設定は複雑だが、SMTP AUTHなど、便利な機能が使えるからだ。

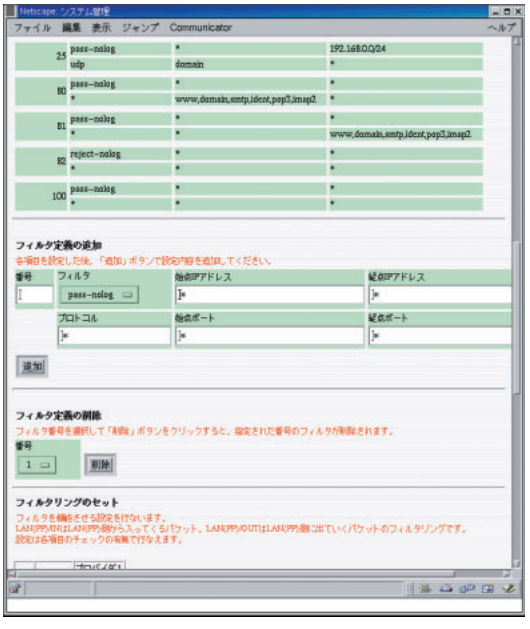
sendmailの設定は、/etc/sendmail.cfで行う。このsendmail.cfファイルの記述方法は非常に複雑で、とてもゼロから書くことはできない。これは、多様なインターネット上のサーバ間でメール配信を実現するため（たとえばUUCPなど）、拡張に拡張を重ねた結果だといえるだろう。

そこで、sendmail.cfファイルを自動的に作成する便利なツールを利用することにする。

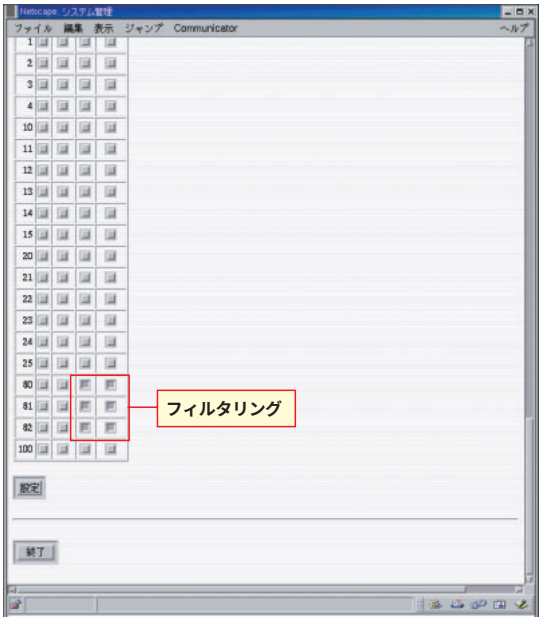
よく使われるのは、WIDE Projectの中村素典氏が作成した、CF（WIDE sendmail.cf Generation Package 3.7）だが、残念ながらバージョンアップが

サーバ	プロトコル	ポート番号	ウェルノウンポート	RTA50iの別名
SMTPサーバ	SMTP	25	smtp	smtp
POPサーバ	POP3	110	pop-3	pop3
IMAPサーバ	IMAP4	143	imap2	imap2
-	-	113	auth	ident

表1　各サーバが使用するプロトコルとポート一覧



画面2　IPパケットフィルタの追加



画面3　IPパケットフィルタの適用

行われておらず、sendmail 8.11.0で使われる定義ファイルのバージョンであるV9には対応していない（実際には設定はできるが、定義ファイルのバージョンが古いという警告が出る）。また、CFではSMTP AUTHの設定を行うことができない。

そこで、定義ファイルのV9に対応したsendmail-cfを使用することにする。sendmail-cfは、sendmailの標準の設定ツールとしてsendmailとともに配布されている。

sendmail-cfは編集部のサーバにはインストールされていなかったので、Red Hat Linux 7J（以下、Red Hat 7J）のDisc 2に収録されているパッケージを利用してインストールすることにした。なお、sendmail-cfは古くからUNIX上のマクロプロセッサとして使われているm4マクロを利用するので、m4マクロがインストールされていない場合は、こちらもインストールする必要がある。m4マクロはRed Hat 7JのDisc 1に収録されている。

さらに、SMTP AUTHが認証する際に利用する、SASL認証ライブラリも必要となる。

ここで必要となるパッケージを**表2**にまとめた。

必要なものがインストールされているかどうかは、rpmコマンドで確認することができる（**画面4**）。

インストールされていない場合は、CD-ROMからインストールするか、Red HatのFTPサイト（ftp://pub/redhat/redhat-7.0/i386/ja/RedHat/RPMS/）からダウンロードする。

インストールもrpmコマンドを利用する。ただし、インストールするにはroot権限が必要だ（**画面5**）。

すべてインストールしたら、sendmail.cfファイルを作成しよう。

sendmail.cfの作成

sendmail.cfファイルを作成するには、sendmail.cfファイルの基となる、sendmail.mcファイルを作成する。sendmail.mcファイルはいくつか雛形があるので、自分の環境に近いものを選んで利用するといいだろう。通常は、redhat.mcファイルを利用すればいいだろう。このファイルをコピーして雛形として利用する。

```
# cd /usr/lib/sendmail-cf/cf
# cp redhat.mc sendmail.mc
```

次に、sendmail.mcファイルを編集して必要な項目を追加する（**リスト3**）。

①は認証に関する設定だ。これにより、外部からメールを送信する際にSASL認証ライブラリによる認証が行われるようになる。なお、LAN内からは認証しなくてもメールを送信することができる。

現在のsendmailでは、デフォルトでメールの中継を禁止しているので、この設定を行わない場合は、LAN外から一切メールが送信できないようになる。

②はメールの「From:」行を書き換

▼リスト2　正引きゾーンファイルに（/var/named/domain）SMTPサーバを登録

```
@       IN      SOA     ns1.ascii-linux.com. rina@hogehoge.co.jp.  (
                                2001012501 ; Serial
                                28800      ; Refresh
                                14400      ; Retry
                                3600000    ; Expire
                                86400 )    ; Minimum

        IN NS    ns1.ascii-linux.com.
        IN NS    ns2.sphere.ad.jp.

        IN      MX      10      mail.ascii-linux.com.          ;SMTPサーバ

ns1     IN A     111.122.133.130
mail    IN A     111.122.133.130

www     IN CNAME ns1.ascii-linux.com.

localhost         IN A 127.0.0.1
```

シリアル番号を書き換える

SMTPサーバを公開

SMTPサーバの別名を定義

```
# rpm -q sendmail
sendmail-8.11.0-8
# rpm -q sendmail-cf
sendmail-cf-8.11.0-8
# rpm -q m4
m4-1.4.1-3
# rpm -q cyrus-sasl
cyrus-sasl-1.5.24-6
```

画面4　インストールされているかどうかの確認

パッケージ	Red Hat 7J収録Disc
sendmail-8.11.0-8.i386.rpm	Disc1
sendmail-cf-8.11.0-8.i386.rpm	Disc2
m4-1.4.1-3.i386.rpm	Disc1
cyrus-sasl-1.5.24-6.i386.rpm	Disc1

表2　SMTPサーバに必要なRPMパッケージ

```
$ su -
# rpm -ivh sendmail-8.11.0-8.i386.rpm
# rpm -ivh sendmail-cf-8.11.0-8.i386.rpm
# rpm -ivh m4-1.4.1-3.i386.rpm
# rpm -ivh cyrus-sasl-1.5.24-6.i386.rpm
```

画面5　RPMパッケージのインストール

えて、必ず「user@ascii-linux」という形式にするためのものだ。この設定を行わないと、SMTPサーバの名前が付加された、「user@mail.ascii-linux.com」といった形式となる。

以上でsendmail.mcファイルの設定は終了だ。早速、このファイルを元にsendmail.cfファイルを作成してみよう(**画面6**)。

sendmail.cfファイルが作成できたら、/etcディレクトリにコピーする。

```
# cp sendmail.cf /etc
```

以上でsendmail.cfファイルの設定は終了だ。

▼リスト3　sendmail.mc

```
divert(-1)
dnl This is the macro config file used to generate the /etc/sendmail.cf
dnl file. If you modify thei file you will have to regenerate the
dnl /etc/sendmail.cf by running this macro config through the m4
dnl preprocessor:
dnl
dnl        m4 /etc/sendmail.mc > /etc/sendmail.cf
dnl
dnl You will need to have the sendmail-cf package installed for this to
dnl work.
include(`../m4/cf.m4')
VERSIONID(`linux setup for Red Hat Linux')dnl
OSTYPE(`linux')
define(`confDEF_USER_ID',``8:12'')dnl
undefine(`UUCP_RELAY')dnl
undefine(`BITNET_RELAY')dnl
define(`confAUTO_REBUILD')dnl
define(`confTO_CONNECT', `1m')dnl
define(`confTRY_NULL_MX_LIST',true)dnl
define(`confDONT_PROBE_INTERFACES',true)dnl
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl
define('ALIAS_FILE','/etc/aliases')dnl
define(`STATUS_FILE', `/var/log/sendmail.st')dnl
define(`UUCP_MAILER_MAX', `2000000')dnl
define(`confUSERDB_SPEC', `/etc/mail/userdb.db')dnl
dnl define(`confPRIVACY_FLAGS', `authwarnings,novrfy,noexpn')dnl
dnl define(`confTO_QUEUEWARN', `4h')dnl
dnl define(`confTO_QUEUERETURN', `5d')dnl
dnl define(`confQUEUE_LA', `12')dnl
dnl define(`confREFUSE_LA', `18')dnl
define(`confAUTH_MECHANISMS', `LOGIN DIGEST-MD5 CRAM-MD5')dnl
TRUST_AUTH_MECH(`LOGIN DIGEST-MD5 CRAM-MD5')dnl
FEATURE(`smrsh',`/usr/sbin/smrsh')dnl
FEATURE(`mailertable',`hash -o /etc/mail/mailertable')dnl
FEATURE(`virtusertable',`hash -o /etc/mail/virtusertable')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(local_procmail)dnl
FEATURE(`access_db')dnl
FEATURE(`blacklist_recipients')dnl
FEATURE(masquerade_envelope)
MASQUERADE_AS(ascii-linux.com)
dnl We strongly recommend to comment this one out if you want to protect
dnl yourself from spam. However, the laptop and users on computers that do
dnl not hav 24x7 DNS do need this.
FEATURE(`accept_unresolvable_domains')dnl
dnl FEATURE(`relay_based_on_MX')dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

①

②

SMTPサーバのエイリアス

sendmail.cfが作成できたら、メール配信の準備ができたことになる。ただし、この状態では、メールアドレスにSMTPサーバ名を付加した、「user@mail.ascii-linux.com」といった形式でないとメールが受信できない。そこで、「user@ascii-linux.com」といった形式のメールアドレスでもメールが受信できるよう、SMTPサーバのエイリアスを設定しておこう。

SMTPサーバのエイリアスは、/etc/mail/local-host-namesに記述する。**リスト4**のようにドメイン名を記述すれば「ユーザー名@ドメイン名」形式のメールを受信することができるようになる。

SASL認証ライブラリの設定

SASL認証ライブラリは、外部からメールを送信する際に正規のユーザーかどうかをチェックするために利用する。SASL認証ライブラリの設定自体は、インストールするだけで特に変更する必要はない。ただし、sendmailでの認証方法を指定するために、/usr/lib/sasl/Sendmail.confに**リスト5**のように記述する。

この記述は、パスワードチェックにSASLライブラリを使用するという意

味だ。Sendmail.confファイルは、「S」が大文字なので注意しよう。

■ユーザーの作成

SMTP AUTHを利用するためには、SASL認証ライブラリデータベースにユーザー名とパスワードを登録しなければならない。これは、ログイン時に利用されるパスワードとは別のもので、SASL認証ライブラリで認証するためのものだ。なお、ユーザー名とパスワードを登録するには、root権限が必要だ。

SASL認証ライブラリデータベースにパスワードを登録するには、saslpasswdコマンドにユーザーを指定し、パスワードを2回入力する。

```
# saslpasswd username
```

SASL認証ライブラリデータベースは、/etc/sasldbに暗号化されて保存される。

パスワードを変更する場合もユーザーの作成と同じように行う。

sendmailの起動とテスト

ここまで設定できたら、sendmailに関する設定は終了だ。sendmailデーモンを再起動しよう。

```
# /etc/init.d/sendmail restart
```

または、

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

この状態でメールが正常に配信できるか確認しておこう。

まずは、sendmailでローカルユーザー宛のメールがきちんと送信できてるかどうか確認しよう。sendmailにユーザー名を指定すればメール送信のテストができる（**画面7**）。なお、「-v」はsendmailの送信状態を冗長表示させるためのオプションだ。

画面7では、user1にメールを送信している。「test」はメールの本文となる文字列で、ピリオド（.）を入力するまで本文を入力することができる。

メールの送信が完了したら、送信したユーザーに正常にメールが送信されているかどうかをmailコマンドを使って確認しよう（**画面8**）。

正常にメールが受信できていれば、「1」と入力することでメールを読むことができるはずだ。メールがきちんと受信できていたら、「x」と入力してmailコマンドを終了しよう。

次は、外部に送信できるかどうかのテストをしてみよう（**画面9**）。

今度はsendmailが相手のSMTPサーバと通信しているようすが表示されるはずだ。正常に送信できていたら、今度は外部からメールを送って正常に受信できるかどうか確認しておこう。

メールの配信が正常に行われていることが確認できたら、ntsysvコマンド（Turbolinuxではturboservice）コマンドで、sendmailがシステム起動時に有効になるように設定する（**画面10**）。

不正中継のテスト

現在のsendmailでは、デフォルトでメールの不正中継を禁止しているが、実際にメールの不正中継が禁止されているかどうかを検証しておいたほうがいいだろう。

メールの不正中継が禁止されているかどうかは、プロバイダなどに接続してLANの外部から実際にメールを送信してみるのが確実だが、WIDE Projectのホームページにある、チェッカーで調べることも可能だ（**画面11**）。

Webブラウザで**表3**のWebページにアクセスし、「ホスト名」に自分の

```
# make sendmail.cf
m -f sendmail.cf
m4 ../m4/cf.m4 sendmail.mc > sendmail.cf || ( rm -f sendmail.cf && exit 1 )
chmod 444 sendmail.cf
```

画面6　sendmail.cfファイルの作成

▼リスト4　SMTPサーバのエイリアス（/etc/mail/local-host-names）

```
# local-host-names - include all aliases for your machine here.
ascii-linux.com
```

▼リスト5　SASL認証ライブラリの指定（/usr/lib/sasl/Sendmail.conf）

```
pwcheck_method: sasldb
```

```
# sendmail -v user1
test
.
user1... Connecting to local...
user1... Sent
```

画面7　ローカルメール配信のテスト

```
# su - user1
$ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/user1": 1 messages 1 new 1 unread
 N  1 root@ascii-linux.com  Sat Jan 27 19:09  10/383
&
```

画面8　ローカルメールの受信確認

SMTPサーバ名を入力し、「そのホストのユーザーを偽装」をチェックして、［SUBMIT］ボタンをクリックする。これで自分のSMTPサーバがメールの不正中継を禁止できているかどうかが確認できる。「不正転送を拒否します」と表示されればOKだ。SMTPサーバを構築したら、必ずテストしておこう。

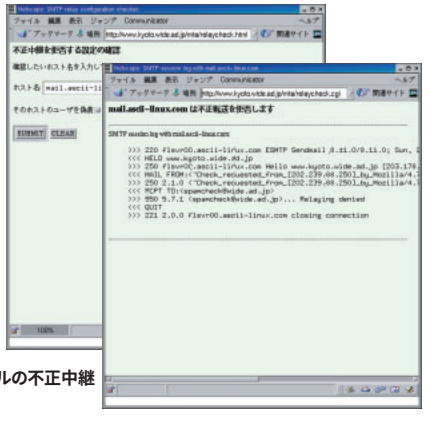
```
# sendmail -v user@hogehoge.co.jp
test
.                                          mail.hogehoge.co.jpと通信開始
user@hogehoge.co.jp Connecting to mail.hogehoge.co.jp. via esmtp...
220 mail.hogehoge.co.jp ESMTP Sendmail 8.9.3/3.7W; Sat, 27 Jan 2001 19:08:03
+0900 (JST)
>>> EHLO mail.ascii-linux.com
250-mail.hogehoge.co.jp Hello IDENT:root@mail.ascii-linux.com
[111.122.133.144], pleased to meet you
250-EXPN                                   mail.ascii-linux.comからの送信
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
>>> MAIL From:<root@ascii-linux.com> SIZE=5
250 <root@ascii-linux.com>... Sender ok
>>> RCPT To:<user@hogehoge.co.jp>
250 <useru@hogehoge.co.jp>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 TAA21691 Message accepted for delivery
user@hogehoge.co.jp... Sent (TAA21691 Message accepted for delivery)
Closing connection to mail.hogehoge.co.jp.
>>> QUIT
221 mail.hogehoge.co.jp closing connection
```

From:が書き換えられて「ascii-linux.com」になっている

送信終了

画面9　外部あてメールの配信テスト

画面10　ntsysvコマンドによるsendmailデーモンの起動



画面11　メールの不正中継のテスト

POPサーバとIMAPサーバ

POPもIMAPも、SMTPサーバに届いたメールを、SMTPサーバ以外で読むための仕組みだ。SMTPサーバのメールスプールにあるメールを、ローカルマシンのメールボックスにすべて転送する。一方、IMAPはSMTPサーバのメールボックスを利用し、必要なメールだけを受け取ることができる。IMAPは、外出先で必要なメールだけを受信することができるので、通信費を節約できることと、メールボックスを1つにまとめられることにある。

ただし、メールをSMTPサーバのメールボックスに保存するため、SMTPサーバのディスクを圧迫する可能性がある。運用時はディスクの空き領域にも十分注意する必要がある。

編集部のサーバでは、POPサーバとIMAPサーバの両方を構築することにした。といっても、今回利用するIMAPのパッケージには、POPサーバとIMAPサーバの両方が含まれているので、一緒にインストールされる。

必要なパッケージは、次のものだ。

```
imap-4.7c2-12.i386.rpm
```

IMAPのパッケージは編集部のサーバにインストールされていなかったので、Red Hat 7JのDisc 2からインストールすることにする。

```
# rpm -ivh imap-4.7c2-12.i386.rpm
```

なお、Red HatのFTPサイトからでも入手することができる。

POPサーバや、IMAPサーバはセキュリティのため、スーパーサーバから起動するのがふつうだ。スーパーサーバから起動されるサーバは、sendmailのようにデーモンとして常に起動しておくのではなく、必要になった場合にのみスーパーサーバが起動する。

それでは、スーパーサーバからPOPサーバとIMAPサーバが起動できるよ

うに設定しよう。

Red Hat 7Jでは、従来のinetdに代わる、xinetdがスーパーサーバとして導入されている。xinetdでは従来のように、/etc/inetd.confファイルで設定するのではなく、/etc/xinetd.dディレクトリにある、各サーバの設定ファイルを変更することで行う。

ここで変更が必要となるのは、ipop3と、imapの2つのファイルだ。それぞれ**リスト6・7**のように「disable」行を変更して、サーバの起動を許可する。

imapとipop3の2つのファイルを変更したら、スーパーサーバを再起動する。

```
# /etc/init.d/xinetd restart
```

または、

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

以上でPOPサーバとIMAPサーバが構築できた。

なお、従来のinetdを使用しているのであれば、/etc/inetd.confにある、「pop-3」と「imap」で始まる行の #（コメント）を削除しておく。スーパーサーバの再起動は、次のようにする。

```
# /etc/init.d/inetd restart
```

または、

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

■ POPとIMAPの認証

POPサーバの認証は、クリアテキスト（平文）で行われる。これはセキュリティ上、問題がある。そこで、POPサーバの認証に暗号化された方式を採用することにしよう。POPの認証にセキュリティを付加したものは、APOPと呼ばれる。APOPでは、パスワードを暗号化して送信するので、セキュリティが強化される。

今回使用するIMAPパッケージは、APOPによる認証もサポートしている。

APOPを利用するには、/etc/cram-md5.pwdを設定する。cram-md5.pwdに、ユーザー名とパスワードを記述することで、APOPによる認証が行われるようになる（**リスト8**）。なお、ここで設定したパスワードは、IMAPの認証時にも使用される。

ただし、cram-md5.pwdに記述するパスワードはクリアテキストなので、十分注意する必要がある。ファイルのパーミッションは、root以外が参照できないように400などに設定しておく。さらに、ログインパスワードなど、ほかのパスワードとは違うものを設定するようにしたほうがいいだろう。仮にこのファイルが見られたとしても、サーバにログインされる危険がないからだ。

▼リスト8　パスワードの設定（/etc/cram-md5.pwd）

```
user1 k2hab926
rina  diff2pass
```

http://www.kyoto.wide.ad.jp/mta/relaycheck.html

表3　メールの不正中継が確認できるWebサイト（WIDE Project）

▼リスト6　ipop3サーバが起動するように変更（/etc/xinetd.d/ipop3）

```
# default: off
# description: The POP3 service allows remote users to access their
mail \
#              using an POP3 client such as Netscape Communicator,
mutt, \
#              or fetchmail.
service pop3
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/ipop3d
        log_on_success          += USERID
        log_on_failure          += USERID
        disable                 = no
}
```

yesからnoに書き換える

▼リスト7　IMAPサーバが起動するように変更（/etc/xinetd.d/imap）

```
# default: off
# description: The IMAP service allows remote users to access their
mail using \
#              an IMAP client such as Mutt, Pine, fetchmail, or
Netscape \
#              Communicator.
service imap
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/imapd
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        disable                 = no
}
```

yesからnoに書き換える

箱の中のペンギンたち

文：みわよしこ
Text : Yoshiko Miwa

第2回
組み込みLinux製品の現在

「開発」という言葉には、一般に「最先端技術」「クール」「カッコいい」といったイメージがあるようです。また、「開発」が公害や製品事故に関連すると「人間性に反する」「企業の利潤のための」、ハイテク製品の急激な進化に関連すると「本来の人間の生活には不要な」といったイメージにつながることもあります。これらを強引にひとくくりにしてしまうと、「人間的でない」「通常の人間の生活から離れた」といった共通点が浮かびあがります。

しかし実際には、世の中に存在するすべての製品には「開発者」という名の人間が関与しています。一般のイメージがどうであれ、開発は人間の仕事です。設計ツールがいかに発達しようが、生産のオートメーション化がいかに進もうが、開発が人間の仕事であることは変わりありません。

本連載の第1回では、「組み込みとは？」ということについて解説し、続く第2回では組み込みLinux製品のいくつかを、開発中のものも含めて紹介しました。今回は、りぬくす工房の製品、「防犯カメラサーバ」を題材に、組み込みLinux製品開発の実際を紹介します。

「防犯カメラサーバ」の開発目的

不況が長引くとともに、小額現金を狙った犯罪が増えてきました。特に、自動販売機を壊して売り上げや釣り銭を盗む事件は、もはや珍しい事件ではなくなっています。現金が盗まれることのダメージはもとより、修理のコストも無視できません。最悪のケースでは、販売機が持ち去られることもあります。1台数百万円の販売機が盗難にあうことは、小さな商店などにとっては壊滅的なダメージになりえます。もちろん、現金が盗まれなくても、酔っぱらいや、荒れる青少年が叩いてへこませたり、傷をつけたり、落書きをしたりする場合にも修理費が必要になります。

自動販売機が開けられたり、移動されたり、大きな衝撃を受けた時に、警告音を発する防犯システムは、すでに一般的になっています。しかし、その音に誰かが気付かない限り、防犯の役には立ちません。防犯カメラを設置して録画することも広く行われていますが、最近ではコトが起こる前に周到な犯人によって、ビデオカメラの電源が切られることも少なくありませんし、カメラや録画装置まで持ち去られてしまってはお手上げです。

このような犯罪に対抗しえる防犯システムが「防犯カメラサーバ」なのです。

この防犯カメラサーバは、自動販売機に振動が加わった時、CCDカメラで周囲のようすを撮影し、その画像が添付されたメールを作成し、接続したPHSなどからダイヤルアップでインターネットに接続し、このメールをオーナー（自動販売機の管理者）に送付するシステムです（**図1**）。ダイヤ

ルアップ着信を可能にしておけば、遠隔地から設定を行うことも可能です。

簡単なシステムではありますが、これだけのことで万一の場合の対処はだいぶ容易になります。万一、防犯カメラサーバごと自動販売機本体が持ち去られてしまったとしても、オーナーに送られた写真に犯人が映っていれば、犯人の特定は容易になるでしょう。

また、防犯以外の応用の可能性も数多く秘めています。たとえばこの機能を応用し、適切なセンサーと組み合わせることで、釣り銭切れや、商品の売り切れをオーナーにメールで通知するシステムを追加することもできます。さらに、自動販売機での利用に限らず、「病弱な老人と離れて暮らさざるを得ない子供所帯が親のようすを知る」など、遠く離れたところでようすを知りたいというニーズに広く応えるポテンシャルを持っています。

開発に着手するまで

ある自動販売機メーカーが、「防犯カメラサーバを開発してもらえないか？」という話をりぬくす工房に持ち込んだのは、昨年10月のことでした。そのメーカーは、自動販売機オーナーのニーズに応えるため、このような製品を開発したいと考え、必要な技術を保有しているりぬくす工房に話を持ちかけたのです。

これは「受注開発」と呼ばれる形態で、この場合は通常、開発費は発注元が支払い、開発された製品は発注元の製品となります。

話を受けたりぬくす工房は内容を検討し、自社製品として開発することにしました。この場合は開発費が発注元から出るわけではないので、開発コストは製品の販売でまかなうことになります。営業方法次第では、受注開発より大きな利益を得ることができますが、開発コストを確実に回収できるという保証はありません。りぬくす工房が自社製品としての開発にこだわったのは、もともとりぬくす工房の方針である、「開発元として、製品に最終的な責任を持てない受注開発をなるべくやらないようにしている」ことからでもあります。

仕様の決定

開発に着手することが決定したら、次にすることは仕様の決定です。仕様を決定する手順は多様ですが、この防犯カメラサーバの場合は以下の手順で決定されました。

■ハードウェア

防犯カメラサーバの場合、動作は単純です。入力としてはカメラの画像があり、出力は画像を添付したメールです。

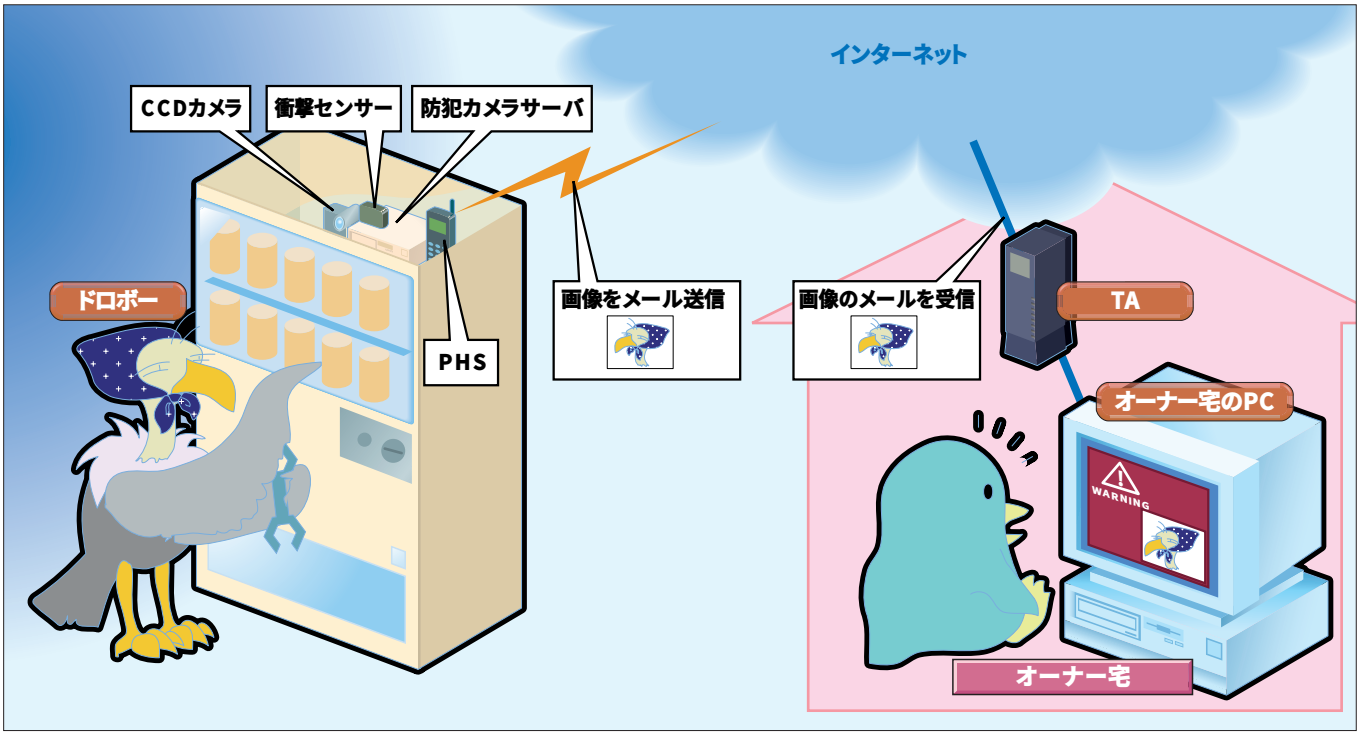


図1　防犯カメラサーバのシステム

これらの動作はすべてパソコンで行うことが可能です。しかし、この防犯カメラサーバの場合、自動販売機の中に組み込めるサイズである必要がありました。従って完成品のパソコンではなく、ボード・コンピュータ（**写真1**）を利用することになりました。

さらに納期が短く、数量が100台と少ないため、市販のボード・コンピュータの中から用途に適したものを選択することになりました。もし、数量が1万台で、十分な作業時間が確保できる状況であれば、オリジナルのボードを開発するという選択肢もありました。

■アーキテクチャ

アーキテクチャは「USB接続のCCDカメラを利用したい」ということから大きく制約されることになりました。

民生品としてショップで安価に販売されているUSB接続のCCDカメラを利用すれば、製品のトータルのコストを抑えることが可能です。また民生品なので、有志によって開発されたLinuxのドライバが無償で利用できます。

ボードの選択に際しては、動作的には486CPUで十分ではあったのですが、USB接続をサポートしているPentiumを採用することになりました。記憶媒体にはコンパクト・フラッシュを採用しています。

■OS

「PHSを利用したダイヤルアップ発信・着信」「本体とともに電源をオフにしてもよい」「CCDカメラから画像を入力する」の3点から、Linuxが採用されました。

次にディストリビューションの選択ですが、この防犯カメ

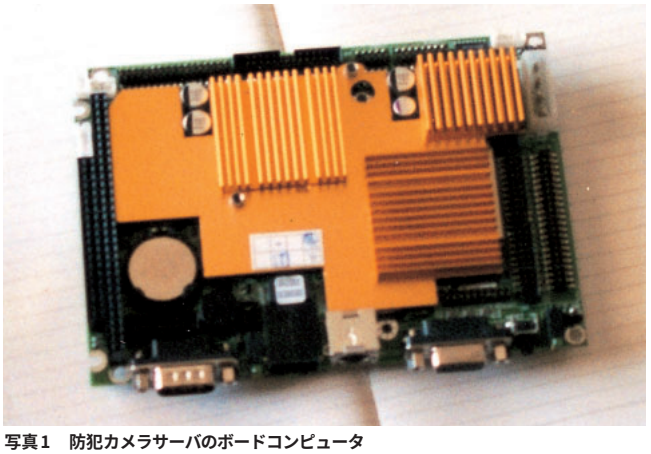


写真1　防犯カメラサーバのボードコンピュータ
組み込みに利用する場合は、大きさも重要なファクタとなる。

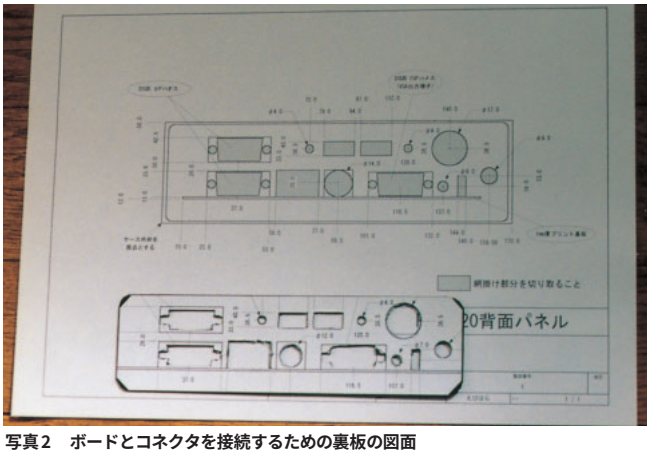


写真2　ボードとコネクタを接続するための裏板の図面
下はプラスチックの下敷きで作成したテスト用の裏板。

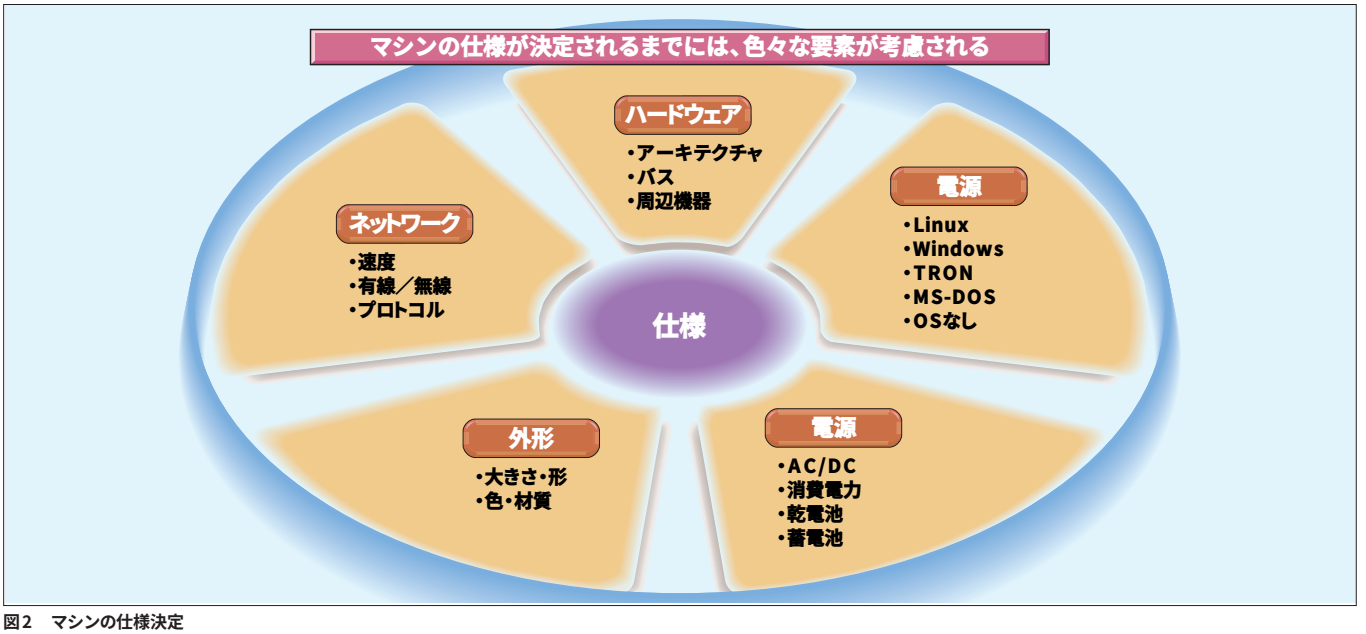


図2　マシンの仕様決定

ラサーバの場合はリアルタイム性がそれほど要求されないので、りぬくす工房社長の海老原祐太郎氏が組み込み用途向けに開発したディストリビューションである「Silicon Linux」から、リアルタイム対応部分を除いたものを採用することになりました。

■外形

たいていの場合は、筐体専門メーカーの既製品から適切な大きさとデザインのものを選択することになります。今回もそうでした。

ただし、実際にボードを組み込んで利用するには、筐体外からボードに附属しているコネクタにケーブルを接続するための加工が必要です。そこで、筐体は裏板を外すことの可能なモデルから選択されることになりました。そうすれば、裏板だけの加工を板金加工業者に依頼することが可能になるからです。

ボードメーカーからボードが届いたあと、裏板の加工のための図面作成が始まりました。

海老原氏は、ボードのコネクタ部分の寸法を定規で0.1mm単位まで計測し（通常、ボードメーカーはコネクタ部分の図面を提供していますが、今回採用されたボードにはなかったということです）、汎用ドローツールで加工のための図面を作成しました（**写真2**）。

念のためにプラスチックの下敷きを同寸法に切って穴を開けて寸法を確認したところ、思わぬ盲点がありました。コネクタのサイズより大きな孔が開けてあるからといって、そこにケーブルが接続可能とは限らないということです（**写真3**）。

ケーブルのコネクタは通常、厚いプラスチックで保護されています。接続に支障がないためには、その厚いプラスチックで保護された部分を筐体の中に入れる必要があります。海老原氏は孔のサイズを少し広げ、十分であることを確認したあと、板金加工業者に依頼を出しました（**写真4・5**）。

ソフトウェアの開発

防犯カメラサーバのソフトウェアは、Linuxともどもコンパクト・フラッシュに格納して利用することになりますが、開発はLinuxを搭載したごく普通のPCで行われました。

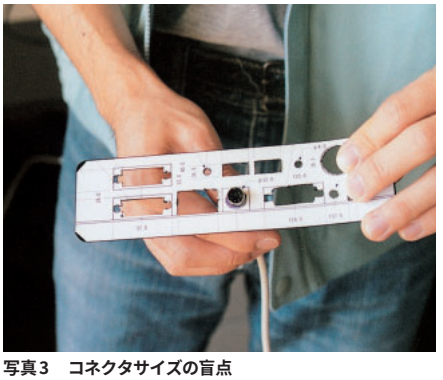


写真3　コネクタサイズの盲点
コネクタによっては、厚いプラスチックによって接続できないことがある。

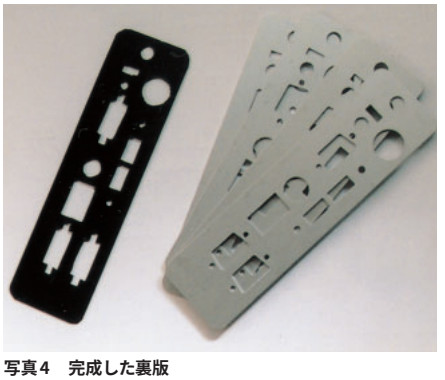


写真4　完成した裏版
綺麗に加工するために、板金業者に依頼。図面を基に加工したもの。

写真5　接続されたコネクタ
コネクタを保護するプラスチックの大きさを考慮することで、きちんと接続が可能となった。

写真6　完成した防犯カメラサーバ（前）
市販のカメラやボード、筐体を利用することで、コストダウンを図った。

写真7　完成した防犯カメラサーバ（後）
このシステムを自動販売機に組み込むことで、異常時に撮影した画像をメールでオーナーに送信することができる。

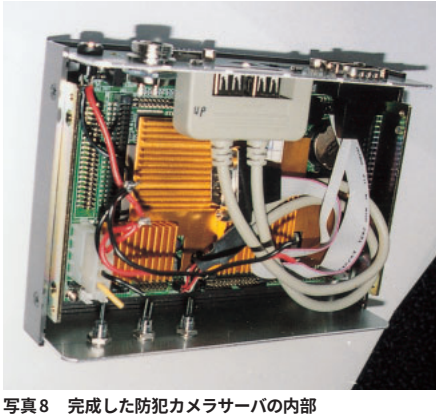


写真8　完成した防犯カメラサーバの内部
別々のパーツを組み合わせたとは思えないほど、ピッタリと筐体に収まっている。

開発されたソフトウェアは、CCDカメラから画像を取り込み、それをメールに添付し、ダイヤルアップ接続してメールを送信するスクリプトと、防犯カメラサーバの動作状態（電源オン・待機中・動作中）に従って、LEDを点灯させるスクリプトでした。今回はソフトウェアを使用するのがエンドユーザーでなく、自動販売機メーカーのエンジニアなので、スクリプトで十分ということになりました。エンドユーザーが利用する製品であれば、エンドユーザーが設定・保守を行うためのGUIシステムの開発や、マニュアルなどのドキュメント整備がもちろん必要になります。

なお、りぬくす工房の現在の社員は、社長の海老原氏とソフトウェア・エンジニアの岩崎剛氏の2人です。

岩崎氏は通常、りぬくす工房と遠く離れた自宅でソフトウェアの開発を行っています。りぬくす工房と岩崎氏の自宅はVPNで接続されているので、岩崎氏はりぬくす工房のPCに、ローカルにログインして開発作業を行うことができます。この防犯カメラサーバの動作テストのように、ソフトウェアの動作をハードウェアで確認する必要がある場合には、Microsoft Netmeetingを利用しています。海老原氏はハードウェアをりぬくす工房のPCに接続し、ハードウェアを電灯で明るく照らし、ハードウェアがCCDカメラに写るようにセットアップします。岩崎氏はりぬくす工房にあるハードウェアの動作状態を自宅のPCの画面で確認しながら、ソフトウェアの動作テストを行うのです。

インターネットを利用することで、このように遠隔地開発が行えるというのは非常に面白いところです。

ちなみに、エンドユーザーが自分自身の用途に合わせてこの防犯カメラサーバを利用できるよう、ブラウザでコントロールするためのソフトウェアを開発し、エンドユーザーに直接販売するという計画が、現在りぬくす工房で進行中です。

なお、今回のインタビューは、海老原氏が自分で起業して現在に至るまでの経緯や、日常考えていることについてうかがいました。

Column

アイデアがボードになるまで

大企業の場合、企画立案→開発→製品化は、ほとんど社内のみで行われます。また製品化を担当している部署が、企画立案にアイデアや意見を提供するといったことは（会社によりますが）非常に稀で、それぞれの部署が独立してその任務を果たすのが普通です。

しかし、中小企業での開発の形態は実に多様です。一個人や一企業のアイデアを、別の企業が製品化することも珍しくありません。アイデアの提供者は製品を利用することができ、製品化した企業は優れたアイデアを自社製品に取り込むことができるので、どちらにもメリットがあるのです。

さて、りぬくす工房の海老原氏は、小型・低消費電力（たとえばCPUはSH3）で、電池による駆動ができ、ネットワークOS（Linuxもそのひとつ）を搭載でき、もちろん入出力も附属しているボードがほしいと以前から考えていました。このようなボードがあると、たとえば工場の部品搬送用のロボットをLinuxで制御することが可能になります。

愛知県春日井市にあるりぬくす工房の近くには、優れた技術力を持つ中小企業が多数操業しています。その一社、愛知県春日井市のエーワンは、以前からOSレスで利用するためのボードコンピュータ「CAT68000」シリーズを、入出力ボード・コントローラボードなども含めて開発・生産・販売していました。

海老原氏が、エーワンに「CAT68000シリーズに、ネットワークOSが搭載できて、開発をC言語以上でやれるボードがあると嬉しいのですが……」と提案したのは、ちょうどエーワンが「そろそろCAT68000シリーズにも、OSを搭載できるラインナップを……」と考えていたころでした。OSを搭載できるボードで、既存のCAT68000シリーズの豊富な周辺機器が利用できれば、これまでになかった応用の方向も見込まれます。

海老原氏が持ち込んだ基本構成図、ブロック図をもとに、エーワンの技術者が詳細な回路を検証してボードを製作しました（**写真9**）。このボードはテストやマニュアル整備のあと、今年4月に販売される予定とのことです。

なお、海老原氏は個人ユーザーにも、この新しく開発したボードをぜひ利用してみてほしいと考えているそうです。関心のある方は海老原氏（ebihara@si-linux.com)に直接連絡してみてください。

写真9　海老原氏とエーワンで開発したボード
中小企業のアイデアが形となった製品だ。今年4月に発売される予定なので、興味のある方は海老原氏と連絡を取ってほしい。

りぬくす工房　海老原祐太郎氏に聞く

Interview

大企業＝安泰という図式が崩れ、一方でITベンチャーの華々しい発展が取りざたされる現在ですが、大企業のような安定もITベンチャーのような華やかさもないけれども、楽しく、なおかつ生活がそれほど不安定になるわけでもないワークスタイルがあります。
今回はりぬくす工房（愛知県春日井市）社長の海老原祐太郎氏に、「起業」に至った経緯や開発者・経営者としての日常を語っていただきました。

――2000年4月に、「りぬくす工房」を設立されて、順調に軌道に乗っているようですね。大企業を退職して起業されて、毎日の生活はどのように変化しましたか？

海老原：今はちょっと寂しいんです（笑）。会社員時代は本当に人には恵まれていて、ありがたかったです。辞める時に申し訳なかったくらい。

今は、岩崎君（本文参照）がこちらに来て作業している時以外は、1人で仕事していますからね。

――岩崎さんは、しばしばこちらに来られるんですか？

海老原：月のうち1週間くらいですね。彼にはソフトウェアの上位設計と、コーディングを主に担当してもらってます。ソフトウェアの開発はどこでもできますから、ふだんは彼は横浜の自宅で作業しているんです。実際にハードウェアと組み合わせての検証は、どうしてもハードウェアのある場所でやる必要があるので……。ここには彼が来る時用のベッドや布団もあるんですよ。彼は「組み込みLinuxじゃなくて、住み込みLinuxだ」なんていってますけど（笑）。

写真　りぬくす工房取締役社長・工場長 海老原祐太郎氏（えびはらゆうたろう）
1973年生まれ。信州大学工学部情報工学科を卒業後、1997年より三菱電機（株）名古屋工場で8086系マイコンを用いた産業機器のシーケンス制御に従事。一方で学生時代からLinuxに関心を持ち、ユーザーグループやコミュニティで幅広く活動している。2000年3月に三菱電機を退社、同年4月（有）りぬくす工房を設立した。現在は、りぬくす工房のハードウェア開発・庶務・営業・企画・経理、そして社長として多忙な日々を送る。

――失礼ですけど、収入面の変化は？

海老原：あまり変わりません。以前の勤務先がメーカーで、給与水準がそれほど高くなかったということもあるのですが。

――会社を辞めると、マイペースで生活できるというメリットはあると思うのですが。

海老原：うーん……ついつい明け方まで仕事して、午前中は寝てるというパターンになってしまいますね。あんまり良くないと自分でも思ってるんですが。

――私、耳が痛いです（笑）。

海老原：会社員時代は職場が工場でしたから、事業所全体がチャイムで動くんですよ。始業とか、食事とか、午後3時のラジオ体操とか。メリハリがありましたね。昼休みには同僚と話して息抜きできましたし。

――意外ですね。そういうのを「画一的だ」って嫌う方も結構いますから。ある程度の規律が、人間にはやっぱり必要なのかもしれませんね。海老原さんは、工場のそういう雰囲気が嫌いでなく、人間関係の悩みがあったわけでもなく、お仕事の内容もある程度はご自分の興味にかなうものだったのですよね？

海老原：はい。大学時代に制御工学に興味を持っていまして、就職して配属された職場でも産業機器のマイコン制御をやっていました。

――会社を辞める理由が見当たらない気がするんですが……。

海老原：2000年の初めごろ、「組み込みLinuxはここ1年が勝負」って思ったんです。

――会社の中で続ける可能性はなかったんですか？

海老原：直属の課長に、RT-Linuxの話をしたらすごく面白がってくれて、社内デモの場を作ってくれたんです。ほかの部署の部課長さんたちとか、10人くらい集めてくれて……。それでRT-Linuxが基礎技術として非常に優れたものであるということを実機デモもやってアピールしたんですが、あんまり仕事に活かせなくて……。

――デモ機は会社で用意してくれたんですか？

海老原：いえ、1999年の夏のボーナスで、7〜8万円くらいのボード・コンピュータを買って。それにRT-Linuxを載せて。

——自分で一歩を踏み出したわけですね。

海老原：そうですね。

——大企業だと、社内の異動で実質的に転職と同じことをする道もあると思うのですが。

海老原：ええ、以前の勤務先でも、研究所ではLinuxの研究をやっています。そこに異動するという選択肢は確かにありました。ですが、転属願いが受理されるまで3年くらいかかりそうでした。この1年が勝負だと思っていたので、3年なんて待てませんでした。

それで2000年3月に退職し、2000年4月に「りぬくす工房」を設立しました。

——新規に起業された方は仕事の確保に苦労されるようですが。

海老原：コンセプトマシンを作ってLinux Conference 99（1999年12月開催）で展示させてもらったりしていたところ、注目して自分にコンタクトしてくれる方がたくさんいたんです。これは本業にしても大丈夫だというメドがついたので退職した、ということもあります。

——夢があるからといって、カスミを食っては生きていけませんものね。海老原さんのお話をうかがっていると、自分が「良い」と思うものを、他人に提示することで次の道が開けているように見えるのですが。

海老原：そうですね。このコンセプトマシンを見てコンタクトしてきた方の中には、板金メーカーさんもいまして、自分がシャーシに開けたスリット（**写真11**）の端が波打ってるのを見て、「こんなことならウチが手伝えるよ」って声をかけてくださったんです。

——何も、完璧なものを提示する必要はないんですね。

海老原：ええ。特にウチも含めて小さい企業では、何もかもを自分のところでやるわけにはいきませんから……。作りたいものを作る方法、実現する方法を考えることも業務のうちなんです。

基板が剥き出しでリード線で止めつけた部品が空中に浮いているようなものでも、とにかく動くものを自分ができるところまで作って、技術パートナーになってくれそうな相手に見てもらって、デモして「ね、動くでしょ、やりましょうよ」と説得することは、しばしばやっています。

——積極的ですね。「機械とだけ付き合っていたい」という技術者は少なくありませんが。

海老原：いや、自分はそんなことはないですね。現場にももっと積極的に行って、現場を見たり、現場の人のニーズを聞いたりしたいです。今は受託業務で、いわゆる下請けの立場になることが多いので難しいんですが。

——技術も結局は、「社会の中で使ってもらってなんぼ」ですからね。

海老原：ええ。自分も世の中の動きを知るために、新聞は毎日読むようにしています。たとえば1人暮らしの老人の孤独死を防いだり、早期発見したりするために「トイレのドアと冷蔵庫の扉が1日動かなかったら、非常事態として地域の保健所や肉親に通知する」というシステムを開発した人がいるんです。

——普通は「老人の身体にセンサーを取り付けて、大きな変

画面　りぬくす工房のホームページ
http://www.si-linux.com/

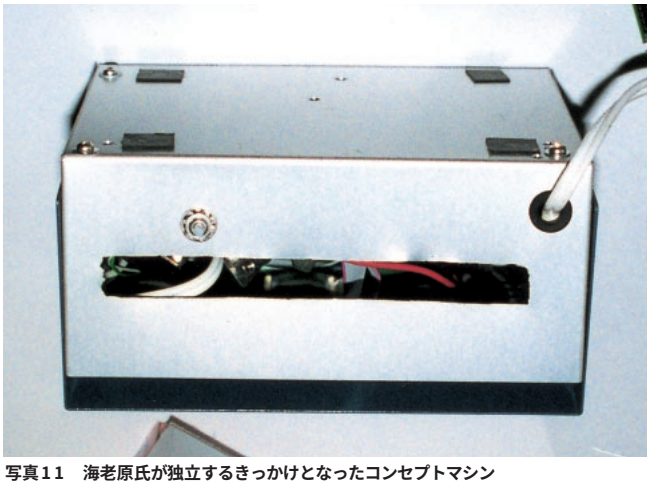


写真11　海老原氏が独立するきっかけとなったコンセプトマシン
写真ではわかりにくいかもしれないが、スリット部が波打っている。

化があったら第三者に知らせて……」という発想になりがちですが、それは実用的でいいですね。

海老原：それと同じような意味で、新聞記事を自分で見ていると「これなら解決できる」と思うことがたくさんあるんです。これなら、自分の技術でソリューションを提供できると。

――企業に勤務していると「言われた仕事をこなす」が全部になりがちですし、それだけで通用する場面も少なくありませんが、それでは起業はできないということですね。

海老原：もちろんです。ただ、最近のベンチャーには「自分の考えているのとはなんか違う」と感じることが少なくないです。

――といいますと？

海老原：特許制度というのがありますが、特許というのはある意味資金力の勝負になってしまいます*。ウチみたいな小さい会社は、資金面からそんなにたくさん特許を申請することはできません。アイデアと技術を守る事はとても大切なことですが、小資本のベンチャーでもフェアに渡り合えるルールができてほしいと思います。

――アイデアや技術の勝負のはずが、資金力の勝負になってしまうのはどこかおかしいですよね。

海老原：ベンチャーなんだから、もっと幅広く出資を募って、いずれはナスダックやマザーズに株式を公開しなきゃダメじゃないかと言われることも時々あります。しかし、株式公開を目標にして会社を大きくしなければいけないのでしょうか。今はたった2人でやって、1人が倒れたら会社のパワーが50%になるわけだから、もう少し安定した体制を作りたいとは思っていますよ。でも、私が考えるベンチャーとは、たとえば同じフロアで同じ仕事で働いている仲間にしても「社員」ではなく、各個人がそれぞれ自分の会社の「社長」という集団のほうがよりベンチャーらしいと思っています。

――会社が大きいことにはそれなりのメリットはありますけどね。

海老原：大企業が良くないとは思わないんです。たとえば、広域ネットワークインフラのような国家プロジェクトは、やはり大企業が主役になって進めるべきでしょうしね。

ただ大企業と中小企業は、どちらが上でどちらが下ということもないと思うんです。今は元請け・下請けという形で上下関係ができてしまうことがやはり多いですけど、将来的には「役割の違う技術パートナー」という形で、大企業と中小企業が協力できるようになってほしいですね。

――海老原さんの方向性は、いわゆる「ITベンチャー」とは全く違いますね。

海老原：ええ。自分たちは開発をやっていきたいから起業したんです。面白いことがやりたいんです。将来もずっとそうでありたいですね。

――最後に、組み込みLinuxの世界の将来への希望を聞かせてください。

海老原：面白い世界なので、もっと盛り上がるといいなあと思っています。もっとたくさんの企業が参入してきて、それぞれの特色を活かして、互いに技術パートナーとして協力し合えるような関係になるといいですね。

――Linuxがこれまでの社会や企業のあり方を変えてしまうかもしれませんね。どうもありがとうございました。

*特許出願には、書類作成・手続きをすべて自分でやる場合で、特許庁への出願料として2万1000円、弁理士に書類作成・手続きを依頼する場合で30万円程度が必要。特許として有効になる（特許庁に審査請求を行い、有効な特許と認められ、さらに一定期間の間に他者からのクレームが発生しない）ためには、さらなる費用と労力が必要になる。

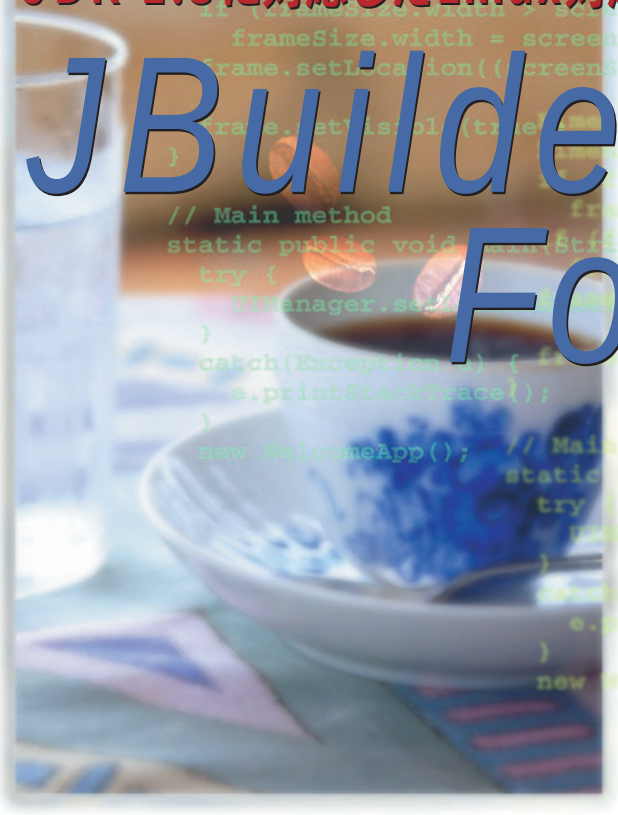
写真12　りぬくす工房の看板
りぬくす工房はマンションの一室で操業している。パート・アルバイト募集中だそうだ。特典は、インターネット使い放題（笑）。

写真13　りぬくす工房に置かれた工具類
これらの工具を使って、プロトタイプマシンやコンセプトマシンを加工している……のだろうか。

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3回　Java言語と親しもう

文：加藤大受
Text：Daiju Kato　dkato@jcom.home.ne.jp

JBuilder 4 Foundationのビジュアルデザイナを利用してユーザーインターフェイスを作成することができます。しかし、Java言語の持つ可能性を理解するには、Java言語の特性を知る必要があります。開発ツールを使っていきながら、少しずつJavaの持つ広大な世界を探検していきましょう。

Java言語はオブジェクト指向プログラミングに対応したプログラミング言語です。オブジェクト指向プログラミングをよくOOPと書きます。これは、Object Oriented Programmingの頭文字を取っています。Java言語の解説書を読むと、必ずといっていいほど前半部分でオブジェクト指向プログラミングについて解説しています。では、Java言語を使用するためには、必ずオブジェクト指向プログラミングの知識が必要なのでしょうか。筆者の見解としては、これはYesでもありNoでもあると思います。あなたがプログラマーでJava言語を利用してプログラムを書くのであれば、必ずオブジェクト指向プログラミングとは何かをマスターすべきでしょう。しかし、あなたがこれからプログラムをやっていこうと思ってJava言語に取り組んでいるのであれば、言語に親しんでさらに細かい内容に触れたいと思ってから、オブジェクト指向プログラミングを勉強していくのがいいでしょう。筆者の経験では、小さなプログラムを作っていきながら、少しずつオブジェクト指向プログラミングを勉強していくのが近道だと思います。どちらにしても、ある程度の規模のプログラムを書く場合には、オブジェクト指向言語を使っている限り、オブジェクト指向プログラミングの知識が必要になってきます。

オブジェクト指向プログラミングとは

では、オブジェクト指向プログラミングとは何でしょうか？

簡単にいってしまうと、プログラムの再利用性を高め、よりメンテナンスのしやすいプログラムを書くことを前提にしたプログラム手法で、オブジェクトという単位で特定の処理をカプセル化し、ほかの処理に影響を受けないように設計されたオブジェクト指向言語を利用してプログラミングを行うことです。

Java言語の場合、クラス、メソッド、プロパティ、インターフェイスというオブジェクト指向の用語をよく使用します。クラスはいわばオブジェクトを作成するためのテンプレートみたいなものです。クラスからオブジェクトを作成するのですが、この動作をインスタンスを作るといい、Java言語の場合だとnewという演算子を利用してインスタンスを作成します。次のコードは、ClassAクラスの新しいインスタンスとして、newClassAというオブジェクトを作成していることを意味しています。

```
ClassA newClassA = new ClassA();
```

オブジェクト内のデータを操作する場合は、メソッドと呼ばれる関数を利

用して操作します。直接データ操作をしないことにより、オブジェクトがカプセル化され、プログラムの安全性を高めることができます。

Java言語では、新規にクラスを作成できるだけでなく、既存のクラスを利用して新しいクラスを作成することができます。このとき、既存のクラスを基底クラスといい、基底クラスを利用して新しいクラスを作成することを継承といいます。継承したクラスは基底クラスに用意されているさまざまな関数を使用することができるだけでなく、独自に拡張していくことができますので、新しい処理に合った形に修正することが可能になるわけです。既存の資産をうまく活用していくことができますので、1からすべてを用意する場合に比べ、生産性が高く、また各オブジェクトがカプセル化されていますので、メンテナンス性も高くなるというわけです。

しかし、オブジェクト指向プログラミングは、全体のオブジェクトの設計がしっかりしていないと活かすことができなくなり、非常に生産性が悪くなる可能性もあります。また、オブジェクト指向に関する非常に高い知識を必要としますので、それほど使いこなしている人がいないのが実状です。Java言語は、ほかのオブジェクト指向の言語、たとえばC++言語などよりもわかりやすく設計されていますので、初心者でも理解しやすくなっています。また、JBuilderを始めとする、さまざまなビジュアル開発ツールなどにより、プログラム開発の敷居が低くなり、プログラム初心者でも手軽にJavaを利用したプログラミングが可能になってきています。

しかし、Javaを利用して大規模システムを開発している人々の半数以上はJDKを利用していると言われており、まだまだ開発ツールはユーザーインターフェイス部分の開発に特化しているように思われがちですが、このような開発ツールを効率よく使用していくことで、生産性を高めることができるでしょう。

Javaのデータ型

すでに簡単なサンプルプログラムを作成していく中で、Javaのデータ型について触れてきましたが、データ型について詳しく説明していきましょう。

Javaはすべての変数が、宣言されたある型を持つ言語です。変数の定義には型の指定が必要となります。Javaには8つの基本型があり、そのうち6つは数値型、1つはUnicodeコーディングで使われる文字型、もう1つは論理型です。6つの数値型には整数型が4つ、浮動小数点型が2つあります。

■整数型

整数型には小数を格納することはできませんが、負の値を格納することができます。C++などの言語では、動作するプラットフォームに整数型の範囲が依存しますが、Java言語は当初から複数のプラットフォームで稼働することを前提としているプログラミング言語のため、プラットフォームに依存せずに常に同じ範囲となります（**表1**）。

Javaの整数型では、長い整数にはLという接尾辞がつきます。また、16進数には0xという接頭辞がつきます。

■浮動小数点型

浮動小数点数型には、倍精度型と単精度型の2種類が用意されています（**表2**）。単精度では精度が足りなくなるため、一般的には倍精度を利用します。単精度は、多くの数値を保存する場合や、高速なパフォーマンスを要求するときに利用します。

■文字型

Javaの文字型は、Unicodeと呼ばれる文字コードでエンコーディングされています。Unicodeは、各国の文字コードの違いを吸収するために用意されている文字コードで、Javaプログラムの内部では、文字コードはすべてUnicodeでエンコーディングされています。しかし、国際化のプログラムや、Web関連のアプリケーションを書かない限りは、それほどUnicodeが使われていることを気にする必要はありません。こういったプログラムを書く必要がある場合は、Javaの解説書でエンコーディングとデコーディングの処理について一読しておくことをお勧めします。特に、Web関連のJava技術を使用する場合には必要不可欠な知識となります。

Javaには文字型としてchar型が用意されていますが、この型は一般的に使用しません。これは、Javaには文字列

型	長さ	範囲
int	4バイト	-2,147,483,648～2,147,483,647
short	2バイト	-32,768～32,767
long	8バイト	-9,223,372,036,854,775,808L～9,223,372,036,854,775,807L
byte	1バイト	-128～127

表1　整数型

型	長さ	範囲
float	4バイト	およそ±3.40282547E+38F（6から7桁）
double	8バイト	およそ±1.79769313486231570E+308（15桁）

表2　浮動小数点型
*float型には接尾辞としてFがつきます。接尾辞がないときはdouble型と判断されます。Javaのすべての浮動小数点型はIEEE-754仕様に準拠しています。

定数のためのStringクラスと、文字列の内容を変えることのできるStringBufferクラスが用意されているため、char型を使用しないでもプログラムの作成が可能だからです。

制御フロー

Java言語には、ほかの言語と同じようにある条件で分岐をしたり、条件に合っている間にループしたりする制御フローが用意されています。

■条件文

条件文はある条件に合ったときに実行されるブロックを指定することができます。また、条件を満たす場合と満たさない場合との処理を分けることが可能です。条件文は、**リスト1**のように記述します。

■不確定ループ

不確定ループは、条件を満たす間、ある特定の処理を繰り返し処理する場合に使用します（**リスト2**）。

最低1回はループを行いたい場合には、**リスト3**の繰り返し処理を使用します。

■確定ループ

確定ループは、ある条件下で、繰り返し処理を行う場合に使用します（**リスト4**）。

式1でカウンタ変数を初期化し、条件を満たしている間、ブロックの処理を行い、1回ループするごとにカウンタ変数の処理を実行します。

■複数選択

複数選択は、char型または整数型の変数に合う条件をすべて実行します（**リスト5**）。

breakを入れないと、条件に合った以降のすべての処理を実行しますので、注意が必要です。

アプリケーションとアプレット

Java言語で書かれたアプリケーションには、大きく分けてアプリケーションとアプレットの2種類があります。アプリケーションは単独で実行できるプログラムであり、アプレットはWebサーバからダウンロードされ、ブラウザ上で実行されるプログラムです。アプレットを使用するためには、Javaに対応したブラウザが必要になります。対応しているJDKのバージョンがブラウザによって異なるため、業務で使用する場合には事前の調査が必要となります。また、アプレットはネットワークからダウンロードされますので、回線速度によるダウンロード時間の考慮が必要となります。

アプリケーション、アプレット以外にも、ブラウザのリクエストに応じて、サーブレットエンジンという環境の上で実行されるサーブレットや、HTMLドキュメントの生成を行う、エンタープライズレベルでの利用を考えたEnterprise JavaBeans（EJB）という技術があります。また、家電などをネットワーク対応させ、コントロールすることを目的としたJINIなど、さまざまな技術が用意されています。Java技術を深く理解していくことで、Javaの持つ可能性の大きさを体感することができるでしょう。

画像ビューアの作成

前回は簡易電卓を作成してみました。前回まではユーザーインターフェイスの作成に重点を置き、ユーザーインターフェイスのレイアウトについて説明

▼リスト1　条件文

```
if（条件）{ ブロック }

または、

if（条件）{ ブロック1 }        条件を満たす場合の処理
else      { ブロック2 }        条件を満たさない場合の処理
```

▼リスト2　不確定ループ（do）

```
while（条件）{ブロック}
```

▼リスト3　不確定ループ（while）

```
do {ブロック} while（条件）;
```

▼リスト4　確定ループ

```
for（式1; 条件; カウンタ変数の処理）  {ブロック};
```

▼リスト5　複数選択

```
switch（char型または整数型の変数）
{
case <変数の値1>:
     ……
     break;
case <変数の値2>:
     ……
     break;
case <変数の値3>:
     ……
     break;
default:
     ……
     break;
}
```

してきましたが、今回は、プログラムの内容に重点を置いて作成します。今回作成するのは、GIFファイルとJPEGファイルに対応した画像ビューアです。画像ファイルを選択後、パネルに画像を表示し、画像ファイルのサイズに合わせてフォームのサイズを調整するものとします。

今回作成するプログラムの流れを簡単にまとめてみると**図1**のようになります。

今までは、文字やボタンなどのオブジェクトが主体となっていましたが、今回はグラフィックスデータを表示することになります。細かいプログラムの説明に入る前に、簡単にJavaアプリケーションのフレームを提供するJFrameクラスの構造について説明しておきましょう。

図2は、JFrameの構造を図にしたものです。JFrameは4つのペインと呼ばれる層を持っています。ルートペイン、レイヤペインは、メニューバーとコンテンツペインを整理するために用意されています。グラスペインは、ルック&フィールを実装するために必要とされます。ユーザーインターフェイスを作成するプログラマーにとって、もっとも重要なのがコンテンツペインで、ボタン、パネル、リストボックスなどのコンポーネントオブジェクトをコンテンツペインに配置していくことになります。パネル、つまりJPanelオブジェクトをコンテンツペインに配置する場合は、**リスト6**のようなコードとなります。しかし、これらのコードはJBuilderのビジュアルデザイナでパネルをフレームに配置することで、自動的に生成してくれますので、ほとんど意識する必要がありません。

パネルは、それ自体が文字列などを表示できるコンポーネントですが、パネルの上にボタンなど、ほかのユーザーインターフェイス要素を構成できるオブジェクトを配置することができます。このような、ほかのオブジェクトを管理できるものをコンテナと呼びます。

文字列などを表示する場合は非常に簡単なコードで済むのですが、画像を表示する場合はコードを少しきちんと書く必要があります。JBuilder Professional以上であれば、JBCLと呼ばれる、ボーランド・ソフトウェア・コーポレーションが作成したユーザーインターフェイスを構成できるコンポーネントが提供されているので、画像を表示するのも簡単なのですが、Foundationには提供されていないので、自分で作成するしかありません。

画像などのグラフィックを取り扱うには、Graphicsオブジェクトを使用します。これは、コンポーネント上やバッファにグラフィックスを描くために使われるクラスで、描画領域（クリップ領域）、色、フォントなどの、描画に関する情報を管理しています。描画を行う対象範囲がクリップ領域となります。つまり、コンポーネントを囲む矩形領域がクリップ領域となりますので、この領域についてはほとんど意識する必要がありません。

Graphicsオブジェクトは2種類の描画を行うことができます。線、四角などの図形や、文字などのプリミティブグラフィックスとイメージです。今回はイメージデータを扱いますので、イメージを扱う場合について説明します。

JDK 1.1の時代では、JPEGの取り扱

▼リスト6　コンテナの利用

```
Container contentPane =
    frame.getContentPane();
JPanel jPanel1 = new JPanel();
contentPane.add(jPanel1);
```

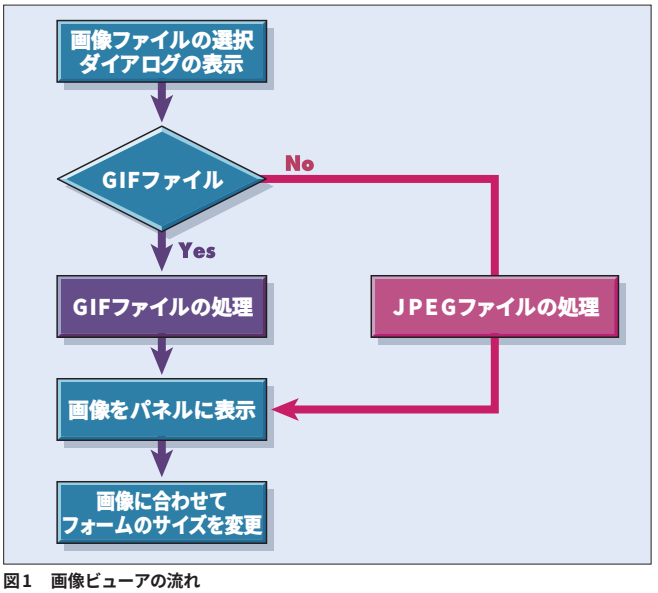


図1　画像ビューアの流れ

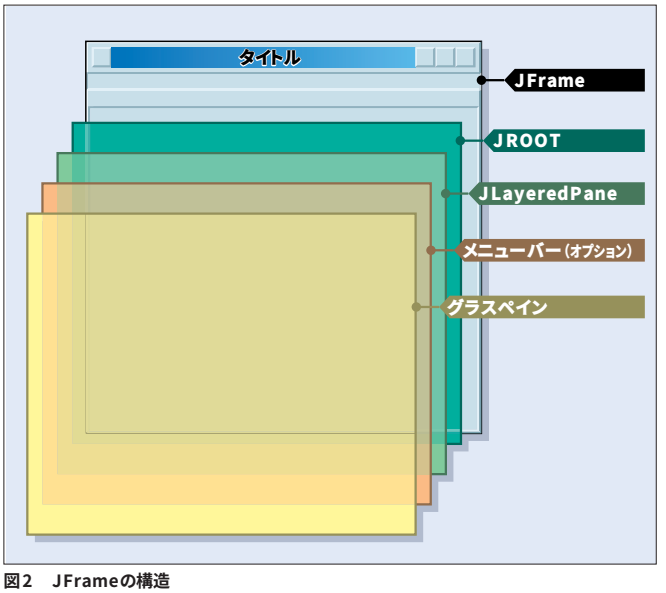


図2　JFrameの構造

いにはテクニックが必要だったのですが、JDK 1.2以降では非常に簡単にJPEGデータを扱うことができるようになりました。今回作成する画像ビューアでは、GIFファイル、JPEGファイルの2種類のフォーマットに対応するものとします。Javaプログラムの中でイメージを扱うには、Imageオブジェクトを使用します。格納されたイメージの情報は、Imageオブジェクトでカプセル化されることになります。

JPEGファイル、GIFファイルをJavaアプリケーションで読み込むには、Toolkitオブジェクトを利用して、Imageオブジェクトに格納します。画像ファイルの読み込みは非常に簡単で、GIFファイルの場合は**リスト7**のような2行のコードで済んでしまいます。

JPEGファイルの場合では、もう少し複雑な処理が必要となります。これは、JPEGファイルは圧縮されているフォーマットなので、一度解凍してからImageオブジェクトに格納しなければならないからです。そのため、解凍処理を行うためにファイルストリームにファイルの内容を格納し、解凍処理を行ってImageオブジェクトに格納することになります（**リスト8**）。

Imageオブジェクトに格納されたイメージファイルを表示するには、フレーム上にイメージデータを表示するコンポーネントが必要になります。イメージファイルを表示する場合、一般的にパネルオブジェクトを使用します。しかし、フレームにパネルオブジェクトを配置するだけでは表示することができません。JPanelを拡張するクラスを定義して、paintComponentメソッドを使用可能にしなければなりません。

paintComponentメソッドは、Graphicsオブジェクトを引数として受け取ることができ、受け取ったGraphicsオブジェクトの描画処理を行うことができるのです。つまり、JPanelクラスを継承した新しいクラスを作成して、paintComponentメソッドを使用可能にする必要があるのです。

それではプロジェクトファイル、フレームクラス、アプリケーションクラスを準備しましょう。

［ファイル］－［新規プロジェクト］でプロジェクトファイルを作成し、その後、［ファイル］－［新規］で表示されるオブジェクトギャラリから、［アプリケーション］を選択してアプリケーションウィザードを起動し、フレームクラス、アプリケーションクラスの雛形を作成しましょう。フレームクラスのオプションは、「メニューバーの生成」と「フレームを画面の中央に配置」をチェックします。

プロジェクトファイル名：

`FileViewer.jpr`

アプリケーションクラス名：

`FileViewerApp`

フレームクラス名：

`FileViewerFrame`

▼リスト7　GIFファイルの読み込み

```
String name = "imagefile.gif";
Image image = Toolkit.getDefaultToolkit().getImage(name);
```

▼リスト8　JPEGファイルの読み込み

```
String name = "imagefile.gif";
FileInputStream fileStream = new FileInputStream(name);
JPEGImageDecoder decoder = JPEGCodec.createJPEGDecoder(fileStream);
Image image = decoder.decodeAsBufferedImage();
```

JPanelを継承するImagePanelクラスの作成

プロジェクトファイル、フレームクラス、アプリケーションクラスができあがりましたので、JPanelを継承するImagePanelクラスを作成します。［ファイル］－［新規］表示されるオブジェクトギャラリで、クラスウィザードを起動します。**画面1**のように、クラス名に「ImagePanel」を入力し、親クラスに「javax.swing.JPanel」を入力して［OK］ボタンをクリックします。これでImagePanelクラスの雛形ができました。

雛形には、paintComponentメソッドがありませんので、まずこのメソッドを定義します。Graphicsオブジェクトを取り扱うには、親クラスが持っているグラフィックオブジェクトを引数に取れる、paintComponentメソッドを明示的に呼び出す必要があるため、superという、常に親クラスであるスーパークラスを意味するキーワードを付けてコンストラクタを呼び出します。

続いて、イメージオブジェクトに格納されたイメージデータを描画します。描画はdrawImageメソッドで行います。drawImageメソッドはイメージファイル、描画開始領域X位置、描画開始領域Y位置、描画状態を監視するオブジェクトという4つの引数を取ることができます。今回作成するパネルにイメージデータを表示するのですから、ImagePanelクラスにイメージファイルを表示する機能を装備するほうが汎用性が高くなります。ここでは、setImageメソッドを作成し、このクラスを使用するプログラムからイメージファイルを設定することができるようにします。

このクラス内では、イメージファイルの読み込みを行っておりませんので、明示的にイメージファイルを表示する必要があります。こういった処理を行う場合は、MediaTrackerクラスを使用するのが簡単です。

MediaTrackerクラスは、非同期的または、同期的なイメージのロードを指定することができ、複数のイメージを扱うときはID番号を付けて管理していくことができ、すべてのイメージがロードされるのを待つことも、特定のID番号を持つイメージだけをロードすることも可能です。今回は単純に1つのイメージですので、waitForAll()メソッドを使います。

ImagePanelクラスの全ソースは**リスト9**のようになります。今までの説明を読み返しながらソースを確認してみてください。

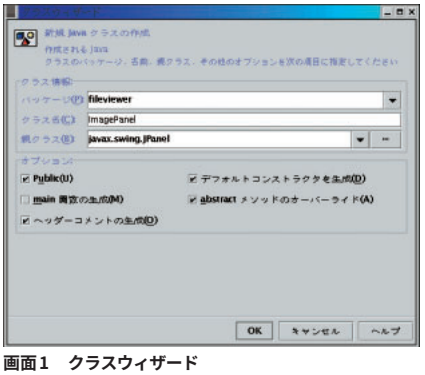
ユーザーインターフェイスの作成

画像を表示できるパネルができましたので、続いてユーザーインターフェイスを作成します。FileViewerFrameクラスを選択して、ビジュアルデザイナを起動してください。**画面2**のような画面が表示されます。コンポーネントツリーにメニューツリーが存在していることがわかると思います。

次に、コンポーネントツリーでjMenuBar1をダブルクリックし、**画面3**のようなメニューデザイナを起動してください。今回は、ヘルプメニューは必要ありませんので、コンポーネントツリーでjMenuHelpを選択して、Deleteキーを押して削除します。

続いて、メニューデザイナで「ファイル」を選択し、ファイルメニューを開きます。ここで「終了」を選択して、Insertキーを押すと、メニューアイテムを追加することができます。新しく作成したメニューアイテムに、「開く」という文字列を入れてください。**画面4**のように、ファイルメニューに「開く」というメニューアイテムを追加することができます。

続いて、メニューアイテム「開く」のnameプロパティを、jMenuItem1からjMenuFileOpenに変更します。「開く」を選択したときに、ファイル選択画面を表示させるようにさせるため、コンポーネントパレットの［Swing Containers］のページの右端にある、FileChooserコンポーネントを選択し、コンポーネントパレットのUIの上に配置します。コンポーネントパレットは**画面5**のようになります。続いて、contentPaneのレイアウトを



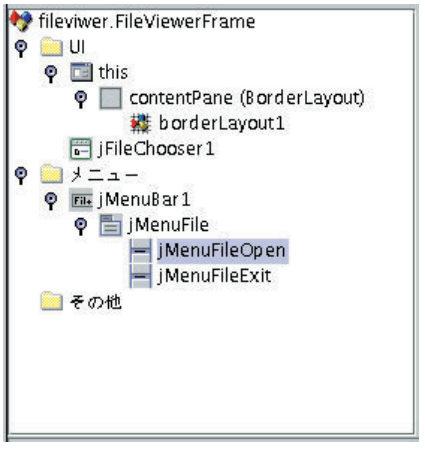
画面1　クラスウィザード

▼リスト9　ImagePanel.java

```
package fileviewer;

import javax.swing.JPanel;
import java.awt.*;

public class ImagePanel extends JPanel {

  private Image image;

  //コンストラクタ
  public ImagePanel() {
  }

  public void paintComponent(Graphics g){
     super.paintComponent(g);
     //イメージがセットされていないときは処理しない
     if (image!=null) {
        g.drawImage(image,0,0,null);
       }
   }

   public void setImage(Image image) {
     this.image=image;
     //MediaTrackerクラスを利用して明示的にイメージを処理
     MediaTracker tracker = new MediaTracker(this);
     //画像ビューアに表示するイメージを追加
     tracker.addImage(image,0);
     try {
       //イメージが表示されるまで待つ
       tracker.waitForAll();
     } catch (Exception e) {
       e.printStackTrace();
     }
   }
}
```

「CardLayout」に変更します。

ここでコードエディタに戻り、コンポーネントツリーでFileViwerFrameを選択します。**リスト10**のように、



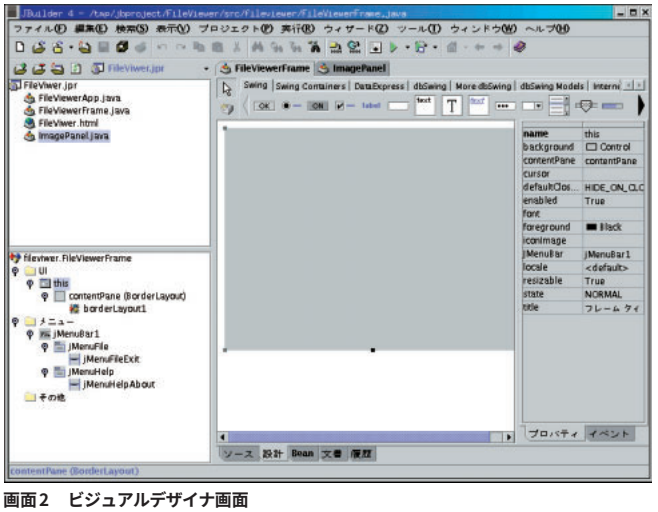
画面5　FileChooserを追加

ImagePanelクラスを継承するimagePanelオブジェクトを定義します。これで画像を表示可能なパネルを扱うオブジェクトが生成されました。続いて、コンポーネントツリーからjbInit()メソッドを選択し、**リスト11**のようにメソッドの最後に、フレームクラスにimagePanelオブジェクトを貼り付けるコードを追加します。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、全ソースをコンパイルします。コンパイル後、ビジュアルデザイナを起動すると**画面6**のようになります。

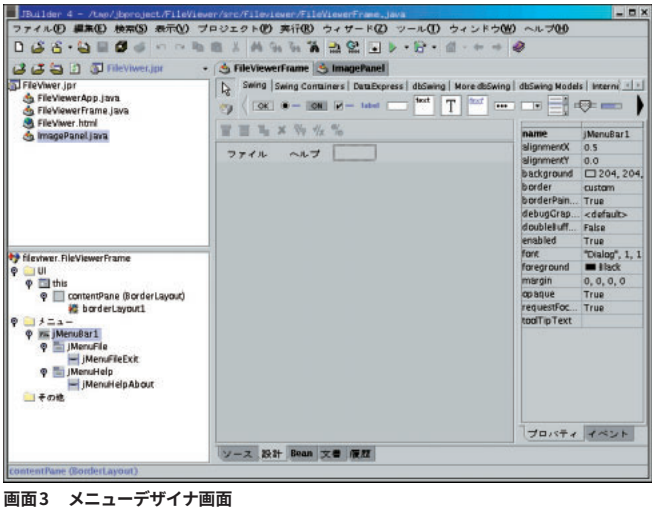
これで、イメージの表示部分ができたので、続いてファイルを選択してイメージを表示する部分を記述します。

コンポーネントツリーから、jMenuFileOpenメニューアイテムを選択して、インスペクタでイベントページを選択し、一番上のaction_Performedイベントをダブルクリックして、「開く」メニューアイテムがクリックされたときのイベントを記述します。開くメニューを実行すると、ファイル選択のダイアログボックスを表示し、GIFファイルまたは、JPEGファイルを選択可能とします。

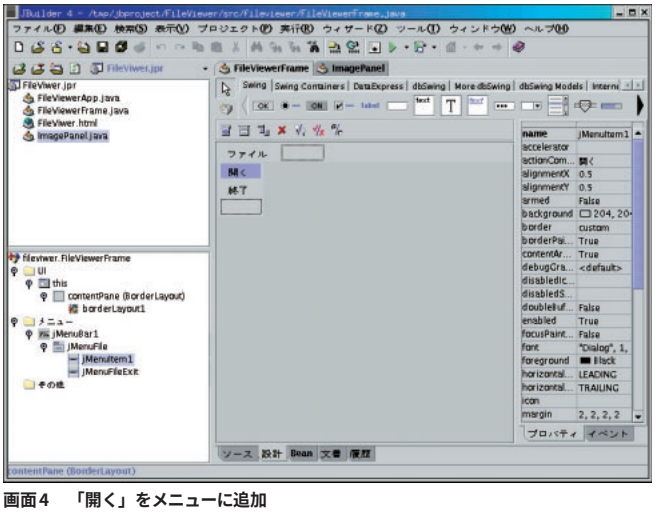
選択したファイルがGIFファイルなのか、あるいはJPEGファイルなのかによって、少し処理が変わります。GIFファイルの場合は、単純にイメージオブジェクトに格納するだけとなります。



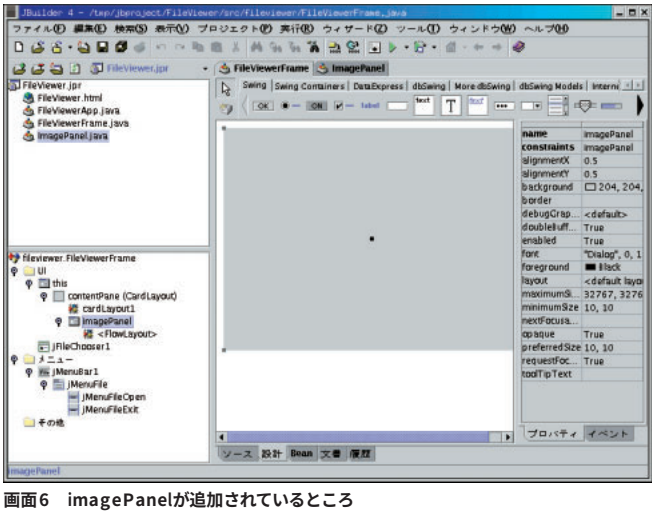
画面2　ビジュアルデザイナ画面



画面3　メニューデザイナ画面



画面4　「開く」をメニューに追加



画面6　imagePanelが追加されているところ

JPEGファイルの場合は、解凍処理が必要となりますので、いったんファイルストリームに格納し、解凍処理（デコード処理）を行い、イメージオブジェクトに格納します。格納したイメージオブジェクトをimagePanelオブジェクトのsetImageメソッドを利用して受け渡すことで、画像を表示することができます。

イメージファイルがフレームのサイズよりも大きい場合も考慮して、開いたイメージファイルの大きさを取得して、フレームサイズを変えるようにしましょう。この方がいかにもビューアらしく動きます。

jMenuFileOpenメニューアイテムのactionPerformedイベントに記述するコードは**リスト12**のようになります。

これで、ファイルビューアとしての機能はできあがりましたが、今の形では［ファイル］－［開く］メニューで表示されるダイアログボックスでは、JPEGファイル、GIFファイル以外のファイルを選択することが可能です。JPEGファイル、GIFファイルしか表示できないように少し細かい部分まで気を遣ってみましょう。

ファイルのフィルタリング

ファイルを開くダイアログボックスは、［Swing Containers］ページにある、FileChooserコンポーネントを使用すると簡単に作成することができます。しかし、拡張子でフィルタをかけるにはちょっとしたテクニックが必要になります。

ファイルのフィルタリングはFileFilterクラスを使用します。JDKのデモの中に、FileChooserクラスを使ったものがあります。これは、$home/JBuilder4/JDK1.3/DEMO/JFC/FileC

▼リスト10　imagePanelオブジェクトの定義

```
public class FileViewerFrame extends JFrame {
  JPanel contentPane;

    ……中略……

  CardLayout cardLayout1 = new CardLayout();
  ImagePanel imagePanel = new ImagePanel();
```

▼リスト11　imagePanelをFileViewerFrameに張り付ける

```
  /**コンポーネントの初期化*/
  private void jbInit() throws Exception  {

    ……中略……

    this.setJMenuBar(jMenuBar1);
    //imagePanelオブジェクトをパネルに追加
    contentPane.add(imagePanel, "imagePanel");
  }
```

▼リスト12　jMenuFileOpen_actionPerformedイベント

```
  /*[ファイル]-[開く]を選択したときの処理 */
  void jMenuFileOpen_actionPerformed(ActionEvent e) {
    //ファイルオープンダイアログを開く
    jFileChooser1.showOpenDialog(this);
    //選択したファイルの名前を取得
    String  filename=jFileChooser1.getSelectedFile().getPath();
    Image image;
    try {
        if (filename.toLowerCase().indexOf(".gif",0)>0) {
          //GIFファイルの場合
          image =
            Toolkit.getDefaultToolkit().createImage(filename);
        } else {
          //JPEGファイルの場合いったんファイルストリームに格納
          FileInputStream fileStream = new
            FileInputStream(filename);
          //デコード処理
          JPEGImageDecoder decoder =
            JPEGCodec.createJPEGDecoder(fileStream);
          //デコードしてイメージオブジェクトに格納
          image = decoder.decodeAsBufferedImage();
        }
        imagePanel.setImage(image);
        int width=image.getWidth(null);
        int hight=image.getHeight(null);
        int MenuHight=this.getHeight()-imagePanel.getHeight();
        int MenuWidth=this.getWidth()-imagePanel.getWidth();
        this.setSize(width+MenuWidth,hight+MenuHight);
        this.paintAll(this.getGraphics());
     } catch (Exception ex) {
        ex.printStackTrace();
     }
  }
```

hooserDemo/SRCに格納されています。このサンプルファイルでは、今回作成しようとしている機能と同じように、GIFファイル、JPEGファイルのフィルタを実装していますので、参考にして記述していきます。

新しいクラスを作成しますので、ImagePanelクラスの場合と同じように、［ファイル］－［新規］で表示されるオブジェクトギャラリでクラスを選択し、クラスウィザードを実行し、次のように設定します。

パッケージ名：

`FileViewer`

クラス名：

`ImageFilter`

親クラス：

`javax.swing.filechooser.FileFilter`

FileFilterクラスは、ディレクトリにある拡張子をFileChooserクラスに受け渡すことで、拡張子によるフィルタを行えるクラスです。今回の場合、JPEGファイル、GIFファイルの2種類なので、ディレクトリ内のファイルの拡張子を取得して、if文を使用して判別していってもいいのですが、汎用性を高めた形にします。Java言語には、ハッシュテーブルという機能があります。これは、いわば一次元の配列です。ハッシュテーブルにフィルタをかけたい拡張子を格納しておき、選択したディレクトリ内にあるファイルの拡張子がハッシュテーブルにある拡張子に一致していれば表示するようにしたいと思います。

ハッシュテーブルの作成は**リスト13**のように行います。ただし、HashTableクラスは、java.util.HashTableパッケージにありますので、import文に追加することを忘れないようにしてください。HashTableへの追加はadd()メソッドで行います。

リスト14は、完成したImageFilterクラスのソースです。完成したソースを見ながら、もう少し詳しく説明していきましょう。

先ほど説明したように、FileFilterクラスを継承したImageFilterクラスでは、選択されたディレクトリ内のファイルの拡張子を判別していきます。この処理はabstractメソッドで行います。選択されたディレクトリのファイル名は必ずこのabstractメソッドを通り、真ならファイルダイアログボックスに表示され、偽なら非表示となります。ファイル名から拡張子を取得する処理は自分で実装する必要があります。ここでは、getExtensionメソッドを作成してファイル名の文字列に「.」が含まれているかどうかを確認し、含まれているときは「.」から右側の文字列を拡張子として取得しています。この取得した拡張子がハッシュテーブルに格納されているフィルタをかける拡張子と一致する場合は、真を返すようにしています。

また、ファイルダイアログボックスにセットされる見出しを管理できるように、getDescriptionメソッド、setDescriptionメソッドを作成しています。

続いて作成したImageFilterクラスを使用できるように、FileViewerFrameクラスで呼び出します。まず、オブジェクトを生成します。**リスト15**の

▼リスト13　ハッシュテーブルの作成

```
HashTable newHashTable = new HashTable();
```

▼リスト15　ImageFilterの追加

```
public class FileViewerFrame extends JFrame {
  JPanel contentPane;

    ……中略……

  ImagePanel imagePanel = new ImagePanel();
  ImageFilter imageFilter = new ImageFilter();
```

▼リスト16　CODECの追加

```
import com.sun.image.codec.jpeg.*;
import java.io.*;
```

▼リスト17　jbInit()メソッドにフィルタ機能を記述

```
  private void jbInit() throws Exception  {

    ……中略……

     contentPane.add(imagePanel, "imagePanel");

    //ファイルを開くときにJPEG,GIFファイルのみにフィルタ
    imageFilter.addExtension("gif");
    imageFilter.addExtension("jpg");
    imageFilter.setDescription("画像ファイル");
    //フィルタをセット
    jFileChooser1.setFileFilter(imageFilter);
  }
```

ようにImageFilterクラスを継承するimageFilterオブジェクトを生成します。

さらに、JPEGファイルを解凍するためのCODECを利用する宣言をFileViewerFrameクラスの先頭に追加します（**リスト16**）。

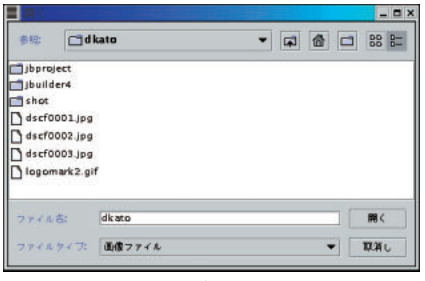
続いて、jbInit()メソッド内で、作成したフィルタオブジェクトに拡張子を追加し、jFileChooser1オブジェクトにフィルタをセットします（**リスト17**）。

これですべてのコーディングが終了しました。［プロジェクト］－［プロジェクトのメイク］を選択してメイクを行い、実行してみましょう。**画面7**のようにファイルダイアログボックスで、GIFファイル、JPEGファイルのフィルタが働き、**画面8**のようにイメージファイルがパネルに表示されます。

付録CD-ROMには、プロジェクトファイルを含めた全ソースを収録したtarファイルが収録されているので、興味がある方はソースファイルを確認しながら機能を拡張してみてください。

●

次回はデータベースとの連携について説明し、Java言語の持つデータベースアプリケーションの機能について解説します。



画面7　ファイルのフィルタ

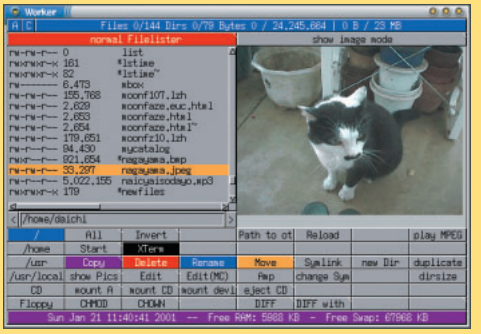
画面8　イメージファイルを表示したところ

▼リスト14　ImageFilter.java

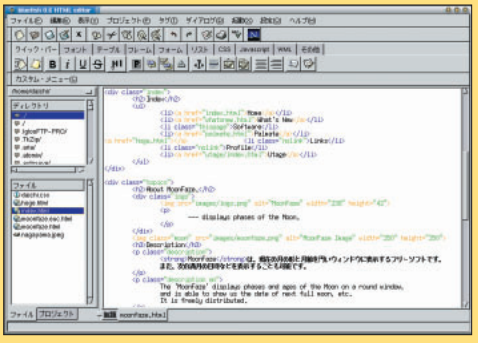
```
package fileviewer;

import javax.swing.filechooser.*;
import java.util.Hashtable;
import java.io.*;
import javax.swing.filechooser.*;

public class ImageFilter extends java.swing.filechooser.FileFilter {

  private String description;       //見出しを管理する変数
  //フィルタしたファイル名を格納するハッシュテーブル
  private Hashtable filters = null;

  //コンストラクタ
  public ImageFilter() {
        this.filters = new Hashtable();        //ハッシュテーブルを作成
  };

  //表示されるファイル名かどうかを判断するメソッド
  public boolean accept(File f) {
    /**@todo: この javax.swing.filechooser.FileFilter
                                        abstract メソッドを実装*/
    if(f != null) {
          //ディレクトリなら表示
      if(f.isDirectory()) {
        return true;
      }
          //拡張子を取得
      String extension = getExtension(f);
          //該当する拡張子なら表示
      if(extension != null &&
         filters.get(getExtension(f)) != null) {
        return true;
      };
    }
    return false;
  }

  //見出しを返すメソッド
  public String getDescription() {
    /**@todo: この javax.swing.filechooser.FileFilter
                                        abstract メソッドを実装*/
      return this.description;
  }

  //見出しをセットするメソッド
  public void setDescription(String description) {
    this.description = description;
  }

  //拡張子を取得するメソッド
  public String getExtension(File f) {
    if(f != null) {
      String filename = f.getName();
      int i = filename.lastIndexOf('.');
      if(i>0 && i<filename.length()-1) {
        return filename.substring(i+1).toLowerCase();
      };
    }
    return null;
  }

  //該当する拡張子をハッシュテーブルに格納するメソッド
  public void addExtension(String extension) {
    if(filters == null) {
      filters = new Hashtable(5);
    }
    filters.put(extension.toLowerCase(), this);
  }
}
```

Free Application Showcase

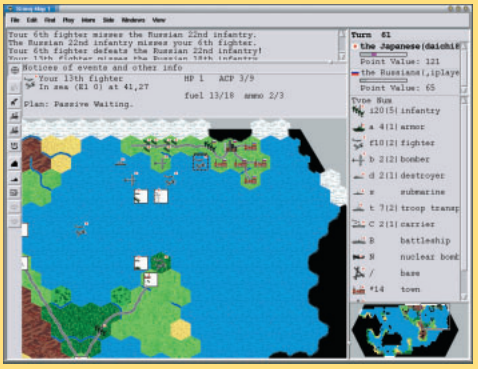
文：出井 一
Text : Hajime Dei



Worker P.110



Bluefish P.114



Xconq P.116

※紹介したソフトは、すべて付録CD-ROMに収録されています。

気軽に使えるパーソナルデータベース

Gaby

バージョン：1.9.98　　ライセンス：GPL

http://gaby.sourceforge.net/

Gabyは、住所録やCD目録などの小規模データを気軽に管理できるパーソナルデータベースソフトだ。国際化について十分に考慮されており、日本語のデータを問題なく入力・編集できる。付属のデータベースを利用するほか、データベースビルダを使って、対話的な操作で独自のデータベースを作成したり、項目の表示順の変更や新たな項目の追加なども可能だ。動作にはGTK＋が必要となる。

●ビルドとインストール

Gabyは、tarボールとRPMバイナリパッケージで配布されている。tarボールを利用する場合は、「./configure」→「make」→「make install」という一般的な手順でビルドとインスールを行える。

一方、バイナリパッケージはXimian（前HelixCode）のGNOMEを必要とするため、国産のディストリビューションでは動作しない可能性が高い。RPMを利用する場合は、tarボールに含まれるSPECファイルを利用して、自分でパッケージを作成しよう。

具体的には、スーパーユーザー（root）になった状態で「rpm -ta gaby-1.9.98.tar.gz」とすればいい。/usr/src/redhat/RPMS以下にRPMのソース・バイナリパッケージが作成されるので、通常の手順でバイナリパッケージをインストールしよう。

●Gabyの起動と画面表示

Gabyを起動するには、コマンドラインから「gaby&」とするか、GNOMEメニューの[プログラム]－[アプリケーション]－[Gaby]を選択する。最初に起動したときには、Gabyの特徴を説明するウィザード（**画面1**）、2回目以降は使い方のヒントなどを教えるTipsウィンドウが表示される。

Gabyではデータの入力や閲覧を各種の「ビュー」で行う。基本的なビューは、レコード単位でデータの入力・編集を行う「フォーム」と、多数のレコードを同時に閲覧する「リスト」の2つ。このほかにも、「拡張リスト」や「フィルタ」などのバリエーションがある。

初期設定では、住所録のフォーム（メインウィンドウ）と、電話帳の拡張リスト（サブウィンドウ）が起動時に開くようになっている。

●住所録フォームにデータを入力

住所録のフォームにデータを入力してみよう（**画面2**）。日本語も問題なく入力できる。1人分のデータを入力したら、[新規]ボタンで新しいレコードの入力に移る。[前][次]ボタンで前後のレコードに切り替えて、設定内容を修正することも可能だ。

なお、日本人の名前がFirst Name（名）→Last Name（姓）という順番で表示されるのは不自然だが、これについては後で説明する「データベースビルダ」で変更できる。また、「First Name」などのフィールド名を日本語化することも可能だ。

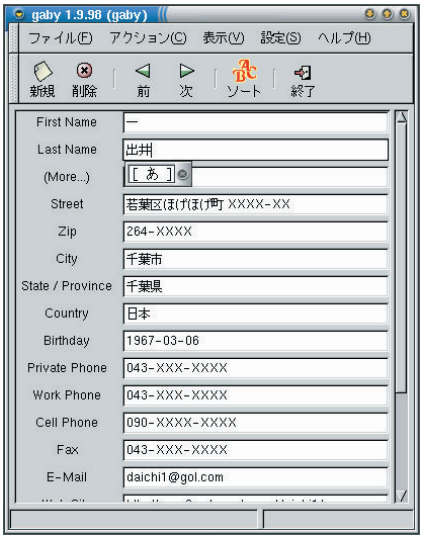
適当なところで[ファイル]－[保存]を選択してデータを保存しよう。こうして作成したデータは、ホームディレクトリの.gaby/table.AddressBookにテキスト形式で保存される。

●拡張リストでレコードを検索

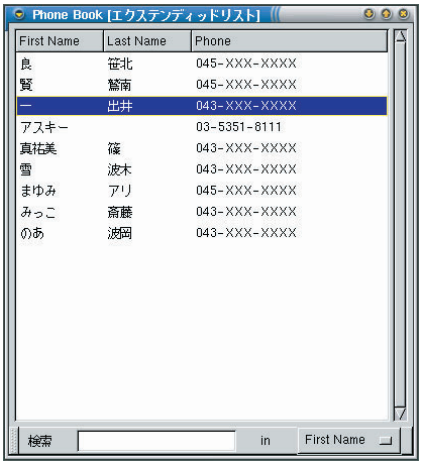
起動時に開くもうひとつのウィンドウは電話帳の拡張リストで、入力したレコードの内容のうち、姓名と（プライベートの）電話番号だけが一覧表示される（**画面3**）。

「拡張」たる由縁は、レコードの検索機能にある。ウィンドウ下の[検索]ボックスに文字列を入力すると、その右のコンボボックスのフィールドの内容が、指定した文字列で始まるレコードだけに絞り込まれるのだ。Emacsなどでおなじみのインクリメンタルサーチ（1文字入力するたびに自動検索する手法）を採用しているので、探しているレコードが見つかった時点で入力

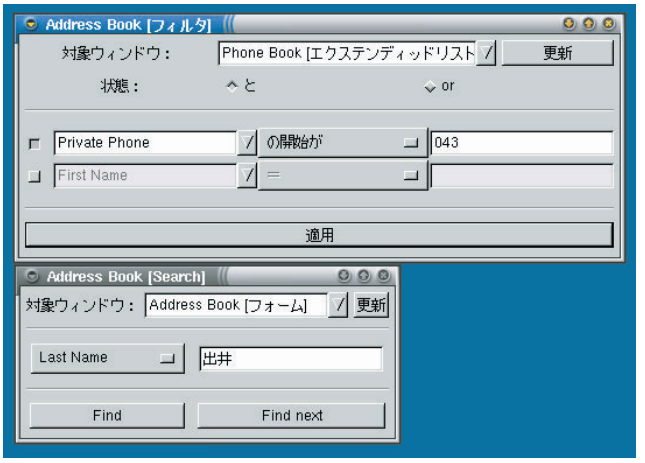
画面1
初めて起動した場合は、機能を説明するウィザードが開く。



画面2
メインウィンドウには住所録の入力用フォームが表示される。



画面3
電話帳の拡張リスト。インクリメンタルサーチで絞り込み可能。



画面4
プラグインで実現されるフィルタやSearchも用意されている。

をやめればいい。

なお、この電話帳の拡張リストは、住所録のフォームと結合（バインド）しているので、フォームで新たなレコードを登録すると自動的に拡張リストの表示も更新されるし、拡張リストのレコードをクリックすると、その内容がフォームに表示される。

●その他のビューやアクション

Gabyの[表示]メニューには、電話帳のみのフォームや住所録のリストなどに加えて、検索を行う「Search」、リストの表示を絞り込む「フィルタ」なども用意されている（**画面4**）。これらはプラグインで実現されており、機能の拡張も容易だ。

一方、[アクション]メニューには、レコード内の電話番号やメールアドレスを利用して、電話をかけたりメールを出したりする項目が用意されている。これらの機能はどれも外部コマンドを呼び出すことで実現されているので、対応するコマンドのインストールや設定が必要だ。

外部コマンドの設定は、[設定]－[設定]で開くダイアログの[Net]ページで行う（**画面5**）。このほか、起動時のTipsウィンドウの有無など、さまざまな設定を変更できる。

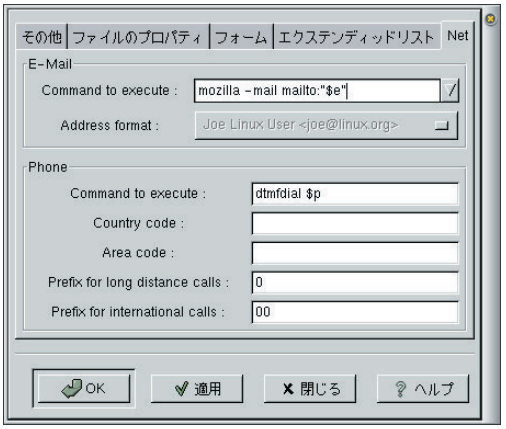
●データベースの作成・修正も可能

Gabyには、住所録のほかにも、CDや書籍の目録などのデータベースが定義済みだ。これらは、[ファイル]－[Databases]以下のメニューから選択するか、「gaby -a gcd」のように起動時の-aオプションでデータベース名（タイトルバーに表示される）を指定することで利用できる（**画面6**）。

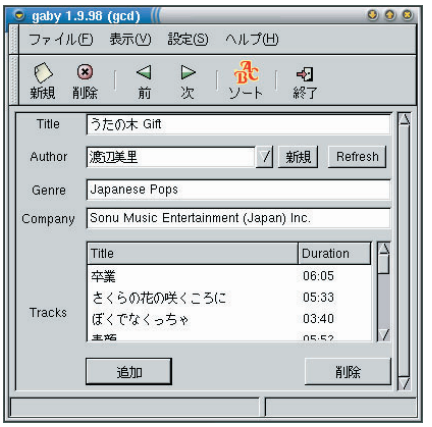
データベースの構造や使用するフォームの構成、各ロケールに対応したフィールド名などは、/usr/etc/gabyに置かれた記述ファイル（desc.＊）に書かれている。住所録の記述ファイルはdesc.gaby、そのほかは「desc.」の後にデータベース名を付けたものだ（CD目録の場合は「desc.gcd」）。

さらに、Gabyに付属するデータベースビルダを使うと、GUI環境で新しいデータベースを作成したり、既存のデータベースの構成を変更したり、各ロケールに対応したフィールド名を追加することが可能だ。

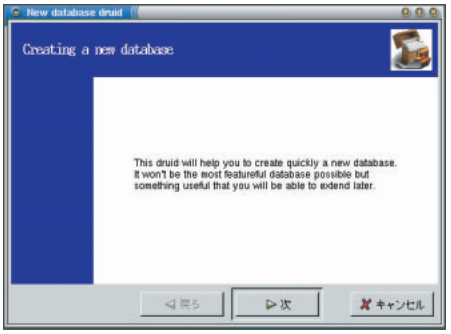
以下では、データベースビルダを使って住所録のデータベースを修正し、フォームやリストのフィールドが



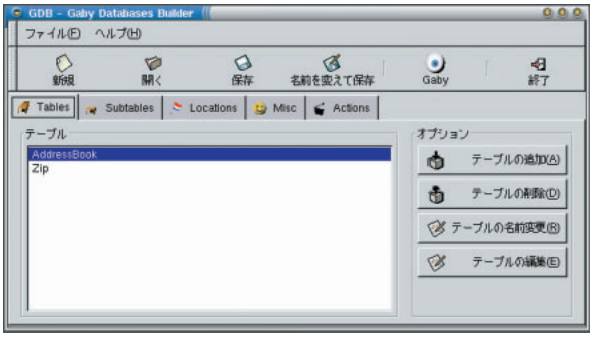
画面5
[アクション]メニューの各項目で使用される外部コマンドを設定。



画面6
CD目録などのデータベースも最初から用意されている。



画面7
対話的な操作で新たなデータベースを作成できる。



画面8
データベースビルダで住所録のデータベースを修正する。

「First Name」→「Last Name」という順番で表示されるようにする。あわせて、「First Name」などのフィールド名の日本語化も行う。

●住所録データベースを修正する

コマンドラインで「gabybuilder&」とするか、Gabyで[ファイル]－[Databases]－[Create a new one]を選択すると、データベース新規作成用のウィザード（**画面7**）が開く。既存のデータベースを修正する場合は、[キャンセル]ボタンを押してデータベースビルダのウィンドウを開こう。

[開く]ボタンを押して、住所録の記述ファイル（/etc/gaby/desc.gaby）を読み込むと、データベースを構成するテーブルや表示用のサブテーブル、各ビューの設定などがタブ付きページに表示される（**画面8**）。

このデータベースビルダでは、データベースのテーブルに新しいフィールドを追加したり、既存のフィールドを削除したりすることも可能だ。ただし、このようにテーブル自体の構成を変更すると、すでに作成したデータとの整合性が失われてしまう。

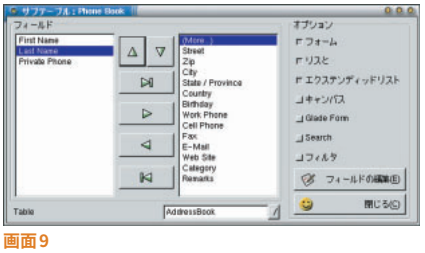
そこで、今回は表示に使われるサブテーブルの構成だけを変更する。[Subtables]ページに切り替え、リスト中の「Phone Book」をダブルクリックしよう。サブテーブルの内容が別ウィンドウに表示されたら（**画面9**）、フィールドのリストから「Last Name」（姓）を選択し、[△]ボタンで「First Name」（名）の前に来るようにすればいい。サブテーブルの「Address Book」についても、同様にフィールドの順番を変更しておこう。

このほか、フィールド名をダブルクリックすることで、ロケールごとの表示名を設定できる。たとえば、「Last Name」の場合は、ダブルクリック後に[追加]ボタンを押し、日本語でのフィールド名「姓」とロケールコード「ja」を設定すればいい（**画面10**）。「First Name」や「Private Phone」についても同様に日本語のフィールド名を設定しよう。

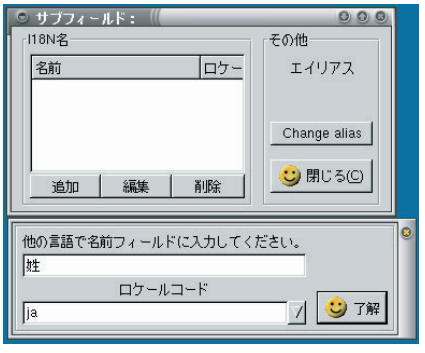
●データベースの保存と確認

一般ユーザーの場合は、修正した記述ファイルを元のファイル（/etc/gaby/desc.gaby）には上書きできないので、[名前を変えて保存]ボタンを押し、ホームディレクトリの.gabyディレクトリに保存するとよい。このディレクトリは、Gaby起動時に/etc/gabyより先に検索されるので、変更した記述ファイルが読み込まれる。

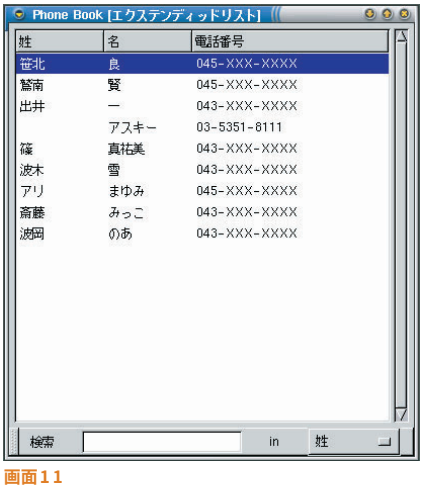
上記のように名前を変えて保存した後、データベースビルダーのツールバーの[Gaby]ボタンを押すと、現在の記述ファイルを読み込んでGabyが起動する。電話帳の拡張リストの名前が日本語風の順番になり、フィールド名が日本語になっていることを確認しよう（**画面11**）。



画面9
表示用のサブテーブルで姓・名のフィールドの順番を変更。



画面10
各フィールドに対応する日本語名とロケールコードを設定する。



画面11
日本語風の表示順に変わった電話帳の拡張リスト。

2ペイン構成のファイルマネージャ

Worker

バージョン：2.0.1　　ライセンス：GPL

http://www.boomerangsworld.de/worker/
http://web.access.net.au/argyll/xli.html（xli）

Worker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。一方のペインにプレビュー画像やファイル情報を表示することもできる。また、ボタンの内容やキーバインド、ファイルタイプ別のコマンドなどは、すべてダイアログで設定できる。拡張子ではなく、ファイルの実際の内容に応じてファイルタイプが判断されるのが特長だ。

●ビルドと初期設定

Workerは、ソース一式をtar＋gzipしたtarボールのみ配布されている。なお、tarボールを展開後、src/wdefines.hの50行目の「xterm」を「kterm」に修正しておこう。

ビルドとインストールは、「./configure」→「make」→「make install」という一般的なもの。このとき、初期設定用スクリプト（worker.inst）もインストールされる。

最初にWorkerを実行するユーザーは、一度だけこのスクリプトを実行して、サンプルの設定ファイルをコピーする必要がある。ktermなどのコマンドラインで「worker.inst」とすると表示用言語の一覧が表示されるので、「1.english」を選択しよう。なお、各ユーザーの設定ファイルは、ホームディレクトリの.workerディレクトリに作成される。

このほか、後述のイメージ表示モードを利用するために、画像ビューアxliが必要だ。こちらは、tarボールを展開後、「xmkmf」→「make」→「make install」という手順でビルドとインストールを行う。

●左右にファイルリストを表示

ktermなどのコマンドラインで「worker&」として起動すると、左右2つのペインとボタンが並んだウィンドウが開く（**画面1**）。

左右のペインには、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る。初期設定の状態では、左ペインには起動時のカレントディレクトリが表示され、右ペインには何も表示されていないはずだ。ディレクトリの切り替えは、ボタン（[/home]など）のクリックや、サブディレクトリのダブルクリック、ディレクトリ表示部へのキー入力で行える。

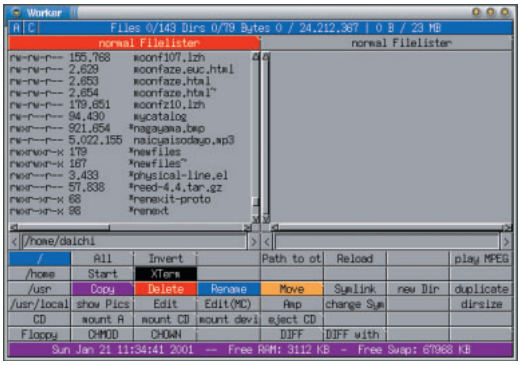
ファイルリストには、ファイル名・属性・サイズ・ファイルタイプ・所有者名などが表示される。情報の一部が隠れている場合は、マウスの右ボタンドラッグで上下左右にスクロールする。このほか、ファイルのソート順を変えたり、ドットファイルを隠したり、表示ファイルを絞り込むフィルタを設定することも可能だ。

なお、ペイン上部を右クリックすると開くダイアログで、[show image mode]のボックスをチェックすると、他方のペインのカーソル位置の画像が表示される「イメージ表示モード」になる（**画面2**）。つまり、Workerをプレビュー機能を持った画像ビューアとして利用できるわけだ。同様に、[information mode]のボックスをチェックすると、他方のペインのカーソル位置のファイル情報を表示する「情報表示モード」になる。

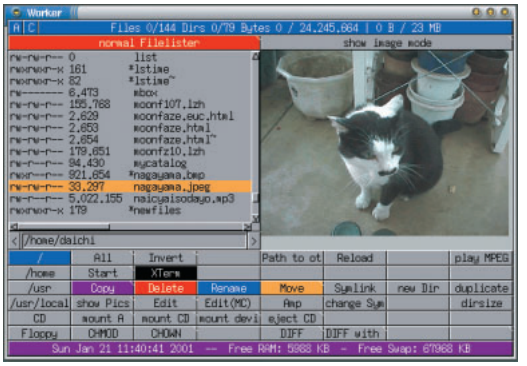
●ボタンを利用した操作を行う

ファイルのコピーや移動、削除といった操作は、ペイン上部が赤く表示されているアクティブなファイルリストの選択ファイルに対して行われる。ただし、ファイルがひとつも選択されていない場合は、カーソル位置のファイルが対象となる。

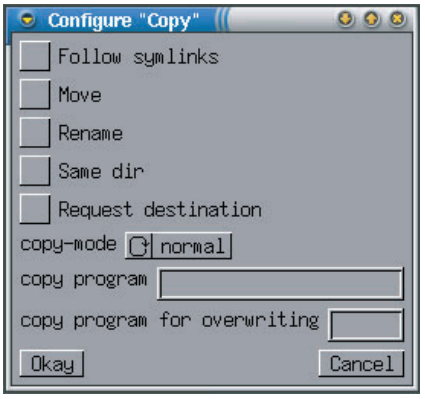
ファイルを選択するには、単にマウスでクリックすればいい。再度クリックすると選択解除となる。また、ドラッグにより複数のファイルを一度に選択することも可能だ。



画面1
Workerでは、2つのディレクトリの内容を同時に表示できる。

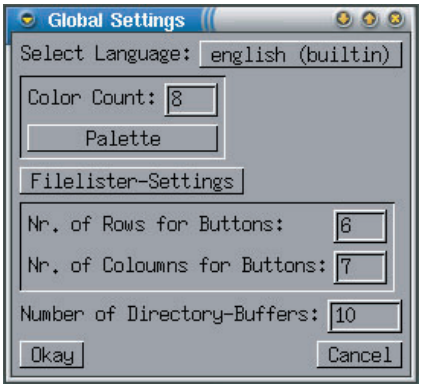


画面2
カーソル位置の画像を別ペインに表示するイメージ表示モード。



画面3
[Copy]ボタンの右クリックで複雑な設定でのコピーが可能だ。

画面4
ジャンル別のボタンが並んだ設定ダイアログ。



画面5
[Globa Setting]では表示用の言語やボタン数などを変更できる。

選択後、ウィンドウ下部に並んだボタンをクリックすると、それぞれの処理が行われる。たとえば、[Copy]や[Move]ボタンでは、アクティブでないペインのディレクトリに選択ファイルがコピー・移動されるし、[Delete]ボタンでは選択ファイルがユーザーの確認後に削除される。

ボタンの右上の印は、右クリックに別の機能が割り当てられていることを示す。たとえば、[Copy]ボタンを右クリックすると、同じディレクトリ内でのリネームコピーなどの設定を行うダイアログが開く（**画面3**）。Workerの終了は、右下のボタンの右クリックに割り当てられている。

なお、こうしたボタンの内容は全部で3バンク用意されており、ウィンドウ最下行のカラーバーの右クリックで切り替え可能だ。

●ファイル内容に基づく処理を実行

ファイルリストのファイルをダブルクリックすると、それぞれのファイルタイプに応じたコマンドが実行される。たとえば、MP3ファイルならxmmsで再生、PSファイルはgvでプレビューといった具合だ。

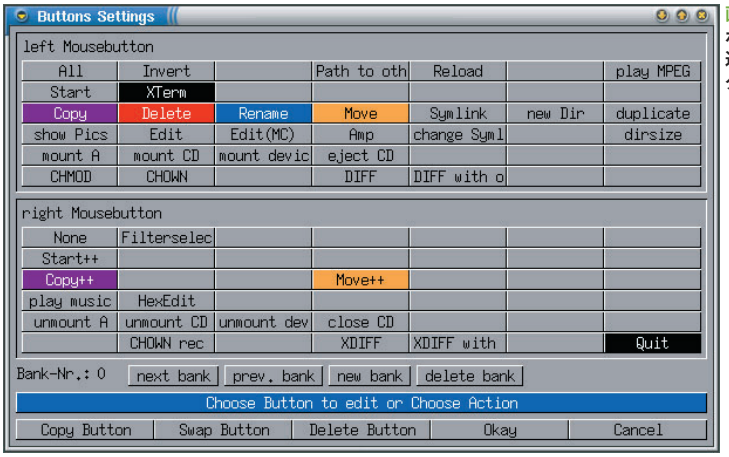
こうした機能は他のファイルマネージャにもあるが、ほとんどの場合はファイル名のパターン（拡張子など）だけから判断している。一方、Workerでは、ファイルの一部を実際に読み込んでファイルタイプを判断する。

たとえば、JPEG画像の拡張子（「.jpeg」など）を削除したり、別の拡張子に変更したりしても、ちゃんとJPEG画像と判断され、ダブルクリックで画像が表示される。逆に、テキストファイルの拡張子を「.jpeg」に変更しても、JPEG画像とは見なされない。

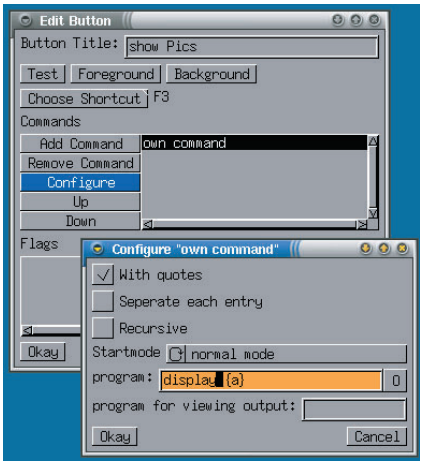
ただし、同じ内容のテキストファイルでも、拡張子が「.c」ならCのソースファイル、「.txt」なら一般的なテキストと見なすなど、ファイルの内容では判別できないケースもある。そこで、Workerではファイル名のパターンによってファイルタイプを判断することも可能になっている。

●ジャンル別に分かれた設定

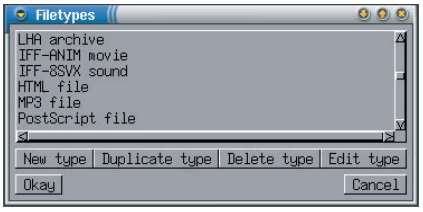
Workerの設定ダイアログは、ウィンドウ左上の[C]ボタンで開く（**画面4**）。設定内容が多岐にわたるため、ジ



画面6
ボタンの設定を追加・修正するダイアログ。



画面7
[show Pics]ボタンで実行するコマンドをdisplayに変更する。



画面8
画像やMP3ファイルなどのファイルタイプが最初から設定済みだ。

ャンル別に用意されたボタンを押すと、それぞれの設定を行うダイアログが開く構成になっている。

[Global Settings]ボタンで開くダイアログでは、表示用の言語（英語・ドイツ語・フランス語）や、縦横のボタン数、表示に使用する色（パレット）、左右のペインの表示形式などを設定できる（**画面5**）。

たとえば、背景色の初期設定（暗い灰色）が暗すぎて文字が見にくいなら、パレットの灰色をGTK+などで使われる明るい灰色（RGBの設定は「192,192,192」）に変更すればいい。

このほか、各部の文字・背景色、起動時のディレクトリ、コマンド用やディレクトリ変更用のボタン、ファイルタイプ、キー割り当て、フォントなどの設定が可能だ。設定後に[Save]ボタンを押すと、設定内容が各ユーザーの設定ファイルに保存される。

●ボタンの設定を変更しよう

ユーザーが新たなボタンを追加したり、ボタンの設定内容を変更することも可能だ。例として、画像を表示する[show Pics]ボタン用のコマンド（初期設定はxv）を、ImageMagickの画像ビューア（display）に変更する。

設定ダイアログの[Buttons Settings]ボタンを押すと、左右クリックに対応したボタンが上下に並んだダイアログが開く（**画面6**）。上の「left Mouse button」に含まれる[show Pics]ボタンを押して、ボタンの内容を設定するダイアログを開こう。

このダイアログでは、[Commands]のリスト中の「own command」を選択して[configure]ボタンを押す。コマンド設定用のダイアログが開いたら、[program:]の設定内容中の「xv」を「display」に変更すればいい（**画面7**）。「{a}」は、実行時に選択ファイル名のリストに置換される。

以後、画像を選択して[show Pics]ボタンを押すと、displayで画像が表示されるようになる。複数の画像を選択した場合は、スペースキーで次の画像に切り替えられる。

このほか、テキストを編集する[Edit]ボタン用のコマンド（初期設定はnedit）も、好みのエディタ（XEmacsなど）に変更するとよいだろう。

●ファイルタイプの設定を修正する

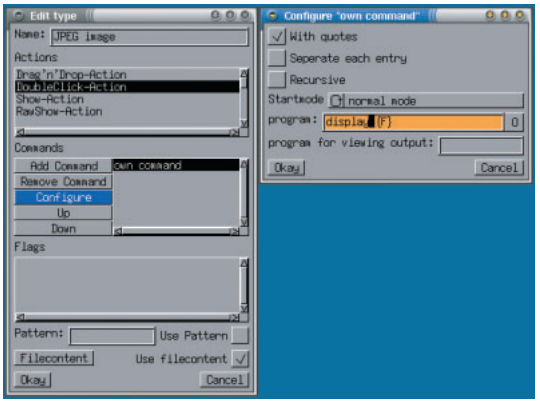
今度は、ファイルタイプの設定を修正してみよう。JPEGなどの各種画像のファイルタイプで使われる表示コマンドを、[show Pics]ボタンと同様にxvからdisplayに変更する。

設定ダイアログで[Filetypes]ボタンを押すと、ファイルタイプの一覧ダイアログが開く（**画面8**）。画像に関するファイルタイプ（「～ image」と表示される）のいずれかを選択し、[Edit type]ボタンでファイルタイプ編集ダイアログを開こう。

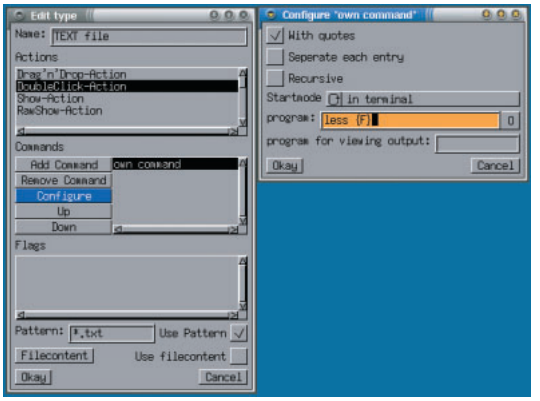
このダイアログでは、まず[Actions]のリストから「DoubleClick-Action」を選択し、続いて[Commands]のリストの「own command」を選択して[Configure]ボタンを押す。コマンド設定ダイアログが開いたら、[program:]の内容に含まれる「xv」を「display」に置き換えればいい（**画面9**）。

xmmsやImageMagickのように、独立したウィンドウを持つコマンドだけでなく、ktermなどの端末内で実行されるコマンドも設定可能だ。たとえば、「*.txt」というファイル名パターンに対して、ダブルクリックでlessを起動する設定を示す（**画面10**）。

なお、最初に説明したソースの修正を行わなかった場合は、端末ソフトとしてxtermが起動される。xtermは、日本語の表示ができないため、日本語テキストを表示させると文字化けしてしまうので注意されたい。



画面9
JPEG画像をダブルクリックするとdisplayが起動されるように修正。



画面10
「*.txt」に対してlessを端末内で実行するファイルタイプ設定。

国産のダウンロード支援ソフト

Aria

バージョン：0.6.0　　ライセンス：GPL

http://www.geocities.co.jp/SiliconValley-Bay/3167/linux/linux_e.html

Ariaは、HTTP／FTPプロトコルを利用してファイルのダウンロードを行うソフトだ。クリップボードやファイルからURLのリストを受け取り、複数のファイルを効率よくダウンロードできる。巨大なファイルの各部を同時にダウンロードする分割ダウンロードも可能だ。国産ソフトだけあって、付属の日本語カタログによる日本語表示で使える点もうれしい。実行にはGTK＋が必要となる。

●ビルドと初期設定スクリプト

Ariaは、tarボールとDEBパッケージで配布されている。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。

Ariaを利用するユーザーは、初めに一度だけ初期設定用のスクリプトを実行する必要がある。ktermなどのコマンドラインで、ソースを展開したディレクトリに移動し、「./mksetup」とすればいい。ホームディレクトリの.ariaディレクトリに設定ファイルなどがコピーされる。

●アイテムの登録とダウンロード

「aria&」として起動すると、上下2ペイン構成のウィンドウが開く。上のペインにはダウンロードするファイル（「アイテム」と呼ぶ）のリストが表示され、下のペインにはさまざまな情報が表示される（**画面1**）。

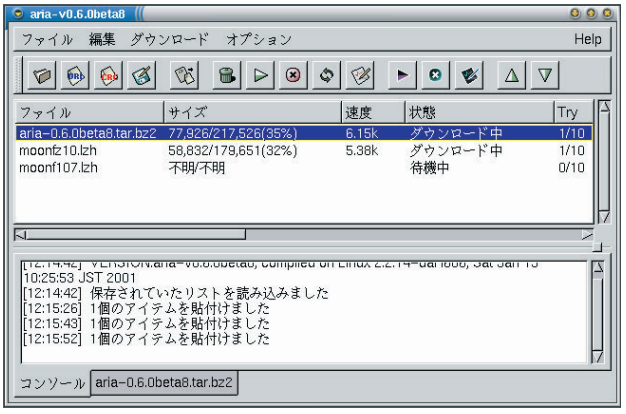
まずは、[デフォルトのアイテム設定]ボタンを押して、全アイテム共通の設定を行おう（**画面2**）。ダウンロード先ディレクトリや、ファイルの分割数、プロキシ関係などの設定が可能だ。なお、こうした設定は後でアイテムごとに変更できる。

ファイルのURLは、他のアプリからクリップボード経由で取得する。Netscapeなら、リンクを右クリックして[リンクの場所をコピー]を選択し、Ariaの[URLリストを貼り付け]ボタンを押せばいい。URLを並べたテキストファイルからの読み込みも可能だ。

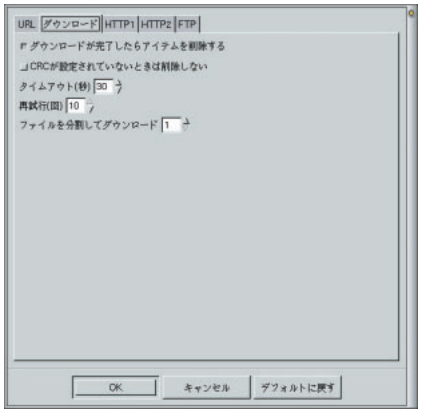
いずれにせよ、URLのリストが表示されるので（**画面3**）、アイテムとして登録したいものを選択し、[貼り付け]ボタンを押そう。登録したアイテムは上部ペインに表示され、設定の変更などが可能だ。

[選択したアイテムのダウンロードを開始]ボタンか、[すべてのアイテムのダウンロードを開始]ボタンを押すとダウンロードが始まる。なお、ダウンロード終了後は、アイテムは自動的に削除される（変更可能）。

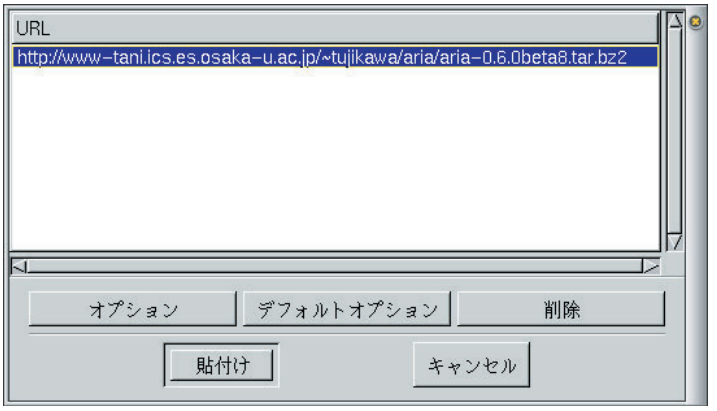
一方、下部ペインのコンソールはタブ付きページになっており、全体的な情報と選択したアイテムの情報を切り替えて表示できる。



画面1
上部にアイテムのリスト、下部にさまざまな情報が表示される。



画面2
全アイテム共通の設定を行う。個別に設定を行うことも可能。



画面3
Netscapeなどからクリップボード経由でURLを貼り付けられる。

日本語に対応したカラー構文表示のHTMLエディタ

Bluefish

バージョン：0.6　　ライセンス：GPL

http://bluefish.openoffice.nl/

Bluefishは、カラー構文表示に対応したタグ挿入型のHTMLエディタだ。ツールバーのボタンを使ってタグを効率よく入力できる。日本語カタログによるメニューの日本語化や、日本語の入力・編集にも対応している（日本語EUCのみ）。また、作成したWebページをWeblintでチェックしたり、実際の表示を外部のブラウザで確認できるなど、外部ソフトとの連携も考えられている。動作にはGTK＋が必要だ。

●ビルドとインストール

Bluefishは、tarボールとRPMパッケージの両方で配布されている。ただし、Red Hat用のRPMはi686バイナリパッケージしか用意されていないので、通常はtarボールを利用したほうがいいだろう。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。

●Bluefishの画面構成

「bluefish&」として起動すると、スプラッシュ画面が表示された後、ボタンが並んだウィンドウが開く（**画面1**）。日本語環境では、メニューなどが日本語で表示されるはずだ。

ツールバーは2段構成で、上段がファイル操作や編集用、下段がタグ入力用となっている。タグ入力用のツールバーはジャンル別にページ分けされており、各ボタンの機能はチップヘルプの説明で確認できる。

左側のファイルブラウザには、ディレクトリ一覧で選択したディレクトリ内のWeb関連ファイルが一覧表示される。ここに表示されたHTMLやスタイルシートは、ファイル名のダブルクリックや編集領域へのドロップで編集できる。また、画像ファイルの場合は、対応するIMG要素がカーソル位置に挿入される。

右側の編集領域では、HTMLファイルをはじめ、スタイルシートや通常のテキストなどを編集できる。複数ファイルを編集する場合は、ウィンドウ下のタブで切り替え可能だ。

●日本語対応のための設定を行う

インストール直後の設定のままだと、編集領域で日本語が文字化けしてしまうので、以下のように設定を変更する必要がある。ツールバー上段の[詳細設定]ボタンで設定ダイアログを開き、GUIページに切り替えよう（**画面2**）。「フォントセットでフォント指定」をチェックし、[保存して閉じる]ボタンを押せば準備完了だ。

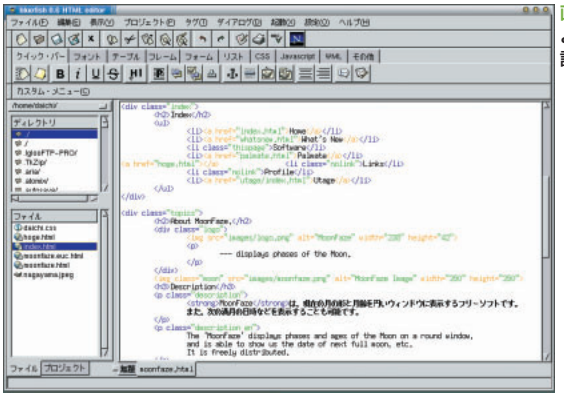
日本語EUCで書かれたHTMLファイルを読み込んで、正常に表示されることを確認しよう。なお、Ctrl－oキーはオープンダイアログに割り当てられているため、日本語入力にはShift－スペースキーを利用する。インライン入力はできないものの、入力や編集は正常に行われる。

●HTMLファイルの入力

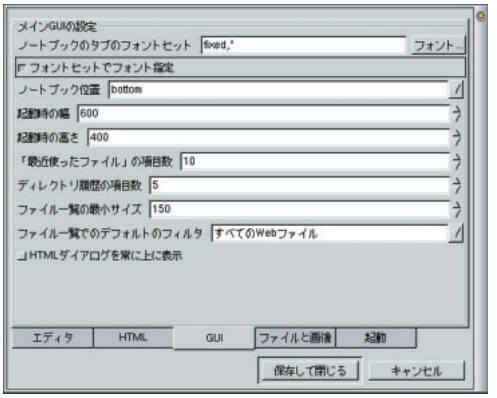
使い方は一般的なタグ挿入型のHTMLエディタと同じで、ツールバーのボタンを押してカーソル位置にタグを挿入し、内容をキーボードから入力する。もちろん、タグ自体をキーボードから入力してもいい。

内容と終了タグを持たない空要素（BRなど）の場合は、ボタンを押すだけでタグが挿入される。一方、「<開始タグ>内容</終了タグ>」という一般的な要素（Pなど）では、ボタンを押してから内容を入力してもいいし、先に入力した内容を選択した状態にしてからボタンを押してもいい。

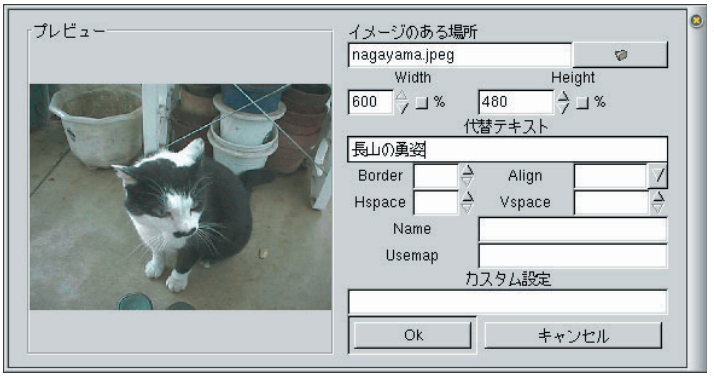
AやIMGのように属性の指定が不可欠の要素では、専用のダイアログが開いて属性値を設定でき（**画面3**）、指定した属性と属性値を持った要素がカー



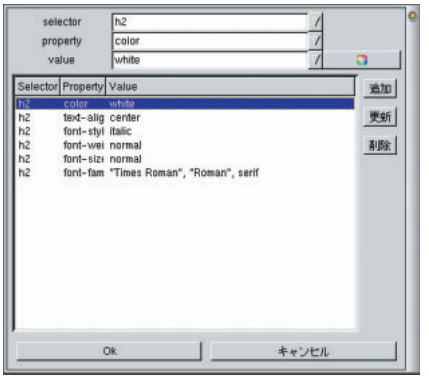
画面1　日本語カタログによってメニューなどが日本語で表示される。



画面2　編集領域で日本語を表示するにはこの設定が必須だ。



画面3　イメージタグの設定ダイアログでは画像をプレビューできる。



画面4　スタイルシートの設定はこのダイアログで行う。

ソル位置に挿入される。その後、右クリックメニューの[タグの編集]により、カーソル位置の要素を再編集することも可能だ。

なお、HTMLファイルを新規作成する場合は、ツールバー下段の[クイックバー]ページ左端の[クイック・スタート]ボタンを使うといい。DTD（文書型定義）やタイトルなどの基本的な設定をダイアログで行い、まとめて要素を挿入できる。

●スタイルの設定も強力に支援

CSSスタイルシートのスタイルは、[CSS]ページ左端の[Create Stylesheet]ボタンで設定する（**画面4**）。セレクタ・プロパティ・値をそれぞれのリストから選択するだけの簡単操作だ。また、スタイルシートで設定済みの各スタイルを範囲選択（反転表示）した状態でこのボタンを押すと、現在の設定が自動的にリスト表示され、設定の変更や追加を行える。

このほか、スタイルの指定にかかわる作業、たとえばHTMLファイルのHEAD要素でのスタイルシートの指定、各要素にクラスを追加といった操作もボタンひとつで実行可能だ。

●日本語の色分けも問題なし

Bluefishの特徴のひとつに、タグや属性、属性値などをカラー表示する色分け（カラー構文表示）機能がある。以前は日本語表示との相性が悪かったが、現在ではこの問題は解消されたので、日本語を含む場合も色分けをオフにする必要はない。

なお、初期設定では編集行のみ自動更新されるので、色分けされていない部分が残ってしまった場合は、F5キーを押して手動で更新しよう。

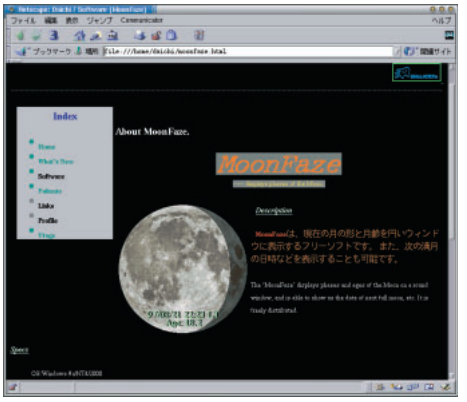
どの部分を何色で表示するかという設定は、[設定]－[リスト]－[タグの色分け]で開くダイアログで行う（**画面5**）。文字色と正規表現パターンを組み合わせて指定するので、柔軟な設定が可能だ。

●ブラウザで実際の表示を確認

Bluefishでは、編集中のHTMLファイルの実際の表示を、任意のブラウザで確認できる（初期設定はNetscape）。Netscapeを利用する場合は、設定ダイアログの[起動]ページで、[ブラウザコマンド]の設定の最後にある「)」を削除しよう（1つ余分についている）。Mozillaやgaleonなど、他のブラウザを使う場合は、コマンド部分も含めて書き換えればいい。

ブラウザの起動は、ツールバー上段の右端の[Netscapeで確認]ボタンで行う。自動的にファイルの保存が行われた後、ブラウザにより編集中のWebページが表示される（**画面6**）。すでにNetscapeが起動していた場合は、そのウィンドウを利用してWebページの表示が行われる。

画面5　タグや属性値などの色分け表示は正規表現パターンで指定する。



画面6　Netscape Navigatorで編集中のページの実際の見栄えを確認。

戦術級シミュレーションゲームシステム

Xconq

バージョン：7.4.1　　ライセンス：GPL

http://sources.redhat.com/xconq/

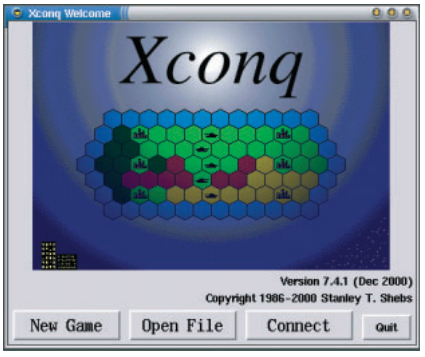
Xconqは、俯瞰マップタイプの戦略・戦術級シミュレーション用ゲームシステムだ。XのほかMacやWindowsでも動作し、コンピュータ相手やネットワーク対戦プレイが可能だ。さまざまなシチュエーション・ルールのゲームが登録されたゲームライブラリで遊ぶだけでなく、ゲーム記述言語（GDL）を使ってオリジナルゲームをデザインできる。実行にはTcl/Tkが必要だ。

●ビルドからプレイ開始まで

Xconqはソース一式をtar＋gzip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。

ktermなどのコマンドラインで「xconq&」としてタイトル画面付きのウィンドウを開こう（**画面1**）。左下の[New Game]ボタンを押すと、ゲームライブラリに登録されたゲームの一覧に切り替わる。このバージョンでは23種類のゲームをプレイ可能だ。

初めてプレイする場合は、基本操作を学ぶ「Introductory Game」、2回目以降はその発展版「Standard Game」を選択するといい。オプション設定やプレイヤー一覧画面を経て、プレイ用ウィンドウが開く（**画面2**）。



画面1　起動時に表示されるタイトル画面。[New Game]ボタンを押そう。

●ユニットを移動させる

Introductory Gameの目的は、自分のユニット（駒）を操作して、敵の都市を占領することだ。どちらの陣営にも属さない独立都市を占領してさまざまなユニットを生産しつつ、敵の都市を目指すことになる。

ゲーム開始直後は、自分の都市に歩兵（infantry）ユニットが1つだけ存在する（自分の都市やユニットには水色の旗が付く）。自分の都市やユニットの周辺しかマップが表示されないので、歩兵ユニットを動かして周囲を探索しよう。

ユニットの操作にはマウスを利用する。通常の「移動モード」（ポインタが丸付き十字）では、カーソル（点滅する四角）に囲まれたユニットが、クリック位置を目指して移動する。ユニットが移動するにつれ、周囲の地形がマップに表示されるはずだ。

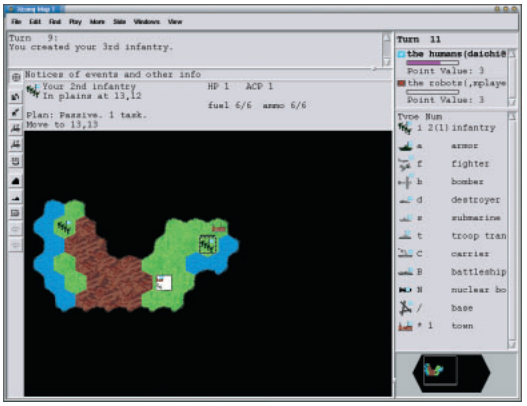
何ターンか進行すると、最初の都市で歩兵ユニットが生産されるので、同じようにして周辺を探索させよう。複数のユニットを同じ場所に重ねること（スタック）も可能だ。

●都市の占領とユニット生産

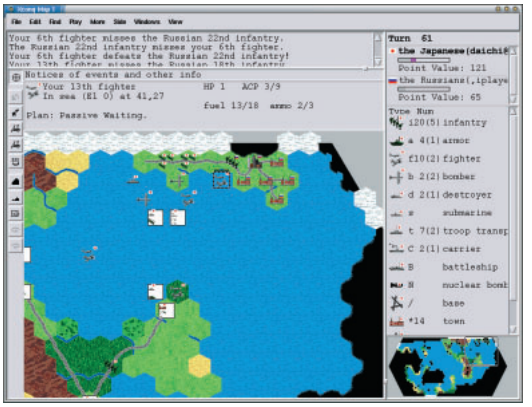
もし、旗が付いていない独立都市を見付けたら、自分のユニットを隣接させ、その街をクリックすることで攻撃できる。ある程度攻撃が成功すると、その街を占領して自分のものにできる（水色の旗が付く）。

占領した街では、新たなユニットの生産が可能だ。街にカーソルが表示された状態で、左端上から2つめのボタンを押し、生産するユニットの種類を選択する。歩兵のほか、機甲部隊（armor）、戦闘機（fighter）、駆逐艦（destroyer）などを生産可能だ。

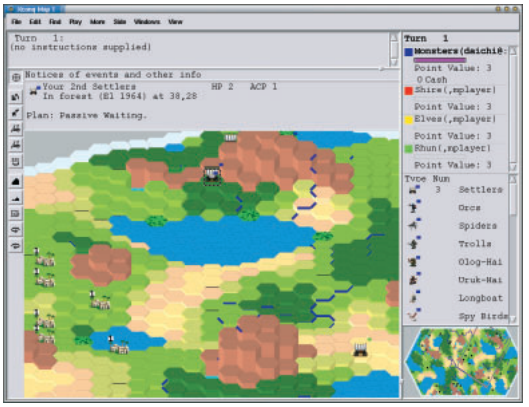
機甲部隊は足が速いが平地しか進めない、戦闘機はどの地形にも進めるが燃料切れの心配がある、といった具合



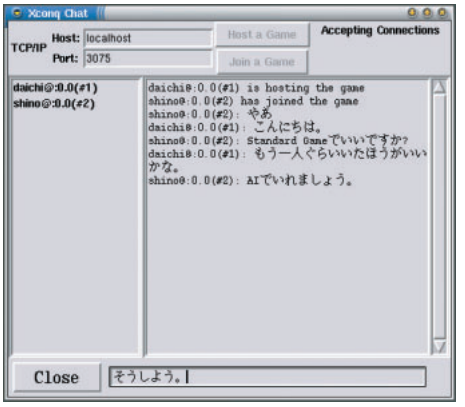
画面2　プレイ用ウィンドウ。マップ表示は三段階に拡大・縮小できる。



画面3　よりマップが広くなり、敵も手ごわくなるStandard Game。



画面4　高低差のあるマップを持つ「Third Age Middle Earth」



画面5　複数のプレイヤーがネットワーク対戦を行うことも可能だ。

に、ユニットごとに長所と短所があるのでバランスよく生産したい。

生産するユニットの種類をあとで変更するには、左上端のボタンを押して「調査モード」に切り替える。その後、目的の都市をクリックし、生産ユニットの種類を選択すればいい。

こうして生産したユニットを動員し、敵の都市を占領できればあなたの勝ち、逆に自分の都市をすべて占領されるあなたの負けだ。

●新たなゲームに挑戦

ゲーム中の操作やゲームシステムに慣れたら、Standard Gameをプレイしてみよう（**画面3**）。ルールはIntroductory Gameと同じだが、マップがより広くなり、敵のAIも手ごわくなる。また、ゲーム開始時のオプション設定により、複数の敵陣営を設定して戦うことも可能だ。

広いマップでは、他のユニットを搭載可能なユニットを活用することが勝利の秘訣だ。たとえば、海を越えて歩兵や機甲部隊を侵攻させるには、爆撃機（bomber）や軍隊輸送艦（troop-transport）を使う。また、燃料切れが心配な戦闘機の運用には航空母艦（carrier）が欠かせない。

その他のカテゴリーのゲームには、Standard Gameとはまったく異なる特徴を備えているものもあるので、マニュアルの「ゲームライブラリ」の解説を参照されたい。たとえば、指輪物語の世界を舞台にした「Third Age Middle Earth」では、高低差のあるマップでプレイする（**画面4**）。

●ネットワークで対戦プレイ

Xconqでは、コンピュータのAIを相手にプレイするだけでなく、複数の人間相手にネットワークプレイを行うことも可能だ。

ネットワーク対戦の場合、いずれか1人のプレイヤーのマシンをホストとして利用する。Xconqのタイトル画面で[Connect]ボタンを押してチャットウィンドウを開き、[Host a game]ボタンを押せばいい。

残りのプレイヤーは、チャットウィンドウ上部の[Host]にホストとなったマシンのホスト名を入力し、[Join a game]ボタンを押せばゲームに参加できる。参加者どうしでチャットすることも可能だ（**画面5**）。

プレイヤーが集まったら、ホストを受け持つプレイヤーがゲームの選択やオプション設定を行い、プレイ開始だ。あとは、コンピュータAIを相手とする場合と同じ操作でゲームを進めればいい。

●日本語版Xconqも開発中

「Japanized Xconq」（http://www.sol.dti.ne.jp/~mniw/）では、英語版マニュアル（CD-ROMに収録）の翻訳と、Xconq 7.5（開発版）をベースとした日本語化が進められており、メニューやダイアログ、一部のゲームの表示が日本語化される（**画面6**）。

ただし、作業中の段階なのでtarボールは用意されていない。日本語版をビルドするためには、上記URLからwgetなどを利用して日本語化されたソースを取得し、CVSを使って取得したXconqの開発版ソースに上書きする必要がある。

画面6　日本語化されたXconq。完成が待ち遠しい。

コンピュータ相手に囲碁をプレイ

Cgoban/Gnu GO

バージョン：1.9.11　　ライセンス：Artistic

http://www.igoweb.org/~wms/comp/cgoban/index.html
http://www.gnu.org/software/gnugo/gnugo.html（Gnu GO）

Cgobanは、美しい盤面で囲碁を楽しめるソフトだ。ローカルで人間どうしが交互に打つほか、IGS／NNGSなどの囲碁サーバに接続してネットワーク対戦を行える。Gnu GOなど「go modemプロトコル」に対応したプログラムを用意すれば、コンピュータ相手にプレイしたり、コンピュータどうしの対戦を観戦することも可能だ。このほか、棋譜を保存したSGFファイルを閲覧・編集する機能も用意されている。

●ビルドとインストール

Cgobanはソース一式をtar＋gzipしたtarボールのみ配布されている。ソースを展開したら、Makefile.inの39行目の「@PREFIX@」を「@prefix@」（小文字）に変更しておこう。

ビルドの手順は、configureスクリプトを使う一般的なものだが、「./configure --prefix=/usr/local」として/usr/local以下へのインストールを指示する必要がある。あとは、「make」→「make install」でOKだ。

Cgoban自身には、コンピュータ対戦用の思考ルーチンは含まれていない。そのため、一人で遊ぶには「go modemプロトコル」対応の囲碁ソフトを別に用意する必要がある。今回は、「Gnu GO」を利用する。Gnu GOは、tarボールのみ配布されており、ビルドとインストールの手順はCgobanと同じだ。

なお、囲碁に興味はあるが、実際に遊んだことはないという人は、最初に「インタラクティブ囲碁入門」（http://playgo.to/interactive/index-j.html）をご覧になることをお勧めする。実際に盤面を操作しながら（Java対応のブラウザが必要）、囲碁のルールやゲームの進め方などの基本知識を学べる優れたページだ。

●人間どうしのローカルプレイ

「cgoban&」として起動すると、ボタンが並んだコントロールウィンドウが開く（**画面1**）。フォントサイズ（12ポイント）が小さすぎるようなら、起動時オプションで変更できる。14ポイントにするなら、「cgoban -font Height 14&」とすればいい。なお、フォントサイズは設定ファイル（~/.cgobanrc）に保存され、次回からは自動的に指定したサイズが使われる。

人間どうしがローカルで対戦するには[New Game]ボタンを押す。ゲームセットアップウィンドウが開くので、プレイヤー名やルール、時間システム、碁盤の大きさ、コミなどを設定しよう（**画面2**）。不明な点があれば、[Help]ボタンでヘルプを参照できる。

セットアップが完了すると、美しい碁盤を持つウィンドウが開いてプレイ開始となる。操作は簡単で、プレイヤーが交互に石を打ちたい場所をクリックすればいい（**画面3**）。

右側に並んだボタンを操作することで、1手ずつ戻す（あるいは進める）ことや、棋譜をSGFファイルに保存して、編集ウィンドウで局面の検討を行うことも可能だ（**画面4**）。

勝負がついたら、死に石の指定などをプレイヤーが行う。[Done]ボタンを押すとコンピュータが地を数えて何目勝ちか判定してくれる。

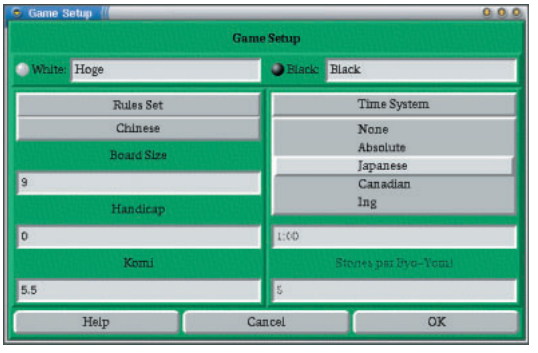
●Gnu GOを相手にプレイする

一緒に遊んでくれる人がいないなら、Gnu GOを使ってコンピュータ相手に打とう。ただし、Gnu GO単独では、画面表示がさびしいので、Cgobanと組み合わせて使う。

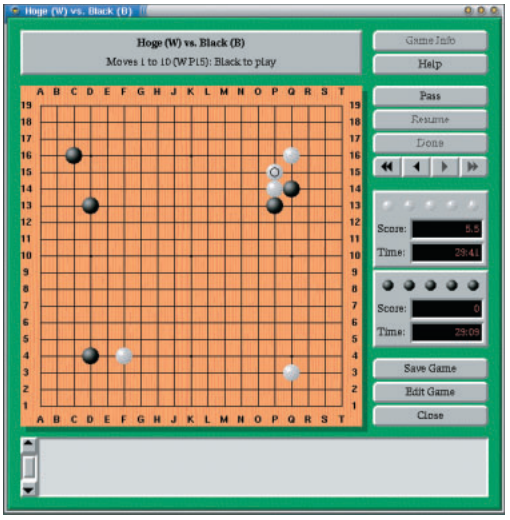
両者の仲立ちをするのが「go modemプロトコル」だ。本来は、モデムを使った通信対戦用のプロトコルだが、今回のように囲碁ソフトとのデータのやり取りにも利用できる。

コントロールウィンドウで[Go Modem]ボタンを押すと、セットアップウィンドウが開く（**画面5**）。あなた

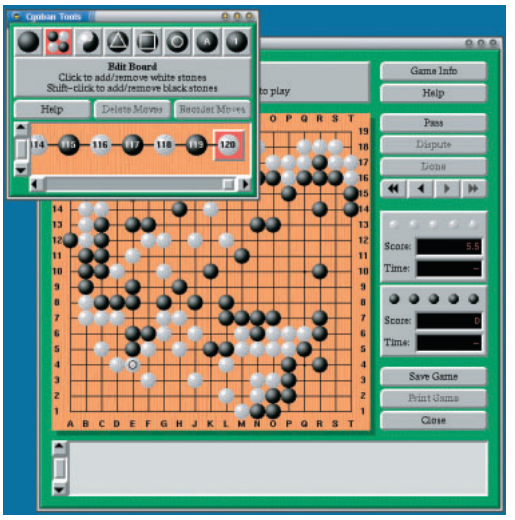
画面1　ボタンが並んだコントロールウィンドウ。



画面2　プレイヤー名や碁盤の大きさなどを設定してプレイ開始だ。



画面3　美しい盤面上にマウスクリックで石を打っていこう。



画面4　棋譜が記録されたSGFファイルを読みこんで対局を検討する。

が担当するプレイヤーは「Human」、Gnu GOが担当するプレイヤーは「Program」を選択し、[Program]には「/usr/local/bin/gnugo --mode gmp --level 2」と入力しよう（レベルの数字が大きいほど弱くなる）。

あとは、人間どうしで対戦する場合と同様にルールを設定し、プレイすればいい。Gnu GOの番になると、自動的に石を打ってくる。レベル設定が小さい（強く設定する）と、一手ごとに時間がかかるので注意しよう。

●囲碁サーバでネットワーク対戦

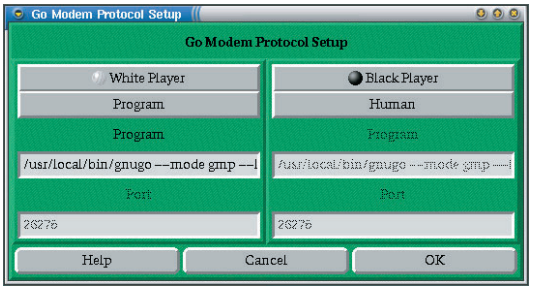
コンピュータとの対戦にも飽きてきたら、いよいよネットワーク対戦に挑戦だ。インターネットで全世界のプレイヤーと囲碁を打てるサーバソフトには、商用の「IGS」（Internet Go Server）や、フリーの「NNGS」（No Name Go Server）などがある。本来はtelnet接続でプレイするのだが、現在ではCgobanなど専用のクライアントソフトを使うのが一般的だ。

国内では、Panda NetがIGS接続サービスを有料で提供しているほか、WINGなどの無料で遊べるNNGSクローンサーバがいくつかある。もちろん、アメリカの本家NNGSや台湾のLGSに接続しても構わない（**リスト1**）。

コントロールウィンドウの左には、サーバ接続用のボタンが2つ用意されている。初期設定では「NNGS」と「IGS」だが、Shift－クリックで切り替え可能だ。サーバの追加や設定変更は[Setup]ボタンで行う。

接続用のボタンを押してユーザー名とパスワードを入力すると、サーバに接続して端末ウィンドウが開く（**画面6**）。プレイヤーやゲームの一覧を表示したり、他人の対局を観戦することも可能だ。対局の申し込みや受諾、メッセージ送信などはコマンドのキー入力で行う。

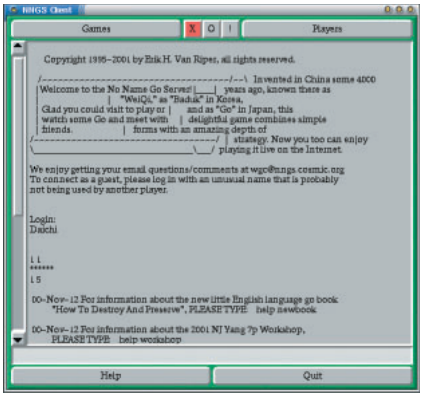
なお、たいていの囲碁サーバでは、ゲストとしてログインした後、会員登録を行うことで、レーティング（強さの指標）などを管理してくれる。会員登録の方法はそれぞれ異なるので、各サーバのWebページの説明などを参照されたい。



画面5　go modemプロトコルを利用してGnu GOを相手にプレイする。

Panda	Net IGS（有料）	http://www.joy.ne.jp/panda/
NNGS	NNGS（無料）	http://nngs.cosmic.org/
WING	NNGS（無料）	http://www2.tky.3web.ne.jp/~komei/wing/indexj.html
LGS	NNGS（無料）	http://www.lgs.hinet.net/eng/

リスト1　インターネット対戦できる囲碁サーバ



画面6　囲碁サーバに接続して、ネットワーク対戦も可能だ。

Xのマウス操作をキーボードで代替する

キーキーマウス

バージョン：1.4α　　ライセンス：フリー

http://www.geocities.co.jp/SiliconValley-Bay/7584/key2mouse/index.html

キーキーマウスは、X上でのマウスポインタの移動やボタンのクリック、ホイールの回転などをキー操作でシミュレートするソフトだ。キーボードから手を離したくない上級者や、ポインティングデバイスが使いにくいノートパソコンユーザーにお勧めだ。デフォルトのキー割り当てはRogueライクゲームのangband風だが、設定ファイルを変更してリビルドすることで、自由に変更できる。

●ビルドとインストール

キーキーマウスは、tarボールとRPMパッケージの両方で配布されているので、自分の環境に合わせて選択しよう。なお、RPMパッケージはデフォルトのキー割り当て（angband風）でビルドされている。

tarボールからのビルドは「make」とするだけだ。インストーラは用意されていないので、作成された実行ファイル（key2mouse）を、/usr/local/binや~/binなど、環境変数PATHに設定されたディレクトリに手動でコピーする必要がある。

●ダブルクリックやドラッグも可能

ktermなどのコマンドラインから「key2mouse&」としてバックグラウンドで実行すると、Ctrl－Meta－jキーで「マウスモード」に切り替わるようになる（MetaキーはAltキーに割り当てられていることが多い）。

マウスモードでは、ポインタの形状が左向きの曲がった矢印に変化する。再度Ctrl－Meta－jキーを押すか、Escキー（Enterキーやqキーでも可）を押すことで元の状態に戻る。

マウスポインタの移動は、hキー（左）、lキー（右）、kキー（上）、jキー（下）に割り当てられている。Shiftキーと組み合わせると動きが小さく、Ctrlキーでは大きくなる。斜め方向への移動も可能だ（**表1**）。

ドキュメントには、「angband風」と記されているキーバインドは、RogueやNethack、viとも共通なので、これらのソフトに慣れていれば、自然に使えるだろう（斜めはNethackのみ）。

マウスの左／中／右ボタンは、それぞれs／d／fキーに割り当てられている（**表2**）。操作方法は本物のボタンとまったく同じで、一度押すとクリック、素早く二度押すとダブルクリック、押しつづけたままポインタを移動させるとドラッグとなる。

このほか、Xの設定ファイル（/etc/X11/XF86Config）のマウスに関する設定で、4、5番目のボタンを割り当てている場合は、↑／↓キーがそれぞれホイールの上／下回転に割り当てられる（詳細はREADMEを参照）。

なお、マウスモード中も本物のマウスによる操作は有効なので、ポインタの移動はマウスを使い、ボタン操作だけキーで行うとことも可能だ。ノートパソコンでは、この組み合わせを使うとドラッグ操作が容易になるのではないだろうか。

●キー割り当てのカスタマイズ

キー割り当てを変更するには、ソースに含まれる設定ファイル（config.h）を直接書き換え、再度ビルドとインストールを行う必要がある。

なお、ソース展開先のexampleディレクトリには、設定ファイルのサンプルがいくつか用意されており、これらを元のconfig.hと差し替えるだけで、簡単にvi風やEmacs風のキー割り当てを実現できる。

さらにこれらの設定ファイルをベースに記述をひとつずつ書き換えて、専用の割り当てを作り出すことも可能だ（詳細はREADMEを参照）。

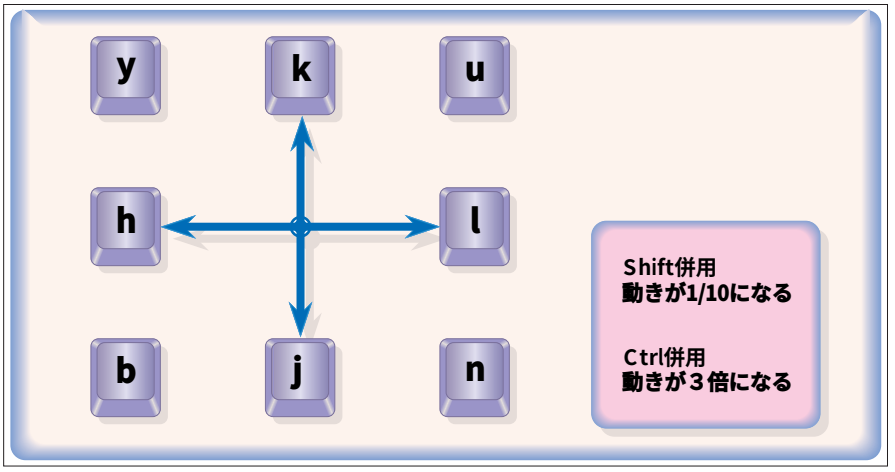


表1　マウスポインタの移動に使用するデフォルトのキー割り当て

s	左ボタン
d	中ボタン
f	右ボタン
↑	ホイール上方向
↓	ホイール下方向

表2　ボタンとホイール操作に使用するキー割り当て

テキストベースのRPM管理ソフト

Purp

バージョン：1.0.0　　ライセンス：GPL

http://www.lysator.liu.se/purp/

Purpは、Red Hatパッケージマネージャ（RPM）をテキストベースで対話的に管理するソフトだ。インストール済みのパッケージや、これからインストールするパッケージの詳しい情報を画面上で確認しながら、インストール・更新・削除などをカーソルキーや英字キーで行える。Xのない環境のコンソールや、telnetで接続した端末ソフトからも使用可能だ。動作にはncursesライブラリが必要となる。

●ビルドとインストール

Purpは、tarボールとRPMパッケージの両方が配布されている。RPMパッケージはRed Hat 6.2用と7用が別に用意されているので、自分の環境に合わせて選択しよう。

Red Hat 7用のパッケージが用意されていることからも分かるように、PurpはRPMの新しいバージョン（4.x）にも対応している。ただし、listdc++とncursesを更新しないと、起動時にクラッシュするので注意されたい。

●対話的な操作でRPMを管理

スーパーユーザー（root）になって、コンソールや端末ソフトのコマンドラインで「purp」とすると、インストール済みのパッケージの一覧が画面左に表示される。なお、X上で利用する場合、ktermでは文字や背景の色が正確に反映されないので、rxvtやxtermを使うとよいだろう。

インストールしたいパッケージがある場合は、「purp /mnt/cdrom/Kondara/RPMS」のように、そのファイルが置かれたディレクトリを指定すると、ディレクトリ内のパッケージ情報が画面右に表示される（**画面1**）。

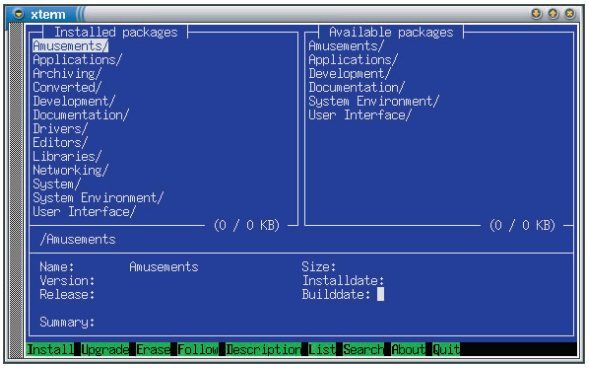
どちらの情報も、パッケージのジャンルに対応した仮想的なディレクトリで分類されている。カーソルキーとEnterキーでディレクトリを切り替え、具体的なパッケージ名が表示されている画面まで進もう。Tabキーを押すと、カーソルを左右の情報に交互に切り替えられる。

具体的なパッケージにカーソルを合わせると、画面下にそのパッケージの概略が表示され、英字キーの入力でさまざまな操作を行える（**画面2**）。

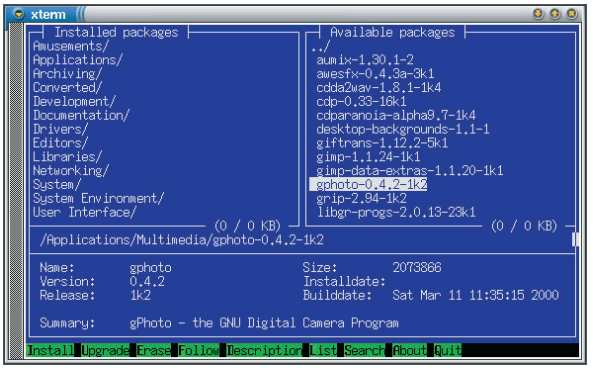
たとえば、dキーではパッケージの詳しい説明が表示されるし、lキーではファイル一覧や依存関係などを参照できる（**画面3**）。sキーによるパッケージの検索や、fキーによる左右のディレクトリの同期も可能だ。

十分な情報を得たら、いよいよインストールだ。画面右側（未インストール）でパッケージにカーソルを合わせ、iキーでインストール、uキーで更新される。また、画面左側のインストール済みパッケージにカーソルを合わせ、eキーを押せば、削除できる。

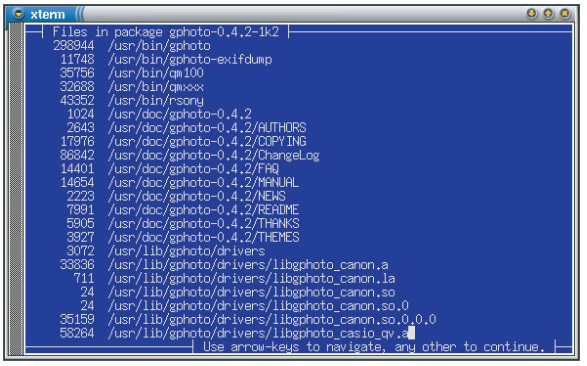
これらのキーは、コマンドラインから利用するrpmコマンドのインストール・更新・削除と同じなので、覚えやすいだろう。



画面1
RPMパッケージは仮想的なディレクトリで分類されている。



画面2
パッケージ名にカーソルを合わせると、概略情報が表示される。



画面3
英字キーの入力で、パッケージに含まれるファイル一覧を確認。

Software Catch Up

豊富な機能と柔軟なカスタマイズが可能なオフィススイート

Applixware Office for Linux 5.0 日本語版

Applixware Officeは、ワープロ、表計算、プレゼンテーション、グラフィックス、データベースアクセス、メールなどが統合されたオフィススイートだ。それぞれの機能も充実しており、完成度の高いソフトウェアだといえるだろう。

文：塩田紳二
Text : Shinji Shioda

価格　1万4800円
問い合わせ先　ターボリナックスジャパン
03-5766-1188
http://www.turbolinux.com/

Applixware Office 5.0（**画面1**。以下、Applixware）は、Vistasource Inc.の製品（日本語版の発売はターボリナックスジャパン）だが、同社は、かつてApplixの一部門であったものが独立した会社である。Applixは現在、企業向けシステムなどがメインのビジネスになっており、Linux用のオフィスアプリケーションであるApplixwareは、少し傾向が違うため別会社での販売という形になったのではないかと思われる。

Applixwareは、Linux用だけでなく、FreeBSDやSolaris、AIX、Alpha UNIX、HP-UX、SGI IRIX、Windows用などがある（ただし、最新のバージョン5.0はx86 Linux用のみ）。このように、Applixwareはもともとワークステーション向けに販売されてきたこともあり、UNIX系オフィスソフトとしては、歴史の長いものになる。

Linux用としては、Turbolinuxのオプションアプリケーションとして、日本語化されたのち、単体アプリケーションとなった。

動作環境として、英語版Applixwareは以下のディストリビューションをサポートしている。

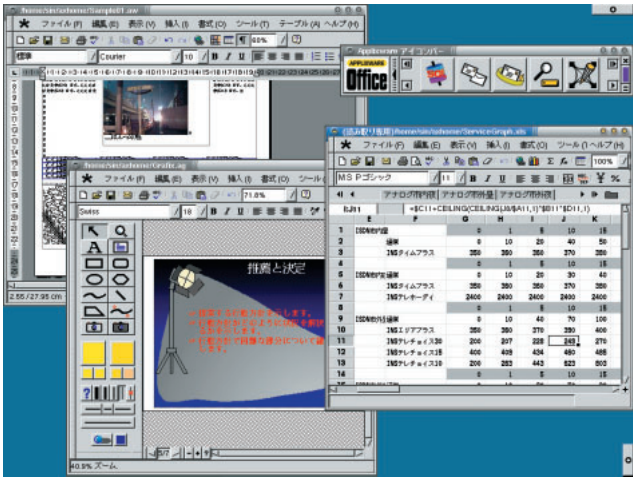
Caldera OpenLinux 2.3、2.4
Debian GNU/Linux 2.1
Red Hat Linux 5.x、6.x
SuSE Linux 5.3、6.x
Corel Linux 1.x
Turbolinux 6.x
Slackware 7.x
Linux Mandrake 7.x

日本語版では、以下のディストリビューションでの動作確認がされているようだ。

Turbolinux Workstation日本語版6.0
Turbolinux Workstation日本語版6.0 リミテッドエディション
LASER5 Linux 6.0
LASER5 Linux 6.2
Red Hat Linux 6.2J
Kondara MNU/Linux 2000

ただし、Turbolinux以外のディストリビューションでの動作保証とサポートは行われない。

必要CPUは、Pentium 166MHz以上、メモリ32Mバイト、ハードディスク容量は最低133Mバイトで、標準インストールを行うには、236Mバイト必要となる。これらのスペックは、現在では、高いとはいえないので、ほとんどの場合には利用可能といえるだろう。



画面1　Applixware Office
Applixware Officeは、ワープロ、表計算、プレゼンテーション作成ソフトなどを含むオフィススイートだ。

なお、Applixware Office 5.0は、GUI用にGTK+（1.2.6）を必要としている。こちらも最近のディストリビューションであればほとんど問題ないと思われる。

このApplixwareは、ELFという言語を使って作られており、このELFはSHELFという開発環境を持っている。Applixwareに含まれる各アプリケーションは、実際にはこのSHELFで作られたものであり、SHELFを使うことで、Applixware自体を制御することもできるようになっている。いわば、Microsoft OfficeのVBA（Visual Basic Application editon）のように、Applixwareの各アプリケーションの動作をELFで細かく制御できるのである。

各アプリケーションはネイティブコードで作られてはいないが、ELFはコンパイルされて動作するため、実行速度はそれほど遅くない。

このSHELFは、Applixware自体にも含まれており、マクロエディタやデバッガなどの環境も用意されている。なお、SHELF自体は単体でも利用可能で、LGPLにより公開されているため、これを使ったアプリケーションを作成することもできる（http://shelf.sourceforge.net/を参照）。

さて、このApplixwareだが、以下のようなアプリケーションが用意されている。

- **アイコンバー**
  起動ツール
- **Applix Words**
  ワードプロセッサ
- **Applix Spredsheets**
  表計算
- **Applix Presents**
  プレゼンテーション作成
- **Applix Graphics**
  グラフィックス作成
- **Applix Data**
  データベースフロントエンド
- **Applix Mail**
  メールソフト
- **Applix HTML Auther**
  HTMLエディタ

ただし、前述のようにSHELFで作られたプログラムであり、それぞれの機能が独立したプログラムではないため、構成としてはちょっと複雑である。実際には、アイコンバーがApplixwareの中心となるプログラム（名前はapplix）で、ここから各アプリケーションに対応するマクロが呼び出され実行される。また、主要なアプリケーションはapplixの起動オプションとしても実行可能だ。

アイコンバーは、簡単にいうとSHELFのプログラムランチャであり、Applixwareの実体そのものでもある。

マクロ関連機能としては、

- **Applix Builder**
  アプリケーション開発環境
- **Applix Macro Editor**
  マクロ作成ツール
- **Applix Dialog Box Editor**
  ダイアログボックス編集GUIツール
- **Applix Bitmap Editor**
  ビットマップ編集ツール

などが用意されている。ELFアプリケーションは、Applix Builderを使って作成する。これは、Windows用の開発言語システムのように統合環境となっており、ここから、マクロエディタやデバッガ、ダイアログボックスエディタなどを起動してアプリケーションを作成する。

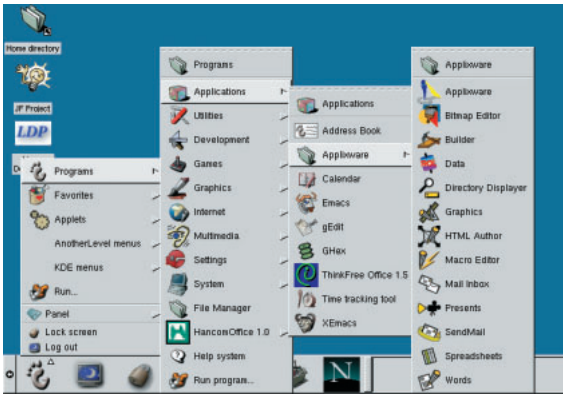
マクロエディタは、Applix Wordsの基本機能を流用して作られたテキストエディタで、ここからもマクロのコンパイルや起動などが可能である。

これ以外に、Applixware用のツールとして、

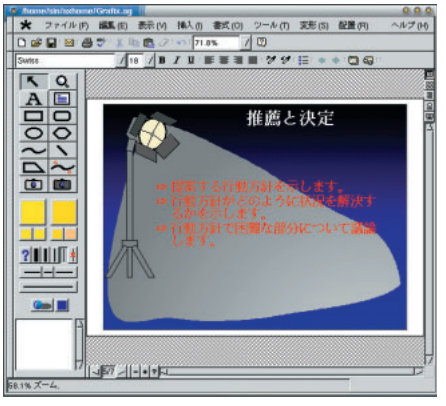
- **環境設定**
- **フォントインストーラ**
- **ディレクトリー覧（Directory Displayer、簡易ファイルマネージャ）**

などが用意されている。

なお、Applix WordsとApplix Spreadsheetsについては起動用のプログラム



画面2　メニューの登録
インストールすると、GNOMEやKDEのメニューにApplixwareのアイコンが登録される。



画面3　ウィンドウ構成
Applixwareのオフィスアプリケーションは、ウィンドウ上部にメニューバーやツールバー（エクスプレスライン）があり、下部にステータスバーが付く。また、グラフィックス関連のツールは、ウィンドウ左側にツールパレットが表示される。

が別にあり、WordsやSpreadsheetsといった名前でも起動することが可能になっている。

インストールは、専用のインストーラを使って行われる。実際にインストールされるプログラムは、RPM形式で格納されている。

このインストーラはGUIを使ったもので、X Window Systemの上で動作する。なお、標準ではApplixwareは、/opt/applixへインストールされる。

インストールの際、インストーラがGNOMEなどのメニューへの登録も自動的に行う。

オフィスアプリケーション

Applixwareは、ワープロ、表計算、プレゼンテーション作成、ベクトルグラフィックス作成と、データベースアクセス機能を持ち、基本的なオフィス作業に必要なものはほとんど揃っている。また、電子メールソフトも含まれており、これを使うことで一般的なアプリケーションユーザーでもLinux上でオフィス作業が行えるようになっている。

■基本GUI

Applixwareは、前述のようにSHELFから構築されており、主要アプリケーションのGUIスタイルは統一されている。ウィンドウは、上部にメニューバーとツールバーが並び、下部にステータスラインが表示される（**画面3**）。

メニューバーには、必ず左端に「＊」マークのメニューがあり、ここでほかのApplixwareアプリケーションを起動したり、アプリケーション自身の環境設定などを行うことができる（**画面4**）。また、右端はかならずヘルプメニューとなっている。メニューは、サブメニューのあるものは、後ろに右向き三角が表示され、ダイアログボックスが表示されるものはメニューテキストの後ろに「...」が付く。また、トグル動作を行うものについては、メニュー文字先頭に四角いトグルボタンが表示される。

いわゆるツールバーは、Applixwareではエクスプレスラインと呼ばれ、表示のオン／オフが指定可能になっている。なお、このエクスプレスラインとメニューバーは、左端の部分をマウスでドラッグすることで取り外してフローティング状態にすることもできる（**画面5**）。ただし、ウィンドウ上部以外の場所に配置することはできない。

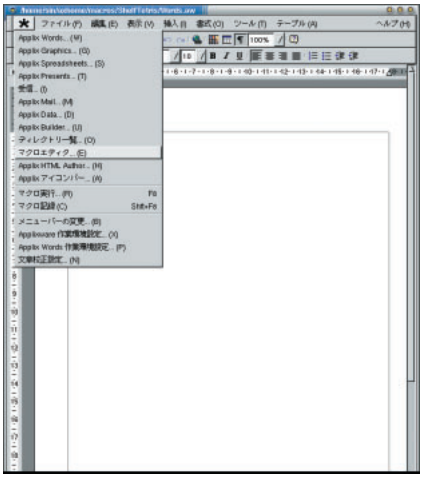
さらに、このメニューバーとエクスプレスラインは、カスタマイズが可能である。これは、「＊」メニューの「メニューバーの変更」で行う（**画面6**）。アプリケーション自体がマクロ言語で記述されているため、メニューやエクスプレスラインのボタンの機能もマクロを指定して行う。このため、自作マクロなどを組み込むことで、簡単に機能拡張が可能となる。

また、「＊」メニューで「マクロの記録」を選べば、ユーザーの行った操作がマクロステートメントとして記録され、特にプログラミングの知識がなくても簡単にマクロを作成することができる。メニューバーのカスタマイズと、このマクロ記録を使えば、たとえば選択範囲をボールドにしてアンダーラインを付けるといった操作を自動処理としてメニューから実行できるようになるわけだ。

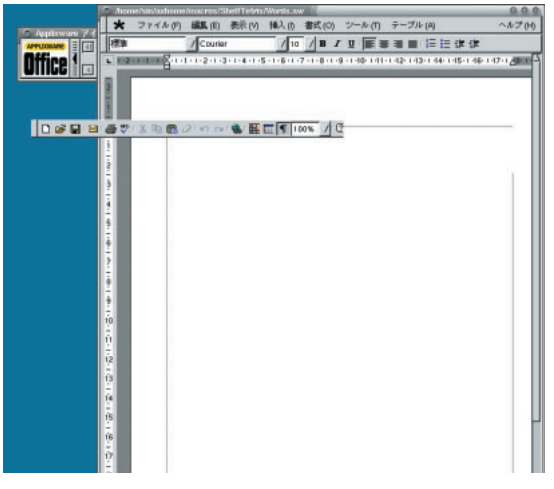
各アプリケーションの作業領域で右クリックを行うと、カーソル位置のオブジェクトなどに応じたポップアップメニューが表示される。多くは、オブジェクトの編集や削除、コピーなどの操作である。

Applix Words

Applix Wordsはワードプロセッサで、印刷を想定したページ単位の編集が可能だ（**画面7**）。メニューバーやツ



画面4　「＊」メニュー
各アプリケーションのメニューバー左端には、「＊」メニューがあり、ここからほかのApplixwareアプリケーションやマクロ機能、メニューバーなどのカスタマイズ、作業環境設定などを行う。



画面5　ツールバーのフローティング
エクスプレスライン（ツールバー）は、標準位置から取り外して、フローティング状態にすることも可能。ただし、ウィンドウ内に配置できるのはメニューバーの下だけである。

ールバーを持ち、WYSIWYG編集が行える。Windows用の平均的なワープロアプリケーションとほとんど同じように扱える。

また、マルチカラムページ（段組み）の作成も可能であり、雑誌記事のような文書の編集にも利用できる。Linux用ワープロソフトの一部のものはマルチカラム編集ができず、その点でWindows用ワープロソフトに見劣りするものがあったが、Applix Wordsはほかの機能を含めて、Microsoft Wordにほぼ近い機能を備えている。この点からみても、移行はそれほど難しくないだろう。

もちろん、DOCファイルの読み込みも可能になっており、Microsoft Wordで作成した文書もそのまま利用できる（**画面9**）。

■GUI

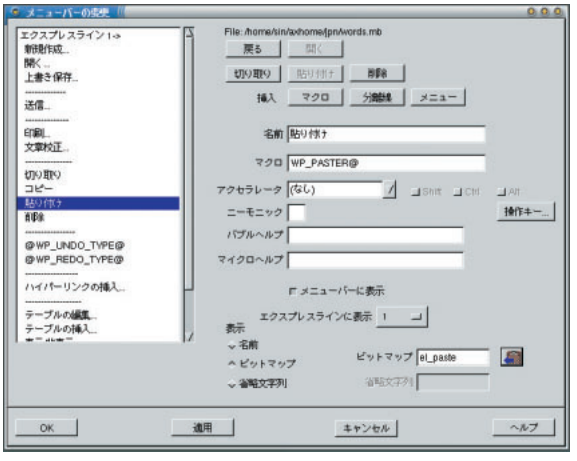
Applix Wordsは、メインウィンドウに文書のページイメージを表示するようになっており、Windows用ワープロソフトが持つようなドラフトモードは持っていない。もっとも、このドラフトモードは、PCの能力が低く、フルレイアウト状態での高速表示が難しかった時代のもので、ないからといってあまり困ることはないだろう。ただし、これに慣れてしまっていると多少違和感があるのも事実だ。なお、ページ表示は拡大も可能なので、小さな文字を使っている場合でも表示を拡大すれば文字が見にくくなることはない。

Applix Wordsには、保存や印刷などの各種ボタンが並んだエクスプレスラインと、書式設定用のエクスプレスラインの2つがある。

文書はページイメージで表示され、ちょうど紙が縦に並んだようにページが作られていく。その周囲にはルーラーが表示され、ルーラーのマージンなどをマウスでドラッグすることで、上下左右の余白などを調整できる。

全般的な感じとして、ダイアログボックスで数値などを入力する場合に、キーボードを使って直接数値を入れさせることが多く、ちょっと面倒な感じがする。

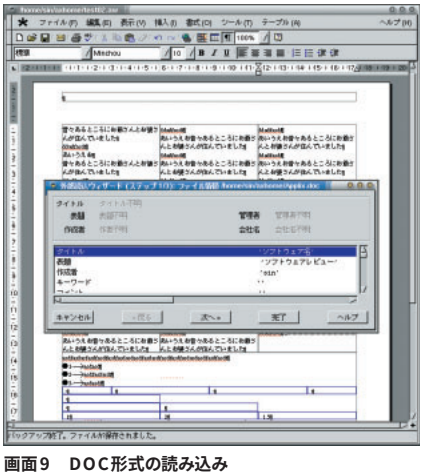
Windows系では、スピンボタンやスクロールバー、あるいはよく利用する数値などは、ボタンなどでキーボードを使わなくとも数値入力が可能になっているアプリケーションが多く、マウスから手を離さずにダイアログボックスへの入力が完了できる。できれば、もう少しこのあたりを見直して欲しいところだ。



画面6　カスタマイズ
各アプリケーションのメニューやエクスプレスラインはカスタマイズが可能だ。ただし、そのためにはSHELFのマクロを使う必要がある。

画面7　Applix Words
Applix Wordsはワードプロセッサであり、文字以外に表やほかのApplixwareアプリケーションのデータや画像ファイルなどの配置が可能。また、段組みにも対応している。

画面8　マルチカラム
簡易なワープロソフトだと、ほとんど実装されていないマルチカラムをApplix Wordsはサポートしている。このため、本誌のページのような段組みの文章をそのまま編集することができる。



画面9　DOC形式の読み込み
Applix WordsやApplix Spreadsheetsなどの主要アプリケーションは、Windows用のほかのアプリケーションのデータファイルの読み込み機能を持つ。読み込み時には、ウィザードが起動し、フォント指定などをどのようにするのかを指定して、元データのレイアウト情報などをなるべく失わない形で読み込みが可能。

■文書作成機能

Microsoft Wordには、文書中に埋め込まれるフィールドと呼ばれる特殊なオブジェクトがあり、これは、たとえば印刷日時などを文中に埋め込む場合などに利用する。Applix Wordsは、このフィールドを持っており、Microsoft WordのDOCファイルに存在するフィールドは、Applix Wordsのフィールドとして利用できる（ただし、Microsoft Wordのすべてのフィールドをサポートしているわけではない）。

また文書は、あらかじめスタイルなどを設定したテンプレートから作成が可能で、一定書式の文書を何度も作成する場合に書式の統一が可能となっている。

文書中には、グラフィックスファイルや表などを挿入でき、フレームを使ってページ内の任意の場所にテキストや図版を置くことができる。このあたりも、Windowsワープロソフトでは一般的な機能である。

さらに、ほかのApplixwareアプリケーションのデータをリンクして埋め込むこともできる。この場合、外部アプリケーションでデータが更新されると、埋め込まれたデータも自動的に更新される。Windowsなどではごく普通に使われている機能であり、このようなアプリケーション間連携ができるのは便利だ。

スペルチェック機能や類義語辞書も用意されているのだが、残念ながら日本語向けのものは用意されていない。英語などの言語用のものは標準で利用できる。

日本語の入力は、標準的なkinput2を使うものであれば利用可能で、カーソル位置での入力変換が行える。ただ、ちょっと困るのがマルチカラム編集の場合で、この場合、ウィンドウサイズに比べて行が短くなり、文書全体の一覧性が低くなってしまう。筆者のように、雑誌原稿などを書く場合、分量の目安や行の区切りを見るためには、段組みを設定したまま編集することが必要で、Microsoft Wordのアウトラインモードとレイアウトモードを切り換えて使っている。しかし、Applixwareではこうした使い分けができず、ちょっと不便な感じがする。もっとも、段組み設定をオン／オフするようなマクロを組み込めばいいのだろうが……。

いわゆる表は、テーブルと呼ばれ、行、列数を指定しての作成や、文字列からの変換による作成が可能だ。セルの結合や分割などの操作は、専用の「テーブル編集」ダイアログボックスを使ってボタン操作で行える（**画面10**）。これは意外に便利だった。なお、テーブル内で自動的に計算を行うような機能はないが、選択テキストを対象とした手動計算機能（合計や平均値など）があり、これを使うことで簡単な集計表程度であれば、作成が可能になっている。

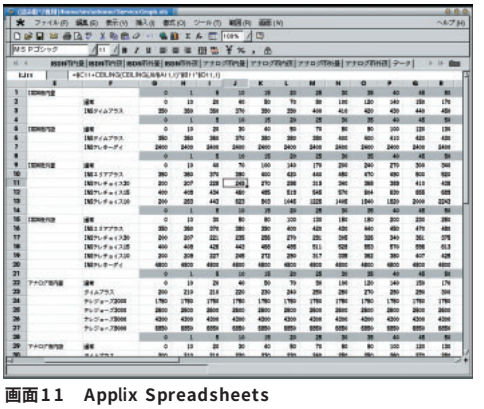
Applix Spreadsheets

Applix Spreadsheetsは（以下、Spreadsheets）表計算プログラムで、全体的な印象は、Microsoft Excelに似ているが、複数ワークシート切り換えのタブが、エクスプレスラインの下に付くなど、多少の違いはある（**画面11**）。

Spreadsheetsは、複数の表計算シートを1つのファイルの中に持つことができる。このあたりは、Microsoft Excelなどと同じである。

実際に使ってみたところ、このSpreadsheetsでは、データ量の多いシートでは多少処理が遅くなるようだ。433MHzのCeleronで8Mバイト程度のMicrosoft Excelファイル（シート数は4枚）の読み込みに7分ほどかかった。

画面10　テーブル編集
Applix Wordsの表機能は、「テーブル編集」ダイアログを使って、比較的簡単に編集が行える。この機能は、エクスプレスラインから起動されるが、マクロで容易に拡張が可能という、Applixwareならではという感じで追加された機能のようだ。



画面11　Applix Spreadsheets
Applix Spreadsheetsは表計算ソフトだ。基本的には、Excelなどと同等の機能を持つ。また、やはりExcelやLotus 1-2-3などのファイルを読み込むことができる。1つのファイル内に複数のシートを持つことができ、Excelなどのファイルもそのまま読み込める。

同じマシンでの比較ではないが、450MHzのPentiumIIIを使ったWindows NT4.0とMicrosoft Excelのシステムでは、7秒ほどで読み込んでしまう。しかも、Spreadsheetsの場合、数式の変換に失敗している部分があった。読み込んだMicrosoft Excelファイルは、数式自体はちょっと長いものの、四則演算とMAX、MIN関数のみのものだ（ただし、大量にある）。具体的には、セルの指定がおかしくなっており、その数式自体が文字列化されている。

Microsoft Excelなどのファイルの変換には、専用のウィザードが起動するようになっており、フィルタと呼ばれる外部マクロを使ってファイル形式の変換をしている。

変換ミスは、おそらくこのフィルタの問題であって、Spreadsheets自体の問題ではないようだが、Spreadsheetsのデータ許容量がMicrosoft Excel並になっているのに、変換が失敗しているのは惜しいところだ。

このほか、シート全体の選択や文字書式の変更などでも少々処理終了を待たされるようになり、このあたりではネイティブコードで書かれたMicrosoft Excelとマクロ（SHELF）で書かれたソフトウェアの違いを感じる。表計算ソフトは、見かけのわりに処理が重いので、処理速度の違いが動作に表れやすい。単純なスクロールやセルの書き換えは、そこそこの速度で動いているので、単純に大量のデータ処理は、マクロが不向きということであろう。

■GUI

Spreadsheetsは、Applix Words同様にメニューバーの下にエクスプレスラインが2つ、そしてその下に数式バー、シートタブが上下に並ぶ。Microsoft Excelのシートタブは、水平スクロールバーと同じ位置にあり、多数のシートを同時に使うとちょっと狭く感じるが、Spreadsheetsではそれぞれが独立しているため、この点は問題ない。

グラフは独立したシートにはできず、ワークシート上にオブジェクトとして配置するのみで、Micorosft Excelなどから変換した場合、空のワークシートが作られ、その上にグラフオブジェクトが配置される（**画面12**）。

グラフオブジェクトはサイズをマウスで変更できる以外、GUIで操作することはできず、すべてメニューやダイアログボックスを使って変更するしかない。Microsoft Excelだと、たとえばX軸をクリックしてプロパティ表示で設定を変更するとか、凡例の位置をマウスでドラッグできるということが可能だが、Spreadsheetsではそこまではできない。感じとしては、初期の頃のMicorosoft ExcelやDOS版のLotus 1-2-3を使っている感じである。

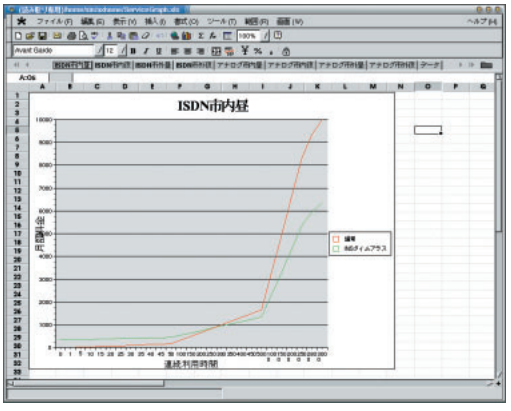
しかしワークシート上では、セルをドラッグして移動したり、選択範囲右下のハンドルを掴んでセルをコピーするなどの操作が行える。このあたりは、Microsoft Excelでも、わりとよく使われる機能で、これがないとMicrosoft Excelに慣れたユーザーは、かなりストレスが高くなってしまう。とりあえずのツボは押さえてあるということだろうか。

数式の入力は、数式バーおよびセル上で直接行うことができる。このあたりもMicrosoft Excelと同じである。

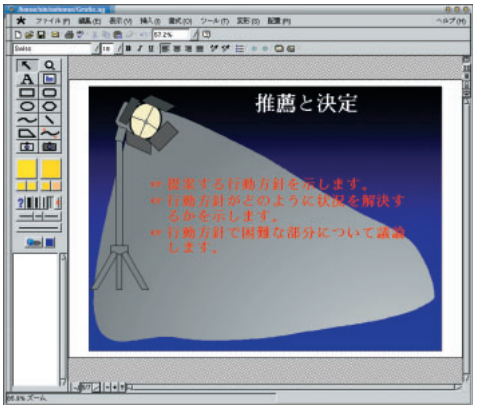
Microsoft Excelでは、1つのワークシートウィンドウを縦横に分割して、たとえば表の見出し部分はそのままに、データ部分だけをスクロールさせるような機能があるが、Spreadsheetsにはその機能はない。しかし、同一ファイルを表示するウィンドウを複数起動でき、それぞれのスクロールを連動させる機能がある。これを使えば、分割表示らしき動作はできるのだが、少し使い勝手が違う。分割表示も使用頻度の高い機能なので、なんとかして欲しいところである。

■計算機能

Spreadsheetsの関数などは248個あり、この中には金融用の関数なども含まれる。なお、一部の関数は外部マク



画面12　グラフ
Applix Speadsheetsのグラフは、ワークシート上に配置するもののみで、Excel側でグラフシートとして作成されていた場合には、空のワークシートが作られそこにグラフが配置される。これにより、実際はほとんど同じ状態になる。



画面13　Applix Presents
Applix Presentsは、プレゼンテーション資料作成ソフトだ。画面上のプレゼンテーションだけでなく、印刷などで資料を作る場合にも対応している。

ロで記述されたアドイン関数で、必要に応じて読み込むことができる。

また関数は、ELFを使って定義することが可能で、マクロエディタで関数を記述し、コンパイルするだけでその関数をワークシート内の式で利用することが可能だ。

シート内には、グラフオブジェクトのほか、ほかのシートの参照や、Applix Words文書、グラフィックスなどを配置することができる。また、マクロボタンを配置し、一連の操作を実行させることもできる。ただし、Microsoft Excelなどではマクロをシートファイル内で管理しているが、Spreadsheetsではマクロを外部の個別ファイルとして管理する。

このほか、データベース機能（シート内のデータをデータベースとして扱う）機能などもある。

インポートできるファイルは、Microsoft Excel、Lotus 1-2-3、SYLK形式、DIF形式、CSV、テキストなどがある。

Applix Presents

Applix Presents（以下、Presents）はプレゼンテーション資料作成ツールだ。Microsoft Officeでいえば、Microsoft PowerPointに相当するソフトである（**画面13**）。ただしこのソフトは、後述のApplix Graphicsとエクスプレスラインやメニューが多少異なっている程度で、ほとんど同じ外観になっている。つまり、基本的な描画機能はApplix Graphicsのものを使い、それにページ切り換えの効果や、スライドショー機能、テンプレートによるスライド作成などの機能を付け加えたものなのである。もっともこの方法は、いままでに見てきた多くのLinux用オフィススイートなどでは普通の作り方でもある。

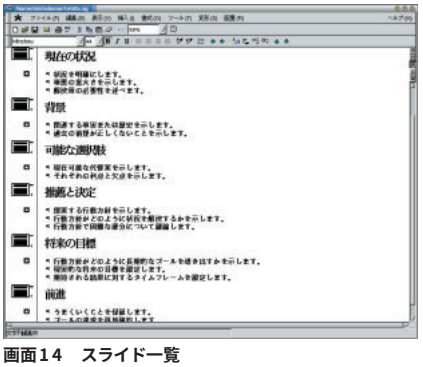
Microsoft Officeでは、従来、Microsoft PowerPointが持っていた描画機能だけをコンポーネント化し、ほかのアプリケーションでも利用できるようにしているが、アプリケーションとしてはPowerPointは単独のアプリケーションである。もともと、Microsoftが他社から会社ごと買い取ったソフトなので、グラフィックスソフトと兼用させて手間を省くといったことは必要なかったわけである。

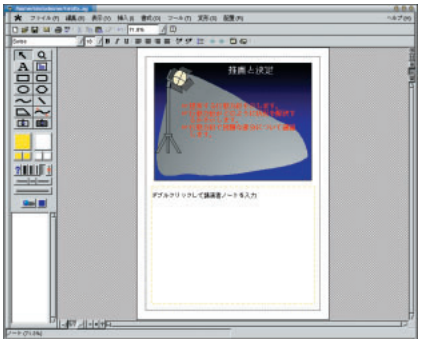
日本国内の世間一般には、PowerPointをグラフィックス作成ツールだと思っている人も多く、PowerPointで絵を作って印刷している人も少なくない。こうした使い方を考えれば、多くのユーザーに必要なのは、グラフィックス描画ソフトなのだが、なぜか、Microsoft Officeには単独のグラフィックス描画ソフトがなく、逆に各ソフトにグラフィックス描画機能が組み込まれている。かつては、それぞれのソフトが別々にグラフィックス描画機能を実現していたのだが、前バージョンあたりから、それがコンポートネント化し、各ソフトで共通になった。逆に、ゼロからアプリケーションを作る側からすれば、ほとんど共通の機能を持つ、グラフィックス描画ソフトとプレゼンテーション作成ソフトを別々に作るのは効率が悪く、グラフィックス描画ソフトを拡張してプレゼンテーション作成ソフトを作るというのは効率が良いわけだ。

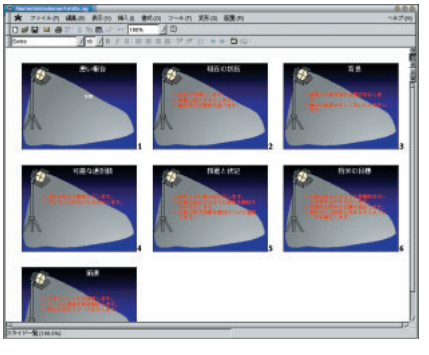
■GUI

Presentsはウィンドウ左側に描画ツールのアイコンなどが並び、その右側に描画領域がある。ウィンドウ上部には、やはりメニューバーやエクスプレスラインがある。

スライドの編集画面のほかに、スライドの並べ替えができる「スライド一覧（**画面14**）」や、テキスト部分のみを編集できる「アウトラインエディタ」、スライドにメモを付ける「ノートエディタ」、画面上でプレゼンテーションを行う「スライドショー」などの機能があり、このあたりは、Microsoft PowerPointと同じだ。







画面14　スライド一覧
Applix Presentsは、通常のスライド編集モード以外に、並べ替えを簡単に行えるスライドリストやテキストのみで編集が行えるアウトラインモード、スライドに対してメモを作成するノートモードなどを備えている。

スライドショーでは、スライド内の項目を効果（**画面15**）を付けて表示（テキストなどを順次表示する）させたり、ページ切り換え効果などを付けられるほか、マウスでスライドに描画することも可能だ（ただし、描画を保存することはできない）。

なお、スライド内のオブジェクトを部品として登録すれば、以後、部品名を指定するだけでそれらを呼び出して利用できるようになっている。この部品リストは、ツールパレットの下に配置されている。

■スライド作成機能

プレゼンテーションファイルは、背景などを指定したテンプレートから作成し、各スライドページは、タイトルや箇条書きなどを配置してある「スライドレイアウト」を使って作成する。スライドレイアウトは、あらかじめ決められている（たとえばタイトルと箇条書きの組み合せレイアウトや、白紙のレイアウトなど）。また、ユーザーがカスタムレイアウトを1つだけ作ることが可能で、これは、最大4つのブロックを配置する基本パターンと、各ブロックの定義（文字列、グラフィックス、表など）を組み合わせて作る。自由度はないが、簡単にカスタムレイアウトが作成可能だ。まったく自由に作りたければ、白紙ページにオブジェクトを配置していけばいいわけで、この方法によりユーザーが頻繁に使うパターンを簡単に登録できるわけだ。

印刷時には、講演者ノートや配付資料（1ページに複数スライドを印刷）、アウトライン、スライドと指定が可能となっている。少なくともMicrosoft PowerPointの基本機能はしっかりと押さえてある。

Applix Graphics

Applix Graphics（以下、Graphics）は、前述のようにPresentsと同じウィンドウ構成になっている。

このため、このソフトは複数ページを同時に作成できるようになっている（**画面16**）。

簡単にいうと、Graphicsは、Presentsからスライドショー関連の機能と、テンプレート、スライドレイアウトなどの機能を省いたものである。まったくの真っ白なページから作業を始めるか、背景やテキストボックスを配置したあとで、作業を始めるかといった違いがあるが、描画機能についてはまったく同じである。

■描画機能

Graphicsは、矩形や円、直線や曲線などのオブジェクトを配置して描画を行う。基本図形は、直線または曲線の組み合せになっており、頂点位置を編集することでさまざまな図形を作ることもできる。

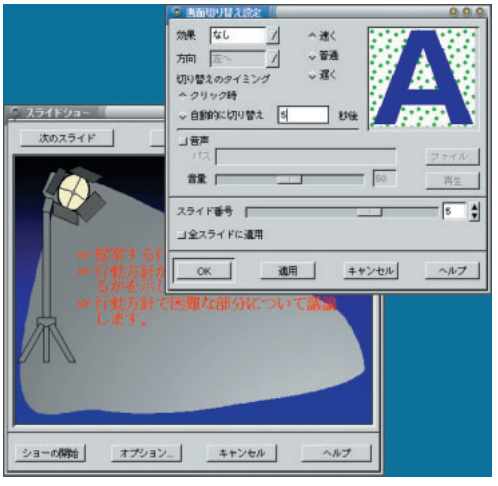
そのほか、ほかのソフトのデータ（たとえば、Spreadsheetsのワークシート）などをオブジェクトとして配置することも可能になっている。

また、各オブジェクトにはコールバックと呼ばれるマクロを関係付けることができ、オブジェクトをダブルクリックしたり、ドラッグしたときにマクロを実行することができる。これを使うと、ちょっとしたGUIアプリケーションなどが作れそうである。

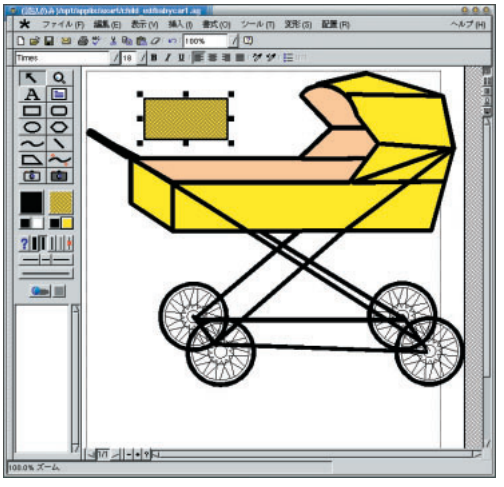
Applix Data

Applix Dataは、SQLを介して、データベースにアクセスするためのフロントエンドプログラムである（**画面17**）。対応しているデータベースは、Informix、Oracle、Sybase、ODBC、ShelfSQL（MySQL）などだ。

このApplix Dataを使うことで、データベースのアクセス結果をオブジェクトとして、Applix WordsやGraphics



画面15　プレゼンテーション効果
画面上でプレゼンテーションを行う場合には、画面の切り替えやオブジェクトの表示に効果を付けることができる。このあたりもMicrosoft PowerPointと同じだ。



画面16　Applix Graphics
Applix Graphicsは、ベクトル図形編集ソフトだ。画面を見てわかるように、実際にはApplix Presentsと画面はほとんど同じだ。このApplix Graphicsをベースに作成されたと思われる。このあたりも、マクロで簡単に機能拡張が可能なApplixwareならではの機能であろう。

などに貼り込んで使うことができる。また、SpreadsheetsやApplix Wordsのテーブルに対してデータベースをリンクさせることも可能だ。

さらに、単体でデータベースを指定して、検索やレコード入力などを行うこともできる。

Applix Mail

Applix Mailは、受信メールの表示を行うInbox（**画面18**）と、メール送信を行うSendmail（**画面19**）の2つのモジュールからなる。Sendmailのほうは、ほかのApplixwareアプリケーションから呼び出され、編集中文書などを送信する場合などに使われている。

Inboxは、POPを介してメールを転送し、受信メッセージをフォルダで管理するためのツールだ。なお、Applixwareには、POPサーバが付属しており、ローカルマシンにPOPサーバがなくとも、POP経由でメールを受信できるようになる。

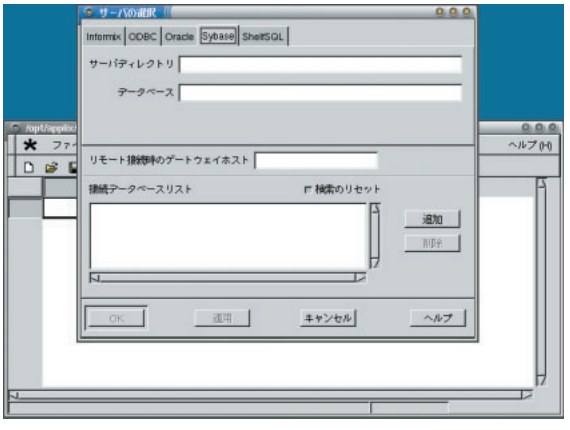
Applixware アイコンバー

Applixwareアイコンバーは、一見、オマケのランチャプログラムのようだが、実際には、SHELFで作成されたELFマクロアプリケーションを動作させる環境を整え、マクロを実行するためのものだ（**画面20**）。一部のアプリケーションは、起動用の実行可能ファイルが用意されているが、Applixwareのアプリケーションの多くは、このアイコンバーからのマクロ起動で実行するようになっている。

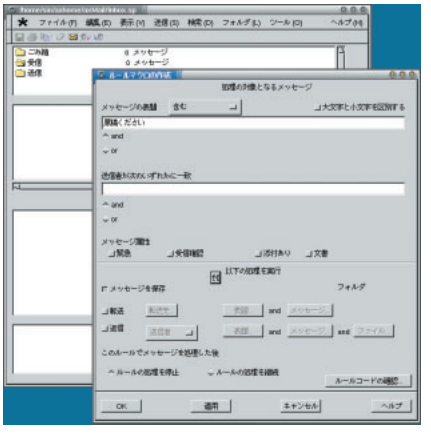
また、Applixware環境設定を使うことで、自分で作ったマクロをアイコンとともにこのアイコンバーに登録することもできる。

アイコンバーウィンドウは、横にアイコンを配置していくもので、ウィンドウサイズは、左右にのみ拡張可能だ。また、ウィンドウ内に表示できないアイコンは、左右にある矢印ボタンでアイコンボタンをスクロールさせて表示する。

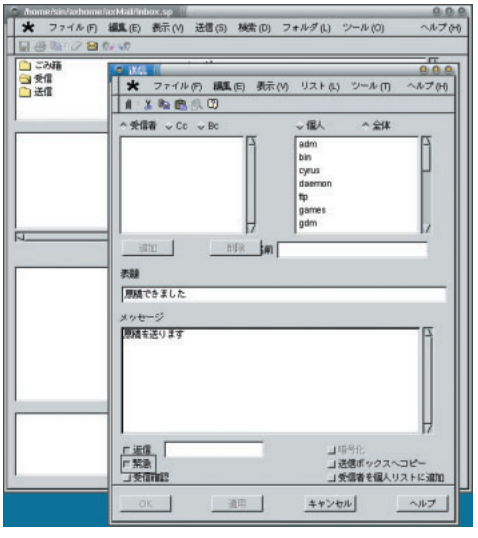
アイコンバーは、ウィンドウ枠などを付けずにウィンドウ左上に配置することもできるし（**画面21**）、アイコンバーを畳んで小さなウィンドウにすることもできるようになっている。



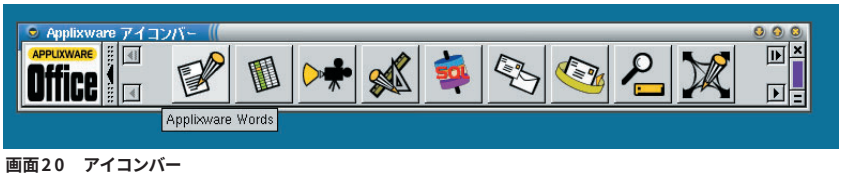
画面17　Applix Data
Applix Dataは、各種データベースのフロントエンドとして動作する。つまり、このApplix Dataを使って、SQLでデータベースに問い合わせを行い、検索結果を取得したり、それらをオブジェクトとしてほかのApplixwareアプリケーションに提供する機能を持つ。



画面18　Applix Mail
Applixwareは、メールソフトも同梱されている。POPによるメールサーバアクセスが可能なため、多くの環境でそのまま利用できる。また、着信メールの振り分け機能なども装備している。



画面19　Sendmail
Sendmailはメール送信ソフトだ。各アプリケーションから、編集中の文書を送信するなどの場合にも利用できる。メッセージは、SMTP経由で送信を行うため、多くの環境で利用できる。



画面20　アイコンバー
Applixwareの各種アプリケーションを起動するためのランチャツール。また、これはSHELFアプリケーションの起動用ツールでもある。カスタマイズにより、ユーザーがSHELFマクロを登録することが可能。

画面21　アイコンバーのカスタマイズ
アイコンバーは、タイトルバーなどを外し、画面左上に配置して使うこともできる。また、ボタン部分をたたんで、小さなウィンドウにすることもできる。

マクロツール

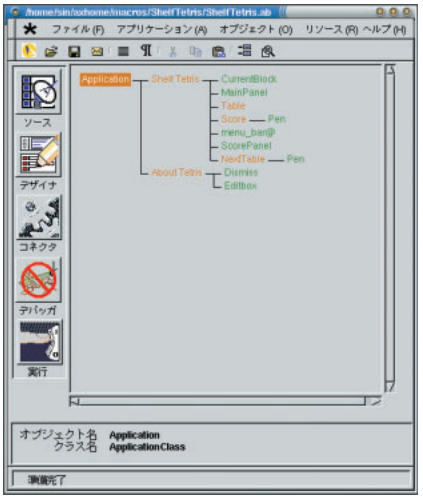
Applixwareのオフィスアプリケーションは、ELFで記述されており、これらの機能拡張やカスタマイズ、あるいは新しいアプリケーションの作成などのために、SHELF開発環境が含まれている。簡単なマクロは、マクロエディタを使って記述することが可能であるし、大規模なアプリケーションもApplix Builderを使うことである程度開発が可能になる。

■Applix Builder

Applix Builderは、データベースやダイアログボックスの表示などを行うアプリケーションを作成するための開発環境である（**画面22**）。ここでは、記述したマクロがツリー表示され、ここから各種マクロツールを起動して開発を行うことができる。

アプリケーションは、データベースなどの情報源である「ソース」、ダイアログボックスを作成、編集する「デザイナ」、ダイアログボックスとソースを結び付ける「コネクタ」および、エディタを使って作成する。「ソース」および「コネクタ」はデータベースのフロントエンドアプリケーションなどを作成する場合に利用し、デザイナやマクロエディタのみでアプリケーションを作ることもできる。

また、作成したアプリケーションは、デバッガを使って、ステップ実行や変数の表示などが行える。



画面22　Applix Builder
Applix BuilderはSHELFの開発環境だ。データソースやダイアログ、マクロエディタ、デバッガなどを統合しており、ここでSHELFアプリケーションの開発が行われる。また、作られたアプリケーションは、メイン領域にツリー構造として表示される。

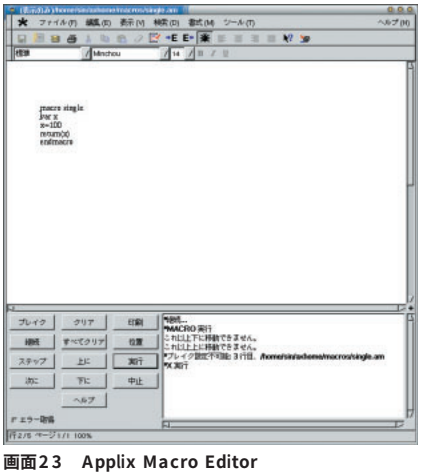
■Applix Macro Editor

Applix Macro Editor（**画面23**）は、Applix Builderからも呼び出されるが、単独でも利用することができる。たとえば、Spreadsheetsの関数などの簡単なプログラムは、このマクロエディタのみで作成することもできる。また、このマクロエディタには、マクロのコンパイルやステップ実行が可能な「デバッガモード」があり、ソースコードを見ながら、動作試験なども行える。

■Applix Dialog Box Editor

Applix Dialog Box Editorは、あらかじめ定義してあるコントロールを配置して、ダイアログボックスを作成するものだ（**画面24**）。SHELFには、ラジオボタンやトグルボタンなど、全部で13種類のコントロールが用意されている。

これらのコントロールをマウスで配置していくことで、ダイアログボックスが作成できる。作成したダイアログボックスは、専用のファイルに格納される。各コントロールには属性があり、また、IDを指定する。なお、マクロ側からは、各コントロールがオブジェクトとして扱えるため、実際には各オブジェクトのイベント処理や、メソッドを記述することでプログラムを作ることができる。

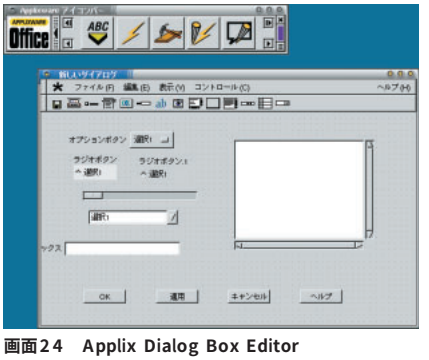


画面23　Applix Macro Editor
簡単なマクロを作成するためには、Applix Builderとは別個にマクロエディタが用意されている。これは、ステップ実行などの簡単なデバッガ機能を持ち、編集、コンパイル、デバッグが行える。

ユーティリティその他

このほかには、環境設定用のツールやディレクトリ一覧表示のツールなどが用意されている。また、Applixwareのヘルプファイルは HTML形式だが、専用のヘルププログラムが用意され、それを使ってヘルプファイルを表示している。

Applixwareは、最近登場したLinux用オフィススイートに比べて、各ソフトの機能が充実している。またカスタマイズが容易で、気に入らない部分があるなら自分でなんとかすることもできるため、使い込んでいく価値はあるかもしれない。



画面24　Applix Dialog Box Editor
SHELFで使うダイアログは、専用のダイアログエディタで作る。これは、GUI形式で部品を配置するだけで、ダイアログボックスを作成できるツールである。

隠喩としてのコンピュータ

携帯電話に嫁入りしたJava

文：豊福 剛
Text：Tsuyoshi Toyofuku

ブラウザ・フォンだった段階の携帯電話機には、あまり積極的な関心がわかなかったのだけど、Javaが搭載されるとなると、ちょっと話が違ってくる。かといって、ケータイJavaでゲームのプログラムを動かすようなことに興味があるわけではない。

そもそも、なぜiモード端末にJavaが搭載されるに至ったのだろうか。夏野剛『iモード・ストラテジー』（日経BP企画）によると、iモードの目指す最終的な目標は、電子マネーを扱うことにある。Java搭載は、そのための布石でもあるのだ。

たとえば、コンビニや自動販売機で小銭を払うかわりに、携帯電話機にインストールした電子マネーを使って少額決済を行うことが想定されている。NTTドコモとローソンとの提携では、店頭にあるマルチメディア端末（MMK）やPOSレジと携帯電話機を、無線インターフェイス経由で通信し、電子マネーのやり取りを実現することが視野に入っているらしい。

そこでやり取りされる電子マネーが、将来的には「ローソン・マネー」になるのか「ドコモ・マネー」になるのか、そのあたりの事情はよくわからないのだが、いきなり本格的な電子マネーを導入するというよりも、むしろ航空会社のマイレージ・ポイントのような「電子バリュー」を扱えるようにすることから始める計画のようだ。ローソンで買い物するたびに、電子的な「ポイント」が発行され、それが携帯電話機のメモリに貯まっていく。ユーザーは、次回の買い物の際に、貯まったポイントに応じて割引が受けられる、というようなサービスであるらしい。

iモード対応Javaのセキュリティ

電子マネーを実現するための技術と環境が普及すればするほど、携帯電話を使った地域通貨を、草の根的に実現する可能性も同時に広がっていくことになるだろう。LETSのような地域通貨を使うためのひとつの道具として、ケータイJavaを利用してみたいのである。

そもそも、電子マネーの実現を想定しているのであれば、暗号関係のスペックがどうなっているのか、気になるところである。NTTドコモのサイトで公開されている「iモード対応Javaのスペック」（http://www.nttdocomo.co.jp/

i/java.html）を見ると、残念ながら java.security パッケージは含まれていなかった。だから、公開鍵暗号を利用するJavaアプリケーションがすぐに開発できるわけではない。とはいえ、電子マネー導入のためのJava搭載なのだとしたら、将来的にはセキュリティ関係のAPIが追加されることを期待している。

iモードJavaのスペックを見ていて興味深かったのは、Java搭載端末のアーキテクチャにはScratchPadと呼ばれるデータストレージ、つまりJavaアプリケーションが利用できる補助記憶領域が含まれている点だ。iモード対応Javaアプリケーションは、ブラウザ上で動くアプレットではなく、ストレージに対して入出力が可能なアプリケーションなのだ。

ストレージがあって、Javaアプリケーションがそれに対して入出力できるとなると、当然セキュリティが問題になるわけだが、ScratchPadは個々のJavaアプリケーションごとに排他的に使用する設計になっている。あるJavaアプリケーションが使用するScratchPadの領域に、別のJavaアプリケーションからアクセスすることはできないというわけだ。さらに、Javaアプリケーションから通信可能な相手はダウンロード元のサーバに制限されるという、アプレットと共通する制限も課せられている。

ScratchPadは、公開鍵暗号で使う秘密鍵を保存しておくのに都合がいいと思う。ただし、他のJavaアプリケーションからはアクセスできないとはいえ、秘密鍵が漏洩してしまう危険性がないわけではない。秘密鍵を扱うJavaアプリケーションそのものに、秘密鍵を読み取り、その内容をネットワーク送信するコードが組み込まれていたら、秘密鍵は漏洩してしまう。それだけでなく、秘密鍵を勝手に書き換えてしまうことも可能だろう。

このような悪意あるコードが含まれていないことを保証するためには、プログラムのソースコードが公開されている必要がある。そして、ソースコードのレベルで安全を確認するだけでなく、使用するバイナリコードが、まさしくそのソースコードをコンパイルして生成されたものであり、それ以外のコードが含まれていないことを、確認する必要がある。この2つの条件がクリアできるのであれば、秘密鍵などの重要な個人データを扱うJavaアプリケーションを信頼することができるはずだ。

無線インターフェイスが欲しい

ケータイJavaをLETSの財布として利用するためには、個人データの扱いが保証されているだけでは十分ではない。ネットワーク通信を実行できる相手がJavaアプリケーションのダウンロード元のサーバだけというのは、かなり大きな制約だ。

やはり、ケータイJava間でLETSのデータを直接やり取りできるのが望ましい。端末に無線インターフェイスが搭載されるのが、半年先なのか1年先なのかはわからないが、ケータイJavaをLETSの財布として使うためには、無線インターフェイスの搭載が大きな鍵になりそうだ。無線インターフェイスの候補としては、IrDA、Bluetooth、非接触ICカードの3つが候補にあがっている。どれが携帯電話機に標準採用されたとしても、これらの無線インターフェイスを利用するJavaアプリケーションを原則的に誰でも開発できるような、透明な仕様であってほしい。

ここまで書いたところで、F503iとP503iの発売開始のニュースが流れてきた。P503iにはIrDAが搭載されていて、P503i間での個人情報メモ、自作曲、スクリーンイラスト、ブックマークなどのデータの送受信ができるらしい。iモードJavaのスペックでは、メーカーごとの独自拡張が許されているのだが、P503iのJavaでIrDAを使えるかどうかは不明だ。

それにしても、ケータイJavaにおいては、“Write Once, Run Anywhere”を期待するのは意味がないことなのだろう。携帯電話機のメーカーにとっては、IrDAを削る分だけ端末のコストを下げたほうが良いという判断もありえるわけで、ケータイJavaの最初の一歩としては、GUIの共通化とSSLの標準サポートが実現されたことだけでも、とりあえずはよしとすべきなのかもしれない。

コンピュータが電話になればいい、その逆ではない

『iモード・ストラテジー』で夏野剛は、ビル・ゲイツが提唱するウォレットPC（手のひらに収まるコンピュータに、外出する際に持っていくものがすべて入る）とiモードの目指す最終ゴールは同じだとしたうえで、ウォレットPCはパソコンを小型化するアプローチであり、機能を

削ぎ落とすものであるのに対して、iモードは携帯電話機に機能をアドオンするアプローチだと述べている。

「削ぎ落とすアプローチにはどうしても後退する印象があるが、アドオンのアプローチは前向きの進化である。技術者の開発意欲を刺激することで、開発スピードはずっと上がるだろう」

たしかに、そういう側面はあるのだろう。でも、たとえばPalmと携帯電話を比べてみても、同じことが成り立つだろうか。わたしにはPalmもアドオンのアプローチであるように思える。そして、W-CDMA対応の次世代携帯端末の試作品を見ていると、PalmのようなPDAとの違いが、ますます曖昧なものになってきている気がする。

コンピュータの発想は、コンピュータとコンピュータが通信するためにインフラがある、コンピュータが中心であり、インフラはコンピュータが利用する手段にすぎないと考える。これに対して、iモードの発想は、あくまでインフラが中心なのであり、端末はインフラを利用させるための手段にすぎないと考えているのである。

このような発想の逆転関係は、iモードに限ったことではなく、携帯電話の本質なのであり、それは電話の本質なのだ。携帯電話に音楽やビデオクリップをダウンロードさせようとする発想は、音楽を聴きたい、ビデオクリップを見たい、という欲望から発想されたのではない。高速通信が実現できるインフラを利用させるための、そして、そこからお金を巻き上げるための方便にすぎない。

第3世代移動体通信ということで、W-CDMAやcdma2000が導入されるわけだが、そのためのインフラ整備にかかる莫大な投資を考えると、ほんとにそれだけの広帯域が移動体通信に必要なのだろうか疑問なのである。移動体通信よりも、むしろ固定電話や専用線の料金を圧倒的に下げるほうが、優先順位としては高いはずではないのか。

インターネットの常時接続がほんとうに普及すれば、そもそも固定電話は不要になるはずだ。いまの電話機は、マイクとスピーカのついたコンピュータで置き換えられるはずだからだ。音声通信のパラダイムからデータ通信のパラダイムに転換するということは、従来の音声通信がデータ通信に飲み込まれてしまうことを意味するのだ。そのとき、電話回線にコンピュータが従属するのではなく、コンピュータのネットワークに電話が包含されなければならない。

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。

ハッキングの自由、コンピュータの自由

文：安田幸弘
Text : Yukihiro Yasuda

AランチとBランチの自由

自由は人間の基本的な権利だなんて言うけれど、そもそも人間ってやつは自由が嫌いなんだという説がある。

たとえば、レストランでメニューを選ぶのがめんどくさいという人は少なくない。フランス料理屋の意味不明のメニューなんてのは、めんどくさいとかいうレベルの話とは次元が違うのでそれはさておき、ファミレスや町の食堂、あるいは飲み屋のレベルでも「今日は何を食べようか」なんて結構真剣に悩んだりする。そんな人が多いから、レストランにはセットメニューだの日替わりランチなんかが用意されているわけだ。さらに定食を選ぶのもめんどくさいという不精者は、同席の友達に「同じモノ注文しといて」で済ませたりする。

AランチとBランチの選択の自由なんて、自由の本質じゃないかもしれない。だけど毎日の生活だって、ほとんどの場合、「規則正しい生活」という名前で、決まったことを決まった通りにやり過ごすことが善しとされる。毎日、違ったことを考えて違ったことをするのがめんどくさいのである。流行のファッションだってそうだ。大多数の選択に自分を合わせることで、何だかウレシイような気がするとしたら、やっぱりこれは一種の自由の放棄といってもいいかもしれない。もっと高尚な（と思われている）思想信条の自由にしたって、○○主義と××主義の選択と、AランチとBランチの選択の自由には、そんなに大きな違いがあるとは思えない。そもそも、自分の思想信条を何かの「主義」で縛り付けるのが思想信条なんだしさ。

要するに、元来人間は自分で決め、その責任を負わなきゃいけないようなめんどくさいことは好まないように生まれついているわけだ。もちろん、本当に一挙手一投足の自由を完全に奪われたら、誰でも正気を保っていられないだろうから、そこそこの自由は欠かせない。人間は自由が嫌いだというのは極論だとしても、自由なんてものは四六時中担いで歩くには少々重たいのは確かである。

GNUとオープンソース

コンピュータの世界も、自由は必ずしも歓迎されるとは限らない。

計算機資源への無限のアクセス、無限の計算の自由を求めて日夜ひたすらハッキングにいそしむごく少数のハッカーが存在する一方で、「結果として仕事ができればいいのさ」という膨大な数の平凡なパソコンユーザーの大群がある。

ハッカーという人種は、もし、自分が使いたいコンピュータを使う自由を制限されれば、あらゆる手段でその制限に抵抗する。その昔、彼らは目的の計算資源にアクセスしようとして、時としていわゆる「不正アクセス」に類する行為も辞さなかった。そのおかげで、最近は「不正侵入者＝ハッカー」と思われるようになってしまったのだけれど、パーミッションをこじ開けることはハッカーの本質じゃない。別に権限外アクセスを弁護するつもりはないけれど、ハッカーがパーミッションにチャレンジするのは、もともと目の前に使えるリソースがあるのに、管理者が規則だの契約だの、あれこれ口実をつけてリソースの利用を制限することへの抗議みたいなものなんだろう。

もちろん、使えるはずのリソースに対する制限は、何もロジカルなパーミッションとは限らない。ロジカルなパーミッションなら、解除の方法を考えることもできる。しかし、法的・制度的なパーミッション、つまり著作権やライセンス、特許などによる制限は、そうはいかない。GNUのGPLは、そうした法的・制度的なパーミッションを使った自由なプログラミングへの制限に対する異議の申立てとしてスタートしたわけだ。

ハッカーにとって、フリーソフトウェアは、ソースコードの無限の自由のための運動といってもいい。ソースは必ず配布されなければならない、ということは、ソースを公開する権利の裏返しでもある。ソースで会話をするハッカーたちにとっ

て、ソースを公開する権利は言論の自由みたいなもの。ライセンスや著作権で窒息させられそうになっていたハッカーたちが、プログラミングの自由を求めてGNUの旗の下に集まってきたのは当然かもしれない。

しかしGNUは、ある意味でハッキングの自由を利用者に押し付けたのかもしれない。GPLの厳格な規定があったからこそ、現在のGNUやオープンソースが育ったということは誰も異論はないはず。だけどGPLの規定は、一般の企業がフリーソフトウェアを気軽に自社製品に組み込もうとすると、相当、悩むことになる。ある意味では、法的・制度的なパーミッションに対する逆パーミッションなんだから、悩むのは当然かもしれない。そこで登場したのが、パーミッションそのものを緩和して、もっと気軽にソースを利用できるようにしたオープンソースライセンスってやつだ。

個人的には、GNUの考え方に共感するけれど、どっちがいいとか悪いとかという話ではないのだろう。自由という意味では、オープンソースはGNUが唱えるような攻撃的な自由ではなく、もっと改良主義的、市民的な自由の発想が基本にあるんだと思う。

自由の重さ

ところで、こうした計算の自由、プログラミングの自由をめぐって先輩ハッカーたちが闘ってきたわけだけど、パソコンユーザーの大多数を占めるごく平凡なパソコンユーザーにとっては、そんな自由なんて別世界の話だというのが現実というものだ。

普通のパソコンユーザーは、GPLだろうがオープンソースだろうが、コンピュータを使う自由なんて、どうでもいいことで、もし彼らがコンピュータを使う自由を制限されれば、喜んでコンピュータそのものを放棄するかもしれない。

まあ、アプリケーションの操作を覚えるのがせいいっぱいで、プログラミングなんて考えたこともないというユーザーにとっては、ソースコードなんて何の使い道もない情報なのだから、あたりまえと言えばあたりまえかもしれない。

しかし、である。インターネットやデジタル放送などの普及で、コンピュータが生活の中に入り込み、普通の人々の知的な活動の道具になったいま、ごく普通の一般市民だって、単に「ソースコードなんて関係ないし」というだけでは済まなくなっているんじゃないかと思う。

コンピュータの自由だって、ハッキングの自由ばかりじゃなく、タダでコンピュータを使う自由、タダでソフトを使う自由、勝手に情報をコピーする自由なんかがあったっていいはずだ。もしかすると本当は、タダでコンピュータを使う権利、タダでソフトを使う権利、勝手に情報をコピーする権利なんてものがあるのかもしれない。GPLだって、最初は「いったい何をとんでもないことを言い出すんだ」と思われていたそうだけど、今では「そういう考え方もあるよね」とそれなりに認められているんだから。

問題があるとすれば、コンピュータユーザーの大部分が、メーカーの言うなりのライセンスを受け入れ、言うなりにソフトを買って、言うなりにバージョンアップを重ねていること、そして、そのことに疑問を感じる余地もないほどあたりまえになっていることなんじゃないだろうか。

自由の押し付けはたまらないけど、みんなが最初から放棄したのでは話にならない。「ソースは隠すもの」、「ソフトは高いもの」という常識をGNUがひっくりかえしたように、常識化しているパソコンをめぐるいろんな制限や制約をひっくり返せるかどうかは、そのときにユーザーが背負い込まなければいけない自由の重さとのバランスなんだろう。

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

1・9 米証券取引委員会、VA LinuxのIPOについて調査
1・5 CES開催
1・4 Linux 2.4カーネルリリース
12・15 3dfx解散を発表
12・12 Palm、次期OSをデモ（PalmSource）
11・17 ICANN新TLDを承認

こういう仕事をしていると、年末ぐらいから、5月の連休ぐらいまでなかなか落ち着けない日々が続く。まず、年末の休みのために仕事が前倒しになり、さらに正月休みが終わると、海外取材などが続くためである。これが落ち着いたら、今度は5月の連休のためにまた仕事が前倒し、さらに4月あたりには、各媒体のリニューアルなどで仕事が増えたりする（減るときもあるのだが……）。というわけで、クリスマスも正月もないままに、21世紀に突入し、3月で年度が終わるというのに、なかなかひと息つけない日々が続いてストレスもたまり気味。年度末でもあり、ニュースも停滞気味である。

今年の年末は静かだったか？

一昨年の年末は、インテルとAMDの最高クロック競争が最も激しい状況で、年末ギリギリにインテルがPentiumIII/800MHzを発表して、1999年最高速の座を取ったと思ったら、去年3月には、もう1GHzのCPUをAMDが発表してしまって、GHz時代へ突入した。しかし、それからは、Pentium4の1.5GHzで落ち着いて、21世紀を迎えたわけだ。

Microsoftは訴訟があって以来、以前よりはニュースが少なくなった。まあ、幹部などが発言に気をつけるようになったためと、Windows 2000という大物の発表が終わってしまい、.NETも発表したら、静かなものになってしまった。

しかし、世の中は止まっていたわけではないし、Microsoftもインテルも休んでいたわけではない。でも、なんかニュースがないとね。

Microsoftはこのところ、.NETに注力中だが、どうなることか。.NET対応製品のβテストなどが始まっているし、いろいろ資料も出てきている。しかし、実体が明かになるにつれて、業界には波紋も出ている。ちょっと大きな話題としては、Visual BASICの変更がある。.NET対応になったVisual BASICは、仕様が大きく変更され、業界では、VB.NETをVB.NOT（http://www.mvps.org/vb/index.html?rants/dotnot.htm）と呼んでいるのだとか（**画面**）。

Visual BASICといえば、PC用の業務ソフト作成などにはいまや必須ともいえるアプリケーションで、数多くのデベロッパーがこれを使っている。また、個人用としても人気が高く、これを使うプログラマーは多い。VBは、簡単にアプリケーションができるわりには、ちゃんとWindowsの機能を使うことができ、Visual C++が、商用パッケージアプリケーション開発用にかなり傾いているのに対して、手軽にプログラムを作る方法として定着している。これが大きく変更され、従来のソフトの移植が難しくなっているのだとか。

インテルがマテル社と提携して、子供向けのPC周辺装置系オモチャ（IntelPlay）を出しているのに対抗してかどうかは知らないが、Microsoftは、ブロック玩具の有名どころLegoと提携。今年Microsoftが出すゲームマシンX-Boxでゲームなどを出すらしい。Legoも最近は、Mindstormなんかで PC寄り路線を見せているので、まあ、両社の接近はありえない組み合わせでもない。だが、組み込みWindowsなんて入るとちょっとイヤだな。

インテルのほうは、AMDに加え、日本のメーカーでの採用が相次いだTransmetaへの対応もしなければならない状態である。それまでインテルは、「モバイルCPUも顧客からよりパワーの大きいものを要求されている」とし

て、高クロック化路線を突っ走っていたが、TransmetaのCPUを日本のノートPCメーカーが相次いで採用すると、路線変更に出たようである。とりあえず、低消費電力を実現するモバイルCPUの開発チームを立ち上げ、IDFなどでは、「CPUだけがノートPCで大きく電力消費を行っているわけではない」などと発言していたが、結局、低消費電力CPUを開発することにしたようである。まあ、ノートPCのバッテリ寿命が伸びるのはいいことだし、筆者はつい最近まで、300MHzのPentium IIのデスクトップマシンで仕事していて、特に不便さを感じなかったので、ノートPCで700も800MHzもの速度は不要と思っている。というわけで、今年後半までには、バッテリ寿命の長いノートPCがあたりまえになるのかも。

グラフィックデバイスはメーカー淘汰へ

昨年末に、3dfx社が解散し、グラフィックスチップメーカーがまた1つ減った。昨年は、各社ともボードメーカーを買収して、自社生産ボードでの販売という方向に動いた。しかしS3は、買ったダイヤモンドマルチメディアのRIOなどが好調で、結局、グラフィックチップビジネスをVIAに売却、アプライアンス系メーカーになってしまい、名前までSonicBlueと変えてしまった。3dfxも、STB Systemsを買収して、自社ボードビジネスに乗り出したのだが、これがどうもいけなかったようだ。

3dfxの資産は、ライバルであるnVIDIAに売却されるのだとか。ある意味、いまのところnVIDIAが一人勝ちという状態。

グラフィックデバイスメーカーは、年々減ってきており、ノートPC用に特化したところもあったが、どこも撤退を余儀なくされた。

昨年は、マザーボード組み込みが流行るとは言われていたものの、それほど多くはなかったが、こうした状況もあって、今年は組み込みマザーボードが増えそうである。

いまあるほとんどのアプリケーションは、グラフィックボードに3次元機能がなくとも、全然問題ないという状態なのに、各社ともに3Dグラフィックに走ってしまったのが問題のような気がする。アプリケーションは、ゲームしかないのだから、ゲームでの採用がデバイスの売り上げを大きく左右するし、ベンチマークによる性能差が見えやすくなってしまい、製品寿命が短くなってしまったわけだ。

しかし、これだけ淘汰されてしまうと、競争が緩くなって、価格や今後の性能なんてところが心配。最近のグラフィックスカードは、3D性能を重視するあまり、2D性能が頭打ちの感があるが、普通のアプリケーションのことをもっと考えてほしいものである。もっとも、なんでも3D表示なんて方法もあるのだけど、3D表示されるファイルマネージャーとか煩わしいって気もするよね（そういえば、SGIのシステムには、そういうサンプルがあったような……）。

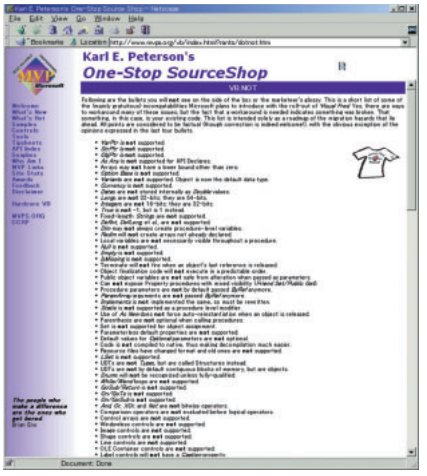
カーネルは出たが……

Linuxのほうは、待望の2.4カーネルが登場。これから春先にかけて、ディストリビューションのバージョンアップが行われるはず。まあ、気の早い人は、今月号の特集を読んで、もう入れているかもしれないが……。

今回の大きな目玉は、USB関係ではないでしょうか。PC用の多くの周辺装置がUSB化して行く現在、やはりUSB機器が使えないと、これからツライものがあるだろう。まあ、サポートドライバの問題なんかもあるので、すぐにはなんでもというわけにもいかないが、それでもこれから除々にサポートされる製品が増えてくると思われる。実際、USBデバイスは、製品が違っても、中に入っているデバイスが一緒なんてこともあるので、どれかひとつがつながると、あとはデバイスIDを変えるだけで使えるなんてことも多いはず。

また、USBになってはいるものの、データ転送モデルとしては、シリアル接続と同じなんてものもあるわけで、これからいろいろ解析が行われるのではないか。特に人気のあるデバイスは、たとえば、RIOやVISORなんかは、もう、解析されているみたいだし、今年はちょっとしたUSBデバイスプロジェクトブームになるかも。

それに、ボードメーカーの一部がLinux対応しているので、Linux用のUSBドライバの提供もメーカーがやってくれるとよいのですが…。ネットワークカードなんかは多くのメーカーが対応しているので、こちらはちょっと脈がありそう。



“VarPtr is not supported.”“StrPtr is not supported.”など、Visual BASICが.NET対応になって互換性がなくなった点を列挙している。
http://www.mvps.org/vb/index.html?rants/dotnot.htm

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

～VA Linux Systems ジャパン事例取材に行ってきました～

2000年12月号の本コーナーにおいて、VA Linux Systems ジャパン（以下VA）の設立と、それに伴うニュース記事などをお伝えした。
実はその後、実際に稼動しているVAのマシンをこの目で見るべく、事例取材を行った（http://www.linux24.com/linux/news/today/article/article345632-000.html）。
今回はその取材裏話をお伝えしたい。

（吉川大郎）

VAの製品を導入したのは、東京農業大学で同大学世田谷キャンパス。早速お邪魔してお話を聞いた。

東京農業大学世田谷キャンパスは、小田急線経堂駅からゆっくり歩いて15分程度のところにある。「農大通り」という商店街を抜けると路地に入り、アパートが目立ってきて、学生っぽい服装の人間が三々五々歩いている。意外だったのが、結構な数の女性が農大に向って歩いていることだ。あとからわかったことなのだが、栄養科学科をはじめとして、女子の比率も高いのだそうだ。ほかにも、バイオサイエンスや醸造、造園、栄養学など、幅広い分野をカバーしているのだそうだ。農大で広報業務を担当する芳野公一氏が、「農業の名を冠した生物系総合大学」であると教えてくれた。それから、歴史好きの方にもうひとこと。農大の設立者は、あの五稜郭の榎本武揚である。

コンピューターセンター内に設置されたVAのシステム。4台の「VA Linux 2200」のほか、テープドライブやUPSなどが1つのラックに収納されている。

裏門（うろうろとしているうちに裏門から学内に入るはめになってしまった）をくぐるとキャンパスなのだが、これがまた緑が豊富ですこぶる環境がいい。いくつか都心に隣接する大学に取材に行った経験があるが、農大の世田谷キャンパスはその中でもトップクラスの環境だと思う。

私を出迎えてくれたのは、前述した東京農業大学学長室の芳野公一氏と、システム導入を行った東京農業大学 国際農業情報学部コンピュータセンター事務主任の田中晋氏、そして東芝エンジニアリングカスタマーサポート＆サービス事業部ソリューション営業部主任・内田正文氏である。

VAのシステムはこう使われている

今回VAが導入されたのは、Web関連をはじめ、入学案内や願書無料請求システム、グループウェアだという。農大のWebページ（http://www.nodai.ac.jp/）を見ると、これらの仕組みが導入されているのがわかるだろう。

農大の場合は、基幹システムはメインフレームとパソコンが半々ということだ。今回Linuxは基幹システムには導入されていない。Linuxの導入自体、農大としてはテストケースというか、「まずはWebから」といったスタンスなのだそうだ。この導入によって、フリーソフトウェアへのノウハウを蓄積していきたいということである。「大学だから、FreeBSDやLinuxの導入は結構自然なことだな」と思っていたのだが、どうやらそういうわけでもなさそうだ。

また、日刊アスキーLinuxの記事にも書いたが、今後は「シラバス」と呼ばれる講義情報をはじめさまざまな

Webサービスを検討中だという。このシラバスという代物、私は大学生だった時代があるにも関わらず初耳だったのだが、履修登録の際などに使う、講義案内の分厚いパンフレットである。

これを学内にWebで公開すれば、印刷コストの削減につながるわけだが、話はそれだけではない。農大としては、やはりLinuxありきではなくシステム全体として考えているわけで、シラバスから最終的には基幹系が処理している履修管理システムまで結びつけていきたいわけだ。だから、シラバスを単純に公開するのではなく、学生データ、教員データ、教室データそしてシラバスがすべて融合したトータルなシステムとしての設計が必要になってくる。そのあたりが課題だということだ。

もっとも、これは一般の企業にも当てはまることで、いわゆるIT産業が顧客に向かってアナウンスしているところの、「基幹データをWebに乗せていく」といった類の話ではあろう。

なんでLinux？ なんでVA？

ところで、なぜLinuxであり、VAなのだろうか？ 今回の導入では、今までの農大のコンピュータ導入パターンとは少し違ったアプローチがとられたそうだ。いままでのシステム選定は、コンピュータセンターがプランを立てて行っていたのだが、今回は農大の学長室や庶務課側からもエンドユーザーとしてのリクエストを出して進めていったのだという。

その要望とはどのようなものなのかを学長室の芳野氏にたずねたところ、たとえばデータの管理機構に関してエンドユーザー側からもアクセスできるようにしてほしい（つまりは学内イントラネット的な仕組み）といったもののほか、フリーソフトウェアを使えるようにしてほしいといったようなものだったという。

エンドユーザー側からのデータ管理はともかく、フリーソフトウェアを使える、という面で見れば、Linuxは最適というわけだ。もっとも、農大としては「初モノ」のLinuxを導入するのはある程度勇気のいる話ではある。そこで出てきたのが、VAだったわけだ。

VAに関しては、東芝エンジニアリングによるサポートの確約がとれたことと、米国での実績が大きくモノをいったという。もともとフリーソフトウェアを使いたいという要望があったわけだから、OSDNの運営元でもあり、オープンソース企業として名高い同社のブランドもプラスに働いた。

●

大学関連のLinuxシステム導入事例というと、理工系の科学計算システムなど専門分野で使う「計算機」としての事例が多かったのだが（こちらの話も興味深く、別途記事作成中である）、今回は一般とも関係の深いWeb関連やいわゆるイントラネット系の事例で、私としても新鮮だった。

農大のケースで注目すべきは、やはりその採用理由だろう。オープンソース活動への貢献（投資）がブランド力に結びつき、それが結果として採用理由のひとつになったという事実である。

つまりVAの例でいえば、旧来の見方でいうと利益とはあまり関係ないOSDNなどへの投資＝コミュニティへの利益還元がブランド力アップにつながり、採用への一助になったという、まことに美しい流れを当てはめることのできるケースなわけだ。もっとも、東芝エンジニアリングのサポート力や、農大のシステム要件に製品が合致したという背景はあるのだが。

もういいかげん「オープンソースビジネス」という言葉も手垢が付いてきたわけだが、いといろなカンファレンスの基調講演で語られてきたことが現実に起こっているという点で、農大取材は印象深いものとなったのである。

東京農業大学世田谷キャンパス。敷地内には巨木が多く、学生が三々五々休憩したり実習を行ったりしていた。

今回お話を聞いた、東京農業大学学長室の芳野公一氏（左）と東京農業大学 国際農業情報学部 コンピュータセンター 事務主任　田中晋氏（右）。

正しきタコへの道

初級Linuxer養成講座

第18回　マルチタスク機能を使ってみる

Linuxとは何かということについて語られるとき、かならず話題に上るのが「マルチタスク」や「マルチプロセス」という言葉である。GUIの上でたくさんのアプリケーションを立ち上げて使うのがあたりまえになった今では、あまりピンとこない人もいるかもしれないが、コマンドラインでこの機能をうまく使えるようになれば、単純に便利なだけでなく、GUIの環境でアプリケーションを使ううえでのコツも身につくはずだ。

文：竹田善太郎
Text : Zentaro Takeda

この10年間ほど、冬になるとほぼ毎週のようにスキーに出かけている。スキーといっても、ゲレンデでする**普通のスキー**ではなく、**クロスカントリースキー**である。「ノルディック複合競技」などでの日本人選手の活躍で、最近は知らない人も少なくなったとは思うが、あのような競技以外にも、**雪道散歩**として楽しむクロスカントリースキーもあるのだ。

簡単に言ってしまえば、**雪道を歩いたり滑ったり**するだけの遊びなのだが、これがなかなか楽しい。**周囲の自然を楽しむ**といったあたりまえの楽しみ方以外にも、**山道を攻略する楽しみ**とでもいうべきものがあるのだ。クロスカントリースキーの板は、ゲレンデスキーの板に比べて細長く、かかとの部分が固定されておらず、さらに金属のエッジがついていない。このためとても不安定なので、**下り坂**は相当に神経を使わないとコントロールを失ってしまう。しかも、競技用のコースと違って、狭い登山道のような場所を滑ることが多いので、**ラインどり**を間違えるとカーブを曲がりきれなくなって**ヤブに突っ込む**ことになる。初心者ならずとも**スリルの連続**ということになるのだ。

自動車の運転でもそうなのだが、広い道でスピードを出すのなら、サルにだってできる。よく、「高速道路で×百キロ出した」と得意げに話す人がいるが、それは単に自分のクルマと「無謀さ」を自慢しているだけであって、腕がいいわけでもなんでもない。しかし、たとえば曲がりくねった細い山道で、センターラインを踏み越えずに、安全に安定した速度で走ろうとすると、これは相当なテクニックがいる。自分の想定したラインから1センチと外さずにコントロールしようとすれば、まさにプロレーサー並みの腕がいるわけだ。

もちろんゲレンデスキーでもそうなのだが、クロスカントリースキーで狙ったラインどおりにコーナリングを決められると、とても気分がいい。しかし、現実は厳しくて、雪の状態が悪かったり、だれかがしりもちをついたあとの**大穴**に足をとられたりする。そして、コースを外れてしまったり、転んでしまったりすることが多い。10年間続けていてもそうなので、自分の運動神経がよっぽど悪いのかもしれないが、それだけ**奥が深い**遊びなのだ、ということにしておきたい。

Linuxを使うということも、ある意味で山道を攻略することにも似ている気がする。使い方が決まっているアプリケーションをただインストールするだけではなく、自分の目的に応じてさまざまなツール類を探し出して、それらを細心の注意をはらいながら組み合わせる。簡単にうまくいくことは少なくて、何度も失敗しながら試行錯誤しなければならない。

そういう過程が苦手な人や**時間の無駄**だと思うような人は、おそらくLinuxなんか使わずにWindowsやMacintoshを使い続けるのだろうが、お仕着せでない**自分専用のパソコンを使う**ということは、買ってきたりダウンロードしてきたりしたアプリケーションを無闇に使うこととは違うような気がする。

Linuxを好きで使っている人の多くは、単に**無料だから**という理由だけではなく、自分の目的に合わせてシステムを**あつらえていく過程**を楽しめるという理由もあるのではないかと思っている。そして、Linuxやその周囲にあるプログラム類の**オープンソース**という性質は、そのような使い方をする際の障害となる**ライセ**

ンスの縛り**から解放してくれるだろう。

マルチタスクとバックグラウンド実行

本連載が始まって間もなくのころ、「Linuxの中身をのぞいてみよう」というような話をしている。Linuxは**マルチタスク**のOSなので、内部では複数の**プロセス**と呼ばれるものが同時に動いていて、それを**psコマンド**や**topコマンド**で見ることができる、というような話をしたはずだ。

試しに今一度、psコマンドでどんなプロセスが動いているのか眺めてみよう。psコマンドで、システム上のすべてのプロセスを表示させるには、「ax」オプションをつければよい。

```
% ps ax
```

すると、**画面1**のように、大量のプロセスが表示されるだろう。

これらのプロセスのほとんどは、ハードウェアを動かすため使われている「デバイスドライバ」の一部や、ネットワークのさまざまな機能を利用できるようにするためのプログラムで、いわゆる「アプリケーション」とは異なるものだ。ユーザーがコマンドなどを直接に入力することはなく、エラーが発生するなどの特別な事態が起こらないかぎり、画面上にメッセージを表示することもない。誰の目にも触れないように、静かに動き続けているのである。

このように、直接ユーザーとの間で情報のやりとりをしないようなプログラムの実行状態のことを、**バックグラウンド**と呼んでいる。バックグラウンドを直訳すれば**背景**というような意味になるが、**舞台裏**や**裏方**のことだと考えたほうがわかりやすい。ユーザーの目に直接触れることはなくても、たとえばインターネット関連のアプリケーションを使ったり、日本語の入力を行ったりすると、それぞれのアプリケーションがこれらのバックグラウンドで実行されているプロセスを利用することになるのだ。

Linuxマシンの使用目的が多くなればなるほど、それに応じてバックグラウンドで動いているプロセスの数も増える。逆に、PDAなどの単目的で小さなデバイス上で動くLinuxでは、バックグラウンドのプロセスの数はごく少なくなる（多くても十数個くらい）。

「タスク」と「プロセス」と「ジョブ」

ところで、ここまでの段階ですでに**タスク**（マルチタスク）と**プロセス**という言葉が出てきて、多少混乱している読者もいるかもしれない。タスクとプロセスという言葉の意味は、どのような場面で使われるかによって変わってくる。

また、これらの言葉は、OSやプログラミング環境の違い、いわゆる**文化の違い**によって意味が微妙に変わってくる。このため、あまり断言してしまうと、その筋から**嘘を書くな！**というお叱りを受けてしまいそうなのだが、ここでは**タスク**とは、コンピュータのプログラムを使って処理しようとする**仕事**のこと、**プロセス**とはその仕事を遂行するために**実行中のプログラム**のことだと考えておいてほしい。

すなわち「マルチタスク」とは、**同時に複数の仕事ができる能力**のことで、「マルチプロセス」といった場合には、**同時に複数のプログラムを実行できる能力**というような意味になる。結局これは、まったく同じことを「目的」と「手段」の立場から言い換えているだけなので、「タスク」も「プロセス」も同じような意味であると考えておいてよい。

Linuxの世界では、さらに**ジョブ**という言葉も使われる。広義のコンピュータ用語では、「ジョブ」はさきほどの「タスク」とほぼ同じような意味で、コンピュータを使って処理する一単位の「仕事」のことを表す。したがって、「タスク」、「プロセス」、「ジョブ」はすべて、突き詰めれば同じ意味になる。しかし、Linuxの世界では多少異なっていて、「ジョブ」とは**実行中のコマンドライン**のことを指すのである。

「実行中のコマンドライン」とはすなわち「プロセス」の

画面1　psコマンドで「プロセス」の一覧を見てみる
「ax」オプションをつけてpsコマンドを実行してみると、このように大量の「プロセス」が動いていることがわかる。そのほとんどは、ユーザーとは直接関係ないところで働いているものだ。この画面ではWebサーバ関連のプロセス（httpd）が多い。

```
kterm
  350 ?        SW     0:00 [honyakuserver]
  370 ?        SL     5:35 /usr/local/bin/ntpd
  379 tty2     SW     0:00 [mingetty]
  380 tty3     SW     0:00 [mingetty]
  381 tty4     SW     0:00 [mingetty]
  382 tty5     SW     0:00 [mingetty]
  383 tty6     SW     0:00 [mingetty]
  474 ?        SL     4:56 ntpd
  480 tty1     SW     0:00 [mingetty]
24052 ?        S      0:00 httpd
24053 ?        S      0:00 httpd
24054 ?        S      0:00 httpd
24055 ?        S      0:00 httpd
24056 ?        S      0:00 httpd
24057 ?        S      0:00 httpd
24058 ?        S      0:00 httpd
24059 ?        S      0:00 httpd
24060 ?        S      0:00 httpd
24061 ?        S      0:00 httpd
25736 ?        S      0:00 in.telnetd: zen06
25737 pts/0    S      0:00 login -- zen-t
25738 pts/0    S      0:00 -bash
25753 pts/0    R      0:00 ps ax
[zen-t@zen02 zen-t]$
```

ことではないか？と思うかもしれない。ある意味でこれは正しい。しかし、Linuxのコマンドラインでは、1つのコマンドラインで複数のコマンドを組み合わせて実行することができる。

たとえば、

```
$ cat file1.txt file2.txt | grep "シロクマ" | less
```

といった具合である。

上の例では、cat、grep、lessの3つのコマンドが実行され、それぞれ1つずつ、合計3つのプロセスが動くことになる。しかし、コマンドラインとしてはあくまで1行なので、この場合、「cat file1.txt…」から「…｜less」までのすべてをひっくるめて、1個の「ジョブ」と考えるのである。catコマンド、grepコマンド、lessコマンドそれぞれが行っている処理は、全体の仕事の一部分だけで、それぞれ単独ではあまり意味はない。「file1.txtとfile2.txtの中から“シロクマ”という単語を含む行を、スクロールして見られるように表示する」というひとかたまりの仕事全体を表わすのが「ジョブ」なのである。

ジョブの開始と終了

「ジョブの開始」と大げさに言っても、これはコマンド行を実行すると言い換えてしまってよい。シェルのコマンドラインでコマンド文字列を入力し、Enterキーを押せば「ジョブを開始」したことになる。同様に「ジョブの終了」は、コマンドの実行が終わって、コマンドラインのプロンプトが再び表示される状態になることで、通常はユーザーが「ジョブの終了操作」をするようなことはない。

ただし、普通にコマンドを実行することを「ジョブを実行する」とは言わないことが多い。また、シェルに組み込まれているコマンド、すなわち内部コマンドを実行するような場合（たとえばechoコマンドなど）も、シェル以外のプロセスを動かすわけではないので、厳密には「ジョブを実行した」とはいえない。

だが、ここではそのような細かいことは気にしないでおこう。情報処理学科の講義をしているつもりはないので。

ジョブの一時停止と中断

実行中のジョブ、すなわち実行中のコマンド（プログラム）は、一時停止することができる。その方法は、コマンドによって異なるのだが、たいていはCtrl＋Zキーを押せば、実行を一時停止してシェルに戻ることができ

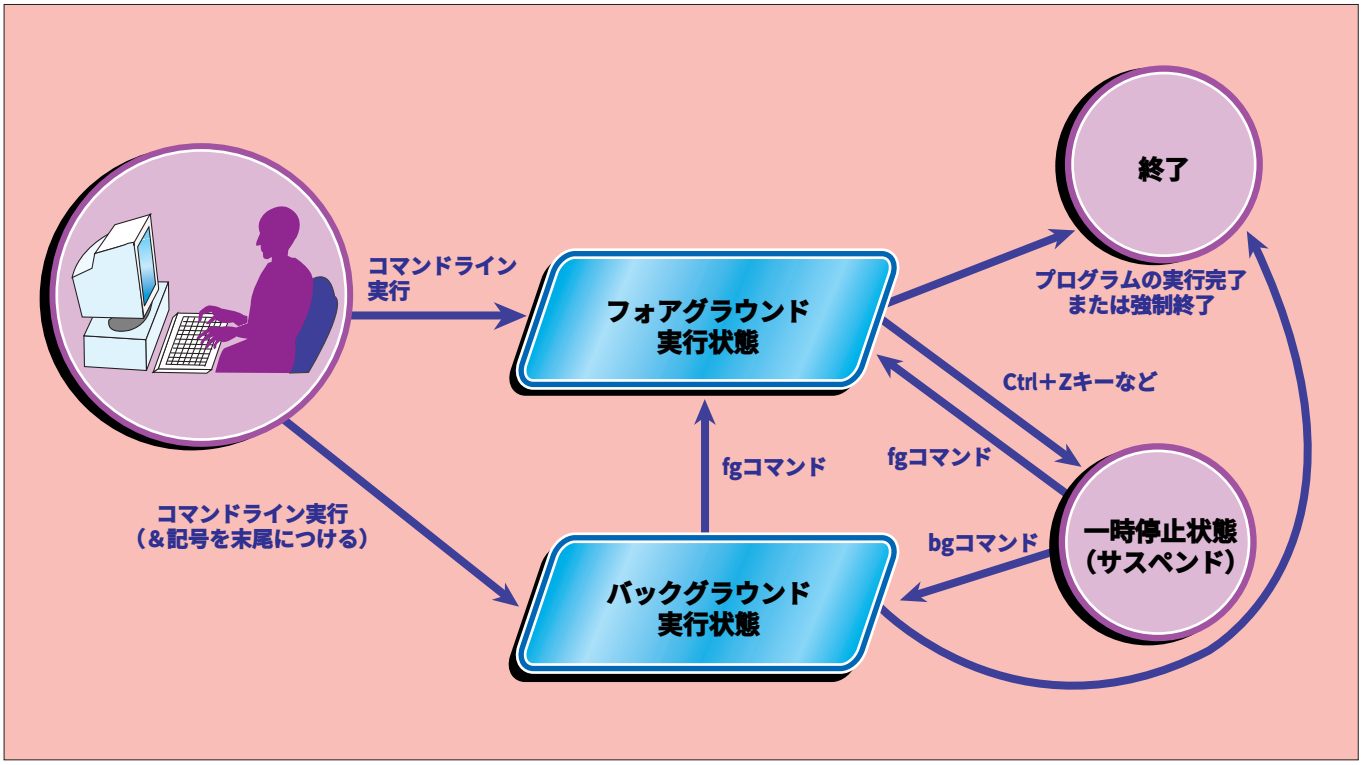


図　ジョブの状態
Linuxのシェルから起動した「ジョブ」は、Ctrl＋Z、Ctrl＋Cなどのキー操作、fgコマンド、bgコマンドなどを使って、一時停止、フォアグラウンド、バックグラウンド、終了のそれぞれの状態の間を行き来させることができる。

る。Ctrl＋Zで止まらないEmacsやMuleなどは、Ctrl＋X Zで一時停止することができる。

一時停止をすると、次のようなメッセージが表示され、コマンドプロンプトの状態になる。

```
[1]+  Stopped  [コマンドラインの内容]
$
```

「Stopped」という文字の前には大括弧で囲まれた数字が、後ろには一時停止したコマンドライン（すなわちジョブ）の内容が表示される。この数字は**ジョブ番号**と呼ばれるもので、一時停止したジョブを再開するなど、さまざまな操作をするときに必要となるものだ。ジョブを一時停止することを、**サスペンド**と呼ぶことも多い。

Ctrl＋Zでジョブが停止されている状態では、通常どおりに別のコマンドを実行することができる。別のアプリケーションを立ち上げたり、そのアプリケーションをまた「一時停止」することもできる。

あるジョブがサスペンドされている状態で、新たに別のジョブをサスペンドすると、別の「ジョブ番号」が割り当てられる。

```
[2]+Stopped tar czf /tmp/backup.
taz home
$
```

上の例では、新たに停止したジョブの番号が、「2」であることを示している。

停止したジョブを再開するには、fgというコマンドを使う。再開したいジョブ番号の前に「%」をつけて、fgコマンドの引数として与えるのだ。

```
$ fg %1
```

あるいは、fgコマンドを省略して、ジョブ番号だけ入力しても同じことである。

```
$ %1
```

いずれの場合も、停止されていたジョブの実行がただちに再開される。そのジョブの実行が完了するか、あるいは再び一時停止するまで、コマンドラインは使えない状態になる（**画面2**）。

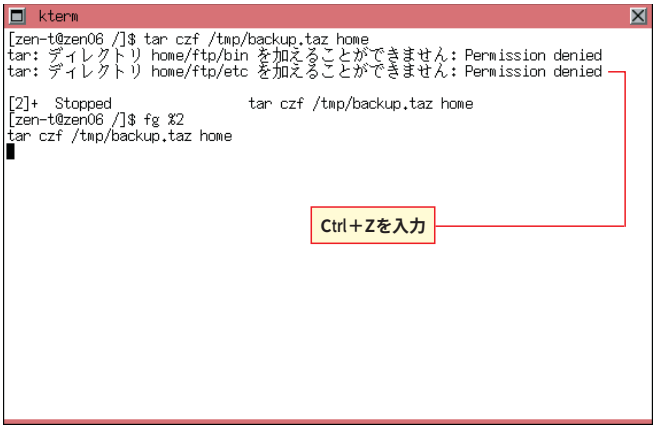
ところで、「fg」コマンドの意味だが、これは**ForeGround**という単語を省略したものである。foregroundとはbackgroundの反対語で、「前景」とか「表面」というような意味になるが、これは後ほど説明する**bgコマンド**と対になるわけである。

ジョブを「一時停止」ではなく**中断**したい、すなわち実行を取りやめたいと思った場合には、これもコマンドによって使える場合と使えない場合があるが、Ctrl＋Cキーを押してみる（**画面3**）。たいていの場合、なにごともなかったかのようにコマンドの実行は中断され、コマンドプロンプトに戻るはずだ。

ジョブを「裏」で動かす

あるジョブを実行すると、多くの場合、コンソールはそのジョブ（起動したアプリケーション）に**占領**された状態になり、ユーザーからのキー入力は、すべてそのアプリケーションが処理する状態になる。

もしアプリケーションがユーザーからのキー入力を受け付けないようなもの、たとえばtarコマンドのような場合、ユーザーはジョブが終了するまでじっ



画面2　ジョブの一時停止と再開
コマンドラインを実行してから、Ctrl＋Zキーを押すと、実行中のジョブを一時停止できる。一時停止したときには、そのジョブの「ジョブ番号」とコマンドラインの内容が表示される。このジョブ番号をfgコマンドに指定すれば、そのジョブを「再開」できる。

画面3　ジョブの中断（強制終了）
フォアグラウンドで実行中のジョブを中断（強制終了）させたい場合には、Ctrl＋Cキーを押してみる。ただし、この方法がいつも使えるとは限らないことに注意。

と待っているか、あるいはCtrl＋Zキーでジョブを一時停止したり、Ctrl＋Cキーでジョブを中断しなければならない。

短時間で終了するジョブなら待っていても問題ないのだが、たとえば、たくさんのファイルをtarコマンドでバックアップしようとするときなど、数分間から数十分間も待たされることになる。

このようなときには、ジョブを**バックグラウンド**で動かすと、その間もコマンドラインで別の作業を続けることができて便利である。

ジョブをバックグラウンドで動かすには、コマンドラインの末尾に&記号をつけてやる。

```
$ tar czf /tmp/backup.taz home &
```

すると、次のようにジョブ番号とプロセス番号が表示される。

```
[1] 426
```

括弧に入っているのが「ジョブ番号」、その右側にあるのが「プロセス番号」である。プロセス番号は、psコマンドなどを使ってプロセス一覧を調べたときに表示される「PID」（プロセスID）と同じものである。

ジョブ番号などが表示されたあとには、直ちにコマンドプロンプトが表示されて、次のコマンドを入力できるようになる。

一見、なにも起こらずにジョブが終了してしまったようにも見えるが、たとえばtarコマンドでのバックアップを実行しているのであれば、ディスクへのアクセスが頻繁に起こって、tarコマンドが確かに動いていることがわかるだろう。

しばらくの間、別のコマンドなどを実行していると、ある時点で次のようなメッセージが画面に表示される（**画面4**）。

```
[1]+  Done    tar czf /tmp/backup.
taz home &
```

これは、バックグラウンドで実行されていたジョブが終了したことを通知するメッセージだ。例によって、最初の括弧内の数字がジョブ番号を示している。

バックグラウンドで現在どのようなジョブが実行中なのか、あるいは現在一時停止状態になっているジョブがあるかどうかを調べる場合は、**jobsコマンド**を使う。

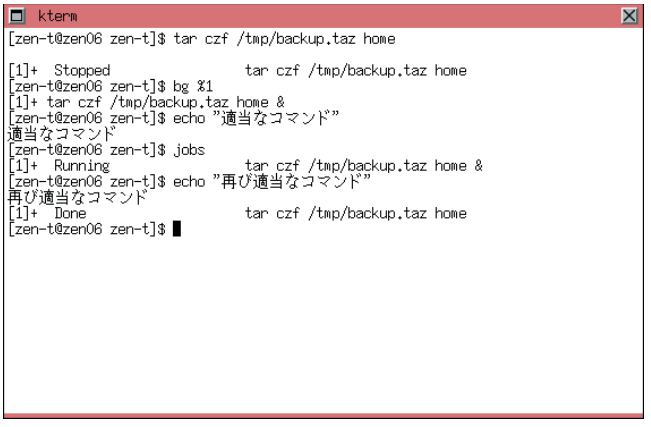
```
$ jobs
[1]- Running   tar czf /tmp/backup.
taz home &
[2]+ Stopped   vi sirokuma.txt
```

表示されるメッセージの意味については、これ以上説明する必要はないだろう。

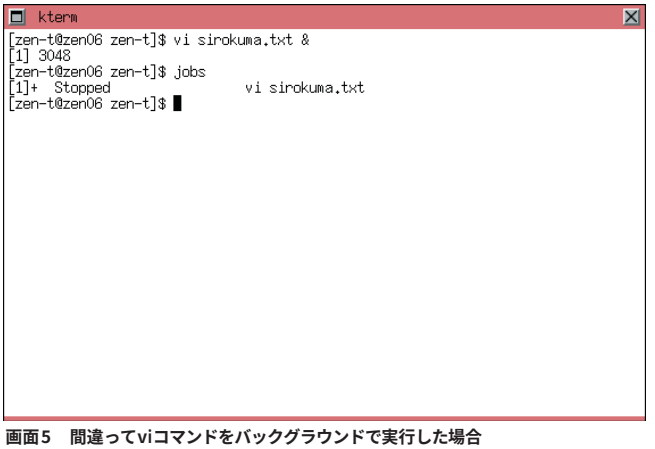
ところで、Emacsやviなどのテキストエディタを起動するときに、間違ってバックグラウンドで実行してしまうことがある。

```
$ vi sirokuma.txt &
```

すると、画面にはジョブ番号とプロセス番号が表示されるだけで、何の変化も起こらずに次のプロンプトが表示され、肝心のviによる編集ができない状態になってしまう（**画面5**）。このような場合には、次に説明する方法で、ジョブの「裏表」を入れ替えてやればよい。



画面4　バックグラウンドジョブの終了メッセージ
バックグラウンドでジョブを動かしている場合、そのジョブが処理を完了して正常に終了すると、「Done」というメッセージが表示される。同様に、ジョブの進行に何らかの支障が生じて、強制的に終了したような場合は「Terminated」、自動的に一時停止状態に移った場合には「Stopped」といったようなメッセージが表示される。



画面5　間違ってviコマンドをバックグラウンドで実行した場合
テキストエディタのように、バックグラウンドで実行しても意味のないコマンドをバックグラウンドジョブとして実行してしまうと、画面上はコマンドラインの入力状態に戻るが、起動したジョブは「一時停止」状態になってしまう。

ジョブの「裏表」を入れ替える

バックグラウンド（裏）で実行されているジョブを、フォアグラウンド（表）で実行させるようにするコマンドがある。先ほど、一時停止中のジョブを再開するときに使ったのと同じ**fgコマンド**を使えばよいのだ。

たとえば、ジョブ番号1で実行中のバックグラウンドジョブをフォアグラウンド状態にしたい場合には、次のコマンドを実行すればよい。

```
$ fg %1
```

先ほどの例でいえば、バックグラウンドで実行中のviコマンド（キー入力待ちをしている）が、フォアグラウンド状態、すなわち通常の実行状態に切り替わって、テキストファイルの編集を行えるようになる（**画面6**）。

fgコマンドを省略して、ジョブ番号だけ指定してもよい。

```
$ %1
```

逆に、たとえば長い時間がかかるtarコマンドをフォアグラウンドで実行してしまったとき、これをあとからバックグラウンド状態に変更することもできる。Ctrl＋Zキーで実行中のジョブを一時中断してから、**bgコマンド**でそのジョブをバックグラウンド状態で再開させるのだ（**画面7**）。

```
$ tar czf /tmp/backup.taz home
(Ctrl-Zを入力)
[1]+ Stopped tar czf /tmp/backup.
taz home
$ bg %1
[1]- tar czf /tmp/backup.taz home
&
```

fgコマンドと同様に、bgコマンドも省略することができる。ただし、再開したいジョブ番号の後ろに&をつける点が異なる。

```
$ %1 &
```

ジョブの「強制終了」

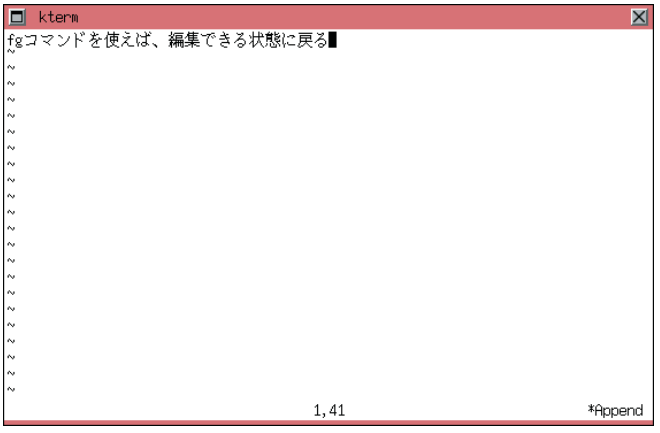
一時停止中のジョブや、バックグラウンドで実行中のジョブを、強制的に終了させることもできる。**killコマンド**を使うのだ。

使い方は簡単で、killコマンドの引数としてジョブ番号を指定すればよい（ここで、ジョブ番号の先頭の%記号を付け忘れると、同じプロセス番号を持ったプロセスを停止しようとすることになってしまうので注意）。

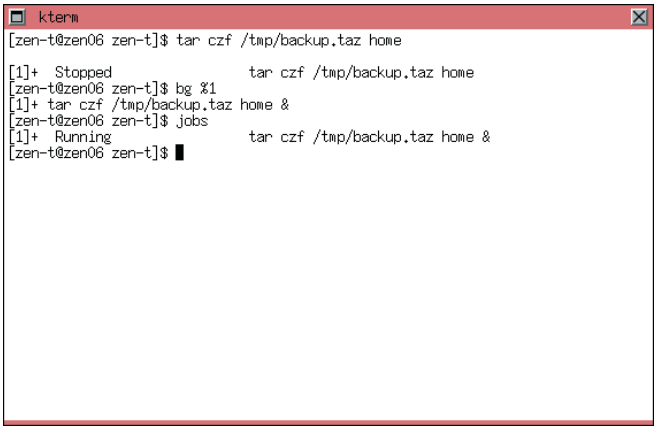
```
$ kill %1
```

ただし、当然だがこの方法はフォアグラウンドで実行中のジョブには使えない。killコマンドを入力できないからだ。前述のように、Ctrl＋Cなどで終了できることもあるが、viなどのアプリケーションの場合には、そのアプリケーション固有の終了方法（viの場合なら、コマンドモードから「quit!」を実行するなど）を使わなければならない。

Ctrl＋Zキーなどで一時中断状態になっている場合なら、killコマンドでの強制終了は可能だ。ただし、その場合は編集中のテキストファイルの中身が失われるなどの可能性もあるので、killコマンドによる強制終了には**それなりの覚悟**が必要である。



画面6　一時停止状態のviを「復活」させる
画面5のような状態からviを使えるようにするには、fgコマンドを使えばよい。viコマンドのジョブ番号が「1」なら、「fg %1」を実行すればよいわけだ。これで、何事もなかったかのように、テキストファイルの編集ができるようになる。



画面7　ジョブを「裏」にまわす
時間のかかるジョブをフォアグラウンドで実行してしまった場合は、まずCtrl＋Zキーなどでジョブを「一時停止」状態にしてから、bgコマンドでそのジョブをバックグラウンド状態で再開させればよい。

InterBase 6.0

今回は、サーバ上で稼働するプログラムであるトリガー、プロシージャについて説明します。今回から簡単なシステムのスキーマを定義しながら、前回までのテーブル設計、制約、インデックスなどについても簡単に復習してみましょう。

Database

第5回　トリガーとプロシージャについて

文：加藤大受
Text:Daiju Kato　kato@jcom.home.ne.jp

1カ月お休みをいただき、InterBase 6.0を使った簡単なシステムを作成しながら、InterBase 6.0の機能について再度調べてみました。ソース公開以来、少しずつ行われていたInterBase 6.0のオープンソースプロジェクトであるfirebirdプロジェクトですが、ようやく活発に行われるようになり、新しいバイナリが数回公開され、さらに安定した製品へと進化しています。なお、現在でもInterBaseの登録商標を米インプライズが持っているため、オープンソースプロジェクトがリリースした製品はfirebirdという名前を使っているようです。内容としては9月にリリースされたインプライズのベータ版をさらに安定化したもので、前バージョンとの互換性を重要視しています。

原稿執筆時点では、昨年12月17日にリリースされたfirebird ビルド0.9が最新版となっており、Classicアーキテクチャ版、SuperServerアーキテクチャ版の両方がリリースされています。また、Windows版、Solaris版だけでなく、FreeBSD版についてもベータ版の公開が始まっており、NetWare版についてもリリースに向けて着々と進んでいます。筆者の個人的な感想ですが、ビルド0.9は9月に公開されたバイナリよりも非常に安定している感じがします。

firebird 0.9のインストールについて

ビルドはRPM形式およびtar.gz形式の両方が用意されています。以前のInterBaseがインストールされている場合は、必ずアンインストールしてから新しいビルドをインストールしてください。インストール方法は、RPM形式の場合はrpmコマンドを利用します（**画面1**）。tar.gz形式の場合はファイルを解凍後に作成されるディレクトリ内のインストールプログラム（install.sh）を起動します（**画面2**）。このインストールプログラムは起動すると、ファイルのインストールと管理者パスワードの設定を行います。管理者パスワードの設定を聞いてきますので、パスワードを必ず設定してください。

RPM形式でインストールした場合、デフォルトの管理者パスワード、つまりSYSDBAのパスワードは/opt/interbase/SYSDBA.passwordに書かれています。

```
# rpm -ivh FirebirdSS-0.9-1.i386.rpm
```

画面1　RPM形式のインストール

firebirdプロジェクトについて	http://sourceforge.net/projects/firebird/
ソースコードの入手方法について	http://www.interbase2000.org/sourcecode.htm
firebird 0.9 for Linuxのダウンロード	ftp://firebird.sourceforge.net/pub/firebird/release/

表　URL一覧

このパスワードを変更する場合は、gsecユーティリティを利用して行います。ここでは従来のInterBaseのデフォルトパスワードに従って、「masterkey」に変更しています(**画面3**)。

見積書発行システムのスキーマ作成

前回まではInterBaseの機能について各機能ごとに説明していきましたが、今回からは簡単なシステムを構築しながら解説することにします。作成するシステムは「見積書発行システム」で、システムの概要は次のようになります。

・Webベースのシステムであること
・各会社ごとに掛け率が異なること
・いつ誰がシステムログインしたか、見積書を発行したかのログが取れること

まずシステムが使用するデータベースを作成しましょう。データベースは/home/db以下に格納されるものとします。今回のシステムでは、いつでもスキーマ構造のバックアップが取れるように、スキーマを作成するSQL文が書かれたSQLファイルを作成し、isqlユーテリティで実行する形とします。

isqlユーティリティを対話型で使用してデータベースを作成する場合と異なり、SQLファイルを作成した場合は、CREATE DATABASE構文にユーザー名とパスワードを指定する必要があります。今回のシステムは、データベースサーバとWebサーバをLinux上で使用するWebベースのシステムなので、ベースとなるキャラクタセットはEUCとします。

作成したデータベースに接続する場合はCONNECT構文を使用しますが、そのまま使用すると接続時のキャラクタセットがNONEになってしまうので、SET NAMES構文を使用してEUCを指定してからデータベースに接続します。

作成したSQLファイルは**リスト1**のようになります。

今回のシステムでは会社マスタ、商品マスタ、見積書マスタ、ログファイルの4つのテーブルが必要となります。

```
# tar zxvf FirebirdSS-0.9-1.tar.gz   ← tar.gzを解凍する
# cd FirebirdCS-0.9-1                ← 解凍したディレクトリに移動する
# ./install.sh                       ← インストールプログラムを起動する
```

画面2　tar.gz形式のインストール

```
# cat /opt/interbase/SYSDBA.password                    ← 現在のパスワードの確認
Firebird generated password
for user SYSDBA is : tY88gvRi                           ← 筆者の環境のデフォルトパスワード
# /opt/interbase/bin/gsec -user sysdba -password tY88gvRi   ← gsecユーティリティでパスワードを変更
GSEC> modify sysdba -password masterkey
GSEC> quit
```

画面3　SYSDBAのパスワードの変更（RPM形式でインストールした場合）

▼リスト1　データベースの作成とデータベースへの接続

```
/*********************************/
/*       データベースの作成          */
/*********************************/

CREATE DATABASE '/home/db/estimate.gdb'
        USER 'SYSDBA'
        PASSWORD 'masterkey'
        PAGE_SIZE 8192
        DEFAULT CHARACTER SET EUCJ_0208;

/**** 作成したデータベースへの接続  ****/
SET NAMES EUCJ_0208;
CONNECT '/home/db/estimate.gdb' USER 'SYSDBA' PASSWORD 'masterkey';
```

図1は、これらのテーブルの関係を図にしてみたものです。各テーブルの最初のフィールドがプライマリキーとなるように書いてあります。図を見てみるとわかるように、見積書マスタに格納される会社コード、商品コードは、それぞれ商品マスタおよび、会社マスタのプライマリキーとなります。ログファイルのログインIDは、会社マスタのログインIDを参照しますので、外部参照を設定するために会社マスタのログインIDにユニークインデックスを設定する必要があります。

各テーブルを設計するSQL文は**リスト2**のようになります。会社マスタのログインIDにユニークインデックスを設定していますので、UNIQUE節を追加しています。見積書マスタでは会社コードと商品コードを、会社マスタと商品マスタから参照していますので、外部キーの設定が行われています。同じようにログファイルでは、ログインIDに外部キーを指定しています。

ジェネレータについて

InterBaseにはジェネレータという、自動的にインクリメントされる値を作成する機能があります。この機能は、ユニークな値を必要とするフィールドなどで一般的に使用されます。ただし、一般的にジェネレータは単独で使用されるのではなく、トリガーと組み合わせされて使用されます。これはジェネレータで作成された値をトリガーで指定された列に設定させるためです。

ジェネレータを作成するにはCREATE GENERATOR構文を使用します（**リスト3**）。

値を使用する場合は、GEN_ID()関数を呼び出してユニークな数を作成します。ジェネレータを初期化したい場合は、SET GENERATOR構文で新しい値に設定します（**リスト4**）。

ジェネレータは非常に便利な機能ですが、会社コードや商品コードを設定するときには向いていません。ジェネレータで作成された値はトランザクションをロールバックしても戻りません。そのため、このようなコードに使用した場合、番号が歯抜け状態になる可能性があります。この点を注意して使用してください。

トリガーについて

トリガーとは、テーブルに対してデータ操作が行われたときに実行されるプログラムです。InterBaseのトリガーには、更新前と更新後に行われる2種類のトリガーがあり、1つのテーブルには65535までのトリガーの設定が可能です。データが追加されるときに実行されるトリガーには、BEFORE INSERTとAFTER INSERTの2種類があり、BEFORE INSERTのトリガーは、ジェネレータと組み合わせてユニークな値をフィールドに指定したりする場合によく使われます。AFTER INSERTのトリガーは、別テーブルにデータを書き込んだりする場合に使用されます。また、トリガーはPOSITIONを指定することで優先順位

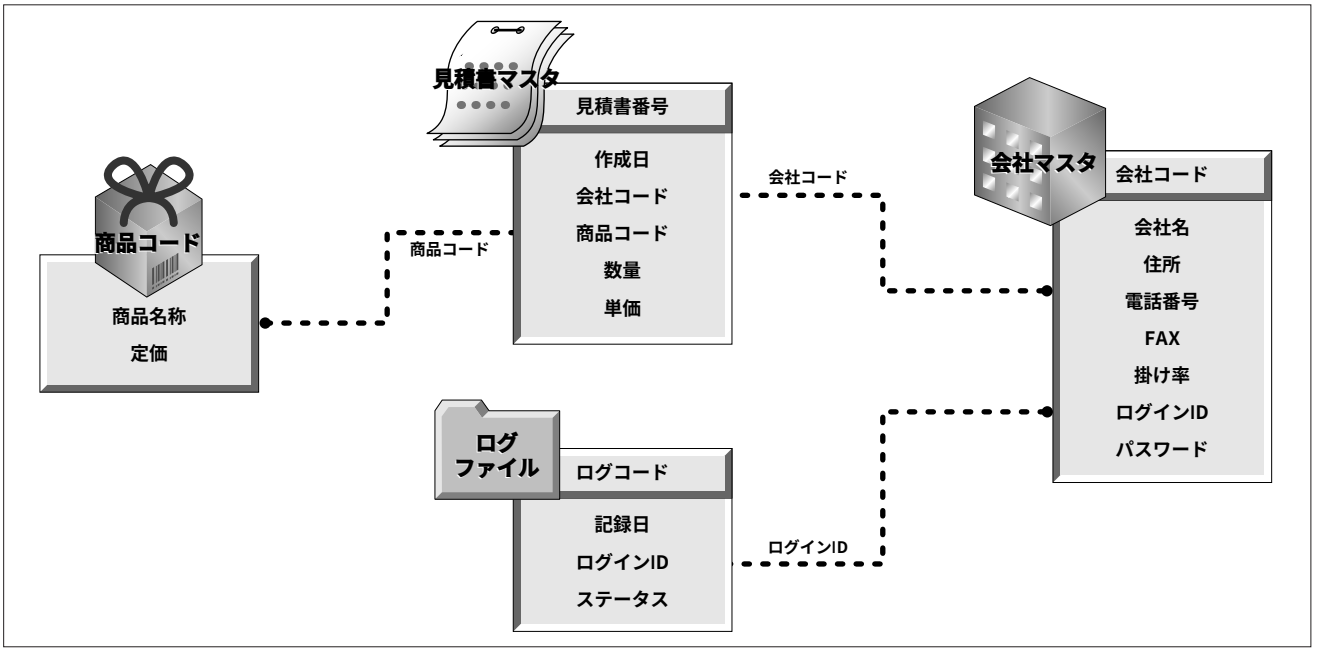


図1　テーブルの関係

を指定することができます。

今回の見積書発行システムでは、会社コード、商品コード、見積書コード、ログIDの設定をトリガーで行います。各テーブルとトリガーとの関係を図にすると**図2**のようになります。

図で示したように、今回のシステムでは6つのトリガーを作成します。InterBaseにはトリガーとプロシージャ作成用の言語が用意されており、この言語はSQL文による操作だけでなく、代入文、フロー制御、コンテキスト変数、イベント通知文、例外、エラー処理などを利用することが可能です。トリガーはCREATE TRIGGER構文で作成します（**リスト5**）。

リスト6はシステムで使用する6つのトリガーを作成するSQL文です。CREATE TRIGGER文の前にSET TERM構文を使用していることに気がつくと思います。これはisqlユーテリティで、トリガーまたはプロシージャを終了させるために使用するターミネータを指定しています。つまり、**リスト6**の例では二重感嘆符（!!）をターミネータとして使用し、どこまでがトリガーやプロシージャの定義かを区別できるようにしています（**リスト7**）。isql

▼リスト2　テーブルを作成するSQL文

```
/******     会社マスタ     *****/
CREATE TABLE COMPANY (
       COMPANY_ID   CHAR(10)  NOT NULL PRIMARY KEY,   /* 会社コード   */
       COMPANY_NAME CHAR(80)  NOT NULL,               /* 会社名称     */
       ADDRESS      CHAR(128) NOT NULL,               /* 住所         */
       TEL          CHAR(15)  NOT NULL,               /* 電話番号     */
       FAX          CHAR(15)  NOT NULL,               /* FAX          */
       RATE         INTEGER   NOT NULL,               /* 掛け率       */
       LOGIN_ID     CHAR(10)  NOT NULL UNIQUE,        /* ログインID   */
       LOGIN_PASS   CHAR(20)  NOT NULL                /* パスワード   */
       );

/******     商品マスタ     *****/
CREATE TABLE PRODUCTS (
       PRODUCT_ID   CHAR(10)  NOT NULL PRIMARY KEY,   /* 商品コード   */
       PRODUCT_NAME CHAR(80)  NOT NULL,               /* 商品名称     */
       LIST_PRICE   INTEGER   NOT NULL                /* 定価         */
       );

/******   見積書マスタ     *****/
CREATE TABLE ESTIMATE_MASTER (
       ESTIMATE_ID CHAR(10) NOT NULL PRIMARY KEY,  /* 見積書番号      */
       MADE_DATE   DATE     DEFAULT 'NOW',         /* 作成日          */
       COMPANY_ID  CHAR(10) NOT NULL,              /* 会社コード      */
       PRODUCT_ID  CHAR(10) NOT NULL,              /* 商品コード      */
       AMOUNT      INTEGER  NOT NULL,              /* 数量            */
       PRICE       INTEGER  NOT NULL,              /* 価格            */
       FOREIGN KEY (COMPANY_ID) REFERENCES COMPANY (COMPANY_ID),
       FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCTS (PRODUCT_ID)
       );

/******     ログファイル     *****/
CREATE TABLE LOGS (
       LOG_ID   CHAR(10)  NOT NULL PRIMARY KEY,       /* ログコード   */
       LOG_DATE DATE      DEFAULT 'NOW',              /* 記録日       */
       LOG_USER CHAR(10)  NOT NULL ,                  /* ログインID   */
       STATUS   CHAR(20)  NOT NULL,                   /* ステータス   */
       FOREIGN KEY (LOG_USER) REFERENCES COMPANY (LOGIN_ID)
```

▼リスト3　CREATE GENERATOR構文

```
CREATE GENERATOR <ジェネレータ名>;
```

▼リスト4　SET GENERATOR構文

```
SET GENERATOR <ジェネレータ名> TO <新しい値>;
```

ユーティリティで対話形式で作成するときはターミネータは必要ありませんが、定義をまとめてSQLファイルとして実行する場合は必要となりますので忘れないようにしてください。

トリガーで変数を指定するときは、AS句の前でDECLARE VARIABLE文を使用します。また、トリガーの定義部ではフィールド名と区別するため、変数名はコロン（：）で開始します。たとえば、SET_COMPANY_IDのトリガーでは、現在の最大値を一時的に格納する変数として、数値型のMAX_IDを定義しています。

トリガー内で条件式を使用することができます。IF～THEN～ELSEは条件が真の時と偽の時とで流れを変えることができます。SET_COMPANY_IDのトリガーでは、MAX_IDがヌルの場合の処理と、ヌルではないときとで処理を変えています。

▼リスト5　CREATE TRIGGER構文

```
CREATE TRIGGER構文
CREATE TRIGGER <トリガー名> FOR <テーブル名>
        [ACTIVE | INACTIVE]
        {BEFORE | AFTER}
        {DELETE | INSERT | UPDATE}
        [POSITION number]
        AS <変数宣言部>
        BEGIN
           <トリガー定義部>
        END
```

ヌルの場合、つまりデータが全く入っていない場合は、会社コードに100000を、すでにデータが入っている場合は最大値＋1を会社コードに代入します。

トリガーでは、NEWとOLDというコンテキスト変数を利用することができます。NEWはINSERT文または、UPDATE文で実行されるトリガーで使用されるコンテキスト変数で、NEW.＜フィールド名＞とすることで、挿入後または更新後の値を指定します。SET_COMANY_IDのトリガーでは、NEW.COMPANY_IDとすることで新しい会社コードの値を示しています。

SET_PRODUCT_ID、SET_ESTIMATE_ID、SET_LOG_IDで行っている処理は、SET_COMPANY_IDとほぼ同じで、商品コード、見積書コード、ログIDを設定しています。

COMPUTED_PRICEのトリガーでは、見積書マスタに書かれた会社コードをキーにして会社マスタからその会社の掛け率を、商品コードをキーにして商品マスタから定価を取り出し、単価を計算しています。データベースを利用するプログラムでこの処理を書いてしまってもかまいませんが、サーバ側でできることは、できるだけサーバ側で処

▼リスト7　ターミネータ指定

```
SET TERM !!;
  <トリガーまたはプロシージャの定義>
!!
SET TERM ;!!
```

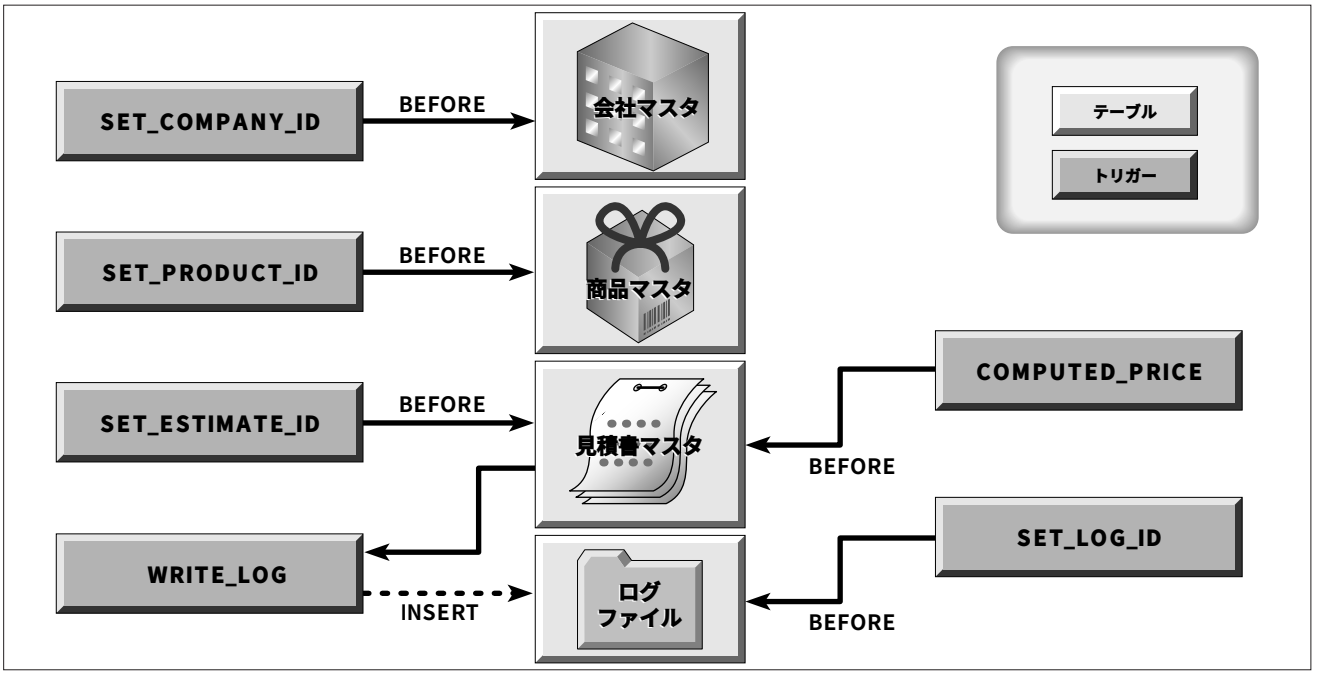


図2　トリガーの関係

▼リスト6　トリガーを作成するSQL文

```
/*********************************/
/*        トリガーの作成           */
/*********************************/

/******  COMPANY_IDを設定するSET_COMPANY_ID ****/
SET TERM !!;
CREATE TRIGGER SET_COMPANY_ID FOR COMPANY
   BEFORE INSERT
   AS
     DECLARE VARIABLE MAX_ID INTEGER;
   BEGIN
     /*現在の最大値を取得*/
     SELECT MAX(COMPANY_ID) FROM COMPANY INTO MAX_ID;
     IF (MAX_ID IS NOT NULL) THEN
        BEGIN
          NEW.COMPANY_ID=MAX_ID+1;
        END
     ELSE
        BEGIN
          NEW.COMPANY_ID=100000;
        END
   END
!!
SET TERM ;!!


/******  PRODUCT_IDを設定するSET_PRODUCT_ID ****/
SET TERM !!;
CREATE TRIGGER SET_PRODUCT_ID FOR PRODUCTS
   BEFORE INSERT
   AS  DECLARE VARIABLE MAX_ID INTEGER;
   BEGIN
     /*現在の最大値を取得*/
     SELECT MAX(PRODUCT_ID) FROM PRODUCTS INTO MAX_ID;
     IF (MAX_ID IS NOT NULL) THEN
        BEGIN
          NEW.PRODUCT_ID=MAX_ID+1;
        END
     ELSE
        BEGIN
          NEW.PRODUCT_ID=100000;
        END
   END
!!
SET TERM ;!!


/******  ESTIMATE_IDを設定するSET_ESTIMATE_ID ****/
SET TERM !!;
CREATE TRIGGER SET_ESTIMATE_ID FOR ESTIMATE_MASTER
   BEFORE INSERT
   AS  DECLARE VARIABLE MAX_ID INTEGER;
   BEGIN
     /*現在の最大値を取得*/
     SELECT MAX(ESTIMATE_ID) FROM ESTIMATE_MASTER INTO MAX_ID;
     IF (MAX_ID IS NOT NULL) THEN
        BEGIN
          NEW.ESTIMATE_ID=MAX_ID+1;
        END
     ELSE
        BEGIN
          NEW.ESTIMATE_ID=100000;
        END
   END
!!
SET TERM ;!!


/***** 見積もりの単価計算を行うCOMPUTED_PRICE ****/
SET TERM !!;
CREATE TRIGGER COMPUTED_PRICE FOR ESTIMATE_MASTER
  BEFORE INSERT
  AS
    DECLARE VARIABLE PRICE_RATE INTEGER;
    DECLARE VARIABLE LISTED_PRICE INTEGER;
  BEGIN
    /*掛け率の取得*/
    SELECT RATE FROM COMPANY
           WHERE COMPANY_ID=NEW.COMPANY_ID INTO :PRICE_RATE;
    /*定価の取得*/
    SELECT LIST_PRICE FROM PRODUCTS
           WHERE PRODUCT_ID=NEW.PRODUCT_ID INTO :LISTED_PRICE;
    /*単価の計算*/
    NEW.PRICE=:LISTED_PRICE*PRICE_RATE/100;
  END
!!
SET TERM ;!!


/******  LOG_IDを設定するSET_LOG_ID ****/
SET TERM !!;
CREATE TRIGGER SET_LOG_ID FOR LOGS
   BEFORE INSERT
   AS
   BEGIN
     NEW.LOG_ID=GEN_ID(GEN_LOG_ID,1);
   END
!!
SET TERM ;!!


/***** 見積もり発行後にログに記録するWRITE_LOG ****/
SET TERM !!;
CREATE TRIGGER WRITE_LOG FOR ESTIMATE_MASTER
   AFTER INSERT
   AS
     DECLARE VARIABLE USERNAME CHAR(10);
   BEGIN
     /*ログインIDの取得*/
     SELECT LOGIN_ID FROM COMPANY
            WHERE COMPANY_ID=NEW.COMPANY_ID INTO :USERNAME;
     /*ログテーブルに書き込み*/
     INSERT INTO LOGS
            (LOG_USER, STATUS)
          VALUES
            (:USERNAME, 'CREATED');
   END
!!
SET TERM ;!!
```

理することで、クライアント側のプログラムを簡素化することができ、パフォーマンスを向上することができます。

見積書にデータが書き込まれると、ログファイルに見積書が作成されたことを示す記録が書き込まれます。この処理は、見積書マスタのAFTER INSERTトリガーであるWRITE_LOGトリガーで行われています。このように、AFTERトリガーとBEFOREトリガーを上手に使い分けることで、さまざまな処理をサーバサイドで実現することができます。

プロシージャの作成

プロシージャは、トリガーと同様にサーバ側で実行されるプログラムです。トリガーと違い、プロシージャは単独で実行されます。InterBaseには、実行プロシージャと選択プロシージャの2種類が用意されており、選択プロシージャは指定された条件から行セットを、実行プロシージャは指定された条件から値を返します。選択プロシージャは、ビューやテーブルと同じように、SELECT構文で実行することができる非常に珍しい機能です。

今回は選択プロシージャは使用しませんが、見積書発行システムへログインするときのユーザー名とパスワードのチェックを実行プロシージャで行います。**リスト8**はシステムへのログインを確認するCHECK_LOGINプロシージャです。プロシージャの作成はCREATE PROCEDURE構文で行います（**リスト9**）。

ここでは、ログインIDとパスワードを入力パラメータとし、システムへのログインに成功するかどうかを示す、ISLOGIN変数を返り値としています。入力されたログインIDをキーにして会社マスタからパスワードを検索し、入力されたパスワードが合っているかどうかをチェックしています。正しい場合は、ログファイルへシステムにログインしたことを記録したあと、ISLOGINに真である1を、間違っているときはISLOGINに0をセットしています。

▼リスト8　CHECK_LOGINプロシージャ

```
/*********************************/
/*     プロシージャの作成          */
/*********************************/

/*--- システムへのログインを確認するCHECK_LOGIN ----------*/
/*      INPUT:USERNAME   CHAR(10)                   */
/*            PASS       CHAR(20)                   */
/*     OUTPUT:ISLOGIN  1:成功  0:失敗                */
/*-------------------------------------------------*/
SET TERM !!;
CREATE PROCEDURE CHECK_LOGIN(USERNAME CHAR(20), PASS CHAR(20))
  RETURNS (ISLOGIN INT)
AS
  DECLARE VARIABLE ORGPASS CHAR(20);
BEGIN
  SELECT UPPER(LOGIN_PASS) FROM COMPANY INTO :ORGPASS;
  IF (:PASS=:ORGPASS) THEN
     BEGIN
        ISLOGIN=1;
     /*ログファイルへ記録*/
     INSERT INTO LOGS
            (LOG_USER, STATUS)
          VALUES
            (:USERNAME, 'LOGIN');
     END
  ELSE
     BEGIN
        ISLOGIN=0;
     END
END
!!
SET TERM ; !!
```

筆者の場合、複雑なSQL文などでできる処理であっても、できるだけトリガーやプロシージャにするようにしています。これは副問い合わせなどを含む、複雑なSQL文はメンテナンスしづらく、またパフォーマンスが問題になりやすいからです。さらに、データベースの移行が難しくなる可能性もあります。誰でも簡単にメンテナンスできるように、できるだけ簡単なSQL文と、トリガーとプロシージャを使用してシステムをわかりやすく構築するようにしています。

SQLスクリプトファイルの実行

それでは完成した見積書発行システムのスキーマを作成する、SQLスクリプトファイルを実行してみましょう。作成したSQLファイルである、schema.sqlは本誌付録CD-ROMに収録していますので、実際に実行してみてください（**リスト10**）。isqlユーティリティで、SQLスクリプトファイルの実行をするときは、-iオプションを指定します。実行すると、データベース、テーブル、インデックス、トリガー、プロシージャが一気に作成されることがわかると思います。実行したあと、対話形式でisqlユーティリティを使用して、正しくスキーマができているか確認してみてください。

付録CD-ROMには、サンプルデータを作成するスクリプトファイルであるdata.sqlも含まれていますので、同様に実行してみてください（**リスト11**）。

ユーザー情報のテーブルへの格納について

今までの説明を見てきた方の中には、ユーザー情報についてはInterBaseサーバ自身のユーザーを使うほうがいいのではと思われる方もおられるでしょう。確かに、InterBaseのユーザーを使用してしまうのもひとつの方法かもしれません。しかし、データベースサーバに何らかのトラブルが発生した場合、せっかくデータベースファイルのバックアップを取っていても、この場合ではシステムが復元できない可能性が高くなってしまいます。また、複数のシステムで共通のサーバを使用する場合は、ユーザー数が多くなってしまい、管理ができなくなる可能性もあります。このようなことを考慮すると、システムへの最初のログインには共通のユーザーIDを使用し、その後入力されたユーザー名とパスワードが一致しているかをデータベース内のテーブルと照合したほうがいいでしょう。SQL文を利用して照合する方法もありますが、SQL文を使用する場合では、使用するプログラム言語によって記述方法が変わってきますので汎用性は乏しくなります。そこで、今回使用したようなログインできるかどうかチェックするプロシージャを作成しておくと、さまざまなシステムで利用していくことができるだけでなく、さまざまなプログラム言語で使用することが可能になります。これにより、汎用性が高くなり、生産性を上げることができます。データベースに慣れていないうちはどれが汎用的なモジュールとなるか、プロシージャにしておくほうがいいのかを決めることは大変かもしれませんが、自分自身でシステムを作成していくときのパターンと、プロシージャを利用する方法を理解しておくことで、徐々にデータベースを使用するシステムの構築にかかる時間が短くなっていくことでしょう。

選択プロシージャの活用について

今回のシステムでは使用していませんが、前述したようにInterBaseには、選択プロシージャと呼ばれるSELECT文で使用できるプロシージャがあります。簡単にいうと、選択プロシージャは複数行からなるデータセット（結果セット）を返すことができるものです。通常、RDBMSでプロシージャというと値を返すものが一般的ですが、このように行セットを返すことができる選択プロシージャがあると、副問い合わせなどの複雑なSELECT文を書くことなく処理を行うことが可能になります。このような機能が用意されているのも、InterBaseの特徴のひとつです。

次回はInterBaseのセキュリティについてと、クライアントの作成について解説します。

▼リスト9　CREATE PROCEDURE構文

```
CREATE PROCEDURE <プロシージャ名>
        <入力パラメータ宣言部>
        RETURNS <出力パラメータ宣言部>
        AS <変数宣言部>
        BEGIN
          <プロシージャ定義部>
        END
```

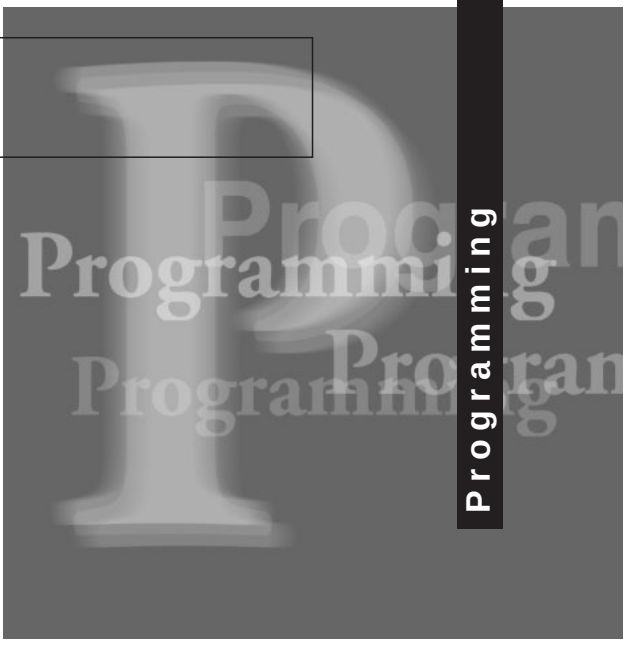
▼リスト10　スキーマの作成

```
#/opt/interbase/bin/isql -i/home/db/schema.sql
```

▼リスト11　サンプルデータの作成

```
#/opt/interbase/bin/isql -i/home/db/data.sql
```

プログラミング工房



文字の処理はコンピュータが得意とする分野だ。AT&Tのベル研で生まれたUNIXは、もともと文書処理を行うために開発されたものである。今月から、C言語で文字を扱う例題プログラムを実際に作成しながら、そのプログラミングテクニックについて解説してみる。

第15回　文字列処理プログラム

文：藤沢敏喜
Text : Toshiki Fujisawa

文字列検索プログラムの作成

PCが一般的になった現在でも、「パソコンができる」というのは、「WordとExcelが操作できる」ことであると誤解している人は多い。また、viやEmacsなどのエディタ画面で、1日の大半を過ごしている読者も多いだろう。このような文書処理を行うアプリケーションを作るうえでは、文字列処理を行うプログラミングが重要になる。

文字を扱うプログラムの花形は、やはりスクリーンエディタであろう。しかしながら、実用的なエディタを作成するためには、相当な分量のプログラムを書かなくてはならず、雑誌の連載で解説するには難しい。また、すでにviやEmacsといったすばらしいエディタがあるので、勉強のためだけにそれほど実用的ではないエディタのプログラムを作成するのは、あまり気の進まない作業である。

そこで今回は、文書から特定のパターンの文字列を検索して、その検索結果を表示するプログラムを作成してみることにする。

BIBLE CODE

年末に、書店のコンピュータ書籍コーナーで、『神の暗号』という本を目にした。こんな本がなぜコンピュータ技術書の中に紛れているのかと不思議に思ったが、裏表紙を見ると「アスキー出版局」と書かれていた。アスキーがこの手の本を出版することは稀なので、書店の人が間違えた場所に置いてしまったようである。

買うかどうか迷ったが、編集人が月刊アスキーの遠藤編集長だったのと、次の日から帰省する予定であり、新幹線の中で暇潰しをするための本が必要だったこともあって購入することにした。

読んでみると、世界のさまざまな事件や個人の人生の出来事が、聖書の中に暗号化されているという内容であった。この本では、著者の知人がガンで亡くなったことまでが暗号化されていたと報告されている。

本当にそんなことがあるのだろうか？ この本を読み終えた20世紀最後の大晦日には、今まで報道されたUFOや超能力はすべて嘘だった、というテレビ番組が放映されていた。また、ここ数年はUFOや超能力などの怪しげな現象を、膨大で詳細な資料の元に否定する書籍も多く発行されるようになってきている。

21世紀初日に帰省先でインターネットを使って調べてみると、この聖書の暗号に関する怪しい点を、数式を基に詳細な確率を計算して論じているWebページが見つかった。聖書の暗号は、すべてコンピュータで処理することができ、その真偽も確率を計算することによって、厳密に議論することができるようである。

したがって、UFO目撃談とは違い、論理的かつ明確に否定できそうな気がする。しかし、書籍に掲載されている

例だけを見て否定するにはデータが不十分なので、暗号の反証例を見つけるためには、簡単に検索できるプログラムが必要になる。

また、この聖書における暗号アルゴリズムはたいへん簡単なものであり、文字列処理プログラミング入門用の教材としては、とても適切であるように思えた。

そこで、この連載で文字列処理プログラミングに必要な手法を解説できるような形で、聖書の暗号を検索できるプログラムを作成することにした。

今回取り上げるプログラミング手法

今回作成するプログラムは、コマンドラインから検索すべき単語を入力すると、その単語が聖書の中のどの位置で、どれくらいの文字間隔で発見されたかを示すものである。

また、コマンドラインから複数の単語を入力すると、その単語が2次元空間でどのように配置されるかを、見やすい形で画面に表示する。

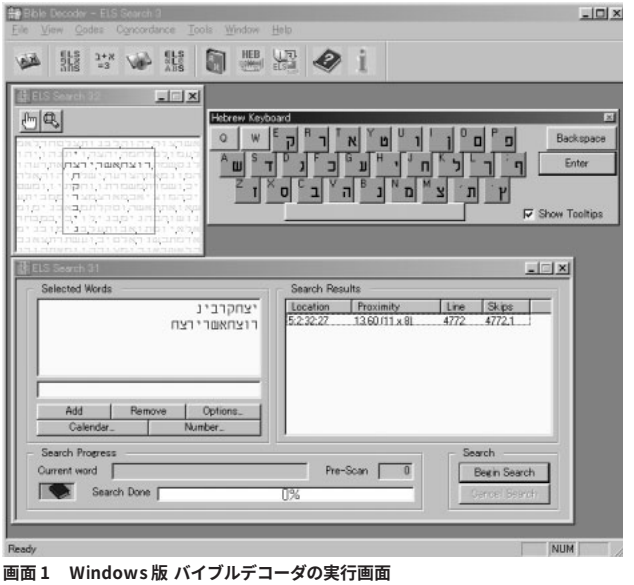
このようなプログラムに必要なプログラミング手法としては、次のようなものが必要になる。

- **コマンドラインのオプション解析**
- **巨大文字データのプログラムへの埋め込み**
- **文字列の検索**
- **単語リストの処理**
- **2次元空間文字列の扱い**
- **エラー処理**
- **文字列の画面表示**

上記のなかでも重要なのは、2次元空間に配置される文字列の扱いと、その画面表示である。これは、たとえばエディタを作成する際に、ファイルに保存された1次元の文字列を行ごとに区切って、CRTなどの2次元画面に表示する際に必要になる技術である。

GUIとコマンドラインインターフェイス

最近のWindows用プログラムは、GUI（グラフィカル・ユーザー・インターフェイス）を使ったものがほとん



画面1　Windows版 バイブルデコーダの実行画面

Column

暗号解析とコンピュータ

コンピュータユーザーには、なぜか暗号解読の分野にも興味を示す人が多いような気がする。

ちょっと前にRC64と呼ばれる暗号解読コンテストが行われたが、全世界にある無数のコンピュータがこの暗号解読のために多くのCPUタイムを費やした。このコンテストは、グループでの参加が認められたため、日本からも多くのグループが参加した。JFUG（Japan FreeBSD Users Group）がかなり上位に食い込み、JLUG（Japan Linux Users Group）もそれに近いところまで追い上げた。

筆者の友人は数十台の高速マシンをブンまわしていたし、筆者も一時期はJFUGのために10台ほどのマシンを投入していたこともある。

最近でも、SETIと呼ばれる宇宙からの電波の中に知的生命体からのメッセージがないかを検索するためのプロジェクトが行われている。筆者の知人にも自宅で24時間運転の冷房代を投入してまで、何台もの高速マシンを稼働させている人が結構多い。

このような人たちは、解読が目的というよりも、CPUを高速に動作させることが目的となっているような気がするのは、気のせいだろうか？ そのような人々にとっては、CPUを聖書の暗号解読のためにガンガン回すのも、またおもしろく感じるのではないだろうか。

聖書の暗号解析は、最近の高速PCを使用しても、1つの単語を検索するのに十数分もの時間がかかる。このため、いくつかの単語の一致を検出するような場合では、何日もの演算を必要とする。このようなCPUを酷使することが可能なプログラムは、CPUのオーバークロックを趣味とする人が楽しむにはもってこいの題材であろう。

また、この演算は分割して、並列に処理することが可能なため、ネットワークで接続した複数のコンピュータで分散処理するのに向いている。そのため、グループで協力して演算を行うというプロジェクトを行うこともまた楽しいかもしれない。

どである。聖書の暗号を解析する「バイブルデコーダ」と呼ばれるソフトウェアも、やはりGUIアプリケーションである（**画面1**）。

このバイブルデコーダのプログラム本体は、グラフォネット社のページ（http://www.biblecodes.com/）から、だれでもダウンロードできるようになっている。ただし、検索を行うためにはレジストキーが必要となる。35ドルをクレジットカードで支払うと、キーが電子メールで送られ

Column

聖書の暗号に関する情報

聖書の暗号に関連する書籍のうち、日本語で出版されているものは、筆者の知る限り次の4冊である。

- **『聖書の暗号』マイケル・ドロズニン著 木原武一訳 新潮社 1997年8月発行 ISBN4-10-535901-0**
- **『聖書の暗号は本当か』久保有政著 徳間書店 1998年7月発行 ISBN4-19-860881-4**
- **『聖書のミステリー』ジュフリー・サティンオーヴァー著 仲村明子訳 徳間書店 1998年12月発行 ISBN4-19-860954-3**
- **『神の暗号』伊達 巌著 アスキー出版局 1999年12月発行 ISBN4-7561-3299-5**

『聖書の暗号』の著者は、ワシントンポストなどで記者をしていたことがある人で、内容は主にアメリカやイスラエルの科学者からの取材で構成されている。この本では、聖書に暗号化されているというさまざまな事件の例が掲載されている（**写真1**）。

『聖書の暗号は本当か』では、聖書アナリストである著者が、自分のWindowsマシンで、バイブルデコーダと呼ばれるソフトウェアを用いて、日本における事件などさまざまな検索を行った結果が報告されている。聖書に詳しい著者であるためか、聖書の内容との関連についても詳しく述べられている（**写真2**）。

『聖書のミステリー』は、ハーバードとMITで医学と数学の学位を取得した精神科医の著作である。数学的な確率の面から聖書の暗号を考察し、その確率計算の結果から神を信じるようになった科学者の話などが書かれている（**写真3**）。

『神の暗号』は、日本の歴史を材料にした検証が多いのが特徴で、邪馬台国や鎌倉時代の昔から、カレー殺人事件まで、現代日本でのさまざまな事件が取り上げられている（**写真4**）。

上記4つの書籍はいずれも、聖書の暗号が存在することを信じる（あるいは信じるようになった）著者が書いたものであり、批判的な立場からの書籍は、日本ではまだ出版されていないようだ。

一方、インターネット上では、このあまりにも無謀な仮説を否定する意見も多いようである。日本語の情報としては、東海大学の春田晴郎氏のWebサイト（http://bosei.cc.u-tokai.ac.jp/~haruta/indexj.html）がたいへん詳しい。

このWebサイトからたどれる聖書の暗号に対する反論は、非常に詳しく、かつ論理的に行われている。

聖書の暗号に関する基本的な問題点や、元データの選択に関する不自然さ、そして問題の論文の査読に関しての疑問や、不明な点などが30ページに渡ってたいへん詳しく解説されている。

なかでも「論文中の暗号がヘブライ語で記述されているのが、読者を混乱させている」という指摘には筆者も強く共感し、普通の英文字を使って検索できる今回のプログラムを作成する強い動機となった。また、春田氏のWebサイトには、以前に翔泳社から出版されて話題になった『神々の指紋』批判ページもあり、こちらのほうもまた詳細で興味深い。

そして、聖書データの所在など、今回のプログラムを作るうえでたいへん参考になったのが、戸塚直哉氏のWebサイト（http://www.naochan.com/index-jp.html）である。

戸塚氏はLinuxユーザーでもあり、Javaを使って聖書の暗号が検索できるようなWebページもある。また、ここにはヘブライ語の文字出現頻度を元にした、かなり詳しい確率計算の解説もあり、8文字の出現確率が10%程度と、さほど珍しくないことなどが示されている。

写真1　『聖書の暗号』

写真2　『聖書の暗号は本当か』

写真3　『聖書のミステリー』

写真4　『神の暗号』

てくる。しかしながら、「神の暗号」の著者によると、バイブルデコーダは文字間隔が2万までしか解析できないなどの制限があったり、Windowsプログラムによくあるように、ハングアップしてしまうことがあるという。

この原稿を書くために試しに購入してみたのだが、レジストキーの送付に日数がかかったため、Windows版のバイブルデコーダを実際に使用してみたのは、原稿を書き上げる数時間前であった。まだよく使い方がわかっていない状態であるためか、慣れないヘブライ文字をGUIの仮想キーボードを使って1文字ずつマウスでクリックして入力する必要があるなど、操作は筆者にとってたいへん難しいものであった。

筆者は、Gimpなどの一部の画像アプリケーションを除けば、マウスでクリックするGUIプログラムよりも、コマンドラインプログラムのほうが、はるかに使いやすいと思っている人間である。コマンドラインプログラムの場合は、多量の処理をバッチ的に扱うことも可能であり、その出力をあとで処理することもまた簡単である。

やはりUNIX使いとしては、ハングアップとは無縁なLinuxやFreeBSD上で動作して、bashのコマンドラインからキーボードを使ってバシバシ使える、コマンドライン・インターフェイスのプログラムがほしいものである。

また、出力結果が単なるテキストファイルであれば、AWKやPerlを使い自由自在に柔軟な解析が可能になる。

```
BIBLE CODE EXTRACT PROGRAM  version-1.00 Jan.8.2001
  Copyright(c) Toshiki Fujisawa (fujisawa@fujisawa.gr.jp)

usage: xbible [options] word...
       -x            : execute command display (all search)
       -g             : grep mode (display near chars)
       -v              : verbose display
       -c {count}      : count of display
       -w {win_size}   : window size of terminal
       -n {num_range}  : number range from top in bible
       -s {skip_range} : skip min and max
       -l {line_div}   : line divide number
       -o {ofset_disp} : offset col and row of display

       {win_size}    -w80x25
       {num_range}   -n20000 -n2000-2999 -n2000+999
       {skip_range}  -s4772  -s4772-6772 -s4772+2000
       {line_div}    -l3 -l3:+1 -l3:-1 -l:4772
       {offset_disp} -o+3+4 -o-3-4

ex1: xbible -x   ycxqr
ex2: xbible -xgv ycxqr
ex3: xbible -n254245 -s4772 -w78x18 -o-1,+2 ycxqrbyn
xcryrsaxcvr
```

画面2　今回作成したプログラム（xbible）のヘルプ画面

コマンドラインオプション

今回作成したプログラムは、聖書の暗号を抽出するプログラムなので、bibleにextractのXを頭に付け、xbibleと命名することにした（将来、X Window System上での表示が可能になった場合も、名称を変更しなくてすむ。ヘブライ語のデータは、インターネット上（163ページのコラム「聖書の電子データとヘブライ語」参照）からダウンロードしたものを利用している。xbibleの実行については、末尾の「次号の内容とプログラムの実行」に記述した。

xbibleは、**画面2**に示すようなヘルプ画面のようにして使うことが可能である。たとえば、ヘブライ語で乾いた大地を意味する「ybsh」と、その前に付く前置詞「l」を引数にして、

```
$ xbible lybsh
```

を実行すると、「lybsh」が見つかった位置を中心として、その周りの聖書の文字列が表示される。この場合は、聖書の最初にある天地創造の部分が示されることになる。

また、イスラエルの故イツハク・ラビン首相のヘブライ語表記「ycxqrbyn」（cはヘブライ文字のツアディを示す）と、暗殺者は暗殺するだろうという意味のヘブライ語である「xcryrsaxcvr」を指定して、

```
$ xbible -x ycxqrbyn xcryrsaxcvr
```

を実行すると、聖書の中から「ycxqrbyn」という文字列がどの場所に、何文字間隔で存在するかをすべて探し出す。この場合、Pentium III 650MHzのPCで実行すると、約13分後にすべての検索が終わる。途中経過を示したい場合は、-vオプションを付けるとよい。検索結果は、

```
xbible -n 254245 -s 4772 ycxqrbyn  xcryrsaxcvr
```

というように表示され、「ycxqrbyn」という文字列が聖書の254245番目の文字から4772文字間隔で見つかったことが示される。

この場合、見つかったのは全文書の中でこの1カ所だけであるが、複数の位置で見つかった場合は、その数の行だけ表示が行われる。ちなみに、検索結果を上記のような形

で表示するのは、シェルのコマンドラインでパイプを使うことにより、検索と表示を同時に行うことができるようにするためである。

上記の-nオプションは、聖書の先頭から数えて何文字目から調べるかを指定し、-sオプションは何文字間隔から検索し始めるかを指定する。したがって、先ほど表示された行、つまり、

```
$ xbible -n 254245 -s 4772 ycxqrbyn  xcryrsaxcvr
```

を実行すると、「ycxqrbyn」と「xcryrsaxcvr」がどのような位置関係にあるかを瞬時に表示できることになる。

ここで、ウィンドウサイズを-wオプションで横78縦18行に指定し、表示位置のオフセットを-oオプションでX方向に-1、Y方向に＋2ずらすように「-w78x18 -o-1,＋2」というオプションを付加すると、コラムの**写真1**で示した『聖書の暗号』の表紙とまったく同じ暗号表が表示される(**図1**)。

さらに、行の分割数を-l3オプションで指定し、ウィンドウサイズを-w78x33にし、表示位置のオフセットを、「-o＋7,2」として指定する。そして、関連するいくつかの単語を追加して次のようなコマンドを実行してみる。

```
$ xbible -l 3 -w 78x33 -o +7,+2 -n 254245 -s4772 \
                ycxqrbyn xcryrsaxcvr ntnyhv \
                smhrcx hmxlmlvmolk omyro
```

この結果として、『聖書の暗号』の86ページにある暗号表が表示される。

コマンドラインオプション解析

上記で述べたような、コマンドラインオプションを解析するプログラムを作成するには、どうしたらよいのだろうか？ 以前のこの連載で解説したように、コマンドラインはmain関数の2つの引数「argc, argv」として渡される。

したがって、この2つの変数を解析すればよいのであるが、複数のオプションをまとめて指定したり、オプションが引数を持つ場合の処理を適切に処理するのは、結構やっかいである。これはたとえば、「-x -v -q」というオプション指定を「-xvq」として指定したり、「-n 80」を「-n80」としても指定できるようにする場合が多いからである。

多くの処理系では、このようなオプション処理を簡単に行うために、getopt関数が用意されている。

この関数は、今回のプログラムでは、

```
getopt(argc, argv, "xgvc:w:n:s:l:o:")
```

というように用いられている。argc、argvが、main関数に渡された変数で、第3引数がオプションの仕様を示している。

「:」は、その左にあるオプション文字が引数を持つことを示している。つまりここでは、x、g、v、cがオプショ

```
=============================================================  n=244702 s=4772
 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1
 s r p l o h m d t a v h r s b t a v h r o t a v y n y o l h r p h t a p r s;
 k l y v r a v y r a t r s a v k r b m k r b t r s a t a y t o d y y k c r a;
 m s y a h n h v r v o p l o b l m y d m c n h v y s n a s y a v g r h l a r;
 v h y r b d r s a k v h v c y v v y l o v y d y t a k m s y v h d o h l k y;
 b s h t a l a r s y y n b t d o l a v n h k h r z o l a l a v h s m l a v a;
 s h y h a l v m d y p r b v n x y v s v l a m v o s y v s v l a b v n x y v;
 m h y n y o b b v t l r m a l d x p l c t v n b l h v h y h v c r s a r b d;
 n y n p l v n y h l a h v h y v h n t|y|v h c h y h m x l m l v m o l k v a;
 l a v h v t o d y l b b v h o r t a[x+c+r-y-r-s-a-x-c-v-r]h m s s n l s m s;
 s n h d b a d o m b k y h l a h v h y|x|l s y h o r c h t a m g v m h y n p;
 l k v y t v c m v v y t p s m v v y t|q|x v v t r m s m t r m s v k y h l a;
 l a r s y l k v m y d b o t y b m m y|r|c m c r a m k a y c v m h k y h l a;
 d o m y n s y p l o v t m v m y n b a|b|m t l q s v h s a h t a v a s y a h;
 r s l k y h l a h v h y r x b m b y k|y|v l y n b m y n h k h v s g n v l x;
 v b a l o v t m v y a l m y n b v m y|n|b l o t v b a v t m v y a l a t x k;
 v k a b b h t a r v r a k n a c t r t s o v k y p l a r g s k t m d a y r p;
 k b v k b b l l k b k y n b v h t a m v y h k v c m y k n a r s a l k k v l;
 b v y s x r v c v m y h l a y a r m a v b v z o v r v c o s p a v d y t l z;
```

図1 「$ xbible -w78x18 -o-1,+2 -n254245 -s4772 ycxqrbyn xcryrsaxcvr」の実行結果

ン引数を持たず、c、w、n、s、l、oがオプション引数を持つことを示している。

xbibleでは、この関数を**リスト1**のように用いている。getopt関数の戻り値は、第3引数で指定したオプションの文字である。また、getopt関数の戻り値が−1の場合は、すべての解析を終了したことを意味する。

したがって、whileループを使い、getoptが−1になるまで、それぞれのオプションを解析すればよいことになる。

オプション引数がある場合、そのオプション引数は、optargという変数に渡されることになっている。したがって、xbibleにおいて表示する数を示す−cオプションの処理は、

```
case 'c':
    count = atoi(optarg);
    break;
```

となる。ここで、atoiは文字（ASCII）から整数

▼リスト1　xbibleで使用するgetopt関数

```
...
#include <unistd.h> /* getopt() */
...
int
main(int argc, char **argv)
{
        extern char     *optarg;
        extern int      optind;
        word_t          **word_list;
        int             c;
        int             count = -1;

        while ((c = getopt(argc, argv, "xgvc:w:n:s:l:o:")) != -1){
                switch (c) {
                case 'x': opt_x = true; break;
                case 'g': opt_g = true; break;
                case 'v': opt_v = true; break;
                case 'c':
                        count = atoi(optarg);
                        break;
                case 'w':
                        setup_range(optarg, "x",  &opt_w_col, &opt_w_row );
                        break;
                case 'n':
                        setup_range(optarg, "+-", &opt_n_min, &opt_n_max );
                        break;
                case 's':
                        setup_range(optarg, "+-", &opt_s_min, &opt_s_max );
                        break;
                case 'l':
                        setup_range(optarg, ":", &opt_l_div, &opt_l_add );
                        break;
                case 'o':
                        setup_range(optarg, ",", &opt_o_col, &opt_o_row );
                        break;
                case '?':
                default:
                        usage();
                }
        }
        opt_c = ( count != -1 ) ? count : ( opt_x ? 0 : 1 );
        argc -= optind;
        argv += optind;
        word_list = get_word_list( argc, argv );
        all_search( opt_n_min, opt_n_max, word_list );
        return 0;
}
```

（Integer）へ変換するライブラリである。

whileループによって、すべてのコマンドの処理が終わると、getoptによって処理された数がoptindに保存されている。したがって、

```
argc -= optind;
argv += optind;
```

を行うことにより、オプション以外の残りの引数の数と、そのポインタを得ることができる。

次号の内容とプログラムの実行

今月号では、オプションの処理だけで話が終わってしまったが、来月号では引き続いて文字列検索部分などの解説を行う予定である。

なお、付録CD-ROMにはxbibleのソースコードを収録してある。実行するためには、まず適当なディレクトリにソースをコピーし、chmodコマンドでmk-dataを実行可能にする。そして、wgetコマンドを使用できるように設定する。

インターネットに接続してmakeを実行すると、必要なファイルが自動的にダウンロードされて、コンパイルが行われ、暗号表が表示されるはずである。

また、空いているWindowsマシンのCPUパワーを活用するため、このソースはWindows上で動作するプログラムも生成できるように、FreeBSD上でWin32コードを生成可能なクロスコンパイラ「mingw32」でもコンパイルできるように書かれている。Windows用実行ファイルの詳細については、http://fujisawa.gr.jp/を参照していただきたい。

Column

聖書の電子データとヘブライ語

ご存じのとおり、聖書は、旧約聖書と新約聖書からなる。このなかで、聖書の暗号が存在するとされるのは、旧約聖書の最初の「創世記」、「出エジプト記」、「レビ記」、「民数記」、「申命記」の部分であり、これらがいわゆる「モーセ五書」と呼ばれるものだ。

この部分はまた、「トーラー」とも呼ばれ、ユダヤ教では非常に神聖なものとして扱われているという。トーラーを写本する場合には、一字一句絶対に間違わないようにすることが義務づけられているそうである。

トーラーはすべてヘブライ語で書かれており、その長さは304805文字である。インターネット上で入手できるものがないかといろいろ探したところ、http://shamash.org/の、

```
/tanach/tanach/text/transliterated.tanach/
```

というディレクトリに、ヘブライ語をアスキー文字列に変換したものがあることがわかった。ここにあるトーラー部分のそれぞれのファイル名は下記のとおりである。

創世記	**bereishit.gross**
出エジプト記	**shmot.gross**
レビ記	**vayikra.gross**
民数記	**bamidbar.gross**
申命記	**devarim.gross**

たとえば、上記のbereishit.grossというファイルの内容は、

```
GEN 001:001 BRASYT BRA ALHYm AT HSMYm VAT HARc.
GEN 001:002 VHARc HYTH THV VBHV VXSk OL-PNY THVm ...
GEN 001:003 VYAMR ALHYm YHY AVR VYHY-AVR.
```

となっている。

GENは創世記を意味し、その次の数字は行数を示している。このため、次のようなコマンドを用いることにより、本文以外の余計な文字を消去することができる。

```
cat 上記の5ファイル ¦ \
sed -e 's/^... ...:... //' -e 's/\.$//' ¦ \
tr -d '[\- \n]' ¦ tr '[+$A-Z]' '[tsa-z]' > bible.txt
```

このコマンドを実行すると、bible.txtという、トーラーの全文字数と正確に一致する304805バイトのファイルができあがる（cksumコマンドによるチェックサムは、3566479439となる）。

このファイルをEmacsなどで開き、Emacsのウィンドウ幅を適当に変更すれば、プログラムを作成しなくても暗号を目視で発見できることになる（もちろん画面の幅以上の文字間隔で暗号化されたものを見るのは無理である）。

上記の例ではtrコマンドを用い、「+」を「t」に、そして「$」を「s」に変換している。その理由は、ここで使ったデータの場合、ヘブライ語で「テット」と呼ばれる文字が「+」に、「サメフ」と呼ばれる文字が「$」としてエンコーディングされていたためである。

また、オリジナルデータには大文字・小文字の区別があるが、ここではそれを無視し、すべて小文字に変換している。

なお、この連載では、ヘブライ語を記述するのに、上記ファイルで使われているエンコード方法を用いて、ヘブライ文字を英小文字で表記することにした。

ちなみに、ヘブライ文字は22文字しかなく、母音がないので、英語や日本語の固有名詞を綴る場合には注意が必要である。このあたりの詳細については、来月また解説してみたい。

特別講座

ステップアップC言語

複数の値を戻り値とする関数

■複数の値を戻り値とするには

C言語の関数は、1つの値しか戻すことができない。たとえば、

```
int add(int x, int y)
{
    return x + y;
}
```

という関数は、xとyの和を返すが、この関数で同時に差も計算して、その値を返したい場合はどうしたらよいだろうか？

■関数に変数のポインタを渡す

関数から複数の値を戻す場合には、変数のポインタを渡すことにより、親関数に計算した結果を返すことが可能になる。たとえば、

```
void add_sub( int x, int y,
        int *add, int *sub )
{
    *add = x + y;
    *sub = x - y;
}
```

というような関数を定義しておき、

```
int a = 2345;
int b = 1111;
int add_val
int sub_val;

add_sub(a, b, &add_val, &sub_val);
printf("add=%d sub=%d\n",
        add_val, sub_val)
```

というように呼び出すことにより、

```
add=3456 sub=1234
```

という表示が行われる。

■今回のプログラムでの実例

今回作成したxbibleでは、ウィンドウ幅80、高さ25を指定するために、-w80x25というオプションを指定することにしている。

この処理は、本文の**リスト1**にあるように、

```
setup_range(optarg, "x",
        &opt_w_col, &opt_w_row );
```

というようにして行われる。

このsetup_rangeという関数では、optargに保存されている、"80x25"という文字列から「x」という文字を区切りとして、80と25という数字を切り分け、それをopt_w_colとopt_w_rowという変数にセットしている。

そして、setup_range関数側では、

```
setup_range(char *arg, char *mode, int
          *ans_min, int *ans_max )
```

と定義され、

```
*ans_min = ...
*ans_max = ...
```

として、この関数を呼んだ側に解析結果、つまり80と25を戻している。

■読みやすい書き方

Perl言語やLISP系の言語では、

```
( $add, $sub) = add_sub($x, $y)
```

という見やすい形でのプログラミングが行えるが、C言語では前述のように、

```
void
add_sub(int x, int y,
        int *add, int *sub)
```

というように、どれが入力で、どれが出力なのか、よくわからない書き方しかできない。

また、次のように返す値を途中で参照する場合には、

```
*add = x + y;
*sub = x - y;
*mul = *add * *sub;
```

というように、「＊」だらけになり、あとで見てよくわからないプログラムになってしまう。

このような場合は、戻り値であることを明示するために、「answer」などが付いた変数名にして、

```
void add_sub( int x, int y,
              int *answer_add,
              int *answer_sub,
              int *answer_mul )
```

という宣言にするとよいだろう。さらに、

```
int add;
int sub;
int mul;

add = x + y;
sub = x - y;
mul = add * sub;
```

というような書き方をして、関数の最後でまとめて、

```
*answer_add = add;
*answer_sub = sub;
*answer_mul = mul;
```

というように書くことにより、大幅にプログラムを見やすくすることができるのだ。

新連載

Perlスクリプト入門

すぐにできる実践Perl

テキスト処理やCGIにおいて、Perlスクリプトは欠かせません。今回から始まるこの連載では、実践的なPerlスクリプトを解説していくことにします。第1回は変数と数値の計算、文字列と条件判断を説明します。

第1回　Perlでテキストデータ加工

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

DTPやWeb関連の作業において、テキストデータの加工をしなければならない場合は結構あるものです。1つのファイルを少々書きかえるくらいならもちろん手作業のほうが早い場合もありますし、エディタやワープロソフトの置換機能も便利なものです。しかし、たとえば、「データ中の電話番号だけを半角数字にする」などといった置換はどうしますか。しかも、対象となるテキストファイルが1000以上もあったとすると、通常は何日もかけて人手で処理しなければならないと考えるでしょう。

Perlを使えば、ほんの数行のプログラムで、1000ファイルであろうが、10000ファイルであろうが、このようなあいまいな置換や複雑な作業を一瞬のうちに処理してしまいます。とすれば、Perlを覚えることによって、何日もかかる労力を数分のものにしてしまうのですから、まさに知らないと大損してしまうともいえるでしょう。

PerlがC言語やBASICといった、ほかのプログラム言語と違うのは、テキストデータやテキストファイルを扱うのが得意だというところです。ファイルから1行のデータを読み込んだり、文字を置換したりするための命令が初めから用意されています。一方、グラフィックや、大きなアプリケーションを作るのには向いていません。ですから、まさに「テキスト加工」のためのプログラミング言語といえるでしょう。インターネットのホームページを構成するHTMLファイルもテキストファイルですから、結構Perlの活躍する場面が多くなっています。

また、C言語などのコンパイラとは違って、Perlはコンパイルせずに動かすことができます。プログラムを書いて即座に動かせる言語をインタプリタともいいますが、1行のプログラムからすぐに動かせることから、Perlはプログラムではなく、スクリプト言語といわれることもあります。通常、C言語で書かれたものはCプログラム、Perlで書かれたものはPerlスクリプトといわれます。プログラムとスクリプトにはそれほど厳密な違いはありませんが、コンパイラと呼ばれる本体によって実行できる形式にコンパイルされてから実行されるものをプログラム、Perlのようにインタプリタ本体が常に動いていて、1行ずつ読み込みながら実行されるものを一般にスクリプトと呼ぶようです。

この連載ではまず、実用になるテキストファイル加工スクリプトの作り方を学びます。これは、次のような課題を満たすスクリプトです。

○課題1　あるフォルダに保存されている複数のHTMLファイルがある。そのすべてのHTMLファイルの中の<h1>タグをタグに、</h1>タグをタグに置きかえる。ただし、<h1>は<H1>と書かれているかもしれない。

```
<h1>C言語やBASIC</h1>といった……
                ↓
<font size=6>C言語やBASIC</font>といった……
```

○課題2　あるフォルダに保存されている複数のHTMLファイルがある。このファイルの数字部分は、全角と半角が混在している。この数字をすべて半角に統一する。

平成8年3月25日……
↓
平成8年3月25日……

さてどうでしょう、以上2つの課題を今、あなたが行わなければならないとすると、どのような方法を思いつきますか。しかも、対象となるファイルが1000ファイルくらいあったら……気が遠くなるか、もうお手上げでしょう。ファイルの大きさにもよりますが、1ファイルを、最初から見ていって、該当箇所があれば修正して……。1ファイル15分かかるとして、1日8時間フルに作業して、1000ファイル完了するのになんと、31日と少しかかってしまう計算になります。また、そんな単純作業を続けて、神経がもつかどうか……。

コンピュータだって、結局は人手と同じ作業を行っているにすぎないのですが、彼らはどれだけ単純作業を課せられても、まったく疲れを知りません。不平不満もいいません（かえって単純でないと時々怒って止まります）。そしてなによりも、この手の作業をやらせれば、人間とは比べ物にならないくらいの（数十万倍以上の）速さで処理するのです。とはいえ、何をどうやって行うかの手順だけは、あらかじめ知らせてやらなければなりませんから、それをPerlスクリプトで書いてコンピュータに渡せば、あとはボタンひとつで「エイッ」とばかりに実行させるだけです。ほんの数分以内に、コーヒーを飲むまもなくすべての作業が終わってしまいます。

スクリプトを書くのに、1～2時間かかったとしても、手作業なら前述のように約1カ月分ですから、約1カ月分の仕事を1時間＋処理数分で終えることになります。あなたの時給は数十万円ですね。

さあ、どうしてもPerlをマスターしたくなってきたのではないでしょうか？　それではまず、先の2つの課題をクリアするために必要なPerlスクリプトを学びましょう。それでも、Perlの持つすべての能力のほんの一部しか使わないのです。

Perlスクリプトの書き方

では実際にPerlスクリプトの例を見てみましょう。

```
@list = (24,35,11,60);
# 合計を計算
$sum = $list[0] + $list[1] + $list[2] + $list[3];
# 平均点を計算
$ave = $sum / 4;
for ($i=0; $i<4; $i++){
    if ( $list[$i] < $ave ){
        print "平均点以下です¥n";
    }
    else{
        print "合格です¥n";
    }
}
```

このようなスクリプトをファイルとして作成し、

```
$ perl <ファイル名>
```

として実行することで、スクリプトが実行されます。

Perlスクリプトの基本的な記述ルールは、

- **原則として文（命令）の最後には、必ずセミコロン（;）をつけなければならない**
- **{ }で囲まれた部分はブロックといい、if、for文などの後には必要**
- **ブロック{ }の後には、;はつける必要がない**

C言語をご存じの方は、PerlスクリプトがC言語に似ていることにお気づきでしょう。C言語との違いとしては、

- **#の後はコメントとして、行末まで無視される**
- **if、for文の後に続く文が1行であっても{ }が必要**
- **変数名には必ず$（または@、%）をつける**

そのほか、細部においての決まりは、それぞれの節で解説します。

変数（スカラー変数）

1つのデータを入れるための変数は、単純変数またはスカラー変数と呼ばれ、変数名の先頭に必ず$をつけます。名前にはアルファベットと数字が使えますが、日本語は使えません。

```
$a、$A、$a1、$var、$kingaku……
```

大文字、小文字は区別されるため、$aと$Aとは別のものです。

そのほかの変数として配列変数、ハッシュ変数などがあります。

Perlで扱えるデータには、文字列（テキスト）や整数、実数などがありますが、それらを格納する変数の型に、文字列型や数値型などの区別はありません。変数には、どんなデータでも入れることができます。

```
$var = 35;                  # 整数
$var = 3.1415;              # 実数
$var = "こんにちは";        # 文字列
```

C言語などとは違って、Perlでは変数を使う前に「int a;」などと宣言する必要がありません。いつでも、どのような変数を使ってもかまわないのですが、そのために、$kingakuのつもりで、$kungakuなどと書いていても、コンピュータはエラーを出してくれません。あくまでも別の変数として扱われるだけです。十分注意してください。

次は重大な間違いの例です。

```
$kosu = 12;
$kingaku = 1500;            # これは正しい

$goukei = $kosu * $kungaku;# $kingakuのつもりが……
```

この例では、$kingakuを掛けるつもりが、$kungakuを掛けてしまい、意図しない結果となるでしょう。

分岐を実現するif制御構造

数値の計算には**表1**にあげた6種類の演算があります。

C言語やBASICといったほかの言語とほとんど同じですが、ほかのプログラム言語をご存じの方は、割り算に気をつけてください。Perlの変数には、整数や小数、文字といった区別がないため、「$a = 1、$b = 3」のとき、「$c = $a / $b」とすると、$cは「0.333333333」になります。

■二項代入演算子

たとえば、ある変数に10を足したり、2を掛けたりして、またその変数に答を戻したい場合は、

```
$a = $a + 10;
$a = $a * 2;
```

とも書けますが、もっと簡単に、

```
$a += 10;
$a *= 2;
```

と書くことができます。演算と代入（=）を同時に行う演算で、これを二項代入演算子と呼びます。二項代入演算子には**表2**のものがあります。このように簡単に書ける場合は、できるだけこの書き方をしましょう。

また、1を足す、1を引く、という処理はよく使われるので、この場合に限って、もっと簡単な書き方ができます。

1を足す場合は、

```
$a++;                       #（$a += 1 と同じ）
```

1を引く場合は、

```
$a--;                       #（$a -= 1 と同じ）
```

種類	記述方法	意味
足し算	$a + $b	$aと$bを足す
引き算	$a − $b	$aから$bを引く
掛け算	$a ＊ $b	$aと$bを掛ける
割り算	$a / $b	$aを$bで割る
べき乗	$a ＊＊ $b	$aの$b乗
余り	$a % $b	$aを$bで割った余り

表1　数値の計算

種類	記述方法	別の記述	意味
足し算	$a += 2	$a = $a + 2と同じ	$a + 2を$aに
引き算	$a −= 2	$a = $a − 2と同じ	$a − 2を$aに
掛け算	$a ＊= 2	$a = $a ＊ 2と同じ	$a ＊ 2を$aに
割り算	$a /= 2	$a = $a / 2と同じ	$a / 2を$aに
べき乗	$a ＊＊= 2	$a = $a ＊＊ 2と同じ	$aの2乗を$aに
余り	$a %= 3	$a = $a % 3と同じ	$aを3で割った余りを$aに

表2　二項代入演算子

と書くことができます。こちらもできるだけこの書き方を使ってください。

文字列

Perlは文字列（テキスト）の処理を得意とするため、文字列を扱うためのさまざまな機能が用意されています。

■文字列の表し方

文字列はふつう、"と"（ダブルクォーテーション）で囲みます。

```
$a = "これは文字列です";
```

■特殊な文字

改行やタブなどの特殊文字は、\をつけたメタ文字と呼ばれる表現を使います（**表3**）。

次の例は、メタ文字を使った表現です。

```
$a = "これは文字列です¥n";  # 文末で改行する
$a = "価格は¥¥50,000";     # 「価格は¥50,000」
$a = "¥x88¥xa4";           # 16進数。シフトJISの
                           # 88a4「愛」という漢字
```

これらは、出力用のprint文で画面に出力することができます。

```
print $a;
```

どのように出力されるか、print文を使って試してみましょう。

■変数展開

文字列の" "の中には変数を含めることもできます（変数展開）。たとえば、

```
$a = 5000;
print "価格は$a円です";
```

メタ文字	意味
\n	改行
\t	タブ
\x2a	文字コード（16進数）
\\	\そのもの

表3　メタ文字

と記述すると、

```
価格は5000円です
```

と表示されます。また、

```
$a = "山田";
$b="太郎";
$name = "$a  $b";
```

とすると、

$nameは「山田　太郎」になります。

それでは練習を兼ね、次のスクリプトをいろいろ書き換えて、変数展開を実感してください。

```
$name = "花子";
$weather = "晴れ";
print "こんにちは$nameさん、今日は$weatherですね¥n";
$greet = "こんにちは$nameさん、今日は$weatherですね¥n";
print $greet;
```

■シングルクォーテーション

$nameなどをそのまま出力したいとき、つまり変数展開をさせたくないときは、" "のかわりに、' '（シングルクォーテーション）を使います。

```
$name = "山田"
print "こんにちは$nameさん¥n";
print 'こんにちは$nameさん¥n';
```

このとき、ダブルクォーテーションでは、「こんにちは山田さん」と表示されますが、シングルクォーテーションでは、「こんにちは$name さん \n」と表示されます。つまり、' ' の間に書かれた文字がまったくそのまま出力されるのです。場合に応じて、" と ' を使い分けましょう。

■文字列の連結

変数展開での例にもあったように、文字列と文字列をつなげるには、それらを" "で囲ってやります。

```
$a = "こんにちは";
$b = "さようなら";
```

のとき、

```
$c = "$a$b";
```

とすれば、$cは"こんにちはさようなら"になります。

また、足し算やひき算のように、

```
$c = $a.$b;
```

としても、$aと$bの文字列をつなげることができます。これでも、$cは"こんにちはさようなら"になります。この場合、ピリオド（.）が連結のための演算子になります。

■大文字、小文字にするメタ文字

少し特殊ですが、文字列の中のアルファベットを大文字、小文字に変換するメタ文字があります（**表4**）。

HTMLのタグなどを処理する場合、大文字、小文字が混在しているので、それらを一括して大文字や小文字に統一する場合に便利です。

```
$a = "Yamada Taro";
print "¥U$a";     # YAMADA TAROと出力される
print "¥L$a";     # yamada taroと出力される
```

■出力コマンド　print文

これまで何度も出てきましたが、データを出力（表示）するためにはprint文を使います。

また、print文は次のような使い方があります。

```
$a = "メロン";
$b = "500";
                              # 出力結果
print "こんにちは";           # こんにちは
print $a;                     # メロン
print "$b円です¥n";           # 500円です
print "$aは$b円です¥n";       # メロンは500円です
print  $a,$b;                 # メロン500
print  $a,"は",$b,"円です";   # メロンは500円です
print "あ" x 4;               # あああああ
```

最後の3つはちょっと特殊です。

カンマ（,）は、変数や文字列を連結して出力するときに使います。

先に説明した「文字列の連結」と同じ効果ですが、" "などで全部を囲んで文字列を連結するときは、変数展開のことを考慮しなければなりません。単にならべるだけなら、print文に限って、カンマを使うことができます。

また、「x」は同じ変数や文字列を複数回くりかえして出力するときに使います。

いずれも便利です。いろいろ試してみてマスターしましょう。

これで、Perlの基本としてもっとも重要な「変数」「文字列」について学びました。ここまでの知識は、とにかくどんなスクリプトを書くにも必要とされる部分なので、あまり面白いものではありませんが、何度も自分でスクリプトを書いて実行してみて、十分にマスターしておいてください。「変数」も「文字列」も、テキストデータの処理には、とても大事なものです。

条件判断（if文）

if文は、「（条件が）～ならば～する」という、条件判断を行うためのコマンドです。条件判断は、条件によってスクリプトの流れを変えることができる重要なものです。コンピュータに条件を判断させることによって、複雑な処理を実現することができるようになるのです。条件判断がなければ、スクリプトは単純な処理だけしか実行することができません。

それでは、いろいろな条件判断を実例で見てみましょう。

■もし～ならば～する

もし、$aが10なら、$aを出力します。

```
if ( $a == 10 )   {  print $a;  }
```

■もし～ならば～する、そうでなければ～する

もし、$aが10なら、$aを、そうでないなら$bを出力します。

```
if ( $a == 10 )   {  print $a;  }
else              {  print $b;  }
```

メタ文字	意味
\U	それ以降、\Eまでを大文字にする
\L	それ以降、\Eまでを小文字にする
\E	\U、\Lの効果を終了する（省略可能）

表4　大文字・小文字にするメタ文字

Perlでは、改行などは自由に行っても、スクリプトとして影響はありません。このため、以下のように書くこともできます。特に、{ } の中身が2行以上の場合は、以下のように書くほうが良いでしょう。

```
if ( $a == 10 ) {
   print $a;
}
else {
   print $b;
}
```

■もし～ならば～、またもし～ならば～、そうでなければ～する

もし、$aが20以上なら$aを、$aが10以上20未満なら$bを、そうでないなら$cを出力します。

```
if ( $a >=20 )          {  print $a;  }
elsif ( $a >= 10 )      {  print $b;  }
else                    {  print $c;  }
```

C言語などでは、elseifと書きますが、Perlではelsifと書きますので注意してください。

if文は特に重要ですので、ここでもう一度ポイントをおさらいしておきましょう。

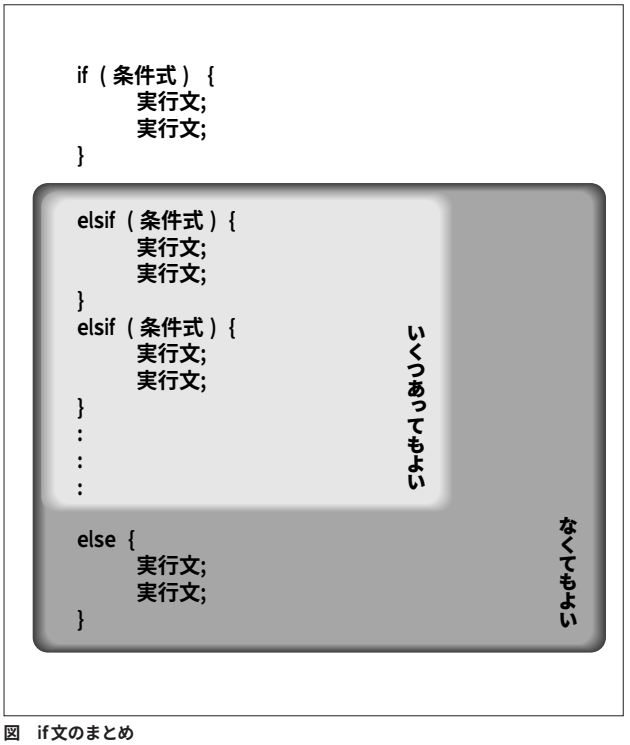


図　if文のまとめ

- **if、elsif、elseの後のブロックは、中の実行文が1つであっても、必ず{ }で囲む。**
- **elsif、elseの部分は条件が1つの場合はなくてもよい。**
- **elsifの部分は、場合によって何度書いてもよい（それぞれちがう条件で）。**

■もし～でなければ～（unless）

「もし～でなければ～」の場合、ifのかわりにunlessを使うこともできます。

もし、$aが10でないなら、$aを出力する。

```
unless ( $a == 10 ) {  print $a;  }
```

これは、

```
if ( $a != 10 )    {  print $a;  }
```

と同じ意味です（!=は、次項「条件式」参照）。

条件式と真偽

Perlに限らず、プログラム（スクリプト）での条件判断は、次のようにして行われます。

①条件を式で表す
②その式が「真」か「偽」かを判断する

「真」とは、その式が成り立つ（正しい）ということを意味し、「偽」は成り立たない（間違い）という意味です。

```
20 > 10                              ……真
$a = 13  のとき  $a < 20             ……真
$a=5  $b=3のとき  ($a + $b) == 9     ……偽
$a="あ"  $b="い"  のとき
  "$a$b" eq "あい"                   ……真
```

■条件式

条件を表す式は、**表5**のような比較演算子を使って作ります。

if文で条件式を使うと、次のようになります。

```
if ( $a >= $b ) { …… }
if ( $a eq "あいう" ) { …… }
```

■否定演算子

条件式の直前に！をつけると、その条件ではない（否定）を表すことになり、真偽が逆になります。

これは、〜ではないならというような場合に使用します。！を付けない場合と逆の処理になるというわけです。

```
$a = 10;
$b = 20;
```

のとき、

```
if ( $a < $b ) { …… }
```

は真になり、{ }内が実行されますが、

```
if ( !($a < $b) ) { …… }
```

は偽になります。もちろん、このような場合、

```
if ( $a >= $b )   { …… }
```

と書いても同じことなのですが、前者は、「$aが$bより小さくないなら」と読めますし、後者は、「$aが$b以上なら」と読めます。

意味はまったく同じですが、どちらの読み方をしたほうがそのプログラムの目的に合っているかを考えて、適切なほうを使うように心がけると、わかりやすいプログラムになります。

また、書籍に掲載されたものや、他人のスクリプトを読むこともあると思います。その場合は、どちらの書き方であっても理解できるように準備しておくべきです。

■条件の組み合わせ

実際の条件判断は、単純なひとつの条件式で済む場合だけではありません。「3以上8以下」といった条件の場合は、条件式を組み合わせて使います。

条件式の組み合わせには、&&（AND、かつ）あるいは‖（OR、または）を使います。

たとえば、$aが3以上8以下の場合の条件式は、「$aが3以上、かつ$aが8以下」ということですから、

```
3 <= $a && $a <= 8
```

となります。逆に、それ以外の場合は「$aが3より小さい、または$aが8より大きい」となりますから、

```
$a < 3 || 8 < $a
```

と書けばよいことになります。

また、&&、‖はいくつも連ねて書くことができます。

```
if ( $a == 3 || $a == 5 || $a == 7 || $a == 9 ){
    ……
}
```

この場合、$aが3、5、7、9の場合に、条件式が真となり、{ }内が実行されます。

■&&、‖の優先順位

&&は、‖よりも優先順位が高くなります。これは、算術演算の＊（掛け算）と、＋（足し算）との関係と同様です。算術演算の場合、「4に、3と5を足したものを掛ける」という計算は、

```
4 * 3 + 5
```

と書くと、＊が優先され、「4と3を掛けたものに5を足す」という計算になってしまうので正しくは、

```
4 * ( 3 + 5 )
```

になります。&&、‖も同様で、&&が＊にあたり、‖が＋にあたります。

「$aが3で、なおかつ$bが6か8」という場合は、

```
if ( $a == 3 && $b == 6 || $b == 8 )
```

と書いてはいけません。これだと「$aが3で$bが6、また

演算子	意味
==	等しい
!=	等しくない
>	より大きい
<	より小さい
>=	より大きいか等しい（以上）
<=	より小さいか等しい（以下）
eq	同じ（文字列のとき）
ne	同じでない（文字列のとき）

表5　比較演算子

は$bが8（$aはなんでもよい）」というように、&&のほうが優先されて、「$a == 3 && $b == 6」と「$b == 8」に分けられてしまいます。

‖のほうを優先したければ、

```
if ( $a == 3 && ($b == 6 || $b == 8) )
```

と、‖のほうを( )で囲んでください。また、たとえばこれに否定が加わって、「$aが3で、なおかつ$bが6でも8でもない」という場合は、

```
if ( $a == 3 && !($b == 6 || $b == 8) )
```

と書きます、間違っても()を忘れて、

```
if ( $a == 3 && !$b == 6 || $b == 8 )
```

などとは書かないでください。とんでもない結果になってしまいます。

条件式と式の真偽は、慣れるまではややこしく感じるものかもしれません。しかし、これらは「判断」という、コンピュータプログラムの根幹をなすものです。そう性急に、慣れなければならないと思い込む必要はありませんから、実際のプログラミングで条件式を作る場合は、いつもこの連載や入門書を参考にして、冷静に、正しく式を書くよう心がけてください。コツは、常に言葉にして読みかえることです。

条件式の注意点

条件式は、書き方を間違えると意図しない結果となってしまうことがよくあります。ここでは、真偽が評価される仕組みと、いくつか条件式の注意点を説明します。

■真と偽の値

コンピュータは、電子計算機といわれるように、処理としては数値しか扱えません。そのため、ここで表現している真（正しい）、偽（間違い）ということも、実は数値に置き換えて処理しています。

真と偽はそれぞれ、次のような数値として、処理されています。

0は偽
0以外の数値はすべて真（たとえば1）

真、偽は、True、Falseと表現される場合もあります。覚えておきましょう。

■特殊な条件式

以下の例は条件式といえるでしょうか。その場合、条件式は真偽のどちらになるでしょう。

①if ($a)
②if ("あいう")
③if ("")

この3例は、いずれも比較演算子（==、<など）がありません。こういう場合は、その変数の中身を見て、直接真偽の判断をします。ですから、これでも条件式なのです。

①の場合、$aの中身を判断しますから、$aが0であれば結果は偽になり、それ以外の値ならば結果は真になります。

②の場合は数値ではありませんが、文字列の場合は文字が1文字でもあれば真になります。

ですから、③は偽です。

■==の注意点

次のif文に注意してください。

```
if ( $a = $b ) { …… }
```

$a == $bと書くところを、間違って$a = $bと書いています。

よくある間違いですが、これは、コンピュータにとってエラーにはならず、平然と実行されてしまいます。しかし、実行結果は予想と反したものになるはずです。

なぜ平然と実行されてしまうのでしょうか？

この式はあくまでも$aに$bの値を代入する式なので、まず$a = $bという代入を行い、その後、等しくなった$aと$bの値が0かどうかを判断してしまいます。ですから、$bの値が0（または空の文字列）であれば偽を、そうでなければ真と判断するのです。前項「特殊な条件式」の①を思い出してください。

このとき$aと$bが等しいかどうかは判断しません。それどころか、$aには$bの値が強制的に代入されてしまいます。

たったひとつの＝の書き忘れというこの間違いは、プロのプログラマーでも起こしてしまう間違いなので、十分注意しましょう。

[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、if制御構造で条件式による分岐を行うためのコマンドtest（あるいは[ ]）の書き方や演算子、複数の条件式の組み合わせ方などを説明したのち、ファイルの種類や属性を日本語で表示するシェルスクリプトを作成する。

第5回　if制御構造を使った分岐処理（その2）

文：大池浩一
Text:Koichi Oike

前回に引き続いて、シェルスクリプトで分岐処理を行う際に使われるif制御構造について紹介する。復習を兼ねて、前回学んだ基本的な部分を押さえておこう。

シェルスクリプトのif制御構造は、次のように条件部に指定したコマンドの終了状態を示す「終了ステータス」の値に基づいて分岐処理を行う。

```
if コマンドA; then
  コマンドAが正常終了した場合に実行する内容
elif コマンドB; then
  コマンドBが正常終了した場合に実行する内容
else
  コマンドBが異常終了した場合に実行する内容
fi
```

これによって、「正常終了した場合だけ処理を続行する」とか「異常終了した場合にはほかのコマンドを実行する」、「それも異常終了した場合にはエラーメッセージを表示する」といった処理が可能になるわけだ。

しかし、シェルスクリプトに求められる分岐処理はこのような単純なものだけではない。C言語などのプログラムの経験があれば、もっと一般的な条件式の評価による分岐を行いたくなるはずだ。

たとえば、「シェル変数の値が特定の文字列と一致する」とか「シェル変数の内容が空でない」ことを調べるといった文字列の比較、あるいは「指定したファイルが実際に存在する」とか「指定したファイルの内容を読み込める」といったファイル属性のチェックなどが考えられる。

こうした条件分岐を行うために、UNIX系OSには条件式を評価して終了ステータスに反映するコマンドtestと[が用意されている。これらをif制御構造と組み合わせることで、単なるコマンドの終了結果にとどまらず、もっと一般的な条件式による分岐が可能になる。

世の中で使われているシェルスクリプトの大半は、if制御構造とtest（あるいは[ ]）の組み合わせが含まれるといっても過言ではないだろう。

testや[を使って分岐を行う

bashのベースとなったBourneシェル（Bシェル）では、testや[は、シェル本体とは独立した外部コマンドとして用意されている。Linuxのディストリビューションでも、/usr/binにtestと[が置かれているはずだ。

一方、Linuxの標準シェルであるbashの場合、どちらもシェル内部で実現される組み込みコマンドになっている。使い方は外部コマンドと同じだ。条件式による分岐は頻繁に使われるので、外部コマンドより高速に実行可能な組み込みコマンドに取り入れられたのだ。

なお、組み込みコマンドは外部コマンドより優先して使われるので、bashのコマンドラインやスクリプトでtestや

[を普通に実行する限り、/usr/bin にある外部コマンドはまったく利用されない。

以下では、

- **testや[の記述のしかた**
- **文字列の比較に使われる演算子**
- **ファイル属性のチェックに使われる演算子**
- **複数の条件を組み合わせる方法**

などについて説明したあと、if制御構造と [を組み合わせてたスクリプトの作成を行う。

●testと[をif制御構造で利用する

条件式の評価を行うtestと [は、どちらも同じ機能を持っているが、見た目の記述方法は少々異なる。なお、今回はif制御構造との組み合わせに限って説明するが、testと [は、繰り返し処理を行うwhile、until制御構造の条件部でも使われる。これらについては次回以降に紹介することにしよう。

(1)testを使った条件式の評価

testを利用して条件式の評価を行う場合、if制御構造は次のように書ける。

```
if test 条件式; then
  条件式が成立した場合に実行する内容
fi
```

「条件式」の中では、文字列の比較やファイル属性のチェックなどさまざまな演算子が利用できる。演算子の詳細については後で取り上げることにして、ここではtestをif制御構造の条件部で実行し、条件式をtestの引数として指定するという点を押さえよう。

条件式はtestによって評価され、真（条件が成立した）ならば正常終了を示す終了ステータス「0」、逆に偽（条件が成立しなかった）なら、異常終了を示す終了ステータス「1」が返される。

if制御構造はこの終了ステータスに基づいて分岐を行うので、条件が成立した場合にだけthenとfiの間の内容が実行されるわけだ。もちろん、前回紹介したelseを使って、条件が成立しなかった場合の処理を行ったり、elifを使って、条件不成立の場合にさらに別の分岐を行ったりしてもいい。

たとえば、指定した文字列の長さが1文字以上ある（すなわち空文字列ではない）場合に真となる-n演算子を利用し、シェル変数hogeの内容の有無によって分岐するスクリプトは以下のようなる。

図　if制御構造とtestを組み合わせる

```
#!/bin/sh
if test -n "$hoge"; then
  echo OK
else
  echo NG
fi
```

シェル変数hogeに何らかの文字列が設定されている場合は、条件式「-n "$hoge"」の評価が真になるので、then以下の「echo OK」が実行され、端末画面に「OK」と表示される。一方、シェル変数が定義されていないか、変数の内容が空文字列の場合は、条件式の評価が偽になるので、else以下の「echo NG」が実行され、端末画面に「NG」と表示される（**図**）。

(2)[を使った条件式の評価

一方、[を利用して条件式の評価を行う場合、if制御構造は以下のように書ける。

```
if [ 条件式 ]; then
  条件式が成立した場合に実行する内容
fi
```

「条件式」の指定方法はtestとまったく同じだ。たとえば、さきほどの例を [を使って書きなおすと、

```
#!/bin/sh
if [ -n "$hoge" ]; then
  echo OK
else
  echo NG
fi
```

というスクリプトになる。

条件式の両側が大カッコで囲まれているため、まるでif制御構造の機能の一部であるかのように見えるのではないだろうか。見た目にもすっきりしているので条件式が目立つ。これが、[という奇妙な名前のコマンドがわざわざ用意されている理由で、世の中で使われているスクリプトも [を使って書かれているものが多い。以下の説明でも見やすさを重視して、[による記述を採用しよう。

ただし、[は実際にはif制御構造の一部ではなく、コマンドであるいうことを気に留めておこう。たとえば、[と条件部の間には1個以上のスペースが必要だ。

このことを忘れて、

```
if [-n "$hoge"]; then
```

のように両者をくっつけて書くと、if制御構造の条件部で、[ではなく[-n というコマンドが実行される。普通はこのようなコマンドは存在していないので、

```
bash: [-n command not found
```

などのエラーメッセージが表示される。このほか、条件部と] の間のスペースや、] 自体を忘れてもエラーになるので気をつけよう。

要点をまとめると、

- **testや [をif制御構造の条件部で実行する。**
- **条件式はtestや [の引数として指定する。**
- **testと [は見た目が違うだけで機能は同じ（スクリプトでは [が使われることが多い）。**
- **[の後、] の前には1個以上のスペースが必要。**

ということになる。

●文字列の比較を行う

条件式の中で使われる演算子について見ていこう。まずは、文字列の比較を行う演算子だ（**表1**）。ほとんどの場合、これらはシェル変数と文字列の比較や、シェル変数の内容が空文字列ではないか調べるなど、シェル変数を変数展開した文字列と一緒に使われる。

これらの演算子は、

- **両側に文字列が必要な二項演算子（binary）**
- **後ろに文字列が必要な単項演算子（unary）**

に大別される。

最初の例で登場した-n演算子は単項演算子なので、「-n

文字列1 = 文字列2	文字列1が文字列2と一致する場合に真
文字列1 != 文字列2	文字列1が文字列2と一致しない場合に真
-z 文字列	文字列の長さが0（空文字列）の場合に真
-n 文字列	文字列の長さが1以上の（空文字列でない）場合に真
文字列	（同上）

表1　文字列の比較を行う演算子

文字列」のように後ろに文字列が1つだけ必要だ。一方、2つの文字列が一致する場合に真となる＝演算子は二項演算子なので、「文字列1 ＝ 文字列2」のように、両側に文字列を指定する。

たとえば、環境変数hogeに「Oike」と設定されていたら「それは私です」と表示するスクリプトは、以下のように書けばいい。

```
#!/bin/sh
if [ "$hoge" = Oike ]; then
  echo それは私です
fi
```

まず、if制御構造の条件部にある[が実行されて、条件式「"$hoge" ＝ Oike」が評価される。そして、シェル変数hogeの内容と文字列「Oike」が一致する場合には真、一致しない場合には偽となる。if制御構造はその結果を踏まえて分岐を行い、一致する場合はechoで「それは私です。」と端末画面に表示、一致しない場合は処理をスキップして終了する。

以下では、シェルスクリプトの初心者が間違えやすい点を3つ紹介することにしよう。

(1)演算子と文字列の間にはスペースが必要

単項演算子では後ろの文字列との間、二項演算子では前後の文字列との間に、1個以上のスペースを入れる必要がある。さもないと、演算子も含めた1つの文字列と見なされてしまうからだ。

条件式に文字列のみを指定した場合、文字列の長さが1以上あれば評価が真になる。このように、本来の演算子とは別の基準で評価が行われるため、得られる結果も同然異なってしまう。たとえば、

```
#!/bin/sh
if [ "$hoge"=Oike ]; then
  echo それは私です
fi
```

のように、「＝」と両側の文字列をくっつけて書いてしまったとしよう。

この場合、ひとつの文字列「"$hoge"=Oike」が条件式と見なされ、文字列の長さが評価される。後半の「=Oike」により条件式の評価は常に真になるため、シェル変数hogeの内容が「Oike」以外の文字列や空文字列であっても、常に「それは私です」と表示されてしまうのだ。

(2)文字列の一致は「＝＝」ではなく「＝」だ

C言語の条件式で値の一致を調べる場合には＝＝演算子が使われる。このため、C言語の経験がある人は、シェルスクリプトでも「＝＝」を使ってしまいがちだ。そのためかどうかは不明だが、bashの新しいバージョン（2.x）では「＝＝」を＝演算子と同様に利用できる。

ただし、bashの古いバージョン（1.x）や、Bourneシェルで使われる外部コマンドのtestや[では、「＝＝」を使うとエラーになる。つまり、起動するシェルによって動いたりエラーになったりするわけだ。さまざまな環境で動作するスクリプトを書く場合には気をつけよう。

(3)シェル変数の周囲は「"」で囲む

シェル変数を展開した文字列を条件式で使うときには、その周囲を必ず「"」（二重引用符）で囲む（クォーティングする）必要がある。さもないと、シェル変数の内容が空文字列だったり、空白文字を含んだりしている場合に誤動作やエラーの原因になるからだ。

例として、コマンドラインの第1引数が指定されたかどうか調べ、指定された場合は「最初の引数は<…>です」と端末画面に表示するスクリプトを作成してみよう。コマンドラインの第1引数は「$1」で参照できるので、-n演算子を使って次のように書ける。

```
#!/bin/sh
if [ -n "$1" ]; then
  echo "最初の引数は<$1>です"
fi
```

ここでもし、条件式中の「$1」をクォーティングしないで「-n $1」と書くとどうなるだろうか。結果が何も変化しないのは、空白文字を1つも含まない文字列を第1引数に指定した場合だけだ。

たとえば、引数を1つも指定しなかった場合は、条件式が「-n」だけになる。testや[では、後ろに文字列が指定されていない「-n」は、演算子ではなく文字列として扱われる。このため、条件式の評価は常に真になり、「最初の引数は<>です」と表示されてしまう。

また、空白文字を含む文字列を「'」で囲んで（たとえば「'hoge hoge'」など）、コマンドラインの第1引数で指定

した場合は、条件式の-n演算子の後ろに複数の文字列が並ぶ（「-n hoge hoge」など）ため、[がエラーメッセージを表示し、その時点でスクリプトが中断してしまう。

以上をまとめると、

- **演算子は、両側に文字列が必要な二項演算子と、後ろに必要な単項演算子に大別できる。**
- **演算子と文字列の間にはスペースが必要。**
- **文字列の一致は「=」で「==」ではない。**
- **シェル変数の周囲は必ず「"」で囲む。**

ということになる。

●ファイル属性のチェックを行う

既存のファイルの内容を読み込む、あるいは設定を追加するといったスクリプトを書くには、ファイルの存在チェックや、読み込み・書き込みが許可されているかといった属性のチェックが不可欠だ。さもないと、存在しないファイルを読み込もうとしたり、書き込み許可のないファイルに書き込もうとしてエラーを引き起こす。

testと[には、ファイルの存在やファイル属性を調べる演算子が豊富に用意されている（**表2**）。なお、このほかにも名前付きパイプやソケットかどうかを調べる演算子などが用意されている。詳細は、bashのマニュアル（man）の組み込みコマンドtestの説明や、外部コマンドtestのマニュアルを参照されたい。

ファイル属性関係の演算子のほとんどは単項演算子で、調べたいファイル名を演算子の後ろに1つだけ指定する。たとえば、ホームディレクトリに「.bashrc」という通常ファイルが存在するかどうか調べて、その内容を表示するスクリプトは以下のようになる。

```
#!/bin/sh
if [ -f ~/.bashrc ]; then
   cat ~/.bashrc
fi
```

-f演算子は、指定されたファイルがディレクトリや特殊ファイル（シンボリックリンク、デバイスファイルなど）ではなく、通常のファイルの場合に真となる。ファイルの存在チェックも兼ねているので、事前にファイルの存在チェックを-e演算子で行う必要はない。

同様にして、ディレクトリや特殊ファイルもチェック可能だ。たとえば、コマンドライン引数で指定したファイルがシンボリックリンクかどうか調べて表示するスクリプトは次のようになる。

```
#!/bin/sh
if [ -n "$1" ]; then
  if [ -L "$1" ]; then
    echo "$1はシンボリックリンクです"
  fi
fi
```

このスクリプトでは、2つのif制御構造がネスト（入れ子）している。外側のif制御構造では、コマンドラインの第1引数（$1で参照）が空文字列ではないことを、条件式「-n "$1"」で調べている。もし、コマンドラインで引数を1つも指定しなかった場合は、この条件式の評価が偽になるので、末尾のfiの直後まで処理がスキップされ、スクリ

-e ファイル	ファイルが存在すれば真
-d ファイル	ディレクトリであれば真
-f ファイル	通常ファイルであれば真
-L ファイル	シンボリックリンクであれば真
-b ファイル	ブロックデバイスであれば真
-c ファイル	キャラクタデバイスであれば真
-s ファイル	ファイルの内容が空でなければ真
-r ファイル	読み込みが許可されていれば真
-w ファイル	書き込みが許可されていれば真
-x ファイル	実行（調査）が許可されていれば真
-u ファイル	suidビットが設定されていれば真
-g ファイル	sgidビットが設定されていれば真
-O ファイル	あなたが所有者ならば真
-G ファイル	あなたのグループが所有していれば真
ファイル1 -nt ファイル2	ファイル1の変更時刻がファイル2より新しければ真
ファイル1 -ot ファイル2	ファイル1の変更時刻がファイル2より古ければ真

表2　ファイル属性を調べる演算子（抜粋）

プトはそのまま終了する。

一方、引数を1つでも指定した場合は条件式の評価が真になり、内側のif制御構造が実行される。こちらは、第1引数で指定したファイルがシンボリックリンクかどうかを第2の条件式「-L "$1"」で調べる。通常のファイルやディレクトリなどの場合は条件式の評価が偽になるので、内側のfiの後ろまで処理がスキップして終了する。逆に、第1引数で指定したファイルがシンボリックリンクの場合は、第2の条件式の評価が真になるので、第1引数で指定したファイル名などがechoで表示される。

この節での要点をまとめると、

・ファイルの存在や属性をチェックする演算子は、ほとんどが単項演算子で、後ろにファイル名を指定する。

ということになる。

●複数の条件を組み合わせる

前のスクリプトでは、2つの条件式がどちらも真となる「かつ」（AND）の組み合わせを、if制御構造のネストで実現した。この方法では、条件が増えるたびにネストが深くなってしまう。以下では、1つのif制御構造の条件部で論理演算子（**表3**）を使って複数の条件式を組み合わせる方法を説明しよう。

2つの条件式を「かつ」（AND）で組み合わせる-a演算子は、どちらの条件式の評価とも真の場合のみ全体の評価が真になる。たとえば、さきほどのスクリプトを-a演算子を使って書きなおすと、以下のようになる。

```
#!/bin/sh
if [ -n "$1" -a -L "$1" ]; then
  echo "$1はシンボリックリンクです"
fi
```

2つの条件式を1つの [で処理しているため、if制御構造のネストが解消された。なお、右の条件式「-L "$1"」が評価されるのは、左の条件式「-n "$1"」の評価が真の場合だけだ。つまり、第1引数が空文字列なら、シンボリックリンクかどうかはチェックされない。

一方、2つの条件式を「または」（OR）で組み合わせる-o演算子のほうは、少なくとも1つの条件式の評価が真であれば全体の評価が真になる。たとえば、ホームディレクトリに「.bash_profile」か「.profile」が通常のファイルとして存在するかチェックするには、

```
if [ -f ~/.bash_profile -o -f ~/.profile ]; then
```

と書けばいい。右の条件式「-f ~/.profile」が評価されるのは、左の条件式「-f ~/.bash_profile」の評価が偽の場合だけだ。つまり、「.bash_profile」がホームディレクトリに存在していれば、「.profile」に関するチェックはまったく行われない。

もし、組み合わせたい条件が3つ以上ある場合は、「条件式1 -a 条件式2 -a 条件式3 …」といった具合に並べて書ける。この場合も条件式は左から順に評価されるが、「\(」と「\)」で囲んで評価順を変更することも可能だ（使い方は数学の数式の「(」「)」と同様）。また、条件式の前に否定演算子の「！」を書くと、条件式の評価の真偽が反転する。

たとえば、/etc/shadowという通常のファイルが存在し、なおかつ読み込み不可である場合にだけメッセージを表示するスクリプトは次のようになる。

```
#!/bin/sh
if [ -f /etc/shadow -a ! -r /etc/shadow ]; then
  echo "読み込み不可の/etc/shadowが存在します"
fi
```

左の条件式「-f /etc/shadow」は、/etc/shadowが通常のファイルとして存在する場合に真、右の条件式「！ -r /etc/shadow」は、ファイルが読み込み不可の場合に真と評価される。その結果、「/etc/shadowが通常のファイルとして存在し、読み込み不可」の場合に真になる。

もし、左の条件式を省略して、

```
if [ ! -r /etc/shadow ]; then
```

と書いた場合はどのように動作するだろうか。-r演算子は、正確に書くと「ファイルが存在し、かつ読み込みが許可されていれば真」と評価される。つまり、/etc/shadowの存在チェックも含まれているのだ。このため、右の条件式「！ -r /etc/shadow」は、正確には「/etc/shadowが存在しないか、または読み込みが許可されていない」場合に真と評価される。つまり、/etc/shadowが存在しない場合にもメッセージが表示されてしまうわけだ。

以上をまとめると、

・複数の条件式を-a、-o演算子で組み合わせられる。
・評価は左の条件式から順に行われる。「\(」と「\)」で評価順序を変えることも可能。
・条件式の前に「！」を置くと評価の真偽が反転する。

となる。

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。今月はページ数の都合上「スクリプトを読む」はお休みし、「スクリプトを書く」では、

・指定したファイル（複数可）のファイルの種類や属性を日本語で表示する「zokusei」

を作成する。それぞれのファイルに対し、

・ファイルの存在チェック
・ファイルの種類（通常のファイル、ディレクトリ、デバイスファイル、シンボリックリンク、その他）の表示
・ファイル属性（読み込み、書き込み、実行）の表示

を行う。

今回説明したif制御構造と[の組み合わせに加え、コマンド置換やシェル変数、for制御構造など以前の連載で説明した内容を含んでいるので、不明な点がある場合は前回までの記事も参照してほしい。

●スクリプトzpkuseiを作成する

コマンドライン引数で指定したファイル名をひとつひとつ処理するには、本連載の第3回で紹介したfor制御構造を利用して、以下のように書けばいい。

```
for f in "$@"; do
  echo "$f"
  ファイルの存在チェック
  ファイルの種類チェック
  ファイルの属性チェック
done
```

まず、「"$@"」が個々のコマンドライン引数（ここではファイル名）を「"」で囲んだリストに置換され、最初の引数から順番にループ変数fに格納される。

繰り返す内容では、最初にファイル名がechoにより表示され、続いて3種類のチェック（および表示）が行われる。それぞれのチェック内容について見ていこう。

(1)ファイルの存在チェック

最初に、ループ変数fに格納されたファイル名のファイルがそもそも存在しているかどうかを調べる。この時点ではファイルの種類は問わないので、[の-e演算子を利用すればいい。条件式は「-e "$f"」と書けるから、以下のような構造になるはずだ。

```
if [ -e "$f" ]; then
  ファイルの種類チェック
  ファイルの属性チェック
else
  echo "  存在しません。"
fi
```

実際には、ファイルの種類と属性のチェックは20行以上あるので、存在しなかった場合の処理よりずっと長い。こうした場合は、！演算子を使って条件式の評価の真偽を反転し、短い処理をthen以下で行うといい。

```
if [ ! -e "$f" ]; then
  echo "  存在しません。"
else
  ファイルの種類チェック
  ファイルの属性チェック
fi
```

実際のスクリプト（**リスト**）では、繰り返し処理の残りの部分をスキップして、次の繰り返し処理に移る組み込みコマンドcontinueを使って、elseを省いている。

条件式1 -a 条件式2	条件式1と条件式2の両方の評価が真の場合のみ真
条件式1 -o 条件式2	条件式1と条件式2の少なくとも一方の評価が真ならば真
！条件式	条件式の評価が偽ならば真

表3　論理演算子

```
if [ ! -e "$f" ]; then
  echo "  存在しません。"
  continue
fi
ファイルの種類チェック
ファイルの属性チェック
```

(2)ファイルの種類チェック

ファイルの種類チェックは簡単だ。[の対応する演算子を、if／elif／else制御構造の条件部に記述して、ひとつずつ調べればいい。通常のファイルは-f、ディレクトリは-d、デバイスファイルは-bまたは-c、シンボリックリンクは-Lでチェックできる。

```
if [ -f "$f" ]; then
  通常のファイルの処理
elif [ -d "$f" ]; then
  ディレクトリの処理
elif [ -b "$f" -o -c "$f" ]; then
  デバイスファイルの処理
elif [ -L "$f" ]; then
  シンボリックリンクの処理
else
  その他のファイルの処理
fi
```

実際のスクリプト（リスト）では、ファイルの種類を表わす文字列をシェル変数filetypeに格納し、チェック終了後のechoで表示している（23行目）。

(3)ファイルの属性チェック

ファイルの属性チェックも [の演算子を利用して行う。読み込み可能かどうかは-r、書き込み可能かどうかは-w、実行（ディレクトリの場合は検索）可能かどうかは-xでチェックできる。

ファイルの種類と違うのは、1つのファイルに複数の属性が設定可能なことだ（たとえば「読み込み可で、かつ書き込み可」など）。このため、それぞれの属性ごとにif制御構造を記述する必要がある。

実際のスクリプト（リスト）では、ファイルの属性を表わす3つの文字列「読み込み・」「書き込み・」「実行」をシェル変数attrに順次追加している。なお、読み込みまたは書き込みが許可され、実行が許可されていないファイルは、attrの内容の末尾に余分な「・」があるので、34行目のコマンド置換内部で実行されるsedでこれを取り除いている。その後、attrの内容の有無に応じて、35～39行目でファイルの属性を表示する。

このほか、実際のスクリプト（リスト）では、コマンドライン引数を指定しなかった場合に、エラーメッセージを表示してスクリプトを終了する処理を追加している（2～5行目）。ここでは、コマンドライン引数の有無を、引数の個数が設定されるシステム変数「#」（$#で参照）の内容が「0」と一致するかどうかでチェックしている。

▼リスト　スクリプトzokusei

```
 1: #!/bin/sh
 2: if [ $# = 0 ]; then
 3:   echo "Usage: zokusei file..." > /dev/stderr
 4:   exit 1
 5: fi
 6: for f in "$@"; do
 7:   echo "$f"
 8:   if [ ! -e "$f" ]; then
 9:     echo "  存在しません。"
10:     continue
11:   fi
12:   if [ -f "$f" ]; then
13:     filetype="通常のファイル"
14:   elif [ -d "$f" ]; then
15:     filetype="ディレクトリ"
16:   elif [ -b "$f" -o -c "$f" ]; then
17:     filetype="デバイスファイル"
18:   elif [ -L "$f" ]; then
19:     filetype="シンボリックリンク"
20:   else
21:     filetype="その他のファイル"
22:   fi
23:   echo "  $filetypeです。"
24:   attr=""
25:   if [ -r "$f" ]; then
26:     attr="読み込み・"
27:   fi
28:   if [ -w "$f" ]; then
29:     attr="$attr書き込み・"
30:   fi
31:   if [ -x "$f" ]; then
32:     attr="$attr実行"
33:   fi
34:   attr=`echo "$attr" | sed 's/・$//'`
35:   if [ -n "$attr" ]; then
36:     echo "  $attrが許可されています。"
37:   else
38:     echo "  何も許可されていません。"
39:   fi
40: done
```

アジアの国にこんにちは！

Emacsはじめました

第12回　第二外国語なEmacs

今回は、中国語や韓国語を読んだり書いたりするお話です。国際感覚が豊かなEmacsは、1つのバッファにいろいろな国の言葉を取りまぜて書くことができます。アジアの隣人はどんな文字を使い、どのように入力するのか？　日本語と中国語を併記できるのはなぜ？　これまで知らなかったことが盛りだくさん。

文：佐々木太良
Text : Taroh Sasaki

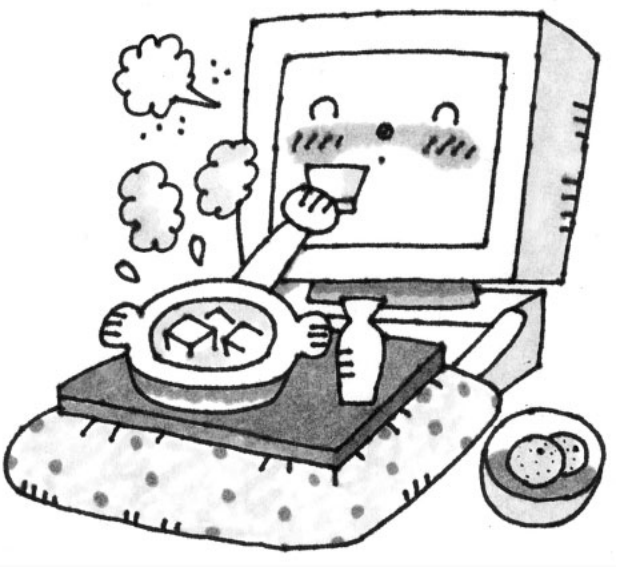


Illustration : Manami Kato

亜細亜でEmacs

14億7000万の東アジア言語圏の皆さんこんにちは（ちょっとでかすぎかな）。日本の読者のなかにも、2002年に開催されるワールドカップに備えてハングル語を勉強しようとか、あるいは中国がWTOに加入したのをきっかけに、仕事で中国語を使わないと……という方がいらっしゃるかもしれません。

日本語を含め、アジアの国々の多くは、英語のアルファベットとは大きく異なる文字を使っています。しかし、コンピュータのOSは英語圏で生まれ育ったものなので、英語または少数の文字で表せるヨーロッパの言語に比べると、文字が256種類以上あってマルチバイトで表現せざるをえないアジア諸国の言語処理環境はなにかと不利です。

この連載の第3回では、日本語の漢字をEmacsでどのように入力するかを見てきました。コンピュータで英語となんらかの言語の2カ国語を扱うことは、今日では普通のことになりました。

今回はさらに、3つ以上の言語コードをとり混ぜて使ってみましょう。この記事を読み終えるころには、Emacsこそ多言語を同時に取り扱えるほとんど唯一かつ最強のテキストエディタだということが実感できると思います。そう、Emacsは、アプリケーションとしても使えるのでしたよね。

多言語化の歴史と哲学

コンピュータが英語のアルファベットとわずかな記号だけしか表現できなかったのは、さほど古い時代ではありません。ここで、今日の多国語化の技術が開発されるまでの軌跡を辿ってみることにしましょう。

■古典的パソコン用OSでの多言語・2言語化

旧来、OSやウィンドウシステムを英語以外の言語に対応させるとき、英語版を開発したあとから各国の言語にそれぞれ対応させる、というアプローチがとられてきました。これを、localizationと呼びます（先頭と末尾の文字の間が10文字なので、l10nとも表記します）。日本語版のDOS/Vなどがそうでした。これらのOSで中国語や韓国語などの3番目の言語も使うには、別売のソフトウェア、フォント、IMEなどが必要でしたが、それらは専用のアプリケーションか限られたアプリケーションでしか使えないなどの問題がありました。

また、当時の非力なパソコンでもOSやアプリケーションが漢字を扱えるように、メーカー主導で日本のシフトJISや台湾・香港のBig5のような文字コードが普及しましたが、他の言語と組み合わせることは考慮されていなかったため、その後の多言語化の障害となった経緯があります。

さすがに現在のOSの多くは、ベースの部分を言語に依

存しないように作成し、簡単な方法でさまざまな言語に対応できるようになっているようです。この方法を、internationalizationアプローチ（i18n）といいます。それでも、Windowsで別の言語を扱おうとすれば、依然として別売の高価な外国語フォントが必要であったり、ブラウザやメーラが多言語対応なのに他のアプリケーションは付属アプリケーションでも未対応だったりと、不便な思いをします。NetscapeやICQクライアントでは、英語と、複数言語のなかから選んだもう1つの、合わせて2つが使えます。

■Linuxでの多言語・2言語化

一方わがLinuxは、オープンでフリーなさまざまのUNIX系OSのツールを寄せ集めて構築されてきました。このため、多言語化はツールそれぞれの実装しだいなのですが、おおむねi18nアプローチだといえましょう。Muleのようにツール自体が複数言語コードの取り扱いにたけていれば、言語による不公平が少なくなります。

その反面、どこの国の言語にも特化していないため、たとえばWindows上のIMEに比べると、X上のkinput2の使い勝手はちょっと不便で不自然な点があるでしょう。

■NEmacsからMuleへ

Emacsを特定の言語ではなく複数言語が扱えるようにしたのは、Mule（Multilingual Emacs）の開発チームです（http://www.m17n.org/）。Muleの歴史は、http://emacs-20.ki.nu/mule/history.shtmlにまとまっています。

当初はEmacsをlocalization（l10n）によって日本語を扱えるようにしたNEmacsが開発されたのですが、どうせ

Column

UnicodeとISO-2022

Javaなどに見られるように、コンピュータの国際化といえばすぐにUnicode、という空気が世界的に広まってきています。Unicodeは、同じような意味の文字はどの言語の文字かとは関係なく同じコードにまとめてしまおうという発想でつくられていますが、主観を許していただければ、これは「漢字を知らない西洋人」の発想です。

字形が激しく違う例を下記に挙げました。書法（小学校の書き取りで、点か「ノ」か、ハネかハライかで厳しく減点された暗い過去が……）や書体の肉付きなど、字形を美しいと感じる感性は言語のアイデンティティと密接に関わっています。

ならば、Unicodeを表示させるときに、表示させる国の字形を使えばよいではないか、と考える向きもあるでしょう。たとえば、日本でUnicodeを表示させるときには日本の文字セットを使えばよさそうです。しかし「日本人のための中国語テキスト」を執筆しようと思ったら、Unicode（日本向け）では下記のようになってしまいます。

そんな用途でもなければ、JISやGBなど、どれか1つの文字コードで閉じていて構わないわけですし、ふだん使わない文字図形を持っている必要があるのか、という議論もあります。またUnicodeは、シフトJISコードと1対1で変換できるようにするため、本来意味が同じはずの「斎」と「齋」を別々の文字図形で持っているなど、首尾一貫していません。

m17n化されているMuleでは、1つのバッファ上に多国語が混在しても、内部ではちゃんと区別されています。

余談ですが、日本では電子メールをネット上でやり取りするときはISO-2022（通称JIS）でね、というお約束が早くからできていたため、かなりお行儀の悪いメーラでもシフトJISコードのメールが飛び出すことはあまりないようです。インターネットを介したメールのやりとりに使用されるISO-2022は、どの言語の文字でも7ビットコードで表わせるようになっています。しかし台湾や中国大陸からのメールを見る限り、ほとんどのメーラがBig5やGBというローカルエンコード方式を使っています。彼らが日本語や韓国語を覚えてメールをやり取りしようと思ったとき困るのではないか、と思うのですが……（余計な心配かも）。

またWebブラウザも、UTF8対応なんていうのがあるのに（見せかけの多国語混在ページはできる）、ISO-2022対応というのは、Emacsのw3以外は見たことがありません。Webページ上で中国語講座、いや真に多国語を混在させようと思ったら、EmacsでWebページを書いてEmacs-w3で見るしかないのが現状です。

JIS： 広東風排骨麺
Big5： 廣東風排骨麵
GB： 广东风排骨面

パイコーメンもJIS、Big5（台湾・香港）、GB（大陸）でこれだけ字形が違う。

言語による字形の違い

[正しい版]

◆明けましておめでとう！！
◇新年快樂（**発音**：ㄒㄧㄣ1　ㄋㄧㄢ2　ㄎㄨㄞ4　ㄌㄜ3）

[意味不明版]

◆明けましておめでとう！！
◇新年快楽（発音：□□□1　□□□2　□□□4　□□3）

※樂と楽の字形の違いに注意

日本人のための中国語テキスト

なら日本語だけでなく複数の言語を同時に扱えるようにしようと、i18n化とともに多言語化――multilingualization（m17n）が開始されたのです。1つのバッファに複数言語をとりまぜて書けるようになったことは、画期的です。

Muleは、Emacs19まではEmacsを拡張するパッチとして改良が重ねられましたが、Emacs20.1からは標準でEmacsの機能として取り込まれました。しかし、Emacs20のMule機能はWnnやCannaといったIMを使う日本語入力がサポートされておらず、不評でした。このバージョンでは、マルチバイト文字の言語がどれでもほぼ共通の操作で変換できる、quailというEmacs内のIMを使うほかなかったのです。そこで、Emacs20でWnnやCannaを使うためのパッチを当てたり、Emacs20にダイナミック・ローディング（Linuxのカーネルモジュールと似ている）を可能にするパッチを当ててWnnモジュールをロードする方法が編みだされました（Linuxでは難しいことがあるので割愛します）。

XEmacsではさらに、leimというインターフェイスを介すことによって、quailだけでなくWnn/Cannaも統一的に使えるようになりました。

いずれにしても、この記事では文字の入力にquailを用いますので、日本語の入力にWnn/Cannaが使えるかどうかの違いは無視することにします。

複数言語の取り扱いという点で、筆者が最も快適と感じるのはEmacs19＋Muleです。しかし、MuleはEmacs19上での改良をやめたようで、せっかく多国語入力ができてもEmacsのアプリケーションが対応していないのでは問題外ですから、本記事では積極的に触れないことにします。

各国語の特性と実装

あるとき、職場を訪れたドイツ人に、日本語の漢字の入力方法を尋ねられました。簡単に説明すると「どーしてそんな面倒くさいことするんだ？」という顔をしていましたが、ほかの言語のことを知ると、なるほどドイツ人が日本語入力メソッドに触れたときの驚きがわかります。

■日本語

日本語は、世界でも最も面倒くさい部類かもしれません。まず入力方法には、ローマ字と、少数派ながらカナやOASYS入力などがあります。ローマ字はかなり年輩の方でも綴れますし、英語キーボード（ラテンキーボード）がそのまま使えますが、いったんひらがなに変換したあと漢字に変換しなければなりません。これは、文節変換（最低でも単語変換）でないと仕事にならず、昨今では携帯電話にまで連文節変換（モバイルWnnなど）が搭載されているくらいです。「はし」に対応する漢字は「橋」「端」……、「橋」の読みは「はし」「きょう」という具合で、読みと漢字の対応が1対1ではないことが理由でしょう。

Emacs20では、quailという多言語入力機能で日本語もサポートされていますが（KKS）、これはデキの悪い単語変換なので、誰も使いたくないのが現実だと思います。

さらに、日本語の文字配列（文字図形の表の配列法）はJISで定義された1種類（水準とか制定年の違いを除く）だけなのに、コーディング法（どういうデータ列で文字を表わすか）は、ISO-2022-JP（通称JIS）、シフトJIS、EUCの3種類があります。

■韓国語

韓国語にも、ひらがなにあたるハングルと、漢字にあたるハンジャがあり、実用的にはほとんどハングルだけで記述されます。ハングルは人工文字なので、入力・表示の面倒はかなり解消されているようです。ハングル文字は「子音＋母音＋子音」の部分で構成されているので、キー配列も左手に子音、右手に母音が割り当てられた2ストローク配列が標準です。これは、quailでも使用できます。

ラテン文字表記で、「金」を「kim」と表現することも可能ですが、入力メソッドとしてはほとんど使われません。韓国語の文字配列はKSX1001とKSX1002の2種類があり、ネットワーク上で流通しているエンコーディングはほとんどがEUC-KR（KSC5601）です。日本語のJISにあたるISO-2022-KRは使われなくなってきました。

韓国語の計算機処理ではほとんど漢字（ハンジャ）が使われないので、日本語のローマ字－かな変換に相当する操作をしておしまい、という感じです。ただしKSXシリーズの文字コード表を見ると、漢字とハングルが分かれて定義されています。このため、単漢字変換を行うハングル文字からハンジャ文字への変換プロセッサが存在します。

なお、北朝鮮では国策からかKPSという漢字コードが使われています。南北統一の動きで今後どのようになるかわかりませんが、現在のEmacsでは取り扱えないようです。

■中国大陸

中国大陸で使用する字形は、簡体字といって、数十年前に「手書きを効率化すれば生産性が上がるだろう」と大幅に簡略にされたものです。日本でも昭和初期に、「實」を

写真1　注音入力に使用するキーボード

「実」にするなどの簡体字化がなされていますが、高度情報化時代になって逆に複雑な字をコンピュータで手軽に書けるようになったのは皮肉なことです。

大陸では漢字を読むためにラテン文字を使う 拼音（ピンイン）が普及し、小学校でも漢字を教えるために英語のアルファベットを先に教えるほど徹底しているので、英語キーボードからの入力には困りません。漢字字形の配列としては国家標準のGBが制定され、それをローカルにコード化したGBK（日本のシフトJISに相当）が使われています。また、GBを別にコード化したISO-2022-CN（日本のJISコードに相当）があり、こちらにはBig5なども含まれています。

quailによる入力は単漢字変換ですが、実用上は問題ありません。たとえば、Windows中国語版のIMEのピンイン入力はquailと同程度です。でも、cWnnなどを使ってみると、やはり連文節変換はあったほうが効率はよいでしょう。このほかにも、漢字のパーツ（「月」「竹」など）に着目して入力する 倉頡（cangjie）という方法がありますが、練習が必要な入力方法はやはり人気が薄いようです。

■台湾および香港

台湾と香港では、繁体字（歴史の古い、省略のない漢字）が用いられています。これらはCNS11643 第1～7面という巨大な文字配列で制定されています。JIS漢字コード表が7面あると思ってください。香港で使う広東語では、口語体の言文一致の文化のために、変な形の漢字（失礼）が多数使われているので、台湾で使われないような文字図形も第3～7面には含まれています。この結果、全部でなんと4万8000以上の字形が定義されています。ちなみに、日本最大の諸橋大漢和辞典でも収録されている字は約5万字です。第1～2面は台湾・香港共通で、これには非力なパソコンでも最低限表示できるべき文字セット（という割には大陸の文字コードや日本語のひらがな・カタカナまである（笑））が定義されています。

コーディングには、電算5社が集まって制定した通称Big5や、シフトJISのようなEUC-TW、ISO-2022-CN（日本のJISコード同様で、大陸と共通）などがありますが、PCのOSの大半ではBig5が主流です。

入力には、ㄅㄆㄇㄈ（ボポモフォ、あるいは注音とも呼ばれる）という特殊な記号を刻印したキーボードを用いるのが主流です（写真1）。日本ならカナ入力といったところでしょうか。1文字分の注音を入力したあと、適当な文字を選ぶのは韓国や大陸の方式と同様です。余談になりますが、日本のカナ入力を思い起こしてもらえばわかるように、注音入力は慣れてもあまり効率が良くなく、またアルファベット入力との整合性が悪いため、台湾の電脳街では、な、な、なんと！ザウルスよろしく手書き文字入力ができるタッチパッドが1000円程度で売られています。チャットでお喋りな人のうち、結構な割合はこれを使っているようです。しかし、Linux環境ではデバイスドライバの入手が難しく、あまり嬉しくない入力方式ではあります。

言葉がラテン文字で表記できないのはなにかと不都合なので、台湾式ピンインというのを教育の場で普及させようという議論もあるようです。

■キリル文字、その他のヨーロッパ文字など

文字の種類がアジア圏の言語に比べて少ないため、基本的にはラテン文字のアルファベット（に対応づけている）キーと、シフトキーなどを用いた特殊記号の入力です。

文字コードも、特殊記号がついた文字を含めても8ビットで足りてしまうため、日本の1バイトカナ（通称半角カナ）みたいなノリですんでしまうようです。

パソコン通信時代は、これらの文字がとんでもないカタカナで表示されてギョッとしましたが、最近ではWanderlustでネットニュースを読んでいると（Content-Type: があるため）ヨーロッパ特殊文字を含むシグネチャがちゃんと現れて、やっぱりギョッとします（笑）。

多国語化への準備

さて、手元のLinuxマシンで複数の言語を表示したり入力したりするためには何が必要でしょうか。

■Xのフォント

当然のことながら、中国語を表示させたければ中国語のフォントが、キリル文字を表示させたければキリル文字のフォントが必要です。Xのフォントは、Xサーバで動作するすべてのアプリケーションが共通に利用できるものなので、一度インストールしてしまえば、フォント指定が合致してる限り、Webブラウザやcxterm（中国語化xterm）などのほかのアプリケーションでも使用可能です。また、Emacsで日本語が使えるのであれば、Muleの機能が組み込まれている公算が大ですから、何もしなくてもメーリングリストの韓国語や中国語のメッセージを読めるでしょう（これだけでは返事は書けません）。

Xの標準の配布には、中国語GB（簡体字）と韓国語のフォントが付属しています。このため、Netscapeで韓国や中国大陸のWebページに飛び込んでも、雰囲気を感じたり、表示されている漢字から大意を捉えることができます。

システムにないフォントは、フォントセットのbdfファイルさえ入手すれば、以下の手順でインストールできます。rootユーザーになるのを忘れないでくださいね。

たとえば中国語Big5のフォントは、ftp://ftp.etl.go.jp/pub/mule/intlfonts-1.2-split/Chinese.tar.gz（Xに標準で付属している以外のフォント集）に含まれていますから、これをダウンロードして展開します。

```
% tar xvzf Chinese.tar.gz
```

次にbdftopcfでbdfフォントをpcfフォントに変換し、Xのディレクトリに置きます。

```
% bdftopcf taipei16.bdf > taipei16.pcf
% bdftopcf taipei24.bdf > taipei24.pcf
% compress *.pcf
% mv *.pcf.Z /usr/X11R6/lib/X11/fonts/misc/
```

NetscapeやEmacsから使えるようにするには、alias名を作成します。リスト1の記述をfonts.aliasに追加します。

```
% cd /usr/X11R6/lib/X11/fonts/misc/
% emacs fonts.alias
```

最後にXのサーバが使えるように設定します。

```
% mkfontdir
```

ここで、

```
% xset q | grep font
```

Column

ディストリビューションと多国語対応

日本人にとって、日本語化の良し悪しは、ディストリビューションの評価の重要なポイントです。大多数の日本人には、まずはお手軽に日本語が使えることがたいせつでしょうから、これに加えて3番目の言語も、というのは高望みなのかもしれません。しかし本文で触れたように、フォント、IM、Emacsの別のバージョンといった多言語環境に必要なさまざまな要素を追加するのが難しいのでは困ってしまいます。

筆者は、エラーメッセージで日本語が出てくるとドキリとしてしまうので、RedHatなどのように英語版と日本語版の両方が選べるような場合、英語のディストリビューションがお気に入りなのですが、エラーメッセージを英語にする方法（set LANGUAGE=C）などを知らなかったり、ブラウザのメニューが日本語に固定されていたりすると、日本に住む外国人にはかなり高い障害になるだろうと思います。

英語と日本語以外に多国語を扱えるようにするためには、ディストリビューションにはcserverやkserver、cxtermなどのl10n化された多言語のアプリケーションが付属していると便利なのですが、筆者が触ってみた範囲では、日本のディストリビューションでそうした配慮をしたものはありませんでした。したがって、set LANGUAGEは、実質的にはja（日本語）かC（英語）以外は意味を持たないことになると思います。

筆者は台湾で、地元のディストリビューションLinpus Linuxをセブン・イレブン（！）で149台幣＝600円弱（！！）で売っているのを見つけました。ちょっとインストールして使ってみた感じでは、日本のディストリビューションが日本語に特化しているのと同様、やはり中国語に特化した環境でした。

ただし、大陸と台湾・香港の両方で売れること、あるいはメールのやり取りができることは重要なようで、GB/Big5の両方が扱えるツール（セッティング）が多いようです。そういう意味では、3カ国語対応といえます。Big5とGBの漢字コンバータなど、面白いツールも付属していました。

▼リスト1　fonts.aliasに追加する記述

```
taipei16 -uw-ming-medium-r-normal-fantizi-16-160-75-75-c-160-big5.eten.3.10-1
taipei24 -kc-ming-medium-r-normal-fantizi-24-240-75-75-c-240-big5.eten.3.10-1
taipei24k -kc-kai-medium-r-normal-fantizi-24-240-75-75-c-240-big5.eten.3.10-1
-taipei-ming16-medium-r-normal-*-16-160-*-*-*-*-big5-0 taipei16
-taipei-ming-medium-r-normal-*-24-240-*-*-*-*-big5-0 taipei24
-taipei-kai-medium-r-normal-*-24-240-*-*-*-*-big5-0 taipei24k
-taipei-li-medium-r-normal-*-24-240-*-*-*-*-big5-0 taipeil24
```

を実行してフォントパスに/usr/X11R6/lib/X11/fonts/miscがなければ、Xサーバを起動し直すか、

```
% xset +fp /usr/X11R6/lib/X11/fonts/misc
```

を実行します。中国語を使用する場合には、Chinese-HOWTO（http://www.linux.or.jp/JF/JFdocs/Chinese-HOWTO.html）という文書が参考になります。

■Emacsのバージョンと Mule

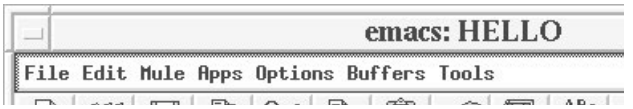
日本のディストリビューションであれば、Emacs/XEmacsにMuleの機能が含まれていると思いますので、Emacsに手を入れる必要はありません。一応、Emacsのメニューに「Mule」という表示があるか確かめましょう（**画面1**）。

■kWnnとcWnn

個人的には、韓国語と中国語の入力にはIMを使わず、quailだけで間に合っています（ばりばり文章を書くほどの語学力がないだけかもしれませんが）。もしどうしてもIMを用いたければ、やはりkWnnとcWnnでしょう。eggの中でkorean.elとchinese.elが定義されているため、Wnn4またはWnn6を使っている人であれば、さほど苦労することもなくcWnnやkWnnを使うことができます。本稿では詳しく触れませんが、http://www.linux.or.jp/JF/JFdocs/Chinese-HOWTO-13.htmlなども参考にしてください。

中国語変換サーバを利用するには、まずcserverをRPMなどから導入します。また変換途中で四声フォント（sish14-etl.bdf/sish16-etl.bdf/sish24-etl.bdf）が必要なので、前記の方法でインストールします。Emacsを立ち上げたら、まず**M-x load-library**と入力して**chinese**を選択し、**M-x set-cserver-host-name**と入力して**localhost**を選択しておきます。この代わりに.emacsに、

```
(load-library chinese)
```



画面1　Mule機能を確認する

```
(set-cserver-host-name "localhost")
```

などと記述しておいてもよいでしょう。最後に**C-x C-k m**でpinyinを選択します。

韓国語変換サーバの場合、kserverを導入します。cserverの場合と同様にM-x load-libraryと入力してkoreanを選択し、M-x set-kserver-host-nameと入力してlocalhostを選択し、最後にC-x C-k mでkoreanを選択します。

台湾・香港で使われている繁体字のBig5には、cserverのセットにtserverというのが付属していますが、eggからtserverを利用する方法は確認できませんでした。

使ってみよう――Emacs19＋Mule編

先にも述べましたが、多国語の使用方法はここでは詳しく述べません。

■quailモードへ突入

C-\でかな漢字変換に入るのと同様にして、quailモードへは**C-]**で入ります。この時点でモード行に、

[한글 2벌식]　**（ハングル2ストローク）**
[한글 3벌식]　**（ハングル3ストローク）**
[拼音]　　**（ピンイン）**

などと表示されます。

■quailの選択

quailモードで**M-s**とタイプすると、入力方式が選択できます。〈SPC〉などによる補完も効きます。ハングル語ならhangul、中国大陸方式ならpyなどを選んでやればよ

いでしょう。残念ながら、台湾や香港では一般的な注音による入力はできません。その代わりに、py-b5というのが利用できます。

quailの選択はquailモードをオンにしていないとできないのですが、オンにしているとquailを選択するための入力ができなくなるというバグがたまに出ます。たとえば、pyからhangulに切り替えようとしても、ミニバッファで「hangul」とタイプできないのです。このようなときは、**C-x o**でquailモードのバッファに移り、**C-]**でquailをオフにしてからミニバッファに戻れば問題を回避できます。

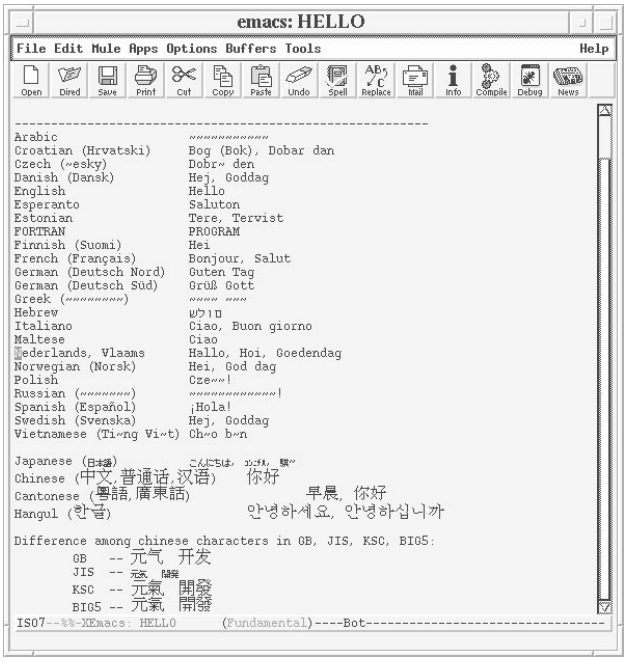
■ファイルへの保存

コラム「UnicodeとISO-2022」で触れたように、Emacs内部では1つのバッファ内に多国語を取りまぜて書くことができます。たとえば「中国語のテキスト」を作ることを想像してみてください。しかし、これをファイルとしてセーブするときに、シフトJISやEUCを選んでいてはいけません。ファイルが日本語の3種類の漢字コードのいずれかであれば自動判別は可能ですが、ファイル内の場所によって文字コードが違っているケースまでは想定していません。また、自動判別の優先順位で日本語を高くしていると、Big5のファイルを壊れたシフトJISのファイルと判別することもあるようです。EUC-JP（日本語）とEUC-TW（中国語繁体字）の組み合わせでは、文章の中身を理解しないかぎり判別はできない相談です。

しかし、ISO-2022は「文字種が変わるところに合図を入れる」という方式なので、これを選べば多言語を混在させたままちゃんと保存できます。べつだん多国語が混在しない場合でも、文字コードの判別を取り違える心配はありません。これが、ISO-2022エンコード法がネットワークでの流通にお勧めされていた所以です。

ISO-2022でのファイル保存を選択するには**C-x C-k f**でファイルの文字コードを設定します。この操作は、EUC-JPとJISとシフトJISの切り替えのために使ったことがある方がいるかもしれませんね。

「＊iso-2022-int＊」を選ぶと、次にこのファイルを開いたときには自動的に同じ文字コードでセーブできます。また、Emacs内部で使われている冗長な文字コード形式（「＊internal＊」）でも、確実に保存ができますが、ほかのアプリケーションとのやりとりはできません。なお、多国語が混在しているバッファから他の文字コードにセットされているバッファにコピー＆ペーストし、いったんセーブして終了（またはkill-buffer）したあと再度開くと、ファイルが壊れていて泣きたくなることがあります。ご注意を。



画面2　各国語で「こんにちは」を表示させる

使ってみよう――Emacs20/21、XEmacs編

■Xのフォント指定

はじめに、メニューの［Mule］－［Show many languages（C-h h）］を選んでみましょう。**画面2**のように、バッファ「HELLO」に各国語で「こんにちは」とあいさつしてくれます。このバッファを見れば、足りないフォントがわかるというわけです。画面2では、アラビア語などのフォントがないので「~~ 」のように表示されています。

使用する言語の文字が表示されなかったら、前記の説明を参考にフォントをインストールしてみてください。

■言語環境の設定

Emacs20/21では、ファイルの保存だけでなく、言語環境や使用するIM（といってもquailですが）の組み合わせなどが一発で設定できます。メニューの［Mule］－［Set language environment］を選んでみましょう。

日本で買ったディストリビューションだけど、韓国語を使うことが多くていちいち優先の設定をするのが面倒という人は、.emacsに次のような設定を書いてしまえば、デフォルトの言語環境を変更できます。

```
(set-language-environment korean)
```

■quailの選択とquailモードへの突入

quailの選択には、メニューの［Mule］－［select input method］か、**C-x**〈RET〉**C-**を使います。quailモードへは**C-**で入れます。このあたりは、キー操作が異なるだけで手順はEmacs19とほぼ同じです。

韓国語と中国語の入力

論より証拠。quailを使って、代表的な韓国語と中国語2種類の入力のようすをお見せしましょう。

■hangulとhangle3モードによる入力

このモードを使うには、まず2ストロークまたは3ストロークキー配列の表を入手します（http://www.taroh.org/Docs/linuxmag/）。これさえあれば、あとはひたすらタッチタイピングを練習するだけです。**図1**は、2ストローク配列（quail名hangul）で「はんぐる」と入力するようすです。ハングル文字がにょろにょろと組み立てられて、面白いですよ。3ストローク配列（hangul3）でも基本的には同じ手順です。

■pyとpy-b5モードによる入力

ピンイン入力では、ミニバッファに次々と表示される綴りや変換後の文字の候補を見て選んでいきます。

図2は「ピンイン」を入力する模様です。ラテン文字の「pinyin」という綴りを入力するのですが、「pin」にも「yin」にも対応する漢字の候補は多数あります。

まず、〈p〉をタイプした段階でミニバッファに表示される「p［aeiou］」は、「pa…」「pe…」「pi…」「po…」「pu…」という綴りが可能だということを示しています（ピンインには「ph」ではじまる綴りはありません）。

次に〈i〉をタイプすると、ミニバッファは「pi」の候補の一覧に変化して、同時にデフォルトで選ばれる漢字がポイント位置にも表示されます。ここではまだ続きがあるので〈n〉をタイプします。たまたま目的の「ピン」が候補1番に表示され、ポイント位置にも表示されています。これでよければ、いきなり次の文字を入力してしまってOKです。つまり、この時点で続けて入力可能な文字は「g」だけですから、それ以外の文字、たとえば〈y〉をタイプすれば、自動的に変換中の文字が確定して次の文字の入力にとりかかれるのです。日本語のIMのつもりで〈RET〉をタイプすると、改行されてしまいます。明示的な確定操作には〈SPC〉を使います。また、カーソル移動など、そのほかのアクションも、入力確定とみなされます。

同様にして〈y〉〈i〉〈n〉とタイプしていくと、今度は目的の「イン」の字が候補リストの6番にあるので、〈6〉をタイプして選択・確定します。リストの前の数字が「(1/4)」などとなっていたら2～4ページ目のリストがあるので、〈>〉をタイプして次に進めましょう（戻るときは〈<〉です）。これも反射的に**C-n**/**C-p**をタイプしたくなるところですが、我慢してください。

BIG5では、同様のpy-b5が選択できます。大陸式のピンインを知らないと入力は難しいでしょうが、注音記号が表示されていくので、台湾・香港の人でもすぐに使えるようになるでしょう（**図3**）。py-b5では中国語の四声によって候補を絞り込むようになっています。ピンインに続けて数字（1～4）が指定可能な場合はそれが表示されます。

キー入力	ミニバッファ	ポイント位置の表示
g	ㅎ	ㅎ
k	ㅎㅏ	하
s	ㅎㅏㄴ	한
r	ㄱ	한ㄱ
m	ㄱㅡ	한그
f	ㄱㅡㄹ	한글

図1　KS配列を使用したハングルの入力

キー入力	ミニバッファ	ポイント位置の表示
p	p[aeiou]	p
i	pi[ean] (1/5) 1.辟 2.坏...	辟
n	pin[g] (1/2) 1.拼 2.拚...	拼
y	y[aeiou]	拼y
i	yi[n] (1/11) 1.一 2.壹...	拼一
n	yin[g] (1/4) 1.烟 2.茵...	拼烟
6		拼音

図2　ピンイン入力

キー入力	ミニバッファ	ポイント位置の表示
p	p[aeiou]	ㄆ
i	pi[1234aen]	pi
n	pin[1234g]	pin
l	pinl (1/1) 1.拼 2.姘 3.嫔...	拼
y	y[aeiou]	拼y
i	yi[1234n]	拼yi
n	yin[1234g]	拼yin
l	yinl (1/3) 1.因 2.音 3.阴	拼因
2		拼音

図3　Big5のピンイン入力（py-b5）

Linux日記

第18回　メール配送（5）

sendmailがメールを送受信するときにはどのような動作をしているのでしょうか。今回は、sendmailのデーモン動作と配信エージェントについてのお話です。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

PCで日本語入力がちゃんと行えるようになったのは1980年代前半だっただろうか。かな漢字変換フロントエンドプロセッサというプログラムが登場し、MS-DOSなどに組み込まれた。これを使うことで、さまざまな部分に漢字混じりの日本語を簡単に入力することができるようになった。漢字混じりの日本語により、それまでの半角カタカナ表記よりも非常に読みやすくなった。

PCで漢字混じりの日本語が使えるようになる前に登場したのが、日本語ワープロという製品である（正確にいえば、日本語ワープロで生まれたかな漢字変換技術がPCの世界に移行してきたということになる）。これは漢字混じりの日本語入力が可能で、画面で文章の編集を行い、プリンタに出力することができた。当時は、漢字混じりの日本語を入力する方法もさまざまだった。もっとも原始的な文字コード入力、漢字一覧表をタッチペンで押すという和文タイプのような方法などがあったが、そんな中でかな入力された文字列を漢字に変換するという技術が開発された（最初に製品化したのは確か東芝だったと思う。1970年代の後半か末だった）。この方式は、読みがなに対する漢字、熟語などをディスク上の辞書に登録する必要があったので、当時のリソース（特に価格）を考えるとかなり高級なアプローチであった。最初のこの種のワープロ専用機は、机くらいの大きさで、数百万円という値段だったと思う。

読みがな入力で漢字変換という方法は、通常の日本語（カナ）キーボードを使うことができ、また、ほかの入力方法に比べて格段に高速な入力が可能なので、最終的にはこの方法が世間に広く普及することになった。

ところで、当時のJISカナキーボードというのはとんでもなく使いにくい代物であった。基本的な配列などは現在の106キーボードとさほど違わないのだが、シフトがすべてロックされるのである。現在のキーボードでは、ロックされるシフトはCaps Lockとカナだけで、左右のシフトキーはノンロックである。当時のJISカナキーボードは、英小文字（下段）、英大文字（上段）、カナ（下段）、カナシフト（上段）の4状態があり、すべてがロックされるシフトだったのである（シフトキーというより、単なる状態遷移キーであった）。上段の記号を打つ前にシフトを押し、下段の文字を打つ前にその状態に戻すキーを押さなければならなかったのだ。そのため、英字混じりの日本語などを打つ場合、シフトキーの打鍵が多くなり、非常に面倒だった。また、数字は英字下段モードでしか入力できなかったので、普通の日本語だけでも辛かったのではないかと思う。

これはあまりに使いにくかったので、1980年代初頭に登場したパソコンは、英数ーカナの切り換えのみをロックシフトとし、上段下段の切り換えはノンロックシフトにすることで多少は使いやすいものにした。そして、そのまま現在に至っている。ましになったとはいうものの、カナを打っている途中で

数字や英記号を入力するためには、カナロックを解除する必要があり、あまり効率的とはいえなかった。

この問題に対して、まったく別のアプローチを行ったのが富士通だった。カナのキー配置を一新し、カナモードのまま数字やよく使う記号を入力できるようにしたのである。JISキーボードの問題は、数字の列にまでカナを割り振ったことだ。そこでカナを英字キーに使われている3列に収める。ただ、カナは文字種が多いので、1つのキーに2文字割り当てることになる。上段のカナを打つために、モーメンタリとはいえシフトキーを押すとなると2ストローク入力となり、打鍵速度が低下する。そこで考え出されたのがすべてのかなを1ストローク入力できる親指シフトである。

親指シフトキーボードには、スペースバーの位置に2つの背の高いシフトキーがある（スペースバーは右横方向に移動した）。そしてそのシフトキーの手前に変換、無変換というキーがある。下段の文字入力はそのキーを押すだけだが、上段カナキーの入力には親指シフトキーを使う。親指シフトの場合、従来のシフトキーと異なり、カナキーと同時に中央のシフトキーを打鍵するのである。シフトを押してからキーを押すという方法では2ストローク入力になるが、同時打鍵なので、1ストロークで上段キーを入力できる。親指シフトキーは中央に2個あり、通常の上段打鍵は打つキーと同じ側の親指シフトキーを押すのだが、反対側の親指シフトキーと同時打鍵すると、濁音が入力される。従って、濁音も1ストローク入力できるのである（**写真1**）。

筆者は、1980年代初頭、まだ原稿を原稿用紙に書いていた時代に、この親指シフトキーボードを備えたOASYSというワープロを使い始めた。これは快適であった。辞書はフロッピーで、変換は低速だったはずなのだが、とにかく高速入力ができた。ローマ字でシャカシャカタイプしている横で、ノタノタ親指シフトキーボードを打っていても、親指シフトキーボードのほうがはるかに高速入力できるのである。

その後、パソコンの日本語対応が進み、MS-DOSの時代には、オペレーティングシステムのレベルにかな漢字変換機能が組み込まれ、ワープロソフトやエディタで日本語入力ができるようになった。親指シフトキーボードはJISに制定されなかったこともあり、富士通と一部の協議会などが推進するだけのローカルな規格となり、パソコンの世界では広く普及するには至らなかった。筆者も、かつてはPC-9801に親指シフトキーボードを接続して使っていたのだが、PC互換機に移行してからはローマ字入力をずっと使っていた。

そんなある日、OASYS V8というWindows用ワープロソフトのパンフレットを見ていて、興味深い機能を見つけた。一般的な106キーボードで親指シフト入力ができるというものだ。スペースバーの左右の変換、無変換キーをシフトキーとして使って親指入力ができるというものだ。この種の機能を実現するソフトは、以前からフリーソフトとして存在していたのだが、純正サポートになったということもあり、ちょっと気になった。で、ものは試しと買ってきて、実際に動かしてみた。

9年ぶりに親指シフト入力を使ってみたのだが、けっこう覚えているものだ。何も考えなければそこそこ入力できた（考えるとダメだったが）。しかし、やはり106キーボードの変換、無変換をシフトキーに使うというのでは位置が悪い。そこで、富士通純正のキーボードも買った。OASYSそのものを使うわけではないので、標準的な106キーを親指シフト化したFMV-KB211という製品である。しかし、まだいまいちである。英字や日本語の切り換え、キーボードドライバの状態遷移のしかたになじめなかったのである。結局、フリーで拾ってきたドライバをインストールすることで、だいぶ改善され、現在はその環境を使っている（かくして、OASYS V8の機能は何ひとつ使われないこととなったのである）。

写真1　親指シフトキーボードFMV-KB211のフルキー部分

というわけで、この原稿は久々に親指シフトで書いている。しかし、実際に使ってみると、やはりローマ字ではなくカナベースということで、英字や記号の入力に難がある。日本語関連の記号は問題ないのだが、この種の文章には英単語や英記号が山ほど登場する。ローマ字のときは、英単語をそのまま入力し、半角変換すればよかったのだが、カナベースの場合は、英字モードに切り換える必要がある。やはりこれはうっとうしい。ローマ字のときのように、入力後に英字に変換し、確定できるようになっていてほしいところだ（筆者が使っているフロントエンドにはこの機能があるのだが、マッピングが106配列用なので使い物にならない）。やはり、究極的な日本語入力環境を実現するには、自分でドライバを書くしかないのだろうか？

さて、sendmailの話の続きだ。前回は、sendmailの動作モードとして、MUAから起動されるモード、デーモンとして動作するモード、関連コマンドとして動作するモードがあるという説明をした。

■メール送信動作

sendmailの動作モードを決定する-bオプションのデフォルト（-bオプションを指定しなかった時の動作）は-bmである。このモードで起動されたsendmailは、パラメータで指定された宛て先に対して、標準入力から与えられたメールを送信する。以前に、sendmailを手動で実行したときのことを思い出そう。「sendmail user@takobeya.com」とコマンドラインで指定すれば、標準入力から受けたメールをuser@takobeya.comに送信する。

メール中にはTo:、Cc:ヘッダで宛て先が指定されているのが普通だが、sendmailはパラメータで指定された宛て先をエンベロープ情報として使用し、メール送信を行う。したがって、メール中のTo:、Cc:は必須ではない。実際、Bcc:だけで宛て先を指定したメールの場合、MUAはBcc:に基づいてsendmailに宛て先パラメータを渡すが、メール中には宛て先情報は含まれていない。この意味については前回説明した。

ところで、メール中のTo:、Cc:、Bcc:の情報に基づいて、sendmailに宛て先を判断させることもできる。-tというオプションを付けて実行すると、sendmailはメール中のTo:、Cc:、Bcc:ヘッダを調べ、その宛て先に基づいてメールを配信する。つまり、To:などのヘッダに基づいて、sendmailがエンベロープ情報を作成し、配信を行うのである（**画面1**）。-tを指定したsendmailに渡すメールには、Bcc:ヘッダが含まれている可能性があるが、このBcc:ヘッダは、エンベロープ情報を生成した後、sendmailによって削除される。

通常、-bm形式のsendmailは、MUAプログラム中から実行される。MUAは、ユーザーが指定した宛て先をTo:、Cc:、Bcc:ヘッダの形でメールに収めるとともに、sendmailに渡す引数として用意し、sendmailを起動することで、メールの送信を行うことができる。あるいは-tオプションを使って、sendmailにエンベロープ情報を用意させることもできる（こちらのほうが簡単だ）。

いずれの場合も、メール本文は、起動したsendmailの標準入力に与えられるようにする。MUAのする仕事はここまでだ。残りの配信処理は、sendmailがすべてやってくれる。

■デーモン動作

-bdオプションを指定して起動すると、sendmailはデーモンモードで動作を開始する。この状態のsendmailの主要な機能は、外部からSMTPで配信されてきたメールを受信することだが、ほかにもメールキューの処理という仕事を行っている。

```
[test ]$ /usr/sbin/sendmail -t
To: test
From: test
Subject: -t option

Test mail of -t option.
.
[test ]$ more /var/spool/mail/test
From test Sun Nov 26 16:08:47 2000
Received: (from test@localhost)
        by mail.takobeya.com (8.9.3/3.7W-2.8compat) id QAA00972;
        Sun, 26 Nov 2000 16:08:09 +0900 (JST)
Date: Sun, 26 Nov 2000 16:08:09 +0900 (JST)
From: Test account <test@takobeya.com>
Message-Id: <200011260708.QAA00972@mail.takobeya.com>
To: test@takobeya.com
Subject: -t option

Test mail of -t option.

[test ]$
```

画面1　sendmailの-tオプション

・SMTP受信

デーモンモードのsendmailは、SMTPポート（25番）への接続を待ち受けている。外部のMTAがこのホストにメールをSMTPで配信する場合、sendmailデーモンがそれを受信する。sendmailは、接続リクエストごとにinetdから起動されるタイプのデーモンではなく、常時稼働しているタイプのデーモンである。sendmailは、多数のメール配信を並行して処理するために、そして1本のメールの処理が完了した時点でデーモンが終了してしまわないように、新たなコネクションごとにプロセスをフォークし、子プロセスが個々のメールの処理を行い、親プロセス側は次のコネクションを待ち受けるという形で動作する。

フォークした子プロセスは、SMTPコネクションでメールを受信し、それを処理する。エンベロープで指定された宛て先を解釈し、必要な配信処理を行う。自ホストで管轄しているユーザー宛のメールであれば、ローカル配信エージェント（binMail）を起動し、受信メールを各ユーザーのメールボックスにスプールする。自ホストのユーザー宛でない場合は、宛て先を管轄する別のMTAに配信を行う。このように、外部から受信したメールを、別のMTAに再送信する機能を、中継機能（リレー機能）という。単純な構成のメールシステムであれば中継機能は必要ないが、大規模組織で複数のMTAを運用している場合や、特殊なメール配信を行う場合などには便利に使える。もっとも、中継機能は悪用されることもあるので注意が必要だ。

・メールキュー

sendmailは、送受信するメールを一時的にキューに保管する。これがメールキューである。MUAから受け取った送信メール、あるいはSMTPで受信したメールを即座に処理（ローカル配信、中継など）できれば、メールはまさに一時的にキューに置かれるだけで、処理後には削除される。従って、SMTP配信しか行っていない環境であれば、正常動作している限り、メールキューはいつも空である。負荷が重く（同時に多数のメールの配信が行われたときなど）配信が間に合わない場合などは、メールキューに一時的にメールが滞留することもある。しかし、多少遅れても配信されるのであれば、さほど問題にはならない。

実際の運用では、一時的な滞留などとは別に、メールキューにいくつかメールが残ることがある。たとえば、何らかの理由で送信先のMTAにメールを配信できない場合（宛て先サーバのダウン、ネットワークトラブルなど）、配信処理において致命的なエラーが発生した場合（送信も返送もできないようなアドレスエラーなど）がある。また、sendmailとは非同期に動作する別種のMTA（UUCPによるメール配信など）を併用している場合は、バッチ処理のためにメールキューにメールが投入されることもある。

何もしなければ、メールキュー中のメールは配信されないまま残ってしまう。送信時にMUAから起動されるsendmailは、エラー時の再送信処理などは行わないからだ。メールキューにたまっているメールの処理は、デーモンモードで動作しているsendmailの仕事である。sendmailデーモンは、一定時間間隔でメールキューをチェックし、配信が保留されているメールの処理を行う。エラーで配信が遅れているメールであれば再送信を試み、別のMTAにより投入されたメールがあれば、その配信を行う。このような処理をデーモンモードのsendmailに一括処理させることにより、ほかのMTAのエラー時処理の負担を軽減することができる。

sendmailデーモンがメールキューをチェックする間隔は、起動時のコマンドラインオプションで指定する。この時間が短ければ、再送信などの処理をすばやく行えるが、メールを大量に扱う場合などは、CPU負担が大きくなる。間隔を長くすれば負担は低下するが、メール配信の遅れが大きくなる可能性が高くなる。通常の運用であれば、15分から1時間程度に指定しておけばいいだろう。

メールキューのチェック間隔は、-qオプションで指定する。-q30mと指定すれば、sendmailの起動後、30分間隔でチェックするようになる。システムの起動スクリプトでは、例えばsendmail -bd -q30mというように指定することになる。

Column

新JISキーボード

JISカナキーボードの使いにくさを改善すべく、新たなJISカナキーボードが制定された。これはカナを3段に収めて、普通のシフトを使って入力するものだったが、普及することはなかった。

その頃かなり普及していた親指シフトキーボードはJISに制定されなかった。

当時、日本の民間企業が開発したものはJISには採用されないという傾向があったようだ（海外の企業の規格は採用されているのだが）。

かくして、使いやすいカナ入力キーボードは普及せず、多くのユーザーはストローク数の多いローマ字入力か、使いにくいカナ入力をいまだに使っている。

メールキューの状態は、mailqコマンドでチェックできる。このコマンドは実際にはsendmailへのリンクで、mailqはsendmail -bpと同じである。mailqを実行すると、現在メールキュー中に残っているメールが表示される。

■配信エージェント

さて、SMTP配信の話に戻ろう。sendmailからmail.localを呼び出せばローカル配信を行える。ではsendmailは、別の配信エージェントを呼び出して、SMTPによるリモート配信を行っているのだろうか。これは半分正しい。確かに別の配信エージェントを使うことにより、メールをリモートホストに配信することができる。しかしsendmailは、SMTP配信については、別プログラムの配信エージェントを呼び出すということはしない。IP接続のSMTP配信エージェント機能は、前にも説明したように、sendmailに内蔵されているからだ。

sendmailの各種配信エージェントは、sendmail.cf中で定義する。ここでは、配信エージェントの名前（sendmail.cf中から参照される）、実際に実行されるプログラムや各種オプションを指定する。

配信エージェントをいろいろ定義しても、実際にメールシステムを運用するためには、どの配信エージェントを呼び出せばいいのか？　あるいは呼び出す配信エージェントをどのようにして求めるのか？　また、配信エージェントに渡すさまざまなオプションはどのような意味を持っているのか？

まさにこれこそが、sendmail.cfとの長く辛い戦いなのである。

■配信エージェントの選択

メールを送信するという面で考えると、MTAの重要な仕事のひとつは、送信に使う配信エージェントの選択である。どの配信エージェントを使うのか、そして配信エージェントにどのようなパラメータを渡すのかが決まれば、後の作業はさほど難しいものではない。ここまで実例を示してきた通りだ。

送信に使う配信エージェントを決めるのが、メールの宛て先アドレス（エンベロープ情報）であることは明らかだ。MTAは宛て先アドレスを調べて、ローカル配信なのかリモート配信なのかを判定する。メールアドレスのドメイン部を調べ、自身のドメインと同じであればローカル配信エージェントを起動し、異なるのであれば、DNSにMXレコードを問い合わせて宛て先のメールエクスチェンジャを調べ、そのホストに対してSMTP配信エージェントを起動すればいい。さほど難しいことではなさそうだ。

だが、ここでちょっと考えてほしい。自身のドメイン名とは何か？

mail.takobeya.comというホストがメールサーバとして機能し、user@takobeya.comという形式のアドレスを持つメールを受信処理しているとしよう。このとき、宛て先がuser@ascii.co.jpのメールをリモート配信するのは明らかだ。user@takobeya.comならローカル配信である。では、user@mail.takobeya.comという宛て先の場合はどうするか？

takobeya.com内で送信されたメールであれば、この形式になることはないはずだが、外部からこのアドレスで送られてくることがあるかもしれない。メールアドレスのドメイン部にメールサーバホスト名まで入っている形式だが、アドレスの意図からすれば、ローカル配信であろう。

それでは、user@writers.takobeya.comというアドレスならどうするか？　単純にエラーとして処理することもできるだろう。しかしtakobeya.comに、writers、editors、designersといったサブドメインがあったとしたらどうだろう。正式なアドレス形式はuser@takobeya.comかもしれないが、user@writers.takobeya.comといったアドレスのメールも正しく配信するのが親切というものだろう（もちろん、冷酷にエラーとして切り捨てるという選択もある）。また、同じtakobeya.comドメインでも、user@programmers.takobeya.comというアドレスで届いたメールは、別のメールサーバmail-svr.programmers.takobeya.comにSMTP配信、つまりリモート配信しなければならないかもしれない。

このように、単に自身のドメイン名といっても、簡単に定まるものではない。takobeya.com、writers.takobeya.com、editors.takobeya.comといったドメイン名は自身の管轄とみなし、programmers.takobeya.comはリモート配信、sales.takobeya.comはエラーとするといった細かい判断が必要になる。

リモート配信についても同じようなことがいえる。リモート配信にSMTPしか使っていないのであれば、たとえ相手が自分のサブドメインのメールサーバだろうが、MXレコードを頼りに同じように配信できる。しかし、UUCP配信も併用していたら、そのための判定処理も必要になる。たとえばkanzume.netというドメインについてはUUCP、そのほかのドメインについてはSMTPといった条件指定も必要になる。

MTAの設定ファイル、つまりsendmail.cfには、こういったことをひたすら記述していかなければならないのだ。

■そろそろsendmail.cfについて

基本的にsendmailは、MUAから呼び出されるメール送信エージェント、外部から送られてくるメールを処理する受信エージェントという2つの機能を持っている。プログラムの動作モードなどは、実行時に指定されたオプションなどで決まるが、MTAとしてメールをどのように処理するかは、いくつかの設定ファイルに基づいている。中核となる設定ファイルが、かの有名なsendmail.cfである。

sendmail.cfは、送信時の配信エージェント選択、メールアドレスの書き換え、各種パラメータ設定など、受信時の配信エージェント選択、中継処理などを記述したテキストファイルである。sendmailは起動時にこのファイルを読み込み、その定義に従ってメール配信、デーモン動作などを行う。

sendmailは、デーモンとして起動されるのは1回だけだが、MUAがメールを送信する時にも起動されるため、プログラムとしての実行回数はかなりの数になる。そのたびにsendmail.cfを読み込み、解釈するというのは、かつてのコンピュータにとってはかなりの負担であった。そのため、sendmail.cfの記述文法は、プログラムがすばやく簡単に解釈できるという構造になっている（さらに負担を軽減するために、かつては、ファイルを解釈した後の内部イメージをファイルに保存し、実行時はそのファイルを直接ロードするという機能があった）。そのかわり、人間にとってはわかりにくい構造になってしまった。およそわけのわからないテキストファイルである。

現在、管理者がsendmail.cfをゼロから記述するということはまずない（もちろん、やりたければやってもよい）。通常は、既存（システムのデフォルト）のsendmail.cfをちょっと書き直すとか、各種sendmail.cf生成ツールを使って、パラメータを列挙したファイルからsendmail.cfを自動生成することになる。筆者は、CFというツールで作成したsendmail.cfを使っている。このようなツールの使い方を知っていれば、sendmail.cfがわからなくてもsendmailを運用できるが、それではsendmailを解説する意味がない。せっかくだから、たとえ書くことはできなくても、読めるくらいにはなっておこう（書けるようになりたければ、コウモリの本を読むべし）。

sendmail.cfの中身の解説に先立って、まずは基本的な構文規則を示しておこう。

・コメント

行頭が#の行はコメントであり、プログラムからは無視される。

・コマンド

行頭の1文字の英字は、コマンドを意味する。行の2文字目以降の内容はコマンド次第である。

・継続行

コマンドは行単位で処理されるが、行をタブまたは空白で開始することにより、前の行に続く継続行とすることができる。

これだけだ。あとはひたすらコマンドとそのパラメータの意味を解釈していくことになる。まずは、どのようなコマンドがあるのかを概観してみよう。

■各種パラメータ設定

sendmailが動作するうえで必須のパラメータ、動作の詳細を定義するオプションパラメータなどを、sendmail.cf中で設定することができる。一部のパラメータはコマンドラインオプションで指定することもできるが、常に指定するパラメータはsendmail.cf中で定義しておくのが簡単だ。パラメータに関する説明は、sendmail.cfの解説がさらに進んでから行うことになると思う。

パラメータを指定するコマンドには**表1**のようなものがある。これらを使い、**リスト1**のように指定することになる。

■次回は

ついにsendmail.cfの解説に入ってしまった。次回は、マクロとクラス、ヘッダ生成や配信エージェント指定などの各種コマンドについて解説する。そして「sendmail.cfはわけがわからない」といわしめている最大の原因であろうルールとルールセットの解説を始める予定だ。

これらのコマンドは、相互に関連している。これらの関係がわかってくれば、多少はsendmail.cfの中身が見えてくるようになるだろう。もっとも、雑誌の簡単な解説記事を読んだ程度では、sendmailの動作を完全に理解するとか、自分で記述できるようになるというのは難しいだろうが。

コマンド	意味
V	sendmailのバージョンを指定する
O	各種オプションパラメータを指定する
T	トラステッドユーザーを指定する。これはメールシステムの運用において、特殊な権限を持つユーザーである

表1　パラメータを設定するコマンド（抜粋）

▼リスト1　オプションなどの例

```
V8/Berkeley                 バージョンの指定
O AliasFile=/etc/aliases    AliasFileというオプションを/etc/aliasesに設定
T root daemon uucp          トラステッドユーザーを指定
```

Devfs（デバイスファイルシステム）－1

文：政久忠由
Text : Tadayoshi Masahisa

正直なところ、Linuxのデバイス管理は、旧石器時代の手法と言ってもいい。とにかくダサいというか、見栄えにしても、使い勝手にしても、よくない。

でもこのことは、Linuxだから、というわけではなくて、Linuxがベースにした UNIX アーキテクチャのデバイス管理がそのように実装されていたので、UNIX クローンとして作成されたLinuxも、表面的、つまりソースコードレベルでのアプリケーションソフトウェアの互換性を考えると、必然的に同じような構造にせざるを得なかった、という理由からである。内部的な管理はいくら異なっていても何の問題もないけど、インターフェイスだけは同じでないとクローンとしての意味がないのだ。

今回は、LinuxのOSとしてのデバイス管理と、その実装に関する新しい試みのひとつを見てみたいと思う。

OSとしてのデバイスの管理

Linuxを含む、一般的なUNIX系OSでは、デバイスは、内部的なデバイス番号と、それを関連付けた特殊なファイルで構成管理されている。

なお、ここで言うデバイスは、ハードウェアからソフトウェア的に作成された仮想的なものまで、OSが扱い、操作するすべてのリソースと考えておけばよいだろう。

すべてのリソースを統一的な方法で操作できるようにするというのは、UNIX の主要コンセプトのひとつである。その実装として、ハードウェアデバイスをはじめ各種デバイスは、ファイルとしてソフトウェア環境に提供され、ソフトウェア環境では、そのファイルを利用して、各種デバイスも通常のファイルと同様に、共通の操作、オープン、クローズ、リード、ライトなどを行うことができるようになっている。まぁデバイスの中には、共通の操作には当てはまらない特殊なコントロールが必要な場合もあるので、それらについては、ioctl()というかたちで別途用意されている。

■デバイス番号

さて、内部的なデバイス番号の割り当ては、各OS依存で自由に定義して構わない。ソフトウェアレベルでは、このデバイス番号でもって直接デバイスを操作することは、基本的にないからだ。

Linuxのデバイス番号は、それぞれ256のメジャー番号とマイナー番号に加えて、キャラクタ（char）とブロック（block）というデバイスタイプで区別される。

キャラクタデバイスとブロックデバイスの違いは、そのデバイスに対するデータのやり取りで、そのデータを中間レイヤでバッファリングをするか、しないかの差である。キャラクタデバイスはバッファリング処理を経由しない、ブロックデバイスはバッファリング処理を経由することになる。たとえば、仮想端末のように、単にデータが通過するだけのデバイスはバッファリングの意味がないのでキャラク

タデバイスとして、ファイルシステムで管理するストレージシステム（ハードディスクドライブなど）のように、データのアクセスがランダムに生じ、同じデータの要求が頻繁に発生するようなデバイスは、バッファリングすると効率が上がるので、ブロックデバイスとして定義されている。

ちなみに、IDEハードディスクドライブは、ブロックデバイス、メジャー番号3が割り当てられている。しかし、これをキャラクタデバイスにしたからといって、バッファリングしないディスクドライブになったりはしない。メジャー番号3のブロックデバイスは、IDEハードディスクドライブだが、そのキャラクタデバイスはPseudo-TTY slavesが割り当てられている。そもそもキャラクタデバイス定義のディスクドライブなんてものはない。

つまり、メジャー番号とマイナー番号、そしてデバイスタイプのすべてが固定的に定義されたもので、変更するには、カーネルのソースコードを書き換える必要がある。なお、メジャーデバイス番号の240から254は、ローカル／実験用途で利用できるよう定義されている。必要であればこれらのデバイス番号を使用して実装することになる。まぁ完全に閉じた環境であれば、自分で自由にすべてのデバイス番号を定義して使ってもよいのだけれども。

Linuxのデバイス番号定義は、/usr/src/linux/Documentation/devices.txtに記述されているので、適宜参照してほしい。

上記以外の特殊デバイスとしては、デバイスタイプFIFOというものがある。これはプロセス間通信のパイプとして機能する特殊なファイルだ。Linuxの/dev/initctlは、このFIFOデバイスで、initプロセスやいくつかのカーネルスレッドデーモンサービス間の情報のやり取りに利用されている。

■デバイスファイル

各種デバイスは、内部的にデバイス番号として定義されているわけだが、これをユーザーやソフトウェアが操作するためには、それ相応のかたちとして提供されなければならない。

先ほど述べたように、Linuxでもデバイスファイルは、/devディレクトリに特殊ファイルとして格納される、というか、誰かが定義されたファイルを格納しておく必要がある。

デバイスファイルは、mknodコマンド（関数）で、そのデバイス名（ファイル名）、デバイスタイプ、メジャーデバイス番号、マイナーデバイス番号を指定して、ファイルシステム上のファイルとして作成することができる。なお、このファイルの作成には、管理者権限が必要となる。

たとえば、IDEハードディスクドライブのディスク1、パーティション3は次のようにして作成できる。

```
# mknod hdb3 b 3 67
```

hdb3というファイル名、デバイスタイプはブロック、メジャーデバイス番号は3、マイナーデバイス番号は67を指定している。これらの番号は先ほど述べたように、デバイスに対して適当に割り当てたものなので、深く意味を考えても無駄だ。各OSのデバイス番号定義リストを参照するしかない。

Linuxでは、IDEインターフェイスのプライマリは、メジャーデバイス番号3で、そのスレーブドライブはマイナー番号64からパーティションごとに1ずつインクリメントしていくようになっているだけなのだ。ちなみに、IDEインターフェイスのセカンダリは、メジャーデバイス番号22と定義されている。このセカンダリには、メジャーデバイス番号3のマイナーデバイス番号の128からを割り当ててもよさそうなものだが、ハードウェアの管理上、分かれている。また3つ目、4つ目のIDEインターフェイスは、メジャーデバイス番号33と34で、10個目のインターフェイスまで定義されている。あと、パーティションに関しては、1から63までの63個までのサポートとなる（こんなに使ったりしないけどね）。

まぁ、デバイス番号の定義の順番は、必要になった順と考えてもいい。必要になったから順次拡張していったというわけで、一応その性格上、分かりやすいように並び替えるタイミングがなかったのである。配布するカーネルソースコード内での整合性は、特に難しい作業ではないが、既存のシステムの/devディレクトリのデバイスファイルを作り直せというのは、一般ユーザーにはかなり酷な作業となる。ファイルシステムはそのままに、カーネルだけアップグレードしていくユーザーは多い。だから場当たり的な拡張のまま進むしかないというわけだ。

さてデバイスファイルは、共通的にアクセスできるよう/devに作成されているが、ファイル自体は、なにも/devディレクトリにしか作成できないというものではないので、各ユーザーのワークディレクトリで試してみるとよいだろう。削除は、rmコマンドでできるのであまり心配する必要はない。

でもまぁ、通常デフォルトで作成されているので、

/dev/hdb3を見るだけでもいい。

```
$ ls -l /dev/hdb3
brw-rw---- 1 root disk 3, 67 8月24 18:00 /dev/hdb3
```

この情報は結構重要なので説明しておく。まず先頭の通常ファイルのパーミッションを表示する部分なんだけど、ここにはデバイスタイプとそのデバイスのパーミッションが表されている。bはブロック、cはキャラクタ、pはFIFOデバイス。続くパーミッションは通常のファイルと同じなので詳しくは説明しないけど、オーナー、オーナーグループ、そしてそれ以外のユーザーに対するアクセス制御がそれぞれ設定される。デバイスファイルでは、実行属性は意味がないので、設定されることはないが、それぞれにリード、ライトのアクセス制御がなされる。上記の例では、rootユーザーとdiskグループはリード／ライトともに操作できるが、それ以外のユーザーは直接扱うことはできないということを表している。

パーミッションの次は、リンクカウントとオーナーユーザー、オーナーグループで、通常ファイルのファイルサイズが表示される部分に、デバイスファイルでは、メジャーデバイス番号とマイナーデバイス番号が表示される。あとは作成した日時とファイル名（デバイス名）。

つまりデバイスファイルは、そのデバイス番号を特定するとともに、そのデバイスへのアクセス制御も同時に提供しているのである。

ここで説明したデバイスファイルの振る舞いについて理解するためには、適当なユーザーのホームディレクトリにデバイスファイルを作成し、そのパーミッションを調整して、実際にアクセスしてみるとよいだろう。以下は、/dev/hdaと同等のデバイスファイルをtestという名前で作成し、そのパーミッションを666に変更している。ここまでは、管理者権限で行う必要があるので、sudoコマンドを利用している（単純にsuでrootになってもいいけど）。その後、通常ユーザーの権限下で、fdiskコマンドでそのファイルをオープンしている。ここでは不必要な操作でhdaのパーティションテーブルを破壊しないよう気を付けながら、操作してみてほしい。最後はテストデバイスファイルの削除（testのオーナーはrootのはずだが、そのディレクトリの操作権限があれば、rootでなくてもファイルは削除できるので、suしたりする必要はない）。

```
$ sudo mknod test b 3 0 -m 666
$ /sbin/fdisk test
$ rm test
```

さて、デバイスファイルがデバイスであるということは、どのレベルで判定されるのだろうか？

これは実装によって異なるが、Linuxでは、VFS（仮想ファイルシステム）層で処理されるようになっている。

デバイスファイルと通常のファイルとの違いは、その属性で、これをもとにVFS層で判別され、予め定義されているデバイス一覧と照会して、必要なデバイスのハンドルが作成されるのである。デバイスファイルの名前は、ユーザーやアプリケーションにとっては意味があるが、内部的には名前は意味を持たなくなり、その代わり、そのデバイス番号とデバイスタイプによって、判別されるようになる。

これらの属性情報は当然、ファイルシステム内に格納されていなければならない。Linux標準のext2ファイルシステムでは、i-nodeにこれらの情報を格納する仕組みがある（特殊ファイル／デバイスタイプ用のフラグが用意されていて、デバイス番号は、各ファイルノードの使用ブロック番号の格納位置に収められる）。この仕組みはReiserfsにもある。でもFAT／FAT32やNTFSにはない。

だから、デバイスファイルを格納する/devディレクトリを作成するファイルシステムには、制限がある（通常ルート（/）と/devを分離することは少ないからルートファイルシステムの制限といってもよいのだが）。たとえば、NTFSの読み書きやアクセス制御の操作が問題なくサポートされたとしても、デバイスファイル（ノード）の格納がファイルシステムとして、いっさい考慮されていないため、特別なトリックを実装しない限り、利用できない（まぁFATでも、ファイルの内容データか、日時格納部の一部を勝手に利用して、デバイス番号を埋め込み、それを解釈するような機能を実装すればよいのだけれど）。

何が問題？

さて、ここまでの説明で、冒頭のLinuxのデバイス管理はダサいと述べた真意は分かっただろうか？　たぶん、ほかのシステムでのデバイス管理の実装を知らないことには比べようがなく、評価できないかもしれない。

最近のオペレーティングシステムの傾向としては、必要なデバイスファイルのみを必要に応じて提供するというものだ。ハードウェアデバイスは、システムをスキャンする、もしくはデバイスドライバの初期化時にほとんど特定さ

れ、どのデバイスがユーザーに提供できるか本当はわかっている。ハードウェア以外の仮想デバイスに関しても、その機能コンポーネントが初期化される段階や、要求があった時に作成したとしても特に問題ない。

これらを実現するために、多くの実装では、ソフトウェア的にデバイスファイルを提供しているのだ。/procの/dev版と考えるとわかりやすいと思う。/procでは、カーネルがシステムやプロセスの状態をソフトウェア的にファイルシステムとして提供し、一般ユーザーアプリケーションからそれらを参照したり、操作したりできるようにする仕組みだ。本当のファイルシステム上には、マウントポイントの/procディレクトリがあるだけで、/procの実体は、カーネルコンポーネントとしてメモリ上に保持し、アクセスに応じて、その処理を行い、結果を返すようになっている。マウントは、/etc/fstabに記述されていることからもわかるように、カーネルの初期化後、通常のファイルシステムをマウントするのと同じ段階に行われる。

#/etc/fstabの抜粋

```
none  /proc  proc  defaults  0 0
```

現状のLinuxでも一部のデバイスは、似たような仕組みで処理されている。それは、/dev/ptsという仮想端末用のデバイスだ。Linuxカーネルのバージョンやその設定にもよるが、最近の一般的な設定では、/etc/fstabに次の項目があると思う。また、カーネル設定では、CONFIG_DEVPTS_FSが有効になっているハズだ。

```
none  /dev/pts  devpts  gid=5,mode=620  0 0
```

これはUnix98 PTYという仕様の仮想端末手法を実装したもので、/dev/ptmxというマスターになる仮想端末（Pseudo Terminal Master）をオープンすると、必要に応じて、/dev/pts/0、/dev/pts/1といった具合に実際にアクセスできる仮想端末のスレーブ側を作成し、それをソフトウェアに返すことができる。そのため、ソフトウェアでは、使用していない仮想端末を検索したり、特定の仮想端末を予約したりしておく手間が省けるのだ。これは、Cランタイムライブラリglibc 2.1で実装されている。またLinuxカーネルとしても、この仕組みをサポートするために、devptsという仮想ファイルシステムを実装し、先ほどのように/dev/ptsにマウントして、動的な仮想端末の作成が容易になるよう支援しているのである。

しかし、Linuxの現状としては、/devディレクトリは従来のままだ。カーネルのソースコードでデバイスを内部的に処理するためのデバイス番号が定義され、デバイスドライバレベルでそのデバイスの操作が可能な状態であったとしても、/devディレクトリにそれを表すファイルがなければ、そのデバイスを操作することができないし、そのデバイスファイルは、管理者か誰かが作成しない限り、決して存在しない。また、実際に/devディレクトリを見てみるとわかるが、使いもしないデバイスファイルが不毛に大量に作成されているのである。

試しに僕の環境の/devディレクトリのファイル数をカウントしてみると次のようになる。

```
$ ls /dev | wc -l
   6241
$ ls /dev -R | wc -l
  15259
```

/devディレクトリ直下に6000以上のファイル、サブディレクトリを含めると1万5000以上のデバイスファイルが作成されている。ここで実際に使用しているものをチェックしてみたいと思う（ptsとttyは多いのでまとめた）。

```
$ su -c "/usr/sbin/lsof | grep \/dev\/ | cut -c 60
| sort | uniq"
/dev/console    /dev/gpmctl
/dev/initctl    /dev/log
/dev/null       /dev/psaux
/dev/ptmx       /dev/pts/0〜4
/dev/tty1〜6
```

これらはソフトウェア的にオープンしているデバイスファイルで、デーモンサービスもそれほど多くなく、プロセス数40個程度、リモートからktermを数個開いている僕の環境だと、この程度である。上記では、lsofというコマンドを使用しているが、/procをls -lRでスキャンしても同様の結果が得られるので、コマンドがない場合は、そちらを試してみるとよいだろう。ちなみに、gpmctlとlogはデバイスファイルではなく、通常のファイル。

またここには、ファイルシステムとしてマウントしているハードウェアデバイス/dev/hda1などは表示されないので、ハンドリングしているすべてのデバイスが把握できるわけではない。とりあえず、/etcにある設定ファイル群を

grepしてみると、それなりにハードウェアデバイスの利用状況（利用するかもしれない）がわかると思う。

結果として、大きく見積もっても、/devで利用する可能性があるデバイスファイルは、僕の環境では50程度でしかない。1万5000以上のデバイスファイルのうち、たったの50である。すべての利用者の必要とするデバイスファイルの和であるため、こんなに大量のデバイスファイルとなってしまう。ユーザーに必要になったら勝手に作成しろ、というわけにもいかないから、仕方のないことではある。でも/devに、いかに無駄なデバイスファイルが定義されているか、理解できると思う。ファイルシステムのi-nodeが無駄に消費されてしまうとか、ファイルの検索に多少なりとも時間がかかるだろ、といったセコイことは、今さら強くは言わないけど、もう少しスマートにならないと、ねぇ。まぁ、不要なものは各自消しちゃえばいいんだけどね。

Devfs（Device File System）

実は、Linuxにもデバイスファイルを仮想ファイルシステムで提供しようという試みがある。Devfs（Device File System）がそれだ。現状、実験の段階だが、基本的な主要デバイスファイルは提供されるようになっている。最終的なデバイスファイルシステムの実装には、単に/devを仮想ファイルシステムとしてメモリ上に構築するだけではなく、アプリケーションからのイベントに応じて、各デバイスドライバが必要なデバイスファイルをその名前空間に登録／削除できなければならないので、完全な実装となるには、もうしばらくかかりそうだが、これを利用すると大量の不毛なデバイスファイルの管理や新しくデバイスファイルを作成する手間など、従来の問題点がすべて解決できる。

Devfsの本体は、Linuxカーネル2.4に含まれている。またオプションコンポーネントのdevfsデーモンプログラムは、このデバイスファイルシステムの開発者であるRichard Gooch氏のホームページ（http://www.atnf.csiro.au/~rgooch/linux/）から入手可能だ。また、Devfsの詳細は、/usr/src/linux/Documentation/filesystems/devfs/のファイルやhttp://www.atnf.csiro.au/~rgooch/linux/docs/devfs.htmlを参考にしてほしい。

Devfsを使ってみる

さて、Linuxカーネル2.4にそのコードは含まれているので、実際に使用してみることにしよう。Devfsを使用するには、まずカーネル構成の次の項目を有効にする。

```
File systems  --->
[*] /dev file system support (EXPERIMENTAL)
[*]   Automatically mount at boot
[*]   Debug devfs
```

単にデバイスファイルシステム機能を有効にし、/devへのマウントを手動で行う場合は、“Automatically mount at boot”は無効のままでよい。これを有効にすると、手動もしくはfstabなどで、/devにデバイスファイルシステムをマウントする設定を行わなくても、カーネルの初期化時に自動的にマウントされるようになる（今回はこちらを試す）。マウントはカーネル内部のファイルノードの上書きなので、マウントポイントとなる/devに手を加える必要はない。そのディレクトリにファイルがあろうとなかろうと、マウントできるし、その内容に手が加わるものでもない。“Debug devfs”を有効にすると、イベントに応じてデバイスドライバが作成されるなどの状況を表すメッセージがカーネルログに書き出されるようになる。最初のテスト時には、有効にしておこう。

カーネル構成の設定としてはこれだけで、あとはコンパイルして出来上がったカーネルイメージを実行すればよいのだが、必ず問題のないカーネルでの起動方法を確立しておきたい。cp arch/i386/boot/bzImage /dev/fd0でテスト用のブートフロッピーを作成するか、既存の/bootのvmlinuxとSystem.mapの組み合わせを適当な名前（vmlinux.okとSystem.map.okとか）にし、それらをlilo.confに登録しておくと、ある程度は大丈夫だと思う。できればルートファイルシステムも完全に別にしておきたいが、まぁそれは状況に応じてでいい。

じゃぁ、再起動、といきたいところだが、あと少しだけ修正しておくことがある。修正の必要があるのは、/etcのinittabとfstab、あと必要に応じてXF86config。

まずinittabでは、仮想コンソールの設定を修正しておく。“1～6:2345:respawn:/sbin/mingetty tty1～6”の設定があると思う。デフォルト状態のデバイスファイルシステムでは、tty＜番号＞は、vc/＜番号＞として提供されているので、これを修正する必要がある。inittabの設定のすべてを修正してしまうと、問題があった場合に困るので、とりあえず4～6を修正しておく。この設定では、Linuxシステムの初期化後、コンソールが表示されないと思う

（tty設定の1から3の仮想コンソールはエラーになるから）。その時は、おもむろにAlt＋F4～6でコンソールを切り替えて使ってほしい。なお、逆に通常のカーネルに戻すと、仮想コンソールの4から6が使えない状態となるので忘れないようにしておこう。

```
4:2345:respawn:/sbin/mingetty vc/4
5:2345:respawn:/sbin/mingetty vc/5
6:2345:respawn:/sbin/mingetty vc/6
```

この設定をし忘れているとローカルのコンソールにはログオンできなくなるので注意。でもリモートからのptsに対するログオンは、問題ないので慌てなくてもよい。

X環境を利用している場合は、マウスデバイスが問題になるので、少し/etc/X11/XF86Configなどの設定ファイルを修正しておく。通常は/dev/mouseがマウスデバイスとして設定されていると思うが、これをPS/2マウスの場合は/dev/misc/psauxといった具合に変更する（USBマウスは何だったかなぁ、次回までに調べておくね）。まぁデバイスファイルシステムに限ったことではないが、カーネルのテストをするのにX環境を起動する必要もない（あとでテストすればいい）ので、INITレベル2か3にしておくのが無難。

あと、最も重要な設定変更をfstabに行わなくてはならない。ひとつは、devptsのマウント設定の解除、もうひとつは、ハードディスクパーティションの特定方法の変更。

デバイスファイルシステムでは、それ自身でdevpts相当のサポート機能があるので、devptsをマウントしてはいけないのである。だからこの項目は先頭に＃を追加してコメントアウトしておこう。この設定だと、通常のカーネルに戻したときに問題となるが、仮想コンソールのttyは生きなので、コンソールにログイン後、fstabを修正して、mount -aとすればdevpts仮想ファイルシステムをマウントできるので、そう気にすることはない。

最後にハードディスクパーティションの特定方法の変更。デバイスファイルシステムでは、hdaといったデバイスファイルは提供されず、次のように存在するデバイス（検出されたデバイス）のみ提供されるようになっている。

```
/dev/ide/host0/bus0/target0/lun0
brw------- 1 root root 3, 0 12月31 1969 disc
brw------- 1 root root 3, 1 12月31 1969 part1
brw------- 1 root root 3, 2 12月31 1969 part2
```

discはhda、hda1はpart1に相当する。しかもディレクトリを見ればわかるようにハードウェア構成をそのまま表現した状態となる。そのため、/dev/hda1は、/dev/ide/host0/bus0/target0/lun0/part1と指定する必要がある。

このことは両カーネルをテストする際、非常に問題となるのだが、最近のmountコマンドは、ボリュームラベルでデバイスを指定することができるので、障害は一気に解決する（拡張パーティションの論理ドライブは、その特定が難しいので、このラベル指定が実装された）。ラベル指定では、上記のデバイス名を使用する必要はなく、次のようにfstabで指定する。

```
＃fstab
LABEL=root  /  ext2  defaults  1 1
```

fstabでデバイスの指定が、LABEL=となっていれば、通常のカーネルでも、デバイスファイルシステムを有効にしたカーネルでも問題なくファイルシステムのマウント処理が行える。もしこの設定をしていないとカーネルの初期化直後のファイルシステムマウントでシステムの初期化処理が停止してしまうので注意したい。なお、ext2ファイルシステムでは、e2labelもしくはtune2fsコマンドでデバイスのボリュームラベルを設定できる（でもReiserfsでは、どうやってラベルを指定すればよいのやら…、とほほ、だから今のところマウントは手動なのねん）。

最低限変更しなければならない項目は以上である。

起動後、まずは/devディレクトリを参照してみてほしい。

```
[ /dev]$ dir
console  full  kmem  misc  ptmx  random  sound  vc
cpu  ide  log  null  pts  root  tty  vcc  discs
initctl  mem  port  pty  shm  urandom  zero
```

とってもシンプルですっきりしている。あとは、順次必要なプログラムを動作させ、問題が生じるかどうかチェックしていこう。僕の環境では、そろそろ完全移行してもいいかなと思うくらい、すこぶる快調に稼動している。

●

というわけで次回は、Devfsの詳細とそのDevfsをサポートするデーモンプログラムを見ていく予定。

PDFファイルも簡単に作成できるようになった
プロフェッショナルユース組版システム

EWB

1999年10月、EWB（Editor's Work Bench）システムが、オープンソース・ソフトウェアとして一般公開された。もともとはアスキーが、文書の電子化を推進する目的で社内用に開発したシステムである。完全ページアップが可能で、さらに面付けもできるようになっている。アスキーが出版している書籍の多くがEWBを利用して作成され、雑誌でもスペック表のような定型ページに用いられたりもしている。今回は2001年1月に、PostScript／PDFへの対応をメインとしたバージョンアップが行われたのに合わせ、簡単な解説をしてみたい。なお、EWBはWeb（http://www2.ascii.co.jp/EWB/）からダウンロードできるほか、付録CD-ROMにも収録されている。

文：中野ケン
Text : Ken Nakano
株式会社アスキー 出版技術部

EWBの特徴

EWBは本を作るためのシステム、つまり組版システムである。EWBは、次のような特徴を持っている。

- **マークアップ方式**
- **簡単なトリガ**
- **文章とレイアウトの分離**
- **TeXによる組版**
- **PostScript／PDFへの対応**

●マークアップ方式

一般に組版システムは、文章のレイアウト作業をどのように行うかで、DTP（WYSIWYG）方式とマークアップ（バッチ）方式とに大きく分類される。DTP方式は、ディスプレイ上で対話的にレイアウトしていく方式で、Quark ExpressやFrameMakerなどに代表される。対して、マークアップ方式は、レイアウト処理用の記号を文章に埋め込み、それをプログラムで一括処理する。TeXやXMLなどがこの方式に属する。EWBもマークアップ方式だ。

DTP方式とEWBでの作業は**図1**のような違いがある。

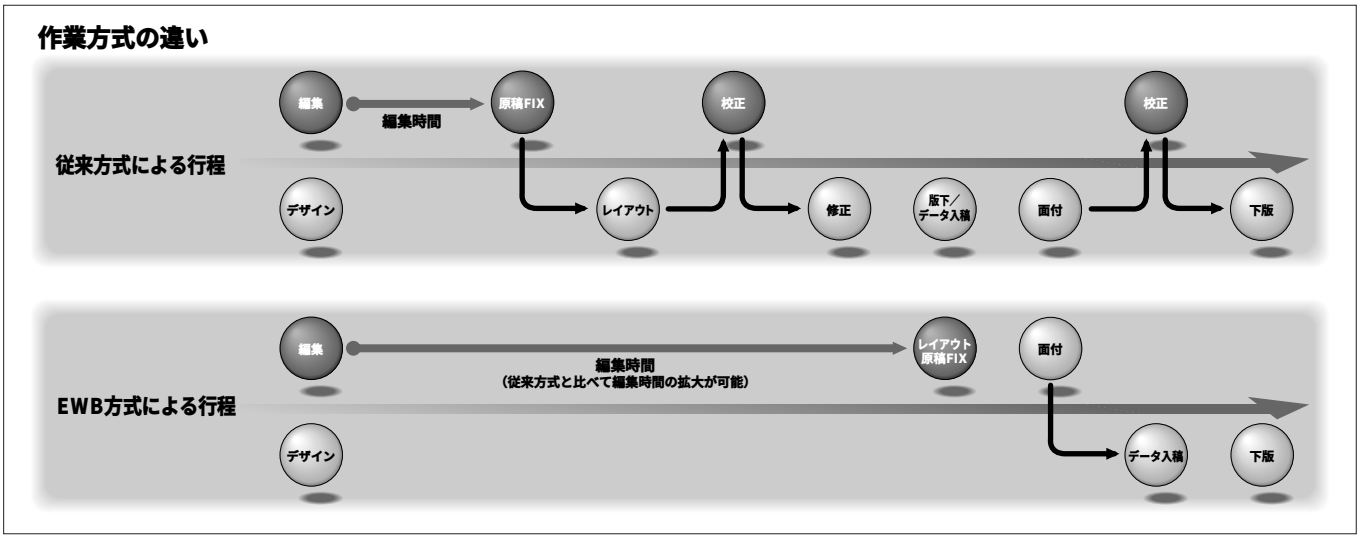


図1 DTPとEWBでの作業の比較

DTPでは、通常、原稿を編集する編集者と、その文章をレイアウトする制作者とが共同で作業を行う。レイアウト後の文章の修正は、編集者が紙の上で行い、制作者がそれを反映をしていく。現在のように、市場への早急な対応が望まれてくると、このやりとりの手間や時間がボトルネックになってくる。一方、EWBでの作業は、編集者は手もとの原稿ファイルを自分で修正するだけで済む。やりとりが少ないぶんだけ、手間も時間もかからない。逆に、同じ時間を与えられたのならば、より丁寧に文章を校正する余裕が生まれる。

●簡単なトリガ

メールや文章を書くとき、○や□などで印を付けたり、＊＊で語句を強調したり、マイナス記号を連続させたラインで区切ったりして、相手が読みやすいように工夫することがある。マークアップ方式で文章に記号を付けるのも、これとたいして変わらない。しかし、マークアップ方式は、記号の入力が面倒だったり、数が膨大だったり、規則が細かかったりすることが難点とされる。

そこで、EWBでは誰もが容易に覚えることができ、すぐに利用することができるよう、記号（トリガ）をシンプルにし、規則もワープロで原稿を作成するときとほとんど変わらないようにしている（**表1**、**リスト1**）。

ここでとくに重要なのは、トリガがレイアウト用の指定をするための命令としてでなく、文章の骨格を示すように意味付けられている点である。つまり、文章を執筆したり、編集したりするときに、レイアウトのことを気にするのではなく、その文章における、内容上の位置付けに注意がいくように設計されているのだ。文章の内容を読み手に伝える、いわばコミュニケーション言語としての役割を担っているのである。

最終的にはDTPソフトで組版されてしまうにしても、EWBのトリガを利用して編集作業をし、そのままEWBのトリガの付いた原稿ファイルを制作者に渡す編集者もいる。渡された制作者側も、トリガをマクロなどで自動処理するような工夫をして、手作業を軽減させることも可能だ。

もちろん、表1のトリガがすべてなのではなく、必要に応じてトリガを定義できるようにもなっている。

//i	大見出し
//ii	中見出し
//iii	小見出し
//g{〜//g}	語句の強調
//k1{〜//k1}	箇条書
//c1{〜//c1}	小組
//list{〜//list}	プログラムリスト
//f番号	図の指定と参照
//t番号	表の指定と参照
//url{〜//url}	ハイパーリンクの指定

表1　EWBのトリガ

●文章とレイアウトの分離

表1のトリガは文章の意味上の役割を示しているだけで、レイアウト的な意味は何もない。たとえば「//g」は強調してほしいという以外のことは何も伝えない。それが新ゴシックであろうが、太ゴシックであるかは文章の内容からすれば関係ないから当然だ。トリガがレイアウト上でどのように表現されるのかは「スタイルファイル」で行う。

よく、コンピュータ言語の仕様書などで関数の引数を特別な書体で示すことがある。このとき、原稿ファイルでは「//arg{x//}」のように特別なトリガでその語句を示しておく。すると、最初はイタリックで表示しようと思ったけれど、後からスラント体にすることにした場合も、原稿ファイルはいっさい修正することなく、スタイルファイルの定義を変更

▼リスト1　原稿ファイルの例

```
//i EWB

//ii はじめに
　1999年10月、EWB（Editor's Work Bench）システムが、オープンソース・ソフトウェアとして一般公開された。もともとはアスキーが、文書の電子化を推進する目的で社内用に開発したシステムである。開発当初の組版機能は文字原稿のいわゆる棒打ち出力だけであったが、pTeXや周辺ツールの開発と改良を重ねるにつれ、完全ページアップも可能となり、さらに面付けもできるようになっている。アスキーが出版している書籍の多くがEWBを利用して作成され、雑誌でもスペック表のような定型ページに用いられたりもしている。
　2001年1月に、PostScript／PDFへの対応をメインとしたバージョンアップが行われたのに合わせ、簡単な解説をしてみたい。なお、EWBはWeb（http://www2.ascii.co.jp/EWB/）からダウンロードできるほか、付録CD-ROMにも収録されている。

//ii EWBの特徴
　EWBは本を作るためのシステム、つまり組版システムである。EWBは、つぎのような特徴を持っている。

//k1{
・マークアップ方式
・簡単なトリガ
・文章とレイアウトの分離
・TeXによる組版
・PostScript／PDFへの対応
//}

//iii マークアップ方式
　一般に組版システムは、文章のレイアウト作業をどのように行うかで、DTP（WYSIWYG）方式とマークアップ（バッチ）方式とに大きく分類される。DTP方式は、ディスプレイ上で対話的にレイアウトしていく方式で、Quark ExpressやFrameMakerなどに代表される。対して、マークアップ方式は、レイアウト処理用の記号を文章に埋め込み、それをプログラムで一括処理する。TeXやXMLなどがこの方式に属する。EWBもマークアップ方式だ。
　DTP方式とEWBでの作業は//f100のような違いがある。
　DTPでは、通常、原稿を編集する編集者と、その文章をレイアウトする制作者とが共同で作業を行う。レイアウト後の文章の修正は、編集者が紙の上で行い、制作者がそれを反映をしていく。現在のように、市場への早急な対応が望まれてくると、このやりとりの手間や時間がボトルネックになってくる。
```

するだけで済んでしまう（**画面1、2**）。

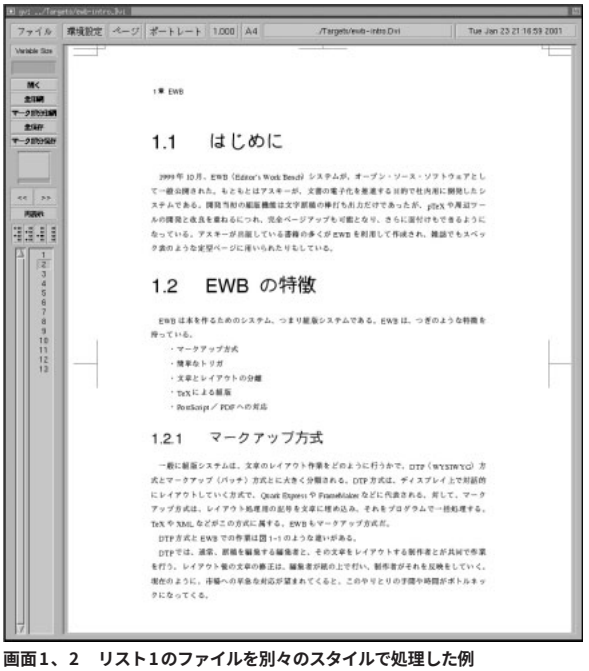
DTPのように文章に具体的な指定を行っているのでは、こうはいかない。すべての変更部分を読んで確認し、手で修正する必要がある。DTPは、現実の作業をそのままコンピュータ上で行っているために、アプリケーション操作は簡単だが、効率的に行える部分も少ないのである。

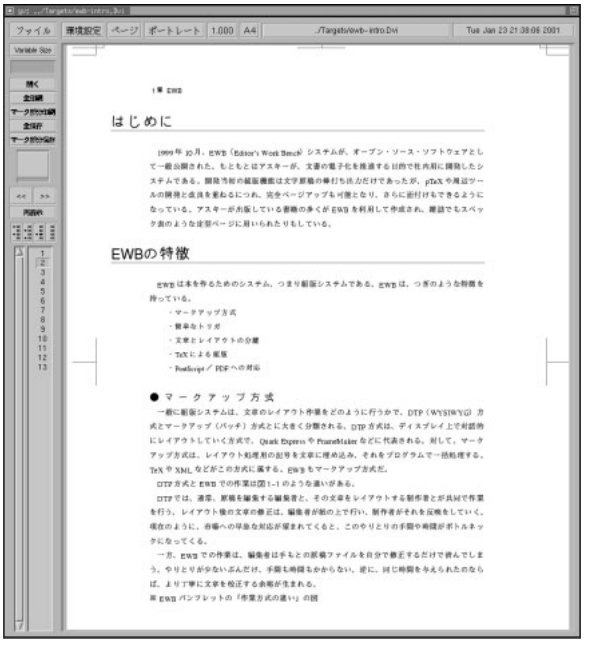
また、マークアップ言語同士には、それぞれで同じ意味を持つ記号を提供していることが少なくない。たとえば、見出しを表す記号としてHTMLには「H1」タグ、LaTeXでは「\chapter」コマンドがある。EWBのトリガを、プログラムによって、それぞれのマークアップ言語の記号に置き換えるだけで、ひとつの文章をさまざまな形式で読者に提供できるわけだ。もちろん、すべての記号が対応するとは限らない。しかし、全部を手作業で変換するよりは、明らかに効率的である。

●TeXによる組版

TeXはUNIX上で広く用いられているフリーの組版ソフトだ。ほとんどのLinuxディストリビューションに付属している。数式組版の能力や、強力なマクロ機能、柔軟性などで評価が高い。その一方で、記号付けの規則が細かくて大変、コマンド操作が面倒、スタイルファイルの作成が難しいともいわれる。

EWBでは、文章への指定はトリガによって簡略化してい





画面1、2　リスト1のファイルを別々のスタイルで処理した例

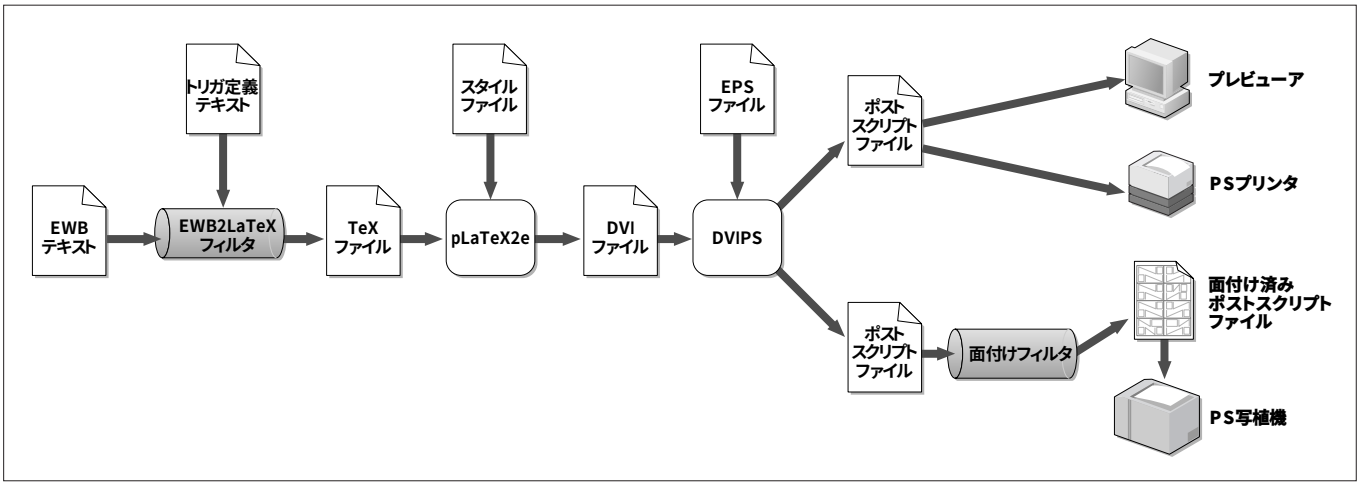


図2 EWBでの処理の流れ

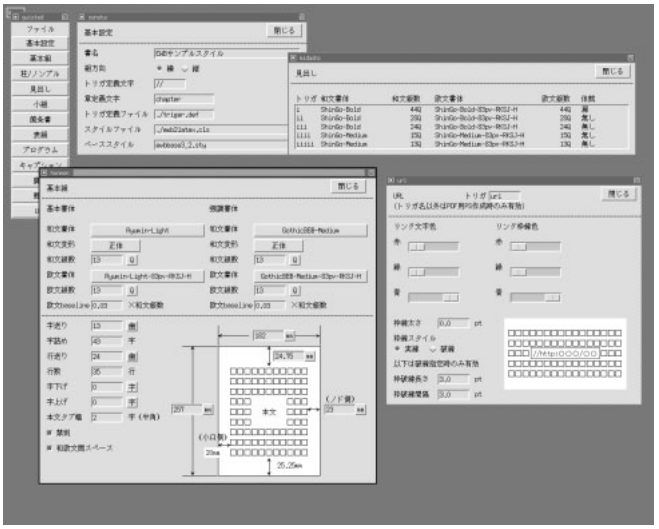
る。それをプログラムでTeXの文書形式に変換し、組版をしている（**図2**）。加えて、原稿ファイルの変換から、PostScriptファイルを作成するまでの操作を対話的に行うための環境を用意している。このインターフェイスを用いると、UNIXのコマンドオペレーションに慣れていないユーザでもスムーズに作業を進めることができる。

スタイルファイルの作成についても、基本組、見出し、小組などの書体や級数、字数といった基本的な体裁を対話的に設定できるようなツールを提供している（**画面3**）。

なお、EWBではTeXのマクロ機能によって、自動的に分割される囲み罫マクロなどを用意している。たとえば、コラムやプログラムリストの周りを罫線で囲むようなレイアウトの場合、それらが1ページに収まるのならば対応できるソフトは多い。けれども、複数ページにまたがる囲み罫を自動的に分割し、レイアウトするソフトはほとんどない。このときは手作業で対処しなければならず、とくにマークアップ方式ではかなり面倒で負担の大きい作業である。EWBでは、自動的に分割する機能を備えることによって実現している。このように、TeXに強力なマクロ機能を用いて組版処理をカスタマイズして、文章に指定するトリガの数や面倒な作業を少なくすることが可能である。

●PostScript／PDFへの対応

EWBでは、最終的な組版データはPostScript形式としてファイルに格納する。WYSIWIG方式の組版ソフトでも最終的にはPostScript形式で出力される。以前は、利用できる日本語書体数やデザインなどから、敬遠されることも多かったが、もはや主流となり、出力環境の充実も著しい。たとえば、従来は出力データは印画紙に出力され、フィルム、刷版の工程を経て、印刷されていた。Postscript環境では、フィルムに直接出力するのも一般的である。さらに、出力データを刷版に直接出力するCTP（Computer To Plate）も増えてきている。中間工程が減る分、時間もコストも抑えられる。もちろん、EWBで組版したデータもCTPで出力可能だ。



画面3　レイアウト設計ツール

PostScriptデータは、Acrobat Distillerを使って、簡単にPDFファイルに変換することができる。

PDFファイルは、Postscriptと比べ、ファイルサイズが小さく、各ページへの参照も効率的にできるため、組版された文章をネットワークやCD-ROMなどで配付するのに最適な形式である。ビューアであるAcrobat Readerも無料で配布されている。

そして、ブックマークのようなしおりを付けたり、ハイパーリンクで別のWebページやPDF文書を表示することも可能である。ただし、このような付加価値は、たいていの場合、DistillerでPDFへと変換したのち、Acrobat Exchange上で手作業によって付与されている。けれども、これでは、本文に修正が入ると、再度PDFを作成し、またもやExchange上で同じ作業をせざるを得ない。

そこで、今回のバージョンアップは、これらの機能の作業の自動化を目指して行われた。PDF上でのハイパーリンクのために、著者あるいは編集者がすべきことは、リンク先の文字列をハイパーリンクトリガ（//url）で指定するだけである。ただし、EWBでは二重スラッシュをトリガとして認識するので、URL表記のところは、次のようにエスケープしなければならない点に注意が必要だ。

```
http:////www2.ascii.co.jp/EWB/
```

索引や目次に現れるページ番号にもハイパーリンクを付けるようにしているが、こちらのほうはとくに何もする必要がない。EWBが勝手にやってくれる。

もうひとつの、しおり機能は、PDFを表示したときに、画面の左側に現れる。これも、たいていはハイパーリンクと同様にExchange上で作成される。しおりとなるデータも自動的に作成される。

本誌付録CD-ROMには、EWBハンドブックのPDF版を収録してある。Readerで表示すると、しおりが作成され、ハイパーリンクの設定もされていることが確認できるはずだ。目次や索引についも、自動的に該当ページへのハイパーリンクが設定されている。これらをすべて手作業でやったときの労

力は想像に難くない。

EWBをインストールする前に

EWBでは、文書処理やプレビューなどをするのに**表2**のようなプログラムを利用する。通常のワークステーションレベルでシステムを構築してあれば、Java環境以外はすでにインストールされているだろう。これらのうち、まだインストールしていないプログラムがあれば、まずはそれらをインストールしておく。Java環境は、次のサイトから入手可能である。

```
http://www.ibm.com/java/jdk/linux130/
http://java.sun.com/j2se/1.3/ja/
http://www.blackdown.com/
```

なお、**表2**のプログラムのうち、sedに注意をしてほしい。インストールされているsedがマルチバイト対応版であっても、日本語を正しく扱えるとは限らないからだ。たとえば、

```
$ echo 'ありがとうございました。' | sed 'y/あいう/アイウ/'
```

としたとき、

アリガトウゴザゥマシタ。

のように変換されてしまったら、そのsedは日本語処理用には向かない。本来、期待する動作は、

アりがとウござイました。

である。後者のようになっていなかったら、別のsedをインストールする必要がある。

ちなみに、Vine Linux 2.1に収録されているマルチバイト対応sed（sed-3.02_mb1.08-vl1）では問題なかった。

EWBのインストール

EWBのインストールはrpmコマンドを用いて次のように行う。

```
$ rpm -ivh ewbptex-3.2-R1.i386.rpm
$ rpm -ivh --replacefiles ewb-3.2-R1.i386.rpm
```

ewb-3.2-R1.i386.rpmをインストールするときに、ewbptexに含まれているファイルがいくつか重複しているため、--replacefilesオプションを付けることに注意してほしい。また、インストール時に、日本語Postscriptフォント用のVFファイルを作成するため、多少時間がかかる。

なお、ファイルはすべて/usr/local/ewb/3.2以下に置かれ、全部で250Mバイト程度のディスク容量を必要とする。

●settopdfの設定

PDFのしおりのデータを作成するのに使われるPerlスクリプトがsettopdfだ。置かれている場所は次の通り。

```
/usr/local/ewb/3.2/bin/settopdf
```

このスクリプト中でJavaインタプリタを呼び出している。インストールした状態では、このスクリプトファイルの38行目は、SunのJDK1.3を使うようになっている。

```
$JAVA = "/usr/local/jdk1.3/jre/bin/java";
```

IBM版やBlackdown版のJDKをインストールしている場合は、この行を自分のJavaインタプリタに合わせて修正する。

●ghostscriptの設定

次に、ghostscriptでの代用書体の設定を行う。設定ファイルは、kconfig.psというファイルだ。おそらく、つぎのどちらかのディレクトリに置かれている。

```
/usr/share/ghostscript/バージョン/vflib
/usr/share/ghostscript/バージョン/kanji
```

どちらのディレクトリもない場合は、

X Windowシステム	XFree86等 (X11R6以降)
X Window漢字端末	kterm
PSインタプリタ	ghostscript (日本語対応版)
PSビューア	ghostview または gv
Tcl/Tkインタプリタ	tcl/tk 8.0 (日本語対応版)
テキストエディタ	jvim (日本語対応 vi) xemacs-sumo
文字置換フィルタ	sed (日本語対応版)
perlインタプリタ	perl5 (日本語対応版)
かな漢字逆変換	kakasi
漢字コード変換	nkf
Java実行環境	JDKまたはJREバージョン1.1.8以上

表2　EWBシステムが利用するプログラム

```
$ gs --help
```

を実行して、「Search Path」の項に表示されたディレクトリのどこかにあるはずだ。

設定ファイルが見つかったら、編集する前に念のためバックアップを取っておこう。kconfig.psが別のファイルへのリンクでなければ、次のようにしてバックアップする。

```
$ cp kconfig.ps kconfig.ps.orig
```

別のファイル（たとえば、kconfig-basic.ps）へのリンクであった場合は、一度kconfig.psを削除し、リンク先のファイルをコピーしてkconfig.psを作成する。

```
$ rm kconfig.ps
$ cp kconfig-basic.ps kconfig.ps
```

設定は、**リスト2**の内容を追加するだけである。これにより、たとえば、Postscriptファイルに指定されたFutoGoB101-Boldの書体は、ghostscriptでGothicBBB-Mediumの書体で表示されることになる。

ディストリビューションによっては、すでにこれらの記述がコメントとして存在しているかもしれない。その場合は、行頭の%記号を外し、有効にするだけでよい。

商用ディストリビューションには、複数の日本語TrueTypeフォントが含まれており、その設定がなされているかもしれない。この場合は、リスト2の左側の書体名を別の書体名にすれば、プレビュー時でも、使われている書体の違いを確認するのに役立つだろう。

▼リスト2　kconfig.psの追加設定

```
/GothicBBB-Medium /FutoGoB101-Bold copycompfont
/Ryumin-Light /FutoMinA101-Bold copycompfont
/GothicBBB-Medium /GothicMB101-Bold copycompfont
/GothicBBB-Medium /GothicMB101-Ult copycompfont
/GothicBBB-Medium /Jun101-Light copycompfont
/GothicBBB-Medium /Jun34-Medium copycompfont
/GothicBBB-Medium /Jun501-Bold copycompfont
/GothicBBB-Medium /MidashiGo-MB31 copycompfont
/Ryumin-Light /MidashiMin-MA31 copycompfont
/Ryumin-Light /Ryumin-Bold copycompfont
/Ryumin-Light /Ryumin-Medium copycompfont
/Ryumin-Light /Ryumin-Ultra copycompfont
/Ryumin-Light /Ryumin-heavy copycompfont
/Ryumin-Light /Ryumin-regular copycompfont
/GothicBBB-Medium /ShinGo-Bold copycompfont
/GothicBBB-Medium /ShinGo-Light copycompfont
/GothicBBB-Medium /ShinGo-Medium copycompfont
/GothicBBB-Medium /ShinGo-Ultra copycompfont
/GothicBBB-Medium /ShinGo-regular copycompfont
```

EWBの操作

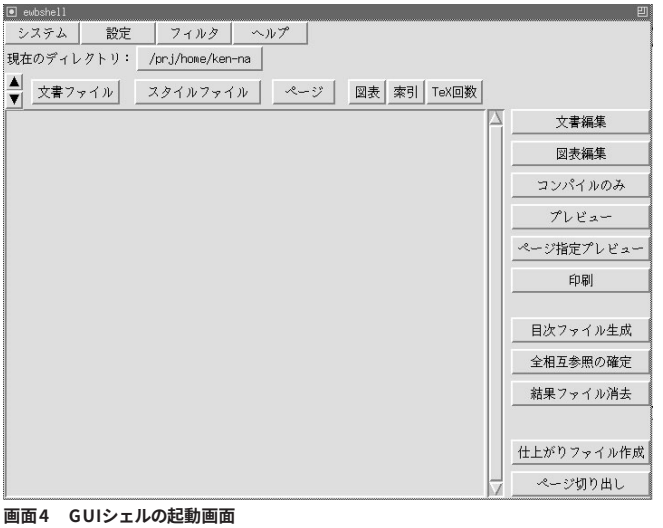
それでは、実際にEWBシステムを使ってみよう。サンプルにはEWBのマニュアルを利用する。これは、ewb-handbook.tar.gzという名前で付録CD-ROMに入っている。適当なディレクトリに移動し、このファイルを展開する。

```
$ tar zxf /cdrom/EWB/ewb-handbook.tar.gz
```

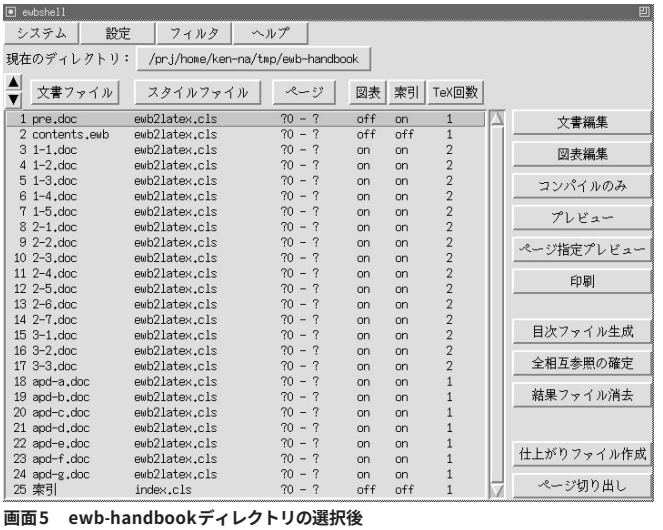
すると、ewb-handbookというディレクトリが作成され、その下に原稿ファイルや図版ファイルが置かれる。

●基本的な操作

次のコマンドを実行すると、EWBのGUIシェルと呼ばれる



画面4　GUIシェルの起動画面



画面5　ewb-handbookディレクトリの選択後

プログラムが起動する（**画面4**）。

```
$ /usr/local/ewb/3.2/bin/guishell
```

GUIとはいっても、DTPソフトのようにレイアウト作業を対話的に行うためのものではない。文書処理操作を対話的に行うための環境だ。Tcl/Tkを利用して実現されている。

つぎに「現在のディレクトリ：」の右側にあるボタンをクリックし、ewb-handbookディレクトリを選択し、OKボタンを押す。このとき、WorkディレクトリとTargetsディレクトリを作成するかどうかが問われるので、OKボタンを押す。すると、原稿ファイルのリストが表示される（**画面5**）。この段階では、まだ何も処理していないので、ページの項は、

```
?0 - ?
```

になっている。ここで、pre.docを選択し、右側の「プレビュー」ボタンを押してみよう。すると、pre.docで使われているEPSファイルの変換、EWB原稿ファイルのLaTeXファイルへの変換、コンパイル、Postscriptへの変換が自動的に実行され、最後にpre.docの処理結果がディスプレイに表示される。また、ページが決定される。

「ページ指定プレビュー」ボタンは、指定したページが最初に表示される。ただし、その原稿ファイルのページ範囲を超えている場合は、先頭ページか最終ページが表示される。「プレビュー」ボタンではなく「コンパイルのみ」ボタンでは、Postscriptへの変換までの一連の処理が行われるが、プレビューはされない。

指定した原稿ファイルがすでに処理され、組版結果が残っている場合、途中の処理は省略される。たとえば「コンパイルのみ」を実行し、原稿ファイルを処理した直後に「プレビュー」した場合、コンパイルやPostscriptへの変換は省略され、既存の結果がプレビューされるのである。前回の処理から、今回までの間に原稿ファイルを編集した場合は、その修正が結果に反映されるため、コンパイルと変換が行われ、結果ファイルが更新される。

●PDF用Postscriptファイルの生成

PDF用のPostscriptファイルを作成するのも簡単である。「仕上がりファイルの作成」ボタンを押す。すると、**画面6**のダイアログが表示され、「PDF用のPSファイル作成」ボタンを押すと、**画面7**のダイアログが表示される。

ここで［作成」ボタンを押すと処理が開始され、終了時に、「TargetsディレクトリにPDF.PSを作成しました」というメッセージが表示される。このファイルをWindowsあるいはMacintosh上のDistillerで処理することで、PDFファイルが完成する。

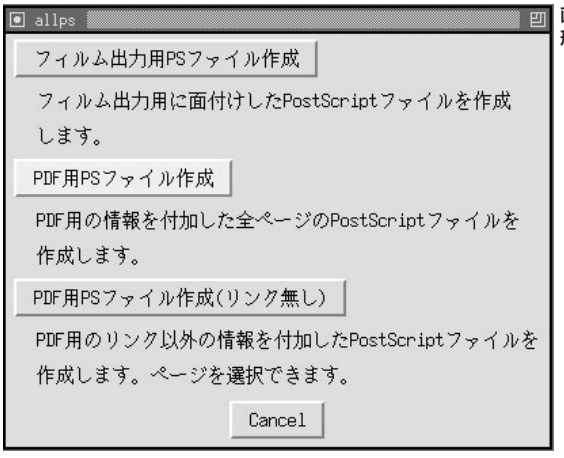
●文書の編集

原稿ファイルを選択し、「文書編集」ボタンを押すと、XEmacsが起動する（**画面8**）。

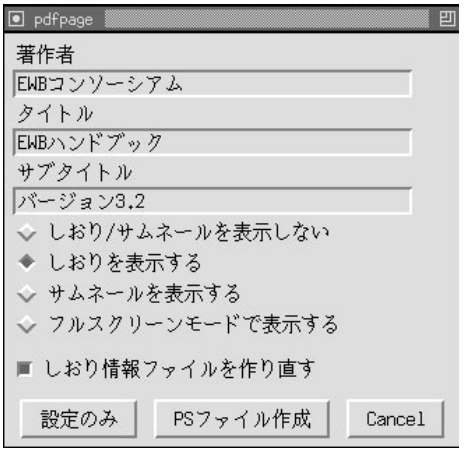
このXEmacs上では、文書を編集するほかに、索引語の指定を対話的に行えるようになっている。今回は、索引語の登録に絞って説明をしよう。

画面は上下に分割されていて、上部に指定した原稿ファイルの内容、下部には文書全体（この場合はewb-handbook）で使われている索引語の内容が表示されている。

索引語で何が面倒かといえば、その並び順を調整することである。英語ではそのままアルファベット順にすればよいのだが、日本語では漢字に複数の読み方がある。辞書順に並べるためには、言葉の読みでソートしなければならない。そのため、コンピュータで索引処理をするとき、索引語を指定す



画面6　仕上がりファイル形式の選択



画面7　PDF用の設定

るために言葉をキーボードで入力し、また読みのために同じ言葉を（今度は漢字変換せずに）入力することが多く、非常に面倒である。

「文書編集」ボタンで起動されるXEmacsには、索引付けを補助するための機能が付加されている。原稿部分で、文字列をマウスで選択し、XEmacsの「索引」ボタンを押すと、一番下にあるステータス行に、選択した文字列が表示される。そして、リターンキーを押すと、その文字列が読みに変換される。指定された文字列に漢字が含まれているならば、自動的にその読みが表示されるのである。単語に複数の読み方がある場合は、その候補が現れる（**画面9**）。ここで読みを確定するには、不必要な文字列を消す（{、|、}も）だけでいい。もし正しい読みの候補がなければ、自分で正しい読み方を入力する。

ステータス行に表示されている読み方が正しければ、リターンキーを押すと「項目2」の入力が求められる。これは、

データベース

更新 …… 36
新規作成 …… 10,25
バックアップ …… 49,87

のように、階層になっている索引を作成するためのものだ。この場合は、「更新」や「新規作成」、「バックアップ」が「項目2」にあたる。必要なければただリターンキーを押すだけでよい。項目2の単語は、CannaやWnnなどを使うか、カット&ペーストで入力する。入力してリターンキーを押すと、最初に入れた索引語のように、自動的に読みに変換される。

同様に「項目3」も必要ならば入力すると、最後に「ページ番号種別」を聞かれる。これは、TeXブックのように、特定のページをイタリック（it）にしたり、強調（bf）にしたりするときに指定する。通常は「なし」を選択する。つまり、何もしないでリターンキーを押す。

これで、索引語が登録されると同時に、原稿ファイルに索引トリガ（//in番号）が振られる。索引語の登録を途中でやめるには「×」ボタンを押す。

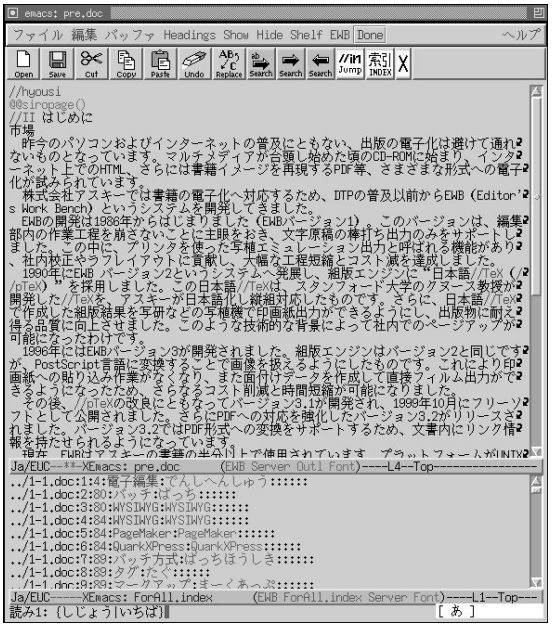
索引トリガと索引語の対応を調べるには、原稿中の索引トリガ部分をマウスでクリックし、「Jump」ボタンを押す。すると、下のバッファ部分に対応する索引語が表示される。逆に、バッファの行を選択し、「Jump」ボタンを押すと、トリガの付いている位置が表示される。

以上の手順を繰り返し、索引語の登録が済んだら、「Save」ボタンで保存をしてから、「File」メニューの一番下の「Exit XEmacs」で終了する。

EWBコンソーシアム

EWBの公開と同時にEWBコンソーシアムが設立されている。コンソーシアムはEWBの普及を通じて、電子出版のための技術の改良や標準化、情報の共有などをはかり、既存出版と電子出版のさらなる発展を目指している。詳細については、http://www2.ascii.co.jp/EWB/を参照していただきたい。

画面8　XEmacsが起動したところ



画面9　複数の読みの候補が最下行に表示される

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11回
【カーネル2.4】（かーねる・に・てん・よん）

カーネル2.4
【かーねる・に・てん・よん】

（以下は2001年1月都内某喫茶店における編集部担当と編者の打ち合わせの再録である）

気まずい雰囲気の中、先に沈黙を破ったのは編集担当者だった。「あいかわらず読者アンケートの結果がふるわないんですよね」。人生も下り坂にさしかかり、疲れきった顔をした担当が眉間のしわを一層深めながらつぶやいた。「はあ……」と怯えながらあいづちを打つ編者。担当は苛立たしそうにペンでテーブルを叩きながら続ける。「“おもしろかったページ”27位ですよ！」「……でもよかった。最下位じゃないんですね」「28位までのランキングで27位なんです！このページは。限りなく最下位じゃないですか！」「28位を取るような不人気ページがあるんですか。編集部もたいへんですね」「28位は“読者プレゼント”です！」「……」

「……読者」。しばしの沈黙をさえぎるように編者が口を開く。「読者サービスが足りないんでしょうか。お色気とか？」「これは青年マンガ誌じゃないっ！」「じゃあ、テコイレは？　実は主人公の生き別れの兄弟で、戦うためだけに生きているやたら強い宇宙人をサブキャラとして登場させるとか」「少年誌でもないっ！　だいたい“キャラ”ってなんだっ！」「こ、こわいよ。四星球、オラを守ってくれ」「守りませんっ！」

「……と、とにかく、“この用語集を信じていたら恥をかきました。恨みます。2ちゃんねるで批判スレッドを立ち上げてやる”“でたらめばっかり。男っていつもそうよね”といった苦情が毎月来るんです。ちょっとはまじめな記事を書いたらどうなんですか。たとえば今月なら、Linuxカーネル2.4についてとか」「カーネル2.4？」「……まさかカーネル2.4、知らないんじゃないでしょうね」「えっ？　知らない？は、はは、まさか。知ってますよ。カーネル2.4といえば、カーネルの2.4のことでしょう？」「ああ、よかった。実はこのあいだ、怪文書が届いたんですよ。このページの編者の家には、Linuxマシンが1台もないって内容の。ドがつく素人がでたらめを書いていいのかって。まさか、そんなはずないですよねえ？」「はははは……。いやはや、世の中にはとんでもないことを言う人もいるものですね……。うちにはありますよ。たくさん。Linux（のCD-ROMが）。」

「いやあ、とにかく今回はカーネル2.4でお願いしますよ。このコーナーがまじめな記事なんだというところを、ビシッと読者に伝えてもらえば、アンケートの結果もおのずと変わってくると思うんです」「1位も夢じゃない、と？」「1位はどうかわかりませんが」「夢の巻頭カラー進出もありえますかねっ？！」「い、いや、マンガじゃないから巻頭カラーというステイタスはちょっと……」「なんだ。ないんですか……」「……わかりました。もし1位を達成できれば、編集長に交渉してみましょう！巻頭カラーを！」「やった！　Linux magazineって、原稿料は安いけど太っ腹ですねっ！」「一言多いわっ！」

「でも……」と、編者はハッと我にかえったようにつぶやいた。「カーネル2.4なんて、どんなトピックを書けばいいんでしょう」「なに、カンタンですよ」と、海千山千の担当は、悠然とたばこをくゆらせながらアドバイスするのだった。「たとえば【スケーラビリティの向上】なんていうのも、ひとつのポイントでしょうね」「スケベ・イビリテー？」「SMPの性能が上がって、さらにメモリも64Gバイトまで対応するようになったじゃないですか。テクノロジーは着実に向上しているんです。Linuxが商用UNIXと肩を並べる日が来るのもそう遠くないかもしれません」などと一人語りをは

じめた担当をよそに、編者は手元のノートに『SMと3Pのテクが向上』とさっそくメモを取る。「ほかにもありますか」「【Raw I/Oデバイスへの対応】ですね。ま、データベースソフトウェアなどが生でI/Oすることで、今までより速く、安全にいろいろできるわけです」「なるほど」とうなずきながら、編者は『生で出したり入れたり。早く終わる可能性があるも安全』とメモを書き殴った。そして、ふとそのペンを止めてしばらく考え込んだあと、顔を上げて担当にたずねた。「こんなこと書いてしまっていいんですか……？」「なにを言うんです。これこそ読者の求めている情報ですよ！　明日のLinuxコミュニティを育むために、われわれは真実をいちはやく伝える義務があるんです！」「そうでしたね。……わかりました！　私も男です！　読者のために、あえて禁断の領域にペンを向けるも厭わず、ですよ！」「そうそう。その意気です。あとですねー、今回のバージョンアップでは対応プラットフォームも増やしていますよ。Windows CEマシンで使われているSuper Hにも対応していますしねえ」「なるほどなるほど。スーパーなHですね。かなり濃厚なんでしょうねえ」「それから【USB】【PCMCIA】そして【IEEE1394】対応も強化されましたよ。これからはプラグ＆プレイ面でも、ほかのデスクトップOSに遜色ない存在になっていくでしょうね、Linuxは。街でちょっと買ってきたものを気軽に挿すだけ。気に入らなければまた抜いて……」と、うっとりする担当を眺めながら、編者は少しおぞましさを覚えていた。（それじゃ乱交じゃないか。いやいや、私も物書きのはしくれ。いかにモラルに外れることでも、事実とあらば読者に伝える義務があるのだ）

「それから、個人的にありがたいのは【LVM】対応ですかねえ」「えっ？　LVMHですか」「なんだかんだ言っても、まだまだ深刻な不況ですからね。私も金欠ですが、うちに余っているディスクを適当に組み合わせて大きなボリュームを作れるのは福音ですよ。あるいは毎月ちょっとずつお金を出して安いディスクを買い、最終的に巨大なストレージに組み上げるのもいい」「……私はどうかと思いますけどね。LVMH」「えっ？　なんですって？」といぶかしむ担当を前に、編者はすでに氷の溶けきったグラスをつかむと水を一気に飲み干して、意を決したように言った。

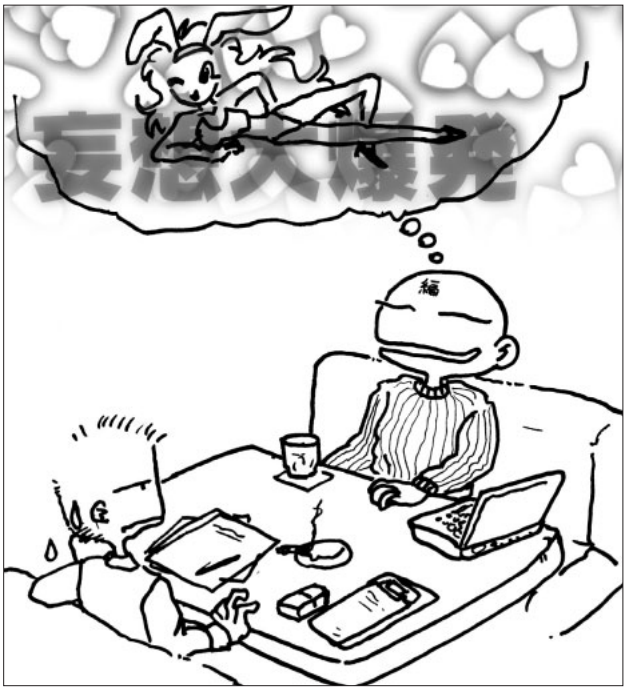
「このあいだも、ウチの愚妻と都心のデパートに出かけたんです。そうしたら、アイツ、つい1カ月前にヴィトンのバッグを買ったばかりだっていうのに、こんどはジバンシィのが欲しいって言うんですよ。なんとかなだめたと思ったら、こんどはケンゾーのジャケットですよ。まあね、月賦払いなら月々安く済むというのはわかります。でも、毎月毎月なけなしの中から血を吐く思いで払った金なんです。おまけに、こういったブランドのほとんどぜんぶが実は同じLVMHグループだと思うと、なにかの陰謀じゃないかと思いたくもなりますよ。私たち、いや女たちはみんな、フランスの巨大企業の手のひらで踊らされてるんですよ」

「あ、あの、なにを言ってるのか……」「私が言いたいのはですね！」と、バンとテーブルを叩く編者。「男はみんな女に甘すぎるってことです！　こういう時代だからこそ、男が質実剛健なところをビシッと見せてやらなくちゃいけない。男のロマンが道を切りひらく。21世紀はそういう時代ですよ！」

「……あなたの言うことはときどきわけがわからなくなりますが」と、担当はつぶやき、静かに編者の手を握りしめながら言った。「女性についてはまったく同意見です」。不遇な男性2人がわかり合えた、とある冬の午後であった。

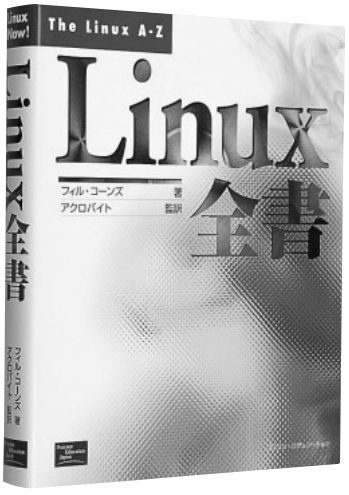
（編集部より）

しのはら氏には、カーネル2.4の主な強化点、および関連する用語の解説をお願いしました。しかしながら、送られてきた原稿が非常に不可解な内容だったため、氏と担当者のあいだでどのような打ち合わせがなされたのかを録音テープを元に再現してお送りいたしました。なお、原稿が到着した日に担当編集者が姿をくらませてしまったため、このレポートの信頼性は検証できておりません。読者のみなさまには、なにとぞご了承いただければと存じます。



都内某所喫茶店にて

Books



Linux全書

フィル・コーンズ 著　アクロバイト 監訳
ピアソン・エデュケーション
B5変形判／544ページ　本体価格 4900円

あるようでなかったLinuxを体系的に学ぶための書籍。本書は前書きにもあるように、大学の講義での使用を主な目的として書かれたテキストだ。「基礎」、「システム管理」、「システムプログラミング」、「デバイスドライバ」、「内部処理」の5部構成で、Linuxを通してオペレーティングシステム内部の構造や処理を理解できるようになっている。ケーススタディや演習問題も用意されているので、独習用の教材としても使うことができる。ただし、どちらかといえばプログラミング寄りの内容のため、基本的なPCに関する知識だけでなく、ある程度のC言語の理解が必要となるだろう。

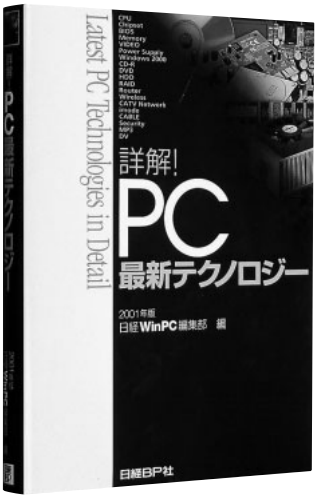
ソースコードが公開されているLinuxは、こうしたオペレーティングシステムそのものを学ぶための素材としては最適だ。「道具」としてのコンピュータに飽き足らず、その内部構造を理解してみたい方や、プログラミングを始めてみようという方にお勧めしたい。

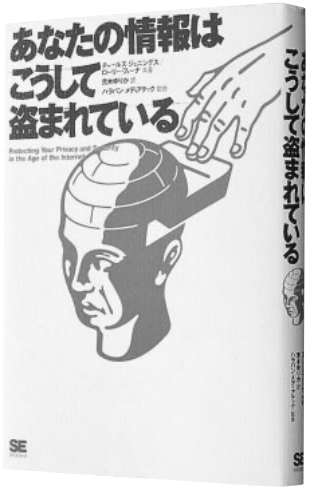
詳解！PC最新テクノロジー 2001年版

日経WinPC編集部 編
日経PB社
A5判／362ページ　本体価格 1800円

12月号の当コーナーで紹介した同じく日経BP社の『最新パソコン技術体系 2001』の姉妹編（？）にあたる一冊だ。こちらのほうが内容はハイブロウで、取り上げる技術も新しいものに絞られている。対象読者は「ユーザー以上マニア未満」といったところだろうか。

CPU、メモリ、チップセットといったPC本体から、周辺の記憶装置／記憶メディア、ネットワーク関連まで、話題は多岐にわたっているが、特にCD-R、DVD、MP3の3つに力が入っている。それぞれの規格についての詳細な解説、関連するソフトウェアとハードウェアについての情報までカバーされていて読み応えがある。で、読んでいてふと気がついたのだが、CD-R、DVD、MP3は取り扱うコンテンツの性格上、いずれも著作権の問題が大きくからんでいる。これらの技術が普及するたびにアーティストや企業、著作権団体などは不正なコピーや配布に悩まされるのだ。JASRACも大変だよなぁと思ったりしたしだいである。

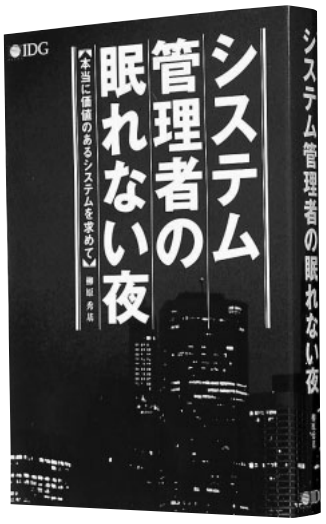




あなたの情報はこうして盗まれている

チャールズ・ジェニングス／ローリー・フィーナ 共著　荒木ゆりか 訳　ハラパン・メディアテック 監修
翔泳社
四六判／416ページ　本体価格 2200円

検索サイトで情報をさがす、同じ趣味を持つ人のホームページを覗いてみる、友達にメールを送る、オンラインショップで買い物をする、ついついエッチな画像を見てしまう、インターネットにアクセスしている間、我々は実にさまざまな行為をしている。特定個人のネット上における「振る舞い」を収集できるとすれば、その人の興味の対象や嗜好、交友関係、さらには性癖さえも明らかになりかねない。少し前の話だが、Windows 98のインストール時に入力されるユーザー情報をMicrosoftがインターネットを通じて収集していたとか、Intelが個々のPCを完全に特定できるID情報をPentiumに埋め込んでいた、といったニュースがあった。米国国家安全保障局が地球的な規模であらゆる通信情報を傍受する「エシュロン」と呼ばれる監視網を張り巡らしているという話も聞く。技術的には不可能ではないのだ。本書を読みながら、日頃のネットでの振る舞いをついつい思い返してしまった。



システム管理者の眠れない夜

柳原秀基 著
IDG Japan
四六判／360ページ **本体価格 1400円**

IDGのネットワーク情報誌『Window NT World』（現在は『Windows 2000 World』）の人気連載の単行本化。著者は現役のシステム管理者で、次々に襲いかかるシステムトラブルに悩まされ、ままならぬユーザーに振り回されるというハードな日常をありのまま伝えてくれている。おそらく、ご同業の方は涙なしには読めないだろう。かく言う私も元同業なのだが、そこは「元」なぶんだけ気楽なもの、関西系のノリにあふれる文章（著者は大阪人）をガハガハ楽しみながら読ませてもらった。Linuxとはまったく関係のない内容ではあるけれど、システム管理者の苦労がしのばれる「トホホ」系のお話（私と同様に気軽な立場の方なら楽しめます）と企業でのシステム管理の在り方についてのヒント（これはご同業の方にも参考になるはずです）が詰まった「1粒で2度おいしい」好著である。ところで、本誌読者にはこの本に登場するような「困ったチャン」ユーザーはいませんよね？

詳説イーサネット

Charles E. Spurgeon 著
桜井 豊 監訳
柏木由美子 訳

オライリー・ジャパン

B5変形判／536ページ

本体価格 4200円

UNIX 2 Linux
ーLinux楽々移植ガイドー

渡辺敏和 著

電気通信協会／オーム社

B5変形判／180ページ／CD-ROM1枚付き

本体価格 2000円

よくわかるネットワークと最新サーバーの基本と仕組み

松下浩明／中村新一郎 共著

秀和システム

A5判／304ページ

本体価格 2000円

Jbuilderではじめる Javaプログラミング入門

掌田 津耶乃 著

秀和システム

B5変形判／368ページ／CD-ROM1枚付き

本体価格 3200円

読者の声

俺にもいわせろ！

今年は全国的に降雪量が多いようです。雪深い地方の方は大変だと思いますが、担当はミニスキーとスキーパンツを新調してワクワクしています。このスキーパンツが優れもので、ウェストのサイズを20cmも調節できるんです。これで体型変化も恐くありません。あとは時間さえあれば完璧！

発売日記載ミスのお詫び

2月号が1月6日出るって1月号に書いてあったのに、9日に出たぞ。毎日、書店に行って余計に出費したじゃないか。遅れるならWebページで言ってくれ。あと、プレゼントコーナーのクイズも立読みされたら意味ないような気がするのですが……。何回送信ボタンを押しても「ページが表示できません」って出るんですど……、これって複数回送信したことになるんですかねぇ？

（國信さん）

1月号の次号予告で2月号の発売日を間違えたのは、編集部のミスです。ご迷惑をお掛けしまして、大変申しわけありませんでした。今後は、このようなことがないよう、一層注意いたします。

プレゼントコーナーのクイズにつきましても、さらに検討を重ねてまいります。また、Webページの件は、連続送信を防止するしくみに引っかかっているのかもしれません。万一、複数回データを送信されてもプレゼント応募が無効になることはありませんのでご安心ください。

新世紀カーネルをお届け

カーネル2.4がリリースされましたので、早く対応のディストリビューションが出てきてほしいということろです（自宅のPCは、マウスとキーボードがUSBなのです）。

（藤原干城さん）

年末の休みを利用して1月号を参考にXFree86を4.0.1にしてみました。なかなか手ごわかったけどなんとか動いてます。 HDbench Cloneのスコアも上がって大満足です。 これからも、いい記事をお願いします。カーネルも2.4の正式版がでたので、それの特集とか……。

（野中厚義さん）

ついに史上最強のLinuxカーネルがリリースされましたね。ディストリビューションの対応を待ちきれない（？）方は、付録CD-ROMに収録したソースを使い、本誌を片手にインストールしてみてください。担当もさっそく新カーネルに入れ換え、サーバとして24時間運用しています。思っていた以上に安定していますよ。

XFree86 4.xのDRI機能に対応するDRMも内包されていますので、対応するグラフィックチップを搭載したカードをお使いでしたら、こちらも試してみてくださいね。

業務システムにLinuxを採用

このたび、会社でLinuxでPostgreSQL＋PHP＋Apacheを用いた業務システムを作成しました。お金と時間がないため、総務課にお願いしてノートパソコンを貸してもらい、5日程度でまとめました。会社で初めてLinuxを導入することと、アプリが自作と聞いて総務からはかなり抵抗を受けましたが、これがないと仕事ができないと強く押し、無事導入にこぎつけました。

（松井敏郎さん）

お仕事で使うシステムへのLinux採用おめでとうございます。Linuxは今後、ますますビジネスの現場で使われるようになるでしょうね。PC UNIXは、部門単位でゲリラ的に導入され、徐々に普及していくパターンが多いと聞きます。会社で正式に導入するとなると、いろいろな人や部門を説得しなければならないというのは、ネットワーク管理者が、「ネットワークのOSI 7階層モデルの上には『政治層』がある」と言うのに似ている気がします。

松井さんのシステムを見て、周りの人もLinuxへの認識を新たにしてくれることでしょう。お仕事頑張ってください。

初級者とはいわないでしょう

いつも、貴誌の記事を興味深く読ませて頂いています。Linux歴約2年の初級者です。導入当初は、マシンのビデ

オカードを設定するために、各種ディストリビューションを試して、いろいろ悩んでいましたが、1年目でどうにかコンパクトなROM化Linuxの設定ができるところまでたどり着けました。最近は本業が忙しく、Linuxを使い倒す暇がなく、悲しい思いをしております。

（shoちゃん）

ROM化Linuxを使いこなすなんて、初級者のレベルはとっくに脱しているじゃないですか。どのようなシステムなのか興味があります。もしよろしければ、編集部に教えてくださいね。『箱の中のペンギンたち』では、組み込み用途でのLinux活用についてを紹介しています。Linuxは広い分野で使われるOSになりました（セガのドリームキャストで動かそうという試みもあるようです）。

Debian検討中

我が家には、今はもうスクラップ寸前になってしまった68k Macが10台以上も元気でいます……（^^; このMac達に今一度立ち上がってもらいたいのです。ぜひ今度Debianを収録してください。やっぱりNetBSDしか選択肢はないのかなぁ……？

（阿部一博さん）

編集部でも、Debian特集と付録への収録は検討しています。しかし、さまざまな理由から実現できていません。Debianを取り上げるからには、フルパッケージでお届けしたい、でもCD-ROMには収まらない……。難しいところです。いましばらくお待ちください（Debianの紹介記事を書いてくださる方も歓迎します）。

初心者を歓迎します

Linuxに興味を持ち始めてはや数カ月。いまだに進歩がありません。ちょっとずつ勉強してます。初心者向けの書籍特集とかもお願いします。自力だけではどうにも……。決して他力本願じゃないです……よ？

（TYPEさん）

私はLinux初心者ですので、貴誌の記事の内容をすべて理解することは到底できません。しかし、それらの理解不能な記事も、今後力をつければ役に立つかもしれず、また、Linuxの奥深さの道しるべともなるので、これからも初心者向けと上級者向けのバランスをとりながら、エキサイティングなLinuxの世界を見せてください。

（小田 純一さん）

朝起きてマシンを起動し、Linuxのインストールを行う毎日を夢見て勉強に励んでいます。でも、朝起きれません。

（別府 航さん）

昔に比べればインストールが易しくなったとはいえ、Linuxは取っつきにくいOSかもしれません。とはいえ、これはUNIX系OSに共通した特徴（？）ですので、使いながら慣れていくしかないのかもしれません。Linuxが逃げてしまうことはありませんので、急ぐことなくマイペースで行きましょう。

漠然と「Linuxを使おう」と思っても、何から手を着ければいいのかわからないという事態に陥ります。Webサーバを構築しよう！　とか、片っ端からゲームを試してみよう！　など、具体的な目標を持つとよいでしょう。

なんて、エラそうなことを書いている担当もまだまだLinux初級者です。ほかの編集部員からいろいろと教えてもらいながらLinuxと格闘する毎日です。

それはそうと、起き抜けいちばんでLinuxをインストールですか。朝起きるのは担当も苦手ですが、起きるなりインストールも……。

3月号の特集へのご意見

3月号の特集「Linuxの小技・裏技・かくし技」は最高でした！　まさに私のほしいテクニックが満載で。

やはりLinux（UNIX）を便利に使うには、CUIの小技を覚えるのがいちばんですね！

（嶋田 以和貴さん）

3月号の特集は小冊子のような形で提供してほしい。

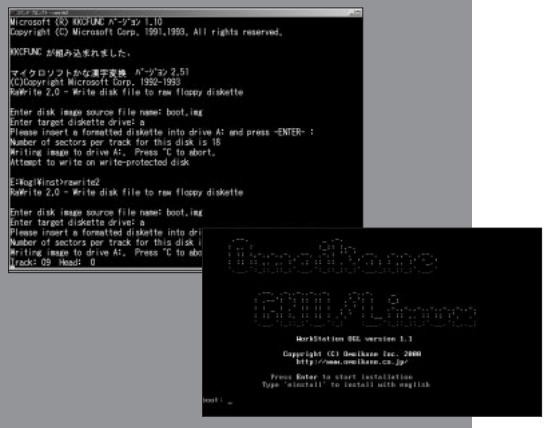
（あんでぃさん）

LinuxやUNIXには、あまり知られていない技がたくさんあります。このような技をいろいろと身につけることで、より便利な環境を作ることができます。また、トラブルから脱出する際にもうまく使えるかもしれません。今回は、小技を中心に（中にはなんだかよくわからないワザもありますが）紹介しました。小冊子にするというのもよいアイディアですね。今後の企画にご期待ください。

今月もたくさんのメッセージをお寄せいただきましてありがとうございます。今年はカーネル2.4、XFree86 4.x、GNOME 1.2、KDE2を標準装備したディストリビューションが次々とリリースされると思われます。x86アーキテクチャのCPUにCrusoeも加わりましたし、gcc 3.0もベータテストが始まったようです。Linuxを取り巻く環境はますます発展するでしょう。

では、また来月お目にかかりましょう。

付録CD-ROMに収録した
Omoikane GNU/Linux 1.1 WorkStation のインストール



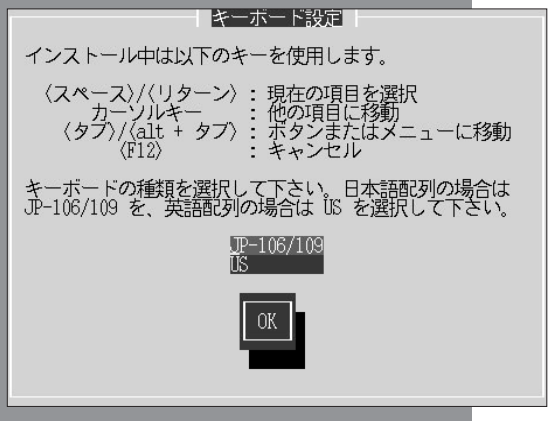
1 ブートディスクの作成とインストーラの起動

CD-ROMブートに対応していないマシンを使う場合は、まず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。手順は以下のとおりです。

（1）Windowsのエクスプローラで、CD-ROMの「ogl」→「inst」とフォルダを開き、その中にある「rawrite2」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、「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。もう一度[Enter]を押すとフロッピーの作成が始まります。

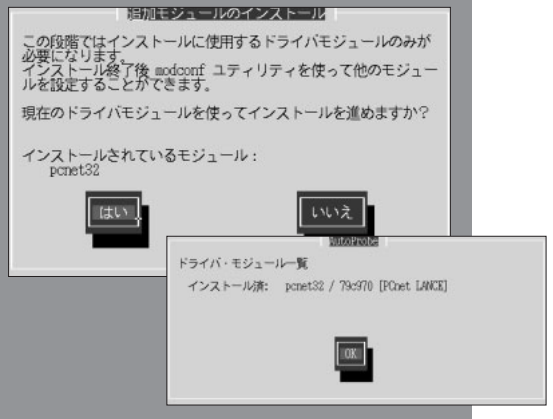
次に、作成したフロッピーかCD-ROMをセットしてマシンを起動します。画面（手前）のように「boot:」プロンプトが表示されたら[Enter]を押します。



2 キーボードの設定

日本語キーボードを使用する場合はデフォルトで選択されている「JP-106/109」を、英語キーボードを使用する場合は「US」を選択します。

Omoikaneのインストーラでは、[Tab]キーや矢印キーを使ってメニューを選択して、[Enter]キーで「OK」ボタンを押します。



3 モジュールの組み込み

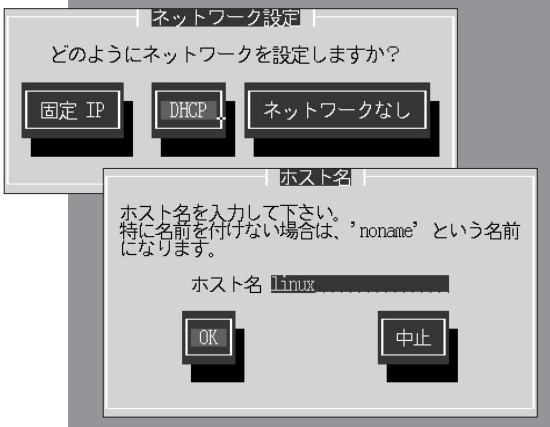
ネットワークカードなどの使用に必要なモジュールを組み込みます（多くのネットワークカードは自動で認識されます）。「追加モジュールのインストール」では「はい」を押して次へ進みます。

4

ネットーワークの設定

DHCPサーバからIPアドレスなどのネットーワーク情報を取得する場合は「DHCP」を、使用するIPアドレスが指定されている場合は「固定IP」を、マシンをネットワークに接続しない場合は「ネットワークなし」を選択します。

「ホスト名」は覚えやすくて短めのものを、あなたの好みで入力してください。

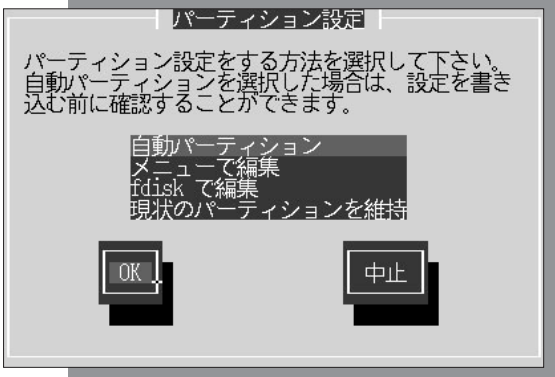


5

パーティションの設定（1）

マシンをLinux専用で使う場合は「自動パーティション」を、すでに設定されているパーティションを使用する場合は「メニューで編集」を、パーティションの作成から始める場合は「fdiskで編集」を選択します。

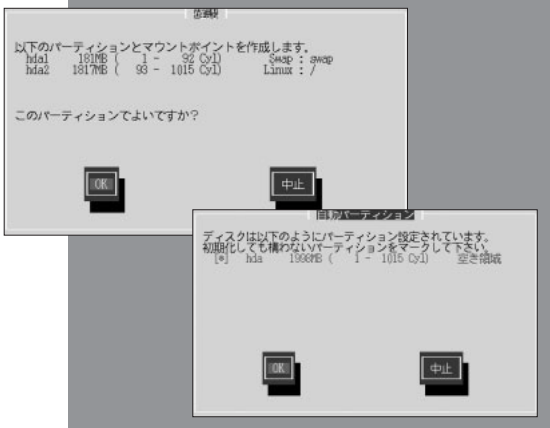
ここではマシンをLinux専用に使うものとして「自動パーティション」を選択します。なお「自動パーティション」を選択すると、現在ハードディスクにある情報はすべて削除されます。



6

パーティションの設定（2）

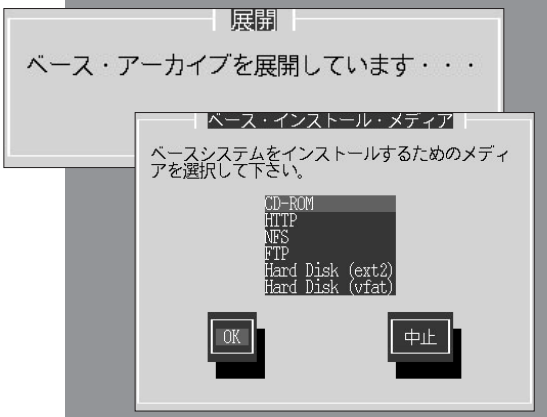
手前の画面で「*」でチェックされているパーティションが「自動パーティション」の対象になり、パーティションが自動で切り直されます。複数台のディスクを接続している場合は、Windowsなどがインストールされているパーティションが「自動パーティション」の対象にならないように、[Space]キーでチェックをはずしてください。



7

インストールメディアの選択

ここでは本誌付録のCD-ROMを使ってインストールするので「CD-ROM」を選択して「OK」を押します。



時刻の設定

「タイムゾーンの設定」はデフォルトで「Japan」が選択されています。このままの状態で「OK」を押して次へ進みます。

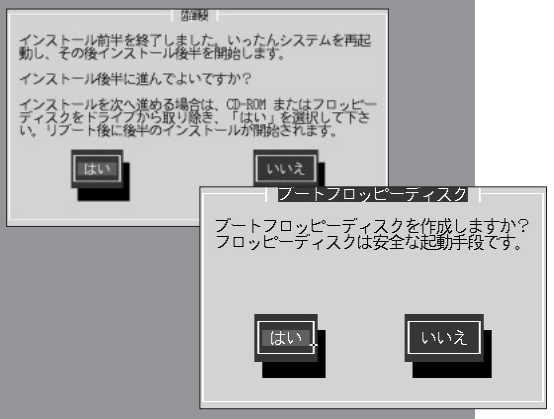
次の「時刻設定」ではデフォルトの「はい」を選択するとよいでしょう。

8

カーネルのインストールとAPMの設定

カーネルのインストールが終わるとAPMの設定です。使用するマシンのBIOSがAPMに対応していてAPMを使用する場合は「はい」を、APMを使用しない場合は「いいえ」を選択します。どちらを選択してよいかわからないユーザーは、デフォルトのままでよいでしょう。

9

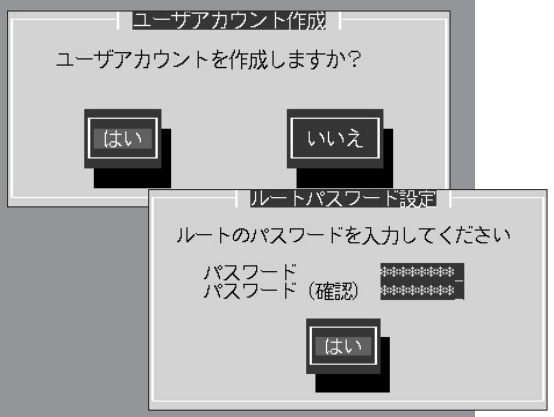


ブートディスクの作成

ハードディスクからLinuxが起動しなくなったときのために、レスキュー用のフロッピーディスクを作成します。空のディスクを1枚ドライブにセットして「はい」を、次の場面でも「はい」を押すとディスクの作成が始まります。

レスキューディスクの作成が終わると、いったんマシンが再起動されます。フロッピーやCD-ROMがセットされていると、インストーラが起動してしまう場合があるので、Linuxが起動するまでCD-ROMなどをドライブから抜いておいてください。

10



ユーザーの設定（1）

Linux管理者用のパスワードを設定します。「パスワード」と「パスワード（確認）」に同じパスワードを入力して「はい」を押します。この管理者のログイン名は「root」です。

次の場面（奥）では「はい」を選択して一般ユーザーの作成に進みます。

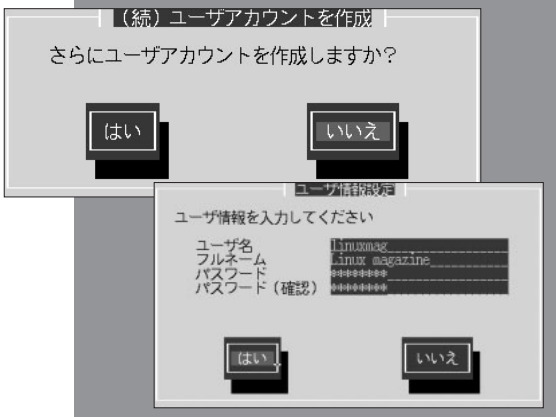
11

ユーザーの設定（2）

一般ユーザーを作成します。「ユーザ名」（ユーザー名にはハイフン（―）を使わないように注意してください）に一般ユーザーのログイン名を、「フルネーム」にユーザーの本名を、「パスワード」と「パスワード（確認）」に同じパスワードを入力します。Linuxなどのマルチユーザー OSでは、一般ユーザーでログインしてメールの読み書きなどを行います。

マシンを1人で使う場合は、「（続）ユーザアカウントの作成」で「いいえ」を選択して次へ進みます。マシンを複数人で共有する場合は、共有する人数分の一般ユーザーを作成します。

一般ユーザーの作成が終わるとパッケージのインストールが始まります。しばらくの間待ちましょう。



マウスと水平同期周波数の設定

マウスが自動認識されると手前の画面が表示されます。「3ボタンエミュレーションを使う」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、「3ボタンエミュレーションを使う」を[Space]キーでチェックして「OK」を押します。

次の水平同期周波数の選択では、使用するモニタのマニュアルを参照しながら適当なものを選択してください。

垂直同期周波数と解像度の設定

垂直同期周波数もモニタのマニュアルを参考にして適当なものを選択します。

次の「解像度の選択」では、まず800×600などの小さい解像度を選択して、Xの表示テストに成功してから、解像度を順に上げていくとよいでしょう。

色数の設定とXの表示テスト

「解像度の設定」と同様に、「色深度の設定」でも小さめの「HighColor」などを選択して、Xの表示に成功してから数値を順に上げていきます。

「OK」を押すとXの表示テストが始まります。表示に成功したら、マウスを使って「終了」を押します。

以上で設定した解像度や色数でXを使用する場合は、「表示テストの結果」で「はい」を選択します。Xの表示に失敗した場合は「いいえ」を、解像度や色数を変更する場合は「再設定」を選択します。

ここで「はい」を選択するとインストール作業は完了です。ログインプロンプトが表示されるので、一般ユーザーでログインして、「startx」とタイプするとXが起動します。

お疲れさまでした。