PentiumIIIデュアルプロセッサ搭載 PLAT'HOME FACTORY 新ラインアップ

ぷらっとホームは、同社のオリジナルプロダクツであるPLAT'HOME FACTORYに、PentiumIII 800MHzデュアルプロセッサ搭載マシンの新製品を発売した。

「Standard System 800DP」は、サーバ用ミドルタワーケースで34万8000円から。「Standard Rack 800DP」は、高さ4Uサイズのラックマウントケースで39万8000円から。「Trus 2200DP」は、高さ2Uサイズのラックマウントケースで47万8000円からとなっている。

全機種とも、PentiumIII800MHzを2基搭載したデュアルプロセッサ構成で、440GXチップセットを採用している。搭載可能なメモリは最大2Gバイトで、オンボードにUltra2 Wide SCSI、GD5480（2MバイトSGRAM）グラフィックス、10/100BASE-Tネットワークコントローラを搭載し、電源には300Wと十分な容量を備えている。そして、標準構成では、128MバイトECCメモリ、18.2Gバイト7200rpmのUltra2 Wide SCSIハードディスクを搭載している。

Standard System 800DPとStandard Rack 800DPのPCIスロットは、66MHzが2スロットと33MHzが4スロットと拡張性の高いデュアルPCIバスとなっていて、高性能なI/Oカードを利用可能だ。Trus 2200DPには、33MHzのPCIスロットが2スロット用意されている。

Trus 2200DPは、前面から保守可能なホットスワップハードディスクベイを4ドライブ分装備していて、RAIDハードディスクに対応可能である。

対応するOSは、Linux（カーネル2.2以降）、FreeBSD（3.x以降）、Solaris 8 for Intel、Windows NT 4.0／2000となっていて、SolarisとWindows 2000以外は注文時にプレインストールを選ぶことができる。

Standard System 800DP
発売日
2000年10月2日
発売
ぷらっとホーム株式会社
TEL
0120-795-123
価格
34万8000円〜
URL
http://www.plathome.co.jp/

Hardware　発売日　2000年10月31日

TurboLinux ServerをバンドルしたPCサーバ apricot Linuxインターネットエントリーサーバセット

URL　http://www.melco.jp/

発売　三菱電機株式会社
TEL　03-3218-9064
価格　28万8000円〜

三菱電機は、apricotスリムタワーにTurboLinux Server日本語版6.1のインストール代行サービスをバンドルした「Linuxインターネットエントリーサーバセット」を10月31日より発売した。

7月に発売した同モデルの性能向上を図ったもので、CPUが変更になったほか、Webブラウザでネットワーク設定などが容易に行えるLinuxサーバ管理ツールに、新バージョンのHDE Linux Controller 2.0 Standard Editionを採用している。

CPUにCeleron 633MHz、メモリ128Mバイト、10Gバイトハードディスク、24倍速CD-ROM、キーボード、マウスなどを標準装備している。ネットワークに10/100BASE-TXを2ポート装備していて、プロキシサーバや簡易ルータとして利用することもできる。

なお、15インチディスプレイと500VA/300Wの無停電電源装置（UPS）「FREQUPS-F」が付属するモデルが用意されている。

Software　発売日　2000年11月10日

Webサイトのアクセスログ解析ソフトウェア サイトトラッカー5.0

URL　http://www.ascii.co.jp/netmedia/

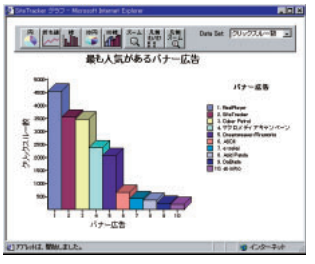
発売　株式会社アスキー
TEL　03-5351-8590
価格　9万8000円〜

アスキーは、ホームページの閲覧状況を解析する「サイトトラッカー5.0」を11月10日に発売する。1つのサイトのアクセスログを解析するプロフェッショナル版、複数のサイトを扱えるエンタープライズ版、大規模サイト用のeビジネス版の3種類のパッケージがある。

サイトトラッカーはサーバマシン上で動作し、操作はWebブラウザで行う。Linux、Windows、Solarisといったほとんどの OSに対応し、Apache、Microsoft IIS、Netscapeなど主要なWebサーバプログラム上で動作可能だ。

またエンタープライズ版とeビジネス版は、Firewall-1や DeleGate、 Squid、 Microsoft Proxy Serverなどのファイアウォール／プロキシ、FTPサーバのアクセスログ解析も行うことができる。

新機能として、訪問者追跡、訪問者プロファイリング、パラメータ解析、トレンドレポート、レポートのメール配信などが追加された。



Software　発売日　2000年11月23日

Windows Me／2000対応パーティショニングツール PartitionMagic 6.0日本語版

URL　http://www.netjapan.co.jp/

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

ネットジャパンは11月23日から、ハードディスク上のデータを保持したまま、パーティション操作が行えるユーティリティ「PartitionMagic 6.0日本語版」を1万5800円で発売する。また、ネットワーク経由でリモートにパーティション操作ができる「PartitionMagic Pro 6.0日本語版」も9万円で同時に発売する。

FAT、NTFS、HPFS、Linux ext2といった各種のファイルシステムに対応しており、パーティションの作成、移動、コピー、削除、結合、サイズの変更、異なるファイルシステム間の変換などが可能だ。ユーザーインターフェイスが変更され、1画面に複数のディスクマップを表示でき、ドラッグ＆ドロップでパーティションのコピーと移動が可能になった。

また、マルチブートマネージャBootMagicと、ドライブレターを変更可能なDriveMapperユーティリティがバンドルされている。

Software　発売日　2000年9月1日

Webポータルやミニライブ機能搭載グループウェア impression office

URL　http://www.asi.co.jp/

発売　アドバンスドソリューションズ株式会社
TEL　03-5543-6331
価格　7万8000円〜

アドバンスドソリューションズは、Webグループウェア「impression office」を9月1日から発売した。動作OSは、Linux、FreeBSD、Windows NT／2000、UNIXで、クライアントは、Webブラウザからと Windows専用プログラムからの両方からアクセスでき、ほぼ同じ操作性を実現している。

利用中のグループウェアの機能が一括して参照できるポータル画面から操作が行え、電子メール、掲示板、フォーラム、共有アドレス帳、スケジューラ、施設予約、文書管理、ToDoリスト、伝言板、ミニライブ、インターネットニュースリーダの機能がある。

ミニライブでは、テレビ電話、内線電話、ボイスメール、ビデオメール、チャットとホワイトボードを利用することができる。

Software　　発売日　2000年11月20日

マルチプラットフォーム対応Java開発ツール Borland JBuilder 4日本語版

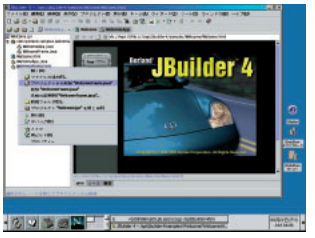
URL　http://www.inprise.co.jp/jbuilder/

発売　インプライズ株式会社
TEL　03-5350-9380
価格　6万8000円〜

インプライズは、Java開発ツールの「Borland JBuilder 4」日本語版を11月20日から発売する。3つの製品があり、Javaの学習に最適なビジュアル開発環境の無償版「JBuilder 4 Foundation」、インターネットを利用したWebアプリケーション開発に対応した「JBuilder 4 Professional」が6万8000円、大規模・分散アプリケーション向けの開発環境「JBuilder 4 Enterprise」が36万円。

Borland JBuilder 4は、Javaサーブレット、JSPによるWebアプリケーションや、EJBをはじめとするアプリケーションサーバの開発機能を持つJava開発ツール。Windows、Linux、Solarisの各プラットフォームに対応する。

今回のバージョンアップでは、アプリケーションサーバ開発機能を大幅に改善したほか、Inprise Application Server 4.1開発キットの搭載と統合機能の提供、BEA WebLogic製品ファミリーのサポート、チーム開発機能の搭載など、大規模・分散システム構築における開発生産性を高める機能が提供される。



Software　　発売日　2000年11月1日

メール配信マーケティングシステム HDE Customers Care 1.2

URL　http://www.hde.co.jp/cc/

発売　株式会社ホライズン・デジタル・エンタープライズ
TEL　03-5456-3260
価格　65万円〜

ホライズン・デジタル・エンタープライズ（HDE）は、LinuxおよびSolaris上で稼動するメール配信マーケティングシステム「HDE Customers Care 1.2」を11月1日に発売した。

HDE Customers Careは、顧客の属性とメールアドレスを統合管理できるマーケティングツールで、Webブラウザから顧客データベースを活用し、条件を指定した配信ターゲットの絞込みや配信日時予約によって最適なメール配信を行うことができる。

顧客の趣味や嗜好をあらかじめ登録しておき、それに基づいて顧客が希望する情報が配信されるオプトインメール配信が行えるようになった。

そのほか、iモード対応、クリックカウント、フォローアップメール配信、記念日配信、リマインドメール配信、データベース最適化、データベースバックアップなどの機能が追加された。

Software　　発売日　2000年10月1日

MS OfficeやPDFファイルからテキストを抽出する DocCat V2.0、DocCat PDFオプション

URL　http://www.dehenken.co.jp/

発売　株式会社データ変換研究所
TEL　075-254-8780
価格　各4600円（パーソナル版）

データ変換研究所は、MS Wordなどの文書ファイルからテキストデータを抽出するプログラム「DocCat V2.0」を10月1日から発売した。

パーソナル版はLinux、FreeBSDに対応し、価格はいずれも4600円。またサーバ版は、Solarisにも対応し16万8000円。

新バージョンでは、MS Office 97／2000ファイルのプロパティ情報の表示機能やExcel 97／2000のファイル情報をCVS形式で表示が可能になった。

また、「DocCat PDFオプション」を付け加えることで、アドビAcrobat ReaderのPDFファイルに含まれている文書の中から、テキスト部分のみを抽出し、EUC、SJIS、JIS、UTF-8、UCS-2で出力することができる。DocCat PDFオプションは、同社のホームページから評価版がダウンロード可能で、製品は2001年1月1日より発売される。

Software　　発売日　2000年10月下旬

Webオンラインショップ構築ソフトウェア Personal WebShop for Java Model X

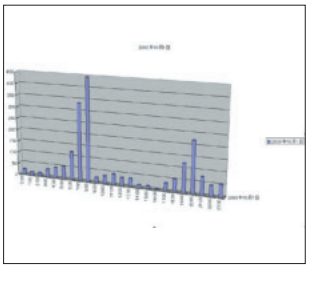
URL　http://www.azi.co.jp/

発売　クラスキャット ネットワークス株式会社
TEL
価格　オープンプライス

クラスキャット ネットワークスは、Webオンラインショップ構築ソフトウェアWebShopシリーズの新モデル「Personal WebShop for Java Model X」を発売した。Linux対応版はPostgreSQL 6.5を、Windows NT対応版はMicrosoft Jetのデータベースを利用する。

すべての操作をWebブラウザから行うことができ、商品を販売する機能のほかに、販売業務に必要な受注管理、顧客管理、会員管理などの機能を備えていて、導入後すぐにオンラインショップを構築することができる。

追加された機能として、各ページへのアクセスログ解析機能とグラフ表示、顧客に対する注文確認メールの自動発信機能、商品一覧ページからショッピングカートに入れて購入する方式が利用可能となっている。



News Headline from 日刊アスキーLinux

www.Linux24.com

オブジェクト指向スクリプト言語 Python 2.0リリース

2000年10月17日

フリーなスクリプト言語「Python 2.0」が10月16日、リリースされた。このバージョンでは新たに、“＋＝”や“＊＝”のような代入演算子を含む新しい文法を導入、Unicode文字列や（リファレンスカウントではない）真のガーベージコレクタなどをサポートした。Python 2.0は、PythonLabsからダウンロードすることができる。

Python 2.0のダウンロード
(http://www.pythonlabs.com/products/python2.0/)

Sun Microsystems StarOfficeのソースコードを公開

2000年10月15日

Sun Microsystemsは10月13日、オフィススイート「StarOffice」のソースコードや関連ドキュメントなどを公開した。CVSサーバはすでに動作しており、Webからスナップショットのソースコードやバイナリをダウンロードすることができる。StarOfficeのオープンソース化は7月に約束されていた。

StarOfficeを最初に開発したのはドイツのStarDivision。1999年にSunは同社を買収、StarOffice 5.2を無償で公開した。その後、Sunはオフィススイートをオープンソース戦略に転換することを決定して、今回、ソースコードをGPL／LGPLとSISSL（Sun Industry Standards Source License）のデュアルライセンスでリリースした。次期バージョンとなるStarOffice 6.0は、OpenOffice.orgの開発成果を活用して作成するという。

OpenOffice.org
(http://www.openoffice.org/)

Compaq、Itanium搭載サーバに無料でログインできるサービスを提供

2000年10月12日

Compaqは10月9日、Itanium搭載サーバへのアクセス権を提供するサービスを開始した。このサービスは、開発者向けにさまざまなOS／ハードウェア環境を無料提供する「Test Drive Program」の一部として実施される。これにより開発者は、IA-64 Linuxが稼動するItanium搭載サーバに実際にログインして、プログラムを実行することができる。

提供されるマシンの名前は“Blazer”。CPUはItanium 666MHz×4、メインメモリは2Gバイトというパワフルなサーバだ。

Itaniumは、Intelがx86アーキテクチャの後継として開発を進めているIA-64アーキテクチャのプロセッサ。IA-64は64ビットアーキテクチャで、命令セットもまったく新しいものになっているため、x86対応の既存のプログラムがうまく動作しない可能性がある。Test Drive Programを利用すれば、開発者はItaniumの出荷前に問題を把握・修正できることになる。

開発者に対してIA-64搭載マシンを提供するサービスは、Compaqのほかに、VA Linux Systemsも実施している。同社が運営するオープンソース開発のサポートサイト“SourceForge”では、さまざまなアーキテクチャでコンパイル環境を提供するサービスを行っており、そこでIA-64サーバを利用することが可能になっている。

Test Drive Program
(http://www.testdrive.compaq.com/)
SourceForge（http://sourceforge.net/）

Linux標準化団体のFree Standards Group、開発プラットフォームの仕様を発表

2000年10月12日

Linux標準化団体の「Free Standards Group」は10月11日、Linuxにおける開発環境の仕様となる「Linux Development Platform Specification（LDPS）1.0」を発表した。一般に入手可能なディストリビューションにおいて、アプリケーションの移植性を確保するのが目的だ。

LDPS 1.0では、異なるディストリビューションでプログラムを問題なく動作させるために、開発ツールやライブラリのバージョンを規定している。ここでは最新バージョンではなく現時点で一般的に使われているバージョンが選択されているが、その理由は、最新版向けに開発すると古いシステムでは動作しないことがあるからだ。結果として、多くのディストリビューションの最大公約数をとった仕様になっている。

Free Standards Groupは、バイナリ形式でソフトウェアを配布する開発者に対して、LDPS 1.0に準拠することを勧めている。

LDPS 1.0で必須のパッケージは次のとおり。

Linuxカーネル 2.2.［14～］
glibc 2.1.［2～3］
XFree86 3.3.［6～］
ncurses 4.2/5
gcc version egcs-2.91またはgcc 2.95.［2～］
binutils 2.9.［1～］.x

このほかに移植性を確保するためのガイドラインとして、C++におけるライブラリの取り扱いや、シェルスクリプトをテストすべきbashのバージョンなど10項目が規定されている。

LDPS 1.0に適合するディストリビューションは次のとおり。

Caldera OpenLinux 2.4
Conectiva Linux 5.1
Linux-Mandrake 7.0
Red Hat Linux 6.2
SuSE Linux 6.4
TurboLinux 6.0
Debian GNU/Linux 2.2
Corel Linux OS Second Edition

LDPSは、移植性を高めるための開発

プラットフォームを規定しただけで、パッケージの互換性のようなディストリビューションの標準化を目的とする文書ではない。そちらの文書については現在Linux Standard Base（LSB）が活動しており、LDPSに取って代わる仕様を完成させる予定だ。

Linux Development Platform Specification
（http://www.freestandards.org/ldps/）
Linux Standard Base
（http://www.linuxbase.org/）

特許フリーな暗号化アルゴリズムの実装を進めるGnuPG

2000年10月10日

GNU版のPGPの実装「GnuPG（GnuPrivacy Guard）」が、新たに2つのアルゴリズムをサポートする。まずサポートされたのは、おそらくもっとも有名な暗号化アルゴリズム「RSA」。RSAはオリジナルのPGPでは初めから採用されていたアルゴリズムだったが、GnuPGはこれまで非標準のモジュールを通してしかサポートしてこなかった。これは、ソフトウェアをフリーに保つために、特許で保護されたアルゴリズムを使用しないという方針があったからだ。

RSAの特許の有効期限が2000年9月26日に切れたのを期に、GnuPGはバージョン1.0.3でRSAをサポートした。

もう1つサポートされたアルゴリズムは、10月2日に次世代標準暗号（AES; Advanced Encryption Standard）としてNIST（米国商務省標準技術局）が発表した「Rijndael」。

AESは、いままで標準暗号として使用されてきたDES（Data Encryption Standard）が計算能力と暗号解読理論の進歩によって脆弱になってきたため、NISTがアルゴリズムを公募して決定したものだ。GnuPGでRijndaelを取り扱うためには、いまのところ実験的なモジュールを動的ロードしなくてはならないが、AESは特許で保護されていないためすぐにGnuPG本体に取り込まれるものとみられる。GnuPGはRFC 2440で定義された「OpenPGP」標準に従っており、OpenPGPではすでにAESの取り扱い方法が定義されているため、これからAESの実装を行うほかのPGPとの互換性も問題ないだろう。

GnuPGは、ファイルやメッセージを暗号化・署名するためのフリーソフトウェア。メールの暗号化や署名などに主に使われている。

GnuPGホームページ
（http://www.gnupg.org/）

AMD、次世代アーキテクチャ「x86-64」シミュレータをリリース

2000年10月7日

米AMDは10月6日、次世代プロセッサアーキテクチャ「x86-64」のシミュレータ「AMD SimNow！」をリリースした。BIOSベンダーやツール開発者、OS開発者などに対するx86-64対応ソフトウェアの開発サポートを目的としている。

x86-64はx86を64ビットに拡張したアーキテクチャで、大規模データベースなどに必要な4Gバイト以上のメモリをサポートする。第1世代のプロセッサとしてコードネーム“Hammer”のリリースが2001年末に予定されている。

SimNow！の対応プラットフォームは、現在のところLinux版のみ。www.x86-64.orgでRed Hat LinuxとSuSE LinuxのRPMパッケージが公開されている。インストールには、Athlon 700MHz以上のCPU（推奨1GHz）や384Mバイト以上のメインメモリ（推奨512Mバイト）、ハードディスクの空き4Gバイト以上などが必要だ。

SimNow！は、CPUやメモリ、ノース／サウスブリッジ、ディスプレイ、ハードディスクやフロッピーなど、PCの主要要素をすべてシミュレートする。そのほか、ログをとるためのデバイスや、レジスタなどの状態をリアルタイムに監視できるデバッガが付属するという。

すでにx86-64コードを出力するgccやbinutilsがリリースされているため、今回のSimNow！のリリースで、とりあえずx86-64のコードをコンパイル／実行できるようになった。IntelのIA-64 Linuxと比較すれば開発環境の充実度はまだまだだが、AMDも着実に進歩しているといえる。

x86-64 Linux（http://www.x86-64.org/）

SGIのファイルシステム「XFS」のベータ版リリース

2000年9月28日

SGIのファイルシステム「XFS」Linux版のベータがリリースされた。

XFSは、IRIX（SGIのUNIX）のファイルシステム「EFS」に性能の限界を感じたSGIが、より高い性能とスケーラビリティを求めた結果開発されたファイルシステムで、メタデータをB＋ツリーで効率的に管理する。

ディスクアクセスの並列化とシーク時間の短縮を目的にディスクを0.5Gバイト～4Gバイトの単位で分割処理する「アクセスグループ」の導入や、4Gバイトを超える巨大ファイルのサポート、数千エントリ以上の巨大ディレクトリの効率的なサポートなど、多くの技術が活用されている。IRIXには1994年から採用されている。

Linuxの一般的にユーザーにとって、XFSの一番の利点は、ジャーナリングのサポートだろう。ジャーナリングは、ファイルシステムのメタデータの更新をログに記録しておく機能で、突然のクラッシュでも、ログを「リプレイ」することによりファイルシステムの一貫性を保つことが可能になる。

現在Linuxで主流のExt2ファイルシステムはジャーナリングをサポートしておらず、これがクラッシュすると、キャッシュにしか保存されていなかった重要な情報が欠けてしまい、修復プログラム「fsck」を実行して正常な状態に直さなければならない。この処理は、大規模なファイルシステムでは長時間を要するため、クラッシュ後のダウンタイムが長くなる大きな要因だった。

XFSのソースコードはSGIのWebサイトからダウロードできる。ただしベータ版のコードは、ファイルシステムの破壊という致命的な結果をもたらす可能性があるため、試用する場合はバックアップを万全にするべきだ。

SGI（http://www.sgi.com/）
XFSベータ版のダウンロード
（http://oss.sgi.com/projects/xfs/beta.html）

Linux West 2000

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

10月5日・6日の両日、大阪城公園に隣接した大阪ビジネスパークTWIN21ギャラリーにおいて「Linux West 2000」が開催された。昨年は、関西圏最大のIT関連イベント「COMMUNET'99」の特別企画として位置づけられたLinux Westだが、今回は単独での開催となった。オープンスペースに設けられた会場の各所では、熱心にブースを見つめるビジネスマンの姿が目立った。

併催されたセミナーも好評。クロージングセッションでは、アクアリウムコンピュータの宮原 徹氏が壇上に立った。

昨年のCOMMUNET'99特別企画から数えて2回目となる今回のLinux West。昨年の南港見本市会場から大阪市中央区の大阪ビジネスパークへと会場を移し、サブテーマに掲げられた「Linuxビジネスの今後を実証する」ための、文字通りビジネス色の強いイベントとして開催された。

昨年度は、日本Linux協会や大学関係など、ユーザーの視点から見たユニークな出展が目立ったが、今回はビジネスユースを視野においた企業からの出展でブースは占められていた。

出展企業は21社。うち7社がカタログ展示のみの出展ということで、会場は比較的こぢんまりとした印象ではあったが、昼休み前後の時間帯を中心に、スーツに身を固めたビジネスマンで会場は賑わっていた。会場となった大阪ビジネスパークは、松下電器の主要部門をはじめ、NEC関西支社やKDD大阪ビルなどIT色の強い大手企業がひしめくオフィス街。2日間の参考入場者数は6214人と発表されたが、ビジネスマン風の来場者の占める割合が非常に高かったのが印象的であった。

ビジネスユースへの提案

会場内でまず目に付いたのが、あちこちのブースにどっかと腰を据えたラックマウントシステム。日本IBMやCompaqなどの大手コンピュータメーカーをはじめ、ノーザンライツコンピュータやフォース、地元大阪のイージーネットなどがそれぞれに特徴あるラックマウントシステムを展示していた。前回のイベントではあまり見かけなかったこうした光景からも、Linuxのビジネス分野進出が本格的になってきたことを再認識させられてしまう。

Compaqはこのほかにも、標準ブラウザでリモートサーバ制御を可能とする「Lights-Out Edition」システムを出展。外部ブラウザからのアクセスで、サーバマシンの電源オンオフまでもが制御できてしまう遠隔操作システムであり、来場者の熱い視線が集まっていた。

ビジネスユース向けシステムといえばWebグループウェアの名前を挙げることができるが、こちらの分野ではネオジャパンが「iOffice2000」を出展。この製品は、通常のインターネット・イントラネットのほか、iモードや各種PDAにも対応したグループウェア。携帯端末に対応したこの仕様からも昨今の時代の潮流が感じられた。

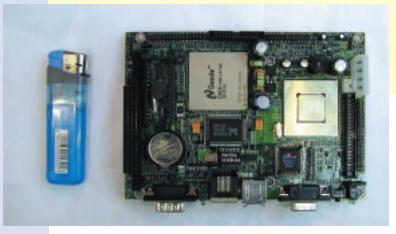
IBMブースの中央に腰を据えたラックマウントサーバ。今回の出展では、特にこうしたシステムが目立った。

Compaq出展の「Lights-Out Edition」システム。標準ブラウザでリモートサーバの電源制御までが可能だという。

Postgresかと思いきや、これはグッディ出展の「Usogres」と命名されたシステム。いかにも大阪らしい名前だ。

この手のイベントではお馴染みとなったペンギンちゃん。でも、よく見ると何と値札がついている。こんなところも大阪らしい。
果たして買い手はついたのだろうか？

アクアリウムコンピュータのマイクロサーバ「silver neon」。前面のLEDがチャームポイントだ。



ケーメックスが出展した超小型マザーボード。このサイズで十分にサーバとして機能する。

こちらもイベント会場ではお馴染みのキャンギャルちゃん。ビジネスマンが多かった会場のオアシスだった。

地元企業からもユニークな出展

大阪を拠点とするITベンチャーの健闘が目立ったのも今回のイベントの特徴。アクアリウムコンピュータは、昨年のLinux Westがそのデビュー期と重なり会場の注目を集めた「Blue Grass」の後継機種や、サーバ監視LEDを搭載した新製品「silver neon」などのSOHO向けマイクロサーバを出展。

Linuxプラットフォームでの各種システム開発を得意分野とするグッディからは、PostgreSQLのさらなる堅牢化を計る二重化システム「Usogres」などが展示された。その名前の由来は、1つに見せかけて実は2つのPostgreSQLが稼働しているから。ウソのPostgresで「Usogres」というわけ。いかにも大阪らしいベタなネーミングであるが、妙に耳に残るところはさすがだ。

いずれも独自発想によるユニークな製品であり、新しい商売に敏感な大阪から、オープンソースのメリットを活かした新たなLinuxビジネスが生まれることに期待が高まるところだ。

ソリューションセミナーも好評

期間中、ツイン21の20階では同イベントの無料セミナーが併催された。SRAの石井達夫氏、グッディの前田青也氏による特別講演に始まり、わずか2日間に計20テーマものLinux関連ソリューションセミナーが開講。各セミナーとも、それこそ1時間刻みの過密スケジュールであったが、平日開催にもかかわらず事前予約だけで各セミナーはほぼ満席という好評ぶりだったという。

クロージングセッションでは、アクアリウムコンピュータの宮原 徹氏が壇上に立ち「Linuxビジネスの明日を考える」をテーマに講演したが、ここでも多くの来場者たちが熱心に話に耳を傾けていた。

●

昨年は、どちらかといえばLinuxに関心が薄い関西の市場に「Linuxとは何ぞや」を問うためのイベントというイメージが強かったこのLinux West。しかし、この1年の間にLinuxはビジネスシーンで至極当たり前の言葉に変わってしまった。

世紀を超える次の1年。このイベントがどのように姿を変え、私たちの前にお目見えするのか、今から楽しみでしかたがない。

Distribution

新着ディストリビューション

HOLON Linux 2.0

どこまでもユーザーフレンドリーに。斬新なデザインで注目を集めたLinux2000Gが、名前を新たにHOLON Linux 2.0として登場した。ハードディスクがすべてWindows領域であるマシンにも簡単にマルチブート環境を構築してくれるディストリビューションだ。随所に見られるそのユニークさを覗いてみよう。

Official Red Hat Linux 7J

まさに先駆け。高度で新しい機能の実装は、いつもRed Hat Linuxから始まる。Linux界をリードし続けるこのディストリビューションは、Cライブラリ、コンパイラ、Xなど、Linuxシステムの根幹部分を一新してメジャーバージョンアップを果たした。FHSやOpenLDAPなど次世代の標準となる技術を採用し、21世紀に向けて抜かりはない。

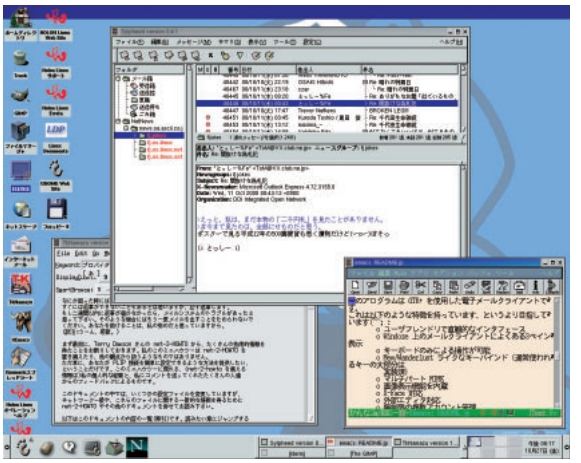
HOLON Linux 2.0

11月2日、真にユーザーフレンドリーな環境を目指して開発されたLinux2000Gの最新バージョンが、名前をHOLON Linux 2.0（以下、HOLON Linux）と改めリリースされた。

HOLON Linuxは、IntelとPowerPC（PPC）の2つのアーキテクチャに対応し、それぞれ「豪華」と「並」の、合計4種類のパッケージが存在する。

「豪華」と「並」は、付属する商用アプリケーションと、提供されるサポートが主な違いで、「豪華」で提供される90日間、件数無制限の電話サポートは、通話料が無料だ。

基本システムには、カーネルに2.2.17が、Cライブラリに2.1.3が、XにXFree86 3.3.6が採用されている。



画面1　ユーザーフレンドリーなデスクトップ
デスクトップに配置されるアイコンから、使用頻度の高いアプリケーションを起動する。

レールに乗って簡単マルチブート

お店でパソコンを買ってくると、たいていのマシンにはWindows 98がインストールされている。付属の大容量ディスクは、通常すべてWindows用の領域となっているので、初心者がここにLinuxをインストールするのは至難のワザである。

HOLON LinuxのCD-ROMをWindowsマシンにセットすると、おもむろにPartitionMagic PrepToolのインストーラが起動する。このツールはパーティション領域の編集ツールとして名高いPartitionMagicの簡易版だ。

このツールのインストールを終えてマシンを再起動すると、Partition Magic PrepToolが立ち上がる。ここでLinux用に使う領域を作成すれば、難しいパーティション作成ツールを使わないLinuxのインストールが可能である。

ユニークな視点

HOLON Linuxに収録される商用ソフトは**表1**のとおりで、日本語入力プログラムといった定番ソフトのほかに、15書体（「並」は5書体）が付属するDyna Fontと、商用のサウンドドライバOSSが特に目をひく。

さらに、デスクトップ（**画面1**）の「インターネットメール」アイコンをダブルクリックすると、Windows環境でおなじみのルック＆フィールを持つメーラSylpheedが起動し、スプレッドシートやLinux関連ドキュメントの検索エンジンもあわせて、ほかのディストリビューションにあまり見られない、よく考えられたデザインとなっている。

「細かいことで悩まない」というのは、初心者にとって最重要命題なのである。

アプリケーション名	概要	Intel版	PPC版
Wnn6 Ver.3	UNIX環境で定番の日本語入力プログラム	共通	共通
DynaFont 5書体	美しい日本語表示のための商用フォント	共通	共通
翻訳魂	使いやすい和訳、英訳ソフト	共通	なし
PartitionMagic PrepTool	パーティション領域編集ツールの簡易版	豪華	なし
BootMagic	PartitionMagic PrepTool付属の簡易ブートセレクタ	豪華	なし
ATOK12SE R.2	WindowsやMac OSで定評のある日本語入力プログラム	豪華	なし
OSS	商用のサウンドドライバ	豪華	なし

表1　HOLON Linux 2.0に付属する商用ソフト
収録されるDynaFontは「豪華」が15書体、「並」が5書体である。

製品名　HOLON Linux 2.0
価格　1万1800円（Intel「豪華」）
3980円（Intel「並」）
9800円（PPC「豪華」）
4800円（PPC「並」）
問い合わせ先　株式会社ホロン
03-5282-5101
http://www.linux2000g.ne.jp/

Official Red Hat Linux 7J

Linux界をリードするディストリビューション、Red Hat Linuxの最新版Red Hat Linux 7J（以下、Red Hat 7J）のリリースである。英語版は9月末にリリースされており、今回紹介する日本語版も10月6日にDeluxe、10月13日にProfessionalと、ほぼタイムラグなしでリリースされた。

注目の基本システムには、カーネルにバージョン2.2.16、XにXFree86 4.0.1、Cライブラリにglibc2.1.93、Cコンパイラにgcc 2.96が採用されている（**表1**）。

カーネルは現行の安定版である2.2系だが、カーネル2.4からのバックポートにより、USB機器へも対応している。USB接続のキーボードとマウスはインストーラ起動時から使用可能だ。

製品ラインナップは2種類

日本語版の製品版ラインナップは、デスクトップ用のDeluxeと、サーバ用のProfessionalの2種類で、6.2Jにあった商用ソフトを含まないスタンダード（廉価版）はなくなっている。

これら製品版は、デスクトップ用の商用アプリケーション（**表2**）、翻訳された活用ガイドなどを含むドキュメントCD、Professionalに限りバンドルされるサーバ用の商用アプリケーションなどで構成される（**表3**）。

また、ユーザー登録後にインストールに関する30日間の電話サポートと、90日間の電子メール、FAXによるサポートが受けられる。さらに、Professionalユーザーに限り、30日間Webサーバ Apacheの設定サポートが提供される。

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

本誌付録CD-ROM収録のRed Hat Linux 7JはFTP版です。非商用ソフトだけが含まれ、製品版を販売しているレッドハット株式会社からのサポートを受けることはできません。

基本システムは総入替え

Red Hat Linuxといえば、最新プログラムへのいち早い対応が特徴である。今回のバージョンアップでは、カーネルこそ2.2系であるが、Cライブラリを始めLnuxの根幹部分が一新された（**表1**）。

■gcc

Linuxシステムは、カーネルを始めとする多くのソフトウェアがC言語で書かれている。そのため、標準で採用されるCコンパイラのgcc（GNU C Compiler）は、システムの中核をなすツールと言える。

さて、Red Hat 7Jで採用されたgcc 2.96は、次期バージョン3.0の開発版スナップショットにあたるものだが、このバージョンについてgccのメンテナンスチームは「we do not recommend using them for production purposes.（実用目的のプログラムのコンパイルに使うことは勧めない）」と注意を促している。

なお、カーネルコンパイルの際、このgccは使用されず、egcs 1.1.2をベースにしたkgccが起動される。

■RPM

一般にいう「Red Hat系」のディストリビューションは、パッケージ管理にRPM（Red Hat Package Manager）という方式を採用している。

現在あるRed Hat系ディストリビューションでは、バージョン3系統のRPMが採用されているが、Red Hat 7Jでは最新バージョンの4.0が採用されている。現時点でリリース版にRPM 4.0を採用するディストリビューションは、このRed Hat 7Jだけである。

RPM 4.0は、パッケージ情報をBerkeley DBというデータベース形式を使って管理する。RPM 4.0で使われるデータベース形式は、これまでの

名称	バージョン	概要	URL
gcc	2.96	GNUがメンテナンスするC言語のコンパイラ	http://gcc.gnu.org/
RPM	4.0	Red Hat社が考案したパッケージ管理方式	http://www.rpm.org/
FHS	2.1	UNIX系OSのディレクトリ構成の指針を目指す規格	http://www.pathname.com/fhs/
glibc	2.1.93	システムの基本構成を決めるライブラリ	http://www.gnu.org/software/libc/
XFree86	4.0.1	多くのディストリビューションで採用されるX Window System	http://www.xfree86.org/

表1　大きく変更された基本システム
Red Hat 7Jではシステムの根幹部分が変更された。glibcやgccは次期バージョンの開発版である。

アプリケーション名	概要
Wnn6 for Linux/BSD Version 3.02	UNIX環境で定番の日本語入力プログラム
ATOK12 SE for Linux R.2	PC環境でユーザーの多い日本語入力プログラム
dp/NOTE for Linux/BSD Version 2.02	UNIX系OSで動作する日本語ワープロ
DynaFont 5書体	美しい日本語表示のための商用フォント
HancomWord	Microsoft Word互換を目指すオフィスアプリケーション
SystemCommanderLite	マルチブート環境構築の定番ソフト

表2　バンドルされる商用ソフト
上はデスクトップ用の商用アプリケーションで、Deluxe、Professional両方に収録される。Professionalにはサーバ用の商用アプリケーションも収録される。

DB1からDB3.1に変更されており、Red Hat 7J以外のディストリビューションでは、Red Hat 7J用に作成されたRPMパッケージが利用できなくなっている。

■glibc

Linuxシステムの基本的なリソースとなるのが、このGNU C Libraryである。Red Hat 7Jで採用されたglibc 2.1.93は、glibcの次期バージョン2.2のテスト版（glibc 2.1.91以降がテスト版に該当）にあたる。

glibc 2.2（その開発版にあたるglibc 2.1.93も）には、日本語ロケールデータが実装され、より完全にマルチバイト対応するのが特徴だ。

しかし、glibc 2.1と2.2の互換性の問題で、Red Hat 7J用に作成されたアプリケーションは、安定版のglibc 2.1系を採用するディストリビューションで動作しない可能性がある。

現在多くの商用ソフトがRed Hat Linuxに対応しているため、ほかのディストリビューションユーザーに大きな影響を及ぼすかもしれない。

■FHS

FHS（Filesystem Hierarchy Standard）は、UNIX系OSで共通のディレクトリ構成とディレクトリ名の使用を目指す一種の規格だ。

よくファイルが格納される場所について、「～Linuxではこの場所」などと、ディストリビューション間のディレクトリ構成の違いに由来する発言を耳にするが、FHSは、これらについてLinuxディストリビューション間の違い、ひいては系統の違うUNIX系OS間の差異をなくそうとしている。

現在多くのLinuxディストリビューションは、FHSの前身にあたるFSSTND（Linux Filesystem Structure）を採用している。このためFHSを採用するディストリビューション（Red Hat 7JやDebian 2.2など）と、そうでないディストリビューションではディレクトリ構成が異なる。

マニュアルや各アプリケーションのドキュメントを保存する場所、デーモンの起動スクリプトの場所などが目立った変更箇所で、ディレクトリ構成の違うディストリビューション間でのパッケージの使いまわしには注意が必要である。

■XFree86

Red Hat 7JはX Window Systemに、XFree86の最新バージョンの4.0.1を採用している。XFree86 4.0では、Mesa 3Dライブラリを利用可能にするGLXや、グラフィックカードの3Dアクセラレーション機能を有効にするDRIがサポートされている。これによりLinuxでも高速な3D表示の利用が期待できる。

ネットワーク上での情報管理

Red Hat Linuxの常に新しい機能を実装するという特徴は、9月にライセンスが開放されたRSAという暗号化アルゴリズムを採用するなど、セキュリティ分野にも大きく表れている。

■Kerberos

Red Hat 7Jのインストーラ画面になにやら見慣れないものがある（**画面1**）。

この中のKerberosは、暗号化と復号化をひとつのキーでまかなう共通鍵方式を採用した暗号技術である。

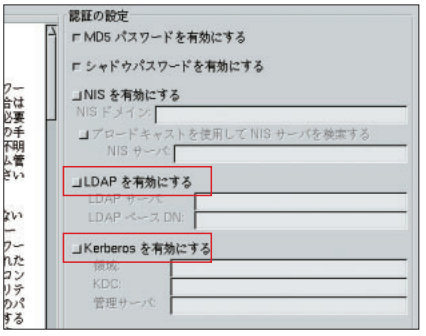
共通鍵方式の暗号技術を利用する際、ユーザーは通信する相手ユーザー数だけ鍵の保管が必要になる。

しかし、鍵の数とともに管理の手間は増えるし、多くのユーザーが複数の場所でPCを使う現状を考えると、鍵の保管について共通鍵方式の利便性の悪さは嬉しくない。

そこで、Kerberosを使えば共通鍵を

収録物	概要
インストールCD	インストール用のCD-ROM（2枚）
ソース（SRPM）CD	ソースコードのパッケージ集
ドキュメンテーションCD	HTML、PDF形式のマニュアルを収録
商用アプリケーションCD（ワークステーション用）	ATOK、HancomWordなどの商用ソフトを収録
PowerTools CD	機能拡張のための追加パッケージ集
IBM SoftWare for Linux2000 CD（評価版）	Lotus DOMINOなど、IBMが開発するLinux用ソフトウェア集
インストールガイド	約100ページのインストールガイド
インストーラ起動用FD	インストーラ起動用のフロッピーディスク
商用アプリケーションCD（サーバ用）	Sybase Databaseなどのサーバ向け商用ソフト集
Developer Module Archive CD	Perlスクリプト集CPANなどを習得

表3　製品版の収録物
Red Hat 7Jにはインストール用のCD-ROMのほかに、活用ガイドなどが収録される。



画面1　強化された認証技術
インストール画面には、今まで見られなかった認証項目が追加された。

ユーザーごとに保管せず、あらかじめ用意しておいた専用のサーバで管理することで、利便性を改善できるというわけだ。

■Open LDAP

新しい項目（**画面1**）のもうひとつはOpen LDAP（LDAP：Lightweight Directory Access Protocol）というユーザー管理システムである。

サイトが大きくなるにつれて手間が増えるのがユーザー管理である。多数のユーザーを一括管理するものとしては、Windowsのみの環境ではWindows NT/2000サーバのドメインコントローラを、UNIX系OSのみの環境ではNISを使うのが一般的な手法である。

しかし、Windows、Linux、Mac OSなど複数のOSが混在する環境が増えている現在、ユーザー情報をまとめて管理する方法がないのが実情だ。

LDAPは、この現状を改善するものとして期待されるユーザー管理技術である。氏名、パスワード、メールアドレス、所属などのユーザー情報をツリー形式（DNSもこの形式と言える）で管理し、LDAPに対応するアプリケーションから、これらのユーザー情報が利用できる。

また、複数のLDAPサーバと連携しての容易なシステム拡張が可能で、前述のKerberosやOpenSSLと連携させれば、安全なユーザー情報管理システムを構築できるというわけだ。

このほかにも、電子メールの暗号化に便利なGnuPG、RSAアルゴリズムを利用したOpenSSLモジュール、安全にリモートマシンと通信できるOpenSSHなど、Red Hat 7Jにはセキュリティ関連のツールが満載だ。

GUI設定ツール

基本システムのほかにも、ユーザーフレンドリーなGUI設定ツールがいくつか追加されている。

■Configure Firewall

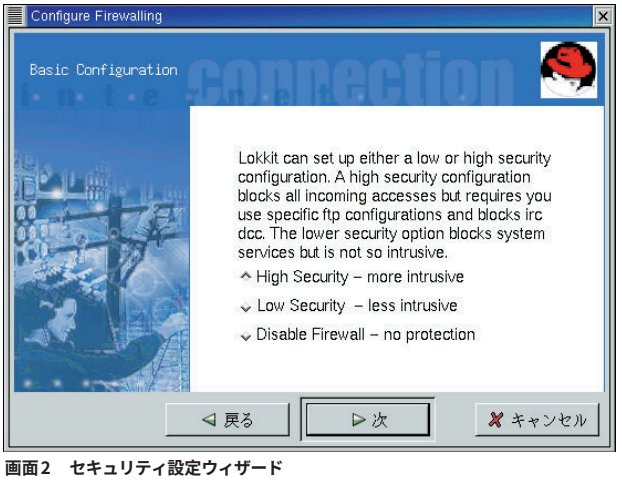
Configure Firewall（**画面2**）は、マシンの大ざっぱなセキュリティレベルや、Webサーバへのアクセス許可などを設定するツールだ。このツールはウィザード形式になっており、テキスト形式の設定ファイルに不慣れなユーザーには嬉しい。

■Firewall Configuration

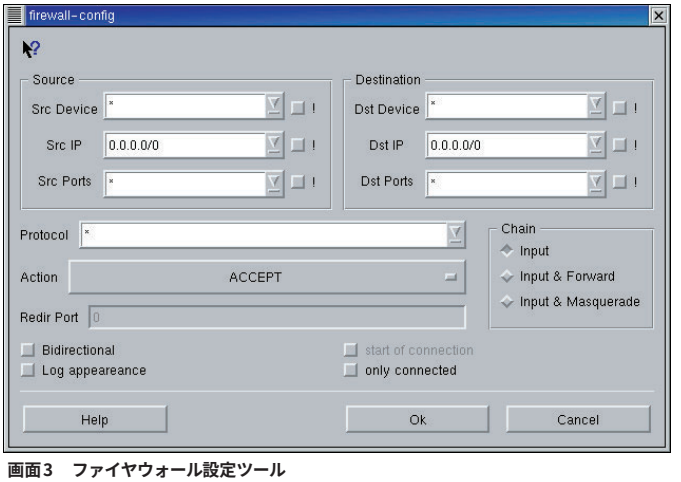
名前は似ているが、Firewall Configuration（**画面3**）はパケットフィルタリングを設定するためのGUIツールだ。デーモンの起動スクリプトファイルにipchainsコマンドを列記する方法が難しいと感じるユーザーには嬉しいツールである。

■Kernel Configurator

Kernel Configurator（**画面4**）は、直接modprobeコマンドを使わずに、GUI操作でモジュールを組み込んで、Linuxに新しいハードウェアを追加するツールである。Windowsでいうデバイスマネージャに似た役割と言ってよいだろう。



画面2　セキュリティ設定ウィザード
マシンのセキュリティレベルなどをウィザード形式で設定する。



画面3　ファイヤウォール設定ツール
ipchainsでの設定をGUIで行う。

■Update Agent

Update AgentはRed Hat Networkというオンラインサービスを利用して、更新されたパッケージのインストールなどを行うツールだ。

しかし、このUpdate Agentで使用するrhnsdというデーモンプログラムにはバグが存在し、Red Hat 7Jをインストールしたままの状態だと、約3週間後にシステムが停止すると報告されている（http://www.redhat.com/support/errata/RHBA-2000-081-05.html）。

Red Hat 7Jを本格的に使おうと考えるユーザーは、この不具合への対処が必要だ。

不具合を修正するためのツールに重大なバグが潜んでいるとは、なんとも皮肉な話である。

さらなる改良

大幅に変更された基本システムや、新しく追加されたGUI設定ツールのほかにも日本語環境などが細かく変更された。

■デスクトップ環境

Red Hat 6.2Jにあった、EmacsからCannaなどの変換サーバを利用できないという不具合がRed Hat 7Jで修正された。Red Hat Linuxは日本語環境の整備にあたり、Project Vineとの協力体制を敷いており、整備されつつあるオフィス用の商用ソフトとあわせて、今後も日本語環境の強化が期待できるだろう。

なおGNOMEとあわせて使われていたウィンドウマネージャのEnlightenmentは、今回からSawfishに変更されている。

■LILO

今回収録されたLILOは（サポート外だが）8Gバイトを超える領域からのOS起動もサポートし、昨今のディスク大容量化に対応した。この機能を使うにはlilo.confにLBA32と記述する。

また、これまで[Tab]キーで表示していたOSメニューがグラフィカルな画面に変更されている（**画面5**）。このあたりは、LILOに不慣れなユーザーを考慮した細やかな改良として評価できる。

■xinetd

これまでSamba、Telnetといったインターネットサービスは、inetdというプロセス名のインターネットスーパーサーバで一括管理されていたが、Red Hat 7Jからは、このサーバがxinetdというプログラムに変更されている。

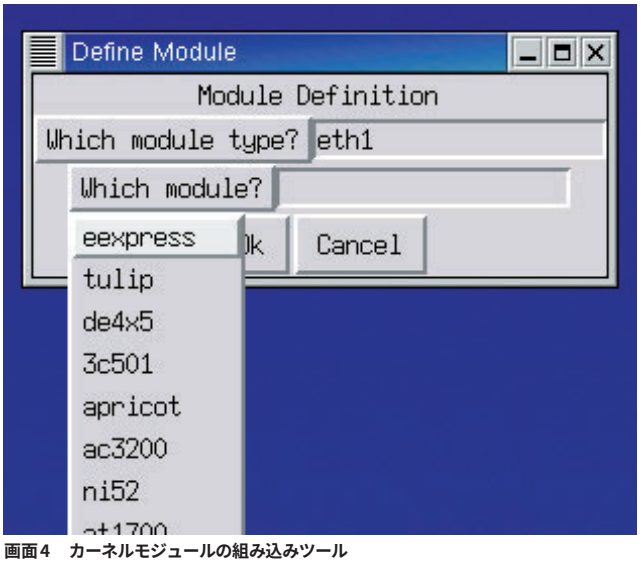
設定ファイルも/etc/inetd.confではなくサービスごとに/etc/xinetd.d以下に配置されているので注意が必要だ。

起点となるディストリビューション

今回のバージョンアップでは、予想以上に大幅な仕様の変更がなされた。新機能やディレクトリ構成の変更などは、Red Hatのようなリーディングディストリビューターが採用してこそ普及するものだから、それらを推進するという点で評価できる。

しかし、テストバージョンが採用されたライブラリやコンパイラは、Linuxシステムの根幹となる部分だけに、その採用については評価が分かれるところだろう。

他のディストリビューションとの互換性については、早くほかが追いついてほしいと願うべきかもしれない。

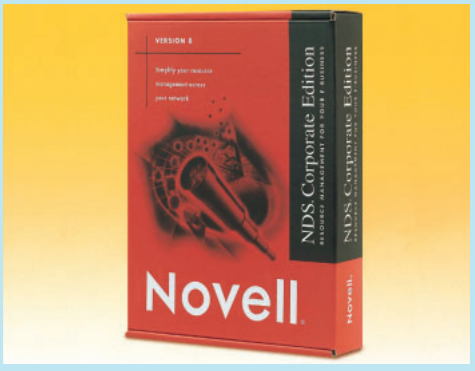


画面4　カーネルモジュールの組み込みツール
ネットワークのモジュールなどを読み込む。

画面5　グラフィカルなLILO画面
これまでのLILOプロンプトからグラフィカルなOS選択画面に変更された。このメニューに各種OSが追加されていく。

Products

Linuxに対応したNDSディレクトリサービス



NDS Corporate Edition

接続されているマシンが増えてくると管理が難しくなる。ディレクトリサービスは、部署名、個人名、マシン名といった階層型に分けて表現することができ、そのシステムが持っているリソースを一元管理することができるユーザー認証データベースシステムである。

製品名	NDS Corporate Edition
価格	オープンプライス
問い合わせ先	ノベル株式会社 TEL 03-5481-1294 http://www.novell.co.jp/

ノベルから、ディレクトリサービスNDS eDirectoryのLinux対応版が発売された。「NDS eDirectory for Linux」と「NDS Corporate Edition for Linux」の2種類が用意されている。

NDS eDirectoryは、DOSやWindows用のネットワークOSとして知られているNetWareを発展させたもので、標準となっている各種のネットワークプロトコル、LDAP（Lightweight Directory Access Protocol）v.3ディレクトリサービスにも対応している。

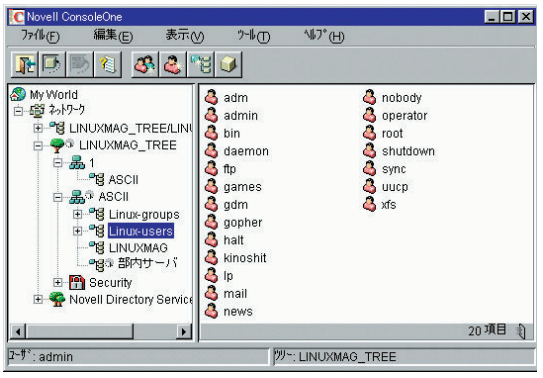
対応するOSは、NetWare、Windows 95／98／NT、Solaris、Linuxで、動作するLinuxディストリビューションは**表1**の通り。製品はパッケージとユーザー数に応じたライセンス販売があり、パッケージに含まれるサーバ用とクライアント用のCD-ROMには、それぞれのOS用のプログラムが収録されている。

NDS Corporate Edition（以下NDS CE）は、NDS eDirectoryの機能に加えてユーザーやグループの管理機能を備えていて、名前の通り企業内のアカウントの統合一括管理を行うことができる。

ひとつだけ覚えればいいパスワード

今回は、NDS CEを利用して、複数のLinuxマシンを共通のアカウントで使えるように設定した。これだけでは、NIS（Network Information Service）を利用した/etc/passwdと/etc/groupファイルの情報共有と変わらないが、NDS CEではLDAPやWindowsマシンも含めて一元管理できるというメリットがある。

まず、Red Hat Linux 6.2Jをインストールしたサーバマシンに、NDS CE



画面1　ConsoleOne（Windows用）
ユーザー、グループ、LDAP、証明書などの管理を行うユーティリティ。

画面2　WindowsのExplorerでの表示
ネットワークコンピュータで、NDS eDirectoryで登録されたツリーにアクセスできる。

をインストールする。テキストベースのインストーラnds-installを実行すると、NDS ServerとNDS User Account Management（UAM）のインストールが行える。前者がディレクトリ機能、後者がユーザー管理機能である。

インストール中に、環境設定ファイルをviで修正する手順があるので、ディレクトリツリーの［Root］管理者名、ツリー名、サーバ名（コンテキスト）などを書き直して、新規にツリーを作成する。LDAPの定義も、必要であればここで行うことができる。管理者のパスワードを入力すれば、インストールは終了である。

次に、NDS管理ユーティリティのインストールだが、現在はWindows用だけが用意されており、Linux（Solaris）用は次バージョンで搭載される予定だ。そのためWindows 98マシンに、ConsoleOneユーティリティとNDS管理ユーティリティをインストールした。

さて、初期状態では、管理者以外のユーザーやグループは設定されていない。ConsoleOneを利用してアカウントを設定することも可能だが、今回はLinuxのユーザーアカウントをNDSのアカウントに移行してみた。

NDS CEをインストールしたLinuxマシンで、migrate2ndsプログラムを実行すると、そのマシン上にある/etc/passwdと/etc/groupファイルに登録されているアカウントを、NDSに簡単に移行できる。その登録した内容を、ConsoleOneで表示したのが**画面1**である。

そして、Linux上での認証をNDSで行うための設定を行う。現在のLinuxの認証は、PAM（Pluggable Authentication Module）と呼ばれる方式で認証モジュールを入れ換えられるようになっている。また、Name Service Switch（NSS）によって、ファイル、NIS、NIS＋といった認証の順番を設定している。

NDS CEをインストールすると、NDS用の認証モジュール(login.nds.sso、rlogin.nds.ssoなど)が/etc/pam.d.nds/に、NDS用のNSS設定ファイルが/etc/nsswitch.conf.ndsにコピーされる。モジュールを/etc/pam.dディレクトリ以下にあるファイルに置き換え、NSS設定ファイルを/etc/nsswitch.confファイルにコピーすれば、NDSよる認証が行われるようになる。

もう1台のLinuxマシンでは、NDS UAMをインストールし、さきほど登録したNDSツリーに参加する形で設定を行い、同様にNDSの認証に変更する。

ConsoleOneを使用してNDSに登録されているユーザーを、そのLinuxマシンのメンバーに登録することで、一度ログイン（認証）すれば、他のマシンにアクセスするときにも新たにユーザー名とパスワードを入力する必要がないというシングルサインオンが実現する。

NTドメインからディレクトリサービスへ

Windowsマシンだけならば、Windows NTサーバを立ててNTドメインでユーザー管理するのがふつうだ。しかし、部署ごとにNTサーバがあってドメインが複数ある場合などで、違う部署のマシン（ファイルサーバやプリンタ）へアクセスするには、ドメインの信頼関係を結んで解決するといった設定が必要だ。

NDS CEを導入することで、WindowsやUNIX、Linuxサーバのアカウント管理の一元化と、アクセス可能な資源の制限といったセキュリティ設定がきめ細かく設定できる。複数のグループで連携して作業を行うような用途では、ディレクトリサービスのありがたみがわかるだろう。

Red Hat Linux 6.2J	カーネル2.2.16-3、glibc:2.1.3-15
TurboLinux Workstation日本語版6.0	カーネル2.2.16-2、glibc 2.1.3-8(UAMのみ対応)
TurboLinux Server日本語版6.1	カーネル2.2.15-8、glibc 2.1.3-8
LASER5 Linux 6.2	カーネル2.2.14-12LL1、glibc 2.1.3-15LL1

表1　対応するLinuxディストリビューション

VAIO PCG-C1VJ/BP

Crusoeを搭載したノートPCは富士通、NECなど各社から発売される予定だが、まず先陣を切って発売したのがソニーである。ノートPCから、携帯端末、Mobile Linuxへと期待のCrusoeだが、本製品でLinuxが利用できるのだろうか興味のあるところだ。

製品名	VAIO PCG-C1VJ/BP
価格	オープンプライス
問い合わせ先	ソニー株式会社 TEL 03-5454-0700 http://vaio.sony.co.jp/

10月7日にソニーよりWindows MeをバンドルしたVAIO PCG-C1VJ（以下C1VJ）が発売された。ラインナップはC1VJとC1VJ/BPの2種類あり、後者はCD-ROMドライブとMicrosoft Office 2000 Personalが付属する。なお、両機種ともフロッピーディスクドライブは別売となっている。

このC1VJは話題のCPU、TransmetaのCrusoe TM5600を採用した世界最初のノートPCだ。

CrusoeというCPU

Crusoeといえば、Linuxerにはお馴染みだろう。Linuxの生みの親である、Linus Torvalds氏が勤めるTransmetaがノート向けとして発表し、世界中に衝撃を与えたCPUだ。

Crusoeの詳しいテクノロジーについては他誌にゆずるとするが、少しだけ触れておく。

CrusoeはIntelやその互換CPUのようなx86アーキテクチャを持つCPUではない。Crusoeでは、コードモーフィングソフトウェア（CMS）により、x86命令をネイティブコードに変換することで、x86用のソフトウェアを動かしている。簡単にいうと、x86エミュレーションだ。

しかし、考えてみるとすでにIntelやAMDのCPUもRISCアーキテクチャを採用したCPUであり、x86コードを変換して実行しているのだ。唯一の違いはCMSというソフトウェアによる変換かハードウェアによる変換かの違いだ。

Crusoe自体は単純なCPUのため、非常に高速に動作する、また、CMSを高速に動作させるためにメインメモリ上に16Mバイトのキャッシュを持つ。これらによって、ソフトウェアエミュレーションでありながら、さほど速度を犠牲にしなくてすむようになっている。

CMSによるメリットは大きい。単純なCPUであるため消費電力が少ない、発熱量が少ないというメリットが生まれるのだ。さらに、LongRunというテクノロジーを採用し、CPUの負荷に合わせて、クロックと電圧を細かく変更している。これによって、驚くほど消費電力を抑え、バッテリによる長時間運用が可能となった。

CPU	Transmeta Crusoe TM5600 600MHz
メモリ	128Mバイト（最大192Mバイト）
システムクロック	100MHz
ビデオチップ	ATI RAGE Mobility M1（8Mバイト）
LCD	1024×480ドットTFTカラー液晶
ハードディスク	約12Gバイト（約8Gバイト/約4Gバイト）
その他の装備	USBポート×1、6インチ35万画素CCD、TypeII（CardBus対応）PCカードスロット×1、メモリスティックスロット（マジックゲート対応）×1、内蔵モデム（V.90/K56Flex自動対応ソフトウェアモデム）など

表1　おもな仕様

新装備

前のモデルからの変更点は、前述したCPUのほか、グラフィックチップが、3Dアクセラレーション機能を持った、ATI RAGE Mobility M1（ビデオメモリ8Mバイト内蔵）に変更されている。また、赤外線ポートがなくなり、AV出力端子が装備されている。

そのほか、マジックゲート対応のメモリスティックスロットを搭載した。

Linuxは動くのか？

まず、インストールに関してだが、CD-ROMドライブが付属しているC1VJ/BPはいいとして、C1VJでは、Windows Me上で、ネットワーク経由でCD-ROMイメージをハードディスクにコピーしてインストールを行うしかない。しかし、ネットワークカードのドライバをインストールするのも一苦労だ。モデム経由でダウンロードするしかない。

Linuxを使うのであれば、C1VJ/BPを選択するか、別売のCD-ROMドラ

イブを購入したほうがいいだろう。

さて、CD-ROMドライブからのインストールができるのであれば、あとはX Window Systemということになる。C1VJに採用されているATI RAGE Mobility M1はXFree 4.0.1でも、3.3.6でも対応済みだ。といいたいところだが、確かにインストーラはグラフィックチップを自動認識するのだが、Xが起動すると、画面が真っ白になってしまう。ただし、カーネル自体はハングアップしていないようだ。

ドライバ自体や、設定などを色々いじってみたが残念ながらVGA以外では表示できなかった。

■それでもX Window System

Xを動かすことを諦めたころ、『Linux怒濤のQ&A』の著者であるぱぱんだ氏のホームページ（http://www.tk.airnet.ne.jp/~papanda/）で、Xが動いたという報告を見つけた。その方法は、フレームバッファを使用し、SVGAドライバを使用するというものだ。早速試したところ次のような問題があるものの起動することができた。

- **仮想コンソール画面が表示できない**
- **Xが終了するとコンソール画面が表示できなくなる**
- **画面下に8ラインほどの白いラインが表示される**

■インストール方法

インストール方法を簡単に説明する。

①CD-ROMからインストールする場合、IDEのオプションを指定し、テキストモードを選択する。インストーラのBootプロンプトで、次のように入力すればいいだろう。

```
boot: text ide2=0x180
```

②インストーラのXに関する設定はスキップする。

③インストールが完了したら、以下のような設定でカーネルを再構築する。

```
CONFIG_FB=y
CONFIG_FB_VGA16=y
CONFIG_FB_ATY=y
```

④XF86configをリスト1のように変更する。

手順は以上だ。ただし、上記の例はXFree 3.6.6での設定であり、4.0.1では動かない。

写真1　非常にコンパクトなC1VJ
本誌の上に配置してみた。持ち歩くならコレぐらいのサイズがいい。

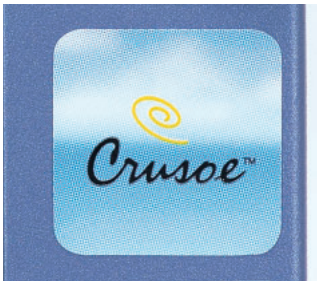


写真3　Crousoeのロゴシール
黄色い渦巻きがちょっと見えづらい。これとは別に黒いロゴも存在するようだ。使い分けは不明。

Linuxを使ってみて

注目の省電力に関しては、標準バッテリでカーネルのコンパイルを繰り返し行ったところ、約1時間15分ほど稼働した。LinuxにはまだLongRunをコントロールするユーティリティなどがないので、CPUはフル稼働していると思われ、さらにハードディスクを使い続けてのこの時間というのは、かなりの好成績だろう。通常の使用であればもっと長い時間使用できるはずだ。ただし、バッテリ残量がなくなると、いきなり電源が落ちてしまうので、バッテリ残には常に気を付けなければならないだろう。

メモリスティックや、CCDカメラなども利用できたという報告もちらほら入ってきている。これら設定などはまた別の機会に紹介したいと思う。

なお、C1VJでLinuxを動かすことはメーカーの保証外である。Linux使う際にはその点に留意してほしい。

▼リスト1　XF86Config（抜粋）

```
Section "Monitor"
    Identifier  "LCD Panel 1024x768"
    HorizSync   31.5-48.5
    VertRefresh 50-100
    Modeline "1024x480" 65.00 1024
    1032 1176 1344 480 491 493 525
    - hsync -vsync
EndSection

Section "Device"
    Identifier  "My Video Card"
    VideoRam    8192
EndSection

Section "Screen"
    Driver      "svga"
    Device      "My Video Card"
    Monitor     "LCD Panel 1024x768"
    Subsection "Display"
        Depth       16
        Modes       "1024x480"
        ViewPort    0 0
    EndSubsection
EndSection
```

GNOME 1.2&KDE 2.0を探る!

最新デスクトップ大

研究

Linuxを圧倒的に使いやすくしたのが総合デスクトップ環境のGNOMEとKDEだ。GNOMEを標準に採用するディストリビューションが多いが、これら2つの実力は甲乙つけがたい。GNOMEを使うか、KDEを使うか……それが問題だ（編集部）。

GNOMEとKDE

文：竹内充彦
Text: Michihiko Takeuchi

近年、Linuxの利用者が急激に伸びた理由はいくつかある。インターネットとの親和性や、サーバとしての信頼性、開発ツールやアプリケーションの充実など、性能面での需要は疑う余地がない。ただし、これらの需要を強力に後押ししているのは、ユーザーにとっての使いやすさ、プログラマーにとっての開発しやすさが、ともに格段に進化したことである。その立役者とも言えるのが、本特集のテーマであるGNOMEとKDEである。

最近のディストリビューションでは、サーバ用ディストリビューションを除けば、ほとんどの場合、標準でどちらかの環境がインストールされるため、読者の多くもおそらくGNOMEかKDEのどちらか、あるいは両方を利用しているに違いない。

よくある質問に、「GNOMEとKDEのどちらがいいのか」というものがある。どうしてもライバルとして比較されやすい両者であるが、これは一概にどちらと言える問題ではない。それぞれが目指す方向性の違いもあれば、ユーザーの好みや、ユーザーの利用環境の違いもあるからだ。ただ言えるのは、あらゆるOSにおいて、どちらも現在最強のデスクトップ環境の域にまで達しているということだ。

本特集では、まずGNOMEとKDEの位置付けを理解し、その最新版を探り、現バージョンでのユーザー向けにちょっとした活用法を紹介する。GNOMEかKDEかと迷うこと自体、Windowsユーザーにはできない贅沢な悩みであるが、本特集がそんな贅沢な悩みを解決する一助となれば幸いである。

ところで、文中「GNOMEとKDE」というようにKDEよりもGNOMEを先に持ってくることについては他意はない。純粋にアルファベット順と思っていただきたい。

GNOMEやKDEっていったい何だ？

まずはちょっとおさらいをしておこう。そもそもGNOMEやKDEとはいったい何だろう？

たいていの場合、「デスクトップ環境である」という答えが返ってくるはずだ。では、デスクトップ環境っていったい何だ？　と、食い下がってみよう。

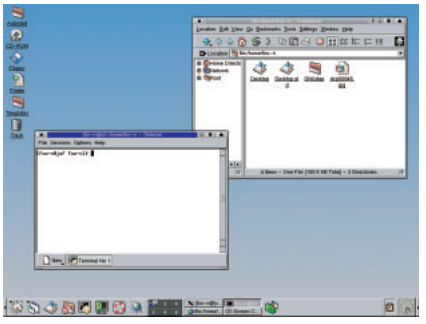
デスクトップ環境とは、見栄えのよいデスクトップ、統一されたルック&フィール、アプリケーションに共通する操作性などを提供する環境のことだ。たしかに、GNOMEもKDEも、デスクトップに自由にアイコンを配置できたり、アプリケーション間でデータアイコンをドラッグ&ドロップできたりと、その操作性は格段に進化している。こうした操作環境を実現するシステムソフトウェアとアプリケーションソフトウェアの集合なのだと考えればいい。

GNOMEもKDEも、目指しているインターフェイスについて共通する部分が多い。相方ともおおむね以下のようなものを提供している。

- **統一的なデザインを持つアプリケーション**
- **アプリケーションやデータを象徴するアイコン**
- **階層構造を持つメニュー**
- **自由度の高いデスクトップ**
- **自由度の高いパネル**
- **ドラッグ&ドロップによる操作性**
- **ファイルの種別に応じたアプリケーションの起動**
- **ネットワークに対する透過性**

デザインの統一は、それぞれの環境内であれば、新しいアプリケーションに直面しても、直感的に操作法が把握できるように配慮されている。たとえば、どんなアプリケーションでも必ず左端に［ファイル］メニューがあり、その中に終了コマンドがあるという統一項目は重要である。また、すべてのユーザーの好みを満たすことは難しいが、デスクトップやパネルについては、カスタマイズ可能とすることで、より

画面1　GNOME 1.2



画面2　KDE 2.0（開発中の2.0RC2）

多くの要求に応えようとしている。GNOMEとKDEのカスタマイズについては、72ページからの「GNOMEのカスタマイズ」、88ページからの「KDEのカスタマイズ」を参照されたい。

■フレームワークとしての側面

このように、ユーザーには操作環境としての側面を持つGNOMEとKDEだが、プログラマーに対しては、もう1つの側面がある。それはGNOMEもKDEも「アプリケーションフレームワーク」であるということだ。

フレームワークとは、クラスやオブジェクトを組み合わせた、ある一定の枠組みを持ったアプリケーションの雛形や、再利用可能なプログラム（ライブラリ）のことだ。クラスライブラリと概念的には近いが、ライブラリを集めただけではなく、オブジェクトをどのように組み合わせればいいかといった、オブジェクト間の関係についての情報も含んでいる。

そもそも、GNOMEはGTK＋というウィジェットを使って構築されている。同様に、KDEはQtというウィジェットを使って構築されている。ウィジェットとは、X Window System上でウィンドウアプリケーションを構築する際に利用可能なオブジェクトであり、ボタンやウインドウの見た目を左右するものである。

こうしたオブジェクトのライブラリを利用すれば、X Window System上で稼動するアプリケーションを構築することは可能だが、それよりも、GNOMEやKDEの環境で提供されるライブラリを利用してフレームワークにそうように構築したほうが、それぞれの環境で提供される高度な機能（たとえばドラッグ＆ドロップなど）が容易に実装できるというわけだ。

ウィンドウマネージャとは違うの？

「あれ？　ウィンドウマネージャのAfterstepにもアイコンドックはあるし、アプリケーションの見た目は似てるような気もするけど、これとは違うのかな？」そう、これまたよくある質問が、「GNOMEやKDEってウィンドウマネージャとは違うの？」というやつだ。

ウィンドウマネージャとは、ユーザーの操作により、ウィンドウの移動、サイズ変更、アイコン化、アクティブウィンドウの切り替え、ウィンドウプロセスの制御などを行うXのクライアントプログラムのことだ。ウィンドウマネージャには、古くはtwmに始まり、fvwm2、純和製のqvwm、NeXT StepライクなAfterstep、美麗なEnlightenment、最近はやりのsawfishなど、フリーのものですぐに見つかるだけでも軽く50種類以上はある。これらのウィンドウマネージャの中には、GNOMEやKDEのようにアプリケーションランチャアイコンやメニューを提供するものも多々ある（**画面3**）。qvwmなどは、Windows 9xにそっくりなSTARTメニューまで備えている。

これらは、いずれもウィンドウマネージャの付加機能と言っていい。使いやすさを主眼に据え、デスクトップ環境としての機能を盛り込んだ結果なのである。この辺が誤解を招きやすいところである。

■ウィンドウマネージャではない！

結論としては、GNOMEもKDEもウィンドウマネージャではない（キッパリ！）。GNOMEは、別にウィンドウマネージャが必要である。GNOMEに対応するウィンドウマネージャは数も多く、選択肢は豊富だ。最近ではEnlightenmentやsawfishといったウィンドウマネージャがよく利用されている。

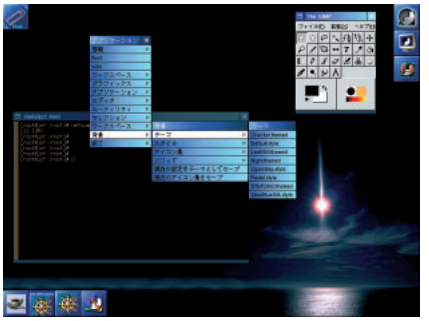
KDEはKwmやKWinというウィンドウマネージャと共に配布されており、通常はこれを利用するために混同されることが多い。しかし、KDEも他のウィンドウマネージャを利用することができるのだ。したがって、ウィンドウマネージャからは独立して存在していると考えてよい。

GNOMEやKDEはウィンドウマネージャではなく、その上で動くLinuxを使いやすくするための環境だと考えるとわかりやすいだろう。

ウィンドウマネージャを入れ替える

ちょっと横道にそれて、試しにウィンドウマネージャを入れ替えてみよう。KDEの場合、入れ替えにちょっと面倒な部分があるので、ここでは比較的容易なGNOMEを例に実験する。

今月号の付録CD-ROMに収録されているRed Hat Linux 7の場合、インストール時にGNOMEを選択すると、ウィンドウマネージャにsawfishがインストールされる。sawfishを利用したGNOME環境を**画面4**に示す。毎度おなじみの画面だ。ウィンドウ上部には、左に［コントロールメニュー］ボタンが、右に［最小化］ボタン、［最大化］ボタン、［閉じる］ボタンがそれぞれ並



画面3　アイコンドックやメニューを備えたウィンドウマネージャ。しかし、アプリケーション間の連携などデスクトップ環境が提供するコアな機能はない。

んでいる。タイトルバーにはWindowsのようなグラデーションもかかっている。ウィンドウの縁取りは細く、洗練された印象を受ける。

実はこれ、sawfishなのでこんな見栄えなのである。ここでは、そのことがよくわかるように、段階的にウィンドウマネージャを入れ替えてみる。

■ウィンドウマネージャを終了する

まずは、ウィンドウマネージャsawfishを終了させてみよう。どうなるだろうか？

root権限で、sawfishを強制的に終了させてみた。**画面5**のようにウィンドウ枠が消え、ウィンドウの移動やサイズ変更、重ね合わせの変更ができなくなってしまう。しかし、パネルやファイルマネージャ、デスクトップアイコンなどは残っているのがわかるだろう。マウスでクリックしようが、ドラッグしようが、画面の配置や重ね合わせは変わらない。しかし、ウィンドウ内の各機能や、アプリケーションの起動、ドラッグ＆ドロップも正常に機能している。

あぁ、なくなって初めてわかるウィンドウマネージャのありがたみである。

■fvwmとtwmで正体見たり！？

次に、この状態から別のウィンドウマネージャfvwm2を起動してみよう。ターミナルウィンドウのコマンドラインから起動する。すると、**画面6**のようになる。

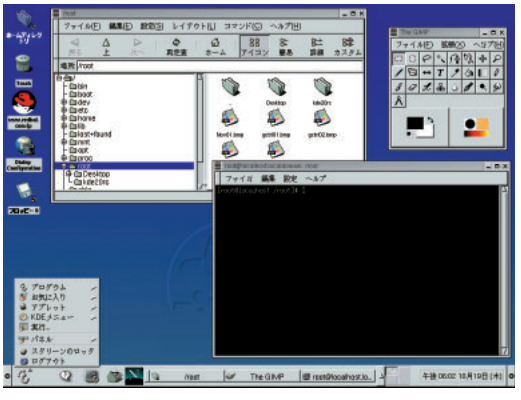
なんか、先ほどに比べるとちょっと野暮ったいウィンドウ枠が付いた。パネルにも枠が付いたりして、カッチョ悪い（注：筆者個人的見解）。さらに、fvwmBottunsを見ると、パネルやデスクトップアイコンもすべてウィンドウプロセスとして表示されている。当然と言えば当然なのだが、なんだか舞台裏が見えてしまった感じだ。

さらに、もっとプリミティブなtwmに変更したのが**画面7**だ。こちらは裏舞台がもっとわかりやすい。デスクトップアイコンにすべてタイトルバーが付いている。デスクトップアイコンは、xeyesのようにウィンドウ枠を持たないアプリケーションだったのね……。

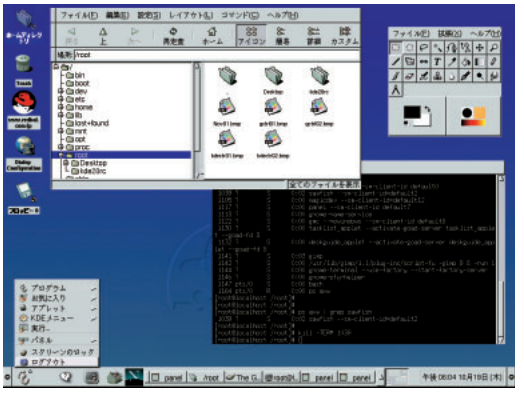
アプリケーションの充実が重要

ここで試したように、sawfishを利用しようが、twmを利用しようが、GNOMEで提供されるアプリケーション間での操作性は変わらないのである。逆に、GNOMEアプリケーションとそれ以外のアプリケーション間では、ドラッグ＆ドロップは使えない。たとえば、ファイルマネージャからXEmacsにアイコンをドラッグしても受け付けてくれない（当然だ）。

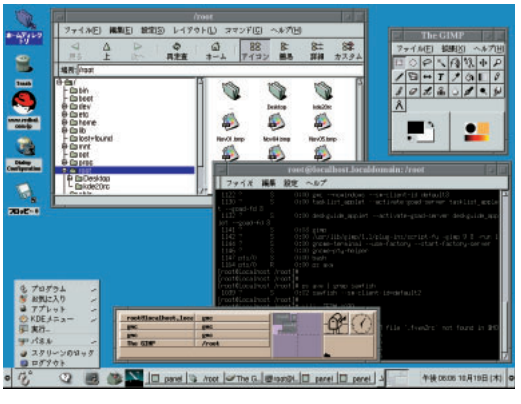
ここでの例はGNOMEだったが、KDEでも同様である。極端な例で言えば、GNOMEが起動している（という



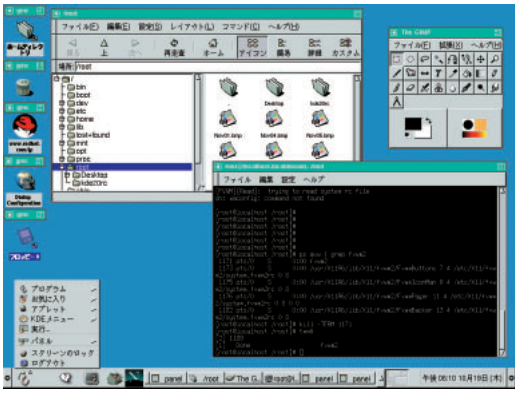
画面4　sawfishを利用したGNOMEアプリケーション。Red Hat Linux 7でログイン時に「GNOME」を選んだ場合の標準的な画面だ。ウィンドウ枠に注目しておこう。



画面5　ウィンドウマネージャがなくなったGNOMEアプリケーション。ウィンドウの枠がなくなり、ウィンドウの移動やサイズ変更ができないが、ウィンドウ内の各機能は正常に働いている。



画面6　ウィンドウマネージャにfvwm2を利用。ウィンドウ枠が変わったことが最初に目に付く。fvwm2のタスク切り替えパネルfvwmBottunsには、デスクトップアイコンやGNOMEパネルもタスクとして表示されている。



画面7　ウィンドウマネージャにtwmを利用。twmではデスクトップアイコンにタイトルバーが付いているようすがわかる。

表現は妥当ではないが）ところで、KDEのファイルマネージャとKDEの多機能エディタKWriteを起動した場合を見てみよう（**画面8**）。この2つのアプリケーション間ではドラッグ＆ドロップが利用可能なのだ。ちなみに、この場合sawfishを利用してKDEアプリケーションを実行していることになる。ただし、GNOMEアプリケーションとKDEアプリケーションとの間では、やはりドラッグ＆ドロップは機能しない（これまた当然）。

このように、GNOMEやKDEのアプリケーションはその環境内では格段に操作性がよいのである。

さてここでわかることは、インターフェイスそのものの良し悪しという前提条件をクリアしているとするならば、デスクトップ環境での快適な操作性をより完成度の高いものにするためには、すべてのアプリケーションをその環境で提供していく必要があるということになる（極論ではあるけれど）。このことから、アプリケーションの充実度と、それを構築するためのライブラリの充実度が重要な決め手になるのだ。

では、GNOMEとKDEはそれぞれどれほど充実しているのか？　それぞれの最新バージョンについては、後述する各解説を参照されたい。

「すべてのアプリケーションを提供」と書いたが、実は、GNOMEもKDEも大物アプリケーションとしてオフィススイートを提供する用意がある。

オフィススイートとは、Windows上のMicrosoft Officeに代表されるように、ワープロ、スプレッドシート（表計算）、データベース管理システム（DBMS）を中心に、いくつかのビジネスアプリケーションを集めたものだ。もちろん、ただ寄せ集めただけではなく、すべてのアプリケーションが連携するように設計されている。

これまで、企業での利用は主にサーバや開発環境用途であったLinuxだが、オフィススイートの登場で、デスクトップPCとしてのビジネス利用も広がるものと予想される。これらオフィススイートの登場が、GNOMEやKDEのさらなる普及の決め手となるかもしれない。

■GNOME Office

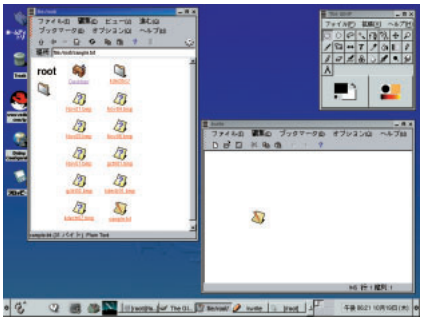
GNOMEには、人気のスプレッドシートのGnumericがあるが、これを含めた「GNOME Office Suite」が提供される予定だ。GNOME陣営の強力な牽引力であるGIMPもオフィススイートに加わる予定だ。既存の有力アプリケーションを採用することで、魅力あるオフィススイートになりそうだが、連携については新たに機能を付加しなければならない。これについては、サンが開発・公開しているOpenOfficeのコアを採用するようだ。

現在、GNOME Office Suiteにラインアップされているアプリケーションは以下のものである。

AbiWord	**ワープロ**
Gnumeric	**スプレッドシート**
GIMP	**画像処理**
Dia	**ダイアグラムエディタ**
EOG	**イメージビューア**
GNOME-PIM	**スケジュール管理**
GNOME-DB	**データベース管理**

Gnome Office Suiteについての詳しい情報は、以下のWebサイトを参照してほしい。

```
http://www.gnome.org/gnome-office/
```



画面8　GNOME上でKDEのファイルマネージャとKWriteを起動している。この2つのアプリケーションの間ではドラッグ＆ドロップも可能だ。

■KOffice

KDEでは、従来からKDE2.0と共に、KOffice 1.0というオフィススイートを提供すると宣言していた。これはすでにKDEのサイトにて開発中のものが公開されていたが、2000年10月23日のKDE2.0のリリースと同時に正式リリースされた。本誌発売日の関係もあって、ぎりぎりのタイミングで、リリース版について言及できないのが残念である。KOfficeは、先行していたプロジェクトだけにその完成度は高く、オフィススイート全体の統一感も高い。かくなるうえは、日本語版登場後に詳しく紹介する機会を得たい。ご期待あれ！

KOfficeに含まれるアプリケーションは以下のものである。

KWord	**ワープロ**
KSpread	**スプレッドシート**
Katabase	**データベース管理**
KPresenter	**プレゼンテーション**
KIllustrator	**ドロー系描画**
KImageShop	**画像処理**
KFormula	**数式エディタ**
KChart	**チャートダイアグラムエディタ**
Kimage	**イメージビューア**

KOfficeに関する詳しい情報は、以下のWebサイトを参照してほしい。

```
http://koffice.kde.org/
```

GNOME！GNOME！GNOME！

文：にゃー@編集部
Text: Nyaa@Linux magazine

ふだん何気なくLinuxをインストールして、何気なくXを立ち上げて操作していると、これまた何気なくGNOMEを使っていたりする。

デスクトップ環境というものは、それと気づかせずにユーザーにコンピュータを使わせるためのインターフェイスなのだから、何気なく使えるというのは実はとても大切なことなのだ。ここでは、そのインターフェイスを実現させているGNOMEが動作する仕組みを、ソフトウェアコンポーネント構成の面から探っていくことにしよう。

GNOMEのコンポーネント

ひとくちにGNOMEといっても、それは数多くのコンポーネントからなるソフトウェアの集合体なのである。配布されているソースコードのアーカイブも、その名のとおりコア部分にあたる「gnome-core」、デスクトップ設定ツールの「control-center」、アプレット集「gnome-applet」というふうに、コンポーネントごとにパッケージングされている。ちなみに、今月号の付録CD-ROMに収録した最新バージョン（ftp.gnome.org/pub/stable/latest/で配布されているソースファイル）の場合、関連するライブラリなどを含めて、総計50個のtarボールが提供されている(**表1**)。

バイナリパッケージの場合はさらに多くて、「rpm -qa | grep gnome」とすると、名前に「gnome」が付くパッケージがズラズラっと表示される。それ以外にも、GIMPやエレクトリックアイズ、Gnumericなどパッケージ名として「gnome」が付かない関連アプリケーションがある。これまたCD-ROMに収録してあるRed Hat Linux 7Jで数えてみたところ、ライブラリまで含めて83ものGNOME関連パッケージがあった（主なものを表1に示してある）。

依存するライブラリ群

GNOMEを利用するには、ベースとなるライブラリ群がインストールされていなければならない。GTK＋をはじめとして、glib、Imlib、ORBit、libxmlなどが必要だ。主なライブラリについて簡単に説明していこう。

■GTK＋

GTKが「GIMP Tool Kit」の略であることからわかるように、もともとはGIPMの開発用に作成されたツールキッ

ソースファイルアーカイブ		RPMパッケージ	
GnomeHello-0.1.tar.gz	gnome-network-1.0.2.tar.gz	audiofile-0.1.9-7.i386.rpm	gnome-users-guide-1.2-2.noarch.rpm
ORBit-0.5.3.tar.gz	gnome-objc-1.0.40.tar.gz	bug-buddy-1.0-6.i386.rpm	gnome-utils-1.2.0-7.i386.rpm
audiofile-0.1.9.tar.gz	gnome-pim-1.2.0.tar.gz	control-center-1.2.1-5j2.i386.rpm	gnorpm-0.9-27.i386.rpm
bug-buddy-1.1.tar.gz	gnome-print-0.24.tar.gz	dia-0.86-1j1.i386.rpm	gnotepad+-1.3.1-3.i386.rpm
control-center-1.2.2.tar.gz	gnome-python-1.0.53.tar.gz	dia-0.86-1j1.i386.rpm	gnumeric-0.54-4j1.i386.rpm
dia-0.86.tar.gz	gnome-utils-1.2.0.tar.gz	ee-0.3.12-1.i386.rpm	gtk+-1.2.8-7.i386.rpm
ee-0.3.12.tar.gz	gnomehack_1.0.1-1.tar.gz	esound-0.2.19-3.i386.rpm	gtk-engines-0.10-9.i386.rpm
esound-0.2.20.tar.gz	gnorpm-0.95.1.tar.gz	gdk-pixbuf-0.8.0-5.i386.rpm	gtop-1.0.9-4.i386.rpm
g-print-0.2.tar.gz	gnotepad+-1.0.8.tar.gz	gedit-0.9.0-3.i386.rpm	imlib-1.9.8.1-2.i386.rpm
gdk-pixbuf-0.8.0.tar.gz	gnumeric-0.57.tar.gz	glade-0.5.9-3.i386.rpm	imlib-cfgeditor-1.9.8.1-2.i386.rpm
gdm-2.0beta4.tar.gz	gtk+-1.2.8.tar.gz	glib-1.2.8-4.i386.rpm	libghttp-1.0.6-4.i386.rpm
gedit-0.9.1.tar.gz	gtk-engines-0.10.tar.gz	glib-devel-1.2.8-4.i386.rpm	libglade-0.13-4.i386.rpm
gfloppy-0.9.2.tar.gz	gtkmm-1.2.1.tar.gz	gmc-4.5.51-18j2.i386.rpm	libgtop-1.0.9-2.i386.rpm
ggv-0.95.tar.gz	gtop-1.0.9.tar.gz	gnome-applets-1.2.1-5.i386.rpm	libPropList-0.10.1-6.i386.rpm
glade-0.5.9.tar.gz	imlib-1.9.8.1.tar.gz	gnome-core-1.2.1-33j2.i386.rpm	libxml-1.8.9-5.i386.rpm
glib-1.2.8.tar.gz	libPropList-0.8.3.tar.gz	gnome-games-1.2.0-9.i386.rpm	mc-4.5.51-18j2.i386.rpm
gnome-admin-1.0.3.tar.gz	libghttp-1.0.7.tar.gz	gnome-kerberos-0.2.1-1.i386.rpm	mcserv-4.5.51-18j2.i386.rpm
gnome-applets-1.2.2.tar.gz	libglade-0.14.tar.gz	gnome-libs-1.2.4-11j2.i386.rpm	netpbm-9.5-5.i386.rpm
gnome-audio-1.0.0.tar.gz	libgtop-1.0.9.tar.gz	gnome-linuxconf-0.33-7.i386.rpm	ORBit-0.5.3-2.i386.rpm
gnome-chess-0.2.4.tar.gz	libole2-0.1.6.tar.gz	gnome-lokkit-0.41-5.i386.rpm	pygnome-1.0.53-4.i386.rpm
gnome-core-1.2.2.1.tar.gz	libxml-1.8.10.tar.gz	gnome-media-1.2.0-7j1.i386.rpm	pygnome-libglade-0.6.6-4.i386.rpm
gnome-games-1.2.0.tar.gz	libxml2-2.2.4.tar.gz	gnome-objc-1.0.2-9.i386.rpm	pygtk-0.6.6-4.i386.rpm
gnome-libs-1.2.3.tar.gz	mc-4.5.51.tar.gz	gnome-pim-1.2.0-5j1.i386.rpm	pygtk-libglade-0.6.6-4.i386.rpm
gnome-linuxconf-0.22.tar.gz	users-guide-1.2.tar.gz	gnome-print-0.20-8j2.i386.rpm	xchat-1.4.2-6j1.i386.rpm
gnome-media-1.2.0.tar.gz	xchat-1.2.1.tar.gz	gnome-users-guide-1.2-2.noarch.rpm	xmms-gnome-1.2.2-4j1.i386.rpm

表1　GNOMEのコンポーネント

ト。GTK＋は、ウィンドウやボタン、テキストラベルといったGUI部品（ウィジェット）などのライブラリであるGTKと、X上でのウィジェットの描画をサポートするライブラリであるGDK（GIMP Draw Kit）から構成される。GNOMEの標準ツールキットとして採用され、多くのプログラムがGTK＋を利用して書かれている。

■Imlib

フォーマットの異なるイメージデータをXのプログラムから透過的に扱えるようにするためのライブラリ。実際のプログラミングではGDKのAPIを介して利用される（gdk-imlib）。PNG、JPEG、TIFFなど、代表的なイメージフォーマットをサポートする。Imlibはさらに、xpm、libjpeg、libpngなどの各種イメージデータの処理をサポートするライブラリと依存関係にあるが、これらはほとんどのディストリビューションにおいてデフォルトでインストールされているはずだ。

なおGNOME 1.2からは、Xのイメージライブラリとして新しくpixbufが一部のコンポーネントで採用されており、gdk-pixbufも必須のライブラリとなっている。

■ORBit

分散オブジェクト管理のためのフレームワークとしてObject Management Group（OMG）により策定されたCORBA（Common Object Request Broker Architecture）に準拠したORB（Object Request Broker）のGNOMEでの実装にあたる。といっても、書いている本人がよくわかっていないのだが、ごく簡単に要約すると、プログラム（オブジェクト）とプログラムとのやり取りを仲介するブローカーの役目を果たす仕組みを提供するものだ。

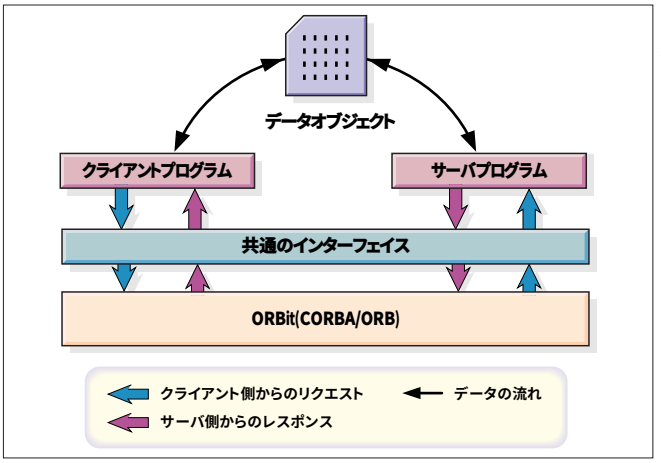


図1　ORBitの機能
この図は基本的な概念をごく簡略化して示している。各オブジェクト（プログラムやデータ）はネットワーク上やローカルコンピュータなど任意の場所に分散していてもかまわない。

それでなにが良いことなのかというと、クライアント／サーバに代表される多階層の分散コンピューティング環境における統一的なオブジェクト処理のメカニズムを提供できるようになるのだ（またわからなくなってきた。**図1**を参照してください）。

■libesd

サウンドの再生やミキシングを制御するためのAPIを提供するライブラリで、esoundパッケージに含まれている。GNOMEの効果音の再生に使われるサウンドサーバ（esd）も、このパッケージに含まれている。音関連では、オーディオファイル（WAV、AUなど）を扱うためのaudiofileライブラリも必要だ。

このほか、ほとんどのGNOMEアプリケーションから参照されるGNOMEのコアライブラリ（gnome-libs）、XML／HTML用のライブラリ（libxml／libghttp）、GTK＋の統合開発環境「glade」のライブラリ（libglade）といった数多くのライブラリがGNOMEの利用に必須となっている。

ややこしいのは、GNOMEがこれらのライブラリに依存するだけでなく、ライブラリ同士にも依存関係があることだ。たとえば、先ほどあげたようにImlibはxpmやlibjpegに依存しているし、gnome-libsはImlibに依存している。このような依存関係は、RPMパッケージにも反映されている。ソースからコンパイルする場合でも、RPMパッケージをインストールする場合でも、各コンポーネント間の依存関係に注意をはらう必要がある。

コアコンポーネント

前提となるライブラリに続いて、GNOMEのコアを構成するコンポーネントに目を向けてみよう。

まず、さきほど登場したgnome-libsがあげられる。gnome-libsは、libgnome、libgnomeui、gtk-xmhtml、libgnorbaなどから構成されるライブラリ集で、当然のことではあるが、GNOMEの実行には必須である。各ライブラリは他のライブラリと連携しながら、GNOMEの動作を支えている（**図2**）。縁の下の力持ち的存在だ。

そしてコア中のコアgnome-coreがある。gnome-coreには、デスクトップ、パネル、メニューシステム、セッションマネージャといったGNOMEの根幹部分が含まれる。また、GNOMEターミナルやヘルプブラウザ、Geditなどの

ユーティリティソフトウェア、いくつかのベーシックなアプレット（デスクガイド、タスクリストなど）も含まれている。そしてなぜだか「GNOMEさかな君」というアプレット（**画面1**）も、このパッケージの一部となっている。魚のイメージがアニメーションするだけのアプレットなのだが、これもコアなのか？

「GNOMEコントロールセンター」もコア部分に含められるだろう。GNOMEを使った経験のある方には、おなじみのデスクトップ設定ツールだ。コントロールセンターは、control-centerという単独のパッケージとして配布されている。

デスクトップの設定（パネル、テーマ、背景など）をはじめ、ファイルとアプリケーションを関連づけるMIME型の設定、起動オプションの指定、ウィンドウマネージャの選択とその設定などなど、GNOMEにかかわるすべての設定を一元的に行える便利なツールだ。コントロールセンターについては、72ページからのカスタマイズ方法の紹介でも解説しているので、そちらも参考にしてほしい。

rootユーザーとしてログインしてGNOMEを起動すると、「rootユーザーとしてファイルマネージャを操作するのはあんまりお勧めできません」という旨を告げるメッセージが表示される。「ファイルマネージャなんて起動してないのになぜ？」といった気がするかもしれないが、実はデスクトップに配置されているアイコンはGNOMEの標準ファイルマネージャであるgmc（GNOME Midnight Commnader）のものなのだ。GNOMEが動作しているときは、つねにgmcも動作している。ターミナルだけを立ち上げて「ps ax」コマンドを実行してみよう。その状態でもgmcが動作していることが確認できるはずだ。

gmcには、Windowsのエクスプローラに似たユーザーインターフェイスも用意されている（**画面2**）。見た目だけでなく、アイコンのドラック＆ドロップによるファイル操作や、ダブルクリックによるアプリケーション起動、マウスの右ボタンクリックで表示されるポップアップメニューなど操作性の面でもWindowsに迫るユーザーインターフェイスを提供してくれる。gmcはGNOMEデスクトップの優れた操作性を実現するために、欠かすことのできないコンポーネントなのである。

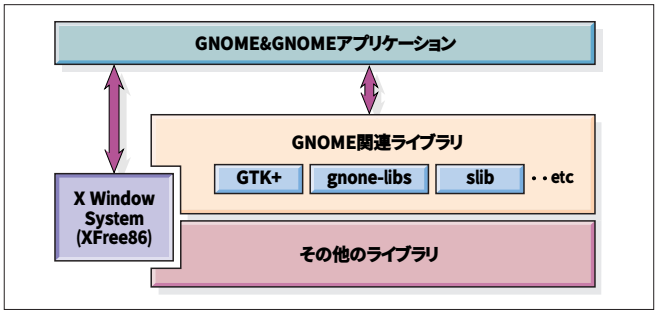


図2　GNOMEと関連ライブラリ
GNOMEの関連ライブラリとその他のライブラリ、そしてX Window Systemは緊密に連携しながら、GNOMEデスクトップの動作をサポートしている。

多彩なアプリケーション

パネル上で動作する小さなアプリケーションである「アプレット」をはじめ、GNOMEには実にさまざまなアプリケーション、ユーティリティが含まれている。

表2にGNOME 1.2の標準アプレットのすべてを紹介している。気に入ったアプレットがあったら、さっそくパネルに追加してみよう（設定方法は72ページを参照）。

アプレット以外のアプリケーションとユーティリティにも少し触れておこう。まずは、イメージビューア「エレクトリックアイズ」から（**画面3**）。ほとんどすべてのイメージファイルフォーマットに対応しており、フォーマットの変換もサポートする。ビューア機

画面1　GNOMEさかな君
すました顔でひたすらパネル内を泳ぐ「さかな君」。とってもアンビエントな存在なのだ。

画面2　GNOME Midnight Commander
メニューでは、たんに［ファイルマネージャ］と表記されている。プログラム名は「gmc」だ。

画面3　エレクトリックアイズ
起動直後に表示されるメインパネル。右ボタンクリックするとメニューが表示される。

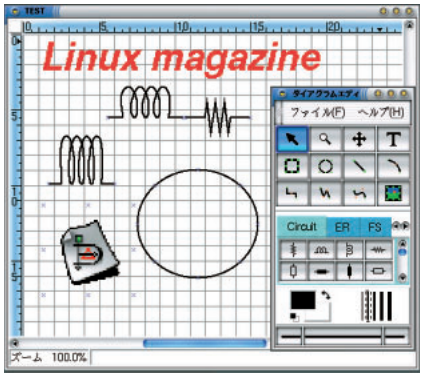
能以外にも、色調の補正、サイズ変更、反転／回転、切り取り、といった基本的な編集機能を備えている。

GNOME Officeにも採用される予定の「Dia」は、ダイアグラムを描画するためのアプリケーションだ（**画面4**）。フローチャート、組織図、地図などの作図に利用できる。部品も各種用意されていて、優れた操作性を備えている。作成したデータは専用フォーマットで保存できるほか、epsやcgm形式にエクスポートすることも可能だ。

やはりGNOME Officeに採用予定のスプレッドシート「Gnumeric」も優れモノのアプリケーションである（**画面5**）。すでに機能的には、ある程度完成されているので、あまりにも多機能になって使い勝手が悪くなることは避けてほしいところ。めったに使わないような特殊な機能は、プラグインによる追加で対応してくれると、必要に応じてオン／オフもできて便利だ。

GNOME Officeのひとつなのになぜ GIMPを紹介しないのか、とお思いの方もいるかもしれない。実は現段階でGIMPはGNOMEとは別に配布されているのだ。プロジェクトのWebサイトも別途設けられている。その意味では別格扱いということもできるだろう。

代表的なユーティリティも2つ紹介し



画面4　ダイアログエディタ「Dia」
矩形や円、ベジエ曲線などの一般的なパターン以外に、電気回線図やフローチャート用の部品が用意されている。

[ネットワーク]	
メールチェック時計	メールチェッカーと時計を組み合わせたアプレット。ちょっぴり多機能
メールチェック	メールチェッカーアプレット。シンプル・イズ・ベスト。場所をとらない
モデムモニタ	モデムの使用状況を表示するアプレット。接続状況、接続時間、スループットを表示できる
ウェブコントロール	パネルからURLを指定してWebページを開くアプレット
GNOME Stock Ticker	株価情報をテキストで表示する。銘柄指定も可能だがTOPIXには対応していないようだ
スラッシュドット	ニュースサイトSlahdot（http://slashdot.org/）からの情報を表示する
[遊び]	
15パズル	1から15までのパネルを番号順に並べていくパズルゲーム
Gnomeさかな君	ひたすら魚のアニメーションを表示する。何もしない
ライフゲーム	パネル上で動作するライフゲーム。もぞもぞ動いてコロニーを形成したりもするのだ
目玉	XeyesのGNOMEアプレットバージョン。マウスポインタを追いかけて目玉が動く
オドメーター	マウスポインタがデスクトップ上を移動した距離をカウントする。上段はトータル、下段はリセット後の距離を示している
[時計]	
クロック	標準のパネル時計。デジタル表示
もう一つの時計	CDE（Common Desktop Envinornment）風のパネル時計。アナログ
AfterStep時計	なかなかイカす外観の時計アプレット。テーマ機能もあり
JBC2進数クロック	2進数で時刻を表示する。み、見づらい
[モニタ]	
バッテリー充電モニタ	APMデーモンからバッテリ情報を取得して表示する。警告メッセージも出してくれるのだ
CPU負荷モニタ	CPU使用率を時間ごとに推移する棒グラフで表示するアプレット
CPU／メモリ使用状況	上段からCPU／メモリ／キャッシュの使用率をメーターで表示する
ディスク使用状況	ディスク使用率を円グラフで表示するアプレット。利用可能なディスク容量もひと目でわかる
メモリ負荷	メモリ使用率を時間ごとに推移する棒グラフで表示する
スワップ負荷	キャッシュ使用率を時間ごとに推移する棒グラフで表示する
ネットワーク負荷	ネットワークの負荷を時間ごとに推移する棒グラフで表示する
負荷平均	システム全体の負荷平均を時間ごとに推移する棒グラフで表示する
[マルチメディア]	
CDプレーヤー	アプレットバージョンの音楽CD再生ソフト。コンパクトさがうれしい
ミキサー	ボリュームコントロールアプレット。パネルから操作できるようにしたのはWindowsの真似？
サウンドモニタ	esound対応。テーマ機能でボリュームメーターやサウンドスコープモニタにカスタマイズ可能
[ユーティリティ]	
キャラクタピッカー	キーボードからタイプできない文字を入力するためのアプレット
デスクガイド	仮想デスクトップの状態を簡易表示する。デスクトップ・ウィンドウ間の移動もサポート
ドライブマウント	ドライブのマウント／アンマウントを行うアプレット。デフォルトはフロッピードライブに設定されているが、変更も可能だ
GKB国際化キーボード	キーボードマップを切り替えるアプレット
ミニコマンダー	メニューにある［実行］のアプレットバージョン。参照と履歴の機能も備えている
プリンタアプレット	ドキュメントファイルをドラッグ＆ドロップして印刷できるアプレット
クイックランチャー	パネル自体もランチャーとして機能するが、こちらのほうがシンプルでコンパクト
タスクリスト	カレントデスクトップで立ち上がっているウィンドウの一覧。もちろんウィンドウ切り替えの機能もある
Tick-a-Stat	coreファイルの生成や平均負荷などのシステムの状態を監視して、表示するアプレット
私はどこ？	マウスポインタのデスクトップ上での位置をXY座標で示すアプレット
お天気GNOME	世界各地の気象情報をネットワークから取得して表示できる。HTTPプロキシ経由の接続もサポート
スクリーンショット	画面をキャプチャしてイメージファイルに保存できる。ファイルフォーマットの変更やイメージの補正など細かな設定が可能
Gnotes！	メモ書きをデスクトップに貼り付けられる付箋ソフト

表2　GNOME 1.2に含まれるアプレット

ておこう。GNOMEシステムモニタ（gtop）とGnoRPMだ。gtopは、システムで動作しているプロセスをCPUとメモリの使用率の高い順にソートして表示するtopコマンドのGUIバージョン（**画面6**）。モニタリングだけでなく、プロセスを選択して、右ボタンメニューから詳細情報の表示やナイス値の変更、プロセスへのシグナルの送信を行える。

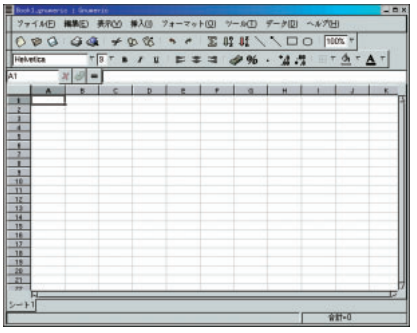
RPM系のディストリビューションを使っているのなら、GnoRPM（**画面7**）をお勧めしたい。rpmコマンドは、多機能なコマンドラインユーティリティとして確かに便利なのだが、同じ機能をGUI操作で利用できるのなら、それに越したことはない。rpmコマンドでは-qaオプションでいちいち確認しなければ、どのRPMパッケージがインストールされているのかわからない。GnoRPMなら、起動してすぐにRPMパッケージがアイコンで表示される。ツリー階層でパッケージを分類してあるのも、わかりやすくていい。この特集にあたっても、パッケージ内容の確認に、依存関係の把握にとこのユーティリティは大活躍したのであった。

KDEと違ってGNOMEはネイティブなウィンドウマネージャを持っていない（KDEの場合は1.X系のkfmと2.0のkwinがある）。したがって、ウィンドウマネージャは別途調達しなければならない。ある時期まで、Enlightenmentが多くのディストリビューションでGNOMEのウィンドウマネージャとして採用されていた。前のパートでも触れたように、Enlightenmentは非常に多機能なウィンドウマネージャで、単体でも統合デスクトップ環境に劣らないGUIを提供している。さきほど紹介したesoundも、もともとはEnlightenmentのイベント効果音の再生用に開発されたものである（esoundの正式名称はEsounD＝Enlightened Sound Daemonなのだ）。

しかし、多機能であるがゆえに動作が重く、特にGNOMEとともに動作させる場合は相当のマシンパワーが要求された。また、デスクトップの設定を行えるということは、機能的にもGNOMEとかぶる部分があるということでもある。たとえば、背景の壁紙をGNOMEとEnlightenmentの双方で設定できたりするのだ。

そこで、本来のウィンドウ管理だけに機能を限定したシンプルさと、GNOMEとGTK＋に対する親和性を備えたウィンドウマネージャが必要とされた。そこに登場してきたのがsawmill（当時。現在はsawfishに名称が変更されている。以下ではsawfishとする）である。

画面8に示すとおり、カスタマイズを加えずに単体で動作させた状態では非常にシンプルだ。ツールで設定できる項目もウィンドウと仮想デスクトップだけに限定されている。そのほかの設定と管理はGNOMEに「おまかせ」な



画面5　Gnumeric
外見や操作性はWindowsアプリケーションとほぼ同じ。はじめて使う場合にも、操作にとまどうことはほとんどないはずだ。

画面6　GNOMEシステムモニター
コンソール版のgtopに比べ各プロセスの状態がより見やすくなった。マウス操作だけで、選択したプロセスにシグナルを送ることができる。

画面7　GnoRPM
2ペイン構成のファイルマネージャライクなインターフェイスにより、直感的な操作が可能となっている。クエリーをかけると、そのパッケージの情報が1枚のパネルにまとめて表示される。

Column

vs. KDE

Red Hat、TurboLinuxをはじめ主要ディストリビューションで標準デスクトップ環境の地位を得ているGNOMEだが、実はライバルであるKDEよりも歴史は浅いのだ。

KEDは1996年に開発が始められ、最初のリリースである1.0が1998年の12月に公開されている。一方、GNOMEは1997年に開発がスタート。1.0の正式なリリースは1999年の3月である。

もっとも、これには至極当然の理由があって、もともとGNOMEは、qtのライセンス問題でフリーソフトウェアであるか議論のあったKDE（このあたりの詳細はXXページを参照）に代わる、完全にGPLに準拠した統合デスクトップ環境とGUIアプリケーション開発フレームワークを目指してプロジェクトが発足したのである。現在ではqtがGPLでも配布されるようになりKDEも同じ土俵にあがったといえる。勝負はこれからだ！？

のである。そのぶん動作も軽快なのだ。

こうした特徴を持ったsawfishは、リリース後の早い時期からGNOME用のウィンドウマネージャとして普及していった。現在では、Red Hat、Turbo Linux、LASER5、Kondaraなど、多くのディストリビューションで採用されている。

Enlightenmentとsawfishとsawfishとsawfish以外にも、scwm、IceWMといったGNOME対応のウィンドウマネージャがいくつかある。それぞれのウィンドウマネージャの説明とGNOMEへの対応状況については日本GNOMEユーザー会のWebサイトに詳しい情報が掲載されているので、興味のある方は参考にしてほしい（http://www.gnome.gr.jp/wm/）。

●

以上、GNOMEについてコンポーネント構成を軸に探ってきた。最後に少し趣きを変えて、GNOMEの今後について話しておこう。

GNOME Foundationとこれからの GNOME

今年8月に、米国のSun MicrosystemsやCompaq Computerなどの大手メーカーが中心となって、GNOMEのサポート団体「GNOME Foundation」が設立された。今後、GNOMEの普及を目標として、財政面や法的な問題への対処を中心にGNOME Projectを支援していく予定だ。

設立メンバーにはHewlett-PackardとIBM、そしてLinux陣営からRed Hat、TurboLinuxも加わっており、Sun Microsystemsとあわせてに有力な商用UNIX系ベンダーがすべて顔を揃えたことになる。各社とも自社製品の統合デスクトップ環境としてGNOMEを採用する方針で（ご存じのとおりRed HatとTurboLinuxはすでに採用している）、Linuxだけでなく UNIXの標準デスクトップとしてGNOMEが広まっていく可能性さえ秘めている。また、特集の最初のパートでも触れたように、Sun MicrosystemsはGNOME Officeへの技術的な支援も約束している。これは、KDEのKOfficeに比べてアプリケーション間の連携などの面で立ち遅れていたGNOMEのオフィススイート開発にはずみをつけるはずだ。

大きな期待をもって迎えられた新団体だが、UNIX陣営には離散集合を繰り返しながら分裂に至った苦い過去がある。今回もユーザーインターフェイスの統一と他のディストリビューションとの差別化という問題を内包している。ただ、現在はオープンソースソフトウェアという考え方がビジネス界にも浸透していること、GNU Projectから続く長い経験をコミュニティ側が経てきていることを考えると、再び同じ轍を踏むことはないだろう。いずれにしろ、今後のGNOME ProjectとGNOME Foundationの動向には要注目である。

Column

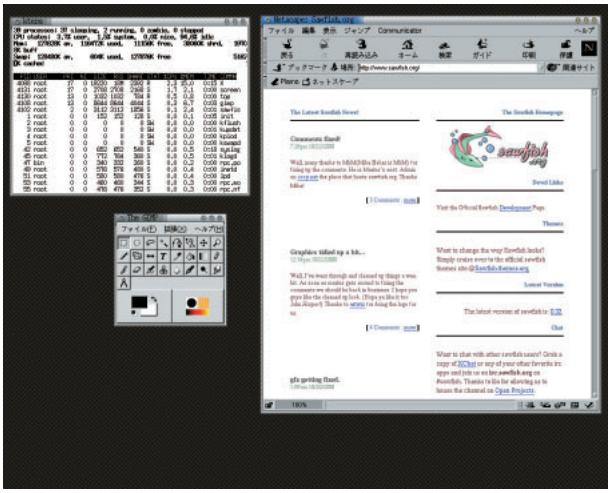
国際化への対応

GNOMEでは、基本的にベースとなるツールキットGTK＋の国際化機能を利用して、マルチバイト処理や多言語ロケールの切り替えをサポートしている。この際に参照されるのが、環境変数「LANG」である。LANGの値を変更するだけで、メニューやメッセージを変更することが可能になっている。たとえば、次のように「fr_FR」を指定してGNOMEを起動すると画面に示すようにメニューなどの表示がフランス語に切り替わる。

```
$ export LANG=fr_FR
```

メッセージやメニューなどのテキストは、各パッケージのソースファイルに拡張子.poを持つテキストファイルとして収録されており、コンパイル時に/usr/local/share/locale/ja/LC_MESSAGES（デフォルトのコンパイルオプションで日本語の場合）にバイナリのカタログファイルとして格納される（カタログファイルの拡張子は.mc）。.poファイルはパッケージごとに用意されているので、それぞれに対応言語が違っている点にも注目したい。これは国際化の不備よりも、その柔軟性を示している。決められた構文にしたがえば、.poファイルをユーザーが変更してカタログを再構築することも可能なのである。

このような高いレベルでの国際化への対応は、GNOMEのアドバンテージのひとつといえるだろう。



画面8　sawfishのデスクトップ
立ち上げた直後は何もないデスクトップのみが表示される。マウスの中ボタンクリックでメニューがポップアップして操作可能になる。

最新版GNOMEをインストールしよう！

新しいソフトウェアをインストールして使ってみるのは楽しい作業だ。GNOMEもまたしかり。少しばかり手間はかかるが、GNOME 1.2をインストールして、新しいデスクトップの世界を体験してみてはいかが？

お手軽なのは……

GNOME 1.2へアップグレードする最も簡単で最も確実な方法は、それが含まれているディストリビューションをまるごとインストールしてしまうことである。

日本語ディストリビューションでは、Kondara MNU/Linux 2000、Plamo Linux 2.1、Red Hat Linux 7Jが1.2系を収録している。アップグレードインストールが可能なら話は簡単だ。もしそうでなければ、データをバックアップしてから、ディストリビューションをインストールする方法が考えられる。この場合、システム全体の環境を再設定する必要があるため手間はかかるが、少なくともGNOMEは確実に動作するようになるはずだ。

ただ、ディストリビューションの変更となると、若干の抵抗を感じる方もいることと思う。細かい部分で設定方法が違っていたり、システムのコアコンポーネントが変わることで予期せぬ問題が起こる可能性もある。やはり、いまの環境でGNOMEだけをアップグレードするほうが得策だろう。

ソースからコンパイルする

GNOMEをアップデートする基本的な方法がソースからのコンパイルだ。そのためには、まず（当然のことながら）ソースコードを入手しなければならない。付録CD-ROMには「ほぼ最新版」のソースアーカイブを収録してあるので、こちらを利用していただくのが一番てっとり早い（なぜ「ほぼ」なのかという事情については、このページのコラムを読んでいただきたい）。

どうしても最新版がほしい場合は、コラムで紹介しているURLから入手しよう。このディレクトリに最新安定版のコンポーネントが収められている。

■バージョンの確認

ソースが入手できたら、コンパイルの前にGNOMEに関連するコンポーネントのバージョンを確認しておこう。

前のパートでも説明したように、GNOMEの各コンポーネントやその他のシステムコンポーネントは、互いに依存関係にある（特にライブラリ）。GNOMEのコンポーネント同士は、同じFTPサイトのディレクトリからダウンロードしたものであればバージョンの問題はないが、「GNOME対その他のシステムコンポーネント」については、それぞれのバージョンをチェックして依存関係に問題がないか確かめる必要があるのだ。

なにはともあれソースコードのtarボールを展開して、展開先のディレクトリにあるREADMEファイルを読んでほしい。依存関係について注意しなければならない場合には、インストールの前提となるコンポーネントとそのバージョンが記載されている。RPM系の場合、rpmコマンドを使ってバージョンを確認できる。

```
$ rpm -q パッケージ名
```

とすると、バージョン付きでパッケージ名が表示される。パッケージ名がわからない場合は、以下のようにして、-qaオプションで探っていくとよいだろう。

```
$ rpm -qa | grep lib
```

ほとんどのGNOME関連ライブラリには、その構成を表示するためのコマ

Column

最新バージョンをゲットするには

執筆時点でのGNOMEの最新版は1.2.3（gnome-coreのバージョン）であるが、付録CD-ROMには製作工程の関係上、10月20日時点でftp://ftp.gnome.org/pub/GNOME/unstable/latest/sources/ディレクトリで配布されていたtarボールを収録している（こちらはgnome-coreのバージョンが1.2.2.1）。

各コンポーネントはかなりの頻度で更新されているので、最新の安定版を確認したいときには、GNOME ProjectのWebサイトにあるインストールページ（http://www.gnome.org/start/installing/）を覗いてみよう。このページには、各コンポーネントの最新バージョンがリストされている。ソースファイルへのリンクも張られているので、ダウンロードも簡単に行えてGoodだ。

ンドが用意されている。コマンド名は「ライブラリ名−config」という形式になっている。このコマンドに「--version」オプションを付けて実行するとバージョンが表示される。たとえばgnome-libsの場合、以下のようになる。

```
$ gnome-config --version
```

共有ライブラリについては、/usr/libなどに置かれているライブラリファイルを参照する方法もある。ライブラリファイルは、「ライブラリ名.so.X.X.X」という名前になっている。Xの部分がバージョンにあたる。例外もあるがほとんどのケースでファイル名からバージョンを判別できるはずだ。上記の方法が使えないときは、これで確認しよう。

■インストールの順序は？

コンパイル〜インストールの段階でも、依存関係に注意が必要となる。「AはBに依存する」ということは、Aをインストールする前に、Bが存在している必要があることになる。つまりインストールの順番が大切なのだ。コラム内で紹介したGNOME ProjectのWebサイトのページに記載されているコンポーネントのリストは、インストールの順序になっている。ところが、FAQのページで解説されている順序はこれと違っていたりするのだ。

そこで、各コンポーネントのREADMEとINSTALLに目を通して依存関係をチェックし、上記のドキュメントを参考にしつつ、インストールに成功した順番が**表1**に示したものだ。表1に載っていないものは、特にインストール順を気にしなくていい。任意のコンポーネントをインストールしていこう。

インストール場所に注意

コンパイル前の作業があと2つ残っている。以下の作業を片づけておこう。

●パッケージ管理システム（RPMやDEB）を使ってインストールされたGNOMEがある場合は、それらのパッケージを削除しておく。

●ソースコードからの再インストールまたはアップグレードの場合は、ホームディレクトリにあるGNOMEの設定に関するディレクトリ（.gnome、.gnome-desktopなど）を削除する。

これも先ほどのダウンロードページに記載されているのだが、気づくのが遅れて、2つとも実行せずに一度インストールしてしまった。それでも動作はしたので、テスト的に導入するだけであれば、特に気にしなくてもかまわないだろう。

準備が整ったところで、いよいよコンパイルに移ろう。tarボールの展開先のディレクトリに移って、おなじみの「./configure」、「make」、「make install」を実行するわけだが、このとき、

```
./configure --prefix=<ディレクトリ名>
```

としてGNOMEのインストール先を指定できる。ほとんどのディストリビューションでは、「/usr」を指定するのが適当だ。この指定で/usr/binにプログラムが、/usr/libにライブラリがインストールされる。パッケージでインストールした場合のパスも、こうなっているディストリビューションが多い。

また、--sysconfdirオプションでシステム設定ファイルが置かれる場所を指定することもできる。これも、ほとんどのディストリビューションで「/etc」と指定して大丈夫なはずだ。これらの値は、そのディストリビューションが従っているディレクトリ配置のポリシーに従うのが賢明である。すでにGNOMEがインストールされていれば、次のコマンドで確認できる。

1.	audiofile
2.	esound
3.	glib
4.	gtk+
5.	imlib
6.	ORBit
7.	libxml
8.	gnome-libs
9.	libghttp
10.	libgtop
11.	libglade
12.	gdk-pixbuf
13.	control-center
14.	gnome-core
15.	mc
16.	gnome-print

表1　GNOMEのインストール順

```
$ gnome-config --prefix
$ gnome-config --sysconfdir
```

Filesystem Hierarchy Standard（FHS）に準拠した一部のディストリビューションの場合、/opt/gnomeと/etc/opt/gnomeとなる（たとえばSuSEがこれにあたる）。

整理すると、以下のコマンドをtarボールを展開したディレクトリで実行すればよいわけだ。

```
$ ./configure --prefix=/usr --sysconfdir=/etc
$ make
$ su
# make install
```

これを先ほどの順番でコンポーネントごとに繰り返していけば、GNOMEのインストールは完了だ。Xを起動してキチンと動作しているか確認しよう。

GNOMEのカスタマイズ

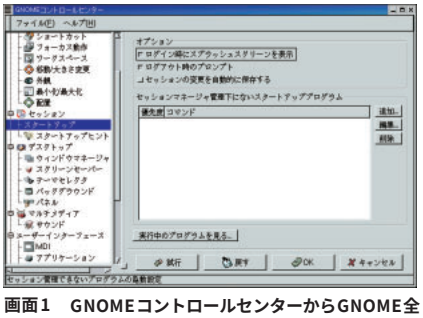
文：竹内充彦
Text: Michihiko Takeuchi

デスクトップ環境で使い勝手を大きく左右するのがデスクトップとパネルである。また、この2つの要素は、画面の見やすさや、カッコよさ、楽しさも左右する。ここではGNOMEのデスクトップとパネルを中心に、カスタマイズの方法を紹介する。自分好みのGNOMEに改造しよう！

デスクトップ環境カスタマイズの基礎

基本的なカスタマイズは、GNOMEコントロールセンターというツールで行える（**画面1**）。このツールは初期設定でパネルにランチャアイコンが用意されているし、メインメニュー（パネルの足跡アイコンをクリックすると表示されるメニュー）の［プログラム］－［デスクトップ設定］のなかにも用意されている。道具箱のアイコンが目印だ。どちらからも起動できる。

コントロールセンター内の基本操作は、①マウスで左側の階層メニューから項目を選択する、②右側にそれに応じた設定項目が表示される、③設定を変更するという手順になる。変更した設定は［OK］ボタンをクリックすると保存される。［試行］ボタンをクリックすると、保存する前に変更した設定を試すことができる。



画面1　GNOMEコントロールセンターからGNOME全体の設定が行える。環境のカスタマイズにはかかせないツールだ。左側の階層メニューから項目を選択し、右側で設定する。

パネルのカスタマイズについては、パネル自身で行う。メインメニューの［パネル］の下に設定項目が用意されている（**画面2**）。もしメインメニューのないパネルの設定をする場合は、対象となるパネルの上でマウスの右ボタンをクリックするとメニューが表示される。

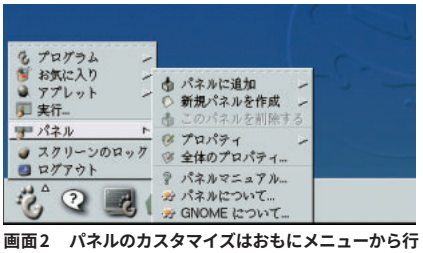
パネルのカスタマイズ

まずはパネルのカスタマイズから始めよう。パネルは初期設定では1つしかなく、しかも画面の下側に表示されている。パネルの個数は増やすことができるし、表示位置も変えられる。しかもGNOME1.2では、表示位置の自由度が増している。

■新規パネルの追加

カスタマイズの練習も含めて、ここで新規パネル（つまり2個目のパネル）を作成しよう。メニューの［パネル］－［新規パネルを追加］を選ぶ。その下にいくつかの項目が並んでいるはずだ。

［メニューパネル］は、メインメニューを持つパネルで、画面上側に表示される。このメニューパネルだけは特別で、画面上に1つしか存在できないうえに、配置を変えることができない。MacOSのようにプルダウンメニューで利用したい場合に向いている。



画面2　パネルのカスタマイズはおもにメニューから行う。メニューのないパネルの設定をする場合は、対象となるパネルの上で右クリックしてメニューを呼び出す。

［エッジパネル］は、画面の端から端までいっぱいに表示されるパネルで、初期設定で用意されているパネルがこのタイプである。

［アラインパネル］は以前、コーナーパネルと呼ばれていたものに近い。パネルに登録しているアイコンや、アプレットの表示に必要な長さだけ表示される。ただし、コーナーだけでなく中央に揃えることもできる。

［スライドパネル］は、［アラインパネル］に近いが、左右（または上下）のコーナー、中央以外に、画面の端から縦または横（択一）に自由にオフセットさせて表示することができる。

［フロートパネル］はさらに自由度が高く、画面の端から縦横ともにオフセットさせて表示することができる。つまり、画面の中央に「浮かせて」配置することも可能なのである。いろいろなパネルを配置してみた例が**画面3**だ。

■表示位置の変更

追加したパネルは画面の上側に表示される。また、［スライドパネル］と［フロートパネル］のオフセット値は、初期設定ではいずれも0であるため、そのままでは［アラインパネル］と区別が付かない。追加したパネルの表示位置や、オフセット値を設定しよう。オフセットさせたいパネルの上で右クリ

ックし、メニューから［パネル］－［プロパティ］－［すべてのプロパティ］を選択する。［パネルのプロパティ］ダイアログボックスが表示されるので、表示位置やオフセット値をピクセル単位で入力する（**画面4**）。

■大きさの変更

パネルが大きいとそれだけ作業する画面が狭くなってしまう。複数のパネルを表示しても同様である。パネルの大きさは、メニューの［パネル］－［プロパティ］－［大きさ］から選ぶことができる。［極小（24ピクセル）］を選択すれば、Windowsのタスクバーのように細いパネルになり、邪魔にならない。もっとも、パネルは両端にあるボタンでたたむこともできるし、［パネル］－［隠蔽方式］－［自動的に隠す］を選択すれば、必要な時以外は表示させないようにすることもできる。なお、後述するアプレットをパネルに登録すると、標準の大きさより小さくできないこともある。

■アプレットの登録

パネルには、アプリケーションランチャやちょっとしたユーティリティを登録することができる。これらのアイテムを小さなアプリケーションという意味でアプレットと呼ぶ。標準でも、メインメニュー、Netscape起動、GNOMEコントロールセンター起動、仮想デスクトップ切り替え、時計などのアプレットが登録されている。

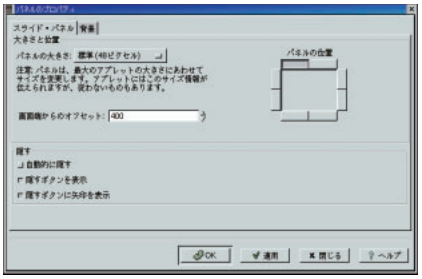
パネルにアプレットを登録する方法はいくつかある。もっとも簡単なのはランチャアプレットの登録だ。たとえばメインメニューにすでに登録されているアプリケーションならば、メニューの項目をそのままパネルの上にドラッグ＆ドロップすればいい。

ファイルマネージャからアプリケーションのアイコンをドラッグ＆ドロップすることもできる。この場合は［ランチャアプレットを作成する］ダイアログボックスが表示される（**画面5**）。ここでアプレットに名前や説明を付けたり、アイコンを選んだりすることができる。

ランチャ以外のアプレットも用意されている。メニューの［アプレット］から選択することもできるが、メニューの［パネル］－［パネルに追加］の下には、より多くのアプレットが用意されている。特徴あるアプレットを**画面6**に示す。

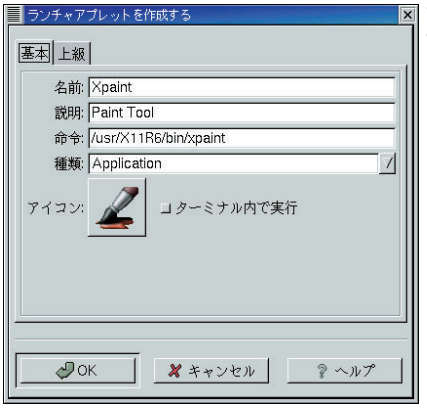
1段目は左から、メインメニュー、お気に入り、引き出し、スクリーンロック、ログアウト、コマンドを指定のアプレットだ。引き出しは、中にアプレットを格納することができる。2段目は左から、Tick-a-Stat（ログの監視）、スクリーンショット、ドライブマウント、CPUメモリ使用状況、ディスク状態状況メーター、メモリ負荷メーター、CPU負荷メーター、ネットワーク負荷モニタを配置している。

3段目は左から、CDプレーヤー、サウンドモニタ、ミキサー、15パズル、Gnomeさかな君（一言メッセージを表



画面4　［パネルのプロパティ］ダイアログボックスでは、パネルの方向（水平・垂直）や、画面の端からのオフセット値を設定することができる。

画面3　いろいろなパネルを配置。スライドパネルやフロートパネルを使えば、ピクセル単位で座標を指定して自由に配置できる。画面では引き出しアプレットなども登録して、さらに複雑なパネルを構成してみた。



画面5　ランチャアプレットに名前や、説明、命令を付けられる。命令には、そのファイルのパスが表示されるので、ここで引数やオプションを指定することもできる。

画面6　代表的なアプレット。

示)、ライフゲーム、オドメーター(マウスポインタの座標)、目玉(マウスポインタを追う)を配置している。

4段目は左から、RH PPP Dialer(モデムのダイアルと状態表示)、メールチェック、URLコントロール、ミニコマンド(ヒストリ付きコマンドライン)、GNotes!(付箋紙。付箋紙はクリックすると画面に付箋紙が表示)を配置している。

アプレットの並べ替えも簡単である。マウスの中央ボタンでドラッグすれば並べ替えられる。アプレットの削除は、アプレットの上で右クリックし、メニューから[パネルから削除]を選べばいい。

■メインメニューの変更

実はメインメニュー自体もパネルに登録されているアプレットだ。メニューの並びは操作性に影響する。メニュー項目は、追加/削除、並び順の変更が可能である。ただし、すべてのメニュー項目を変更するにはrootの権限が必要だ。一般ユーザーに開放されているのは「お気に入り(ユーザーメニュー)」の下に登録されている項目である(初期設定では何も登録されていない)。

カスタマイズには、メインメニューの[プログラム]-[デスクトップ設定]-[メニューエディタ]を選択する。すると、GNOMEメニューエディタが起動する(**画面7**)。

いちばん簡単なのは、既存のメニューの並べ替えだ。エディタの左側にメニューの階層が表示されているが、このメニュー項目を任意の場所にドラッグすればいい。並び順だけでなく階層の変更も可能だ。ユーザーメニューの変更はそのユーザーにしか反映されないが、それ以外のメニューの変更は、すべてのユーザーの環境に反映されるので注意しよう。

新規項目を追加するには、エディタ画面上部に並ぶ[新規項目]ボタンをクリックする。すると、右側に[名前]、[説明]、[命令]、[種類]という入力欄が表示される。名前はメニューに表示される文字列で、説明は覚え書き、命令は実行したいアプリケーションのパス、種類は登録する項目の種類である。種類には、「Application」、「Directory」、「URL」、「Applet」が選べる。Directoryを選べば、その下の階層に項目が登録できる。URLを選ぶと、メニュー項目からNetscapeを起動しブラウズできるようになる。Appletは、GNOMEパネルのアプレット用のもので、メニュー項目からパネルに登録される。入力欄の下にあるボタンをクリックすると、アイコンデザインを選べる。

既存のメニュー項目を削除するには、左側の階層メニューから削除したい項目を選択し、エディタ画面上部に並ぶ[削除]ボタンをクリックすればいい。

■背景色の変更と画像の貼り付け

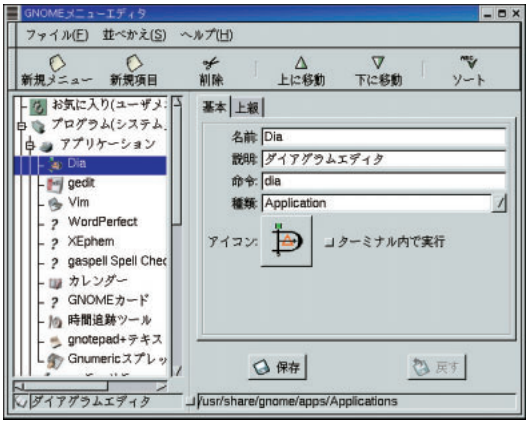
パネルの初期設定では色はグレーで無地だが、色は自由に変更できるし、画像を貼りつけることもできる。変更したいパネルの上で右クリックしてメニューから[パネル]-[プロパティ]-[すべてのプロパティ]を選択すると、[パネルのプロパティ]ダイアログボックスが表示される。ダイアログボックス内の[背景]タブを選択すると、パネルの色や、貼り付ける画像が選択できる(**画面8**)。

デスクトップのカスタマイズ

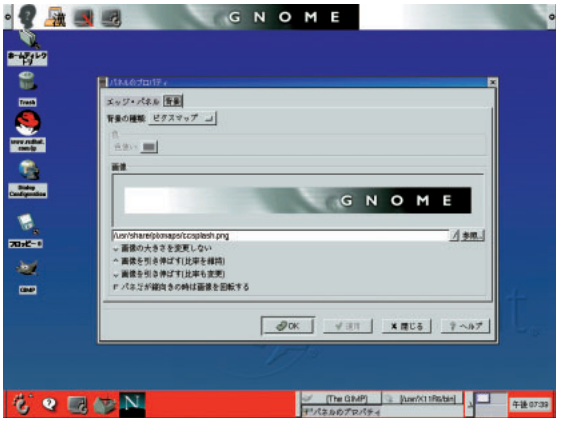
パネルが主に使い勝手を目的としたカスタマイズだとすれば、デスクトップはおもに見やすさや見た目の楽しさを目的としたカスタマイズだ。ただし、GNOMEコントロールパネルからsawfishの機能も設定できるため、仮想デスクトップの設定も可能である。

■スクリーンセーバの変更

スクリーンセーバを変更するには、GNOMEコントロールセンターの左側の階層メニューから[デスクトッ



画面7　GNOMEメニューエディタ。一般ユーザーが変更できるのは、[お気に入り](ユーザーメニュー)の部分だけだ。



画面8　[パネルのプロパティ]ダイアログボックスでは、パネルの色を変更したり、パネルに画像を貼りつけることができる。

プ］－［スクリーンセーバー］を選ぶ。右側のリストからスクリーンセーバの種類と、起動するまでの時間、解除時のパスワード要求の有無、プロセスの優先度、パワーマネージメントの利用などが設定できる（**画面9**）。ランダムスクリーンセーバは、リストにあるスクリーンセーバが無作為に選ばれ実行される。

大勢が出入りする場所で利用する場合は、パスワード要求のオプションを有効にしておくといいだろう。最近では、凝ったスクリーンセーバも多く、多大な処理能力を消費する。サーバマシンなど、コンソールでの利用者が退席中でも処理能力が必要とされるマシンでは、スクリーンセーバの優先度を下げておくといいだろう。

■背景色の変更と壁紙の貼り付け

Red Hat Linux 7の初期設定では、デスクトップの壁紙がRed Hatロゴをあしらったものになっている。これは、GNOMEコントロールセンターの［デスクトップ］－［バックグランド］で変更することができる。

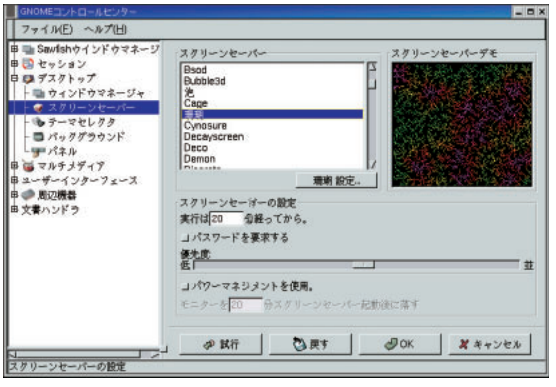
画像を貼り付けるには、［壁紙］セクションで画像ファイルを選択する。JPEG、TIFF、GIF、PNGなど、多くの画像フォーマットに対応している。デジカメ写真を取り込むもよし、GIMPでロゴを作成するもよしである（**画面10**）。画像の配置方法もオプションで選べる。なお、［Embossed Logo］オプションを選択すると、初期設定のRed Hatロゴのように、凹凸感のある表示になる（**画面11**）。

［色］セクションでは背景色が指定できる。グラデーションオプションを選択すると、画面の縦方向、または横方向に段階的に色を変えることができる。［プライマリーカラー］で選択した色から［セカンダリカラー］で選択した色へと段階的に背景の色が変わる（**画面12**）。

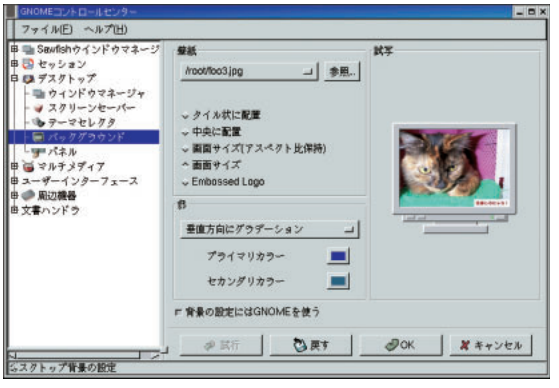
■デスクトップアイコンの登録

デスクトップ画面が広い場合、メニューからアプリケーションを起動するよりも、デスクトップにランチャアイコンを並べておいたほうが便利かもしれない。デスクトップアイコンを登録するのは簡単だ。登録したいアイコンを、ファイルマネージャからデスクトップに、あるいはメニューからデスクトップにドラッグ&ドロップすればいい。

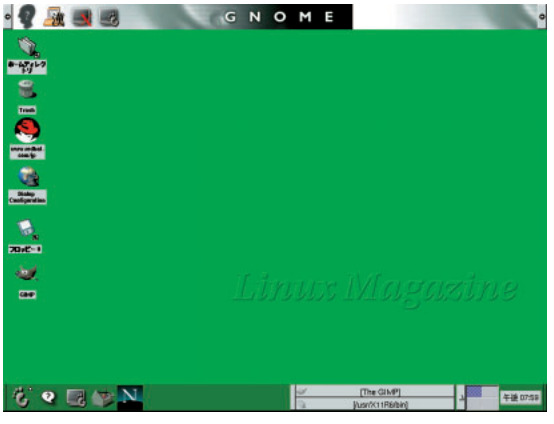
ただし、ファイルマネージャからそのままドラッグ&ドロップすると、通常はファイルが移動してしまう（ファイルマネージャ間でも同様）。たとえば、/usr/X11R6/binのアプリケーションに対してこれをやってしまうと、一般ユーザーの場合、元のファイルが



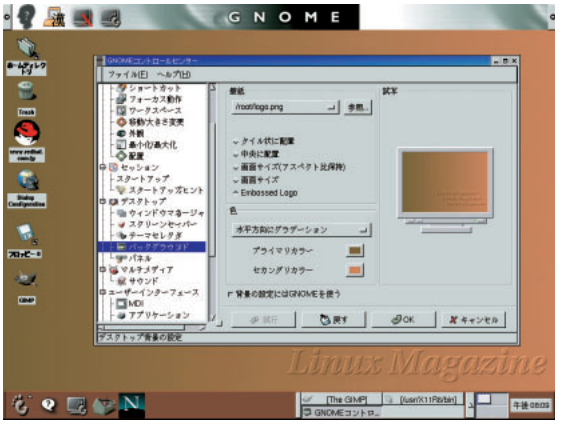
画面9　スクリーンセーバーは他人に画面の内容を見せないためにも設定しておきたい。しかし、サーバマシンの場合、それでパワーが下がっては意味がないので、優先度も設定しておくとよい。



画面10　壁紙に写真を貼り付ける場合は、［Embossed Logo］オプションをオフにしておく。中央に配置することも、タイリングにすることも、画面サイズに合わせることもできる。



画面11　［Embossed Logo］を使う場合は、元になるロゴ画像を白黒の2色で作成しておくだけで、立体的に表示してくれる。



画面12　グラデーションは水平方向と垂直方向のどちらかにかけることができる。画面は16ビット（65536色）モードでの表示。

削除できないという旨のメッセージが表示され、元のファイルを残したまま、そのコピーがデスクトップに作成される。もしroot権限でこれをやってしまうと、元のファイルが削除されてしまう。それに、デスクトップに置かれているアイコンの実体は、~/.gnome-desktop/というディレクトリの下に置かれるため、アプリケーションを各自のホームディレクトリの下にコピーすることになり、無用なディスク消費を招いたり、アプリケーションのバージョンアップに対応できないなどの問題が発生してしまう。

画面16　実画面と同じ広さのデスクトップがあと3つ用意されている。切り替えには、このアプレットの画面内をクリックすればいい。

これを防ぐには、Ctrl＋Shiftキーを押しながらドラッグ&ドロップをするといい。こうすると、デスクトップにはシンボリックリンクファイルが登録される（**画面13**）。シンボリックリンクファイルの内容は元ファイルを参照する情報だけなので、容量も小さいし、元ファイルのバージョンが変わっても対応できる。

デスクトップに配置したファイルのアイコンを変更するには、アイコン上で右クリックし、メニューから［プロパティ］を選ぶ。すると前出の**画面5**と同様なダイアログボックスが表示され、アイコンを変更することができる。

■テーマの変更

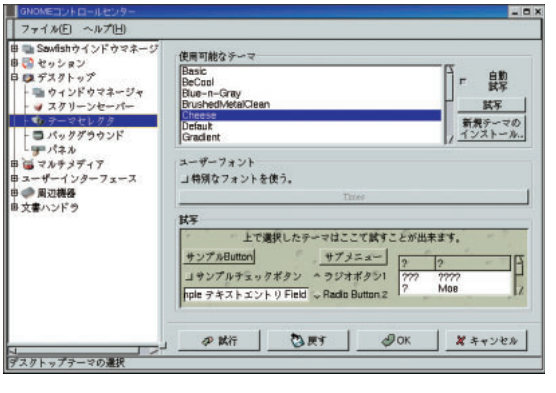
ダイアログボックスに配置されるボタン、チェックボックス、テキストボックス、リストボックスなどの各パーツのデザインも変更できる。GNOMEコントロールセンターの［デスクトップ］－［テーマ］を選択すると、右側にこれらのパーツの色などを変更する「テーマ」の一覧が表示され、変更することができる（**画面14**）。

■ウィンドウの外観の変更

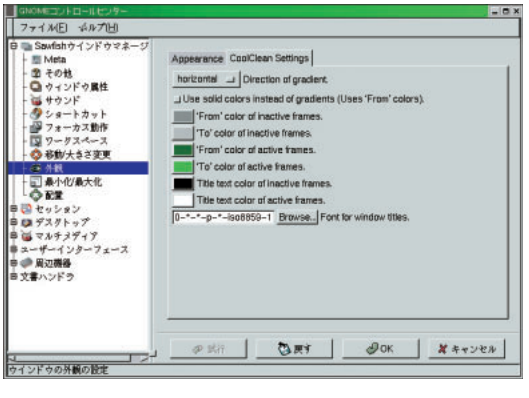
ウィンドウ枠のデザインはウィンドウマネージャが提供していると述べたが、sawfishはウィンドウ枠のデザインをカスタマイズすることができるようになっている。しかも、このデザインはGNOMEコントロールセンターから変更することができる。［Sawfishウィンドウマネージャ］－［外観］を選択する。

ここではウィンドウのタイトルバーの色を変更することができる。デスクトップ同様、段階的に色を変えることができる。［CoolClearn］タブをクリックしよう。［Direction of gradient］では、グラデーションの方向を選ぶ。

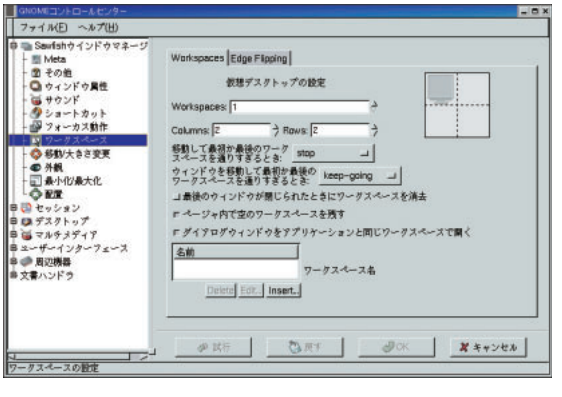
画面13　xpaintを、Ctrl＋Shiftキーを押しながらドラッグ&ドロップしたところ。アイコンにリンクファイルであることを示す小さな矢印マークが付いているのがわかる。



画面14　「テーマ」の変更でダイアログ内のボタンやリストボックスなどのパーツのデザインを変更することができる。画面は「Cheese」を選択したところ。［試写］の欄で確認できる。



画面15　sawfishが提供するタイトルバーの色を変更したところ。ここで非アクティブウィンドウについても同様に設定できる。また、タイトルバーに使用するフォントもここで設定可能だ。



画面17　仮想デスクトップの分割方法を指定する。ここでの設定は、sawfishを再起動しないと反映されないので注意！

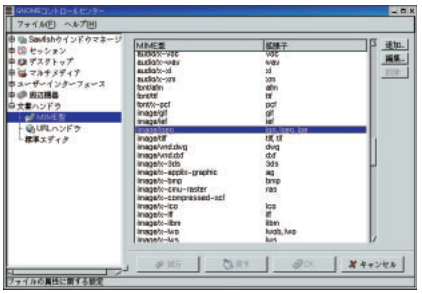
斜め（diagonal）、垂直（vertical）、水平（horizontal）が選べる。その下に、固定色を使うためのチェックボックスがあるが、これを選択（凹に見える）してしまうとグラデーション効果は得られない。その下に縦に4つ並ぶ色の付いたボタンは、上から順に、非アクティブウィンドウの開始色、同終了色、アクティブウィンドウの開始色、同終了色となっている。変更した例を**画面15**に示す。

■仮想デスクトップの分割と動作

sawfishは仮想デスクトップという機能を提供している。これは、仮想的に、1つの広いデスクトップを持ち、あたかもその一部が現在の画面に表示されているかのように扱う仕組みのことだ。Red Hat Linux 7の初期インストール状態では、仮想デスクトップは縦2画面分、横2画面分あり、4倍の大きさに設定されている。表示される部分は、パネルの［デスクガイド］アプレットで切り替え可能だ（**画面16**）。

この仮想デスクトップの広さや、縦横の並びを変えることができる。GNOMEコントロールセンターの［Sawfishウィンドウマネージャ］－［ワークスペース］を選択する。［Workspace］タブで、縦（rows）、横（columns）の画面数を指定する（**画面17**）。また、［Edge Flipping］では、マウスポインタが画面の端に来たとき、自動的に（仮想的に）隣のデスクトップに切り替わるように設定することもできる。

なお、ここでの変更は、sawfishを再起動したときに反映される。



画面18　image/jpegには.jpeg、.jpg、.jpeという拡張子が登録されている。

MIME型とアプリケーションの関連付け

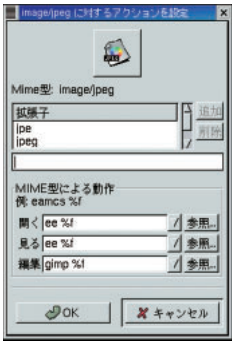
デスクトップとパネル以外の部分のカスタマイズを紹介しておこう。

GNOMEのデスクトップやファイルマネージャでは、データファイルのアイコンをダブルクリックするだけで、そのファイルに応じたアプリケーションが起動するようになっている。たとえば、JPEG画像のファイルをダブルクリックすると、必ず「ee」という画像ビューアが起動する。これは、ファイルのデータの種類と、起動すべきアプリケーションが関連付けされているためである。

この関係付けを変更したり、新たな関連付けを追加することもできる。GNOMEコントロールセンターの［文書ハンドラ］－［MIME型］を選択すればいい。右側にMIME型と拡張子の一覧が表示される。

たとえば、JPEGファイルの関連付けは「image/jpeg」という欄を見る（**画面18**）。拡張子に「jpg、.jpeg、.jpe」が関係付けられている。この欄を選んでおいて、［編集］ボタンをクリックすると、［image/jpegに対するアクションの設定］ダイアログボックスが表示される（**画面19**）。ここで、起動すべきアプリケーションを指定する。［MIME型による動作］という項目がそれだ。［開く］、［見る］、［編集］という3つの項目が設定できるが、これらは、ファイルマネージャでファイルアイコンを右クリックしたときに表示されるメニューにそれぞれ対応する。

また、ファイルマネージャで表示されるアイコンのデザインもここで変更することができる。



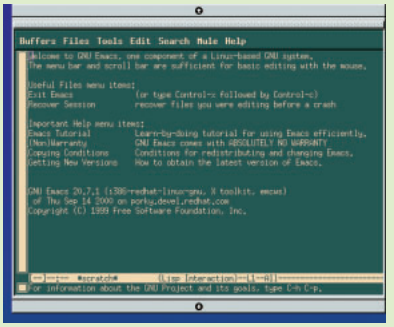
画面19　image/jpegに対するアプリケーションの関連付け。ここにeeが登録されているので、必ずeeが起動するのだ。%fの部分は、実行時にファイル名に置き換えられる。

Column

パネルの中でEmacsを実行！？スワロードアプリ

GNOMEが提供しているアプレットの中に「スワロードアプリ...」というものがある。これを使うとパネルの中にアプリケーションを「飲み込む」ことができるのだ。他の環境で作成されたアプレット（小さいアプリケーション）をパネルに並べることができるので、GNOME環境での見た目の統一に役立つ。

本来は、たとえばxbiffのような小さいアプリケーションを飲み込むのが正しい使い方なのだが、なんとEmacsもパネル内に飲み込めるのであった……（飲み込めないアプリケーションもあるので注意！）。



画面　パネルの中でEmacsがっ……！？　実は、これ、できるというだけでパネルが大きすぎて（500×400ピクセルほどある）ハッキリ言って邪魔なだけ。xbiffなどを飲み込むのが正しい。

統合デスクトップ環境KDE

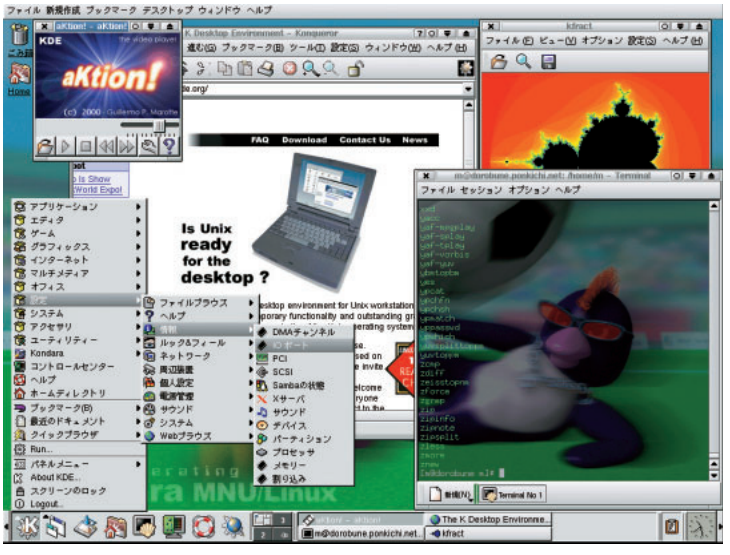
@K of the North Star[wakashi]
wakashi@kondara.org

KDEとは、「The K Desktop Environment」の略であり、優れた操作性、洗練されたデザインを併せ持つUNIX系OS用の統合デスクトップ環境のことである。使いやすくかつ美しいインターフェイスを持つ快適なデスクトップ環境を、UNIX系OSユーザーにも提供しようというものだ。

間違われやすいのだが、KDEはウィンドウマネージャではない。KDEのデスクトップはさまざまなパーツによって構成されており、その中で、ウィンドウマネージャと呼ばれるものは、KDEのバージョン1.x系においては「Kwm」、バージョン2.x系においては「KWin」である（以下KDE1、KDE2）。

起動時、デスクトップの下に位置するバー状のものがパネル（Kpanel）であり、アプリケーションスタータ、クイックランチと呼ばれるショートカットアイコンなどを持っている。デスクトップの上端にあるものはタスクバーと呼ばれ、現在開かれているウィンドウを表示する。それから、デスクトップ自身。そして、KDEの各種アプリケーションを起動した際のフレームになるウィンドウで構成されている（**画面1**）。

これらは全てKDEコントロールセンターからGUIでカスタマイズできるようになっている。これまで同様、設定ファイルを編集することで細かい設定をすることも可能だが、設定ファイルが複数にわたるため、KDEコントロールセンターを利用するほうが便利だろう。KDEコントロールセンターの細かい設定方法に関しては後述する。



画面1　さまざまなパーツからなるKDEの構成図

豊富なツール群

操作性に重点を置くKDEには、さまざまなツールが用意されている。たとえば、root権限でKDEを起動した際に使用できるツールには、GUIでinitを操作できる「Ksysv」や、ユーザーの追加や管理が行える「Kuser」などがあり、これまでコマンドライン上で操作していたよりも手軽にシステム管理ができる（root権限でXを起動するのはあまりお勧めできないのだが……）。そのほか、計算機のようなシンプルなツールから、スケジュール管理アプリケーションに至るまで、必要と思われるツールはほとんど揃っており、特別困ることもないだろう。さらに、KDE2からはオフィスアプリケーションが標準パッケージとなり、まさに痒いところにも手が届くソフトウェアといえる。KDEを便利と感じるか否か、あとはユーザーの使い方次第ということだろう。

KDEの目指すもの

このように、あらゆるツールやGUIアプリケーションが揃っているという点や、インターフェイスがWindowsライクであることから、KDEは大きな誤解を受けてきた。それは「操作性に優れている＝初心者向けのデスクトップ環境」だと思われ、プロフェッショナルユーザーからは敬遠されがちだったことだ。そもそも、KDEはユーザーが親しみやすいデスクトップ環境を提供することを目的としている。多くのユーザーを獲得するということは、初心者から開発者まで、幅広い層のニーズを満足させるものでなければならず、すなわち使いやすく、かつカスタマイズ性の高いソフトウェアでなければならないことを意味する。KDEはその条件を十分満たしているといえるのではなかろうか。

KDE2においては、よりWindowsライクになった感もあるが、これは将来的

にWindowsからの移行者が増加することを視野に入れたうえでのことだろう。「エンドユーザー、プロフェッショナルユーザーを問わず、幅広いユーザーの方に快適なUNIX（Linux）環境を提供する」さらに、「最強（Best）と呼ばれる統合デスクトップ環境になる」これがKDEが目指すものだ。

世界中に広がる開発者＆ユーザーグループネットワーク

KDEの開発には、おもに「The KDE Team」と呼ばれる、世界中に広がる開発者グループが、ネットワークを通じボランティアベースにて参加している。さらには、世界中のユーザーグループもさまざまな形で参加している。日本には、「日本KDEユーザー会（JKUG）」があり、本家からの情報を日本語に翻訳して公開したり、パッチをサイトにて公開するなどして、KDEのさらなる普及に日夜努力している。最近ではKDE2の日本語化への対応の方法に関する情報も紹介しているのでぜひ参考にしてほしい（**表1**）。

KDEの近況

ちょうど1年くらい前には、日本語の対応が不十分であったKDEだが、KDE1.1.2では通常使用するアプリケーションにおいては何の問題もなくなった。KDE2.0に関しては、まだ日本語への対応は不完全だが、KDE自身がi18n（Internationalization：国際化を意味する）を念頭においたUTF-8ベースでの開発となっているので、KDE自身の安定はむろんのこと、KDE以外の環境の安定いかんによっては早期実現が可能だと予想できる（**画面4**）。KDE1.1.2がリリースされた1999年9月13日から約1年後の2000年10月23日には、KDE2.0がリリースされた。

これまでのKDE1系はGUIツールキットであるQt1.x系をベースに、KDE2はQt2.2をベースにしている。

Qt1.x系の時には、QPLというライセンスの元で配布されており、改変不可だったのだが、Qt2.0以降はGPL/QPLのデュアルライセンスという形となり、誰もが改変および配布できるようになった。これで、かねてからのライセンス問題も解消された。

執筆時点の安定版であるKDE1.1.2は、すでに多くのメジャーなディストリビューションに搭載されている。詳しくは、http://www.kde.org/kdeships.htmlを参照してほしい。KDE2.0に関しては、まだまだ安定版と呼ぶには少々時間がかかりそうだが、KDE1系のときにも、安定するにはKDE1.1.2までの経過が必要だったのだから仕方ないだろう。

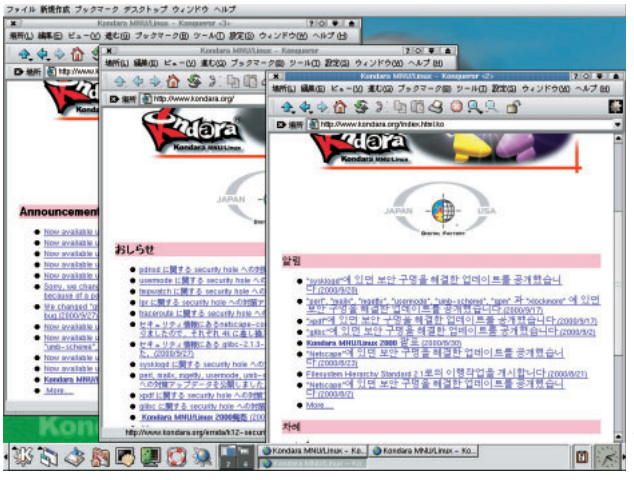
なお、来年の3月頃にはQt3をベースとした、KDE2.2がリリース予定になっている。執筆時点ではQt3および、KDE2.2に関する詳しい情報は皆無といっていいのだが、ライセンス問題の解消によって、世界中のプログラマーがKDE開発に参加してくると予想される。今後、開発スピードがさらに向上するのは間違いないだろう。

KDE本家Webサイト	http://www.kde.org/
日本KDEユーザー会Webサイト	http://www.kde.gr.jp/

表1　開発者＆ユーザーグループネットワーク

画面2　本家KDEのWebサイト（http://www.kde.org/）

画面3　日本KDEユーザーズグループ（http://www.kde.gr.jp）



画面4　3カ国語で表示されているKonqueror

KDE1.xとKDE2の比較

KDE1.xを最初に起動した時には、OpenLinuxなど一部のディスリビューションを除いて、ナビゲータキャラなどは登場しなかったが、KDE2では「Kandalf」という魔法使いのおじいさんが、Tipsをたずさえて登場する。Kandalfには、アプリケーションスタータメニューの［アクセサリ］－［Kandalfの一言］を選択することによってもお目にかかれる（**画面5**）。

ウィンドウマネージャは前述したように、「Kwm」から「KWin」となり、ファイルマネージャである「Kfm」は「Konqueror」となる。そして、最も大きな違いといえば、KDE2にはオフィススイートであるKOfficeが含まれることだろう。これまでKOfficeがなかったわけではないのだが、KDEの標準パッケージとしては提供されていなかった。このKOfficeが標準パッケージとして利用できるというのは嬉しい限りだ。

それでは、これら個々の注目アプリケーションについて見ていこう。

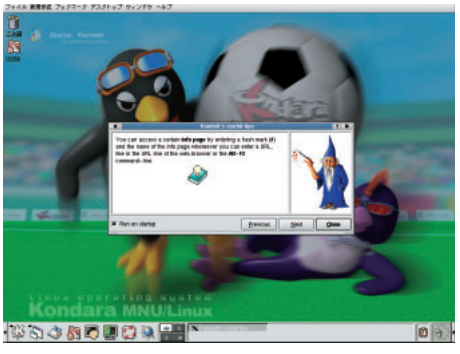
次世代のマルチブラウザ Konqueror

Kfmに代わる次世代のファイルマネージャ＆Webブラウザとして期待されているのがKonquerorだが、Kfmに備わっていた機能が数段高機能になったのはいうまでもない。Konquerorのおもな機能は以下の通りだ。

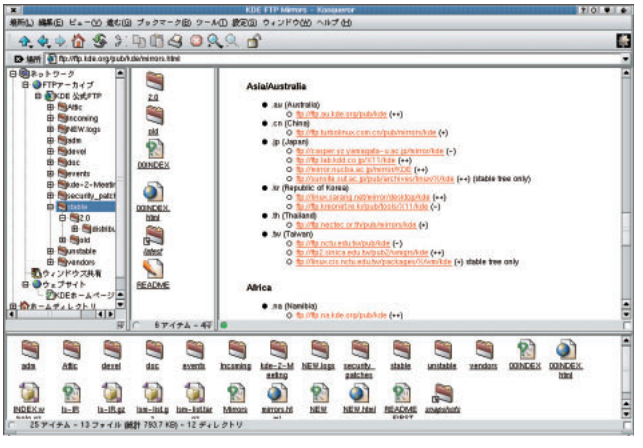
- **ファイルマネージャ**
- **Webブラウザ**
- **ドキュメントブラウザ**

まずはファイルマネージャとしての機能だが、Sambaを経由したネットワーク上のWindowsのファイル情報や、マウントしたローカルファイルの情報まで閲覧可能となっている。従来通り、FTPクライアントとしての機能も当然備わっている（**画面6**）。

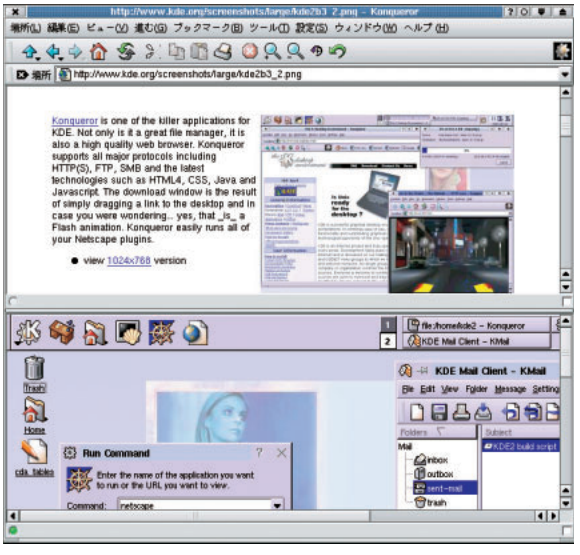
次にWebブラウザとしてのKonquerorだが、課題とされていた新しいバージョンのHTMLやスタイルシート、SSLへの対応も実装されている。Javaアプレット、JavaScriptに関して



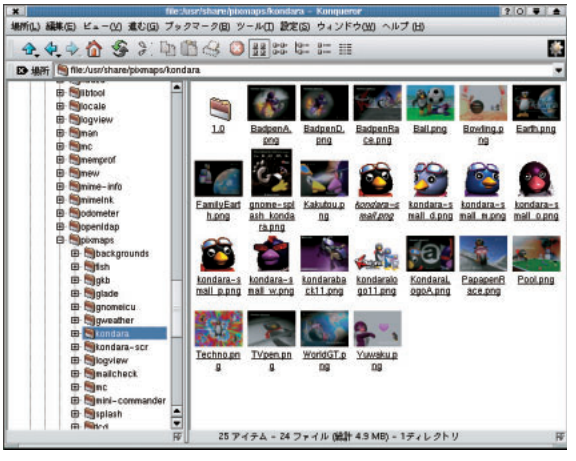
画面5　案内役のKandalfが登場



画面6　FTPクライアントとして機能するKonqueror



画面7　Webブラウザとして機能するKonqueror



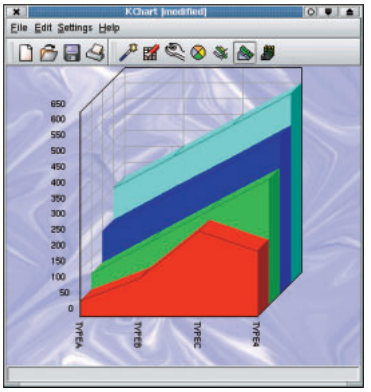
画面8　ツリー表示とファイル詳細(イメージビューア)を実現している

も対応済みだ。残念ながら執筆時点ではフレームには対応していないように思われるが、それを除けばNetscapeからの移行を考慮できる域に達したかもしれない（**画面7**）。

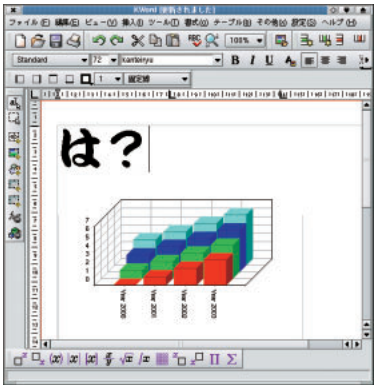
3番めのドキュメントブラウザとしての機能だが、これまでKfm上でプレーンテキストをクリックすると、Keditが起動してテキストを表示していたが、Konquerorでは同一ウィンドウ上にテキストを表示するようになっている。プレーンテキストだけではない。Kfmにおいても部分的に実装されていたイメージビューアとしての機能もさらに充実し、同一ウィンドウにてのサムネイル方式での閲覧、個別の画像閲覧もできるようになった。

驚くべきは、KOfficeのドキュメントを特別なビューアを用意しなくとも閲覧できるという点だ。そのほか、自分が閲覧したいと思うファイルタイプを登録するだけで、あらゆるファイルタイプが閲覧可能となる（**画面8**）。

おまけに、都合に合わせて任意に画面を区切ることができるので、複数のKonquerorを起動する必要もない。当然、これまでと同様にドラッグ&ドロップも実装されており、便利さ200%といったところだ。使い方に関してはのちほど詳しく述べるとしよう。



画面9　kchart



画面10
kwordとkchartの連携

KDE2の期待の星「KOffice」

KDE2の目玉の一角を担うのが「KOffice」だ。KOfficeには、「KWord」、「KSpread」、「KIllustrator」、「KPresentor」などが含まれる。これらはKDE2用にある程度機能がフリーズされたものだが、これ以外にもKOfficeにはさまざまなアプリケーションがそろっている。KOfficeに含まれるアプリケーションはそれぞれ**表2**の通りとなっている。KOfficeに関しての詳しい内容はKOfficeのオフィシャルWebサイト（http://www.koffice.org/）を参照してほしい。

KOfficeアプリケーション間にはすべて互換性があり、たとえば、KPresentorにKSpreadの表を貼り付けるということも可能だ。

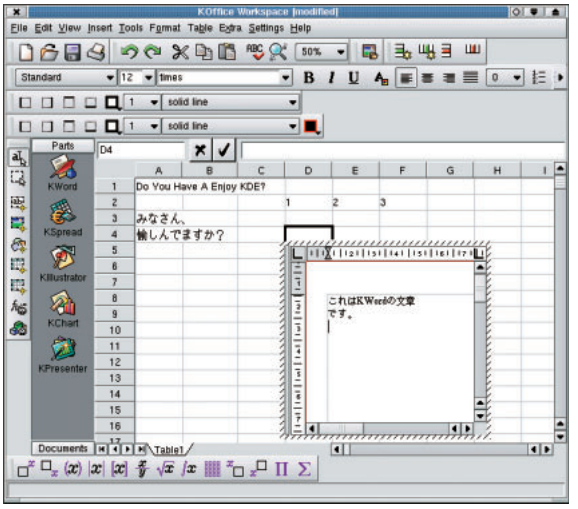
ほかのOfficeアプリケーションにも引けを取らない仕様となっているKOfficeだが、正式版がリリースされたばかりであり、つい最近までは日本語への対応など、さまざまな課題も残されていた。だがここにきて、不完全ながら日本語入力可能な状態にまでにたどり着き、安定感もさらに増してきた。もちろん、これですべてが満点な状態で動作するわけではないのだが、フリーにて使用できるオフィスアプリケーションとしては大きな躍進といえるだろう。

どうだろう、だんだんKDE2が魅力的なものに思えてきて、一刻も早く使ってみたい気持ちになったのではないだろうか。

しかしながら、KDE2が安定してないうちは、すでに安定しているKDE1.1.2を使いつつ、KDE2も使ってみたいというユーザーも多くいることと思う。

新旧KDEの共存方法については、のちほど詳しく解説することにする。

画面11　KOffice Workspace



名称	説明
KWord	ワードプロセッサアプリケーション
KSpread	スプレッドシートアプリケーション
KIllustrator	ベクタードローイングアプリケーション
KImageShop	ビットマップドローイングアプリケーション
KPresentor	プレゼンテーション用アプリケーション
Katabase	データベースアプリケーション
KFormula	公式ドローイングアプリケーション
KChart	チャート&ダイアグラムドローイングアプリケーション
KImage	シンプルなイメージビューア

表2
KOfficeのアプリケーション

KDE2.0のインストール

何はともあれ、KDEを入手しなくては始まらない。KDE2.0を含め、KDEは本家を始め、世界中のFTPミラーサイトから入手できる。

それよりもまず、KDEをインストールし動作させるためには、Qtが必要となる。KDEをインストールする前に必ずQtをインストールする。

KDE1.1.2においてはQt1.42もしくはそれ以上、KDE2.0においてはQt2.2以上が必要となる。ただし、KDE1.1.2はQt2.0もしくはそれ以上では動作しないのでご注意いただきたい。

Qtの入手に関してはTrolltechのWebサイトを参照してほしい。自分でKDEをコンパイルしたい場合などには、そのほかの開発ツールやライブラリが必要となるかもしれない。その際、自分のシステムに必要と思われる開発ツールやライブラリなどを各自で用意していただきたい。

なお、KDE2で日本語を使用するためには、Qtにパッチをあてる必要があるので、こちらも入手する。下記のサイトには、KDE2関係の修正パッチも置いてある。

KDE2と一口に言っても、実際には複数のパッケージで構成されている。パッケージの構成と内容については**表4**を参照してほしい。

KDE2のソースはKDEのFTPサイトで入手できる。なお、今月号の付属CD-ROMに収録時点の最新版のKDE2.0とQtも収録してあるので利用してほしい。原稿執筆時点の最新版は、2.0RC2（Release Candidate 2：Final Beta版）である。

ftp.kde.orgミラーサイト一覧	http://www.kde.org/mirrors.html
Qtのダウンロードサイト	http://www.trolltech.com/products/download/
QtのFTPサイト	ftp://ftp.trolltech.com/qt/source/
KDE2.0およびQtのパッチ入手サイト	http://www.kde.gr.jp/patch/index-ja.html
KDE2.0のソースダウンロードサイト	ftp://ftp.kde.org/pub/kde/unstable/distribution/
.qti18nrcの書き方の解説サイト	http://www.kde.gr.jp/~tino/qti18nrc.html

表3　KDE2.0およびQtの入手サイト一覧

▼リスト1　Qtのパッチあてとmake

```
# tar zxvf qt-x11-2.2.1.tar.gz
# patch -p0 < qt-2.2.1-codec-20001008.diff
# patch -p0 < qt-2.2.1-xim-20001014.diff
# patch -p0 < qt-2.2.1-m17n-20001008.diff
# patch -p0 < qt-2.2.1-qpsprinter-20001008.diff
# patch -p0 < qt-2.2.1-qclipboard-20001007.diff
# cd qt-2.2.1
# export QTDIR=`/bin/pwd`
# ./configure
#  make
```

▼リスト2　Qtのインストール

```
# cd ..
# mv qt-2.2.1 /usr/local/qt
```

▼リスト3　環境変数の設定

```
# QTDIR=/usr/local/qt
# PATH=$QTDIR/bin:$PATH
# MANPATH=$QTDIR/man:$MANPATH
# LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
# export QTDIR PATH MANPATH LD_LIBRARY_PATH
```

インストール方法

KDE 2.0をコンパイルするには、まずQtをインストールしなければならない。原稿執筆時点で公開されているQtの最新バージョンは2.2.1である。qt-x11-2.2.1.tar.gzを入手し、**リスト1**のようにソースアーカイブを展開し、日本語を利用するのであれば**表5**のパッチをあて、makeする。なお、glibc-2.1.9xではDEBUGオプションを使用してQtをコンパイルしないとKDE2が起動できないという問題があるようだ。

以上でQtのコンパイルは終了だ。Qtはmake installする必要はないが、/usr/local/qtなどのディレクトリにmvするといいだろう（**リスト2**）。

ここで環境変数をセットする必要がある。.bash_profileなどに記述してもいいだろう。QTDIRはQtをインストールしたディレクトリを指定する（**リスト3**）。

Qtをインストールしたら、いよいよKDEをインストールしよう。

KDEパッケージをインストールするには順番がある。どのパッケージよりも先にインストールしなければならないのはkdesupportだ。2番めにkdelibs、3番めにkdebaseという順でインストールしなければならないのでご注意いただきたい。

①kdesupport

②kdelibs
③kdebase

そのほかのパッケージに関しては特に順番は気にする必要はない。

まず、KDE2をインストールするディレクトリを環境変数KDEDIRにセットする。

環境変数をセットしたら、次の手順でパッチあてとコンパイル、インストールを行う（**リスト4**）。

以上でインストールは完了だ。

KDE1.XからKDE2へアップグレードする

KDE2とKDE1.xは環境設定ファイルとして、どちらも「.kderc」、「.kde」、「Desktop」といった、ファイルおよびディレクトリを使用している。現在、KDE1.xを使用している環境にKDE2をインストールしてKDE2を起動すると、元々KDE1.xで使用していた上記の環境設定ファイルなどを上書きアップグレードしてしまう。

これまでの設定がなくなってしまったと後悔しなくてもいいように、必要ならば必ずバックアップを取っておこう。そのほか、メールや大事なドキュメント類も同様にバックアップを取っておいたほうが無難だろう。後述の共存を行うために、ここでは、「.kde1x」、「.kderc1x」など、ファイル名の後に「1x」をつけてバックアップしよう。

前述したが、アップグレードするといっても、KDE2をインストールするための環境と、これまでのKDE1.xに必要とされた環境では少々異なるので必ず確認してほしい。

画面12　KDE／Qtパッチ集。KDE2.0およびQtのパッチ入手サイト（http://www.kde.gr.jp/patch/index-ja.html）

パッケージ	説明
kdesupport	以下の全てのパッケージで必要とされるライブラリ。KDEをインストールするには必ず必要となる
kdelibs	以下のKDEパッケージで必要とされるKDEの共有ライブラリ
kdebase	KDEで動作するアプリケーションのベースとなるパネルやウィンドウ、デスクトップを形成するためのアプリケーション
kdegames	KDEで動作するさまざまなゲーム集
kdegraphics	イメージビューアやドローイングアプリケーションといったアプリケーション
kdeutils	エディタや計算機アプリケーションのようなさまざまなデスクトップツール群
kdemultimedia	音楽を聞く、編集するというようなマルチメディアに関するアプリケーション
kdenetwork	メーラ、ニュースグループリーダのようなネットワーク関連のアプリケーション
kdeadmin	管理者用ツール
kdetoys	おもちゃやアクセサリ系アプリケーション
kdoc	さまざまなヘルプファイルを作成するのに必要なツール
koffice	オフィススイートアプリケーション
kdepim	スケジューラ関係アプリケーション
kde-i18n	各国語メッセージカタログおよび、ヘルプファイル。これとは別に各国語別にまとめられたものもある（kde-i18n-ja[日本語]）

表4　KDE2のパッケージ構成

パッチ	説明
qt-2.2.1-codec-20001008.diff	日本語や中国語の扱いを補正する
qt-2.2.1-xim-20001014.diff	文字入力の必要がないウィンドウでXIMを抑制する
qt-2.2.1-m17n-20001008.diff	ISO 8859-1をUnicodeに拡張することで多国語表示を可能にする。このパッチは上記2つのパッチを当てた上に当てる必要がある
qt-2.2.1-qpsprinter-20001008.diff	PostScript出力で日本語を出力する。このパッチは上記すべてのパッチを当てた上に当てる必要がある
qt-2.2.1-qclipboard-20001007.diff	ktermやXEmacsの間で日本語を含む文字のコピー&ペーストを実現する
kdelibs-2.0rc2-khtml-m17n-20001013.diff	Konqueror で日本語のHTMLを表示する
kdebase-2.0rc2-kwin-nofreeze-20001011.diff	起動時にフリーズする問題を解消する

表5　KDE2.0とQtのパッチ内容

▼リスト4　KDE2.0のmakeとインストール

```
# export KDEDIR=/usr/local/kde2
# tar Ixvf kdesupport-2.0rc2.tar.bz2
# tar Ixvf kdelibs-2.0rc2.tar.bz2
# tar Ixvf kdebase-2.0rc2.tar.bz2
# patch -p0 < kdelibs-2.0rc2-khtml-m17n-20001013.diff
# patch -p0 < kdebase-2.0rc2-kwin-nofreeze-20001011.diff
# cd kdesupport-2.0rc2
# ./configure --prefix=/usr/local/kde2
# make
# make install
# cd ../kdelibs-2.0rc2
# ./configure --prefix=/usr/local/kde2
# make
# make install
# cd ../kdebase-2.0rc2
# ./configure --prefix=/usr/local/kde2
# make
# make install
```

準備が整ったところで早速アップグレードに取りかかるとしよう。

KDE1.xとKDE2の共存

まず結論からいうと、同一のユーザーがKDE1.xとKDE2を動的に切り替えて使用することは難しい。KDE1.xとKDE2の2つをインストールして使用する最も簡単な方法としては、ユーザー自体を切り替える方法だ。

hogeというユーザーでKDE1.xを使用し、guhaというユーザーでKDE2を使用する。

だが、これではあまりにも不便なので、同一ユーザーで切り替えて使用する方法を説明しよう。

手前味噌な話で申し訳ないのだが、私が開発に参加しているKondara MNU/Linuxでは、ログインマネージャであるwdm（sdr）により、ユーザー、ウィンドウマネージャ、XIM、使用言語を切り替えて使用できるようになっている。このwdm（sdr）のウィンドウマネージャの項目でKDE2、または、KDE1.xを選択すればよい。また、先ほど環境設定ファイルなどを「Desktop1x」「.kde1x」、「.kderc1x」とリネームしておいたので、KDE2を起動すると新規にKDE2用の「Desktop」、「.kde」、「.kderc」が作成される。今度はこれらのファイルをそれぞれ「Desktop2」、「.kde2」、「.kderc2」とリネームする。

・KDE1.x用

```
Desktop1x, .kde1x, .kderc1x
```

・KDE2用

```
Desktop2, .kde2, .kderc2
```

そして、シンボリックリンクの張り方をを変えて環境を切り換えればいい。

・KDE1.xを起動する際

```
Desktop -> Desktop1x
.kde -> .kde1x
.kderc -> .kderc1x
```

・KDE2を起動する際

```
Desktop -> Desktop2
.kde -> .kde2
.kderc -> .kderc2
```

このようにすればとりあえずは共存という形はとれる。

現在、KDE1.xを使用している方は、ゆくゆくはKDE2へと移行されると思うので、少々不便だがKDE2が安定するまでのつなぎとしてこのようにして使っていただきたい。

なお、私が開発に参加しているKondara MNU/Linuxでは、このようなリンクの張り替えを行うスクリプトを作成して、wdmを使ってKDE1x（KDE2）起動時にそのスクリプトを実行する形にしようと思っていたが、後述のライセンスの問題からKDE1.x自体を配布するかは疑問になったので、スクリプトを作成するのを見合わせたという経緯がある。

メニューなどで日本語を表示する

KDE2のメニューなどに日本語を表示させるためには、あらかじめ日本語リソースパッケージであるkde-i18n-jaがインストールされている必要がある。もしまだならば、インストールしておこう。

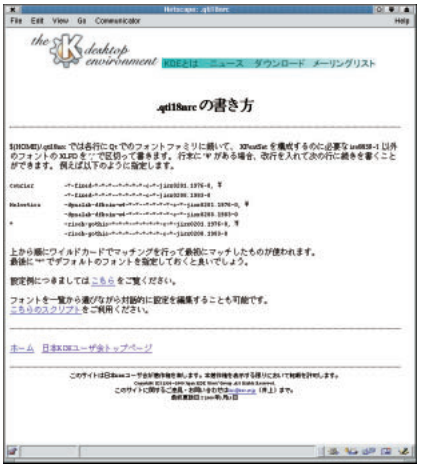
それでは、日本語の設定をしてみよう。コントロールパネルを起動し、左側のツリービューの [Personalization]-[Country & Language]を選択する。右側にLocaleタブが表示されるので、[Conuntry:]の部分で[日本(jp)]を選択する。すると次のように変更される。

```
Contory:  日本(jp)
Language: 日本語(ja)
Charset: iso8859-1
  (jisx0208.1983-0ではない)
```

選択に間違いがなければ、[OK]ボタンを押し、コントロールパネルを終了していったんログアウトする。次回、KDE2を起動した際は、アプリケーションスタータメニューおよび、アプリケーションのメニューなどで日本語が表示される。

メニューなどのフォント変更については、~/.qti18nrcにて設定ができる。設定方法など、詳細については.qti18nrcの書き方の解説サイトを参照してほしい。

なお、本文中ではKDE 2.0RC2について記述しているが、執筆後に2.0の正式版がリリースされた。残念ながら、正式版に関する詳しい記述を加筆することはできないが、基本的なインストール方法などは同じであるので、本文を参考にして2.0をインストールしてもいいだろう。また、本誌発売時点では日本語対応版もリリースされているかもしれないのでチェックしてほしい。



画面13　.qti18nrcの書き方解説サイト
(http://www.kde.gr.jp/~tino/qti18nrc.html)

魅せる、使う、実感するKDE ～KDEの利用 Tips～

さて、インストールが無事完了したならば、早速魅せるためのデスクトップ作りに取りかかってみよう。KDEのデスクトップをカスタマイズするにはいくつかの方法がある。ひとつはKDEコントロールセンターを利用し、各部分を個別にカスタマイズする方法だ。もうひとつは気に入ったテーマをテーママネージャで選択する方法だ。ここでは、KDEコントロールセンターを利用したデスクトップのカスタマイズ方法を紹介しよう。

KDEコントロールセンターの使い方

KDEコントロールセンターでは、デスクトップの背景画像、ウィンドウの色、アクション、イベントに割り当てる音の設定など、デスクトップの設定に関することならばほとんどすべてGUIで設定が行える。当然のことながら設定ファイルを操作してカスタマイズすることもできる。

パネルのKDEコントロールセンターのアイコンをクリックすると、KDEコントロールセンターが起動する。また、デスクトップ上やパネル上の何もないところで右クリックし［ディスプレイ設定］－［設定］をそれぞれ選択することで、各部分に関する設定画面だけを起動させることもできる。

起動したKDEコントロールセンターの左側には各項目がツリー表示され、画面右側に設定画面が表示され、そこで各項目がカスタマイズできるようになっている。デスクトップは背景やウィンドウ用にイメージ画像ファイルを割り当てるなど、簡単にカスタマイズできる。自分なりに工夫して個性的なデスクトップを作成してほしい（**画面14**）。

Konquerorの使い方

Kfmの時代からそうであったように、KDEのファイルマネージャはFTPサイトにログインする機能も持っている。KDE2.0のKonqueorの場合も同様であり、アノニマスFTPサイトにログインするには、ウィンドウのURLボックスにFTPサイトのURLを記述すればよい。では、アノニマスでない場合にはどうしたらいいのだろうか。

Konquerorを利用して、アノニマスFTPではないFTPサイトにログインするにはURLの項目に、ftp://username@ftp.somehost.comのように入力する。すると、Konquerorがパスワードを尋ねるプロンプトを表示するので、そこにパスワードを入力するとFTPサーバに接続を試みる（**画面15**）。

■Webブラウザの移行

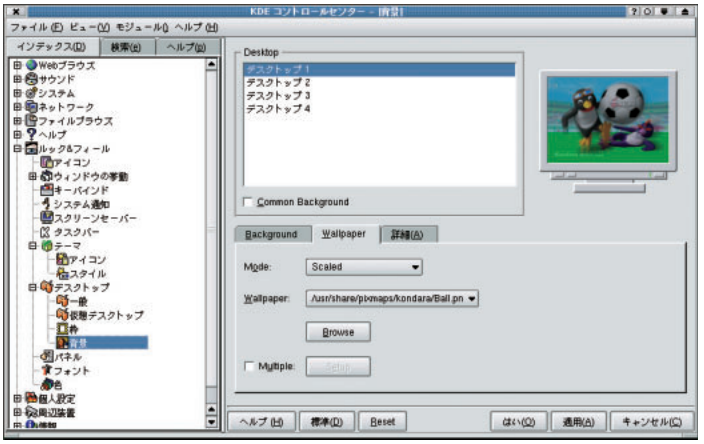
NetscapeからKonquerorに移行しようとする際、気になるのがBookmarkだろう。日頃チェックしているBookmarkメニューを一から設定するのはかなり根気のいる作業だろう。だが心配にはおよばない。Konquerorは、特別に指定しなくても自動的にNetscapeのBookmarksメニューが利用できるようになっている。

デスクトップにショートカットアイコンを配置

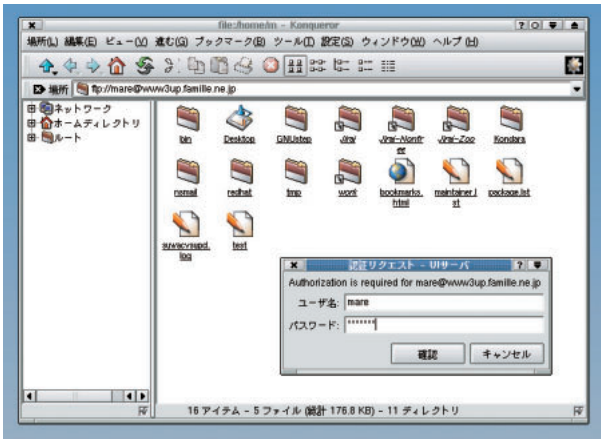
頻繁にアクセスするファイルはショートカットを作成すると便利だ。

まず、デスクトップ上で右クリックすると表示されるメニューから［新規作成］－［アプリケーション］または［ファイル］などを選択する。

一般タブにある「アプリケーション」と書いたテキストボックスは、ショー



画面14　KDEコントロールセンター



画面15　KonquerorによるFTPサーバへの接続

トカットアイコンの下に表示される文字となるので、わかりやすいショートカットの説明などを入力する。たとえば、Netscapeのショートカットアイコンとして登録したい場合、「ネットスケープ」とするとアイコンとともに「ネットスケープ」という文字が表示されることになる。

次に「実行」タブをクリックし、その画面のテキストボックスに、実行したいプログラムの名前を入力する。この場合であれば、「netscape」と入力する。パスが通っている場合にはこれでよいが、パスが通っていない場合にはフルパスを指定する。

アイコンを変更するには、一般タブの画面左側に表示されているアイコンをクリックする。あとは画面下の［はい］ボタンをクリックすれば完了だ。

パネルへのショートカット（Kpanelの場合はクイックランチという）を配置するのはもっと簡単で、KDEメニューに登録されているアプリケーションをパネルに登録するには、［アプリケーションスタータ］－［パネルメニュー］を選び、［追加］［アプリケーション］の順に選択すればよい。また、KDE2からはアプリケーションスタータのメニューに登録されていないアプリケーションの場合でも、KDEアプリケーションとほぼ同様の操作で登録できるようになった。

なお、デスクトップに登録したショートカットアイコンをパネルにドラッグ＆ドロップすることでもパネルへの登録が可能だ。

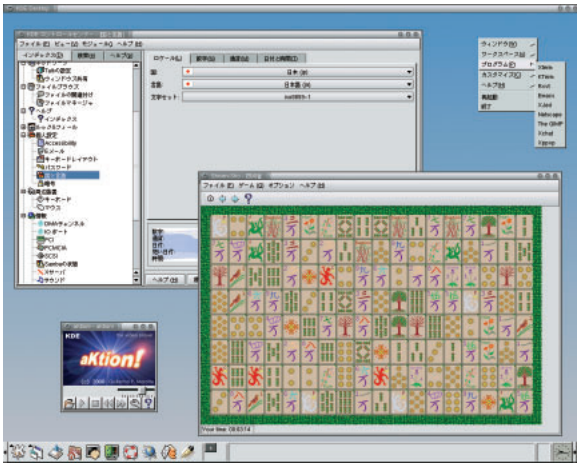
ちなみに、アプリケーションスタータのメニューを編集するにはメニューエディタを使用すると便利だ。従来よりもさらに使いやすくなったメニューエディタをぜひ実感してほしい。

お気に入りのウィンドウマネージャをKDEでも使用する

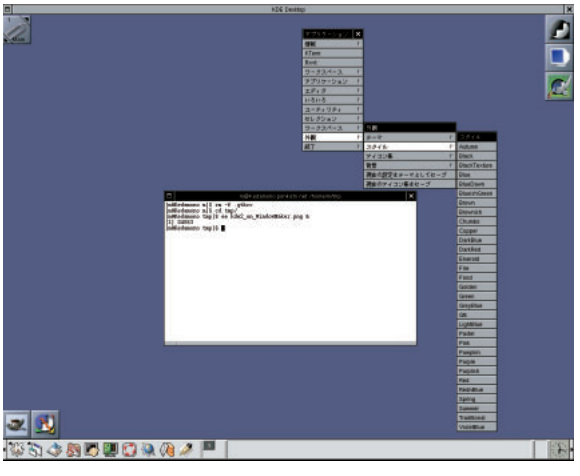
KDEの操作性や美観という点から考えれば、ほかのウィンドウマネージャをKDE上で動かすというのはマイナスとも思えるが、スペックの低いマシンにはKDEは重いという声も時々耳にする。

そんな時には、KDE2＋sawfishという組み合せはどうだろうか（**画面16**）。KDE上で、KwmまたはKWin以外のウィンドウマネージャを起動するには、startkdeスクリプトを編集する。

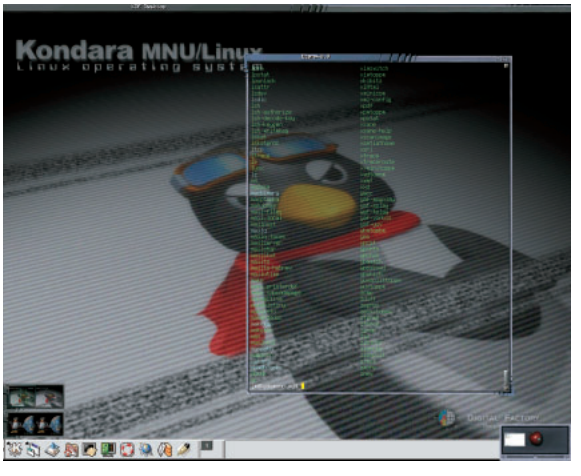
KDE1.xのstartkdeスクリプトでは、最終行にあるKwmコマンドを記述している行を、起動したいウィンドウマネージャのコマンドに変更すればよかったが、KDE2のstartkdeスクリプトでは、ksmserverを起動している行に、「--windowmanager ＜ウィンドウマネージャ名＞」というオプションを記述する。これを利用することによって、KDE2＋WindowMaker（**画面17**）、KDE2＋Enlightenment（**画面18**）なんていう起動も実現できる。



画面16　KDE2でsawfishを使う



画面17　KDE2でWindow Makerを使う



画面18　KDE2でEnlightenmentを使う

傲慢の極みより愛を込めて ～KDEライセンスの話～

@K of the North Star[wakashi]
wakashi@kondara.org

先にも少し触れたが、これまでQtのライセンスの問題上、KDEをハックするにはかなりの制限があった。これまでのQt1.xの場合、ライセンスはQPLであり、改変不可、個人的な使用はフリーだが、公の場への再配布は不可という、なんとも開発する立場の人間を困らせるには十分なものだったわけだ。正論から言えば、「Qt1.xおよび、それを使用しているKDE1.xは、ディストリビューションに含めてはならない」という解釈を述べる方も多く存在する。ライセンスの問題を取り上げれば、問題が出てくるLinux上のアプリケーションも少なくない。ライセンス完全保守論者の意見に従うとすれば、KDEはむろんのこと、NetscapeもpTeXもディストリビューションとして配布してはならないことになる。各ディストリビュータでも、これには頭を抱えたに違いない。なぜ頭を抱えるのか？　それはディストリビューターが、ユーザーのニーズを無視しない立場を取っているからだ。ディストリビューションの中には、明確に法律的権利を尊重し、「XXXはパッケージに含めません」という姿勢のものもあるが、すべてのディストリビューションが同じような姿勢を示したならばどうなるか？　ユーザーはそれぞれ、自分の使用しているディストリビューションで動くようにパッケージを作成することから始めなければならなくなる。「それがユーザーの本当のニーズなのです」というのならば、ディストリビュータ達は何も悩むことはない。現在、配布している「ライセンス的に配布可が明確でないパッケージ」を、すべて/dev/nullすればよいだけだ。ユーザーのニーズがそうでない場合はどうなのだろうか……。

今度は視点を変えて、KDEやQtの側から考えてみよう。上記のようなライセンス問題が持ちあがり、ユーザーからの反発を一斉に受け、極端にユーザーが減ることを想像していただきたい。はたして、Qtを開発したTrolltech、KDEを開発しているKDE開発チームは、どちらも何とも思わないものだろうか。ユーザーが減って彼らに何の得があろう。きっと、Trolltechもそう考えたに違いない。

2000年9月4日、TrolltechはついにQt2.0以降をGPL/QPLのデュアルライセンスにすると発表した。

これで、これまでライセンスの不明確さに振り回されてきた我々開発者も、心おきなく開発できるようになった。

さて、ここで少しだけKDEの枠を飛び出して、傲慢の極みと知りつつ、あえてライセンス問題がもたらす影響や教訓について考えてみようと思う。

皆さんは「めんちょ*1」をご存じか。そう、顔にできるあのおできだ。鼻の頭のめんちょ。こんなに厄介なものはない。いじればさらに大きくなって、下手すると命にかかわる。

だからといってほっとくわけにもいかない。はて、どうしたものか。めんちょを治す手立てを持たず、痛みも感じない他人は「めんちょは作ってはいけない」、「めんちょを作るのは恥ずかしい」と騒ぎたてる。あげくの果てに鼻の頭をつついては、「痛いかどうか述べてみろ」という。

「なぜ、めんちょの話などするのか？」と不思議に思われた方もいるかもしれない。この文章を読んで、あなたはどの登場人物だっただろうか？「めんちょ」ができた人間か？　それとも「めんちょ」をつついて痛いかと尋ねる人間か？　はたまた、「めんちょ」そのものか？

改変不可ライセンス付きアプリケーションの開発者、ディストリビューター、ユーザー、いつ何時、どの立場になるかわからないのが現在のLinux業界だ。「めんちょ」も然り。フリーで便利に使えればそれでよしとするユーザーの立場ならば、ライセンスなど無用の長物に過ぎないだろう。開発者の立場からいえば、自分が開発したプログラムなどの権利は守られるべきものであり、他人から無価にされるのは耐えがたいだろう。「めんちょ」の経験を持たない、今後も「めんちょ」の経験を持つこともないであろう人間が、めんちょを持つ人間に対し、あれやこれやと命令するのはおかしくないか？　意見するなとはいわない。ここは言論の自由が認められた日本だ。だが逆に、責任のない発言を聞かなければいけない義務はどこにもないし、意見を採用する自由も当然あるはずだ。

「めんちょ」をつついても痛いだけだ。だがその痛みは、つついた本人ではなく「めんちょ」を持つほうが痛いのだ。「めんちょ」が元で命を失えば、めんちょが元で亡くなった人間が、今まで果してきた役割に大きな穴が開き、多くの人に迷惑がかかるというものだ。

さて、頭の切れる読者の方々はきっとわかっていただけただろう。Linux開発者同士、潰しあって何になる。使いにくいディストリビューションを作って何の得がある。GPLができた経緯、ディストリビューションができた経緯を振り返ってみる時期にきたのではないだろうか。

＊1　編集部注
めんちょとは、面疔（めんちょう）の方言として一般的に使われている言葉で、顔にできる悪性のはれものを指す。

KDEのカスタマイズ

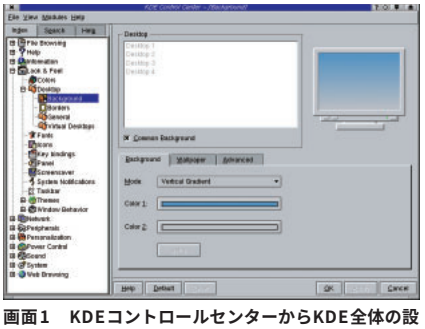
文：竹内充彦
Text: Michihiko Takeuchi

10月23日にリリースされたばかりの、KDE2.0のカスタマイズ方法を紹介する。正式な日本語版はまだないので、ここでは英語版でお届けする。最新デスクトップ環境の一端に触れて欲しい。

デスクトップ環境カスタマイズの基礎

KDEの基本的なカスタマイズは、KDEコントロールセンター（kcontorol）で行える（**画面1**）。KDEコントロールセンターは、初期設定で、画面下にあるパネルにランチャアイコンが用意されているし、パネルに用意されているメインメニューからも起動可能だ。

コントロールセンター内の基本操作は、①マウスで左側の階層から項目を選択する、②右側にそれに応じた設定項目が表示される、③設定を変更するという手順になる。変更した設定は［OK］（確認）ボタンをクリックすると反映され、右側の項目設定が閉じる。［Apply］（適用）ボタンをクリックすると、右側の項目設定を閉じることなく反映させることができる。［Default］（デフォルト）ボタンをクリックすると、いつでも初期設定に戻すことができる。



画面1　KDEコントロールセンターからKDE全体の設定が行える。環境のカスタマイズには欠かせないツールだ。左側の階層メニューから項目を選択し、右側で設定する。

パネルのカスタマイズ

まずはパネルのカスタマイズから紹介しよう。KDE1.1.2では、パネルとタスクバーが画面の上下に分かれていたが（**画面2**）、KDE2.0のデフォルトではそれらが統合され、パネル上にタスクが表示されるようになった（**画面3**）。

■ 表示位置と大きさの変更

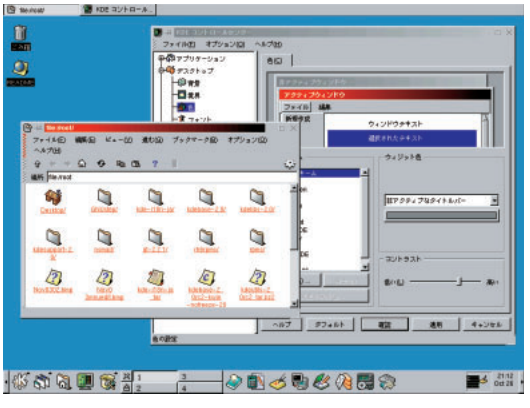
パネルは、画面の上下左右に配置可能だ。また、パネルは大きさを変更したり、自動的に隠すようにも設定できる（**画面4**）。

KDEコントロールセンターの階層メニューから［Look&Feel］－［Panel］を選択すると、パネルとタスクバー全般の設定ができる。設定可能な項目については、KDE1.1.2と同様だ。

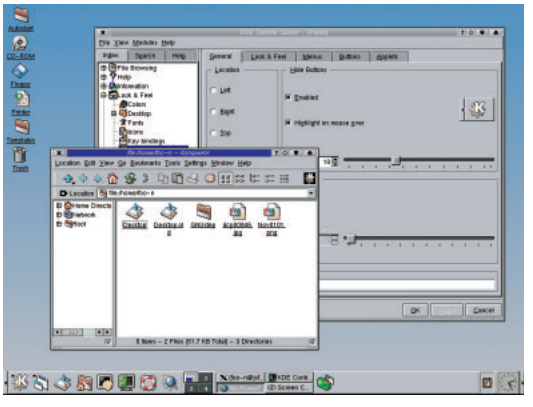
［General］タブでは、パネルの表示位置と大きさ、隠すかどうかを設定できる。［Location］でパネルの表示位置を、［Size］でパネルの大きさを、［Hide Buttons］でパネル両端の隠すボタンの見え方を、［Auto Hide］で自動的に隠すかどうかを設定できる。

■ パネルとアイコンの背景画像

KDEコントロールセンターの［Look&Feel］－［Panel］では、パネルとランチャアイコンへの背景画像の貼り付けも可能だ。［Look&Feel］タブの［Background Theme］でパネルの背景画像が、［Buttons］タブでランチャアイコンの背景画像がそれぞれ設定できる（**画面5**）。ランチャアイコンは、メインメニュー、アプリケーション、ウィンドウリスト、レガシーアプリケーションなど、種類ごとに背景画像を変えることができ、アイコンの視認性をよりいっそう高めてくれる（**画面6**）。



画面2　KDE1.1.2では画面下にパネル、画面上にタスクバーというように役割が分担されていた



画面3　KDE2.0では画面下のパネル上にタスクが表示されるようになり、タスクバーがなくなっている。

■ ランチャアイコンの登録

KDEでは、パネルにランチャアイコンを登録したり、いくつかのアプリケーションの状態を表示させることができる（ドック化）。アイコンの登録方法にはいくつか方法がある。簡単なのは、ファイルマネージャからのドラッグ&ドロップだ。メニューに登録されているアプリケーションならば、メニューの［Panel Menu］－［Add］から選択することで登録できる。パネルに登録したランチャアイコンは、マウスの中央ボタンで並べ替えられる。パネルから削除したければ、アイコン上でマウスを右クリックし、メニューから［Remove］を選択すればいい。

いくつかのアプリケーションは、パネル内のドックで実行される。CPUモニタやクリップボードなどがそれだ（**画面7**）。これらのドック化アプリケーションは設定が保存され、次回ログイン時にも自動的に起動する。もし、終了させたければ、ドックのアプリケーション上でマウスを右クリックし、表示されるメニューから終了できる。

■ メインメニューの変更

メニューの並びは操作性に影響する。メニュー項目は、追加／削除、並び順の変更が可能である。KDEのメインメニューの項目を並べ替える操作は非常に簡単で、メインメニュー上の項目をドラッグ&ドロップするだけである。これには専用のツールも必要ない。

メニューに新規に項目を作りたい場合や、すでにある項目を削除したい場合には、メニューエディタを使うことになる。KDE2.0ではこのメニューエディタのインターフェイスが大きく変わった（**画面8・画面9**）。むしろGNOMEのそれに近くなったと言っていい。メインメニューで［Panel Menu］－［Configure］－［Menu Editor］を選ぶとメニューエディタが起動する。

新規項目を追加するには、エディタ画面上部に並ぶ［New Item］ボタンをクリックする。サブメニューを追加したければ［New Submenu］ボタンをクリックする。クリックすると、項目名を問い合わせるダイアログボックスが表示されるので、入力し［OK］ボタンをクリックする。項目（またはサブメニュー）が階層メニューに追加されるので、画面右側の設定項目に、説明、命令、種類を入力すればいい。

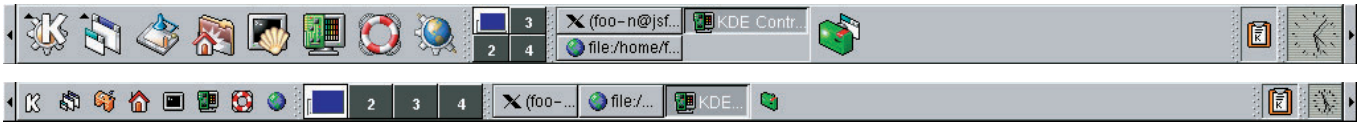
デスクトップのカスタマイズ

パネルが主に使い勝手を目的としたカスタマイズとすれば、デスクトップは主に見やすさや見た目の楽しさを目的としたカスタマイズだ。

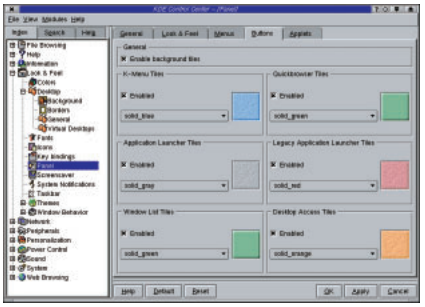
■ スクリーンセーバの変更

スクリーンセーバを変更するには、KDEコントロールセンターの階層メニューから［Look&Feel］－［Screensaver］を選ぶ。画面右側のリストからスクリーンセーバの種類を選び、そして解除時のパスワード要求の有無とプロセスの優先度を設定する。

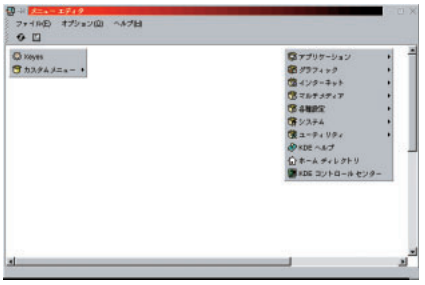
画面7　パネル上でドック化されたアプリケーション。画面の例ではクリップボードと、ティータイマーが稼動している。



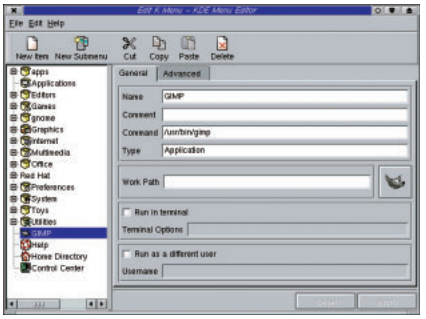
画面4　上が標準サイズのパネル、下が小さいパネル。Windowsなどに慣れているユーザーは小さいパネルのほうがシックリくるかもしれない。



画面5　アイコンの背景にビットマップ画像を貼り付けられる。アイコンの種類ごとに貼り付ける画像が変えられる。



画面8　KDE1.1.2のメニューエディタ。実際のメニューと同様に表示されるものだった。



画面9　KDE 2.0のメニューエディタは、GNOMEのそれに近くなった。

画面6　パネルとアイコンに背景画像を貼り付けたところ。アイコンの色でアプリケーションの種類がわかりやすい。

■ 背景色の変更と壁紙の貼り付け

デスクトップの色の変更と壁紙の貼り付けは、KDEコントロールセンターの［Look&Feel］－［Desktop］－［Background］で行える（**画面10**）。ここでの新機能はデスクトップの色を段階的に変えるグラデーションと、ビットマップ画像を貼りつける壁紙をブレンドさせることができるということだ。たとえば、［Background］タブでグラデーションをかける（**画面11**）。［Wallpaper］タブで壁紙にビットマップ画像を貼る（**画面12**）。それらを［Advanced］タブでブレンドする（**画面13**）。グラデーションの方法や、ブレンドの方法はいくつか用意されているので、これらを組み合わせることで、かなり多様なデスクトップが実現できるだろう。

■ デスクトップアイコンの登録

KDE2.0では、タスクの表示がパネルに移行したことで、デスクトップアイコンの活躍の場が増えるかもしれない。

デスクトップアイコンを登録するのは簡単だ。登録したいアイコンを、ファイルマネージャからデスクトップに、あるいはメニューからデスクトップにドラッグ＆ドロップすればいい。

この時、ドロップした瞬間に画面にメニューが表示され、コピー、移動、リンクが選択できる。基本的にリンクを選ぶべきだ。

デスクトップに置かれているアイコンの実体は、~/Desktop/というディレクトリの下に置かれる。アプリケーションを各自のホームディレクトリの下にコピーしてしまうと、無用なディスク消費を招いたり、アプリケーションのバージョンアップに対応できないなどの問題が発生してしまう。

デスクトップアイコンの削除は、そのままゴミ箱にドラッグ＆ドロップすればいい。

■ ウィンドウの外観の変更

KDEもウィンドウマネージャを差し替え可能だが、ほとんどの場合は一緒に配布されている専用のウィンドウマネージャを利用することになるだろう。ウィンドウマネージャは、KDE1.xではKwmだったが、KDE2.0ではKWinになった。このKWinの提供するウィンドウ周りのデザインもKDEコントロールセンターから変えることができる。KDEコントロールセンターの［Look&Feel］－［Colors］から行う。ここでは、タイトルバーの背景、文字、ボタンの文字、デスクトップアイコンの文字の色などが設定できる。

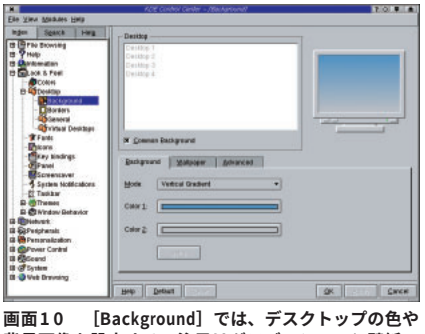
いくつか色の組み合わせが用意されているので、それを選んでもいいし、自分で組み合わせたものを登録することもできる。

■ テーマの変更

KDEでは、「テーマ」を導入することで、デスクトップ全体を文字通り、あるテーマにそったビジュアルに変更できる。テーマは通常、デスクトップアイコンや壁紙、ウィンドウの外観、ボタンやチェックボックスのデザインが組み合わされており、モジュールで提供される。このテーマの導入もKDEコントロールセンターから行える。［Look&Feel］－［Theme］を選び、［Browse］で目的のテーマを指定する（**画面14**）。

■ 仮想デスクトップの分割と動作

KWinは仮想デスクトップ機能を提供している。初期インストール状態では、仮想デスクトップは縦2画面分、横2画面分あり、4倍の大きさに設定されている。表示される部分は、パネル上で切り替え可能だ（**画面15**）。

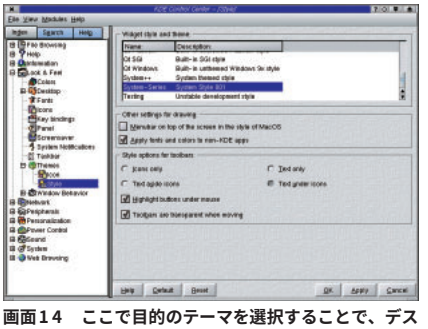


画面10　［Background］では、デスクトップの色や背景画像を設定する。注目はグラデーションと壁紙のブレンドだ。

画面11　色の設定でグラデーションを設定する。グラデーションパターンもいくつか選択可能。

画面12　壁紙に画像ファイルを貼る。中央、タイリング、画面の大きさに合わせるなど、こちらも貼り方が選べる。

画面13　グラデーションと壁紙をブレンド。ブレンドの度合いもスライダーボリュームで調節できる。多様なデスクトップがデザイン可能になるだろう。



画面14　ここで目的のテーマを選択することで、デスクトップをカスタマイズできる。

この仮想デスクトップの広さや、縦横の並びを変えることができる。KDEコントロールセンターの［Look&Feel］－［Desktop］－［Virtual Desktop］を選択し、スライダーで仮想画面の広さを設定できる。最大で16画面分の広さまで広げられる（**画面16**）。

画面15　パネルには、仮想デスクトップの状態が表示される。ここをクリックしてほかの仮想デスクトップを表示させることができる。

その他のカスタマイズ

デスクトップとパネル以外の部分のカスタマイズを紹介しておこう。

KDEのデスクトップやファイルマネージャでは、データファイルのアイコンをダブルクリックするだけで、そのファイルに応じたアプリケーションが起動するようになっている。たとえば、JPEG画像のファイルをクリックすると、ファイルマネージャ上でイメージビューアが起動する。これは、ファイルのデータの種類と、起動すべきアプリケーションが関連付けされているためである。

この関係付けを変更したり、新たな関連付けを追加することもできる。KDEコントロールセンターの［FileBrowsing］－［File Assosiations］を選択すればいい。右側にMIME型の階層、拡張子、起動されるアプリケーションなどが表示される（**画面17**）。

たとえば、JPEGファイルにGIMPを関連付けるには、まず、階層の「image」の下の「jpeg」という欄を選ぶ。拡張子に「.jpg、.jpeg、.jpe」が関連付けられている。この欄を選んでおいて、［Service Preference Order］の欄の［Add］ボタンをクリックすると、ダイアログボックスが表示されるので、そこに関連付けるアプリケーションを設定する。

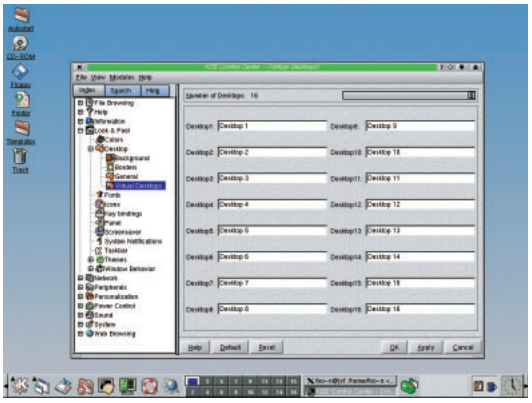
もし、すでにメインメニューに登録されているアプリケーションならば、リストから選択する。そうでなければ、テキストボックスに直接パスを入力するか、テキストボックス横のフォルダボタンをクリックしてファイルマネージャからファイルを指定することもできるようになっている。

ここでは直接パスを入力するので、「/usr/bin/gimp %f」と入力して［OK］ボタンをクリックする。リストボックスに登録されたGIMPを選択しておき、［Move Up］ボタンでリストの最上位に移動しておく。

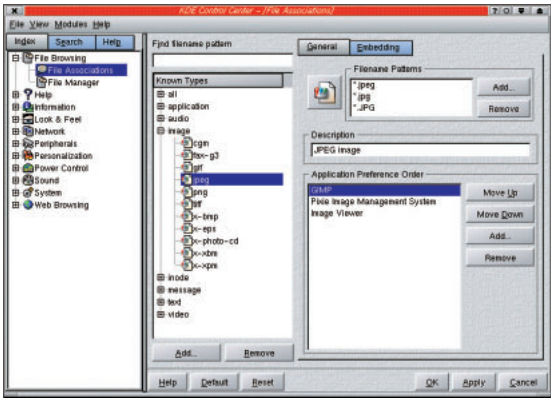
実はこのままでは、ファイルマネージャでJPEGファイルアイコンをクリックしてもGIMPは起動しない。KDEでは、1つの文書内に異なるデータ型が混在する、いわゆる複合文書（Composit Document）に対応するため、ファイルマネージャ内でシームレスにアプリケーションを切り替える仕組みが用意されている。そのため、KDEアプリケーションではないGIMPは初期設定のままでは起動しないのだ。そこで、さらに［Embedding］タブで［Show file in seperate viewer］オプションを選択する必要がある。こうしておけば、別アプリケーションとして起動するため、GIMPも起動するというわけだ。MP3プレーヤにKmp3を使わずにXMMSを利用したい場合などもこの設定が必要だ。

なお、ここで新たに登録したアプリケーションはメニューにも自動的に追加されている。メニューでの表示項目などは、前述のメニューエディタで整えておこう。

最後に、KDE1.xのときのように、パネルとタスクバーを画面の上下に分けて表示する方法を紹介する。方法は簡単だ。デスクトップを右クリックした際に表示されるメニューで表示／非表示を切り替えることができる。



画面16　仮想デスクトップの広さを16倍に広げたところ。パネル上の仮想デスクトップも16分割されている様子がわかる。



画面17　JPEGとGIMPを関連付けしてみたところ。さらに［Embedding］タブで［Show file in seperate viewer］オプションを選択する必要がある。

Linuxデスクトップギャラリー ～世界のテーマから～

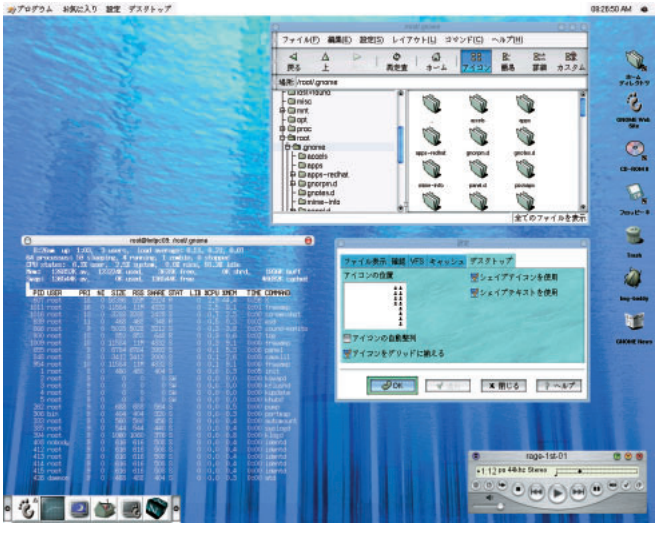
文：にゃー@編集部
Text: Nyaa@Linux magazine

本特集の締めくくりとして、Linuxデスクトップの「美」を追求していこうと思う。デスクトップ環境における本然的な美は、その機能に包含されているべきものであろう。ユーザーインターフェイス、あるいはアプリケーションフレームワークとしての整合性にこそ、求められる本質があるからだ。しかしながら、ここで問いたいのはインターフェイスの表層であるデスクトップ空間そのものが持つ美についてである。それはあくまでフェイクであって、フェイクであるからこそ、その美は純粋なのである。

そもそもの始まりは、Enlightenmentにあった。起動時の派手なギミック、豊富なカスタマイズ機能、テーマによるデスクトップの大胆な変容……。瞬く間に絶大なる支持を受けた「E」は、それまでデスクトップ空間に潜んでいた可能性に対してLinuxer達の目を開かせたのだ。それは大きな流れを生んだ。今やその潮流は、すべてのデスクトップへと到達している。そしてなおも加速しつづけているのだ。

では、デスクトップ空間の美の深遠へとご案内することにしよう。

GNOME



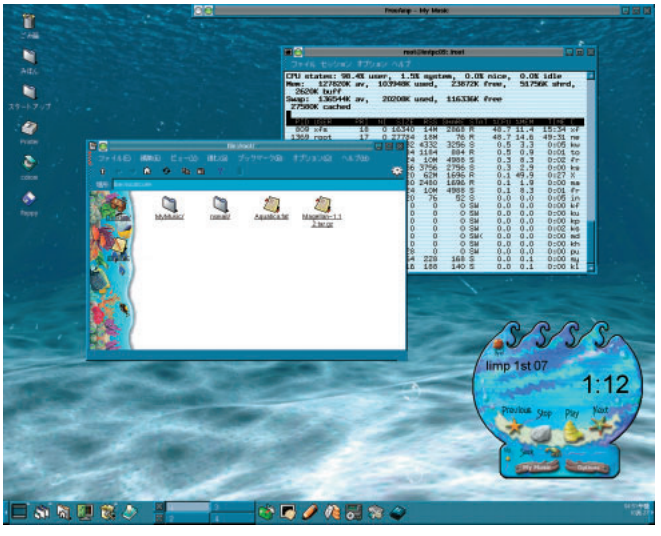
アイ・ライク・○ac OS X

ふー疲れた。何がって？　気取った文章書くのがですよ。さぁこれからは、通常文体でお送りしよう（Switch！）。

まず最初にご覧いただいているのが、GTK＋のテーマ「AquaX」とsawfishのテーマ「Aqua2」の合作によるM○c風。GNOMEの場合、このようにパネルやダイアログボックスの背景はGTK＋のテーマ、ウィンドウ枠はウィンドウマネージャのテーマという2つのテーマでデスクトップを美化できる。

GTK＋のテーマは、tar+gzip形式で提供されていて、インストールはGNOMEコントロールセンターの［デスクトップ］にある［テーマセレクタ］から行える。［新規テーマのインストール］ボタンをクリックしてtarボールを選択すればよい。これで［使用可能なテーマ］リストに新しいテーマが表示されるようになる。sawfishの場合、ホームディレクトリの.sawfishにthemesディレクトリを作成して、ここにダウンロードしたtarボールを展開すればよい。

KDE



海底散歩

前の作品に続いて水をモチーフにした作品。KDEのテーマ「magellan」を使った。イメージに合わせて、FreeAmpのテーマも「aqatica」に変更している。

KDE 2.0はリリースされたばかりなので、2.0専用のテーマはまだまだ数が少ない。http://kde.themes.org/でも、まだ3つしか登録されていなかった（それでもMac ○S X風のものがすでにあった。LinuxerはMacに憧れがあるのだろうか？）。ちなみに、このテーマは1.1.2用のものだ。1.1.2のテーマは、tar＋gzip形式で配布されている。KDEの場合、1.1.1と1.1.2で、テーマファイルの形式が違っていて、互換性がないことに注意しよう。これは設定を行うテーママネージャが変更されたためだ。新規テーマのインストールは、テーママネージャの［インストール］タブで［Add］ボタンをクリックしてファイルを選択するだけでいい。デスクトップがサムネイルっぽく表示されるので、気に入ったら適用しよう。

木星／ガニメデ／クリーチャー

GNOME

ガニメデは木星の第7の衛星。木星の衛星のなかで最大の大きさを誇る。そのガニメデをイメージしたEnlightenmentとGTK＋の同名のテーマが「Ganymede」だ。スペイシーでダークな感じに合わせて壁紙を変更し、FreeAmpのテーマには「Giger」を使った。FreeAmpのテーマは、拡張子が.fatのファイルで配布されている。新規インストールの方法を説明しておこう。まずFreeAPMのパネルでオプションボタンをクリックする。表示されるダイアログの［Themes］タブで［Add Themes］ボタンを押して、ファイルを選択すれば完了だ。

Enlightenmentのテーマも、独自形式（拡張子.etheme）で提供されている。新規インストールの方法は、sawfishの場合と似ている。まずホームディレクトリの.enlightenmentにthemesディレクトリを作成する。ただし、ここにファイルをコピーするだけでいい。これでデスクトップをマウスで中クリックして表示されるメニューで設定を行えるようになる。

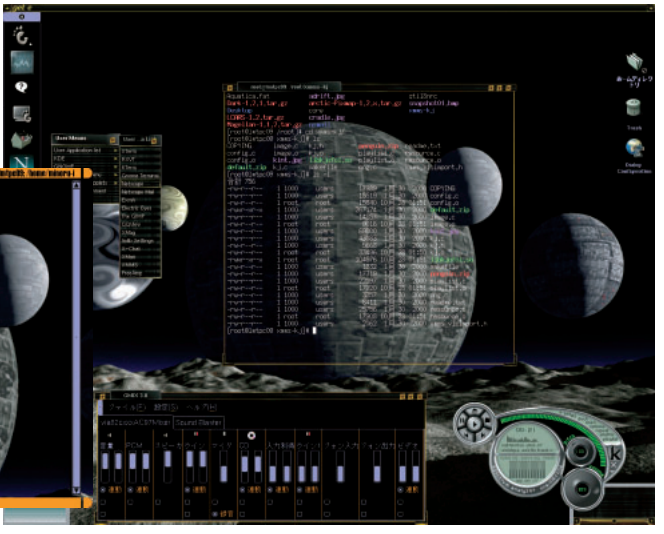
鬱だ……

KDE

「死」をイメージしたKDEのテーマ「Dead」。会社ではちょと使えないテーマである（プライベートでもじゅうぶん嫌だが）。パネルやウィンドウに血がしたたっていては、こちらの人間性まで疑われそうである。陰鬱な感じに合わせて、FreeAmpのテーマも追加している。こちらのテーマ名は「Doom」、やはり死とか破滅を意味する言葉だ。画面ではどこがMP3プレーヤなのかわかりづらいが、デスクトップの右上の楕円のパネルがそれである。

さて、気分を変えてここでテーマの入手方法を紹介しておこう。http://www.theme.org/には、KDE、GTK＋、Enlightenment、sawfish、IceWMなど、統合デスクトップ環境で利用できるさまざまなテーマが集められている。サーチ機能もあるので、名前やカテゴリーで検索可能だ。壁紙などは、テーマに付属しているものや、ネットワークで入手できるフリーの素材集を探すといい。GIMPを使えば自分好みに加工もできるぞ。

ふたたび宇宙へ



GNOME

GTK＋のテーマ「LCARS」とEnlightenmentのテーマ「get-e」。実はLCARSと同じ作者による同名のEnlightenmentのテーマもあったのだが、少し意地悪をしてほかのテーマと組み合わせてみた（申し訳ないので純正LCARSも載せておこう）。デスクトップの装飾ということになると、やはりEnlightenmentの独壇場といった感じだ。sawfishは機能がシンプルなぶん、凝った壁紙などで工夫が必要になる。また、よく「重い」といわれるEnlightenmentだが、ある程度のマシンパワーがあればそれほどストレスを感じることもないだろう。

IDE RAIDカードで作る

使えるパーソナルRAIDシステム

小規模なサーバ用途にマッチした、IDE RAIDカードに人気が集まっている。サーバ用途ということを考慮してか、Linux用のドライバを提供するベンダーも現れはじめた。そこで、これらのIDE RAIDカードとLinuxのソフトウェアRAID機能について検証した。

Text : Linux magazine Lab.
Photo : Shuichi Mito (Dee)

RAIDの基礎知識

一般的なユーザーがハードディスクを買う際には、容量、速度、そして価格を基準に、自分の用途に合った製品を選んでいくのが普通だ。よほどのマニアでないかぎり、まず予算があって、それに合わせて容量や速度で妥協をしていくものだ。

内蔵用の製品に限ると、ハードディスクの接続方式は、IDEとSCSIの2種類があるが、上記のような選択方法だとまず確実にIDEを買うことになるだろう。単位容量あたりの価格比が、IDEとSCSIで1対3〜4と大きな差があるからだ。たとえば約40 Gバイトの製品で比較すると、IDEが1万7000円で買えるのに対し、SCSIでは6万円前後の値段が付けられている（10月現在）。

この価格の開きは、SCSIのハードディスクが主に高性能ワークステーションやサーバなどのハイエンドマシンで利用されるのに対し、IDEハードディスクは普及価格帯の安価なPCを中心に利用されるために生じる。普及価格帯向けのパーツは、量産の効果でコストが下がるし、メーカーもシェア確保のために、1台あたりの利益を低く抑えて、その分台数を増やして儲けをあげようとするからだ。

ハードディスクの容量が今より小さかった昔は、この価格差がもっと開いており、（当時としては）大容量のハードディスクは、きわめて高価だった。そのため、より小さなハードディスクを組み合わせたほうが、安価に大容量を実現できたのだ。

しかしディスクの台数が多ければ、それだけ故障の確率も高くなるので、何の工夫もせずに、複数のハードディスクを用いれば、システム全体の危険度も高くなってしまう。

RAID

上述の危険度を下げるための工夫が、RAID（レイド・Redundant Array of Inexpensive Disks）である。文字どおり、比較的安価なディスクの組み合わせに、冗長性を持たせることで、全体として安全なシステムを構築することができる。“UCB”の略号で知られる、カリフォルニア大学バークレイ校のDavid A. Patterson氏のグループによって、1987年に提唱された。

RAIDは、冗長さの度合を目的によって選択することが可能で、さまざまな用途への適用ができる。そのためハイエンドシステムで広く利用されることとなった。

余談だが、RAIDの普及により、ハイエンドシステムの記憶装置を大容量のディスク1台で構成するということは、ほとんどなくなってしまった。そのため、今ではRAIDの“I”は、“Inexpensive”（安価な）ではなく、“Independent”（独立した）とされることが多くなっているようだ。しかしこの言い替えでは、かえって意味不明に近づいている気がする。

RAIDの種類

RAID装置を販売する各メーカーが、それぞれ独自の拡張を行ったため、同じ構成に別の名前が付いているなど、RAIDの名称は多少混乱している。最初に定義されたのは、RAID 0から5までの6段階である。各レベルの特徴は以下のとおりだ。

■RAID 0

ストライピングとも呼ばれる。データは、ブロック単位にRAIDを構成するディスクに分散して記録される（**図1**）。RAID 0は、冗長性がまったくないので、正確にはRAIDではないとも言える。

ディスクの読み書き速度は、単独のディスク性能×台数に近くなり、高性能が得られる。最近の高速なディスクを用いれば、PCIバスの帯域を越えるような構成も可能になる。また、エラー訂正用の仕組みを持たないため、装置の構成はシンプルである。その反面、冗長性がないので、1台でもRAID 0内のディスクが故障すれば、データ全体が失われることになり、信頼性は低い。

ビデオ編集など、広帯域が要求される仕事の作業ファイルを置くのに適した構成である。

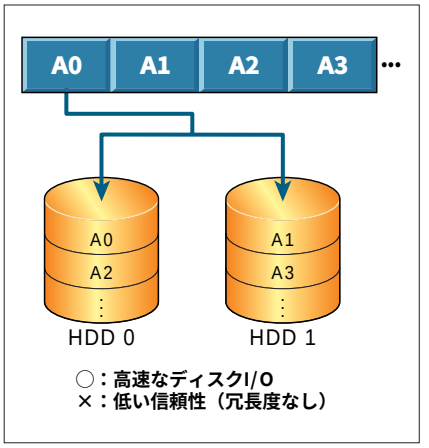


図1　RAID 0（ストライピング）

■RAID 1

2つのディスクに同一のデータを書き込んでいく方式で、鏡に写ったように同じ状態のディスクが2つできることから、ミラーリングとも呼ばれる（**図2**）。書き込み速度はディスク1台よりも少し遅くなる。また読み出し速度は、2台のディスクから交互に読み出すようにすることで、ストライピングと同等レベルにできる。だが実際の製品でそのように設計されているものは多くないようで、読み込みについても、ディスク1台とほぼ同レベルというのが一般的だ。

RAID 1は、常にディスクのスペアを作っているといってもよく、1台が故障しても、もう1台からデータが読み出せる。また、故障したディスクを新品に交換する際にも、正常なディスクからコピーするだけで済む。欠点は、ディスクの使用効率が全RAID中でもっとも低い（50%）ことだ。また読み書きともに、性能向上はあまり期待できない。

性能よりも、信頼性、可用性を重視する用途に適した構成だ。

■RAID 2

データをビット単位でストライピングし、複数のディスクに書き込み、同時にハミングコードと呼ばれるエラー訂正コード（ECC）を生成して、複数のディスクに書き込む方式だ。ワークステーションやサーバのメインメモリで用いられているECCを、ハードディスクで行っていると思えばいいだろう。

理論的には、RAID 0に近い読み出し速度が出せるうえに、1台のディスクが故障しても、ハミングコードを用いて、リアルタイムに正しいデータが再現できる。

しかし今の製品は、ハードディスクコントローラ自身にもエラーの検出／訂正機能があり、RAIDのレベルで再度エラー検出の仕組みを付加するのは、無駄になってしまう。また、ECCデータを書き込むディスクの性能が非常に高くなければならない。

このようにRAID 2は、現実の製品に適合しないシステムなので、現状ではRAID 2を用いた装置は販売されていない。

■RAID 3

データのブロックをRAID 0よりさらに小さい単位（たとえばセクタ）に分割し、複数のディスクに分散して書き込む。ここまではストライピングと同様だが、分散されたデータから、パリティと呼ばれるエラー検出コードを生成し、専用のパリティディスクに書き込んでいる（**図3**）。

データの格納方法は、ストライピングに近いので、読み出し速度はRAID 0なみに高速化できる。また、パリティディスクがあるので、ディスクが1台故障しても、残りのディスクからデータの復元が可能だ。ディスクの台数－1台（パリティ用）分が実効容量として利用できるので、RAIDを構成する台数が多ければ、ディスクの使用効率も相対的に上がる。

しかし、書き込み性能を最大限に上げるためには、データディスクの回転を同期させる必要がある。また、書き込み性能はパリティ用ディスクの能力に左右されることになる。

■RAID 4

図で書くと、RAID 3と区別が付かないのが、RAID 4だ（**図3**）。ストライピングの単位が、ブロック単位になっている点が異なる。RAID 3はデータを読み込む際に必ずすべてのデータディスクをアクセスする必要があるのに対し、RAID 4ではブロックが記録されたデータのみアクセスすればよい。このため、RAID 4ではディスクの同期は必要ない。ディスクが1台故障してもデータが復元できる点は、RAID 3と同様である。

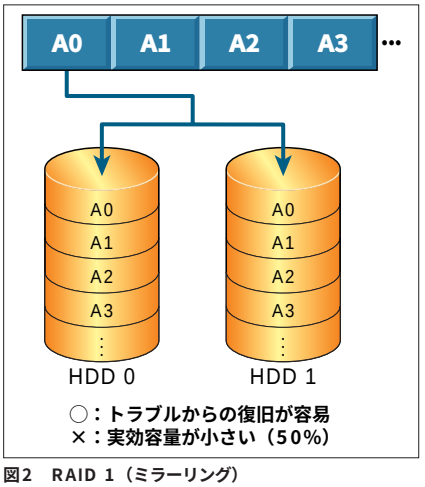


図2　RAID 1（ミラーリング）

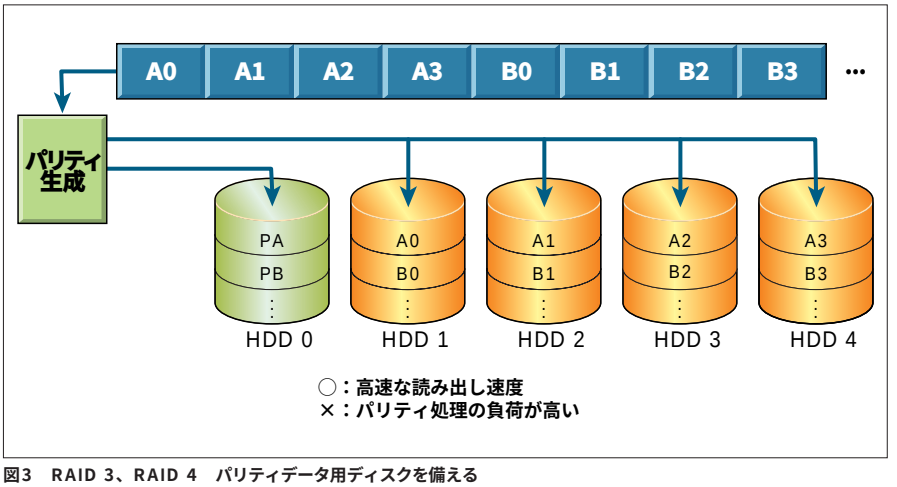


図3　RAID 3、RAID 4　パリティデータ用ディスクを備える

しかし、データ書き込み時は、ほかのディスクからもデータを読み出し、パリティデータを更新しなければならないので、RAID 3よりも処理に時間がかかる。

■RAID 5

RAID 4では、パリティデータを専用のディスクに書き込んでいたが、これを各ディスクに分散して書き込むようにしたのが、RAID 5だ（**図4**）。

データ読み出し速度は、ストライピングと同程度に高速化できる。書き込み時にすべてのディスクからデータを読み出し、パリティデータを更新する必要があるのは、RAID 4と同じだが、1台のパリティディスクにアクセスが集中することがないので、処理時間はRAID 4よりも短くなる。もちろん、ディスク1台が故障しても、データの復元が可能だ。

■そのほかのレベル

正式に規定されたRAIDレベルは以上だが、ほかにも各ベンダーが独自に（勝手に？）定義したRAIDが多数存在している。中でもRAID 0と1の組み合わせであるRAID 0＋1は比較的ポピュラーだが、これをRAID 10と呼ぶベンダーもあるので、注意が必要だ。

ソフト？ ハード？

RAIDシステムを実現するためには、各ディスクへのデータの割り振り、パリティデータの生成などを行う必要がある。この仕事をどこで行うかによって、ソフトウェアRAID／ハードウェアRAIDに大別される。

SCSIやIDEインターフェイスで接続したハードディスクを、ソフトウェア的に1つの記憶装置に見えるようにしたのが、ソフトウェアRAIDである。Linuxでも、カーネルに対するRAIDパッチが配布されており、ユーティリティのraid toolsと合わせて用いることで、ソフトウェアRAIDを利用できる。

Red Hat Linux 6.2などいくつかのRed Hat系のディストリビューションでは、標準でインストールされるカーネルにRAIDパッチが含まれており、複数のハードディスクを接続して設定すれば、RAIDシステムを構築できる。

このように、ソフトウェアRAIDは比較的安価にシステムが作れるが、RAID関係の処理をすべてCPUで行うため、特にディスクアクセスが多い環境ではプログラムの実行速度が低下する。また、多くのユーティリティやドキュメントが提供されているとはいえ、適切な設定で運用するためには、かなり経験を積む必要がある。

これに対してハードウェアRAIDは、拡張カードや外付けの記憶装置にRAID処理の機能を搭載したものだ。RAID処理は装置内のコントローラで行われるので、CPUへの負荷はそれほど高くない。

カード型のRAIDコントローラについては、いくつかのメーカーからLinux用のドライバが提供されており、適切に設定すれば、LinuxからハードウェアRAID環境が利用できる。

外付けのRAIDシステムは、SCSIインターフェイスを利用してPC本体と接続して用いるタイプがほとんどだ。このような製品では、PCからは単体のSCSIハードディスクのように見えるのが一般的で、OSに関係なく利用でき、インストールも通常のディスクと同じように行える。また、電源を切らずにディスクドライブを交換（ホットスワップ）できる製品もあり、運転を止められないシステムに用いられている。

変わるRAID

RAIDという概念が発表された'87年と今を比較すると、ハードディスクを取り巻く状況は、大きく変わっていると言えるだろう。たとえばRAID 1は、ディスクの効率が低く（50%）、性能も単体のディスクと同程度だ。しかし容量あたりの価格が非常に下がり、単体のディスク性能が向上した今では、これらの欠点は、以前よりも影響が小さくなっている。

また、今回紹介するような低価格のRAIDカードを用いれば、全体の導入費用も低く抑えられる。身近になったRAIDシステムで、効果的なパワーアップを図ってみよう。

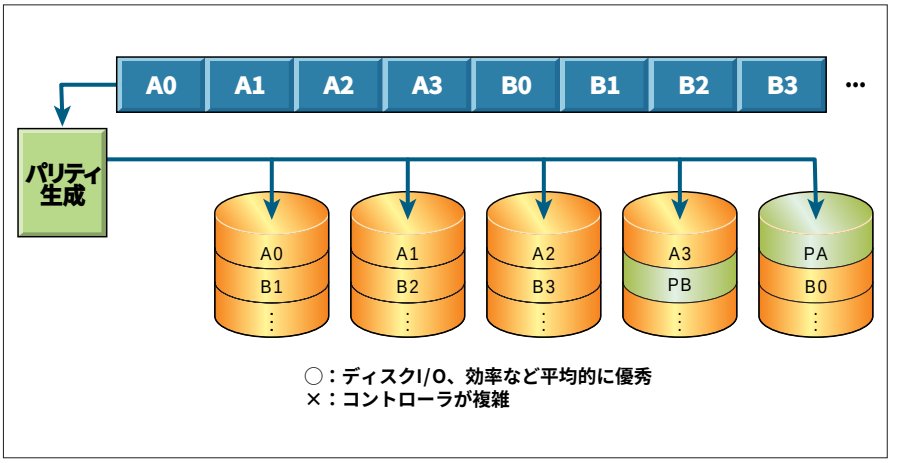


図4　RAID 5　パリティデータを分散記録

IDE RAIDカード3機種

かつてはRAIDといえばハイエンド用途に限られたものであり、値段が高く一般のユーザーとは無縁の存在であった。また必然的に接続方法はSCSIが主流であった。

しかしIDEディスクの低価格化と、UltraDMA/66、UltraDMA/100といった、転送速度の点ではSCSIに匹敵する上位規格の登場によって、状況が変化してきた。高性能化が進むIDEのディスクを利用できるRAIDカードが登場してきたのだ。これらの製品自体も、以前の物と比べれば価格が安いため、全体として低価格のRAIDシステムが構築できるようになった。今では、高性能を指向するPC DIY市場での人気製品と言えるだろう。

IDE RAIDカードは、ハイエンドの製品と比較して安価（実売価格1万円台の製品が多い）だが、機能的な差はあるのだろうか。

まずIDEとSCSIという方式が異なるため、接続できるディスクの台数が少ないことがあげられる。チャンネルあたり7～15台のディスクを接続できるSCSIの製品に比較して、IDE RAIDカードでは2つのコネクタを持ち、合計4台までのディスクを接続可能というのが一般的だ。このため、利用できるRAIDレベルは、RAID 0（ストライピング）、RAID 1（ミラーリング）および両者をあわせたRAID 0＋1（RAID 10と呼ぶベンダーもある）のみという製品が多いようだ。

また、より本質的な違いとして、多くのIDE RAIDカードが、実はソフトウェアRAIDだという点がある。RAIDボリュームの構築、ディスク交換後の再構築といった機能こそ、カード上のBIOSで提供されているが、ストライピング時の各ディスクへのデータの振り分けなどはCPUが行っているのだ。RAID 5をサポートしない製品が多いのは、パリティデータ生成をソフトウェアで行った場合、十分な性能が得られないためなのかもしれない。

IDE RAIDカードのLinuxでの利用を検討して、Linux magazineラボでは、次に紹介する3製品を購入し、試験を行った。代表的な2製品に加え、多少高価だが、個性的な製品を選択した。

3ware
Escalade 3W-6400

購入価格：3万6800円

http://www.3ware.com/

Escalade 3W-6400は、比較的新しい企業である3ware社が開発したIDE RAIDカードだ。カード上のキャッシュメモリとして、通常のDRAMよりも低レイテンシのSRAMを用いるなど、非常に個性的な作りになっている。

フルサイズPCIカードの上には、3個のコントローラチップと4つのIDEコネクタが備わっている。4チャンネル／4台までのディスクを接続してRAID 0、RAID 1、RAID 0＋1のシステムを構築できる。また、サポートしている規格は、UltraDMA/66までだ。

カード上に空きパターンがあることから分かるように、8チャンネルの上位機種3W-6800や、より短い基板を用いた、2チャンネルの3W-6200も存在する。

一般的なIDE RAIDカードと異なり、カード上にRAIDコントローラを搭載した、ハードウェアRAIDカードである。接続したディスクは、BIOSでRAIDボリュームを構築した後は、OSからは1つのディスクとして見えるようになる。

Windows 98／NT 4.0／2000に加えて、Linuxを正式にサポートしており、Red Hat Linux、SuSE Linuxに対応したドライバが付属している。また、管理用のユーティリティについても、Windows用だけでなく、Linux用が提供されている。このユーティリティは、障害発生のメールによる通知や、内蔵されたWebサーバによる、ネットワーク越しのRAIDコントロールが可能な優れもので、比較的小規模な業務使用にも足るレベルといえる。個人用途向けとしては、多少高価な製品だが、業務用のSCSI RAIDカードと比較すると、非常に優れたコストパフォーマンスだ。IDEとSCSIのディスクの価格差も含めると、本格的なRAIDシステムを低コストで構築できるだろう。

ABIT

Hot Rod 100 Pro

購入価格：8000円

http://www.abit.com.tw/

Hot Rod 100 Proは、マザーボードメーカーとして知られるABIT社の製品だ。同社は、デュアルCeleron（インテル社はCeleronのデュアル動作をサポートしていない）用としか思えないマザーボードや、BIOSから1MHzきざみでFSBを設定可能で、オーバークロック用途に適したマザーボードを出すことで知られている。マニアご用達のブランドといってもいいだろう。

同社のマザーボードとは違い、Hot Rod 100 Proは、IDE RAIDカードとしてはごく一般的なスペックの製品である。そのうえ、価格も1万円以下とたいへん安くなっている。

コントローラチップにHighPoint Technologies社のHPT370を用いており、最新のUltraDMA/100まですべてのIDE規格に対応している。2チャンネルのコネクタを持ち、合計4台までのディスクを接続可能で、RAID 0、RAID 1、RAID 0＋1のシステムを構築できる。HPT370は、ABIT社をはじめとしたマザーボードメーカー各社で採用されており、同チップを搭載して、「UltraDMA/100対応」をうたったマザーボードは数多い。

Windows 9x／NT 4.0／2000に加え、DOS 5.0以降に対応し、さらにABIT社独自のLinuxディストリビューションである、Gentus 2.0にも対応する。おそらくGentus 2.0には、Hot Rod 100 Pro用のドライバを組み込んであるのだろう。

残念ながら、本稿作成時にWebサイト（http://www.gentus.com/）はアクセス不能であり、Gentus 2.0の入手もできなかった。よって、今回の実験では、Hot Rod 100 ProのRAID機能は利用せず、ただの拡張IDEカードとして扱い、Linuxカーネルのソフトウェア RAID機能を利用した。

Promise Technology

FastTrak100

購入価格：1万5800円

http://www.promise.com/

FastTrak100は、Promise Technology社から発売されているIDE RAIDカードだ。このカードの前身であるFastTrak66は、IDE RAIDカードの先駆者的存在であった。まだUltraDMA/33が主流であったころに発売されたUltraDMA/66対応のFastTrak66は、高性能を指向するマニア層に受け入れられた。

FastTrak100は、コントローラチップとして、同社のPD20267を採用しており、UltraDMAを含む全IDE規格に対応している。同じチップを用いた姉妹製品として、RAID機能を持たないUltra100がある。FastTrak66にも同様にUltra66という姉妹製品があったが、FastTrak66とUltra66はまったく同じボードを用いており、Webサイト上で改造方法が公開されるなどして、物議をかもしていた。FastTrak100とUltra100で同様の改造ができるかどうかは、不明だ。

ボード上に2チャンネル分のコネクタを持ち、最大4台までのディスクを接続できる。RAID 0、RAID 1、RAID 0＋1のシステムを構築可能で、さらに3ドライブを用いたスペアディスク付きRAID 1といても利用できる。PDC20267を搭載した、UltraDMA/100 RAID対応マザーボードも、Gigabyteなどのメーカーから発売されている。

対応OSは、Windows 3.1／9x／NT 4.0／3.5／2000、DOSだ。Linuxについては、最近になって同社のFTPサイト上で、ベータ版ドライバのバイナリおよびソースが公開され、利用可能になった。RAIDシステムをベータ版で運用するのはリスキーだが、FastTrak100は、拡張IDEカードとして用いることができないため、Linuxでの利用には、ドライバが必須なのだ。幸いにも今回のテスト中には、問題は起きなかった。

LinuxでRAIDシステムを作る

LinuxでRAIDシステムを構築する方法は2つに大別できる。RAID機器の導入というハードウェアによるソリューションと、Linuxのソフトウェア RAID機能を利用するソリューションだ。

■IDE RAIDカードによるRAID

RAID機器にはさまざまなものがあるが、最近ではIDEハードディスクを接続してRAID 0（ストライピング）、RAID 1（ミラーリング）を実現する低価格RAIDカードがいくつか発売されている。IDEハードディスクは、大容量化と低価格化が著しく進みつつあり、これらのカードを利用すれば個人でも手軽にRAIDシステムを手に入れられる。

しかし、現状ではLinuxで利用できるIDE RAIDカードは多くない。ほとんどの機種では、Linux用のデバイスドライバが提供されていないためだ。ディスクアレイ装置では、それ自体が1つのディスクとして認識されるため、特別なデバイスドライバは必要ないものもあるが、カードの場合はデバイスドライバがないとOSから認識できない。また、高速な転送モードに対応するIDEディスクコントローラチップと、RAID支援機能を持つBIOSを組み合わせたタイプのIDE RAIDカードでは、RAID機能の多くをデバイスドライバによるソフトウェア処理にまかせていることが多いようだ（一種のソフトウェアRAIDと言ってもよいかもしれない）。

写真1　80芯と40芯のIDEケーブル
UltraDMA/66、100モードを利用するには耐ノイズ性能が高い80芯ケーブルが必要だ（写真左）。

■ソフトウェアによるRAID

安定版のLinuxカーネル2.2には、ソフトウェアRAID機能が不完全ながら盛り込まれている。実際にこの機能を使うためには、カーネルのソースにパッチをあてて再構築する必要があるのだが、多くのディストリビューションでは、RAIDパッチを適用済みのカーネルを採用している。

LinuxのソフトウェアRAID機能は、複数のハードディスクをデバイスドライバで制御することで実現されている。そのため、Linuxでふつうに利用できるハードディスクであれば、IDE／SCSIなど接続インターフェイスの種類も問わずに利用可能だ。もちろん、標準のハードディスクコントローラだけでもRAIDを構成できるので、コストパフォーマンスは抜群だ。さらに、3台以上のディスクを用意すれば、RAID 5環境を作ることもできる。

ただし、すべての処理をCPUが行うため、ハードウェアでRAIDの処理を行うのに比べるとシステムへの負荷が高くなる可能性が大きい。また、RAIDカードではBIOSで各種設定を行えるのに対し、すべての設定をコマンドで行わなければならない。

■RAID構築に必要なもの

RAID環境を実現するためには、アレイを構築するために複数のハードディスクとケーブルが必要だ（**写真1**）。それ以外には何が必要なのかをチェックしてみよう。

●IDE RAIDカードを使う場合

- **IDE RAIDカード**
- **デバイスドライバ**

●ソフトウェアRAIDを利用する場合

- **RAIDサポート機能を持つカーネル**
- **ディスクをつなぐ空きポート**

IDEでは、1ポートにマスタ、スレーブ2つのディスクをつなぐことが可能だが、パフォーマンスを考慮するなら1ポートに1台のディスクだけを接続することが望ましい。1つのポートに高速なハードディスクを2台接続すると、バスが飽和してディスクの性能を活かしきれない可能性があるからだ。

現在販売されているIDEハードディスクの多くはUltraDMA/100やUltraDMA/66といった高速な転送モードに対応している。標準のディスクコントローラがこれらのモードをサポートしていないなら、高速な転送モードに対応する拡張IDEカードを増設してもよいだろう。ただし、拡張IDEカードを使うには、カーネルにパッチをあてて再構築する必要がある。

次ページから各IDE RAIDカードごとの設定方法を解説する。Hot Rod 100 Proは、デバイスドライバがないためソフトウェアRAIDで使用した。

Escalade 3W-6400の設定

Escalade 3W-6400は、Linuxを正式にサポートする数少ないIDE RAIDカードだ。パッケージにはRed Hat Linux 6.1／6.2、SUSE 6.3／6.4に対応するドライバディスクが付属する。ドライバのソースも収録されているが、2.2.17カーネルではすでに標準デバイスとなっている（モジュールバイナリを用意しているディストリビューションは少ないかもしれない）。

以下、Red Hat Linux 6.2J 改訂版を例に、設定方法を説明する。

■RAIDアレイの作成

初めにRAIDアレイを作る必要がある。カードとディスクを装着してから、電源を入れ、Alt＋3キーを押してBIOS設定画面を呼び出す。ここでRAIDアレイを構成するディスクや、ストライピング／ミラーリングなどの選択を行えるほか、アレイのメンテナンスを行うことも可能だ。

■既存のシステムにRAIDを追加する

稼働しているLinuxマシンに、データ領域としてRAIDディスクを追加する場合は次のようにする。

Linuxを起動してEscalade 3W-6400のドライバディスクをマウントし、モジュールをハードディスクにコピーする（**画面1**）。2.2.14-5.0というディレクトリ名は現在使用しているカーネルのリリース番号である。ディストリビューションなどによって異なるので、uname －rを実行して表示されるリリース番号に読み替えてほしい。

続いて、起動時にモジュールが組み込まれるように設定する。/etc/rc.d/rc.sysinitファイルをエディタで開き、swaponコマンドを実行している部分を検索してそのあとにモジュール組み込みの設定を追加する（**リスト1**）。

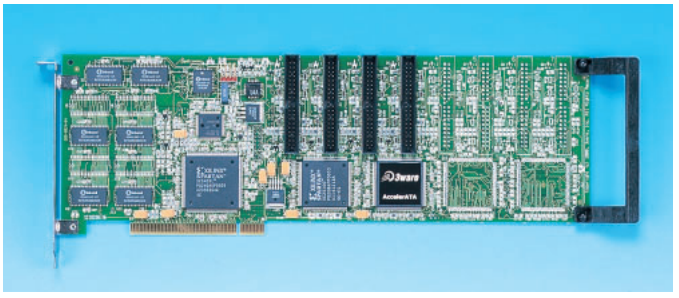


写真2　Escalade 3W-6400
4ポートのIDEコネクタとオンボードキャッシュメモリを装備する。

```
# mount -t msdos /dev/fd0 /mnt/floppy
# install -m 644 /mnt/floppy/3w-xxxx.o /lib/modules/2.2.14-5.0/scsi/
```

画面1　ドライバモジュールをハードディスクにコピーする。

▼リスト1　/etc/rc.d/rc.sysinitファイルにinsmod行を追加する

```
action "Activating swap partitions" swapon -a
/sbin/insmod /lib/modules/2.2.14-5.0/scsi/3w-xxxx.o
```

この行のあとに
この行を追加する

設定が済んだらLinuxを再起動する。起動途中でKudzuデーモンが新しい機器を発見するので、Configureボタンを押す。

モジュールを設定する代わりに、ドライバを組み込んだカーネルを再構築してもよいだろう。

RAIDアレイは、1つのSCSIドライブとして見える。ほかに（本物の）SCSIドライブがない場合は、/dev/sdaというデバイスがRAIDアレイだ。

あとは通常のハードディスクと同様にfdiskでパーティションを切り、mke2fsなどでフォーマットしたうえで、適当なマウントポイントにマウントすれば利用可能になる。

■RAIDにLinuxをインストール

Red Hat系のディストリビューションでは、インストーラを起動する際、「boot:」プロンプトに対し、「expert」と入力する。ドライバディスクを求められたら付属のディスクを入れる。これで、デバイスを認識するのでそのままインストールするだけだ。

写真3　コントローラチップ
XILINX社製のFPGAをコントローラとして採用。基板上の部品点数を減らすとともに、チップ内の論理回路をソフトウェアで変更できる。

Promise FastTrak100の設定

FastTrak100はUltraDMA/100モードに対応したIDE RAIDカードだ。このカードの前身にあたるFastTrak66は、IDE RAIDカードブームの火付け役となった。Linux magazineでも2000年5月号のRAID特集で取り上げたのだが、このときはデバイスドライバがなかったためにLinuxで利用できなかった。

待つことおよそ半年。2000年9月にFastTrak100、FastTrak66のベータ版Linuxドライバがベンダーより配布されるようになった（**表1**）。さらに喜ばしいことに、10月に入ってソースコードも提供された。

RAIDカードでベータ版ドライバを利用するのは非常にリスクが高い行為だ。試用する場合は自己責任でお願いしたい。

■RAIDアレイの作成

Escalade 3W-6400の場合と同様、最初にRAIDアレイを作成する。FastTrak100では、マシンの起動時にCtrl＋Fキーを押すとBIOSの設定画面を呼び出すことができる。

ストライピング／ミラーリングなどの種別設定のほか、ディスクアレイのメンテナンスができるのもEscalade 3W-6400と同じだ。

■既存のシステムにRAIDを追加する

Linuxのシステムにデータディスクとして RAIDを追加するなら、Escalade 3W-6400と同様の作業手順となる。**表1**のサイトからドライバモジュールのft.oを入手し（FTPサイトには、なぜかFT.Oという大文字のファイル名で置かれているので、ft.oにリネームしておこう）、/lib/modules/2.2.14-5.0/scsiにインストールする。“2.2.14-5.0”の部分をuname -rで表示されるリリース番号に変更するのを忘れずに。

FastTrak100でもRAIDアレイはSCSIディスクとして認識されるようになる。これは、現在のLinuxカーネルがIDEのデバイスドライバをモジュールとして扱うことができないためだそうだ。IDE RAIDカードをSCSIデバイスとして扱うためには、カーネルがSCSIをサポートするように構築されていなければならない。たいていのディストリビューションでは、SCSIをサポートするカーネルがインストールされるので問題ないが、自分でカーネルを再構築しているなら注意が必要だ。

■RAIDにインストールできるか

FastTrak100には、Linuxのインストーラにデバイスドライバを認識させるドライバディスクは用意されていない。そのため、RAIDディスクにLinuxをインストールするのは少々面倒だ。

ドライバディスク、あるいはドライバを含めたインストールディスクを作成するという方法もあるが、ディストリビューションごとにその形式もまちまちなので簡単ではない。

ではどうすればよいのか？　FastTrak100のBIOSには、指定したディスクの内容をほかのディスクにコピーしてミラーアレイを作成する機能がある。この機能を使えば、インストール済みのLinuxシステムディスクをミラーで

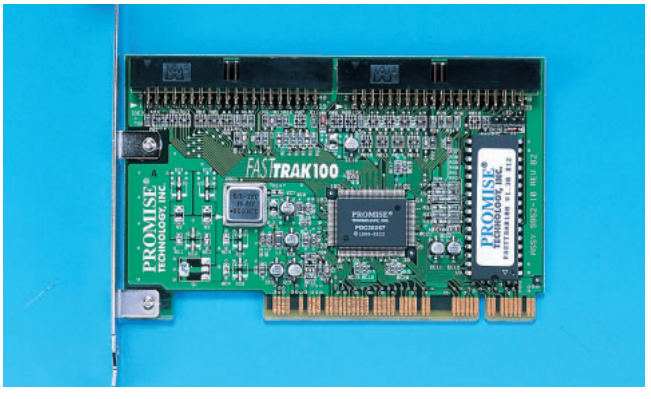


写真4　Promise FastTrak 100
2ポートのIDEコネクタを装備。メジャーな製品なのでたいていのショップで入手できる。

製品	URL
FastTrak100	ftp://ftp.promise.com/Controllers/IDE/FastTrak100/LinuxBETA/
FastTrak66	ftp://ftp.promise.com/Controllers/IDE/FastTrak66/LinuxBETA/

表1　FastTrak100、FastTrak66のベータ版ドライバのURL

写真5　IDEコントローラPromise PDC20267
姉妹製品の拡張IDEカードUltra100でも使われているチップだ。

きる。残念ながら、ストライピングはできないので、システムをストライプさせたいならインストールディスクかドライバディスクを作る必要がある。

■ミラーリングの元ディスクを作る

まず、1台のハードディスクをマザーボードの標準IDEポートに接続し、Linuxをインストールする。インストールが済んだら、FastTrak100につなぎ替えてミラーの元ディスクとするのだ。ここで1つ注意してほしいのは、fdiskなどでパーティションを作成するときに、ハードディスクの最後の2シリンダには、いかなるパーティションも割り当てないことだ。FastTrak100は、この2シリンダにRAIDアレイの管理情報を書き込む。そのため、この部分がデータで上書きされるとミラーアレイが破壊されてしまうのだ。

また、ブートディスクは必ず作成しておこう。このあとの作業に失敗して、Linuxが起動しなくなった場合、復旧作業をするために必要となる。

Linuxをインストールしたら、デバイスドライバのft.oを/lib/modules/2.2.14-5.0/scsiディレクトリにインストールする。

次に、FastTrak100のドライバを含んだRAMディスクイメージを作成する。ドライバはモジュールとして提供されているため、カーネルが起動してから組み込まれることになる。しかし、デバイスドライバを組み込まないとマウントできないパーティションにドライバを置いたのでは缶詰の中にある缶切り状態となってしまう。そこで、ドライバの入ったRAMディスクイメージを作っておき、起動時に展開されたRAMディスクからドライバを読み出すのだ。**画面2**のようにすればイメージファイルが作成される。

■LILOの設定

Linuxをインストールしたのが1番目のIDEドライブなら、それは/dev/hdaというデバイスだ。/etc/lilo.confを見ると、bootや、rootというキーワードのあとに/dev/hdaという文字列が書かれているはずだ。これは、LILOを書き込む場所や、/（ルート）パーティションを指定している。

FastTrak100は、LinuxからはSCSIドライブに見えるために、このハードディスクをFastTrak100につなぎ変えると/dev/hdaは/dev/sdaとして認識される（ほかにSCSIディスクがない場合）。そこで、/etc/lilo.conf を**リスト2**のように書き換える必要がある。

書き換えが済んだら、/sbin/liloを実行して設定を反映させる。

■fstabの設定

最後に、/etc/fstabを書き換える。このファイルは、システムの起動時に、どのパーティションをどこにマウントするかを指示するファイルだ。LILOの設定と同様に、hdaをsdaに書き換える。

■ミラーを作成

すべての設定を済ませたら、電源を切り、Linuxをインストールしたディスクを FastTrak100につなぎ替える。FastTrak100のBIOSでミラーアレイを作るときに「Create and Duplicate」を選び、「Source Disk」にLinuxをインストールしたハードディスクを指定すれば、ディスク内容が複製され、ミラーアレイが作られる。

```
# mkinitrd --preload ft /boot/initrd-ft.img 2.2.14-5.0
```

画面2　RAMディスクイメージファイルを作る

▼リスト2　/etc/lilo.confを書き換える

```
boot=/dev/hda   # 現在LILOが書き込まれているところ（ここはそのまま）
disk=/dev/sda   # FastTrakにつないだときに見えるデバイス名（追加）
bios=0x80       # BIOSドライブ番号。最初のディスクは0x80（追加）
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
linear
default=linux

image=/boot/vmlinuz-2.2.14-5.0
        label=linux
        initrd=/boot/initrd-ft.img      # 作成したRAMディスクイメージ
        read-only
        root=/dev/sda1  # /（ルート）パーティションを指定（hda→sdaに）
```

▼リスト3　/etc/fstabの例（“hda”を“sda”に書き換える）

```
/dev/sda1   /            ext2    defaults        1 1
/dev/sda5   /home        ext2    defaults        1 1
/dev/sda6   swap         swap    defaults        0 0
/dev/cdrom  /mnt/cdrom   iso9660 noauto,owner,ro 0 0
/dev/fd0    /mnt/floppy  ext2    noauto,owner    0 0
none        /proc        proc    defaults        0 0
none        /dev/pts     devpts  gid=5,mode=620  0 0
```

Hot Rod 100 Proを使ったソフトウェアRAIDの設定

Hot Rod 100 Proは1万円を切るLowプライスのIDE RAIDカードだ。しかし、残念ながらLinux用のRAIDドライバがないため、RAIDカードとして利用することはできない。

ところが、Hot Rod 100 Proが搭載しているIDEコントローラHPT370は、カーネルにIDEパッチをあてることでサポートされる。このパッチをあてたカーネルで試したところ、UltraDMA/100モードに対応する拡張IDEカード（RAID支援機能は使えない）として利用することができた。そこで、ここではRAIDカードとしてではなく、高速な転送モードが利用できる拡張IDEカードとしてHot Rod 100 Proを使い、Linuxのソフトウェア RAID機能を使ってみよう。

■カーネルにパッチをあてる

前述のとおり、カーネル2.2でソフトウェアRAID機能を使うには、RAIDサポートパッチを適用しなければならない。ソフトウェアRAIDを利用できるカーネルをインストールするディストリビューションも多いが、オリジナルのカーネルソースをコンパイルする場合は、自分でパッチをあててカーネルを作る必要がある。

また、ここで取り上げるHot Rod 100 Proのほか、Promise Ultra100／66など、最近のIDEコントローラチップを搭載した拡張IDEカードを使う場合は、RAIDパッチのほかにIDEサポートパッチも必要になる。

カーネル2.2.17に適用可能なパッチファイルを付録CD-ROM Disc 3に収録しているので、必要に応じて利用してほしい。また、最新版は**表2**に示すURLから入手できる。カレントディレクトリをカーネルのソースディレクトリに移し、**画面3**のようにすればパッチあてが完了する。

パッチを適用したら、カーネルのコンフィグレーションを行う。以下、make xconfigを例に説明する。RAID機能を使うためには、「Block devices」の「Multiple devices driver support」、「Autodetect RAID partitions」を「y」にする。そして「RAID-0（striping）mode」など、必要なRAIDモードのドライバを「y」または「m」とする（**画面4**）。

また、利用するIDEコントローラチップへのサポートも「y」にしよう。Hot Rod 100 Proには、HPT370が搭載されている。このチップは、「HTP366 chipset support」を「y」にすることで利用可能だ。

あとは通常どおりカーネルを構築すればよい。

■RAIDデバイスの作成

新しいカーネルで起動すると、RAIDのために用意した2台のハードディスクはそれぞれ/dev/hde、/dev/hdgとして認識された。fdiskを使って両方のディスクに同じ容量のパーティションを用意する。今回は、/dev/hde1、/dev/hdg1を作った。

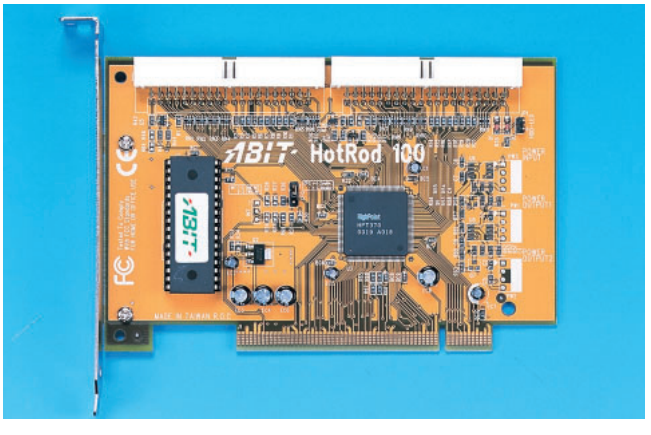


写真6　Hot Rod 100 Pro
低価格が魅力だが、Linux用のRAIDドライバはない。

RAIDサポートパッチ
http://people.redhat.com/mingo/raid-patches/
IDEサポートパッチ
ftp://ftp.jp.kernel.org/pub/linux/kernel/people/hedrick/

表2　パッチファイルの置かれているURL

写真7　IDEコントローラHighPoint HPT370
HPT366／368の後継チップ。UltraDMA/100モードに対応する。

RAIDアレイの設定は/etc/raidtabというファイルで行う。ストライピングなら**リスト4**のように、ミラーリングなら**リスト5**のようにすればよい。Red Hat系のディストリビューションなら、/usr/doc/raidtools-0.90/以下にサンプルファイルが用意されているので参考になるだろう。

raidtabを作成したら、raidtoolsというパッケージに含まれる、mkraidコマンドでRAIDデバイスを作成する。

```
# mkraid /dev/md0
```

/dev/md0は、raidtabのraiddevで指定したデバイス名である。作成したRAIDアレイは/dev/md0というデバイスとして認識され、ふつうのパーティションと同じように扱えるのだ。mke2fsなどでフォーマットすればマウントして利用できるようになる。

```
# mke2fs /dev/md0
# mkdir /data
# mount /dev/md0 /data
```

このようにすれば、/dataにRAIDデバイスがマウントされる。raidtabには、/dev/md0、/dev/md1、……と複数のRAIDデバイスを定義しておくことができる。また、RAIDデバイスの状態は、/proc/mdstatの内容を見ることで調べられる。たとえば、**画面5**を見ると、2台のハードディスク/dev/hdeと/dev/hdgでミラー(RAID 1)のデバイスが3つ作られていることがわかる。

■RAIDへのLinuxインストール

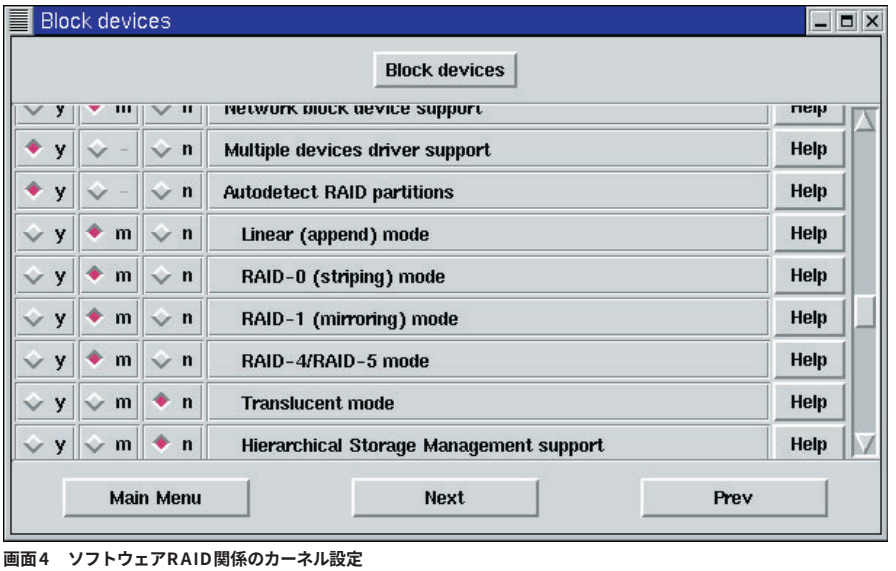
Red Hat Linux 6.xや、TurboLinux Workstation 日本語版6.0 LEアップデート版では、ソフトウェアRAIDに対応したインストーラを提供している。

ただし、これらのインストーラが認識できない拡張IDEカードでRAIDを構築する場合は、そのままではRAIDへのインストールができない。また、古いバージョンのLILOは、/dev/mdデバイスを扱えないので注意が必要だ。

特別なカードを用意しなくても、ディスクさえあれば手軽にRAIDを構築できるのは、ソフトウェアRAIDの大きな利点だ。データディスクとしてRAIDを導入するだけなら比較的設定も簡単なので、まずは試してみるというのもよいアプローチだ。

```
# cd /usr/src/linux-2.2.17
# cat raid-2.2.17-A0 | patch -p1
# zcat ide.2.2.17.all.20000904.patch.gz | patch -p1
```

画面3　パッチあての手順



画面4　ソフトウェアRAID関係のカーネル設定

```
# cat /proc/mdstat
Personalities : [linear] [raid0] [raid1] [raid5] [translucent]
read_ahead 1024 sectors
md0 : active raid1 hdg1[1] hde1[0] 20544 blocks [2/2] [UU]
md1 : active raid1 hdg5[1] hde5[0] 5120000 blocks [2/2] [UU]
md2 : active raid1 hdg6[1] hde6[0] 131392 blocks [2/2] [UU]
unused devices: <none>
```

画面5　/proc/mdstatでRAIDデバイスの状態を見る

▼リスト4　/etc/raidtabの例（RAID 0）

```
raiddev                 /dev/md0
raid-level              0
nr-raid-disks           2
chunk-size              64k
persistent-superblock   1
    device              /dev/hde1
    raid-disk           0
    device              /dev/hdg1
    raid-disk           1
```

▼リスト5　/etc/raidtabの例（RAID 1）

```
raiddev                 /dev/md0
raid-level              1
nr-raid-disks           2
chunk-size              64k
persistent-superblock   1
nr-spare-disks          0
    device              /dev/hde1
    raid-disk           0
    device              /dev/hdg1
    raid-disk           1
```

テストと評価

今回取り上げたIDE RAIDカード3機種は、Linuxで利用する場合の実現方法がそれぞれ異なる。それに価格の面では、Escalade 3W-6400のターゲット層は、ほかの2機種とは異なっている。そのため、数値的なデータを横並びで比較するだけでは、評価としては不完全と言える。そこで今回は、

- **RAID 0での性能比較**
- **RAID 1での性能比較**

に加え、

- **RAID 1での使い勝手**

の検証も行い、総合的な評価を行うこととした。

RAID 0は、冗長度がなく、データのバックアップについては、別途検討する必要がある環境である。それに対してRAID 1は、データの二重化によって、故障などの障害が起きても、運転をなるべく止めないためのシステムである。同じRAIDカードでも、そのために必要な作業量や手間は異なると思われるからだ。言い換えれば、楽に管理ができ、トラブルからの復活が速やかな製品を見つけようということだ。

RAID 0、1ともに、紹介した3種類のIDE RAIDカードに加え、比較対象用にマザーボードのプライマリIDEコネクタに接続したドライブでもデータを測定している。チップセットのApollo Pro133Aに組み合わされているI／Oコントローラ（サウスブリッジ）は、UltraDMA/66に対応している。

テスト環境

テストに用いたPCの構成は、**表1**のとおりだ。“ / ”（ルート）パーティションには、マザーボード上のプライマリIDEコネクタに接続したIBM DTLA-307030を用いた。先頭から10Gバイトのパーティションをテスト用にした。

またRAID環境は、各カードにIBM DTLA-307030を2台接続し、RAID 0またはRAID 1のボリュームを作成して、先頭から20Gバイトのパーティションを設定した。なお、どのRAIDカードでも2台のディスクは、それぞれ別のコネクタに接続してある。

ファイルシステムは、標準のExt2を用いている。フォーマット時には特にオプションを指定せず、ブロックサイズ、iノードの個数などはデフォルトのままとした。

ディストリビューションは、Red Hat Linux 6.2Jを用いた。カーネルは、Hot Rod 100 Proについては2.2.17、それ以外は2.2.14を用いている。これはHot Rod 100 Proを利用するために必要なパッチが、カーネル2.2.17用であるためだ。

転送モードの設定

今回使用したDTLA-307030は、最新のUltraDMA/100に対応したディスクだ。しかし、そのことがOS側に正しく認識され、転送モードを適切に設定されてはじめて、その性能をフルに発揮することができる。

Linuxでは、IDEディスクのチューニング用にhdparmというコマンドが用意されており、転送モードの設定や、現在の設定の確認などが行える。今回の環境では、どのカードに接続した場合でも正しく認識され、最適な転送モードが選択されることを確認した。すなわち、FastTrak100、Hot Rod 100 Proの2機種では、UltraDMA/100（UltraDMA mode5）、Escalade 3W-6400とマザーボードのコネクタに接続した場合は、UltraDMA/66（UltraDMA mode4）である。DTLA-307030は、ピーク転送速度でも66Mバイト／秒を越えないので、UltraDMA/100と

マザーボード	Microstar MS-6309
チップセット	VIA Apollo Pro133A（VT82C694X）
CPU	Intel Pentium III 733MHz（FSB 133MHz）
メモリ	PC133 SDRAM 256Mバイト
ハードディスク	IBM DTLA-307030（30Gバイト）×3
グラフィックスカード	Canopus SPECTRA 7400DDR（GeForce256DDR）
OS	Red Hat Linux 6.2J
カーネル	2.2.14、2.2.17

表1　テストに使用したPCのスペック

プログラム	URL
HDBENCH Clone	http://www.enjoy.ne.jp/~gm/program/hdbench/index-ja.html
Bonnie＋＋	http://www.coker.com.au/bonnie++/

表2　使用したベンチマークプログラムの入手先

UltraDMA/66の違いによる影響は、ほとんどないと考えられる。

ベンチマークソフト

今回ベンチマークソフトとして用いたのは、HDBENCH CloneとBonnie++の2種類だ（**表2**）。

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、描画速度なども測定できるが、今回はディスクの読み書き速度の測定項目のみ使用した。キャッシュの影響を受けないようにするために、測定するサイズを実メモリの2倍である512Mバイトに設定した。各カードごとに測定を3回行い、平均値を求めている。

Bonnie++は、前身であるBonnieに拡張を加えたものだ。最新バージョンである1.00dを使用した。Bonnie++は、POSIX標準のC関数を用いてファイルシステムにアクセスし、その転送速度とCPU使用率を測定する。単一のファイルを用いた転送速度の測定以外に、多数のファイルを生成する測定項目（ファイルシステムの性能測定に利用）がある。今回はディスクやインターフェイスの性能に深く関係する、ブロック単位のシーケンシャル読み出し／書き込みの項目を利用した。

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

```
$ bonnie++ -s 500 -d /raid -m
 test01
```

上の例では、/raidディレクトリに500Mバイトのファイルを作成してテストを行い、結果をtest01というファイルに出力している。サイズの指定が小さ過ぎると、メモリの2倍にするように注意されてしまうようだ。

実際の測定でも、上記のコマンドラインのように指定している。HDBENCH Cloneと同様、キャッシュの影響を受けないように、ファイルのサイズを実メモリの約2倍の500Mバイトに指定した。各カードごとに測定を3回行い、平均値を測定値としている。

アプリケーションベンチマーク

HDBENCH CloneやBonnie++は、ベンチマーク専用のプログラムであり、ディスクの読み書きを一定の基準で測定し、転送速度という数値で結果を示している。今回はこれ以外に、ディスクアクセスが比較的多いと思われる、実際のプログラムを動作させて、所要時間で性能比較をしてみることにした。

用いたのは、カーネルコンパイルとmkisofsコマンドによるisoイメージファイルの作成である。

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、

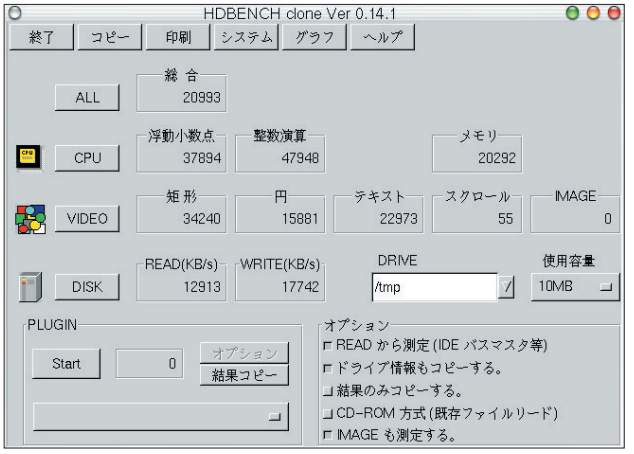
```
# make dep
# make clean
# time make bzImage
```

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、2.2.14である。各カードごとに測定を3回行い、平均値を測定値としている。なお、キャッシュの影響を小さくするために、メインメモリを64Mバイトに制限して行っている。

isoイメージファイルの作成は、mkisofsコマンドの所要時間を測定した。測定時間を長くするために、CD-ROM 3枚分（Red Hat Linux 7Jのdisc 1〜3）のファイルから約1.6Gバイトの巨大なisoイメージを作成した（現状ではこのようなisoイメージを焼けるメディアは存在しない）。カーネルコンパイルと同様に、

```
$ time mkisofs -o test.iso ./test/
```

として所要時間を測定した。各カードごとに測定を3回行い、平均値を測定値としている。



画面1　HDBENCH Clone

RAID 0

まずは、性能向上を目的としたRAID 0構成での結果から始めよう。**グラフ1**、**2**は、それぞれHDBENCH Clone、Bonnie＋＋による転送速度の測定結果だ。比較用のドライブ1台でも読み書きともに30Mバイト／秒を上回る性能が出ている。IBM DTLA-3070xxシリーズが、現状で最も速いIDEディスクのひとつであることを、あらためて見せ付けられた感がある。

さらにRAID 0構成では、ドライブ1台の場合と比較して、最低でも50%以上転送速度が向上していることが分かる。ディスクの読み書きが多い作業の速度改善が期待できる。

Hot Rod 100 Proは最も安い製品ではあるが、実売価格で4倍以上の開きがあるEscaladeと同等の性能を示している。Hot Rod 100 Proは、専用ドライバが存在しないため、Linuxカーネルの機能を用いたソフトウェアRAIDで動作させている。それでもこの性能が出せることを考えると、「ソフトウェアRAIDだから遅い」と単純に結論づけるわけにはいかない。テストマシンに搭載されている、Pentium III 733 MHzクラスの比較的高速なCPUがあれば、ソフトウェアRAIDもハードウェアRAIDと同等の転送速度が出せるようだ。

一方、**表3**にBonnie＋＋測定時のCPU使用率を示した。ディスク単体がDMA転送に対応していても、RAID 0のシステムでは、CPU使用率は上昇する傾向にあるようだ。それでも、ハードウェアRAIDのEscalade 3W-6400は、ほかの2機種と比較すれば、低めの値を示している。

カーネルコンパイル

グラフ3は、カーネルコンパイルの所要時間を比較したものだ。どのIDE RAIDカードでも、ドライブ1台とほとんど変わらないという結果に終わった。コンパイルは、ソースファイルの読み込みと、オブジェクトファイルの書き出しを行うので、速いディスクがあれば高速化できると言われるが、今回の実験ではほとんど影響がなかった。

通説と正反対の結果に終わった理由は、以下の二点と考えられる。

■キャッシュが効いている

メインメモリを64Mバイトに制限しても、常にある程度のメモリがキャッシュに利用されている。カーネルのソースは、大多数が10～100Kバイト程度の小さなファイルであり、そこから生成する中間ファイルもそれほど大きくはならないはずだ。したがってほとんどの場合、中間ファイルはキャッシュ内から読み出されるので、ディスクの速度の影響を受けなかったのであろう。

■単体のディスクが充分に速い

timeコマンドは、所要時間とともに、CPU使用率も算出するが、カーネルコンパイルの測定時には、どの条件でもほぼ100%の値が出ていた。つまり、単体で30Mバイト／秒以上の性能を持つDTLA-307030ならば、Pentium III 733MHzを待たせることはほとんどないということだ。ここからさらに性能を上げようとするならば、より高速なCPUに変えるか、あるいはデュアルCPU化するほうが有効であろう。

mkisofs

グラフ4は、約1.6Gバイトのイメージファイル作成の所要時間を比較したものだ。こちらは、見事にストライピ

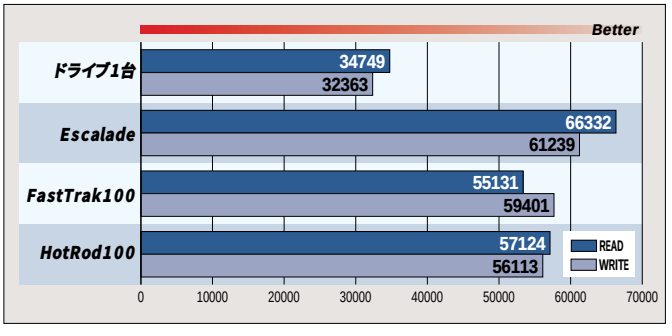
Bonnie＋＋	Read	Write
ドライブ1台	44%	16%
Escalade	61%	42%
FastTrak100	88%	58%
Hot Rod 100 Pro	68%	70%

表3　Bonnie＋＋実行時のCPU使用率（%）

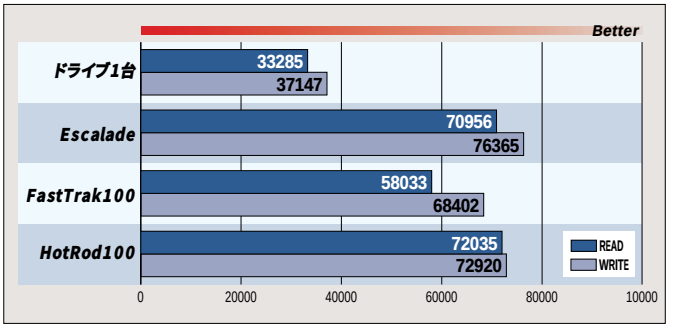
mkisofs	
ドライブ1台	34%
Escalade	34%
FastTrak100	44%
Hot Rod 100 Pro	42%

表4　isoイメージファイル作成時のCPU使用率（%）

グラフ1　HDBench Clone（Kバイト／秒・RAID 0）



グラフ2　Bonnie++（Kバイト／秒・RAID 0）



ングの効果が表れる結果となった。

表4には、イメージファイル作成時のCPU使用率を示す。ハードウェアRAIDのEscalade 3W-6400では、ドライブ1台の場合とほぼ同じ値を示しているのに対し、FastTrak100とHot Rod 100 Proは、10%ほど高い値になっている。isoイメージファイルの作成は、ほとんどディスクの読み書きだけの作業であり、カーネルコンパイルよりもディスク転送速度の影響を大きく受けることが分かるだろう。

RAID 0を使うべきか？

RAIDが提唱されてから10年以上が経過し、その間にコンピュータ部品の性能は、飛躍的に向上した。その結果、RAIDシステムの意義も変化してきたように感じられる。

たとえば、かつてはRAID 0の転送速度が必要だった用途も、ハードディスクの性能が上がったため、1台のディスクでも間に合ってしまうようになってきた。カーネルコンパイルがその好例だ。測定時には、メインメモリを減らして、ディスクI／Oの影響が出やすいようにしても、ほとんど差がつかなかった。実際の運用時には、もっと潤沢にメモリを搭載するので、ディスク性能の影響はさらに小さくなるだろう。プログラム開発のように、比較的小さなファイルを順次読み書きするような用途なら、高性能なハードディスク1台とデュアルCPUといった構成のほうが、適しているだろう。

その一方で、高速なディスクを2台束ねて作った、より速いディスク環境が有効な用途もある。今回はisoイメージファイル作成に使ってみたが、動画や音声データなど、数100M～Gバイトオーダーの巨大なファイルを加工・編集する場合には、RAID 0は必須とも言えるだろう。

筆者は、毎月のLinux magazine CD-ROMの作成も担当している。isoイメージファイルの作成は、一世代前のハードディスク（15Gバイト）を載せたマシンで行っているのだが、650Mバイトのイメージファイルを作るのに、3分くらいかかっている。いつもしめきり間際になって、「これも入れてね」と収録するファイルが増えるので、何回もイメージファイルを作ることになる。そのたびに3分待つのは非常につらいのだが、IBM DTLA-3070xxシリーズ（**写真1**）のような最新の製品でRAID 0を組めば、650Mバイトのイメージファイルは1分以下で作れるのだ。この差は大きい。評価に使用したRAIDカードとディスクを確保して、次号から使うことを固く決意している次第である。

どれを選ぶべきか？

価格やユーザー層の異なる3機種を評価してきたわけだが、RAID 0で利用する場合には、それほど大きな差はないというのが正直なところだ。確かに転送速度やCPU使用率の点で、Escalade 3W-6400は優れてはいるが、約4倍の価格差を払うだけの価値があるかどうかは疑問だ。

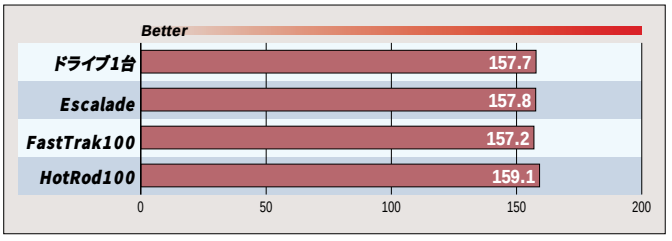
今回ソフトウェアRAIDとして利用したHot Rod 100 Proが、意外に良い成績を出したこともあるので、RAID機能を持たない（＝もっと安い）拡張IDEカードを選んで、ソフトウェアRAIDで利用するという選択肢も考慮すべきだろう。どうせ今時のCPUは、パワーが有り余っているのだから。

ここまでRAID 0の速度だけを強調してきたが、実際に運用する際には、冗長度がないために、1台のディスクの故障が即、全データの消失につながるということを忘れてはならない。間違っても、多くのユーザーが利用するマシンの/homeディレクトリには使わないほうがいいだろう。

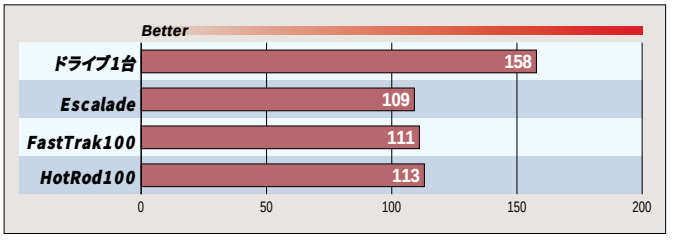
RAID 0システムは、適切な用途に使えば、あなたの環境を確実に改善してくれるだろう。

写真1　IBM DTLA-307030

グラフ3　カーネルコンパイル（秒・RAID 0）



グラフ4　mkisofs（秒・RADI 0）



RAID1（ミラーリング）での運用

ミラーリング環境での運用中にディスク異常が発生したら、ディスクを交換し、ミラーアレイを再構築しなければ耐障害性が低下する。

ここでは、3W-6400、FastTrak100、Linuxのソフトウェア RAID（Hot Rod 100 Pro）それぞれについて、ミラーリング環境下でディスク障害が起きた場合の動作、障害からの復旧手順を見ることにする。実際の動作を紹介する前に、運用にあたって注意してほしい点を先に述べておこう。

ミラーリングの落とし穴

ミラーリングによって、ディスクシステムの耐障害性能は向上するが、これだけではデータ保護が万全ではないことに気をつけたい。たとえば、ユーザーの誤操作によるファイルの消去や書き換えに対して、ミラーリングはまったくの無力である。また、電源異常や火災などの理由によって、2台のディスクが同時に壊れる可能性もある。これらの問題に対応できるのは、ミラーリングではなくバックアップである。従って、ミラーリングしたからといってバックアップの必要性が低下することはない。火災や地震に対しては、バックアップメディアを物理的に離れた場所に保管する必要がある。

忘れがちなのがRAIDカードの故障対策だ。RAIDカードが壊れてしまうと、いくらディスクを多重化していても意味がない。異なる種類のRAIDカード間では、RAIDアレイの互換性は期待できない。そのため、稼働しているものと同一のRAIDカードを予備として用意することが望ましい。

さらに、スペアディスクを装備するのもよいだろう。スペアディスクとは、ミラーアレイを構成するディスクの片方に障害が起きたときに、それを代替する予備のディスクのことだ。スペアディスクを接続しておくことで、ディスク障害の発生と同時に、自動的にミラーアレイを再構築できる。本特集でとりあげた、Escalade 3W-6400、Promise FastTrak100、Linuxのソフトウェア RAIDのいずれもスペアディスクを使うことが可能だ。

それでは、ミラーリング環境での動作を機種ごとに紹介し、最後にパフォーマンスを検証してみよう。

Column

PCIの限界

今回使用しているIDE RAIDカードは、いずれも一般的なPCIスロット用の拡張カードである。PCIバスの転送速度は、

32ビット×33MHz＝133Mバイト／秒

のピーク帯域を持つが、実効帯域は80〜90Mバイト／秒程度と言われている。

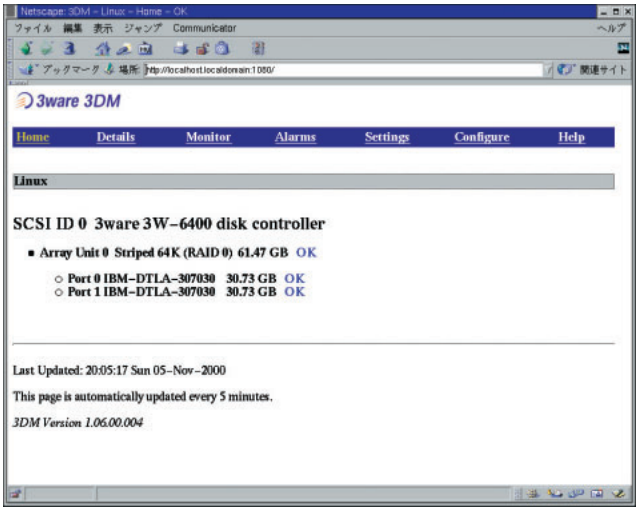
ここまでのテスト結果からわかるように、ディスク2台によるストライピング時の転送速度は、70Mバイト／秒前後に達しており、これだけでPCIバスの実効帯域を大部分使いきっている。テストは行っていないが、仮にディスクを3または4台に増やしてストライピングを行ったとしても、転送速度は2台の場合の1.5〜2倍にはならず、頭打ちになるだろう。そのうえ信頼性は、台数が増えることで、確実に低下する。結局、PCI接続のIDE RAIDカードを用いて高速なファイルシステムを構築するなら、ディスク2台の構成がコストパフォーマンス的、バランス的に優れているといえる。

インテルの440BX以前の製品や、VIAのAppolo Proシリーズは、チップセットどうしもPCIバスを介して接続している。そこで、PCIバスの混雑具合を少しでも緩和するため、チップセット間を専用の高速バスで接続する製品が登場している。インテルの800番台のチップセットやVIAのDDR SDRAM対応チップセット（未発売）などがその例だ。今後はこのデザインが主流になるだろう。

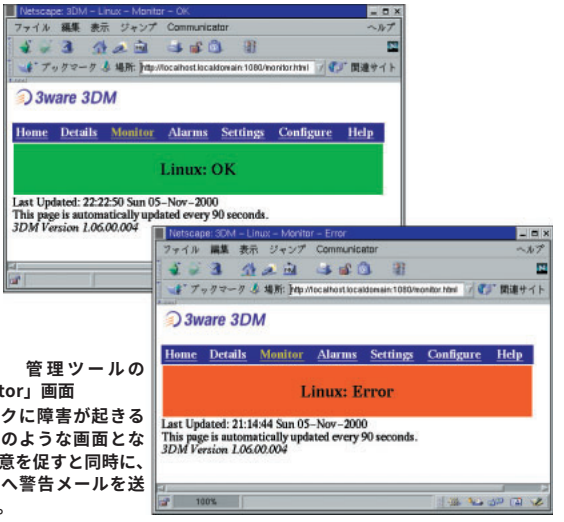
PCIバス自体の拡張としては、33MHz／32ビットから、66MHz化または64ビット化、もしくはその両方というものが規格化されており、サーバなど高帯域のI／Oが必要なマシンでは、すでに利用されている。これにより最大で533Mバイト／秒までピーク帯域は拡大するが、64ビットPCIバスはコネクタや拡張カード自体の巨大化につながるため、一般向けマシンでは採用しにくい。また66MHzのPCIバスはマザーボードの設計が難しくなるため、これまた一般用のマシンで採用されることはないと思われる。

ほかにもIBM、HP、Compaqが推進するPCI-Xや、インテルが推進するInfiniBandといった次世代の拡張バス規格は存在するが、どれも（少なくとも最初のうちは）ハイエンドのサーバ／ワークステーション用途向けであり、仮に一般ユーザーが利用できるとしても、それは何年も先のことになるだろう。

結局、当分の間我々は、PCIバスの133Mバイト／秒という帯域を、なんとかやりくりしていくしかないようだ。



画面2　3W-6400の管理ユーティリティ
Webブラウザで RAIDのモニタ、メンテナンスを行うことが可能。デフォルト設定では1080番のポートを使う。



画面3　管理ツールの「Monitor」画面
ディスクに障害が起きると右下のような画面となり、注意を促すと同時に、管理者へ警告メールを送信する。

ミラーリング環境での動作を調べるためには、ディスクに障害を起こす必要がある。ハードディスクを、破壊してしまうわけにはいかないので、RAIDアレイをマウントした状態で、1台のハードディスクの電源コネクタを抜くことにした。仮想的にディスク障害を発生させたわけである。この状態でRAID構成のディレクトリにファイルを作成し、動作を調べた。

Escalade 3W-6400

Escalade 3W-6400には、3DM Management Utilityという管理ソフトウェアが付属する。Linux版も用意されており、Webブラウザを使ってRAIDアレイの状態をモニタしたり、メンテナンスを行うことが可能だ。

このプログラムはデーモンとして動作しており、3W-6400のデバイスドライバと通信することで各種の機能を実現している。cshスクリプトで書かれたインストーラを起動すると、次に示す質問が表示されるので、環境に合わせて答えを入力する。

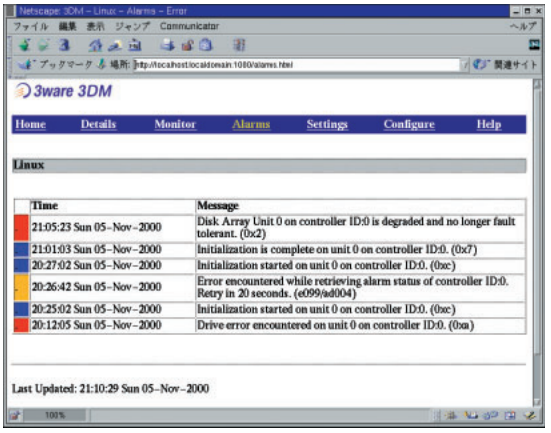
- **イベントをメールで知らせるか？**
- **メールサーバの指定**
- **メールを送るアドレス**
- **メールの送り元**
- **イベントでビープ音をならすか？**
- **Webインターフェイスで使うポート**
- **ドキュメントのインストール先**
- **ディストリビューション**

これで、管理ツールが自動的に動作するようになる。

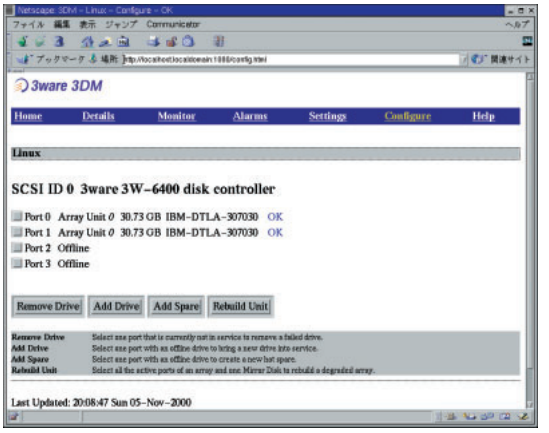
■強力な管理ツール

Webブラウザを使い、管理ツールにアクセスすると**画面2**のようなページが開かれる。RAIDアレイの状況を示す「Monitor」ページは、90秒ごとに自動更新されるので、まさに“モニタ”用途に使えるだろう（**画面3**）。イベントログを残せるのも特長のひとつだ（**画面4**）。「Details」というリンクを選べば、RAIDアレイやコントローラ、ディスクの情報をさらに詳しく表示することもできる。

このツールの目玉となるのが「Configure」画面だ（**画面5**）。この画面から、RAIDアレイへディスクを追



画面4　管理ツールの「Alarm」画面
ディスクのエラー発生やアレイの再構築など、今までのイベントログを参照できる。



画面5　管理ツールの「Configure」画面
Linuxを動作させたままでRAIDアレイのメンテナンスができる。

加／削除したり、スペアディスクを追加できる。さらに、問題の生じたディスクを交換したあとに必要なアレイの再構築もこの画面でできるのだ。

■3W-6400での障害復旧

ディスクに障害が発生すると、メールや「Monitor」画面で管理者へ警告し、syslogに対しエラー発生のログを送る。Linuxの動作自体はそのまま続けることができるが、この状態では耐障害性はないので注意したい。

シャットダウンして、ディスクの障害を取り除いてから起動すると、3W-6400のBIOS表示で、「DEGRADED」と表示され、アレイがおかしいことを示した。ここでBIOS設定画面を呼び出し、アレイを再構築してもよいが、そのままLinuxを起動し、管理ツールの「Configure」画面から再構築することも可能だ。

BIOSで再構築する場合は、しばらくの間（30Gバイトのディスクで1時間弱）システムがダウンする。これに対し、Linuxを起動して再構築する場合はディスクのパフォーマンスはかなり低下するものの、システムを止める時間は少しで済む。

Promise FastTrak100

現時点では、FastTrak100のLinux用の管理ツールは提供されていない。今後、ベンダーあるいはユーザーから発表される可能性はあるので、期待したいところだ。

■FastTrak100での障害復旧

ディスク障害発生時は、コンソール画面にエラーが表示されるとともに、**リスト1**のようなログがsyslog経由で/var/log/messageファイルに書き込まれる。この例では、Channel 0にマスタドライブとしてつないだハードディスク（IBM DTLA-307030）がDisk Offlineでエラーになっていることがわかる（4123はDisk Offlineのエラーコード）。

1台のディスクを使い、継続してLinuxを利用できるが、もちろん耐障害性はない状態である。

マシンを再起動するとFastTrak100のBIOSを表示したところで停止する。交換するなどして、ディスクの問題を取り除いているなら、ここでBIOS設定画面を呼び出してアレイの再構築を行うことも可能だが、Linuxを起動すれば、ドライバが自動的にアレイの再構築を開始する。

ソフトウェアRAID

LinuxのソフトウエアRAIDを利用する場合は、前述の/proc/mdstatに状況が書き込まれる。アレイの状況を見たいときは、次のようにすればよい。

```
cat /proc/mdstat
```

IDEドライブでソフトウェアRAIDを利用する場合、ディスクに障害が発生すると、IDEのドライバがエラーを起こして、たいていはシステム自体が停止する。シャットダウンを行うこともできないので、電源を切るしかない。

ソフトウェアRAIDでは、復旧作業もLinuxを起動してから行うことになる。今回のテストのように、電源コネクタを抜いたのを元に戻して起動した場合は、自動的に再構築スレッドが起動され、アレイを作り直してくれた（**画面6**）。Linuxの起動途中でエラーを表示するとともに、rootユーザーのパスワードを求めてくることもある。この場合はパスワードを入力するとシェルの画面になるので、コマンドを駆使してディスクの状況を見たり、設定を変更することになる。

ミラーリングに向くのは？

ソフトウェアRAIDは、専用のBIOSを持つRAIDカードに比べるとトラブルシュートが難しくなるが、その仕組みを考えると仕方がないのかもしれない。ミラーリングに関するテストでは、強力な管理ツールが付属するEscarade 3W-6400の使い勝手が際だっていた。FastTrak100にもLinuxで利用可能な管理ツールが登場することを願ってやまない。

▼リスト1　ディスク障害発生時にFastTrak100が出力したログ

```
Oct 22 05:03:45 lmtpc03 kernel: lalal
Oct 22 05:03:46 lmtpc03 kernel: FASTTRAK|scsi0|4123|Error|Disk
Offline: IBM-DTLA-307030 (Channel 0, Master)
```

```
Personalities : [linear] [raid0] [raid1] [raid5] [translucent]
read_ahead 1024 sectors
md0 : active raid1 hdg1[1] hde1[0] 20544 blocks [2/2] [UU] resync=DELAYED
md1 : active raid1 hdg5[1] hde5[0] 5120000 blocks [2/2] [UU] resync=19%
finish=3.2min
md2 : active raid1 hdg6[1] hde6[0] 131392 blocks [2/2] [UU]
unused devices: <none>
```

画面6　ミラーアレイ再構築中の/proc/mdstat

ミラーリング環境でのディスクパフォーマンス

ミラーリングは耐障害性を向上させるのが目的であり、ディスクパフォーマンスの向上は本来見込めないものである。とはいえ、やはりパフォーマンスも気になるものだ。そこで、106ページからのストライピングのパフォーマンステストと同環境において、ミラーリングでのパフォーマンスを測定してみた。

「RAIDの基礎知識」でも解説したように、ミラーリングは複数のハードディスクに同じデータを記録するため、ハードディスク1台にデータを記録するのに比べるとパフォーマンスが低下する可能性が高い。その一方で、データの読み出しはストライピングと同様の手法が使えるので、ハードディスク単体で使うより高いパフォーマンスを得ることもできる（ドライバがそのように実装されていればだが）。では、各ベンチマークテストごとに結果を見てみよう。

■HDBench Clone

グラフ5を見ると、Escalade 3W-6400とFastTrak100はドライブ単体とほぼ同じくらいのパフォーマンスを発揮した。Hot Rod 100 Proでのソフトウェアの RAIDでは、データ書き込み時の速度が低くなっている。

■Bonnie++

驚いたことに、Escalade 3W-6400とFastTrak100はドライブ単体よりもよい結果を出した（**グラフ6**）。ソフトウェアRAIDでは、書き込み時の性能があまりよくなかった。

■カーネルコンパイル

まったくといってよいほど差がない（**グラフ7**）。グラフには記載していないが、いずれの場合もコンパイル開始から終了までの平均CPU稼働率が98～99%となっていたので、所要時間への影響はCPU性能が大部分を占め、ディスク性能の影響は極めて少ないのだと考えられる。

このテストではPentium III 733MHzを搭載したマシンを使ったが、もっとCPU性能が低いマシンでは結果に差が出たかもしれない。

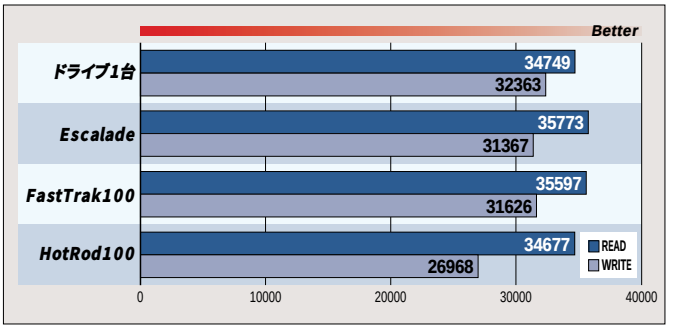
■mkisofs

ディスク上のデータをひたすら読みながら、巨大なファイルに書き込んでいくテストだ。Escalade 3W-6400とFastTrak100が非常によい結果を出したのに対し、Hot Rod 100 Proでのソフトウェア RAIDではパフォーマンスの低下が認められる（**グラフ8**）。しかし、気になるほどの速度低下ではない。また、CPU稼働率はソフトウェアRAIDが最も低く抑えられていた。

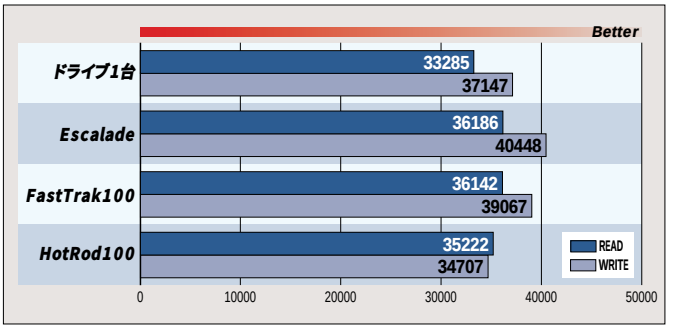
テスト結果のまとめ

全体を通してみると、ミラーリングは遅いという考えは捨てなければならない。ハードディスクはもちろん、CPUも高速になったことで、ソフトウェアRAIDであっても単体のハードディスクに迫るパフォーマンスを発揮することが判明した。

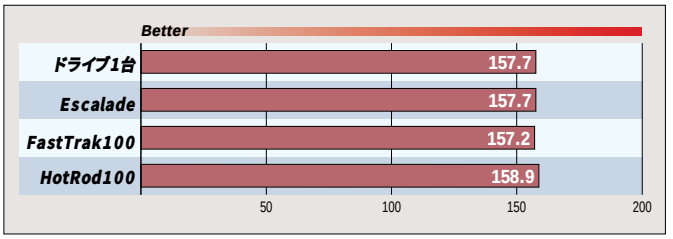
グラフ5　HDBench Clone（Kバイト／秒）



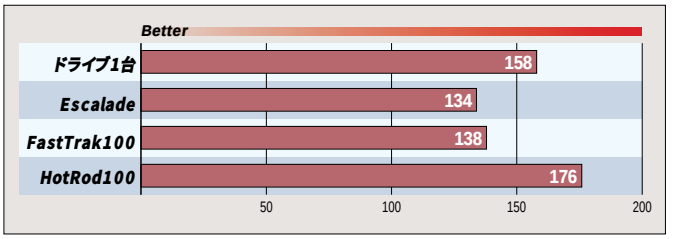
グラフ6　Bonnie++（Kバイト／秒）



グラフ7　カーネルコンパイル（秒）



グラフ8　mkisofs（秒）



2台から始める Linuxクラスタ

最終回　クラスタサーバを使う

Linuxクラスタの連載も今回で終了である。最終回は、TurboLinux Cluster Serverの応用とユーザー事例、そしてBeowulf、enFuzionといった高速クラスタについてと、Webクラスタを簡単に構築できるKondara MNU/Linux Web Cluster 2000を紹介する。なお、付録CD-ROMにTurboLinux Cluster Server 6の評価版を収録したので、興味のある方は試してほしい。

Photo : Takashi Shinoha (Dee)

クラスタ応用編
ユーザー事例と高速クラスタの使い方

文：川村 智　日立エンジニアリング（株）コミュニケーションシステム部
Text : Satoshi Kawamura

先月はTurboLinux Cluster Serverを使用して、2ノードのWebサーバ構築手順について説明した。最終回はその応用例を説明したあと、導入ユーザー事例を紹介する。そして入門編の最後として、高速クラスタを説明する。

TurboLinux Cluster Server応用編

先月号の続きとして監視エージェントの話や、その他のオプション設定の注意点について説明する。

■2台以上の場合

前回の例では、1台のマシンでアドバンスドトラフィックマネージャとサービスクラスタを兼ねていたが、**図1**のように、アドバンスドトラフィックマネージャとサービスクラスタが分離する場合は、サービスクラスタにはTurboLinux Cluster Server 6.0をインストールする必要はない。その場合、サービスクラスタに対してクラスタ用の仮想サーバIPアドレスを割り当てておく必要がある。具体的には、

```
ifconfig eth0:1 仮想IPアドレス up
ifconfig lo:1 仮想IPアドレス netmask
255.255.255.255 up
```

を/etc/rc.d/rc.localの最後の行に追加しておくとよい。

また、サービスクラスタは、Linuxに限らずWindowsやSolarisプラットフォームを利用することも可能だ。

■標準で添付されている監視エージェント

Webサーバ以外のサービスもできるように、監視エージェントプログラムが同梱されている（**表1**）。FTPやメール、さらにはOracleやDB2などの商用リレーショナルデータベースもサポートしている。

ただし、データベースを並列化して使用する場合は参照のみに限定しないと同じ内容をクライアントに提供できなくなるので注意して欲しい。

更新を行いたい場合は更新用のDBサーバを設けてサービスクラスタから更新用DBサーバにアクセスするようにする。そして更新専用DBサーバから参照DBに対してレプリケーションを実施するような設定にするとよいだろう（**図2**）。

アプリケーションが若干複雑になるので面倒だと思われる方は、発表予定のLinux用パラレルOracleを待つのもよいだろう。最大で4台までの計算機を使って参照＆更新が並列に処理できる予定になっている（と聞いている）。

■独自のエージェント作成

また独自にエージェントを作成することも可能だ。

標準の監視エージェントよりきめ細かいチェックをしたい場合や、ユーザー側で独自のTCP/IP通信プログラムを持っている場合、インターフェイス仕様（コマンドラインの引数）を合わせた監視用プログラム（シェルスクリプトを含む）を作成すればいい。

具体的には引数として「監視ホスト名」「ポート番号」「通信プロトコル」の順に渡される。通信プロトコルは、

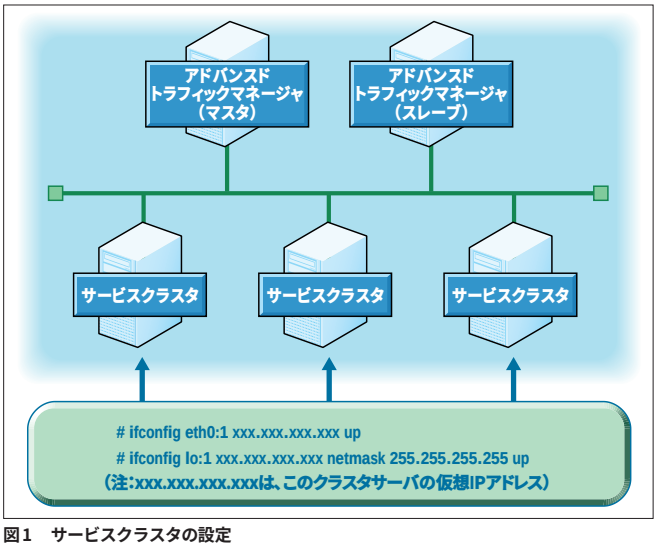


図1　サービスクラスタの設定
アドバンスドトラフィックマネージャとサービスクラスタが別のマシンの時には、サービスクラスタに仮想IPアドレスを設定する。

エージェント名	監視するサービス（サーバ）
oracleAgent	Oracleサーバ
db2Agent	DB2サーバ
dnsAgent	DNSサーバ
ftpAgent	FTPサーバ
httpAgent	HTTPサーバ
httpsAgent	HTTPSサーバ
http10Agent	キャッシュサーバ（squid）
imapAgent	IMAPサーバ
popAgent	POP3サーバ
smtpAgent	SMTPサーバ
nntpAgent	ネットニュースサーバ

表1　監視エージェントプログラム

TCPの場合には1が、UDPの場合には0が指定される。そして、監視した結果が正常な時には0を、異常だったら1（エラー）を返すようにする。

■WebでCGIを使用する場合

先月号で説明したSetting Servicesの画面でデフォルトのサービスが、「(＊) Load Balance」になっていたが、これに加えて、「[＊] Allow Session Persistency」を設定するとクライアントとサービスクラスタの接続関係を固定化できる。

こうしておくとクライアント端末から接続要求があったとき、最初に受信したサービスクラスタが端末とのコネクションが切れるまでそのクライアントのサービスを連続して行う。

ホームページの中にCGIが含まれている場合、CGIの変数内容や処理データを次の画面に引き継ぐ必要がある。そのタイミングでサービスノードを固定していないと画面間の連携がうまくいかなくなる。そのようなときにこのオプションを指定するのだ。

しかしCGIがファイルに書き込み、その内容を別のプログラムが利用するタイプ（たとえばアクセスカウンタ）の場合には、別のサーバ上にNFSを利用して共有ディレクトリを作成し、その配下に書き込むファイルを置くように環境構築する必要がある。

■Failoverオプション

万が一のときに予備系に切り替えて使用するようなフェイルオーバークラスタとして構築する場合、Setting Servicesの画面で「(＊) Failover」を選択するようにする。

ダウンしては困るDNSサーバやMailサーバに有効なオプションである（ただし実行中のトランザクションは引き継がない）。

■ クラスタユーザー事例

IT革命といわれる現代においてクラスタシステムが必要とされる代表例は、24時間365日眠らないビジネスを支えるインターネットサーバであろう。

日本では株式会社メガ(http://www.mega.co.jp/)が、いち早くTurboLinux Cluster Serverを導入した（**写真1**）。

同社では数年前からLinuxベースのソリューションビジネスを展開していた。業績の拡大に伴い順次個別のサーバを追加していく従来の方式では限界を感じていたためだ。

最近では企業向けのサーバホスティングの仕事が増加し、信頼性やサーバの応答性を維持することが非常に重要になってきている。たとえば特定の企業にアクセスが集中し負荷が高くなると、そのサーバに同居している別企業のホームページの表示も遅くなってしまう。同じ料金を支払っているのにサーバによってレスポンスの差異が生まれるのは好ましい事態ではない。

また、サーバを維持するのは社会的使命でもあるのでハードウェアに障害があったときには、ただちに復旧作業をしなくてはならないのは当然だが、年々システムが複雑になり、結果として運用コストが徐々にアップしている状況だった。

そんな中、登録企業数が急激に伸びてきたため、サーバの統合やシステム構成の変更に迫られた。数社のベンダ

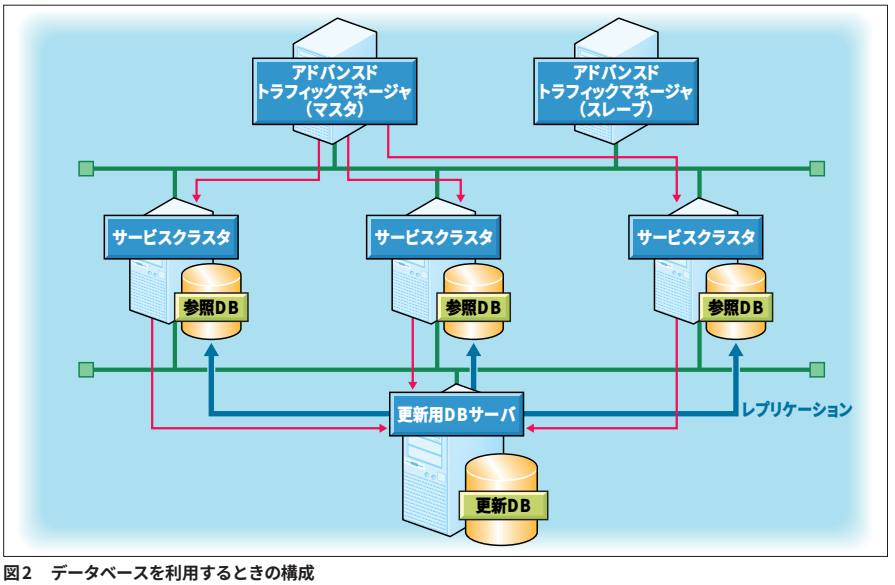


図2　データベースを利用するときの構成
クラスタでデータベースを利用するときには、更新用DBは専用サーバに置き、各クラスタにはレプリケーションで参照用DBを持たせる。

写真1　メガで使用しているサーバ（出荷直前に工場内で撮影）

ーにクラスタ構成の提案をもらい、検討した結果、Linuxベースのクラスタシステムの導入に踏み切ったのだ。

企業向けのホスティング業務をしていた同社では、ハードウェア方式のロードバランサ装置の導入は検討から除外した。数年経過したときにキチンと保守部品の交換などのメンテナンスができるかどうか、またそのハードメーカーが数年後存在しているのか、新しい技術に対応できるのか、予想外のクラック攻撃に耐えられるのかということまで考えると、特定のハードメーカーの技術に依存したシステム構成は避けるべきであると判断したからだ。

さらに技術革新の激しいネットワーク関連メーカーの浮き沈みは激しく、プロバイダ業者として今後の動作保証を考えて、これまでの同社の方針を踏襲しPC＋Linuxベースでクラスタシステムを構築するほうを選んだのだった。

スケーラブル＆フェイルオーバークラスタ構成にしておくことで、各種サービスを停止することなく、OSやハードを部分的に停止しながらアップグレードすることができるというメリットがある。実際、導入当初はTurboLinux Cluster Server 4.0だったが現在では6.0となり、文字通りシステム全体を停止せずにバージョンアップできた。

同時にカーネルや各種ドライバのチューニングを行った結果、導入時の10倍以上のパフォーマンスが得られるようになったのでプロジェクトメンバー全員ビックリしている。よくLinuxはバージョンアップするごとに性能が向上するといわれているが、これは極端な事例である。

詳細な内容は現時点で発表できないが、さらに同社では別のクラスタサーバを構築する予定で、新タイプのASP事業を実施するべく準備を着々と進めている状況である。

Beowulf型クラスタによる高速クラスタ

Beowulf型クラスタではPVM（Parallel Virtual Machine）やMPI（Message Passing Interface）を使用した並列計算ロジックの話になることが多い。MPIの名前や概念は知っている読者は多いと思うが実際に使った人は少ないと思う。

そこでこのMPIを使って簡単なプログラムを実行して実際に体験してみよう。こんなふうに簡単にできるのがLinuxの良いところだ。

■MPIの入手場所

今回はLAM（Local Area Multicomputer）/MPIを使用する。Linux用MPIのプログラムは、

```
http://www.mpi.nd.edu/lam/download/
```

にある。

■MPIのコンパイル

ソースファイルをダウンロードし、展開したあと、そのディレクトリで、

```
# ./configure
```

を実行する。ただしFortranが入っていないLinuxの場合、

```
# ./configure -without-fc
```

というオプションが必要だ。そして、makeを実行すると、/usr/local/lam-x.x.xにインストールされる（x.x.xにはバージョン番号が入る）。その下のbinディレクトリにプログラムがあるので、

```
# PATH=$PATH:/usr/local/lam-x.x.x/bin
```

とPATHに追加して実行可能にする。

同様の手順で並列処理対象のすべての計算機にインストールする。

■管理デーモンの実行

デーモンが起動していないと実行することができないので、lambootコマンドでlamというプログラムを起動する。

単体計算機の場合には、引数は不要だが複数の計算機で実行する場合には、対象ホスト名やIPアドレスを記述したbhost.lamファイルを作成する。

たとえば、sv1とsv2の計算機でデーモンを起動する場合、

```
sv1
sv2
```

と書いたbhost.lamファイルを準備する。そして、これを引数にして

```
# lamboot -v bhost.lam
```

と起動すると2台同時に立ちあがる。

■プログラムの実行

「Hello Cluster!! toホスト名」を表示するサンプルプログラムを**リスト1**に示す。これを先ほどmakeしたhccコンパイラでコンパイルする。

```
# hcc -o sample sample.c
```

正常にコンパイルできたら、下記のようにして実行する。2台の場合、

```
# mpirun -v -c 2 -s h ./sample
```

と実行して、

```
Hello Cluster!! to ホスト名称
Hello Cluster!! to ホスト名称
```

と表示されれば並列に処理されたことになるのだ。非常に簡単だが並列処理の雰囲気は味わえると思う。このfprintfの代わりにキチンとしたロジックを記述すれば、あなたも立派な並列プログラマーとして仲間入りできるのだ！

ただし、この簡単なプログラムの動作を見て頭脳明晰な読者にはMPIの使用上の注意点に気がつくはずだ。帳票集計などの実際の業務プログラムでは、並列に処理された出力結果を整理し編集するプログラムが別途必要だ。

たとえば出力結果を小さい順に表示する場合、sortコマンドで並列処理結果を整理するが、そこは1台の計算機で処理することになる。

したがって並列化プログラムを作成する場合は、並列部分とその結果を整理するためのシングルタスク部分の切り分けを、プログラマーやシステム設計者が十分に検討する必要があるのだ。

一般業務アプリケーションで並列処理をする場合は、かなりロジックが複雑になることが推察できると思う。

このように各要求仕様に合わせて、並列用のロジックを作成し、スクリプトなどを使って処理結果をまとめ、さらにその結果を別の並列処理プログラムに渡して……、と本格的なアプリケーションを作成していくと、第三者がメンテナンスできない複雑なプログラムになってしまう（作った本人も困惑するだろう）。

enFuzionの登場！

ところが、enFuzionを使用すれば、この制御部分がかなり簡単になり、プログラマーは単純に並列業務ロジックに専念できる。2台以上のPCを持っている読者なら実際に体験してみよう。

■enFuzionの入手

評価用ソフトが米TurboLinuxのホームページ（http://www.turbolinux.com/）からダウンロードできる。該当の製品のリンクページをたどってダウンロードする。

enFuzionが動作するベースのLinuxバージョンは、2.2.Xであれば基本的に動作するはずだ（筆者はTurboLinux Workstation日本語版6.0で動作検証した）。

ダウンロードしたファイルはtar形式のアーカイブになっている。これを/home/usr/など適切なユーザーディレクトリで、

```
# tar xvfz enfuzion.6.0.1-linux.2.2-i486.tar.gz
```

としてファイルを解凍すると、enfuzion.6.0.1-linux.2.2-i486というディレクトリが作られ、その下に各種プログラムが展開されているはずだ。

■動作チェック

インストール実行の前に、このLinux上でenFuzion自体が動作するかどうか動作検証を行う。X上のコンソールから、

```
# cd enfuzion.6.0.1-linux.2.2-i486
# ./enfdispatcher simple.run
```

を実行してエラーが発生しなければ、このLinux上でenFuzionが動くことが事前に検証できる。

■動作環境の設定

enFuzionはFTPやtelnetを使用するので、/etc/inetd.confのFTPやtelnetの設定内容を修正する。

```
ftp stream tcp ...  in.ftpd -l -a
telnet stream tcp ... in.telnetd -h
```

▼リスト1　MPIを利用するサンプルプログラム（sample.c）

```
#include <stdio.h>
#include <unistd.h>
#include "mpi.h"

main(argc, argv)
int     argc;
char    **argv;
{
        char    Hostname[128];

        MPI_Init(&argc, &argv);
        if( 0 != gethostname(Hostname, sizeof(Hostname)) ) {
                fprintf(stderr, "Error\n");
                exit(1);
        }
        fprintf(stdout, "Hello Cluster!! to %s\n", Hostname);
        MPI_Finalize();

        exit(0);
}
```

という2行の先頭に#がない状態にしておく。

またリモートで実行する計算機には、制御ノードであるルートノードからのアクセスを許可する必要がある。/etc/hosts.allowに、ネットワークサーバデーモン名と通信を許可するホスト名を記述する。先ほどの例であれば、

```
in.ftpd: ルートホスト名
in.telnetd: ルートホスト名
```

となる。これは各自の環境で異なる可能性があるので、/etc/inetd.confの内容をよく確認して定義して欲しい。たとえばproftpdを使用している場合は、in.proftpdなどになるはずだ。

なお、inetd.confやhosts.allowファイルを修正した場合は、下記のように行ってinetdデーモンを再起動すること。

```
# killall -HUP inetd
```

■並列処理対象ノードの設定

次に、enfuzion.6.0.1-linux.2.2-i486ディレクトリにある、enfuzion.nodesファイルを編集する。たとえば、enfuzionというユーザー名で、sv1.local.comとsv2.local.comの2台のマシンでenFuzionを利用する場合には、ホスト名、ユーザー名、パスワードの順で記述する。

```
sv1.local.com enfuzion パスワード
sv2.local.com enfuzion パスワード
```

■enFusionのインストール

準備ができたところで、

```
# ./enfinstall enfuzion
```

と入力する。インストーラの質問にしたがって入力していくとインストールが完了する。

■評価用ライセンスの入手

先ほどダウンロードした米TurboLinuxのenFuzionのWebページにある、ライセンスの申し込みページより評価用ライセンスを依頼する。しばらくするとメールでライセンス情報が送られてくる。

ライセンスがないとローカル環境でしか動作しないので必ず申し込もう。評価用ライセンスは無料で入手できる。

ライセンスをインストールするには、送られてきたメールの指示にしたがって、ライセンス行をenflicenseファイルに書き込み、

```
# ./enfinstall license
```

をルートホスト上で実行すると、関連ノードにライセンスが配布される。

インストールの確認を行うために、各enFuzionノードの状態を表示してみる。

```
# ./enfinstall verify
```

エラー表示がないことを確認しておこう。

enFuzionの実行

X Window Systemのコンソールから、先ほどインストールした、enfuzion.6.0.1-linux.2.2-i486のディレクトリで、

```
# ./enfuzion
```

と入力する。**画面1**のようにenFuzionの初期画面が表示される。enFuzionでは、左のボタンから順番にGUIベースで設定を進めていくことができる。

①Preparator：パラメータの定義
②Generator：並列ジョブの定義
③Dispatcher：並列ジョブの制御
④Exit：終了

今回は入門編なのでプログラミングの話は省略して、enFuzionの使い方を中心に説明しよう。

そこで店舗経営シミュレーションのshopという、待ち合わせ行列問題を解くプログラムが、あらかじめ用意されているという前提で話を進める。

このshopプログラムをどうやって並列処理させるかを検討してみよう。shopには引数が4つあるものとする。それぞれ、

-q：待ち行列数
-s：店舗内での顧客対応者数
-a：単位時間あたりの平均来店者数
-t：顧客1人あたりの平均対応時間

を意味する引数で、-qが5人、-sが6人、-aが10人、そして-tが0.25時間の場合、各引数をセットして実行すると、

```
% shop -q 5 -s 6 -a 10 -t 0.25
53.9842  5  6  10  0.25
```

のように1時間当たりに処理可能な顧客処理数と、その時の条件が表示される。

このソフトで4つの引数の値をいろいろ変化させて、顧客処理数が最大になるときの条件を求めてみよう。

といっても無限の組み合わせがあるので、説明のために下記の簡単な組み合わせで考えてみる。

-q：6通り
-s：7通り

-a：10通り
-t：5通り

たったこれだけの組み合わせでも、6×7×10×5＝2100通りあることがわかる。

これを手動で求めるには、

```
shop -q 1 -s 1 -a 10 -t 0.1 > output.1
shop -q 5 -s 2 -a 4 -t 1.0 > output.2
shop -q 9 -s 3 -a 7 -t 0.45 > output.3
:
```

というように順番に実行し、全部（2100回！）終了したあとに結果をソートして、一番大きな値を求めることになる。

```
% cat output.* | sort -n | tail -1
67.8488  9  3  7  0.45
```

手作業で2100通りすべて実施するのは非現実的なので、別途、多重ループ構造のシェルプログラムを作ることだろう。しかし、たとえ作ったとしても別の条件の組み合わせを考える場合には、スクリプトの修正作業が発生する。

これをLinuxやUNIXに精通したエンジニアが変更するのなら、あまり問題になることはないが、この種の経営シミュレーションプログラムは会社の経営権を握っている重役が使うものだ。泥臭いプログラミングの世界と無縁のエクゼクティブユーザーに対してはマウスやテンキー入力で使えるようにしないといけない（マウスも使えない重役はどうすればいいのかという相談を実際に受けるが……。回答不能です）。

これをenFuzionのツールを使うとGUIで非常に簡単に記述でき、実際に利用するユーザーもGUIで簡単に操作できるようになる。

■変数の定義

まずenFuzionで扱う変数を定義する。たとえば、各変数の名称を、

```
-q：numq     (Number of Queues)
-s：nums     (Number of Servers)
-a：arrivalr  (Arrival Rate)
-t：servtim (Average Service Time)
```

として作業を進める。

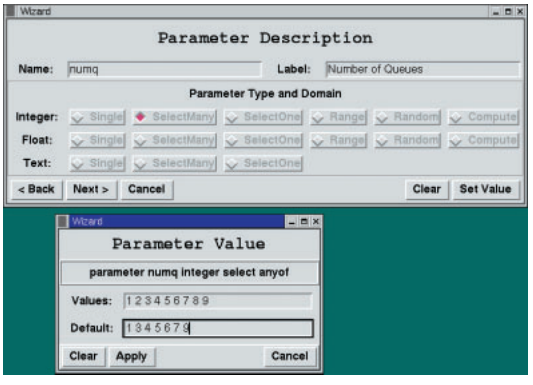
［Preparator］ボタンを押して、パラメータ作成ウイザードを呼び出して、待ち行列数をセットする（**画面2**）。

NameとLabelに適当な名前を入力し、Integerタイプの行の［SelectMany］タイプをセレクトする。［Set Value］ボタンを押すと任意の整数値が入力できるようになる。画面の例では7通りの値として、「1,3,4,5,6,7,9」を選択している。同様に、Number of Serversの条件を設定する。

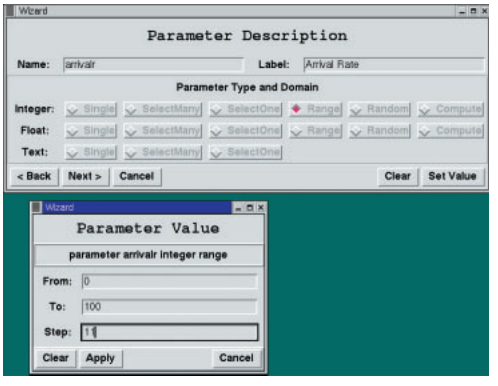
単位時間当たりの平均来店者数（Arrival Rate）の設定では［Range］タイプを指定している。この例では0から100までの間で、＋11ずつリニアに増加させていくパターンを示している（**画面3**）。

同様に、顧客1人当たりの平均対応時間（Average Service Time）では、0から5の間で、1.25ずつ増加させる。

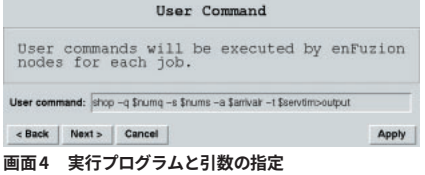
画面1　enFuzionの初期画面



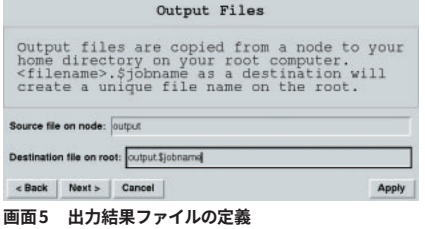
画面2　待ち行列数のデータの入力



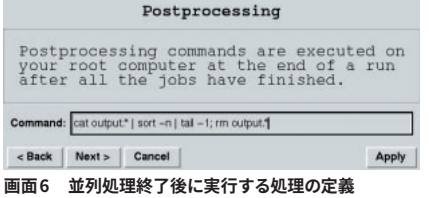
画面3　単位時間当たりの平均来店者数の入力



画面4　実行プログラムと引数の指定



画面5　出力結果ファイルの定義



画面6　並列処理終了後に実行する処理の定義

■実行プログラムの定義

変数の入力が終了したら、［Next>］ボタンを押す。「User Command」（**画面4**）ではプログラム名に続けて定義した変数に$マークを付けて記述する。

■出力ファイル名の定義

個々のノードで出力するファイル名をoutputと定義したのが**画面5**である。その個々のノードからルート計算機に送信するファイル名はユニークになるように、「output.$jobname」としておこう。

$jobnameはenFuzionで用意されている変数で、ジョブごとにユニークな番号が自動的に付けられる。

■並列処理終了のあと処理

output.＊ファイルをまとめ、ソート処理を行い、最大値となる最後の1行を表示して、output.＊ファイルすべてを削除する処理を定義したのが**画面6**である。

■プランファイルの内容確認

ここまで無事に完了すると「おめでとう！プランの作成が成功しました」という意味のメッセージが表示される。

ここで「shop.pln」というように適当な名前を付けてプランファイルを保存しよう。このプランファイルを、次のGeneratorで利用することになる。

この段階のshop.plnファイルの内容は、**リスト2**のようになっていて、かなりわかりやすい定義ファイルになっている。自信があればviなどのエディタで直接編集してもOKだ。ここまでがエンジニアの初期設定の仕事だ。

■Generatorによるジョブの生成

さてこれからがエクゼクティブユーザーの出番である。

［Generator］ボタンを押して、Fileプルダウンメニューからshop.plnを読み込ませると、2100通りの組み合わせを生成してくれる（**画面7**）。

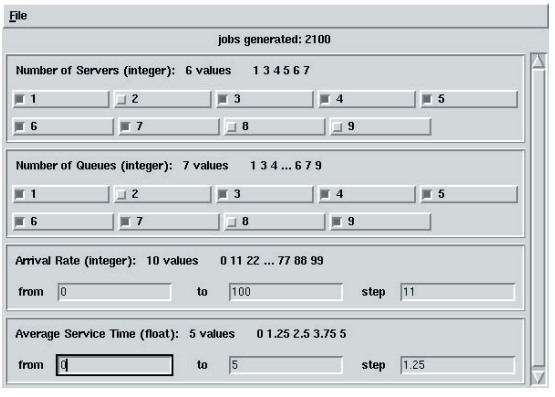
マウスを使って各種パラメータをクリックしてシミュレーション条件を変更することができる。例では顧客1人あたりの平均対応時間を変更しようとしている。初期値が0というのは現実的でないので、変更したらFileメニューで「shop.run」というファイル名で保存する。これが実際に次のDispatcherで利用される。

■Dispatcherによる実行

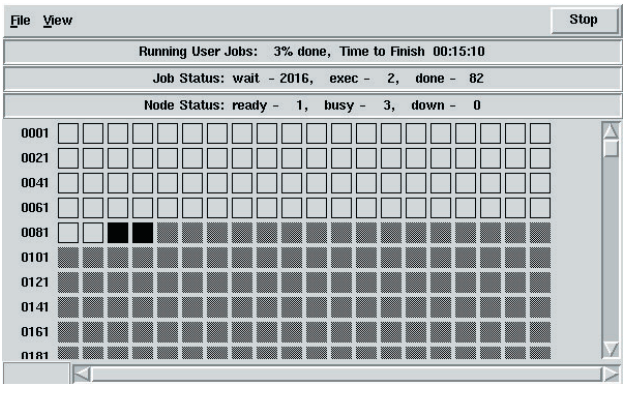
［Dispatcher］ボタンを押して、Fileメニューより「shop.run」を入力し、Startボタンを押して実行させているのが**画面8**である。2100個の並列処理で、終了したのが82個、実行中が2個、待ち状態のジョブが2016個という状態を示している。

全部終了すると標準出力先コンソールに、

```
99.295  7   3  11  1.25
```



画面7　プランファイルshop.plnから2100のジョブを生成



画面8　2100の並列ジョブを実行中のDispatcher画面

▼リスト2　enFuzionで使用する制御ファイル（プランファイル）

```
parameter nums label "Number of Servers" integer select anyof 1 2 3 4 5 6 7 8 9 default 1 3 4 5 6 7;
parameter numq label "Number of Queues" integer select anyof 1 2 3 4 5 6 7 8 9 default 1 3 4 5 6 7 9;
parameter arrivalr label "Arrival Rate" integer range from 0 to 100 step 11;
parameter servtim label "Average Service Time" float range from 0 to 5 step 1.25;
task main
        node:execute shop -q $numq -s $nums -a $arrivalr -t $servtim>output
        copy node:output output.$jobname
endtask
task rootfinish
        execute cat output.* | sort -n | tail -1; rm output.*
endtask
```

パラメータの定義
分散処理の定義
あと処理の定義

などのように結果が出てくる。このようにenFuzionを利用すれば並列処理の定義が簡単に作成でき、エンドユーザーがマウスで簡単に使えるのだ。

より進化した超並列手法

生産計画、輸送計画などの複雑なシミュレーションプログラムでは、先ほどの総当たりロジックでは現実的な要求時間内に最適解を求めることはできないし、その必要性がない場合も多い。このうち最適解が実在することは確実だが、1分、または1日とか有限で、少なくとも一生より短い時間では解けない問題を「NP完全問題」という。

この説明によく使われる例が、「巡回セールスマン問題」である。これは「ある出発地点から出発して①から⑩の都市を一度だけ訪問して出発地点に戻るのに、最も距離が短いルートや最もコストが小さいルートを決定する」という問題である。この経路の種類は、10！＝362万8800通りもあるので、1つの経路を検討するのに1秒間だとしても全部を計算するのに42日間もかかる。単純な総当たりロジックではスピーディに答えを求めることはできないのだ。

■遺伝的計算方法

次ページの夫婦の会話や兄弟の会話を持ち込んだのは、人間社会では親と子供では身体や性格などを引き継ぐ場合がある（良いところと悪いところを含めて……）。

また仲の良い兄弟の例のように、弟がわからない問題（どうやって子供が生まれるのか？）が発生しても、兄のほうが答えを知っていれば兄の年齢にならずとも短期間で知識を習得することができるのである。これを大きなレベルで見ると、人間の歴史はこのノウハウの伝授の繰り返しをやって、今日の高度に発達した文明や文化を形成しているのだ。

兄弟の会話でお兄さんがDNAの話をしたので、こちらも発想の転換をしてみよう。都市の並びを生命の遺伝子であるDNAに見立てて、進化・遺伝的な仕組みを利用して短時間で求めようというアイデアである。

これを「遺伝的アルゴリズム」といい、情報科学分野では立派な理論体系になっている。この仕組みを簡単に説明すると、

DNA＝（①、②、③、④、……⑩）

という並びのどこかに最適な組み合わせが存在するので、最初は適当に並べた異なる組み合わせの初期値を複数の各計算機に与え、個々の計算機で並列に実行させる。

初期状態の組み合わせよりも良い組み合わせが見つかれば、その結果を共有ディスクに書き込む。また近傍の計算機から問い合わせがあった場合は「こんな巡回経路があるよ」と教えることもするのだ。教えてもらった計算機側では、自分が検討している巡回経路より優れていれば採用し、そこからさらに良い方法がないか継続して調べる。

もし決められた時間になってもまったく答が見つからなければ、近傍の計算機から情報を入手して終了する。

終了したプログラムを検出すると、そのプログラムが共有ディスクに残した答を初期データとして、さらに別のプログラムを起動して、より良い巡回経路がないか調べる。これを世代交代と呼んでいる。

この各計算機の動作を注目すると、自律的に、非同期に、かつ互いに協力して問題を解決しようとしている人間活動のようにも見える。

この画期的な方法により、従来の方法よりも効率のよい並列処理ができるようになった。

日立エンジニアリングでは、このアルゴリズムを利用したサプライチェーンシステムをUNIXシステム上で開発して、1999年に財団法人ソフトウェア情報センターより、ソフトウェア・プロダクト・オブ・ザ・イヤー'99のビジネスアプリケーション部門で最優秀賞を受賞した。今年はTurboLinuxに移植して同様の成果が得られることを実証している。

Linuxに対応したことにより、ハードウェアコストやライセンスコストが下がり、システム価格の大幅な低減が可能になった。

おわりに

第1回目で書いたように半導体素子の大きさやクロックの物理的限界は、すぐそこまできている。これ以上配線パターンを小さくして基盤を小さくしていくと、やがて回路を構成する線の太さは原子に近づき「電気」という形で伝わっていくことが保証されなくなるからだ。

逆に「量子的なふるまい」が大きな問題になってくる。まだ実用的なレベルには達していないが、すでに各国で量子コンピュータの研究が進んでおり、特にIT先進国のアメリカでは活発な研究がなされている。

21世紀の初頭は、並列コンピュータが主役になって各種産業を強力に推進する主力エンジンとなるだろう。21世紀は、クラスタの時代で始まり、量子コンピュータの時代になるかもしれない。

ある夫婦の会話（10月号からの続き）

ある日の夕食にて

夫：やはり君の作った料理は最高にウマイね。

妻：あなたってお世辞がウマイのね。ところで先月のクラスタの件だけど、最低２台から構築できるの？

夫：そう。２台から構築可能だよ。

妻：それじゃ週末にあなたのPCと私のPCを使ってクラスタを構築してみせてよ。

夫：よしわかった！おれが教えたクラスタの種類の中でどのタイプのクラスタを構築しようか？

妻：２台で構築する場合、たしかフェイルオーバークラスタやスケーラビリティクラスタなどの信頼性重視クラスタを構築する場合が多いのよね。そうね、性能と信頼性が向上するスケーラブルWebクラスタを構築してみせてよ！

夫：よしわかった。次の土曜日にチャレンジしてみよう。

　ところで子供たちはどうした？　今日は食事した？

妻：いいえ宿題が終わるまで食事は与えないことにしたの。

夫：そんな……ちょっとかわいそうだよ！

妻：違うの。最近宿題をやってこない傾向があると、担任の先生から注意されたばかりなの。最近なにやら別のことに興味を示しているようなの。

夫：君だって先月パソコンに夢中になって夕食を作るのを忘れていたじゃないか！あの時は食材や米もなかったぞ！（10月号参照）

妻：それとこれとは別よ！

夫：なにか夢中になって大切なことを忘れるのは、君の性格が子供に反映されているからだよ！

妻：（プチ！）なんですって！

夫：なんだよ！やる気か？

子供部屋では……

兄：また夫婦喧嘩が始まったよ。今日は本当に夕食がないかも！

弟：あ〜あ、最初は雰囲気良かったのに……。急に悪くなっちゃった！（君たちが勉強しないのが主原因だ）

　僕はどうしてこんな家庭に生まれたの？

兄：小学校の先生から習ったのだけど、なにかモノを作る場合は設計図が必要なのだ。特に精巧なものは、多くの設計書を作成しなくてはいけないのさ。（質問の答えと少しずれているような気がするが……）

弟：ふうん？（なにやら難しそう）

兄：人間の場合も非公開の別のインターフェイスを使って、男と女の間で設計図の元になる複数のDNAという情報を交換しあうのだよ。その膨大な情報を元に赤ちゃんが作られるのさ。

　最近では、そのDNAのひとつが完全に解読されたとして大きなニュースになってたぞ。

弟：そのDNAを交換しあうためのインターフェイスはどのようになっているの？

兄：う〜ん。それは勝手に公開すると犯罪になるのだ！

弟：えっ。本当？　兄弟のあいだでも？

兄：う〜ん。日本の法律では、なぜか18才未満だと禁止されているのだ。言葉ではなかなか説明できないな……。そうだ、絵で説明しよう。WとXとYがあって……。

弟：ふ〜ん（なにやら怪しげだな）。

妻：（この場合、母）こらっ！何のマンガ書いているの？　キチンと勉強しているの？

兄：ドキッ！英単語を弟に教えていたんだよ！

妻：ふうん……（怪しいけど）。宿題は終わったの？

兄弟：まだ……。

妻：今日は特別に食事をとってもいいわよ。その代わり食べ終わったら、すぐに宿題の続きをするのよ。

兄弟：ハーイ。

兄弟：でも、パパとママはいつの間にか仲直りするのはどうして？

夫：子供（クライアント端末）にわからないように、状態が切り替わるのが優秀な夫婦（クラスタサーバ）なのだよ。

妻：ちゃんと2人で勉強してたのね。とにかくうちの兄弟は仲がいいわね。

兄弟：（大きな誤解だと思うが……）

夫：まあ、お互いに助け合って生きているのが人間というものだ。このあいだの夕食事件の時に夕食にありつけたのも私のおかげだよ。あれはいわゆる家庭におけるフェイルオーバー機能だな。

妻：まるで私が妻として機能していなかったような……。

夫：（うっ、爆発するとまずい！）いやそうではなくて、人間もなにか夢中になれば、ほかのことができなくなるのさ。

妻：そうね。そのようなときに助け合っていくのが大事なのよ。

兄：ヒソヒソ（さっきは、2人とも何か大事なことを忘れていたような……。このクラスタ夫婦は、イマイチ信頼性がないような気がする。ま、ここは黙って食事するのが安全だ）

弟：（うん）

Kondara MNU/Linux Web Cluster 2000の概要と動作

文：市村元信　デジタルファクトリ株式会社IT事業部
Text : Motonobu Ichimura　IT Depertment, Digital Factory Co.,Ltd.

インターネット人口の増加から、最近のビジネスはその場をインターネットに移しつつある。製品情報の提供からオンラインショッピングまで、Webベースのサービス提供が欠かせないものになっている。

こうした中、サーバダウンや負荷増大に関わらず、常に高品質なサービスを提供し続けることが求められるようになってきた。

ここでは、こうした要求に応えるために9月にデジタルファクトリが発売した、Kondara MNU/Linux Web Cluster 2000（以下Kondara Web Cluster）の概要と運用について解説する。

Kondara Web Clusterは、弊社が開発した単機能サーバ専用ディストリビューションの第1弾であり、Webクラスタ用途に特化したため、簡単なインストールと運用を実現している。

動作環境として、PC/AT互換機のほかに、Alphaシステムにも対応する。

まずKondara Web Clusterの特徴を簡単に示そう。

- **フェイルオーバー**
- **負荷分散**
- **障害通知**
- **Webブラウザによるリモート管理**
- **インストールするだけの簡単設定**
- **コンテンツの同期**
- **各ノードが正常に動作するまでサポート**

このうち、フェイルオーバー／負荷分散／障害通知／リモート管理／簡単設定は、kwcsd（Kondara Web Cluster Server Daemon＝サーバデーモン）により実現され、コンテンツの同期はサーバデーモンとrsyncの協調により実現される。

そして、クライアント（インターネット）からのWebアクセスは、クラスタを構成する実際のマシンIPアドレスではなく仮想サーバのバーチャルIPに対して行う。なお、仮想サーバは親ノードがIPエイリアスによって実現している。この仕組みをKondara WebClusterではV-RBS（仮想負荷分散システム）と呼んでいる。

また、上記の機能を実現するために、次のような制限がある。

- **Webクラスタを構成できるのはKondara Web Clusterがインストールされたものに限定される。**
- **Webクラスタは同一ネットワーク上に構成されていなければならない。**

kwcsdサーバデーモンの概要

kwcsdサーバデーモンは主に、kwcscc（Kondara Web Cluster Server Control Client＝コントロールクライアント）とkwcscs（Kondara Web Cluster Server Control Server＝コントロー

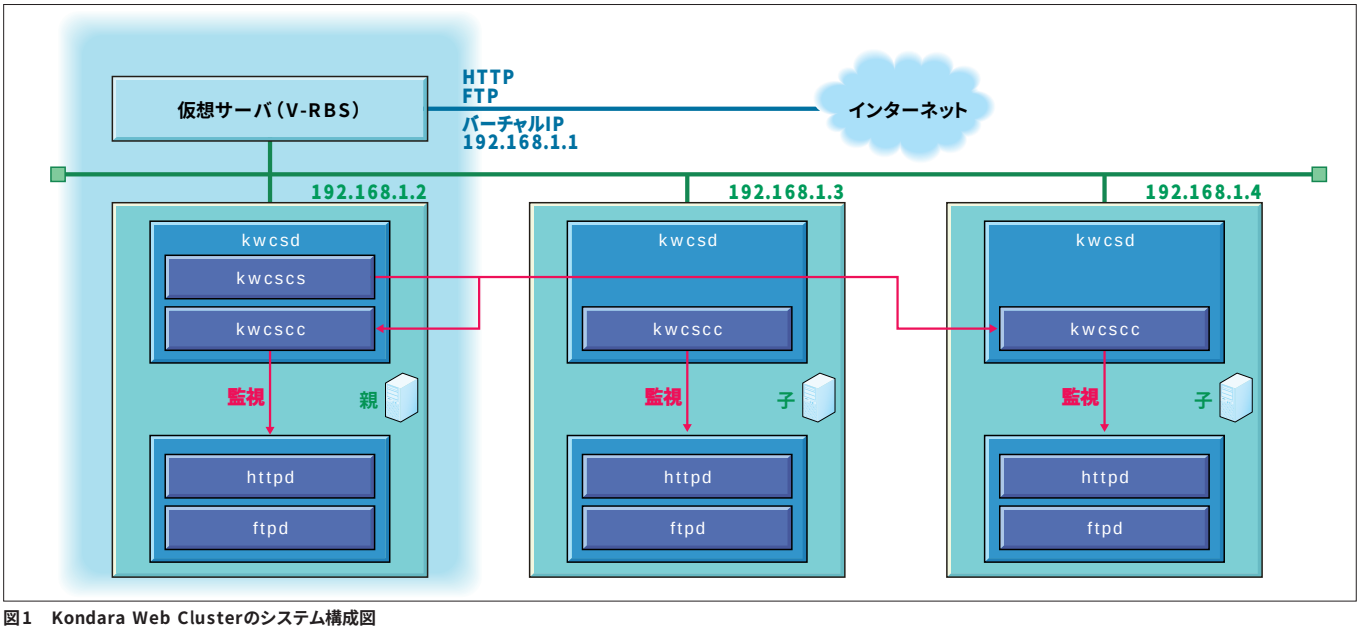


図1　Kondara Web Clusterのシステム構成図
仮想サーバは、親ノードがIPエイリアスで実現している。クライアントからは、仮想サーバにWeb／FTPアクセスする。

ルサーバ）の、2つの機能を持ったプロセスによって構成される。

■kwcscc（コントロールクライアント）

コントロールクライアントは、自ノードのHTTPサーバの状態やディスクの残容量を監視する。もし、ディスクがあふれてしまったり、HTTPサーバが停止した場合には、kwcp（Kondara Web cluster Control Protocol）メッセージをコントロールサーバに送信する。

また、コントロールクライアントはコントロールサーバが動作中であるかどうかも監視しており、コントロールサーバが停止状態に遷移した場合は、コントロールサーバプロセスを生成して自分が親ノードとなり、Webクラスタの監視体制に入る。

■kwcscs（コントロールサーバ）

コントロールサーバは各ノードのコントロールクライアントの状態を監視しており、コントロールクライアントからの応答がなくなった場合に、そのノードをWebクラスタの構成から外す。また、バーチャルIPのHTTPポートへのリクエストを各ノードにフォワードする役目も行う。

なお、60秒に1回、/home/httpdディレクトリ以下のデータの同期が行われる。このディレクトリは管理ツール（**画面1**、**画面2**）によって/home以下に変更することができる。そのほかWebサーバコンテンツなどの更新をFTPで行う場合に、FTPポートを監視することで自動的にデータの同期を行うように設定することも可能だ。

kwcsdサーバデーモンの動作

サーバデーモンが起動すると、まずはコントロールクライアントプロセスとなり、起動したサーバ上のHTTPサーバ、ディスク容量を監視するプロセスとなる。その後、kwcpメッセージを発信し、同一サブネット内にすでに親ノード状態（コントロールサーバ）になっているサーバデーモンが存在するかどうか問い合わせを行う。

もし、親ノードが存在している場合は、親ノードからWebクラスタIDを含んだkwcpメッセージが応答される。このWebクラスタIDが自分の持つWebクラスタIDと同一のものであれば、そのWebクラスタに子ノードとして参加する。

また、親ノードからメッセージ応答がない場合、または応答があってもWebクラスタIDが違う場合は、サーバデーモンはコントロールサーバプロセスを新たに生成し、親ノードとしてWebクラスタを構成し、監視状態に入る。

バーチャルIPは、管理ツールによって設定するか、/etc/kwcsd.confファイルを修正し、kwcsdを再起動することで変更することができる。

なお、kwcpメッセージのサイズは可変長で、**リスト1**のように定義されている。

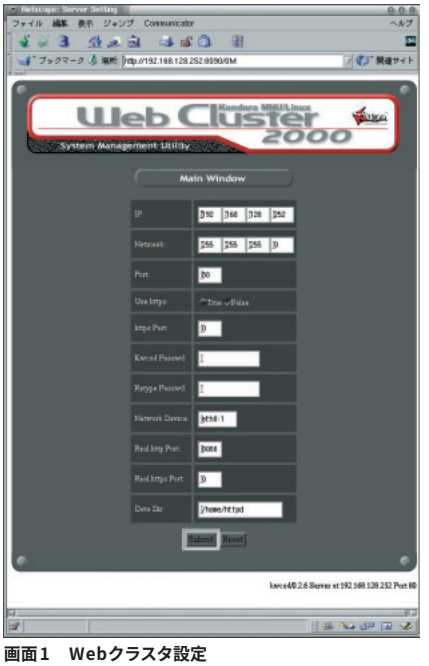
Kondara Web Clusterのインストール

Kondara Web Clusterのインストールは、通常のKondara MNU/Linuxのインストーラを利用しているので、今までにKondara MNU/Linuxのインストールをしたことがあれば非常に簡単である。Kondara Web Clusterのインストーラでは、ServerタイプかCustomタイプのみ選択できるようになっている。通常はServerタイプを選択すればよいだろう。

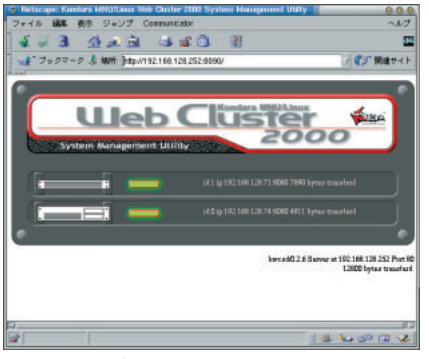
実際にWebクラスタを構築するために必要な作業はこれだけである。インストールが済んでしまえば、自動でWebクラスタを構成してしまうので、あとは2台、3台とインストールしていくだけで、2台構成、3台構成のWebクラスタが構築されることになる。

■簡単な理由

Kondara Web Clusterでは、容易にWebクラスタを構成できるようにするため、さまざまな工夫をこらしている。前述のとおり、サーバデーモンは起動時にkwcpメッセージをブロードキ



画面1　Webクラスタ設定
クラスタ設定をWebベースのシステム設定ツールで変更できる。



画面2　ステータスページ
ノードのステータスを確認することができる。

ャストする。このとき、サーバデーモンは同一ネットワーク上で利用されていないIPアドレスを自動で検出し、特に指定がされていなければ自動的にそのIPアドレスを利用してWebクラスタを構成する。このため、親ノードのIPアドレスを設定しないで、Webクラスタを構築できるのである。

もちろん、Webクラスタを構築してから、Webベースの管理ツールを利用して、親ノードのIPアドレスも変更できる。

もし、親ノードが存在していた場合は、kwcpメッセージをブロードキャストすると親ノードのIPアドレスを含んだkwcpメッセージが返ってくるため、この情報を利用して既存のWebクラスタに子ノードとして参加する。

また、これらの情報はサーバデーモンが自動的に設定ファイルに書き込むため、ここでも特に設定をする必要はない。Webクラスタに参加してしまえば、自動的にデータの同期も行ってくれる。

▼リスト1　Kondara Web cluster Control Protcolのフォーマット

```
typedef struct _kwcp_pack{
        kwcp_msg msg;
        guint8 domain;
        guint32 size;
        union {
                guint32 v4;
                guint32 v6[4];
        }server_ip;
        union {
                guint32 v4;
                guint32 v6[4];
        }client_ip;
        guint32 data;
}kwcp_pack;

typedef struct _join_pack{
        kwcp_pack kwcp;
        guint8 server_mode;
        guint16 port;
        guint16 sport;
        gchar data_path[PATH_MAX];
        gchar net_dev[15];
}join_pack;

typedef enum _kwcp_msg{
        OK,
        Fail,
        Hello,
        HowAreYou,
        Fine,
        Join,
        KillMe,
        WebForward,
        ClusterSetting,
        NodeSetting,
        NodeOn,
        NodeOff
}kwcp_msg;
```

送受信するメッセージ
INET domainかINET6 domainかを表す
このkwcpメッセージの大きさ
マスタのIPアドレス
ノードのIPアドレス
データ領域
data領域の内容の例
msgが認識するメッセージ

ユーザー事例

弊社は、Kondara Web Clusterを始めとした各種Linuxソフトウェアの開発を始め、Linuxを中心としたシステムインテグレーション事業にも取り組んでいる。

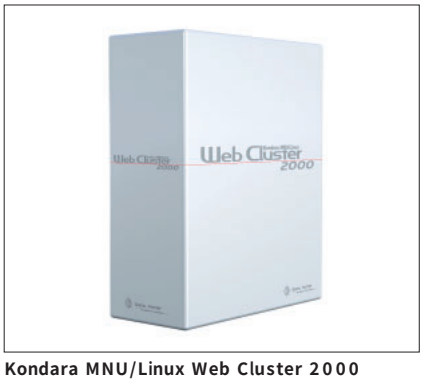
Kondara Web Clusterは、株式会社ウェブマスターが運営している不動産賃貸物件情報サイト「物件くん」に採用され、稼働している。

「物件くん」では、関西地区を中心に約7万件におよぶ賃貸物件の情報を提供しており、データベースに登録されている賃貸物件を、管理する各会社が保有する自社物件情報を随時更新できる機能を備えているので、迅速で正確な情報提供が可能になっている。

今後ますますWebクラスタを利用したシステムインテグレーションが増えてくると予想しており、さまざまなソリューションを展開していく予定だ。

■入手方法

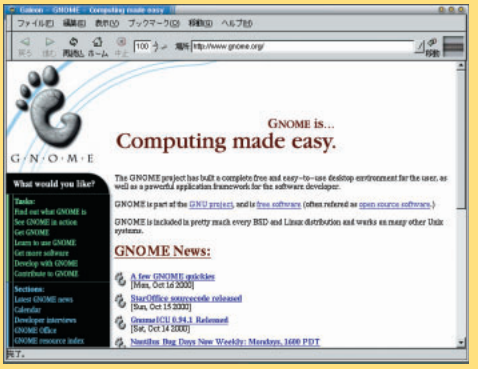
Kondara MNU/Linux Web Cluster 2000の入手方法は、http://www.digitalfactory.co.jp/から購入するか、またはPCショップなどで予約購入することができる。



Kondara MNU/Linux Web Cluster 2000
価格　12万8000円（2ノード版）
　　24万8000円（ノード数無制限版）
デジタルファクトリ株式会社
http://www.digitalfactory.co.jp/

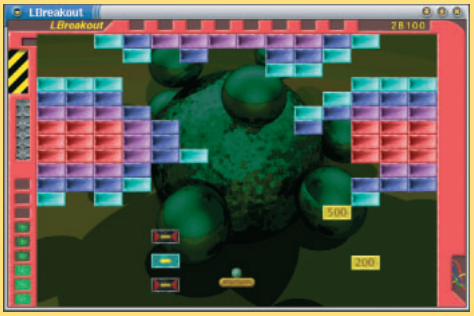
Free Application Showcase

文：出井 一
Text : Hajime Dei



Galeon　P.134

Tux Racer　P.143



LBreakout　P.145

※紹介したソフトは、すべて付録CD-ROMに収録されています。

GUIによるCD-R作成統合環境

gcombust

バージョン：0.1.36　　ライセンス：GPL

http://nemo.sby.abo.fi/~jm/gcombust/
http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html (cdrecord)
http://www.red-bean.com/~bwf/software/cdlabelgen/(cdlabelgen)

gcombustは、CD-Rの作成をGTK＋ベースのGUI環境で行うソフトだ。cdrecordやmkisofs、cdlabelgenなどのフロントエンドとして動作し、ISOイメージの生成やWAVEファイルの吸出し、CD-Rへの書き込み、ケース用のラベルの作成などをボタンやリストの操作だけで行える。付属の日本語カタログにより、ボタンや説明が日本語表示される点もうれしい。実行にはcdrecordなどのツール類が別途必要だ。

●ビルドとインストール

(1) ツール類のインストール

まず、作成関係のツール（cdrecord／mkisofs／cdda2wav）をインストールする。最新版1.9のRPMパッケージがVinePlus2.0に含まれているので、Red Hat系ディストリビューションではこれを利用するとよい。tarボールの場合は、DEFAULTS/Defaults.linuxの30行目を「INS_BASE= /usr/local」に変更後、「./Gmake.linux」→「make install」とすればOKだ。

CDケース用のラベルを生成するcdlabelgenは、RPMパッケージとtarボールが配布されている。実体はPerlスクリプトなので、tarボールの場合もビルドの必要はなく、「make install」とするだけでいい。

(2) CD-Rドライブの準備

最近のSCSI、ATAPI接続のCD-Rドライブのほとんどは MMCに準拠しているので、Linuxで動作すると思って差し支えない。

ATAPI接続のCD-Rドライブの場合は、カーネルのSCSIエミュレーションサポートなどを有効にしてカーネル再構築（あるいはモジュール作成）を行う必要がある。作業の詳細は、本誌先月号の特集「周辺機器バトルロイヤル」の52ページや、「CD-Writing HOWTO」（http://www.linux.or.jp/JF/JFdocs/CD-Writing-HOWTO.html）の第2章を参照されたい。

(3) gcombustのインストール

gcombustの最新版0.1.36は、現時点では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。/usr/local以下に実行ファイルやカタログファイルがインストールされる。

●一般ユーザーで起動するには

cdrecordを利用してCD-Rに書き込むには、スーパーユーザー権限が必要だ。一般ユーザーのまま書き込みを行うには、rootになった状態で、

```
# chmod 4111 $(which cdrecord)
```

などとして、cdrecordにsetuidビットを設定する必要がある。

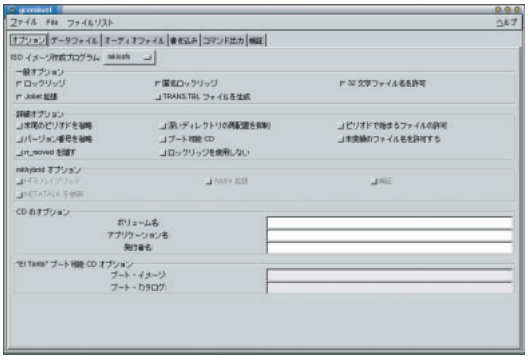
ただし、ネットワークに接続されたマシンでは、セキュリティ上のリスクになる。心配なら、「su -」としてrootになってからgcombustを起動するとよいだろう。

「gcombust&」として起動すると、複数のページで構成されたウィンドウが開く（**画面1**）。日本語カタログのおかげで、メニューやボタンなどすべて日本語表示だ。

●初期設定

初めて起動した場合は、設定ダイアログが自動的に開くので、以下の設定を行っておこう。

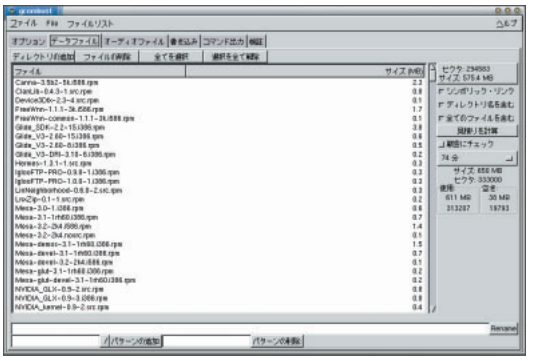
[プログラムのパス]ページ（**画面2**）では、使用するツール（cdrecordなど）のパスを設定する。初回起動時にgcombustが自動的に検索してくれるが、見つからないツールは「NOT_FOUND」になる。このままだと、そのツールに関連する機能をgcombustで利用できないので注意されたい。



画面1
gcombustでは、GUIによる操作で簡単にCD-Rの作成を行える。

画面2
gcombustから起動される各種ツールのパスを設定する。

画面3
CD-Rドライブのデバイスの指定や書き込み速度の設定を行う。



画面4
データCD作成用のディレクトリやファイルをリストに登録。

[デバイス]ページ（**画面3**）では、書き込みに使うCD-Rドライブを指定する。デバイスを特定する3つの数値がわからなくても、[SCSIドライブの検出]ボタンで自動検出できる。複数のドライブを接続している場合は、ボタンを押すたびに各ドライブが順番に検出されるので、CD-Rドライブかどうかを[SCSI設定の確認]ボタンで確認するといい。CD-Rの書き込み速度もここで設定しておこう。

●オプションの設定

gcombustでの作業は、ウィンドウ上部のタブを左から右へと切り替えながら行う。各ページの項目の意味がわからない場合は、項目上にカーソルを置くと表示されるチップヘルプを参考にしよう。

一番左の[オプション]ページでは、CD-Rの作成に関するオプションを設定する。これらの設定は、mkisofs（またはmkhybrid）起動時のコマンドラインオプションに反映される。

通常は初期設定を変更する必要はなく、[CDのオプション]の各項目（ボリューム名など）を設定するだけで十分だ。このほか、Macでも読めるハイブリッドCDや、直接ブート可能なCDも作成できる。

●データファイルの登録

データCDを作成する場合は、[データファイル]ページのリストに、ISOイメージの元になるディレクトリやファイルを登録する（**画面4**）。ISOイメージをディスク上に作成しないで直接書き込むオン・ザ・フライ方式でも、このリストが使われる。

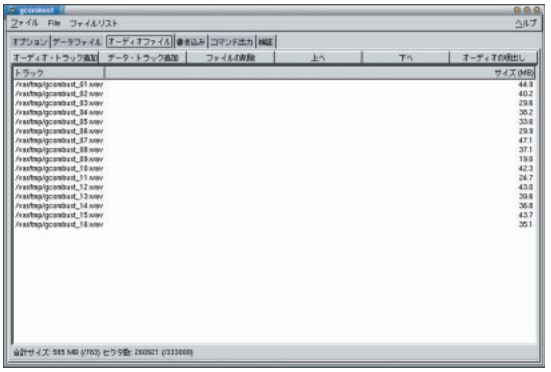
[ディレクトリの追加]ボタンを押してダイアログを開き、ディレクトリやファイルを選択する（複数選択可）。なお、ディレクトリを登録した場合は、その下のファイルやサブディレクトリもあわせて登録されるが、リストには表示されないので注意しよう。

登録後は、ディレクトリやファイルの名前を変更したり、リストから削除することが可能だ。ファイルの合計サイズがCD-Rの容量以下になるように、除外するファイルを自動的に選択する機能も用意されている。

●オーディオファイルの登録

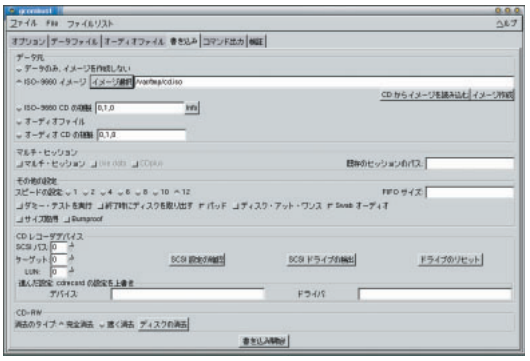
音楽CDを作成する場合は、[オーディオファイル]ページのリストに、トラックごとのWAVEファイルを登録する（**画面5**）。なお、CD-Rドライブとは別にCD-ROMドライブがある場合は、この操作を行わずに、直接コピーすることもできる。

音楽CDからWAVEファイルを吸い出す（リップする）には、右上の[オーディオの吸出し]ボタンを押して、ダイアログを開く（**画面6**）。音楽CDを入れたドライブや保存先ディレクトリなどを指定して[吸出し]ボタンを押すと、cdda2wavが実行され、トラックごとに「gcombust_XX.wav」（XXはトラ

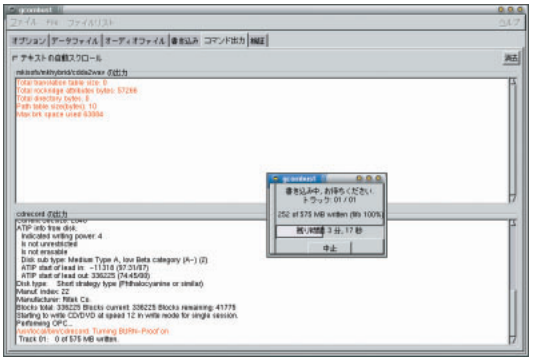


画面5　音楽CD用のトラックごとのWAVEファイルを登録する。

画面6　cdda2wavで音楽CDからWAVEファイルをリップする。



画面7
データ元などの設定を行った後、実際に書き込みを行う。



画面8
作成中は、各コマンドの出力などが表示される。

ック番号）というファイル名で保存され、自動的にリストに登録される。

登録後は、WAVEファイルの順番変更や削除が可能だ。また、データトラックを先頭に持つ「ミックスモード」のCDも作成できる。

●CD-Rに書き込みを行う

データファイルやオーディオファイルの登録が完了したら、[書き込み]ページに切り替え、以下のようにデータ元などを選択する（**画面7**）。

（1）データCDを作成する場合

いったんISOイメージをディスク上に作成する場合や、すでにISOイメージがディスク上にある場合は、[ISO9660イメージ]をチェックし、イメージファイル名を指定する。

ISOイメージを作成するには、ファイル名を入力して[イメージ作成]ボタンを押せばいい。mkisofs／mkhybridが起動され、[データファイル]ページのリストの内容のISOイメージファイルが作成される。

最近のマシンなら、ISOイメージファイルを作らずに直接書き込むことも可能だ（オン・ザ・フライ）。その場合は、[データのみ、イメージを作成しない]をチェックすればいい。CD-R書き込み時にmkisofs／mkhybridが起動され、ISOイメージが直接cdrecordに渡される。

（2）音楽CDを作成する場合

[オーディオファイル]ページのWAVEファイルのリストに基づいて音楽CDを作成する場合は、[オーディオファイル]をチェックする。

CD-ROMドライブが別にあるなら、音楽CDの内容を直接コピーすることも可能だ。[オーディオCDの複製]をチェックし、CD-ROMドライブのデバイスを指定すればいい。

なお、どちらの方法もトラックごとのコピーのため、曲間のギャップの長さは再現されない。ギャップも含めて音楽CDを正確にコピーしたいなら、cdrdaoを利用するとよいだろう。

このほか、マルチセッションやダミーテスト、ディスクアットワンス、BURN-proofの有効・無効などを設定できる。CD-Rの書き込み速度やデバイスをここで変更することも可能だ。

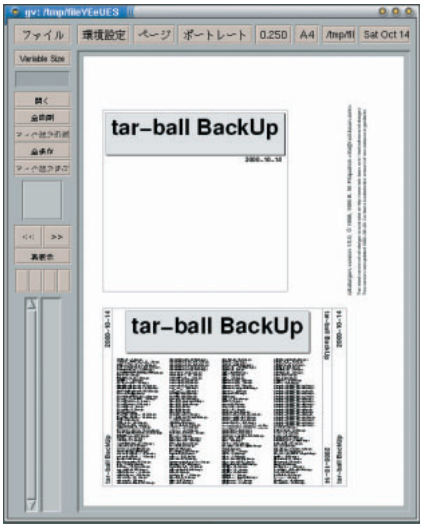
すべての設定を終えたのち、一番下の[書き込み開始]ボタンを押すと、[コマンド出力]ページに切り替わって、mkisofs／mkhybridやcdrecordの出力が表示され（**画面8**）、進行状況を示すウィンドウが開く。「正常に終了しました！」というメッセージが表示されれば書き込み作業は終了だ。

●CDのラベルを作成する

最後に、[ファイル]－[表紙の印刷]を選択して、cdlabelgenを利用したCDケースのラベル作成を行う。こちらも、他のツールと同様にGUIによる設定が可能だ。

表示項目として、データトラック・オーディオトラック・ディレクトリの一覧を取り込める。また、カテゴリ・サブカテゴリの設定、日付や飾り枠の有無、EPS形式の画像の貼り付けなどが可能だ。作成したラベルは、PS形式でファイルに保存できるほか、gvによるプレビュー（**画面9**）や、lpr経由での印刷なども行える。

cdlabelgenによるCDラベルは表示形式が固定されているため、さらに凝ったラベルを作成したいなら、本誌2000年10月号で紹介した「KCDLabel」などのCDラベル作成ソフトを別途利用するとよいだろう。



画面9
ファイルの一覧などを取り込んでCDのラベルを作成できる。

Gecko利用の軽量Webブラウザ

Galeon

バージョン：0.7.7　　ライセンス：GPL

http://galeon.sourceforge.net/
http://www.mozilla.org/ （Mozilla）
http://people.redhat.com/blizzard/software/RH6/RPMS/（Mozilla RPM）

Galeonは、Mozillaプロジェクトで開発されたGeckoエンジンを利用した、GNOME 1.2用のWebブラウザだ。メールクライアントなどを含むMozillaに対し、Webブラウザに特化したGaleonは軽量で、メニュー表示なども高速だ。Geckoによる表示はスタイルシートへの対応に優れ、日本語のWebページも問題なく表示できる。また、付属の日本語カタログにより、メニューなどはすべて日本語で表示される。

●ビルドとインストール

Galeonの実行には、GNOME 1.2のほか、libxml 1.8以降、libglade 0.13以降などのライブラリが必要だ。また、Geckoエンジンは単独配布されていないため、Mozilla M18も導入する必要がある（libxml、libgladeの最新版は、付録CD-ROMのGNOMEディレクトリに収録されている）

Mozilla M18のRPMパッケージがRed Hatのサイト（上記URLを参照）に用意されているので、Red Hat系ディストリビューションではこれを利用するとよいだろう（**画面1**）。

Galeon自体は、tarボールとRPMパッケージの両方が配布されているので、ディストリビューションに合った方を利用しよう。tarボールからのビルドとインストールも、「./configure」→「make」→「make install」という一般的な手順でOKだ（ビルド時のエラーについては付属のFAQを参照）。

●起動と初期設定

「galeon&」として起動すると、シンプルなウィンドウが開いて、GNOMEプロジェクトのWebページが表示される（**画面2**）。

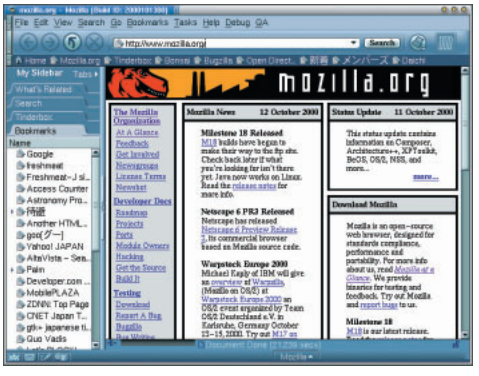
初めて起動したときには、Netscapeのブックマークをインポートするか質問され、[はい]を選択するとブックマークが取り込まれる。

まずは、[編集]－[設定]で設定ダイアログを開こう（**画面3**）。設定項目はジャンル別のツリーで分類されており、説明も日本語なので戸惑うことはないだろう。

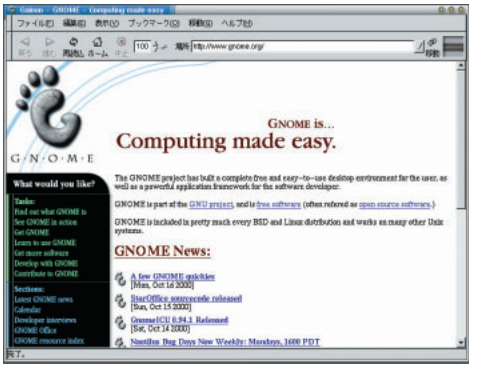
日本語環境では、[レンダリング]－[言語]ページの設定で、[キャラクタセット]の[自動検出]を「日本語」に、[デフォルトキャラクタセット]を「日本語（EUC-JP）」などに変更しておけばいい。

●使い心地は快適

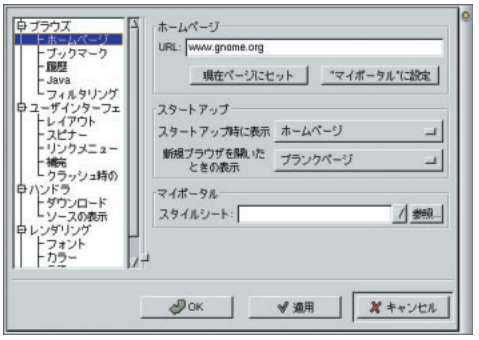
ブラウザとしての使い心地は、NetscapeやMozillaとほとんど同じだ。左ボタンのクリックでリンク先への切り替え、中ボタンのクリックで新しいウィンドウを開いてリンク先を表示、右ボタンクリックでコンテキストメニ



画面1
メール機能なども含んだMozilla M18。これはブラウザウィンドウ。

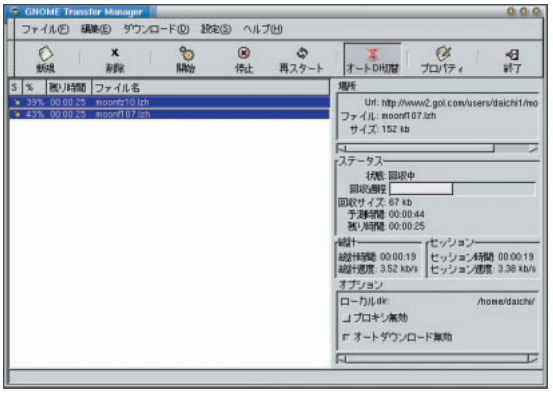


画面2
GaleonでGNOMEプロジェクトのWebページを表示している。

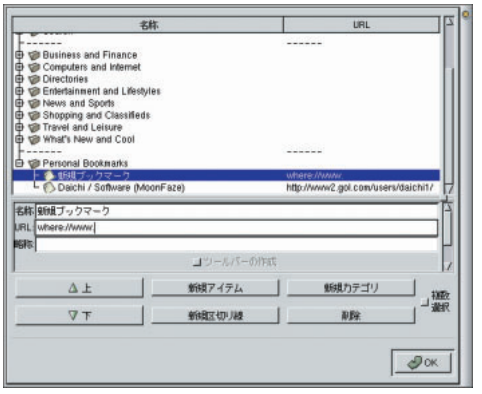


画面3
ジャンル別のツリーで分類された分かりやすい設定ダイアログ。

画面4
Geckoエンジンによる Webページのレンダリングは優秀だ。



画面5
GTransferManagerと連動してファイルのダウンロードを行える。



画面6
自由に順番を変えられる永続ブックマークの編集画面。

ューがポップアップする。また、ホイール付きのマウスでは、特に設定を行うことなくホイールの回転によるスクロールが可能だ。

Geckoエンジンによる表示はNetscapeより高速で、スタイルシート（CSS）への対応に優れている。Netscape 4.xでは見るも無惨な結果になったフロート指定による文字の回り込みや、フォントサイズの指定も正しく反映される（**画面4**）。なお、ツールバー中央部のテキストボックスとボタンにより文字サイズの変更が可能だ。

ファイルのダウンロードはGaleon自身で行えるほか、右クリックメニューの[ダウンロードファイル]を選択すると、本誌2000年10月号で紹介した「GTransferManager」にファイルのURLが渡され、複数のファイルを効率よくダウンロードできる（**画面5**）。

●ブックマークの管理

Galeonのブックマークは、仮想フォルダによる階層構造で分類されており、[ブックマーク]メニューから直接選択できる。NetscapeやMozillaなどからインポートしたり、編集ウィンドウで順番や名前、URLなどを変更することも可能だ（**画面6**）。

また、こうしたブックマーク（永続ブックマーク）とは別に、一時的にURLを登録・管理できる「一時ブックマーク」が用意されている（**画面7**）。ユーザーが追加したブックマークはいったんここに登録され、その中で指定したものだけが永続ブックマークに変換される。このため、ブックマークの数が爆発的に増えてしまい、肝心のブックマークが見つからないといった事態を避けられるわけだ。

一時ブックマークの名称変更や削除、永続ブックマークへの変換、分類用カテゴリ（フォルダ）の作成などは、すべてボタンで操作できる。

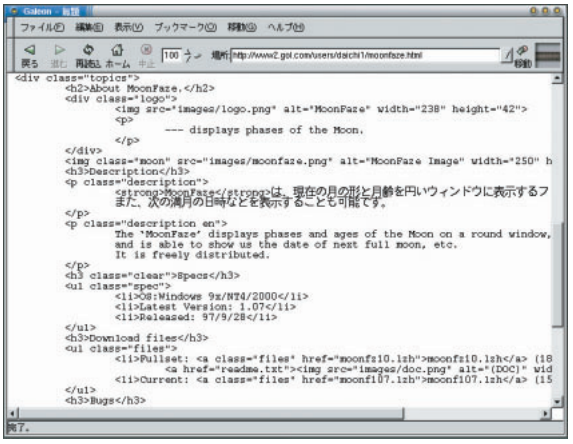
また、左右の三角ボタンで前後のブックマークのページに切り替わるので、頻繁にブラウズするURLを登録しておいて、巡回に利用するという使い方も可能だ。

●Webページのソースを表示する

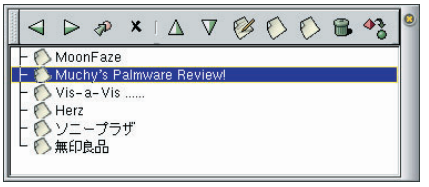
ブラウズ中に[表示]−[ソースの表示]を選択すると、そのWebページのHTMLソースが新しいウィンドウに表示される（**画面8**）。なお、M18からはソースの色分け表示が廃止された。

HTMLソースをエディタで編集したいなら、Galeonの設定ダイアログの[ハンドラ]−[ソースの表示]ページで、[ソースの表示には外部プログラムを使う]をチェックし、外部プログラムとしてgEditやXEmacsなどの好みのエディタを選択すればいい。

以後、[表示]−[ソースの表示]を選択すると、指定したエディタが起動してHTMLソースを読み込むようになる。この状態でも、[表示]−[ソース表示モード]をチェックすると、GaleonによりHTMLソースが表示される。



画面8　HTMLソースを表示する。外部エディタに渡すことも可能だ。



画面7　一時ブックマークを利用してWebページを巡回することもできる。

高機能でフリーなFTPクライアント

DeadFTP

バージョン：0.0.9　　ライセンス：GPL

http://deadftp.sourceforge.net/

DeadFTPは、高機能で使いやすいFTPクライアントソフトだ。7月にファーストリリースされたばかりだが、ローカルとリモートのファイル一覧表示や、FTPサイトを管理するホストマネージャ、転送するファイルを登録するキュー、クリップボードなどにコピーされるURLの監視など、定番ソフトのIglooFTP Proに見劣りしない機能を備えつつある。実行にはGNOME 1.2とlibxml 1.8.9以降が必要だ。

●ビルドから起動まで

DeadFTPは、tarボールとRPMパッケージが両方とも配布されているので、ディストリビューションに合わせて選択しよう。tarボールからのインストールは、「./configure」→「make」→「make install」という一般的な手順でOKだ。

ktermなどのコマンドラインから「deadftp&」として起動すると、4つの領域（ペイン）に分割されたウィンドウが開く（**画面1**）。日本語カタログは付属しないものの、GNOME共通のメニュー項目（「ファイル」など）は日本語で表示される。

操作の中心となるのは、上部2つのペインで、左上にはローカル側のファイル一覧、右上にはFTPサイトの一覧やリモート（接続先のFTPサイト）側のファイル一覧が表示される。各ペインのサイズは自由に変更可能だ。

●ホストマネージャへの登録

FTPサイトに接続していない状態では、右上のペインにはFTPサイトのブックマーク（ホスト）が表示される。仮想フォルダを利用した階層型の分類管理が可能だ。

初期設定では、ホストはひとつも登録されておらず、ルートに相当する「General」フォルダしか存在しない。頻繁に利用するFTPサイトを登録しておくとよいだろう。

ツールバーの[Hosts]ボタンを押すと、ホストの登録や削除、内容の変更などを行うホストマネージャのウィンドウが開いて、ツリーが左側に、それぞれのホストの情報が右側に表示される（**画面2**）。

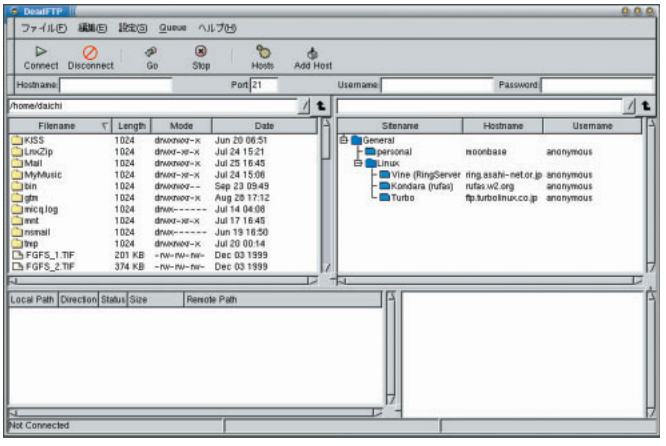
ホストを登録するには、格納先のフォルダを選択した後、右側の[Host information]ページに、FTPサイト名（日本語不可）、IPアドレスまたはホスト名、ユーザー名、パスワード、ローカル・リモートの初期ディレクトリなどを設定する。

なお、だれでも接続可能なアノニマスFTPサイトでは、[Anonymous]をチェックして、ユーザー名とパスワードの設定を省略する。この場合、（慣習としてメールアドレスを記述する）パスワードには「deadftp@sourceforge.net」が使われる。自分のメールアドレスを記述したいなら、[Anonymous]のチェックを外し、自分でユーザー名（anonymous）とパスワード（メールアドレス）を設定しよう。

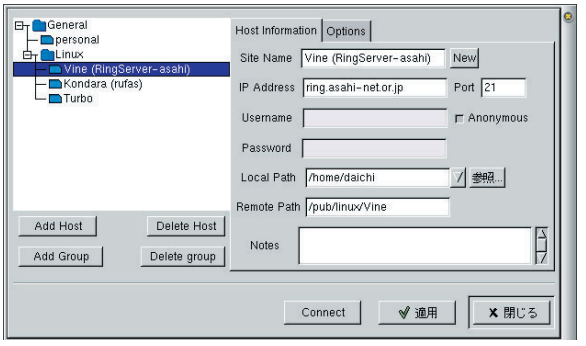
最後に、左下の[Add Host]ボタンを押すと、FTPサイト名のホストがツリーに追加される。ほかのボタンにより、登録したホストの削除や、仮想フォルダの作成・削除も可能だ。

●接続と基本的な操作

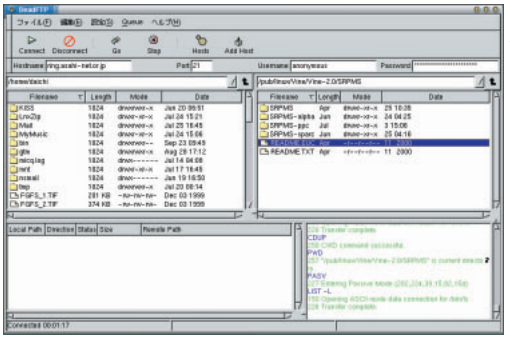
ホストを登録したFTPサイトについては、メインウィンドウのツリーでホストをダブルクリックするか、ホストマネージャのウィンドウで[Connect]ボタンを押すだけで接続処理が行われる。また、一時的なアクセスなら、ホスト名やアカウントなどをツールバーの下の「クイックコネクトバー」に入力し、



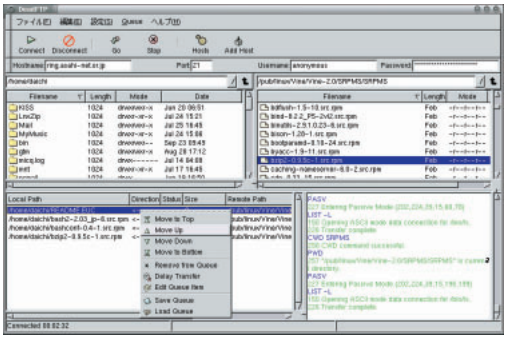
画面1
ファイル一覧をはじめ、さまざまな情報が表示されるウィンドウ。



画面2　FTPサイトの分類管理を行うホストマネージャ。



画面3
左右のペインにローカル・リモートのファイル一覧を同時表示。



画面4
送受信したいファイルをキューに登録する。順番の変更も可能。

ツールバー左端の[Connect]ボタンを押せばいい。

いずれにせよ、接続情報が右下のコンソールに表示され、無事に接続すると、右上のペインがリモートのファイル一覧に切り替わる。つまり、ローカルとリモートのファイルを同時に参照できるわけだ（**画面3**）。

どちらも、サブディレクトリのダブルクリックやディレクトリ名の直接入力でディレクトリを切り替えられる。また、ひとつ上のディレクトリに移動するボタンも用意されている。

ファイルを選択するには、マウスでクリックすればいい。Shift／Ctrl－クリックによる範囲選択や複数選択にも対応している。

●キューを利用したファイル転送

DeadFTPでは、送受信するファイルの情報を、いったん「キュー」に溜めておき、ユーザーが転送を指定した時点でまとめて一括転送する方法を採用している。

ローカルやリモートのファイル一覧でファイルを選択した後、右クリックメニューの[Add to Queue]を選択すると、それらが左下のキューペインに追加される（**画面4**）。

なお、同名のファイルがすでに転送先に存在する場合は、追加の際にダイアログが表示されるので、「上書き」または「リジューム」を選択しよう。ファイル名を変えて受信するには、「上書き」を選択してキューに追加した後、右クリックメニューの[Edit Queue Item]により開くダイアログ（**画面5**）で、[Local filename]の内容を修正すればいい。

ツールバーの[Go]ボタンを押すと、キューペイン上部のファイルから順にファイル転送が開始される。順番の変更や削除も可能だ。

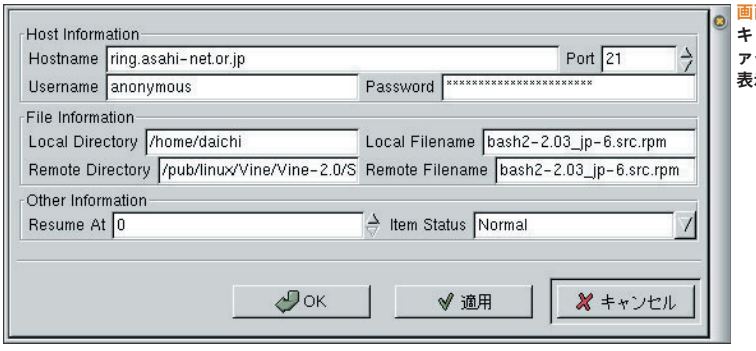
なお、キューの内容はFTPサイトとの接続を終了しても削除されず、[Go]ボタンを押した時点でFTPサイトに再接続される。このため、複数のFTPサイトを巡回して、興味のあるファイルを登録しておき、後でまとめて受信するという使い方が可能だ。

●クリップボードの監視も可能

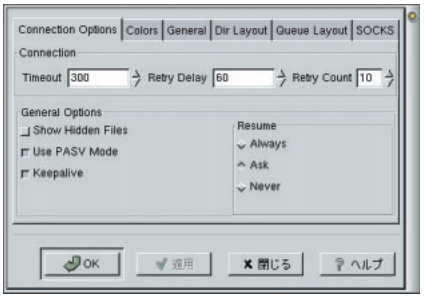
DeadFTPには、クリップボードやセレクションを監視して、FTPプロトコルのURLがコピーされると自動的にキューに追加する「モニター」機能が用意されている。

たとえば、DeadFTPを実行中に、GaleonでFTPへのリンクを右クリックし、[リンクの場所をコピー]を選択すると、そのリンクのURLで指定されたファイルが自動的にDeadFTPのキューに登録される（受信開始は人間が指示する必要がある）。

この機能は、メニューの[設定]－[設定]で開くダイアログ（**画面6**）の[General]ページで設定できる。初期値は、クリックプボード監視がオン、セレクション監視がオフなので、必要に応じて設定を変更しておこう。



画面5
キューに登録されたファイルの情報は個別に表示・修正できる。



画面6　クリップボードの監視などの設定は設定ダイアログで行う。

覚え書き（Todo）をツリーで管理

yank

バージョン：0.1.5　　ライセンス：QPL

http://home.ins.de/~m.hussmann/software/yank/

yankは、覚え書き（Todo）などのノートやメモをツリー管理するソフトだ。テキスト・チェック・Todoの3種類のノートを扱える。付属の日本語カタログによりメニューなどが日本語表示され、ノートのタイトルや内容にも日本語を利用可能だ。このほか、正規表現を利用した検索やプラグインによる拡張、gnome-printライブラリによる印刷にも対応している。動作にはGNOMEとlibxml 1.8.0以降が必要だ。

●ビルドとインストール

yankは、ソース一式をtar＋bzip2したtarボールのみ配布されている。ビルドとインストールは、「./configure」→「make」→「make install」という一般的な手順だ。

また、tarボールにはspecファイルが含まれているので、RPMパッケージを作成することもできる。

```
$ bzip2 -dc yank-0.1.5.tar.bz2 |
gzip - > yank-0.1.5.tar.gz
```

としていったんtar＋gzipの形式に変換してから、

```
$ rpm -ta yank-0.1.5.tar.gz
```

とすればいい。

●3種類のノートを使い分けよう

「yank&」として起動すると、左右2つのペイン（領域）で構成されたウィンドウが開く。左側のペインには、ノートの階層構造がツリー表示され、右側のペインには、Todoノートのリストや現在選択しているノートの設定内容が表示される。

yankで扱うノートは、「テキスト」「チェック」「Todo」の3種類で、それぞれ書類・丸・四角のアイコンで表わされる。三者の違いは設定項目の数で、「テキスト」ではノートの名前・内容・終了日だけなのに対し、「チェック」では完了を示す印が追加され、「Todo」ではさらに締切日と優先度も設定できる。

ノートを追加するには、ツールバーの[テキスト]・[チェック]・[Todo]ボタンのいずれかを押す。続いて、右側のペインにタイトルや内容などを入力し、[OK]ボタンを押せばいい（**画面1**）。新しいノートは、ツリーで選択しているノートの下に作られる。作成後にドラッグ&ドロップで位置を変更することも可能だ。

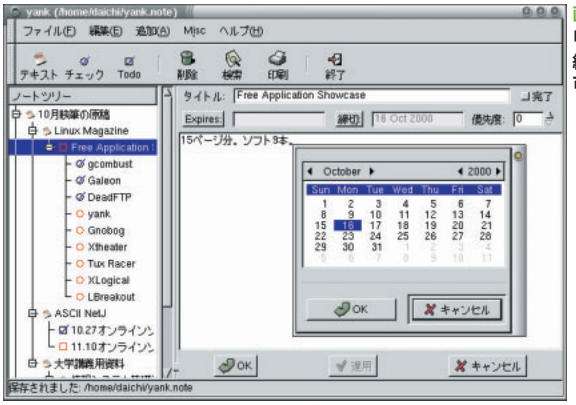
なお、ノートの種類はいつでも変換できるため、作成する段階でどれを選ぶか悩む必要はない。たとえば「チェック」として作成したノートに締切が発生したら、その時点で「Todo」に変換すればいい。

●作成後のノートの取り扱い

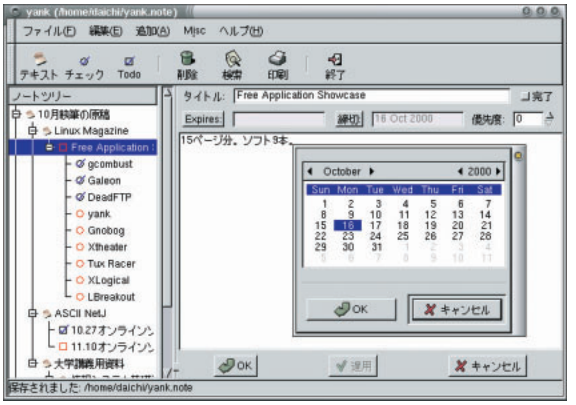
ツリーに追加されたノートをクリックで選択し、右のペインの設定内容を編集することで設定を変更できる。たとえば、「チェック」や「Todo」のノートでに用意された[完了]をチェックすると、ツリー表示ペインの対応するノートのアイコンがチェック付きのものに変化する。

また、「Todo」のノートのみ設定できる締切日と優先度を活用するには、ツリー表示の選択を解除して（[ノートツリー]部分をクリックする）、Todoリストを表示するとよい（**画面2**）。このリストには、「Todo」のノートだけが一覧表示され、締切日や優先度でソートできる。

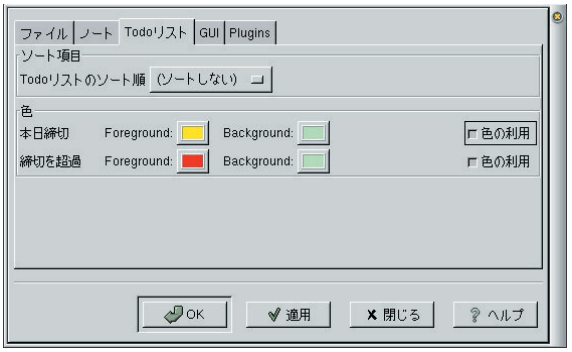
また、今日が締切日のノートや、締切日を過ぎてしまったノートの文字や背景の色を変えて目立たせることもできる。メニューの[編集]－[設定]で開く設定ダイアログの[Todoリスト]ページ



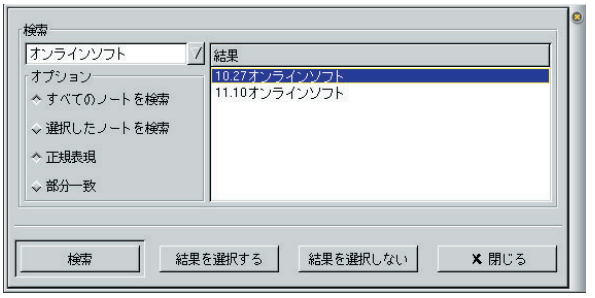
画面1
「Todo」ノートでは、締切日や優先度も設定可能だ。



画面2
「Todo」として設定されたノートの一覧が表示される。



画面3
締切日当日や締切日を過ぎたノートの文字・背景色を設定。



画面4
指定したパターンを含むノートが検索され、一覧表示される。

（**画面3**）で、各部の色を設定し、[色の使用]をチェックすればいい。

●検索機能でノートを絞り込め

ノートの数が増えるにつれ、目的のノートがどこにあるのか探すのが面倒になる。yankでは、ツールバーの[検索]ボタンで開く検索ダイアログ（**画面4**）により、すべてのノート（あるいは選択したノート）の中から、指定したパターンを含むノートだけを簡単に見つけられる。

使い方は簡単で、左上のコンボボックスに検索パターンを入力し、[検索]ボタンを押せばいい。初期設定では、[正規表現]がチェックされており、正規表現によるパターン指定が可能だ。正規表現に馴れていない場合は、[部分一致]をチェックすることで、通常の検索に切り替えられる。

検索結果に表示されたノートのタイトルをクリックすると、メインウィンドウの右のペインにそのノートの内容が表示される。検索されたノートだけを対象に絞り込み検索を行ったり、検索されたノートをまとめて選択状態にすることも可能だ。

●ノートの印刷を行う

印刷は、gnome-printライブラリを利用して行われる。先頭にノートのタイトルだけのページ、それに続いて内容を含むページが印刷される。

クリックやShift／Ctrl－クリック、検索機能などを利用して、印刷したいノートを選択し、ツールバーの[印刷]ボタンを押して、プリンタやファイルへの出力を選択する（**画面5**）。印刷イメージを別ウィンドウでプレビューすることも可能だ（**画面6**）。

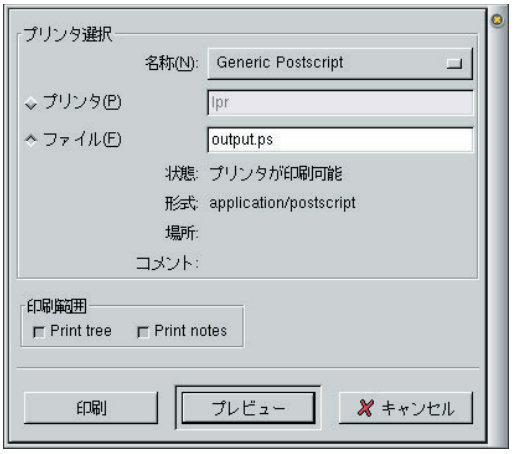
なお、ビルド時にgnome-printがインストールされていない場合、自動的に印刷機能が取り除かれ、yankのツールバーやメニューにも印刷関係の項目がいっさい現れなくなってしまうので注意されたい。

●覚え書き以外の使い道も

このほか、yankのちょっと変わった機能としては、メニューの[編集]－[テキストの選択]や、右クリックメニューの[選択]以下に用意された、テキストの選択領域に対するコマンドの実行機能が挙げられる。

これは、選択領域のテキストをURLと見なしてブラウザで閲覧したり（ブラウザで開く）、ファイル名と見なしてGNOMEのMIME設定に基づくコマンドを起動したり（MIMEとして閲覧）、コマンドラインやパイプラインと見なして直接実行したり（コマンド／パイプとして実行）できるというものだ。

うまく使えば、yankを単なる覚え書きの管理だけではなく、ブックマークやプレイリスト、画像などの総合管理を行うソフトとしても利用できるだろう。



画面5
選択したノートのタイトル一覧と内容を印刷する。

画面6
印刷イメージを画面で確認できるプレビューウィンドウ。

ブックマークの統合管理

Gnobog

バージョン：0.4.1　　ライセンス：GPL

http://gnobog.sourceforge.net/

Gnobogは、WebページのブックマークをGNOME用ソフト。NetscapeやMozillaなどのブラウザをはじめ、さまざまなソフトに分散しているブックマークを統合し、一元的に管理することを目標としている。現在はまだ未完成な部分もあるが、作成したブックマークを使ってNetscapeでブラウズでき、NetscapeやMozillaのブックマークとして保存することも可能だ。実行にはGNOME 1.2が必要となる。

●ビルドから起動まで

Gnobogは、tarボールとRPMパッケージの両方が配布されているので、使用しているディストリビューションに合わせて選択しよう。tarボールの場合、ビルドとインストールは「./configure」→「make」→「make isntall」という一般的な手順だ。

「gnobog&」として起動するか、GNOMEメニューの[プログラム]－[インターネット]－[Gnobog]を選択すると、上下に2つのペインが並んだウィンドウが開く。ウィンドウサイズの初期値は少し小さめなので、適当に拡大するとよいだろう（**画面1**）。

上部のペイン（Document list）には、現在開いているブックマークファイルのリストが一覧表示される（ブックマーク自体はそれぞれ別のウィンドウに表示される）。

一方、下部のペイン（Storing Zone）は、そうしたブックマークファイルの内容の一部を一時的に取り込んで、自由に編集するための領域だ。両者の比率は、境界部の四角をドラッグすることで変更できる。

●ブラウザのブックマークを開く

まずは、現在使っているWebブラウザのブックマークを開いてみよう。将来はIEのお気に入りやGNOME／KDEのリンクなどもインポート可能になる予定だが、現時点ではNetscape／Mozilla形式のブックマークファイルにのみ対応している。

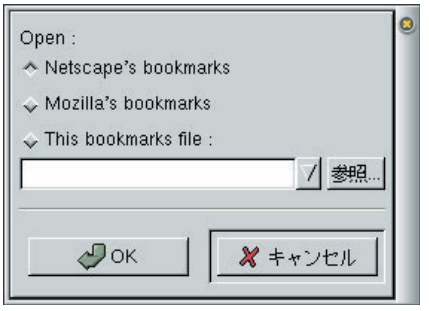
ツールバーの[Open]ボタンを押してダイアログを開き、「Netscape's bookmarks」か「Mozilla's bookmarks」を選択すればいい（**画面2**）。どちらの場合も、ブラウザで実際に使用しているブックマークファイルが読み込まれる。たとえば、Netscapeの場合は、ホームディレクトリの「.netscape/bookmarks.html」だ。

続いて、ブックマークファイル名がメインウィンドウの上部ペインに表示され、ブックマークをツリー表示する別のウィンドウが開く。ウィンドウのサイズが小さすぎるので、適当な大きさに広げよう（**画面3**）。

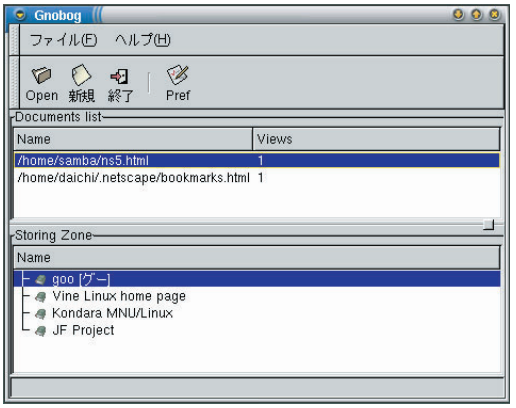
なお、Mozillaのブックマークでは、日本語コードにUTF-8が採用されているため、そのまま読み込むと日本語部分が文字化けしてしまう。どうしてもインポートしたいなら、UTF-8を扱えるエディタ（MGEditなど）でブックマークファイルを読み込み、日本語EUCで別ファイルに保存した後、Gnobogのオープンダイアログの「This bookmarks file」を選択して、そのファイルを指定すればいい。

●ブックマークを編集する

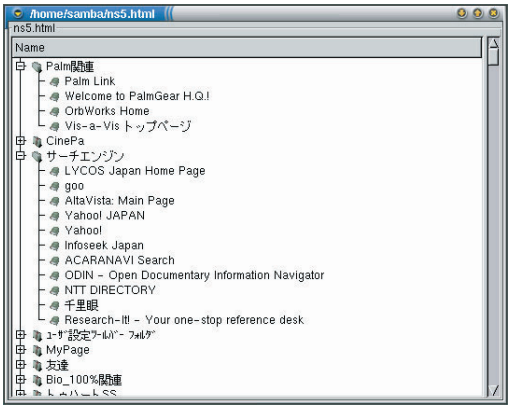
ブックマークがツリー表示されたウィンドウでは、右クリックメニューに



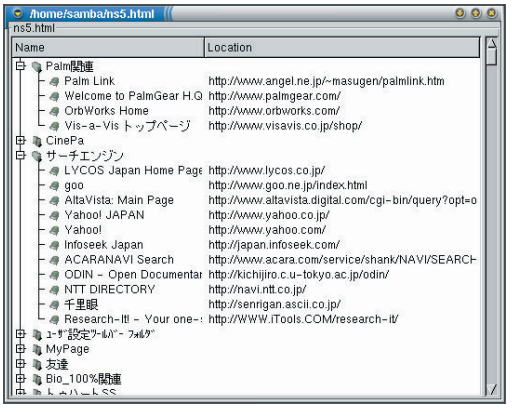
画面2
Netscape/Mozillaのブックマークファイルをインポートできる。



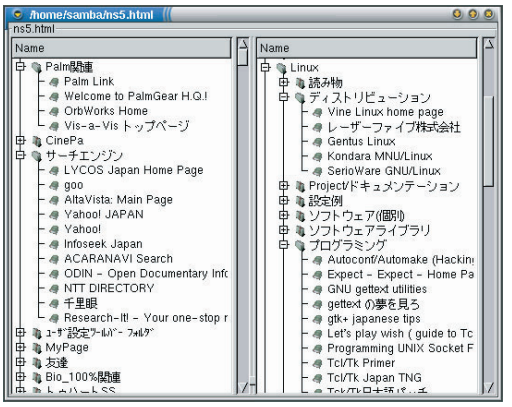
画面1
ブックマークファイルのリストと一時的な編集領域で構成される。



画面3
ブックマークのツリーは独立したウィンドウに表示される。



画面4
ブックマークの名前とともにURLを表示することも可能だ。



画面5
同じファイル内の2カ所で作業する場合に便利な分割表示。

より、ブックマークの内容の編集や並べ替え、削除などを行える。[Show]－[Locations]をチェックして、ブックマークのURLを表示したり（**画面4**）、[Window]－[Split]をチェックして、ウィンドウ内部を2分割することも可能だ（**画面5**）。

ただし、このままだと各ブラウザのブックマークを直接変更することになるので危険だ（特にUTF-8を使用するMozillaの場合）。新たなブックマークファイルを作成して作業を行うほうがよいだろう。

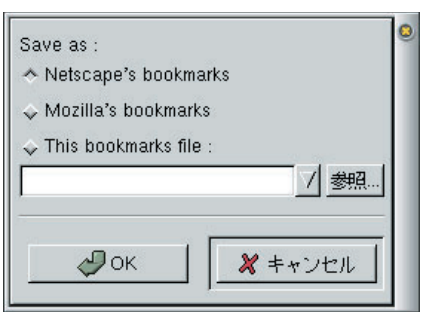
ツールバーの[新規]ボタンを押すと、「New Documment」という仮ファイル名のブックマークファイルが新規作成され、新たなウィンドウが開く。先ほどと同様に適当なサイズになるまでウィンドウを拡大しよう。

ブックマークの内容をコピーするには、ウィンドウ間でブックマークをドラッグ&ドロップすればいい。フォルダ以下をまとめてドラッグ&ドロップすることも可能だ。このほか、右クリックメニューの[Selection]以下のメニューも利用できる。

また、ブックマークの削除や並べ替えなどの作業が必要な場合は、メインウィンドウの下部ペインを一時的な保管場所として利用し、編集終了後に目的のウィンドウにコピーするとよいだろう。これなら、誤って別のブックマークを削除してしまっても、被害が最小限で済む。

内容を変更したブックマークファイルは、右クリックメニューの[ファイル]－[保存]（あるいは[名前を付けて保存]）で保存できる（**画面6**）。

ブラウザが直接利用するブックマークファイルを上書きすることも可能だが、安全のため、新規作成したブックマークは、「This bookmarks file」を選択して、適当なファイル名で保存したほうがよいだろう。



画面6
Netscape／Mozillaのブックマークファイルを上書きできる。

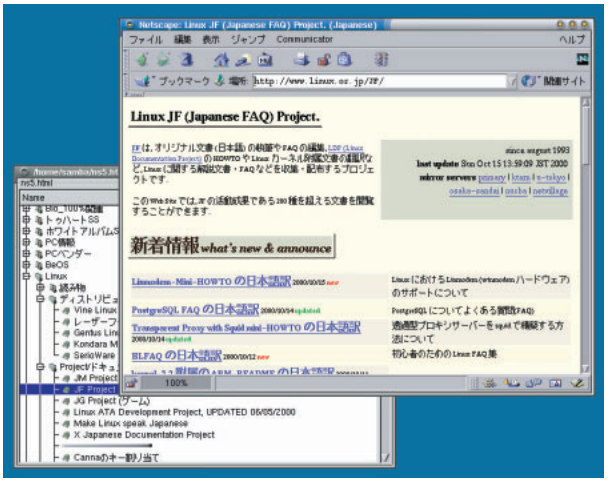
●ブラウザとの連携

Gnobogに登録されたブックマークのWebページをNetscapeで開くには、単にブックマークをダブルクリックすればいい。新たなNetscapeのウィンドウが開いて、該当するWebページが表示される（**画面7**）。

また、右クリックメニューの[Browse]－[Browse in Netscape existing window]を選択すると、現在開いているNetscapeのウィンドウを利用して、該当するWebページが表示される。

どちらの場合も、あらかじめNetscapeが起動している必要がある。また、起動するブラウザがソース内でNetscapeに固定されているため、MozillaやGaleonなど他のブラウザを利用できない。これらの欠点は、今後のバージョンアップにより解消されるだろうと思われる。

画面7
現時点では連携の対象となるブラウザはNetscapeのみ。



GTK＋ベースのマルチメディアプレーヤ

Xtheater

バージョン：0.5.3　　ライセンス：GPL

http://xtheater.sourceforge.net/
http://www.libsdl.org/（SDL）
http://www.lokigames.com/development/smpeg.php3（SMPEG）

Xtheaterは、MPEG／AVI／VideoCDなどの動画やMP3の再生に対応したGTK＋ベースのマルチメディアプレーヤだ。MPEG形式の動画の再生にはSDLとSPMEGライブラリ、AVI形式の動画の再生にはWindows用のコーデックDLLをプラグインとするlibaviplayを利用する。このほか、HTTPやFTP経由でのストリーム再生にも対応している。

●ビルドとインストール

最初に、Xtheaterが利用するライブラリを、SDL→SMPEG→libaviplayの順にインストールする。SDL 1.1.5とSMPEG 0.4.1については、tarボールとRPMパッケージの両方が配布されているので、使用しているディストリビューションに合わせて選択しよう。

libaviplayは、Windows用コーデックDLLをプラグインとしてロードし、AVIファイルの再生を行うライブラリで、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」とすればOKだ。

コーデックDLLは、/usr/local/lib/Xtheater/win32に置く。Windowsが別パーティションにあるなら、マウント先のSystemディレクトリのシンボリックリンクを作るだけでいい。たとえば、マウントポイントが「/mnt/dosc」なら、rootになった状態で、

```
# ln -s /mnt/dosc/WINDOWS/SYSTEM
/usr/local/lib/Xtheater/win32
```

とすればOKだ。

Xtheater自体は、tarボールのみ配布されている。ビルドとインストールの手順はlibaviplayと同じだ。なお、ビルドする前に、plugins/ui/gtk-ui.cの148行目「gtk_init（argc,argv）;」の前に、「gtk_set_locale（）;」という一行を挿入すると、ダイアログの文字化けを解消できる。

●動画の再生は別ウィンドウで

「xtheater&」として起動すると、シンプルなウィンドウが開く。ウィンドウ下部のイジェクトボタンを押すとオープンダイアログが開くので、ディスク上のMPEG／AVI／MP3ファイルなどを選択しよう。

ウィンドウに再生・一時停止・巻き戻しなどのボタン、シークやボリューム用のスライダが表示され、再生が始まる（**画面1**）。動画は別ウィンドウに表示される（**画面2**）。[Fullscreen]などのボタンで動画の表示サイズを変更することも可能だ。

なお、フルスクリーン表示では、Xの画面解像度を、再生する画像の大きさに合わせて切り替えている。コーデックと画像ファイル形式の組み合わせによっては、音声と画像が同期しなくなるなどの問題もあるようだ。フルスクリーン表示からウィンドウ表示に戻るには、マウスボタンをクリックすればいい。

このほか、[Media]以下のメニューにより、URLを指定してHTTP／FTP経由でのストリーム再生を行ったり、/dev/cdromに挿入されたVideoCDを再生することも可能だ。なお、現時点ではVideoCDに関してはシーク機能が実装されていない。

画面1　再生時のウィンドウにはさまざまなボタンやスライダが表示される。

画面2　動画は独立したウィンドウに表示される。

ペンギンが雪山を滑り降りる3Dゲーム

Tux Racer

バージョン：0.60-3　　ライセンス：GPL

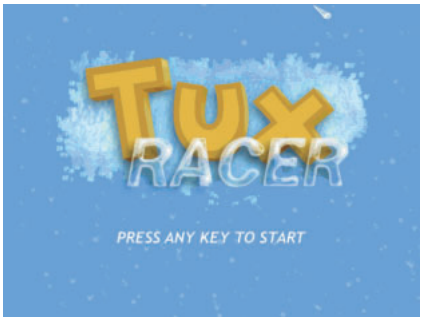
http://www.tuxracer.com/
http://www.libsdl.org/（SDL）

Tux Racerは、Linuxのマスコットであるペンギンの Tux君が雪山を滑り降りる3Dアクションゲームだ。従来から人気のあったゲームだが、今回、タイトル画面やBGMの追加、コースのリニューアル、クリアタイムとアイテム獲得数によるトライアル制の導入など、ゲームとしての完成度が大幅にアップして再登場した。動作にはMesa 3.2以降とGLUT、SDL、SDL_mixerの各ライブラリが必要。快適に遊ぶには3Dアクセラレータが必須だ。

●ビルドとインストール

まずはライブラリ類のインストールを済ませておく。SDL 1.1.5とSDL_mixer 1.0.6については、tarボールとRPMパッケージの両方が配布されているので、使用しているディストリビューションに合わせて選択しよう。

また、ビルドにはSGI提供のヘッダファイル「glext.h」（http://oss.sgi.com/projects/ogl-sample/ABI/glext.h）が必須だ。入手後、/usr/include/GLにコピーしておく。



画面1　コミカルなBGMとともに表示されるタイトル画面。

Tux Racerの最新版0.60は、tarボールでソース一式とデータが配布されている。ソース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。データのほうは、/usr/local/shareで展開後、「mv tuxracer-data-0.60 tuxracer」として、ディレクトリ名をtuxracerに変更する。

●コースの先を読んで方向を決めろ

「tuxracer&」として起動すると、コミカルなBGMとともに雪が降るタイトル画面が表示され（**画面1**）、キーを押すとメニューが表示される。通常のゲームをプレイするには[Enter an event]を選択すればいい。

イベントとカップ、レース（コース）の選択などを行うと、いよいよプレイ開始だ。Tux君が登場して、雪山を滑降し始める（**画面2**）。

プレイ中の操作方法は、←／→キーで左右にカーブ、↑キーでパドリング（加速）、↓キーでブレーキだ。木や山に衝突してスピードが落ちてしまったら、↑キーでパドリングして勢いをつけよう。

3Dアクセラレータ環境でのスピード感は快適だ。勝負を忘れてスピード狂になるのもいいだろう。

各コースには、クリアに必要なタイムと獲得すべきアイテム（ニシン）の数がそれぞれ設定されている。設定タイム以下でゴールに達し、獲得したアイテムが設定値以上ならそのレースはクリアで、次のレースに挑戦できるようになる。

変化に富んだコースが13種類用意されており、時間や天候などによりさらにバリエーションが増える。なお、最初のメニューで[Practice]を選択すると、レースのコースや天候などの設定を自由に設定して、気軽に各コースを楽しめる（**画面3**）。

画面2　プレイ開始だ。Tux君が腹ばいになって滑降し始める。

画面3　変化に富んだコースが13種類も用意されている。

4色のボールを使ったアクションパズル

XLogical

バージョン：1.0-3　　ライセンス：GPL

http://changeling.dynip.com/xlogical/
http://www.libsdl.org/（SDL）

XLogicalは、4色のボールと回転する「スピナー」を使って面をクリアしていくアクションパズルゲームだ。ルールはシンプルで覚えやすいが、同時にいくつものスピナーを操作する必要があるのでプレイ中はとても忙しい。面が進むにつれ、ボールの動きや色を変えるさまざまなギミックが登場し、プレイヤーを飽きさせない。動作にはSDL、SDL_image、SDL_mixerの各ライブラリが必要だ。

●ビルドとインストール

まずはライブラリ類のインストールを済ませておく。SDL 1.1.5とSDL_mixer 1.0.6、SDL_image 1.0.10については、tarボールとRPMパッケージの両方が配布されているので、使用しているディストリビューションに合わせて選択しよう。

XLogicalについても、tarボールとRPMパッケージの両方が配布されている。tarボールの場合、「./configure」→「make」→「make install」という一般的な手順でビルドとインストールが行われる。

●同色のボールをスピナーに入れろ

「xlogical&」として起動すると、BGMとともにサイケなタイトル画面が表示される（**画面1**）。Escキーを押してメニューに切り替えよう。[START GAME]を選択して、SPACEキーを押すとプレイ開始だ。

画面には、ボールが通過する通路と、ボールを入れる4つの穴を持つ「スピナー」が表示され、赤・青・緑・黄のいずれかの色のボールが発射される（**画面2**）。

画面1
サイケデリックな雰囲気の漂うタイトル画面。

発射直後のボールには「ボールタイマー」が設定されており、これが0になるまでにスピナーの穴に入れなければならない。通路に面したスピナーの穴にすでにボールが入っていると、新たなボールはそちらには移動しないので、プレイヤーがスピナーを回転させたり、入っているボールをリリースする必要がある。

操作にはマウスを使用する。スピナーを右クリックすると反時計方向に回転し、スピナー内のボールを左クリックすると通路（または隣接するスピナー）にリリースされる。

スピナーの4つの穴すべてに同じ色のボールを入れると、それらのボールとともに中央部の点滅が消える。すべてのスピナーの点滅を消すと1面クリアだ。

面が進むに連れ、特定の色のボールしか通さない「ブロッカー」や、別の場所にボールを転送する「テレポーター」、ボールの色を変えてしまう「ペインター」など、さまざまなギミックが登場する（**画面3**）。

画面2
同じ色のボールを4つ、中央が赤く点滅するスピナーに集めよう。

画面3
面が進むにつれ、「テレポーター」などのギミックが登場する。

昔なつかしのブロック崩しゲーム

LBreakout

バージョン：001014　　ライセンス：GPL

http://lgames.sourceforge.net/
http://www.libsdl.org/ （SDL）

LBreakoutは、昔なつかしいブロック崩しゲームだ。派手な効果音や美しい背景、ブロックから落ちてくるアイテムによるパワーアップなど、ユーザーを退屈させない完成度の高いゲームに仕上がっている。動作にはSDL 1.1.5が必要だ。

●ビルドとインストール

動作に必要なSDL 1.1.5は、tarボールとRPMパッケージの両方が配布されているので、ディストリビューションに合わせて選択しよう。

LBreakout本体は、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。

●マウスでパドルを操作

「lbreakout&」として起動すると、メニューを含むタイトル画面が表示される（**画面1**）。[New Game]を選択してあなたの名前を設定し、再度[New Game]を選択するとプレイ開始だ。

ルールはいまさら説明する必要もないだろう。画面下部のパドルを操作してボールを打ち返し、画面上のブロックをすべて消せばいい（**画面2**）。

操作にはキーボードとマウスを利用する。←／→キーかマウス移動でパドルの移動、スペースキーか左クリックでボールの発射だ。思い通りの位置でパドルを止めるには、マウス操作のほうが有利だ。ただし、マウスカーソルがウィンドウの外に出るとパドルを動かせなくなるので注意されたい。

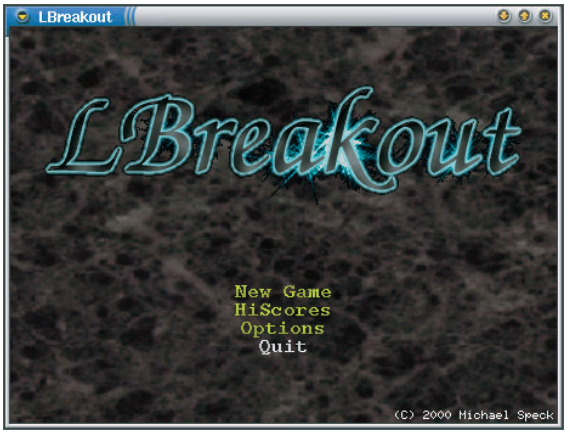
ブロックの中には、消える際にアイテムを落とすものがある（**画面3**）。これをパドルで拾うと、バドルが長くなる、ボールが増えるといったさまざまな効果が現れる。中には、一定時間パドルが動かなくなる、パドルが短くなるといったマイナスの効果を生むものもあるので気をつけよう。

●オプション設定

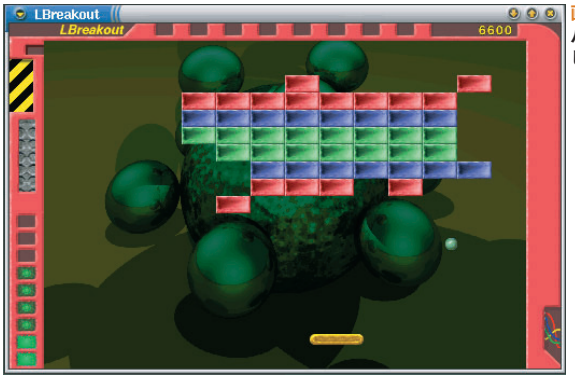
タイトル画面の[Options]以下のメニューでは、ゲームの難易度や操作方法、グラフィック、サウンドに関する設定が可能だ。

たとえば、プレイしてみて難しすぎる（あるいは簡単過ぎる）ようなら、[Options]－[Game]の画面で、[Difficulty]の設定を「Easy」や「Hard」に変更すればいい。

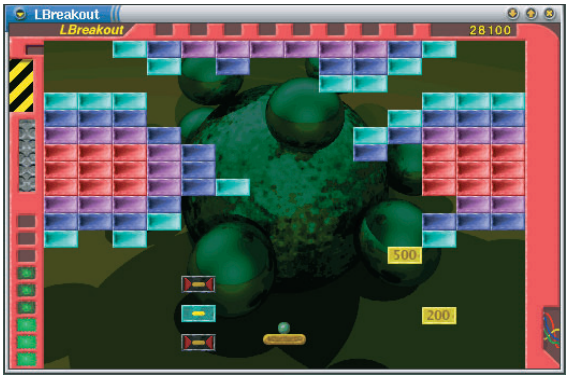
また、[Options]－[Control]の画面で[Warp Mouse]の設定を「On」に変更すると、プレイ中にマウスカーソルがウィンドウの外に出なくなり、常にパドルを制御できるようになる。



画面1
メニューを含んだLBreakoutのタイトル画面。



画面2
パドルをマウスで操作してボールを打ち返せ。



画面3
さまざまな効果を発揮するアイテムが落ちてくる。

Software Catch Up

韓国生まれの日本語オフィススイート

HancomOffice 1.0

韓国生まれの日本語ワープロとしてHancomWordを紹介したのは記憶に新しいところだが、わずか数カ月でオフィススイートとして再登場することになった。はたしてHancomOfficeの実力はいかがなものだろうか。

文：塩田紳二
Text : Shinji Shioda

価格	未定
問い合わせ先	HancomLinux Japan, Inc. Tel:03-3258-3401 URL:http://hancom.com/jp/

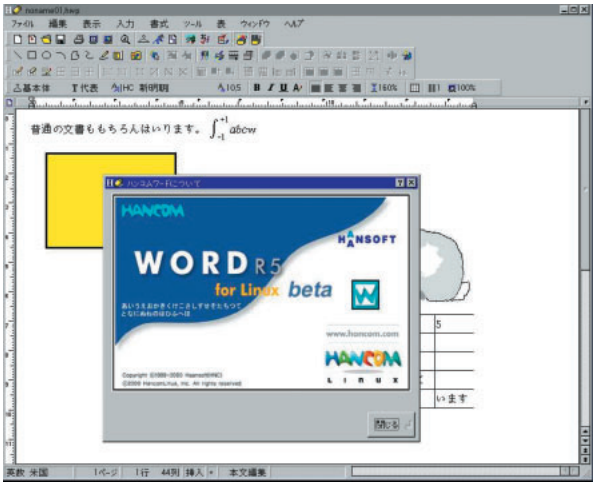
HancomOffice for Linuxは、韓国HaansoftのオフィススイートHancom OfficeのLinux版という位置付けだ。開発は、Hancom Linuxが行っている。本誌9月号で紹介したHancom Wordは、ワープロ部分のみだったが、表計算ソフトやプレゼンテーションソフトなども用意され、オフィススイートとして開発がすすめられている。今回、β版を入手することができたので、そのレポートをいち早くお届けする。なお、このレポートは開発中のβ版に基づくものであり、実際の製品化にあたっては機能や画面などが変更されている可能性があることをあらかじめ断りしておく。

今回のHancomOffice 1.0 for Linuxに含まれるのは、

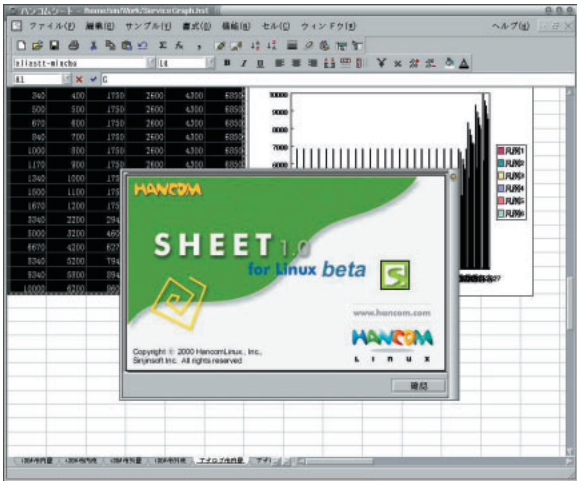
①HancomWord R5 for Linux
ワープロ（画面1）
②HancomSheet 1.0 for Linux
表計算（画面2）
③HancomPresenter 1.0 for Linux
プレゼンテーション作成（画面3）
④HancomPainter 1.0 for Linux
ビットマップ描画（画面4）

という4つのアプリケーションと、これらを起動するためのHancomShell（**画面5**）である。

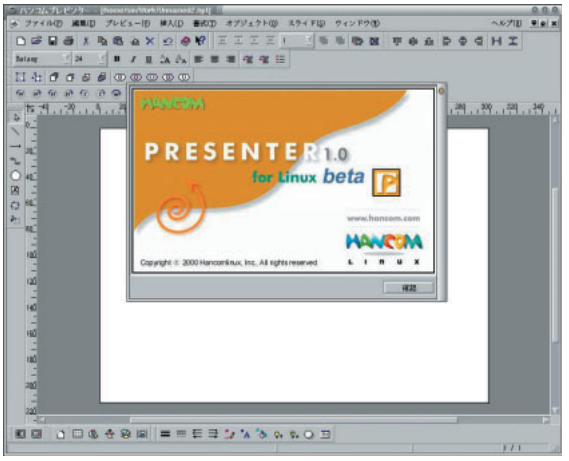
HancomWordは、もともとHaansoftがWindows用として販売していたものからの移植であり、古くからの製品でもあるのでバージョンがR5となっているが、それ以外のアプリケーションは、今回新たに作られたもののようである。なお、Windows版の発売元であるHaansoftのホームページ（英語）を見る限り、米国で販売されているHancomOfficeは、HancomWordとLotusの123やFreelanceとの組み合せのようである（韓国語のページは筆者が韓国語を理解しない関係でよくわからなかったが……）。したがって、このHancomOffice for Linuxに含まれる表計算ソフトなどは、Linux版独自の製品（同社のオリジナルかどうかは不明



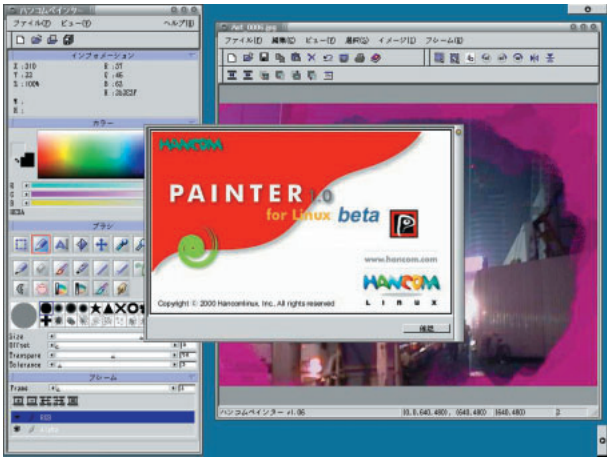
画面1
HancomWord R5 for Linux



画面2
HancomSheet 1.0 for Linux



画面3
HancomPresenter
1.0 for Linux



画面4
HancomPainter
1.0 for Linux

だが）のようである。バージョンが1.0となっているのは、これがその最初のものであるという意味なのだろう。なお、このHancomOffice for Linuxには日本語版のほか英語版、韓国語版があるようだ。

HancomShellは、各アプリケーションおよび、Hancom Linuxのホームページを表示するための5つのボタン（終了ボタンを入れれば6つ）を持つだけの簡単なものだ。これを表示させておけばアプリケーションの起動が簡単になるが、後述するように、メニューにも登録が行われるため必ずしも必要というわけではない。

なお、前回紹介したように、HancomWordはWindowsエミュレータであるWineを利用したアプリケーションであるが、それ以外のHancom SheetやHancomPresenterは、Wineを利用しないLinuxネイティブのアプリケーションのようである。

インストール

HancomWordのインストールはちょっと煩雑だったが、今回のHancom Officeには簡単にインストールできるインストーラが付属し、基本的にはX Window System内からインストーラを起動するだけで作業は終了する。また、実際のインストールイメージは、RPMパッケージとなっており、手動でのインストールも不可能ではないし、インストール後の管理は、ほかのパッケージ同様rpmコマンドで行うことができる。

ただし、インストール作業はrootで行う必要がある。また、インストールプログラムがX Window Systemを呼ぶ関係で、一般ユーザーとしてログインし、単にターミナルウィンドウ内でsuでrootになってインストーラを起動しても、「can not connect to X server」とエラーになる。rootでログインしたうえでインストールを行うか、一般ユーザーでログインした場合には、

```
$ su -c ./install
```

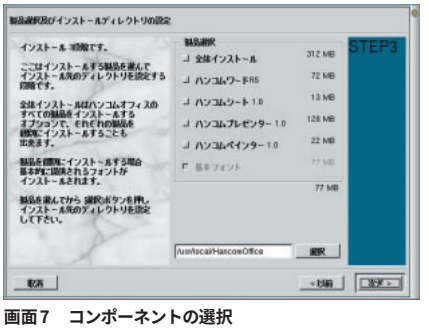
として現在の環境を引き継いだまま起動する必要がある。

画面5　HancomShell

インストーラでは、最初に使用許諾契約の同意（**画面6**。β版なのでReadmeファイルが表示される）があり、次にインストールするコンポーネントの選択を行う（**画面7**）。ただ画面では、このコンポーネントの選択はSTEP3になっており、製品版では利用許諾契約との間にもう1つ画面が追加されるのかもしれない（ライセンスコードの入力などであろうか）。作業自体はボタンを押すだけで進行し、特に面倒なことはない（**画面8・9**）。

このコンポーネントの選択では、HancomOfficeのすべてのアプリケーションのインストールのほか、個別のアプリケーションごとにインストールの選択が可能になっている。また、イン

画面6　使用許諾画面



画面7　コンポーネントの選択

ストール先の選択も可能である（デフォルトは、/usr/local/HancomOffice）。

ただし実際には、指定したディレクトリ以外にもいくつかのディレクトリに関連のファイルがインストールされるようである。たとえば、起動用のシェルスクリプトは、/usr/binにインストールされ、そこから/usr/local/HancomOfficeにある各モジュールが呼び出される。ただし、HancomWordだけは/usr/hwpxにリンクがあり、ここを経由して/usr/local/HancomOffice/Word内のモジュールが起動される。β版のためか、このあたりが整理されていない感じではある。ただし、HancomOfficeにはアンインストール用のコマンドもあるので、多少インストール先が散らばっていてもふつうは問題ないだろう。

なお、インストールするとGNOMEのメニューにHancomOfficeのメニューが追加される（**画面10**）。このため、起動には、HancomShellを必ずしも使う必要はないだろう。

HancomWord

HancomWord（**画面11**）は前回紹介したものから、多少変更が行われている。以前のバージョンでは、同じ意味を持つボタンが2つ表示されたり、一部ハングル文字が出るなど、ユーザーインターフェイス部分に若干混乱がみられたが、今回のものではそのあたりが修正されているようだ（ただし、フォント名などでまだハングル文字が表示されるところがある）。また、メニュー文字や、付属ファイル（テンプレート）などが変更されている。

ただし、ざっと見たところ細かい部分の修正だけで基本的な機能については変わっていないようである。前回の繰り返しになるので今回は簡単に解説する。

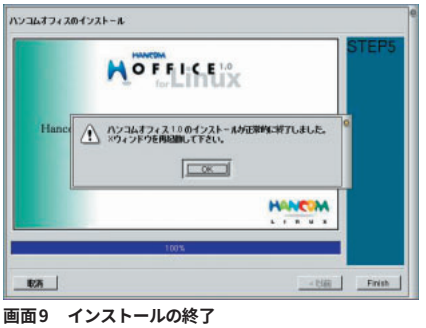
ファイルとしてはWindowsで使われているMicrosoft WordのDOCファイルや一太郎のjswファイル、WordPerfectのWPSファイルなどに対応しており、そのほかにHTMLやRTF（Rich Text Format：マイクロソフトのテキスト表現された書式付き文書ファイル）、Lotusのワークシート（wk?）なども読み込むことが可能となっている（**画面12**）。

また、段組みなども可能で、文書内にはグラフィックやベクトル図形（描画機能あり）、表、数式などを入れることも可能だ（**画面13**）。

簡易プレゼンテーション機能を持ち、表内では計算式も利用できる。このあたりは、ワープロ単体でさまざまな作業に対応するためのものだが、今回はそれぞれ専用のアプリケーションが用意されているため、これを使うことはあまりないと思われる。

また、前回評価したときに比べて安定している感じだ。前回評価したHancomWordでは、ウィンドウの移動時などにハングすることがあった。今回の評価中にもそのような事態は全く

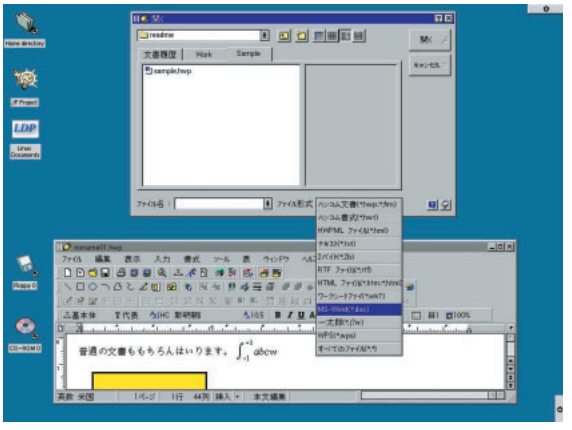
画面8　インストール中画面

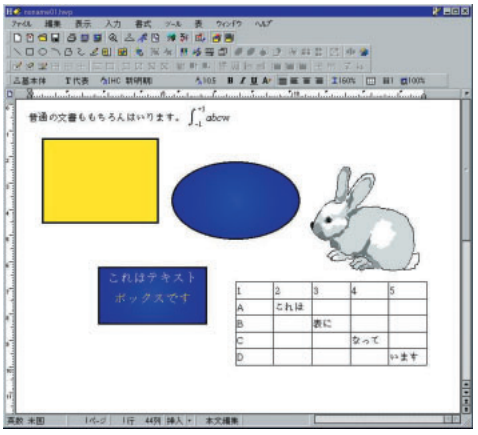


画面9　インストールの終了

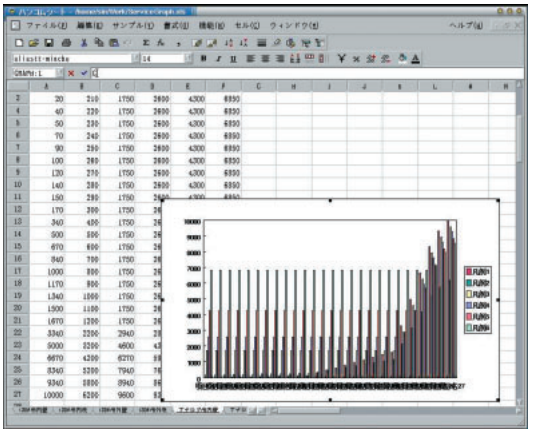

画面10　GNOMEのメニューに追加されたHancom Office


画面11　HancomWordの編集画面


画面12　他文書形式の読み込み



画面13
グラフィック、ベクトル図形、表、数式の作成



画面14
HancomSheetの編集画面

起きなかったわけではないが、確率的には低くなったような感じである。正式版になれば、さらに安定度は増すことだろう。

HancomSheet

HancomSheet（**画面14**）は、表計算ソフトである。ファイルフォーマットとしてはWindowsで使われているMicrosoft ExcelのXLSファイルの読み込みに対応している。

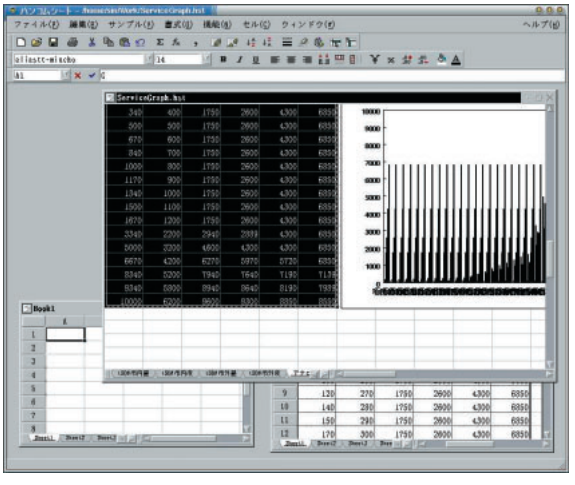
メインウィンドウは、メニューバー、ツールバー、そして数式入力バーを持つ、ごく一般的な表計算ソフトの形式になっており、1つのファイルがその内部に子ウィンドウとして表示される（**画面15**）。子ウィンドウは、メインウィンドウから外にでることはできず、メインウィンドウの端でクリッピング表示される）。複数のファイルを開くことが可能だが、その場合には複数の子ウィンドウが表示される。なお、1つのファイル内には複数のワークシートを持つことができ、これはウィンドウ下部にあるタブで切り替える。

GUI操作は、Microsoft Excelに似ており、セルをドラッグしてセル内容の移動やコピーが行え、選択枠下のハンドルをドラッグすることで、連続値の自動入力などが行える。

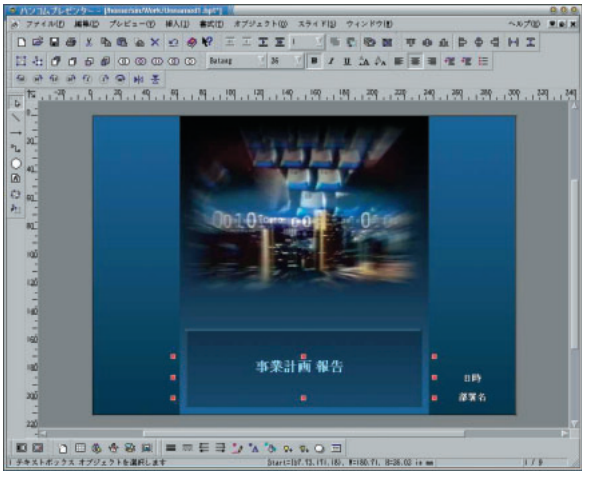
セル範囲を選択して簡単にグラフを作成することができるが、グラフはワークシート上にのみ配置でき、グラフをシート化してワークシートと同等に扱うことはできないようだ。ただ、このβ版ではグラフの種類なども指定できない状態であるが、これは開発途上のためと考えられる。なお、発売元のHancom Linuxのホームページにある製品紹介によると、ウィザード形式でグラフを作る機能が装備されるようである。また、同じくワークシート上にベクトル図形の描画が可能になるらしい（β版にはそのような機能のためのボタンは装備されていなかった）。

関数などは、数学用、統計、会計（金融）用など85個用意されており、一般的な利用であれば十分な数であろう。ただし、ユーザーが関数を定義する方法はないようである。もっとも、かなり複雑な式を使うのでなければ、必要なセルをコピーすることで、関数定義を行わずに同等の計算式を定義することが簡単に行えるため、実用上それほど問題になることはないだろう。

たとえば、Windows側のデータベー



画面15
MDI形式のウィンドウ



画面16
HancomPresenterの編集画面

スと連携するとか、VBA（Visual Basic Application Editon：マイクロソフトのオフィス系アプリケーションに組み込まれているスクリプト言語）を使っているといった複雑な作業や、シートのアウトライン、自動フィルタといった高度な機能を使ったシートでなければ、おそらくこのHancomSheetでも同様に作業できると思われる。

HancomPresenter

HancomPresenter（**画面16**）は、プレゼンテーション資料の作成ツールで、Microsoft PowerPointのファイルの読み込みが可能となっている。

資料作成には、ページタイプ（図版の有無やタイトルの位置など）、背景タイプ（**画面17**）、作成ずみテンプレート（特定の目的用にあらかじめ作成された複数ページからなるテンプレート）の3つの方法で作成が可能だ。

ページ内には、テキストブロックやベクトル図形、ビットマップ画像などが配置でき、別途、バックグラウンドを指定することができる。また、描画に使う色のセットがあらかじめ作られており、切り替えが可能となっている（**画面18**）。これは、一般にプレゼンテーション資料では、画像や背景などを除いて、あまり多数の色を使わないほうが見やすくなること、色の組み合せによりテキストなどが見やすくなること（逆にいえば、テキストが読みにくくなる組み合せにならないように注意すべき）などから、最適な色のセットがあらかじめ用意されているのである。

印刷は、ページのみ、配布資料用、発表者ノート（別途各ページに指定したメモを同時に印刷する）の3タイプが選べ、1枚の用紙にページをいくつ入れるか、そのときの縦横の並びなどの指定も行える。

このほか、画面を使ったスライドショーも可能で、このときには作成したページから任意のものを選んで、指定した順序でのスライドショーが行える。このため、1つの資料を作り直すことなく、聴衆や場所に合わせて再構成して利用可能だ。

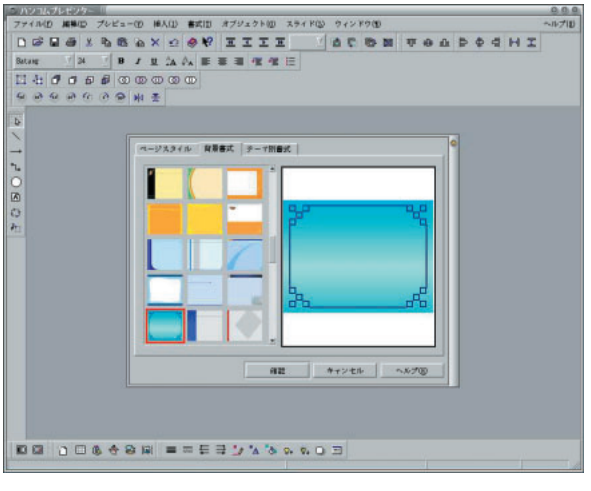
HancomPainter

HancomPainter（**画面19**）は、ビットマップ画像の編集、作成ツールで、オリジナルのファイルフォーマットのほか、GIF、JPEG、BMP（Windowsのビットマップファイル）、XBM、XPM、PNG、PNMファイルの読み込み、書き出しが可能になっている。

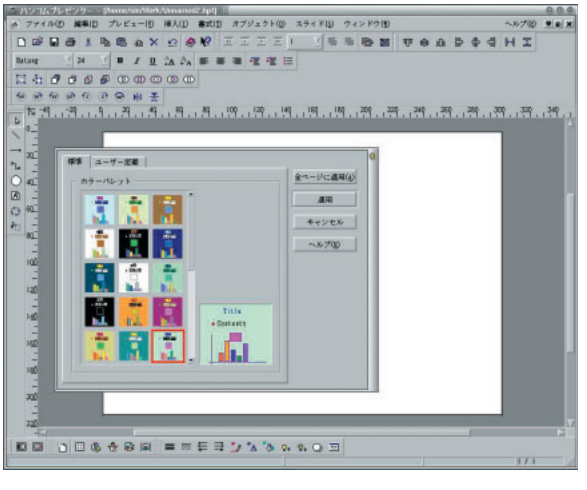
このソフトは、画像を表示するメインウィンドウと、ブラシの選択やカラー選択などのパレット類を表示するウィンドウの2つから構成されている。どちらのウィンドウにもメニューとツールバーがあり、ちょっと変わった印象を受ける。というのは、パレットを表示するウィンドウの幅がそんなに小さいものではなく、大きなウィンドウを2つ使うからである。高解像度のシステムを使うぶんにはそれほど問題はないだろうが、横幅が1024～1152ドット程度だと、どうしても2つのウィンドウが重なってしまい、切り替え時にウィンドウの再描画が行われるためちょっと使いづらい感じがする。

画像は、最初に指定した解像度の範囲内で描画を行うが、1つのファイルに複数のフレームを定義してアニメーションファイルを作成することもできる。また、このときアルファチャンネル（画像のマスキングなどに使用）を使うこともできる。

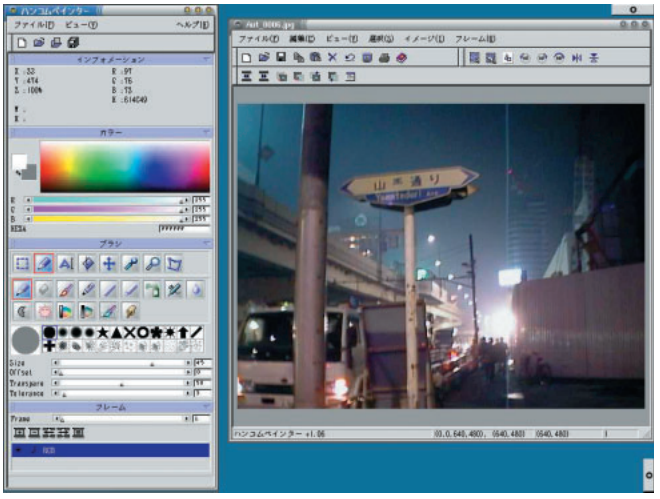
描画機能は豊富にあるが、画像の調整は、明るさやコントラスト程度（**画面20**）で、いわゆるフィルタ機能を持っていないため、デジタルカメラで撮影した画像の簡単なレタッチは可能だが、本格的な処理には少々力不足という感じがある。どちらかというと、ビ



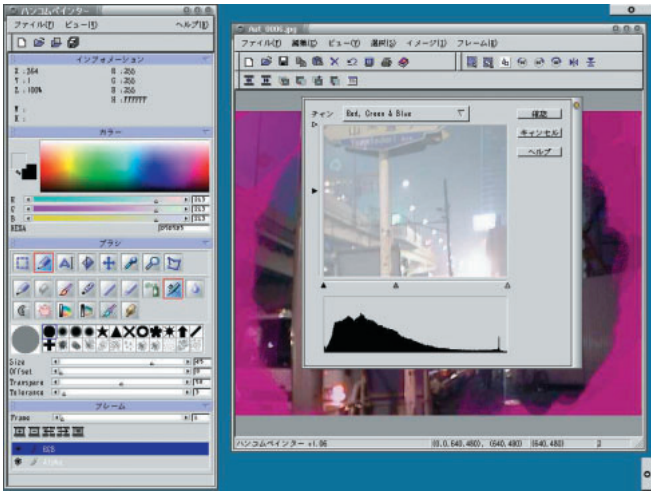
画面17　背景タイプの指定



画面18　描画色のセット



画面19　HancomPainterの編集画面



画面20　明るさの変更

ットマップで簡単な画像を描くという感じになるだろう。

ヘルプと印刷

ヘルプファイルは、β版ではHancom Wordにのみ用意されていた。これは、どうもWindows版のWindows HELPファイルを使ったもののようである。他のアプリケーションは、ヘルプメニューを開くと、Webブラウザ（Netscape）が起動するがβ版ではまだ、HELPファイルがないので表示が行われない。

また、HancomOfficeの印刷関連の機能は、Postscriptプリンタを前提としている。Wineを使うHancomWordとそれ以外のプログラムでは印刷用のダイアログが違い、HancomWord以外のアプリケーションは、Printcapに定義されたプリンタが列挙され、そこに対してPostscriptで出力を行うようである（実際に印刷を行ってみたが、プリンタキューにはなにも送られなかった）。

その他

残念ながら評価時間が短かったため、あまり細かい部分まで確認することができなかった。

なお、β版のためかUndoの動作がおかしいなどいくつかの不具合が見られた。もっとも、本来想定している動作がわからないため不具合なのかどうかわからないものもある。

雑感

さて、全体としてHancomOfficeは、一般的に必要なアプリケーションが1つにまとまっており、いわゆる普通の非定型業務といわれるレポート作成や資料作成といった作業を行わせることができる。

多少残念に思えたのは、各アプリケーション間の連携機能だ。

たとえば、HancomSheetのセル範囲をコピーし、HancomWordに貼り付けると、単なるタブ区切りのテキストとしてしか挿入できなかった。一応、そのあとタブ区切り文字列を表に変換することはできるが、できればここは、Windowsなどのようにオリジナルのシートファイルへのリンクや埋め込みを行ってほしい部分である（任意のアプリケーション間で実現してほしいとまでは思わないが……）。

あるいは、HancomWordの表機能には、関数を含む計算式を定義できるので、HancomWordの表機能への変換も同一メーカーの製品であれば不可能ではないと思われる。

日本語が利用できるオフィススイートとして、お隣の韓国で作られたHancomOfficeは、国内のLinuxユーザーにとって、Linuxマシンをデスクトップで使う場合の選択肢の1つとはなろう。英語版をベースにしたオフィススイートの移植版に比べると、直接入力が可能であること、入手のしやすさ（11月に出荷予定で同社のサイトからオンラインで購入可能）など有利な点もいくつかある。

気になる価格だが、現時点では未定だというものの、1万2000円前後になるだろうということだ。なかなか手軽な値段でオフィススイートが入手できるということになる。

現状を考えると、Windowsのオフィスユーザーが Linuxマシンをすぐに利用する可能性はそう高くないが、たとえばログの処理やそれらを使ったレポートの作成など、技術系のユーザーでも、こうしたオフィススイートを利用して生産性を上げることは可能である。そう考えると、それほど小さい市場とも思えないので、ほかのソフトハウスにも注目していただきたい分野である。

隠喩としてのコンピュータ

ふたつのC2C、オークションと地域通貨

文：豊福 剛
Text：Tsuyoshi Toyofuku

インターネットでのオークションにハマっている人が、このところ急増しているようだ。オークションのサイト、海外ではeBayが有名だが、国内ではYahoo!オークションに人気が集まっているらしい。ためしに利用してみたら、そこに出品されている品物の充実ぶりに驚いてしまった。15年前くらいから、あちこちの古本屋で探していた本が、あっけなく見つかった。私のまわりにも、品物を買うだけでなく、自分で出品している人もいて、古本屋に売るよりもかなりいい値段で売れたらしい。

5年くらい前になるが、インターネットがビジネスに与えるインパクトや影響について、さまざまな議論があった。インターネットで通販をやるなら、本やCDが適していることぐらい、誰でも思い付いただろう。洋書が安く早く購入できるサービスがあるといいな、と思っていたら、Amazon.comが登場した。求人情報もインターネットが中心になるだろうと予想していたら、そのとおりになった。売ります・買います情報を扱うサイトも出てくるはずだと考えていたから、eBayが登場したときにも、さほど驚きはしなかった。とはいえ、オークションの細かな機能について、具体的に考えをつめていたわけではない。

もちろん売ります・買います情報と違って、オークションの基本は、出品された品物に対して、入札者同士が入札額を提示し合うことで、競り落とすところにある。オークションとは、落札という勝利をめざして展開されるゲームなのだ。

個人間商取引（C2C）のゲーム性

Yahoo!オークションを利用してみて、よく考えられているなと感心したのは、現在進行中の入札についての履歴、すなわち、誰がどれだけの入札額を提示してきたかが見られるのはもちろん、オークション参加者の利用履歴、具体的には、その参加者がどんな品物を落札してきたのか、また、どんな品物を出品してきているのかが、確認できる点だ。オークションに限らず、ネットでの品物の売買では、売り手も買い手も互いに相手の顔が見えないし、品物そのものを目で見て確認することもできない。したがって、相手が取引相手として信用できるかどうかを判断する材料として、出品および落札に関する履歴情報が公開されているわけだ。こうした出品・落札履歴は、その人の趣味・嗜好がわかるので、実際にある品物を入札するしないに関係なく、見ていて楽しめる。Yahoo!オークションでは、使用済みの下着の出品は禁止されているのだが、使用済みの水着、

それも男性用のものが出品・落札されていたのを見たときは、世の中には、いろいろな趣味・嗜好の人がいるものだと変に感心したりもした。

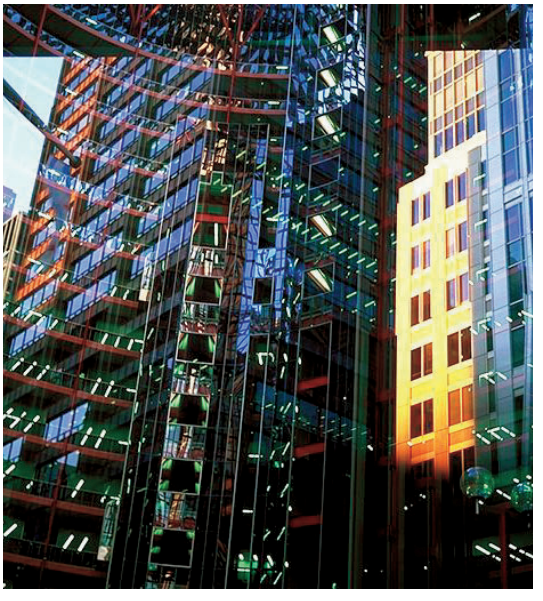
ネットオークションでは、さまざまなトラブルも続いている。ネットオークションは、ネットの中だけで完結しない。出品者は落札者からお金を払ってもらわなければならないし、落札者は出品者から品物を発送してもらわなければならない。見知らぬ他人同士の取引である以上、詐欺にとっても格好の場所である。お金を払ったのに品物が送られてこなかったり、送られてきた品物が傷んでたり、相手と連絡がとれなかったりといったことも多いようだ。そのためYahoo!オークションでは、エスクローと呼ばれる仲介サービスが導入された。これは出品者と落札者の間で代金と品物のやり取りを仲介するサービスで、Yahoo!ではなく第三者企業が提供する。手数料が発生するとはいえ、エスクローサービスの導入によって取引の安全性は向上した。

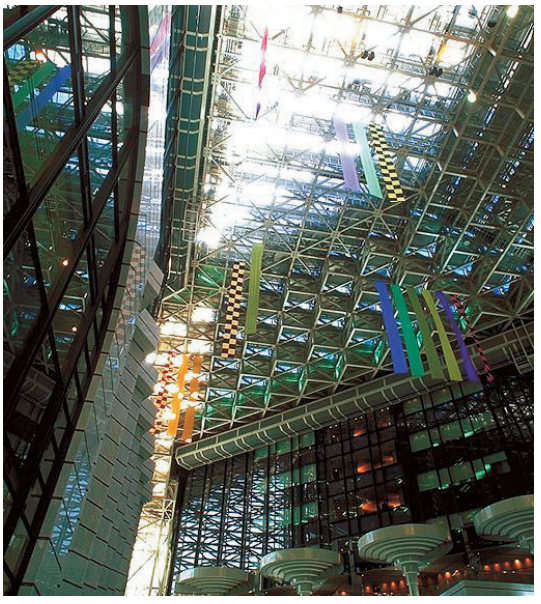
ともあれ、ネットオークションは落札者にとっては商品を買うための場であり、ちょっと特殊な通信販売と見なすことができる。すると、ネットオークションの本質は個人が出品者として商品を売る立場に立つための場、しかもかなり効率の良い場を提供している点にあるといえるだろう。

無形のサービスを循環させるには

ネットオークションは、何らかの物理的な品物を売買するための場である。それでは、有形の品物だけでなく、無形のサービスを売買することはできないだろうか。売るとは、品物やサービスを提供する代わりに、お金を受け取ることである。買うとは、品物やサービスを受け取る代わりに、お金を提供することである。もちろん、ネットオークションは、あくまで有形の品物を競り落とすものであり、その市場で無形のサービスを扱うには無理がある。ネットオークションに出品されている品物は、それがどういうものであるか、判断することがかなり容易であるのに対して、無形のサービスについては、その質を判断するのが困難である。また、品物と違って、サービスは、それを提供する側も提供される側も、ともに身体が場所と時間に拘束される。品物の取引であれば、北海道と九州のやり取りも可能だが、サービスとなると、そういうわけにはいかない。

そもそも、売ることを前提としたサービスというのは、賃労働であったり、アルバイトであったりするわけだが、個人事業を別にすれば、それは雇用関係を前提に成り立つものである。ただし、雇用関係ではない関係、お金による





報酬を前提にしたくないサービスというものもある。

たとえば、私は、CGIのプログラミングやサーバの設定といった仕事に対しては、金銭による報酬をいただくことにしている。しかし、パソコンのトラブルに関する相談を受けることが多く、そのトラブルを解決するために先方まで出かけることもあるのだが、そのような場合には、原則として金銭を要求したりはしない。とはいえ、無料より高いものはない、というように、そうしたサポートに対して、何かお礼なりお返しをしなければ、という心理が働くようだ。人によってさまざまなのだが、ケーキをもらったこともあるし、食事やお酒を奢ってもらったこともある。しかし、それが目的で相談に乗っているわけではなく、かつて自分がパソコンを始めたときも、同じようにサポートしてもらったり、困ったときに助けてもらった経験があるからなのだ。それに何より、トラブルが解決したときに相手の喜んでいる顔を見るのは楽しい。逆に、お金を払っているんだから、サポートして当然、問題を解決できて当たり前、といった態度をとられるのは嫌なものだ。

ほんとうは、誰かに何かをしたとき、その返礼として、相手から何かをしてもらえれば、それが一番いいのだろうと思う。しかし、なかなかそううまくはいかない。相手にしてあげられることと、相手にしてもらえることとが、うまく合致するとは限らないからだ。

それでは、これが二者関係ではなく、三者関係になったら、どうなるだろうか。AさんがBさんの求めに応じて何かしてあげる。そのBさんはCさんの求めに応じて何かしてあげたとする。もしCさんがAさんの求めに応じて何かしてあげられたとすれば、この三者の間で、何かしてあげる連鎖が一巡することになる。こうした連鎖を実現するための仕掛けが、地域通貨には備わっているように思う。

地域通貨とP2Pの親和性

『Linuxはいかにしてビジネスになったか』（國領二郎＝監修、佐々木裕一、北山聡＝著、NTT出版）によると、「現在では世界には2000以上もの地域貨幣が存在するといわれている」そうだ。中でも有名なのが、マイケル・リントンの考案によって1983年にカナダのバンクーバー島で創始され、現在も運営されているLETS（Local Exchange Trading System）だ。

――「たとえば、その地域に伝わる民話に詳しいおじいさんがいたとする。彼は地元の子供たちに得意の民話を披露して、子供たちからお礼を地域通貨でもらう。おじいさん

は自分の食生活に必要な食料品を買うのに、食料品については国民通貨で払うが、家まで届けてくれる地元のお店に対する配達費は地域通貨で払う。地元のお店は貯まった地域通貨を地元の日曜大工好きの人たちに一気に払って、週末に店舗を改装する」

「価値のない＝国民通貨で払いにくい」と思われるちょっとしたサービスを取引するために、コミュニティ内で循環する通貨LETSが使われているらしい。

同書ではLETSの特徴として、

(1) 現在のサービス同士をやりとりする
(2) サービスの提供者と受益者の交渉によってサービスの価格が決まる
(3) コミュニティ内の通貨量の管理を重視せず、メンバーが自由に発行できる

とまとめられている。

西部 忠は『〈地域〉通貨LETS　貨幣・信用を超えるメディア』(『可能なるコミュニズム』太田出版に収録)の中で、LETSの4原則として、

(1) 同意……参加や脱退のみならず、あらゆる取引は同意に基づく
(2) 無利子……口座の正負いずれの残高にも利子がつかない)
(3) 共有……LETSのサポートサービスを参加者の誰かが非営利のコストベースで行い、そのコストを全参加者が利用状況に応じて共同で負担する
(4) 情報公開……使用者が行為に際して情報が与えられていることを保証する

を指摘している。

このようなLETSの仕組みを知って、gnutella(ちなみにgPulpという次世代gnutellaプロトコルもあるようだ。http://gnutellang.wego.com/)に代表されるようなP2P(ピアツーピア)のアプリケーションとの類似性を直観した。おそらくP2Pのネットワークは、地域通貨を媒介にしたコミュニティ経済における互酬的で相互扶助的な特質と共通するものがあるからだろう。

個人が提供できるサービス、提供してもらいたいサービスは、個人管理下のノード上に登録し、その情報に対する検索や更新処理は、ネットワーク全体で分散処理させるアプローチが可能だろう。おそらくタイムスケジュールの調整も盛り込むことができるはずだ。各人の帳簿が勝手に改ざんされていないかチェックするには、暗号技術が使えるだろう。地域通貨とP2Pのアーキテクチャは、親和的な関係にあるはずだ。

Profile

とよふく つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』(オライリージャパン)『GIMPパーフェクトガイド』(エムディーエヌコーポレーション)などがある。

セクシーなGUIと半分の自由

文：安田幸弘
Text : Yukihiro Yasuda

あるとき、ニュージーランドのオープンソース原理主義者が、「マイクロソフト？ アップル？ フン、ちょっと『セクシー』なインターフェイスで人をたぶらかす奴らだね」なんて言っていたのを思い出したのは、先日、MacOS Xのパブリックベータ版を触った次の瞬間、「目がハート」状態になってしまったときのことである。

常日頃、「やっぱり、オープンソースのシステムが一番」とか何とか言いながら、実際、Linuxの上でGNOMEあたりを使っているわけなのだが、どっちかというと技術至上主義を感じさせるGNOMEに慣れ切ってコマーシャルソフトの世界の免疫がなくなっていたせいか、MacOS Xに搭載されている新しいインターフェイス、Aquaが放出するフェロモンにすっかりやられてしまったようだ。結局、1週間ほどMacOS Xと戯れ続けていた。

まだろくにネイティブアプリケーションもないMacOS Xだが、シェルが使えてgccが走れば怖いものはない。そこらのソースを拾ってきてmakeすれば、とりあえず不自由はなくなる。残念ながら「Make一発」というわけにはいかないのだが、そこがまた面白かったりするわけで、MacOS Xを触っているのかDarwin（MacOS Xに採用されたMach/BSDカーネル）と戯れているのかよくわからなかったりするけれど、まあそんなことはどうでもよろしい。モダンなGUIのAquaと、トラッド感覚のUNIXシェルのアンバランスもまた楽しからずや、である。

オープンソースなれど偉さも中くらい

別にここでMacOS Xを賞賛しようというつもりはないが、とりあえずカーネルだけとはいえ、自社の主力製品のソースを公開したアップルは偉い……のだけど、そもそもMacOS Xのカーネルは元祖オープンソースのMach（2.5）やBSDがベースだったりするわけで、最初から公開しとけよという気もしなくはない。それに、製品としてのMacOS Xの本当の核心部分とも言えるGUIの部分はやっぱり非公開のままなので、まあ、「偉さ」も中くらいかな。

でも、アップルが自社製品のカーネルを公開したことの意味は決して小さくはない。少なくとも、コアの部分のインターフェイスが透明になったことは、開発者やユーザーにとって、ある程度、メーカーの束縛から自由になれることを意味する。

これはMacに限らずWindowsでもコマーシャルのUNIXでもそうなのだが、非公開のバイナリ配布では、何をするにもメーカーの言う通り、何か簡単な機能が欠けていたとしても、メーカーがサポートしないと決めればそれまでだ。開発者でもない末端のユーザーは、どんなに技術があってもごく簡単な機能を付けることもできなかった。だけどオープンソースなら話は違う。

たとえばサーバOSでよくある不可解なライセンス制限。以前、Windows NT上のデータベースをインターネットで無制限に使わせる場合のライセンス料の額を聞いて、死ぬほど驚いたことがある。オープンソースのMacOS Xなら、Linuxのようなオープンソースサーバと同様、少なくともOSレベルでのライセンスなんてものは無意味になる。サーバ部門が弱いアップルだからできたのかもしれないが、これで何万円だか何百万円だかを払ってライセンス証書を買ってくる必要がなくなるわけだ。

カーネルをLinuxにしたら

それと、なんとなくアップル社自身、「誰か、やってくれないかなあ」と期待してるような気がするけど、嬉しいことがあるかどうかは別として、理屈としては2.5ベースの現在のカーネルをMach 3.0やRT-Machをベースにしたものに置き替えることもできるように思う。もしかしたら、分散処理なんかもできちゃうのかもしれない。そうした

ら、Macの得意分野でありながら、従来のMacでは遅くて不安定で悩ましかったハイエンドのグラフィックスアプリケーションを使ってる人は、泣いて喜ぶんじゃないだろうか。

さらにMach/BSDの代わりにPPC LinuxをカーネルにしてMacOS Xを動かしたり、LinuxやGNOMEのエミュレータを組み込むという可能性だって考えられる。MacOS XにXF86を乗せる作業が進められているという話は見かけたことがあるけれど、Xのアプリケーションが Aquaと共存できれば、喜ぶ人は少なくないに違いない。

そこまでディープなハックをしないユーザーのレベルでも、BSD系カーネルのインターフェイスが使えるとなれば、これまでメーカーが対応してくれないからあきらめていたようなこともできるようになる。

たとえばMacOS XはWindowsのCIFS（SMB）を使ったファイル共有ができない。AppleShareがあるんだからそれでいいじゃないかと言われればその通りなんだけど、世の中にはMacよりWindowsのほうが多いのだ。そしてNT Serverのマッキントッシュサービスを除けば、WindowsではAppleShareが使えない。アップルかマイクロソフトか、どっちかが何とかしてくれなければ—どっちも未だに何もしてくれないんだけど—WindowsとMacが混在するネットワークのユーザーはファイル共有ができない不便を覚悟しなければならなかった。

ちょっとしたテキスト処理をしたいと思っても、コマンドシェルのない従来のMacOSでは、AppleScriptあたりでノタノタとアプリケーションのウィンドウを操作してやるしか方法がなかった。

MacOSでデータベースを使いたいと思っても、MacOS用のデータベースの選択肢はものすごく限られたものだった。MacOS用のデータベースソフトといえばせいぜいファイルメーカーか、あるいはバカみたいに高いRDBを買うか。

でもMacOS Xは、このようなメーカー依存から一定程度、ユーザーを解放してくれる。土台はPOSIXなんだから、オープンソースのアプリケーションやらサーバやらが自由に使えるのだ。WindowsとファイルをやりとりしたければSambaを入れればいい。テキスト処理ならgrepでもsedでもAWKでもPerlでも好きなツールが使える。頑丈なRDBが欲しけりゃMySQLでもPostgreSQLでも好きなやつを突っ込めばいい。

はたして一般のMacユーザーがそんなことをするかどうかはわからない。だが、MacOS Xがオープンソースのカーネルをベースに動く限り、メーカーの製品開発ポリシーと関係なく、自力で問題を解決する可能性が開かれているわけだ。サードパーティにとってもビジネスチャンスかもしれない。

やっぱりGUIのソースが見たい

こうして見るとMacOS Xの未来は明るそうに思えてくるが、世の中そんなに甘くはない。「MacOS」として一番重要なコンポーネントであるGUI部分が公開される気配はない。アップルとしても、あのセクシーなGUI部分は技術的な面ばかりでなく、経営戦略の面でも命綱なのだろうからそう簡単に公開するというわけにはいかないのかもしれない。オープンソースにすればすべて解決するわけでもないとしても、たとえばSolarisのライセンスのような形態でも、GUI部分のソースを公開する可能性はないものだろうか。

そういえば昔のApple IIのマニュアルには、モニタのソースが当たり前のように載っていたよね、ジョブスさん？

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

10・4 MaxtorがQuantamを買収
10・2 MSがCorelに出資
9・29 インテルTimna計画を放棄
9・27 AOLとNTTドコモが提携
9・19 SUNがCobaltを買収

このところ事情があって、何枚か新しいマザーボードを買ったので、我が家のLinuxサーバのマザーボードを古くなったマザーボード（とはいっても、インテルの440BX採用のもの）と交換した。いままで、MMX Pentium233MHzというスペックでLinuxを使っていたのだが、やはりサーバが速くなると、IMAP4やHTTPの反応が良くなった。なんだか今まで損していたような感じ。もっと早く新しいマザーボードに交換すればよかった……。

今月もインテルはニュース続き

インテルは、統合型CPUであるTimna（コードネーム）の計画を放棄したようである。このCPUは周辺回路などを統合したもので、低価格マシン用に計画された。ただし、当初の計画ではRambusメモリを使うようになっていたので、急遽、Rambus用の信号をSDRAMに変換するチップセットを開発して組み合わせる予定だった。ところが、それではチップの数という点では現在出回っている810／810Eチップセットなどと同じになってしまうのである。そうなると、部品数や基板面積の縮小という点でメリットが少なくなってしまう。

もうひとつの理由は、Celeronを使ったマシンの価格がAMDとの競争で下がってきており、Celeron搭載マシンが本来Timnaがカバーすべき領域にまで食い込んできているためだ。つまり、商売的に見ると、Timnaを出しても、Celeronの分を食うだけとなってしまう。

だいたい、インテルは、Pentium4出荷後も、PentiumIIIを存続させる予定で、そうなるとCPUの系列が4系列となってしまう。ところが実際には、PCは低価格帯とその上のメインストリーム帯があるだけで、インテルなどがあるというワークステーションクラスのパソコンの市場などはなきに等しい。というのは、メインストリーム帯のCPUがすでに1GHzクラスとなっているからである。そうなると、Pentium4が登場すれば、当然PentiumIIIは下に下がらざる得なくなり、これに押されてさらにCeleronが下がり、結局Timnaの存在場所がなくなってしまうからである。

さて、インテルは、そのPentium4の発表を遅らせたようである。技術的な問題があったらしい。まあ、発表は第4四半期中というのが当初からの予定で、公式には遅れたことにはなっていないようであるが……。

アップルも悪いニュースが続く

アップルコンピュータは、2000年度の第4四半期の業績予測を下方修正した。アップルによると、「世界的な景気の失速」、「教育市場の需要が低下」、「G4Cubeの立ち上がりが悪い」の3つが原因だとか。そのG4 Cubeだが、あの透明なケースについた「線」がユーザーの間で話題を呼んでいる。アップルによれば、これは製造上どうしても入るものであって、傷ではないということだが、あの透明な姿に惚れて買ったユーザーには収まりがつかないようである。また、一部の機種に電源が勝手に入ったり切れたりする現象が見られ、こちらも怒ったユーザーの一人が日本のYahoo!オークションに売りに出し、その掲示板で欠陥とアップルの対応について文句を言ったところ、あっという間に同じような現象に悩んでいる人や、これから買おうとする人からのメッセージが書き込まれ、ちょっと

した事件になった（オークションサイトって掲示板の代わりにもなるんですねぇ）。とりあえずアップルは、G4 Cubeの売上げを伸ばすために、キャッシュバックを行う予定。

Macユーザーによると、アップルの対応には疑問に思うことが多いとか。そういえば、アップルコンピュータに対する日本の公正取引委員会の調査が終了して、アップルコンピュータは、独占禁止法の規定に違反するおそれがあるものとして警告を受けた。詳しくは、公正取引委員会のホームページにある文書（http://www.jftc.admix.go.jp/pressrelease/00.october/001003.htm・**画面1**）を見ていただきたい。これによれば、

「……同法第19条（(1) ア及び (2) の行為は不公正な取引方法第12項第1号〔再販売価格の拘束〕に、(1) イの行為は不公正な取引方法第13項〔拘束条件付取引〕に、それぞれ該当）の規定に違反するおそれがあるものとして、今後、このような行為を行わないよう警告を行った」

となっているのに対して、アップルコンピュータのホームページにあるリリースには、こんな文章が書かれている（http://www.apple.co.jp/news/2000/oct/03jftc.html・**画面2**）。

「この調査の終結には、アップルが事業活動を日本の法律を遵守して行っているというわれわれの見解が反映されています」

公正取引委員会の文章は、どう読んでも、アップルコンピュータが「法律を遵守して」いるとは読めないのだが……。

マイクロソフトは幹部が次々と辞職

司法省との裁判が始まり、ビル・ゲイツは経営から手を引き、スティーブ・バルマーが社長になったという体制の変化もあるのだろうが、マイクロソフトの役員が次々と辞めている。そして、創業メンバーの一人であるポール・アレン氏もついにマイクロソフトを引退することになった。もともとアレン氏は億万長者だし、マイクロソフトの仕事には興味がなくなったのかも。

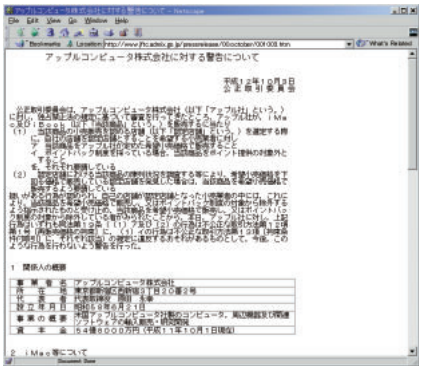
また、14年間勤めていたポール・マリッツ副社長も退職するとか。これで、マイクロソフトの役員から古株がいなくなったわけで、ある意味バルマー社長の体制固めができたともいえる。ただ、意地悪く見れば、裁判で評判も下がったし、マイクロソフトの株もここいらが売り時と見て役員が辞めているのかもしれない。

裁判のほうは控訴して、とりあえず分割についてはすぐには行われない状態となり、マイクロソフトの思惑通り、最高裁での裁判ではなく控訴裁判所での裁判となったようだが、この裁判、かなり長期化する可能性が出てきた。しかも、アメリカは大統領選挙を控えている。もし、ここで政権がブッシュに代わるようなら、この裁判も今後どうなるかわからなくなってきたからである。ちなみに、同じように独禁法で分割などが検討されたIBMの裁判も、レーガン政権時代に継続は無意味として提訴取り下げになっている。

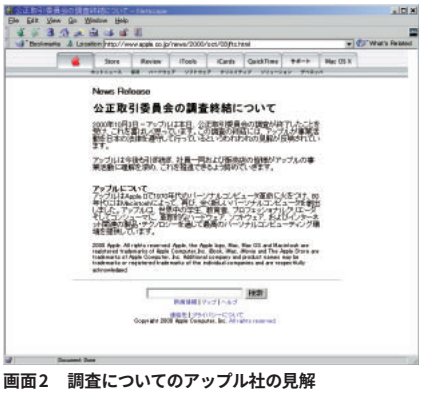
そのマイクロソフトだが、ライバルであるCorelに1億3500万ドルを投資した。これは、このところ調子の悪いCorelを救うためである。インプライズとの合併計画がおじゃんになり、今年の夏には、業績不調の責任をとって、創業者でもあるコプランド博士が辞任した。そのあとを継いだ暫定CEOがマイクロソフトの社員と懇意だった関係で、出資となったようである。

コプランド博士は、他の経営者同様、マイクロソフトをずっと批判していたが、その人物がいなくなり、財政的に危機に陥ったCorelに助け船を出した格好である。裏があるのかどうかは知らないが、なんでも、マイクロソフトは、.NET計画のため、CorelにLinuxをターゲットにした移植を行わせる予定があるのだとか。

ほんとうにLinux用にマイクロソフトがプロダクツを移植するつもりがあるのかどうかは知らないが、ひとつの選択肢として、裁判対策には有効なのかもしれない。



画面1　アップル社に対する公正取引委員会の警告
http://www.jftc.admix.go.jp/pressrelease/00.october/001003.htm



画面2　調査についてのアップル社の見解
http://www.apple.co.jp/news/2000/oct/03jftc.html

日刊アスキーLinux on Linux magazine

http://www.linux24.com/

日刊アスキー Linuxの舞台裏

～VA Linuxが日本で本格始動～

米VA Linux Systemsの日本法人「VAリナックスシステムズジャパン」が、活動を本格的に開始する。日刊アスキーLinuxでは、さっそく同社にインタビューをして、今後の活動などについて記事にすることができた。

今回はこれまでたびたび日刊アスキーLinux上に登場したVA Linuxの記事を振り返りつつ、VA Linuxのありようと日本法人の今後を展望してみよう。

（日刊アスキー編集部・吉川大郎）

VA Linux Systemsについては、昨年秋のLinuxWorld Expo/Tokyo'99でCEOのLarry Augstin氏にインタビューしたときから現在まで、何度も記事にしてきた。まずは、日刊アスキーLinuxで取り上げた記事一覧をご覧いただきたい（**表**）。

これを見ると、やはりCEOであるLarry Augstin氏の話題が多い。逆に、VA本来の業務である製品についての話題がないのは、やはり日本ではVAの新製品は馴染みが薄かったせいもあるだろう。VAリナックスシステムズジャパンができることで、今後は同社製品がもっと多く日本国内の目に触れることになると思われる。

今後ニュースに登場する頻度が高くなると思われるVAリナックスシステムズジャパンだけに、ここで本社VA Linux Systemsの全体像と、インタビューから得たVAリナックスシステムズジャパンの今後の方針とをご紹介する。

まず、米VA Linux Systemsの特徴をあげてみよう。

1. Linuxに特化したハードウェアを製作、ダイレクト販売のほかコンサルティングも行う

2. SourceForge、Freshmeat、Linux.Com、Slashdotを含む一大オープンソースポータルサイト「OSDN（Open Source Development Network）」を運営

3. CEOのLarry Augstin氏は、Linux関連のイベントでは必ずといっていいほど講演を行うキーマン。また、表にもあるとおり、Eric S.Raimond氏が在籍するなど、オープンソース関連の人材も強力

VAの製品群

VA Linuxの製品は、Linuxを組み込んだサーバやワークステーション、ストレージ（NAS：Network Attach Strage）などが用意されている。サーバ関連は、現在流行している1U／2Uからデスクトップタイプまでがラインナップされているが、特徴的なのが（当然といえば当然だが）導入してすぐにLinuxが動く環境にあるということだ。これは、たとえば既存のPCサーバを購入して何らかのディストリビューションをインストールしてチューニングして……といった手間を省くことができる。最近では大手を中心に多くのPCサーバベンダーがLinuxをサポートするに至ったが、VA LinuxはLinuxに長く関わっているだけあって、安心感は高いだろう。米国では他社へのOEMも行われている。

Web関連の活動

Linux関連でVA Linuxをもっとも印象づけているのが、OSDNを筆頭にした開発者向けWebサイトの運営だ。上記のサイトが、それぞれ独立して多大なる人気を誇っているのは周知のとおりである。一応紹介しておくと、Linux.ComはLinux関連のポータルサイトで、ニュースやTips、コラムなどが掲載されている。日刊アスキーLinuxも同サイトと提携し、セキュリティ関連のコラムを掲載している。

SourceFogeはCVSによるバージョン管理が可能な開発者向けASPサービス（これはUNIX系に限らず、Windows系の開発にも用いることができる）で、現在DebianのWebページのような国際化が進んでいる。http://sourcefoge.net/に接続すると、日本語のメニューが表示されるので（中身は英語だが）、ご覧になっていただきたい。

Slashdotも有名な書き込み系（？）ニュースサイトだ。いろいろな人が書き込むので、とにかく情報は早い。FreshmeatはLinux関連ソフトウェアのインデックスサイトである。国内では、日経BP社が「Freshmeat-J（http://bizit.nikkeibp.co.jp/it/fresh/）」を運営している。

VAリナックスシステムズジャパンの今後

上記のような米国本社の活動に対し、VAリナックスシステムズジャパンは国内でどのような展開を図っていくのだろうか？ 大前提として最初はサーバ製品をしっかりと売っていくとのことだ。Web関連の展開はいずれは何かしらやりたいとのことだが、まずはサーバビジネスに注力していくという。前述した1U／2Uのサーバやストレージ製品など、同社が得意とする分野から展開しようというわけだ。

現在すでに、VAリナックスシステムズジャパンのパートナー企業である東芝エンジニアリングにより、東京農業大学のWeb関連サーバとして導入されている。マシンは2Uタイプの「VA Linux 2200」である。この事例は来月少し詳しく紹介したいが、やはり安心してLinuxを使用できるという点が、導入のポイントになったとのことだ。

●

Linux関連の大物といっても、VA Linux Systemsはサーバ関連の業務がメインなので、ディストリビュータが上陸した場合に比べると一般の関心はそれほど高くないかもしれない。だが、オープンソース企業という側面からも、Linux専門ハードウェアメーカーという面からも、今後国内のLinux業界がますます拡大していく中で、キープレイヤーとしての役割りを果たしていく企業であることは間違いがない。

VA LiNUX Systems社長兼CEO、Larry M.Augstin氏の講演
1999年10月4日
http://www.linux24.com/linux/news/today/article/article289504-000.html
LinuxWorld Expo/Tokyo '99にて、オープンソース界でも有名なLarry Augstin氏が、ソフトウェア開発においてオープンソースを活用する利点や、企業が自社のソフトウェアをオープンソースにする場合の注意点などを紹介。ビジネスにおけるオープンソースの活用を語った。面白かったのが、Q&Aコーナーで語ったRed Hat Linuxに関するエピソードだ。オープンソースとビジネスはあまり結びつかないのではないかという問いに対して、彼はBob Young氏のエピソードで「フリー版を配付すると、その結果パッケージの売り上げが上がる」という現象を紹介。これをもって、オープンソースビジネスでは、既存のビジネスモデル流の見方を変えねばならない点を説明した。

VA Linux Systems社長兼CEOのLarry Augustin氏に聞く
1999年11月3日
http://www.linux24.com/linux/news/today/article/article313984-000.html
日刊アスキー Linuxが、LinuxWorld Expo/Tokyo '99のために来日していたLarry Augstin氏にインタビュー。掲載日は11月だが、インタビューは上記講演直後に行っている。彼が宿泊したホテルの一室にて、くつろいだ雰囲気でインタビューが行われた。途中でAugstin氏が自分のノートPCを日刊アスキー Linuxに見せる下りがあるが、実際彼はLinux大好き人間といった印象で、Linuxの話題についてひとしきり盛り上がった。性格的にも茶目っ気のある人物といった印象だ（記事に掲載している彼の写真を見ていただければわかると思う）。
このインタビューは、のちにVAリナックスシステムズジャパンを立ち上げることになる住友商事の余語氏、上田氏にご協力をいただいて実現した。

米VA Linux SystemsのIPO（株式公開）―初値は299ドル
1999年12月10日
http://www.linux24.com/linux/news/today/article/article340672-000.html
Red Hatに続いてVAもIPOを実現。公開当初から株価がうなぎ昇りになった。公開株価30ドルに対し、初日終値は239.25ドル（2000年10月19日現在は35.625ドル）。次にどの企業がIPOをするか、といったことがよく話題になった。

思いがけない財産
Eric S. Raymond ― Surprised By Wealth （Dec 10, 1999, 07:10 UTC）
Eric S. Raymond [著]、日下部圭子 [訳]
2000年1月20日
http://www.linux24.com/linux/linuxtoday/article/article366112-000.html
VA Linux SystemsのIPOによって巨万の富を得たEric S.Raymond氏のコラムがLinuxTodayに掲載された。ここで彼は、なぜ自分がこのコラムを執筆するのか、IPOによって得た資金をどのように使うのかといった点に触れた。

LinuxWorld Conference & EXPO New York レポート その3
2000年2月5日
VA Linux Systems社長　Larry Augstin氏講演
http://www.linux24.com/linux/news/today/article/article376608-000.html
日刊アスキー Linuxでコラムを執筆中の宮原 徹氏が、LinuxWorld Conference & Expo New Yorkをレポート。その中にAugstin氏の基調講演のもようが含まれている。また、VA Linux Systemsのもうひとつの顔といってもよいWeb関連の展開についても紹介している。

LinuxWorld Expo/Tokyo 2000展示会場レポート（その1）
2000年5月12日
http://www.linux24.com/linux/news/today/article/article451936-000.html
このLinuxWorld Expo/Tokyo 2000において、ついにVAリナックスシステムズジャパンの設立が発表された。開催当日の新聞発表を見て会場に入ると、入り口付近に大きなVA Linuxの看板が設置され、VA Linux本社や住友商事のスタッフが忙しそうに動き回っていた。また、VA Linuxのブース以外でも、パートナー各社が製品を展示し、それぞれのVA Linux製品を使用したソリューションを紹介していた。

MySQL、商用利用に制限のあるライセンスからGPLに変更
2000年6月29日
http://www.linux24.com/linux/news/today/article/article491104-000.html
PostgreSQLとともにフリーのRDBMSとして知られる「MySQL」のライセンスがGPLに変更になったニュース。このMySQLに対し、VA Linux Systemsが出資することも同時に発表された。同時に、SourceForge.netでMySQLの開発をサポートするという。

表　これまでに掲載された記事


LinuxWorldが終わり、我々のインタビューを受けてくれたLarry Augstin氏

LinuxWorld Expo/Tokyo 2000のVA Linux Systemsブース

正しきタコへの道

初級Linuxer養成講座

第15回　シェルの小技〜迷ったらTabキーを押せ

今回は、システム寄りの難しい話から少し離れて、Linuxのコマンドラインで「生活」するのにあたって、知っておくと便利な「小技」を紹介する。「迷ったらTabキー」、これだけ覚えておけば、「GUI」のしがらみから解放されて、Linux「初心者」脱出のゴールが一気に近くなる。

文：竹田善太郎
Text : Zentaro Takeda

会社勤めを辞めて自宅でひとりで仕事をするようになってから、毎日頭を悩ませているのが**昼飯**の問題である。都心のオフィス街なら、だいたいは**いきつけの店**みたいなものがあって、それほど困ることはないと思うのだが、筆者が仕事をしている自宅はどちらかといえば郊外にあるので、近所には**ファミレス**とか、**会員制そば屋**のような店しかない。それに、原稿仕事では身入りの少ない時期も多く、あまり外食ばかりもしていられないので、**自炊**することが多くなった。

とはいえ、自由業だからといっても、昼食ごときに余分な時間をかけるわけにはいかないので、**インスタント食品**や**レトルト食品**のお世話になることが多い。このようなお手軽食品に、こまかい文句をつけるのは大人げないとは思うのだが、どうも最近、**奇妙な味**のする品が多いのが気になる。見ためは良く、おいしそうな匂いもするのだが、口にすると、なんだか**ぬるぬる**する感触があるのに、それらしい「味」がしない。無味無臭のぬるぬる物質が舌の上を覆い、その向こう側を塩味や化学調味料の味が**素通り**するという感じなのである。舌に触れずに直接胃袋に流れ込んでしまうので、なんだか食べた後に胸焼けとも膨満感ともつかない、嫌な感じばかりが残ってしまう。インスタント食品ならすべてそんな感じがするのか、というとそうでもなくて、きちんとした**食べ応え**のあるインスタント食品もある。だから、筆者の味覚や体がおかしい、ということではなさそうだ。

あるとき、食感がよくない製品の原材料をみると、きまって**キサンタンガム**や**増粘多糖類**などの項目があるのに気が付いた。これらの成分は、食品に**粘り気**を加えたり、成分が分離してしまうのを防止するためのものだそうだが、食品の味を**安っぽくする**副作用もあるようだ。

これは筆者の想像でしかないので、間違っていたら申し訳ないのだが、これらの増粘剤を入れると、ソースやスープの**からみ**がよくなり、口当りもなめらかになるので、加工食品の**もっともらしさ**を強調するために多用されているのではと思っている。この「もっともらしさ」を専門用語では**調理感**というらしい(その対極は**素材感**)。マスプロ食品のメーカーの会議室では、新商品の企画をするときに、Jリーグ選手みたいなぼさぼさの髪型をして、妙に平べったい色つきメガネをかけたマーケティング担当者が、**今度の新製品は調理感を強調して**……なんてことをプレゼンし、ダークスーツの製品企画担当者が**ではソースにとろみを**……と受け、白衣の開発担当者が**では、キサンタンガムを増量**……と締めくくるようなやりとりがなされているのでは、と疑っている。

筆者は、食品添加物を必要以上に忌避する**自然食品信奉者**ではないし、逆に食品添加物のありがたみもわかっているつもりだ。増粘剤についても、油脂の代用品として食餌療法に使ったり、食品の嚥下を補助するために使われていることも知っている。しかし、食品の外見や舌触りだけを良くして、中身の貧相さをごまかすために、多量に使うのはどうかと思う。なにより、**まずい**ものを買わされるのは勘弁してほしいのだ。

聞くところによると、清涼飲料水やインスタント食品は、次々と新製品を送り出しては膨大な広告費を費やしているが、それでも、消費者が**ためしに1つ買ってみる**だけで十分に採

算が取れるようになっているらしい。消費者が味に失望して**もう、二度と買うものか**と思っても、後の祭りなのである。

インスタント食品の「製品寿命」は、ごく一部の定番商品を除いて、数カ月から、短いものでは数週間なのだそうだ。製品の移り変わりが激しくて、ついていくのが大変だと思っていたパソコン業界も、それから見ればかわいいものなのである。しかし、特にノート型PCの分野では、余計な**添加物**（アクセサリ）をごてごてつけたり、**みてくれ**だけの製品でユーザーをごまかそうするような動きがないわけでもない。そういう製品を出すメーカーは、製品の寿命（新製品が出るまでのサイクル）も短い傾向があるようだ。でも、10万円単位のパソコンを**インスタント食品化**するのは、できればやめてほしい。

キーボード操作を快適に

Linuxに抵抗感を持つユーザーの多くは、**複雑なコマンドを入力する**のが難しくて面倒と感じているようだ。とくに、Linuxの場合はファイルに長い名前をつけたり、複雑なディレクトリ階層の中にファイルを保存することが多いので、どうしてもコマンドラインに入力する文字の数が増えてしまう。たとえば、

```
$ /usr/local/bin/emacs /home/httpd
/cgi-bin/mydata/index.cgi
```

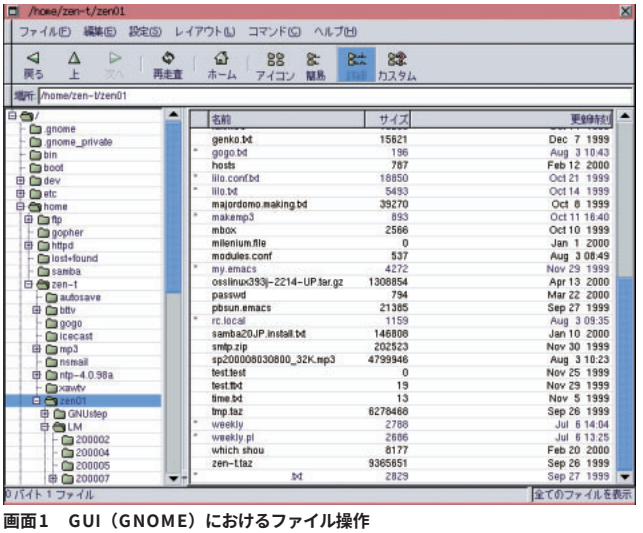
などというコマンドラインを入力することを考えるだけで、頭がくらくらとしてしまうユーザーもいるかもしれない。実際は、ここまでひどくはなくて、たとえばコマンド名の入力については、通常は**コマンドパス**があらかじめ設定されているので、

```
$ emacs
```

だけで目的のコマンド（プログラム）をLinuxのシェルが見つけ出してきてくれるようになっている（コマンドパスについては、いずれ詳しく説明する）。しかし、コマンドに渡したいファイル名などは、カレントディレクトリのファイル以外の場合は、相変わらず長ったらしいディレクトリ名つきのファイル名（これを**フルパス名**と呼ぶ）で入力しなければならない。

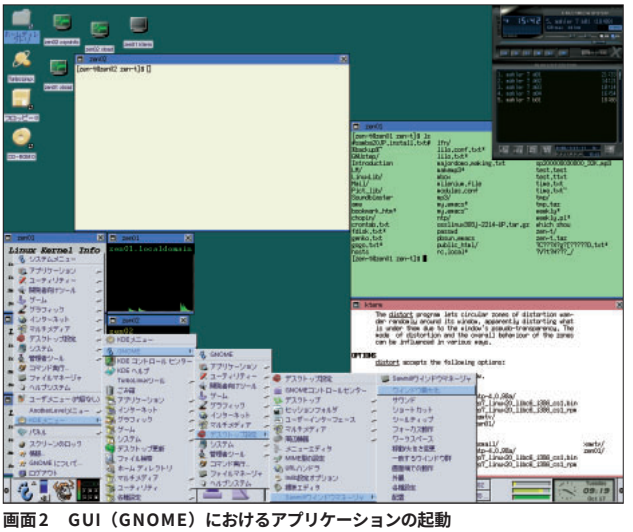
各種ウィンドウシステム、すなわち「GUI」の最大の利点は、このような長いコマンドラインをキーボードから入力しなくて済むことであるのは、読者の皆さんもご存知だろう。**画面1**のようなウィンドウで、ディレクトリを開いたり閉じたりしながら目的のファイルを探し、ダブルクリックしたりドラッグアンドドロップしたりするだけで、そのファイルをアプリケーション（コマンド）で開くことができる。見たままの直感で操作できるわけだ。

しかし、Linuxの扱いに慣れてくると、いずれはこのようなGUIによる操作を**まだるっこしい**と感じるようになる。1つのディレクトリ中に大量のファイルがあるような場合など、膨大なリストの中から、**自分の目で**目的のファイルを探し出さなければならなくなるので、負担は相当なものになる。もちろん、GUIのファイルツールには、たいてい**ファイル検索機能**



画面1　GUI（GNOME）におけるファイル操作

WindowsやMacintoshと同様に、Linuxの世界でもGUIによるファイル操作やコマンド操作があたりまえになりつつある。目で見たとおりに操作できる（いわゆるWYSIWYG）が「ヒトにやさしい」ということになっているが、大量のファイルが並んでいる中から目的のファイルを見つけ出すのは、なかなか大変な作業である。



画面2　GUI（GNOME）におけるアプリケーションの起動

この画面は、「やらせ」に近いかもしれないが、インストールするアプリケーションが増えれば増えるほど、アプリケーションメニューの内容は複雑になって、目的のアプリケーションまでたどり着くのはひと仕事になってしまう。これがはたして「ヒトにやさしい」インターフェイスと言えるだろうか？

もついているので、それを使えば少しは楽になるはずだが、それにしても、メニューから検索機能を起動して、検索用文字列を入力して……、という操作は必要になり、**それほど省力化できない**場合も多い。インストールされているアプリケーションの数が増えると、メニューから目的のアプリケーションを見つけ出すという操作も、大変なものになってくる（**画面2**）。これについても、**ショートカット**（GUIシステムによって呼び方はまちまちだが）機能を使えばよいのだが、Linuxを使い始めたばかりのユーザーが、はたして自分でショートカット機能を使いこなせるかどうか、筆者は疑問に思っている。

これに比べると、コマンドラインの世界はもっと単純である。**操作の対象**となるのは、キーボードだけなので、**広大な画面の中から目的のアイコンを見つけ出す**という作業は必要ない。たかだか、数十個のキーを指でたたくだけだし、慣れてしまえばキーボードなど見ずに、しゃべるのと同じような感覚で入力できるようになるのである。長年UNIXやMS-DOSを使い続けてきたベテランユーザーはもちろんだが、GUI環境が一般的になってからLinuxを使い始めたユーザーであっても、キーボードとコマンドラインによる操作を習得できれば、Linuxをもっとスマートに使いこなせるようになるのだ。

そして、現在のLinuxのコマンドラインは、昔のUNIXとは比べ物にならないほど快適になっている。シェルの**自動補完機能**（英語では“Auto Completion”）を活用すればよいのである。「自動補完機能」とは、またずいぶん堅苦しい言葉だが、要するに、長いコマンド名やファイル名を入力するときに、シェルがその入力を助けてくれる機能だと思えばよい。難しいキー操作を覚える必要もなくて、**迷ったら、とにかくTabキーを押してみる**ということだけ頭において置けばよい。実際にはTabキー以外のキー操作もあるのだが、覚えなくても自動補完機能の恩恵は十分受けることができる。たとえば、

```
$ yuvsplittoppm  /home/lmuser/pictures/capture01 640 480 >/home/lmuser/pictures/capture01.ppm
```

という長大、かつ暗号のように複雑なコマンドラインを入力するとき、

```
$ yuvsp
```

まで入力してからTabキーを押すと、自動的にコマンド名が補完されて、

```
$ yuvsplittoppm
```

という部分まで入力された状態になる。続いて、ファイル名を入力する部分についても、すべての文字を入力する必要はなくて、たとえば、

```
/ho[Tab]lm[Tab]pic[Tab]cap[Tab]01
```

というように入力するだけで（[Tab]の部分はTabキーを1回押すことを意味する）、ファイル名の部分も補完されて入力される。ここまでで、すでに**45文字**のコマンドラインを入力するのに、**23文字**のキー入力で済んだことになる。**省力化率**（？）は、なんと51%である。「640 480」などのパラメータ部分については、自分で入力する必要があるが、その後のリダイレクト先のファイル名を入力するときも、同じように自動補完機能が使える。結果として、100文字近くのコマンドラインを入力する場合でも、実際のキー入力は数十文字で済むことになるのだ。

自動補完機能の利点は、入力するキーの数を減らすことだけではない。ミスタイプによる誤操作を防止する働きもあるのだ。たとえば、前出の「yuvsplittoppm」という舌を噛みそうな名前のコマンドは、画像ファイルの形式を変換するコマンドの一種なのだが、この呪文のようなコマンド名をキーボードから間違いなく入力するのは難しい。1文字でも間違えてしまえば、command not foundというエラーメッセージに直面することになるし、ファイル名の入力を間違えれば、目的外のファイルを変換してしまったり、別のファイルにデータを書き込んでしまったりすることになりかねない。しかし、自動補完機能を使っていれば、必ず正しいコマンド名が入力されるし、最初の入力の時点で、

```
$ yvusp
```

などと間違った入力をしていた場合は、Tabキーを押してもコマンドが補完されないので、入力ミスに気づくことができる。

自動補完機能は、昔は一部のシェル（tcshなど）にしか備わっていなかったが、現在ではLinuxで事実上の標準シェルとなっている「bash」でも高度な補完機能が使えるようになっている。ここからは、bashの自動補完機能について、もう少し詳しく見てみよう。

ファイル名の補完

bashの自動補完機能を使うもっとも一般的な場面は、コマンドに与えるフ

ファイル名を入力する場面だろう。たとえば、カレントディレクトリにある「approved.txt」というファイルをlessコマンドで表示させたい場合、

```
$ less a
```

まで入力した段階でTabキーを押してみる。もし、カレントディレクトリの中に、名前が「a」で始まるファイルがapproved.txtしか存在しない場合なら、画面上のコマンドラインは、ただちに、

```
$ less approved.txt
```

という状態に変わる。ここでEnterキーを押せば、lessコマンドでapproved.txtファイルが表示されるようになる。

もし、approved.txtのほかに、たとえば「angel.jpg」というようなファイルが存在する場合には、Tabキーを押した時点でビープ音が鳴るはずだ。これは、bashが**この情報では補完するのに不十分**ということを訴えているのである。このような場合は、

```
$ less ap
```

まで入力した段階で、もう一度Tabキーを押せば、完全にファイル名を補完することができる。同様に、カレントディレクトリに、さらに「apple.doc」というようなファイルもあった場合なら、

```
$ less appr
```

まで入力してからTabキーを押せばよい（**図1**）。

■「候補」の表示

しかし、カレントディレクトリにあるファイルの数が多く、いちいちlsコマンドなどでどのようなファイルがあるか把握するのが面倒な場合、ファイル名をどこまで入力すれば補完できるかすぐにはわからないこともある。このようなときには、

```
$ less a
```

まで入力した時点で、**Tabキーを2回**押してみるとよい。すると、画面上には、名前が「a」で始まるファイルの一覧が表示されてから、途中まで入力された状態のコマンドラインが、あらためて表示されるはずだ。

```
$ less a[Tab][Tab]
angel.jpg      apple.doc      approved.txt
$ less a
```

これを見れば、approved.txtを入力するのなら、「appr」まで入力すれば補完できることが判断できるだろう。

■「拡張子」による補完

自動補完機能が便利な場面の一例として、**「拡張子」**だけが異なる同一名のファイルがたくさんあり、その中のひとつをコマンドから利用したい場合がある。厳密にはLinuxのファイルには拡張子というものはないのだが、Windowsのファイルシステムと同じように、「.」（ピリオド）に続く数個の文字列のことを拡張子と呼んでいる。拡張子とはなにかとか、その使い分けの方法については、あらためてここで述べるまでもないだろう。

たとえば、angel.jpg、angel.txt、angel.mp3、angel.idxというようなファイルがカレントディレクトリにあるとしよう。この中の、angel.mp3というファイルを、mpeg123コマンドで再生したいとしよう。

```
$ mpeg123 angel.mp3
```

このコマンドラインは、全部で17文字になるが、補完機能を使えば次のように12文字で入力できる。

```
$ mpeg123 a[Tab]m[tab]
```

「a」を入力してTabキーを押すと、「angel.」まで入力された状態になる（この段階でTabキーを2回押せば、angel.で始まるファイルの一覧が表示される）。入力したいのは「.mp3」という拡張子なので、次に「m」を入力して再びTabキーを押せば、「angel.mp3」まで入力された状態にな

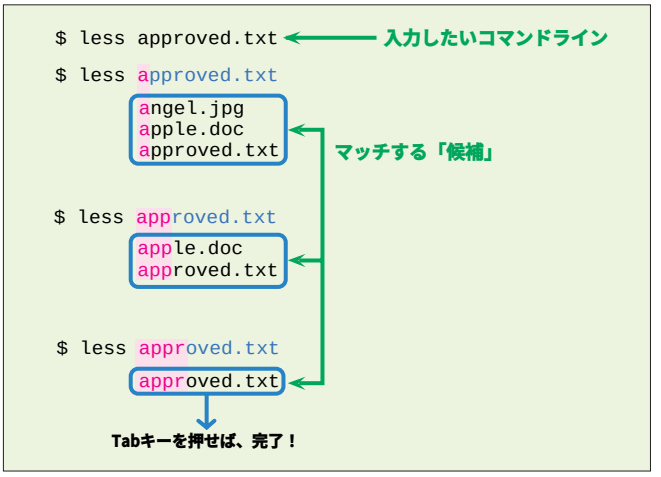


図1　bashの自動補完機能を使う
「ファイル名の先頭の数文字を入力して、Tabキーを押す」というのが、自動補完の基本操作である。同じ文字（列）で始まる複数のファイルがディレクトリ中に存在する場合は、ビープ音が鳴るので、さらに文字列を追加して「候補」の数を絞り込んでゆく。候補が1つしかない状態になれば、Tabキーを押した時点で完全なファイル名が入力されて「補完」は完了するのだ。

るのだ。同様の手順で、たとえば「angel.txt」ファイルを開きたいのであれば、「angel.」まで入力された状態で「t」を、「angel.idx」なら「i」を入力してからTabキーを押せばよい（**図2**）。

■複雑なファイル名

長大で複雑な名前をもつファイルを扱うときは、とくに便利である。Linuxの場合、アプリケーションなどのバイナリパッケージをダウンロードすると、たとえばRPM形式のファイルの場合は、次のようにとても長いファイル名がついていることが多い。

```
apache+php-1.2.6+2.0.1-2TL.i386.rpm
```

このようなファイルの名前を、コマンドラインで間違いなく入力することなど、**ほぼ不可能**だ。しかし、シェルの自動補完機能を使えば簡単である。このRPMファイルをシステムにインストールする場合なら、

```
# rpm -i apache
```

まで入力してから、Tabキーを押せばよい。もし、同じパッケージの複数のバージョンのファイルがある場合でも、Tabキーを押した時点で、複数のファイルで名前が共通している部分まで入力された状態になるので、適宜バージョン番号などを補ってやればよい。たとえば、apache-1.3.2-1TL.i386.rpmとapache-1.3.3-1TL.i386というファイルがある場合なら、

```
# rpm -i a
```

まで入力してTabキーを押すと、

```
# rpm -i apache-1.3.
```

というように、バージョン番号が共通する部分まで表示された状態で、ビープ音が鳴る。ここで、「1.3.3」のほうをインストールしたいのなら、

```
# rpm -i apache-1.3.3
```

のように、最後の「3」を補ってやって、もう一度Tabキーを押せば、

```
# rpm -i apache.1.3.3-1TL.i386.rpm
```

まで一気に入力することができる。

ファイル名の自動補完は、使い慣れるとどんな場面でも手放せなくなる機能だが、Linuxの場合、**個人的に一番ありがたみを感じる**のは、このようにRPMパッケージをインストールするときだ。

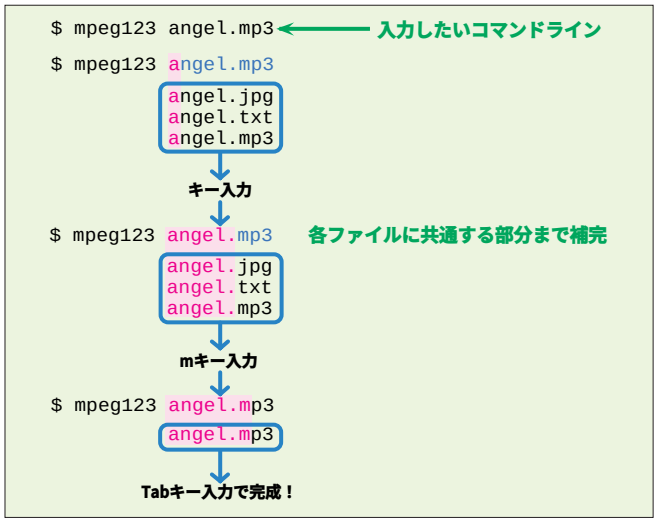


図2　「拡張子」が異なるファイル名の補完
ファイル名の先頭部分が同じで、いわゆる「拡張子」にあたる部分だけ異なるファイルが複数存在する場合、先頭部分の補完が完了したあとで、「拡張子」の先頭文字を入力して改めてTabキーを押せば、ファイル名全体の補完が完了する。実際は、bashが「拡張子」を認識しているわけでなく、図1の場合とまったく同じ処理をしているだけだが、人間の側ではそのように意識して補完操作をすると理解しやすいだろう。

パス名の補完

名前を入力したいファイルが、カレントディレクトリ以外のディレクトリに存在する場合、コマンドラインには**パス名**を入力することになる。先に触れたように、このような場合にも自動補完機能が有効である。bashの自動補完機能では、ディレクトリ名の一部を入力してやれば、それに該当するディレクトリ名を自動的に補完してくれるのだ。

たとえば、「/home/httpd/cgi-bin」ディレクトリにある「myindex.cgi」というファイルをlessコマンドで表示させたい場合、

```
$ less /h
```

まで入力した段階でTabキーを押すと、

```
$ less /home/
```

というように、最初のディレクトリ階層「/home」まで入力された状態になる。続いて、

```
$ less /home/h
```

まで入力してTabキーを押すと、

```
$ less /home/httpd/
```

続いて、「c」を入力してTabキーを押すと、

```
$ less /home/httpd/cgi-bin/
```

となり、最後に、「my」などと入力してからTabキーを押せば、目的のコマンドライン、

```
$ less /home/httpd/cgi-bin/myindex.cgi
```

が入力できる。

パス名を自動補完で入力する場合、ディレクトリ名の後の区切り文字「/」（スラッシュ）がついた形で補完されるので、**ディレクトリの階層を下りながら、頭の数文字だけ入力してTabを押す**という操作をテンポよく繰り返せば、目的のファイルにたどり着けるのである。なお、パス名の補完は「./」や「../」などで始まる**相対パス名**についても有効なので、いろいろと試してみてほしい。「相対パスとは何か」については、別の回で説明したいと思う。

ワイルドカードを使った補完

ファイル名の自動補完を行うときは、lsコマンドなどでファイルを表示するときと同じように**ワイルドカード**文字を使うことができる。もともと、ワイルドカードの機能はlsなどのコマンドではなく、**シェル自体が備えている**機能なので、あたりまえなのだが、使用できる文字の種類や機能は、通常のコマンドラインでワイルドカードを使う場合とまったく同じである。

たとえば、カレントディレクトリにある「approved.txt」というファイルをlessコマンドで表示させたい場合、前出の例のように「appr……」まで入力してTabキーを押す代わりに、

```
$ less *.txt
```

と入力してから、Tabキーを押してもよい。すると、ワイルドカード文字の展開が行われて、

```
$ less approved.txt
```

というコマンドラインが表示される。もし、カレントディレクトリに「.txt」という拡張子を持つファイルが複数ある場合は、ビープ音が鳴って、補完できなかったことを知らせてくれる。ただしこの場合は、Tabキーを2回押しても候補の一覧は表示されない。改めて、コマンドラインに、

```
$ less app*.txt
```

のように入力しなおしてからTabキーを押せば、目的の補完結果が得られる。もちろん、類似の名前のファイルがほかにもある場合は、適宜入力する文字を補ってからTabキーを押すことになる。

コマンド名の補完

ファイル名の入力ほどは頻繁に使わないかもしれないが、bashではコマンド名についても自動補完が利用できる。とくに、コマンド名が**うろ覚え**できちんと思い出せないようなときに便利である。

冒頭の例で説明したように、コマンド名の補完の使い方はファイル名の補完と同じように、頭の数文字を入力してからTabキーを押すだけである。たとえば、

```
$ cho
```

と入力してからTabキーを押せば、

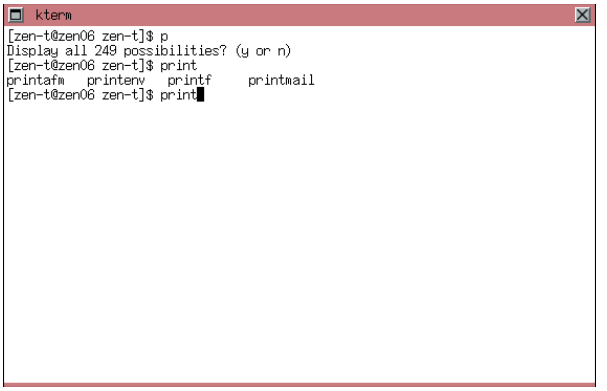
```
$ chown
```

というように、ファイルの所有権を変更するchownコマンドが入力できる。同じ文字列で始まるコマンドが複数ある場合には、ファイル名の補完の場合と同様に、候補の一覧が表示される（**画面3**）。

「pで始まるコマンド」のように、候補の数がとてつもなく多くなるような場合には、「Display all xxx possibilities?」というように、すべて表示するかどうかを確認するメッセージが表示されるので親切である。ただし、ファイル名の補完の場合と違い、ワイルドカードを使った指定はできない。

自動補完機能にはまだまだ便利な使い方があるのだが、ここで誌面が尽きてしまったようなので、次回はファイル名補完のさらに便利な使い方と、過去に入力したコマンドラインの「再利用」をする「ヒストリ機能」について触れてみたいと思う。

画面3　コマンド名の補完
コマンドを入力するときにも、Tabキーを使った自動補完ができる。たとえば、「p」と入力してからTabキーを押すと、コマンドパス中に存在する「p」で始まる名前のコマンドを探すが、数が多すぎる場合には見つかったコマンドの数を表示し、表示してもよいかどうかを問い合わせてくる。このような場合は、「n」を入力してから、コマンドの先頭文字をさらに数文字続けて入力し、あらためてTabキーを押せば、候補が表示されるようになる。



InterBase 6.0

今回は、以前のバージョンとの違いと移行方法および、InterBase 6.0のサポートするデータ型について説明します。

第3回　InterBaseのサポートするデータ型について

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

前回はInterBaseのトランザクション機能と他国語対応について解説しました。

安定したリレーショナルデータベースシステムを作成するうえで、トランザクションを理解し、排他処理の仕組みを考えることが最も重要なことであるでしょう。トランザクション処理についてあまり詳しくない方は、ぜひ前回の記事を一読したあとに、今回の後半で解説するデータ型の解説を読まれることをお勧めします。

今回はまず最初に、InterBase 6で追加されたDialectについて解説します。

Dialectについて

InterBase 5.xまでは基本的にANSI SQL-92に対応するように、データ型およびSQL文のサポートが行われていました。InterBase 6.0ではANSI SQL-99（旧SQL-3）に対応するように64ビットIntegerの対応、TIME、TIMESTAMP型の追加が行われており、できる限りANSI SQL-99の仕様に準拠するように拡張されています。また、旧バージョンのデータベースと互換性を保つため、新たにDialectという概念が提供されています。Dialectという言葉は「方言」や「文体」という意味の単語です。ここではバージョン間のデータベースの互換性を保つためのアクセス基準に使われています。Dialectについてまとめてみると次のようになります。

■Dialect 1

InterBaseサーバはInterBase 5.xと互換性を持った形で動作するモードです。このモードでアクセスしたとき、InterBaseサーバは以下のような振る舞いをします。また、以前のバージョンのクライアントからInterBase 6サーバにアクセスすることができます。

- ダブルクォーテーションで囲まれた文字列は定数として扱われ、区切り記号で囲まれた文字列を識別子として認識させることはできません。
- DATE型は時間情報を含み、TIMESTAMP型として動作します。従来のバージョンでも時間情報は含まれていましたので動作は変わりません。クライアントは従来と同じように時間情報を含む形で値が返されます。Dialect 1では、DATE型とTIMESTAMP型は完全に一致します。
- TIME型は利用できません。
- データベース内では精度が9以上のNUMERIC型およびDECIMAL型は、64ビットではなく、通常の倍精度として保存されています。
  Dialect 1モードでアクセスしているクライアントは、NUMERIC型およびDECIMAL型をアクセスしたとき、倍精度のデータとして受け取ります。Dialect 1のクライアントはDialect 3が提供する64ビットの数値データにはアクセスすることができません。

・Dialect 1モードにおいても、InterBaseサーバはすべてのInterBase 6が提供するデータベース機能および、クライアント機能を認識することができますが、InterBase 6のクライアントがDialect 1モードでアクセスしたときは警告を送ります。しかし、クライアントがInterBase 6のベータ版であった場合は、クライアントがその警告を理解することができないので、警告を送らないようになっています。

■Dialect 2

このモードはクライアント側でのみ利用できます。このモードはデータベースをDialect 1からDialect 3に移行するときに発生した警告を修正するときに使用します。Dialect 3で使用する予約語とフィールド名などのメタデータの名前が衝突してしまうときなどの問題に対応するときに、Dialect 2のモードでisqlを使用してメタデータ名を変更したりするために使用します。Dialect 2モードでメタデータを修正後、再度データベースをDialect 3に移行します。

■Dialect 3

InterBaseサーバはANSI SQL-99準拠の動作をします。また、InterBase 6で提供されているすべての機能が使用できます。

・ダブルクォーテーションは区切り記号として認識されます。
・DATE型が日付のみのデータとなります。クライアントがDialect 3モードでアクセスしたときは日付データのみを返します。
・TIME型が時間のみのデータとなります。
・データベースがDialect 3で作られている場合、DECIMAL型およびNUMERIC型の精度が9以上であれば64ビットの数値として格納されます。クライアントがDialect 3モードでアクセスした場合のみ、これらのデータにアクセスすることができます。

このように、以前のバージョンで作られたクライアントをそのまま使用する場合などを考慮して新しいバージョンが作られています。このDialectの機能がなければサーバのバージョンを上げてしまうと、クライアントも同じように移行しなければならなくなりますが、この機能により以前のバージョンのクライアントでもアクセスすることが可能です。

また、少しずつデータベースの移行を行っていきたい場合などについても便利です。あるデータベースはDialect 1で、他のデータベースはDialect 3でなどといったような運用が可能です。

実際にDialectのモードを設定するには次の2通りがあります。

・isqlのコマンドオプションを使用

isqlの起動時に-sql_dialect ＜モード＞と指定します。**画面1**の例は、InterBase 6に付属しているemployee.gdbにDialect 3のモードでアクセスした場合です。employee.gdbはdialect 1モードで作成しているため、警告メッセージが表示されます。

・SET SQL DIALECTコマンドを使用

SET SQL DIALECT ＜モード＞と指定することで動的にDialectを設定することができます（**画面2**）。

```
# ./isql -U SYSDBA -P masterkey -sql_dialect 3
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect /opt/interbase/examples/employee.gdb;
WARNING: This database speaks SQL dialect 1 but Client SQL dialect was set to 3.
Database:  /opt/interbase/examples/employee.gdb, User: SYSDBA
SQL> show tables;
       COUNTRY                         CUSTOMER
       DEPARTMENT                      EMPLOYEE
       EMPLOYEE_PROJECT                JOB
       PHONE_LIST                      PROJECT
       PROJ_DEPT_BUDGET                SALARY_HISTORY
       SALES
SQL> quit;
```

画面1　isqlのコマンドオプションを使用

isqlなどで何の設定もせずにデータベースに接続した場合、そのデータベースが作られたDialectのモードでクライアントは接続されます。データベースが作成された場合は、すでにDialectが設定されていればそのモードで、設定されていない場合はDialect 3でデータベースが作成されます。

データベースの移行

それでは以前のバージョンのデータベースをDialect 3形式のデータベースに移行する方法について説明します。Dialectのモードの移行はgfixというコマンドラインツールを使用して行います（**画面3**）。gfixは本来、データベースが壊れたときなどに使用する修復ユーティリティですが、InterBase 6ではこのgfixにDialectの移行機能が提供されました。gfixは**表1**のようなオプションが用意されています。

もし、Windowsのクライアントから行いたい場合は、IBConsoleを使用することで行うことができます。IBConsoleを起動して、移行したいデータベースを管理しているサーバにログインし、続いてデータベースに接続します。データベースに接続したら、データベースのアイコンを選択し、マウスの右クリックでスピードメニューを表示します（**画面4**）。次に、[Properties]を選択して、データベースのプロパティを表示します。続いて、[General]のページを表示し、[Options]から[Database Dialect]のリストボックスを選択して変更します（**画面5**）。ここでDialectを変更すると、IBConsoleがgfixを呼び出してDialectを指定したモードに変更します。

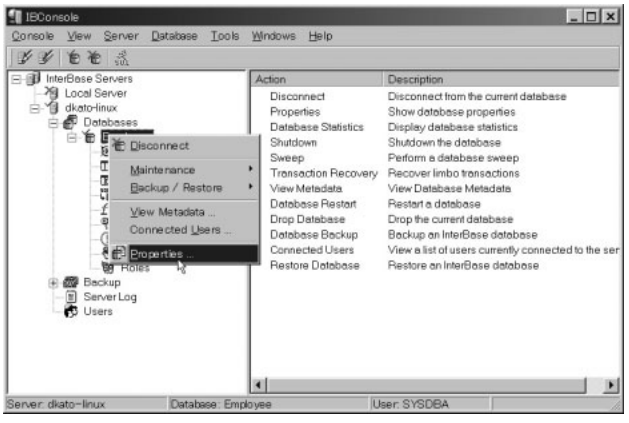
IBConsoleで行った場合、簡単にDialectを変更できて

```
# ./isql -U SYSDBA -P masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> SET SQL DIALECT 3;                                    ← Dialectを3にセットする
SQL> connect /opt/interbase/examples/employee.gdb;

WARNING: This database speaks SQL dialect 1 but Client SQL dialect was set to 3.
Database:  /opt/interbase/examples/employee.gdb, User: SYSDBA
SQL> SET SQL DIALECT 1;                                    ← Dialectを1に変更する
SQL> show database;
Database: /opt/interbase/examples/employee.gdb
        Owner: SYSDBA
PAGE_SIZE 4096
Number of DB pages allocated = 247
Sweep interval = 20000
SQL> quit;
```

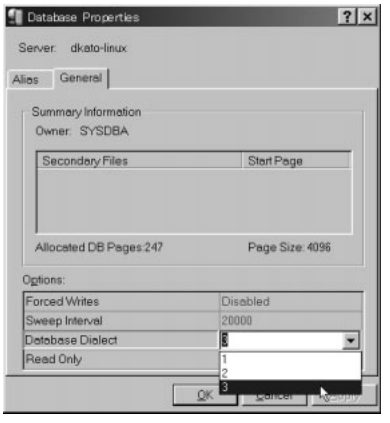
画面2　SET SQL DIALECTコマンドを使用

```
                    モード    データベース名
# ./gfix -sql_dialect 3 /opt/interbase/examples/employee.gdb
```

画面3　gfixによるデータベースの移行



画面4　[Properties]メニューの呼び出し



画面5　IBConsoleでのDialectの変更

しまうので、実施するときは注意して行ってください。

このようにDialectの移行を実施するのは非常に簡単ですが、必ず既存のクライアントアプリケーションとの互換性を検討してから行ってください。また、データベースファイルのコピーを作成してから行うことをお勧めします。

InterBaseのサポートするデータ型

Dialectの説明と移行方法について解説しましたので、本題のInterBaseのサポートするデータ型について説明し

オプション	タスク	説明	互換性
- at[tach]n	シャットダウン	n秒のタイムアウト時間内に新規データベースを接続できないようにするために-shutと一緒に使用されます。n秒後に接続されたプロセスが残っている場合は、シャットダウンはキャンセルされます	5.x互換
- b[uffers] n	キャッシュバッファ	データベースに対してデフォルトのキャッシュバッファをnページに設定します	5.x互換
- ca[che] n		将来の使用のために予約	5.x互換
- c[ommit] {ID ¦ all}	トランザクションリカバリ	IDで指定されたlimboトランザクションをコミットします。または、すべてのlimboトランザクションをコミットします	5.x互換
- f[orce] n	シャットダウン	n秒後にデータベースを強制的にシャットダウンするために、-shutと一緒に使用されます。これは危険な方法なので注意して使用してください	5.x互換
- f[ull]	データ修復	レコードとページ構造をチェックするために、-vと一緒に使用されます。未割り当ての断片は解放します	5.x互換
- h[ousekeeping] n	スイープ	自動スイープ間隔のトランザクション数をnに変更します。nを0にセットすると、スイープを使用不可にします。デフォルトの間隔は20,000です。排他的アクセスの必要はありません	5.x互換
- i[gnore]	データ修復	検査またはスイープしているときにチェックサムエラーを無視します	5.x互換
- l[ist]	トランザクションリカバリ	各limboトランザクションのIDを表示します。-tとともに使用すると、自動2相復元の結果を示します	5.x互換
- m[end]	データ修復	破損レコードを使用不可としてマークします。マークされたレコードは（バックアップ時などには）無視されます	5.x互換
- n[o_update]	データ修復	破損または割り当て漏れの構造体を検査するために-vと一緒に使用します。構造体はレポートされますが修復はされません	5.x互換
- o[nline]	シャットダウン	予定されている-shut操作をキャンセルします。または、現在有効なシャットダウンを無効にします	5.x互換
- pa[ssword] text	リモートアクセス	データベースをアクセスする前にパスワードをチェックします	5.x互換
- p[rompt]	トランザクションリカバリ	トランザクション復元中にアクションをプロンプトするために-lと一緒に使用されます	5.x互換
- r[ollback] {ID ¦ all}	トランザクションリカバリ	IDで指定されたlimboトランザクション、またはすべてのlimboトランザクションをロールバックします	5.x互換
- s[weep]	スイープ	データベースを即時スイープします。自動的なスイープが使用できないときに有用です。排他的アクセスは必要ありません	5.x互換
- sh[ut]	シャットダウン	データベースをシャットダウンします。-attach、-force、-tranのいずれかを指定しなければなりません	5.x互換
- t[wo_phase] {ID ¦ all}	トランザクションリカバリ	IDで指定したlimboトランザクション、またはすべてのlimboトランザクションに対して自動2相復元を実行します	5.x互換
- tr[an] n	シャットダウン	n秒のタイムアウトの間、新規トランザクションを起動できないようにするために、-shutと一緒に使用されます。n秒後にまだアクティブなトランザクションがある場合はシャットダウンはキャンセルされます	5.x互換
- user name	リモートアクセス	リモートデータベースをアクセスする前にユーザー名をチェックします	5.x互換
- v[alidate]	データ修復	割り当てられていても、データ構造に実際に使用されていないページを検索して解放します。破損データベースをレポートします	5.x互換
- w[rite] {sync ¦ async}	データベース書き込み	強制的（同期）書き込みを有効または無効にします。syncは強制的書き込みを使用可能に、asyncはバッファ書き込みを使用可能にします	5.x互換
- z		gfixとInterBaseエンジンのバージョンを表示します	5.x互換
-mo[de] {read_write ¦ read_only}	アクセスモード	データベースを読み書きできるモードにするか、読み取りのみにするかを設定します。デフォルトは読み書き可能です。このオプションを使用するためには、排他的に開く必要があります	6.0～
-s[ql_dialect]n	Dialect設定	データベースのDialectを変更します Dialect 1 = InterBase 5.5互換 Dialect 3 = InterBase 6機能の対応	6.0～

表1　gfixのオプション一覧

ます。InterBaseは他のリレーショナルデータベースとほぼ同じようなデータ型を提供していますが、多少サポートしているデータ型は少ない気がします。たとえば、Microsoft SQL Serverなどで用意されている金額型などは用意されていません。しかし、配列型のサポート、柔軟なBLOB型のサポートなど、他のデータベースでは用意されていない、あるいは上位のバージョンでないとサポートされていないものなどが用意されています。

文字型については、固定長のCHAR型と可変長のVARCHAR型が用意されています。CHAR型とVARCHAR型の使い分けですが、商品コード、会社コードなど常に桁数を一定にしておきたいものや、郵便番号や電話番号など、桁数が決まっているものは固定長であるCHAR型を使用し、住所や氏名など長さが不定のものはVARCHAR型を使います。すべてCHAR型を使用すると無駄なデータ領域を必要としてしまいます。

CHAR型に比べVARCHAR型の方が若干パフォーマンスが遅くなります。

常に検索で使われるようなコードなどはCHAR型を使用するようにしてください。また、253文字以上のVARCHAR型ではインデックスが作成できませんし、ODBCやSQL-Linkで接続する場合は256文字以上のVARCHAR型ではテキストBLOBデータとして扱われますので注意してください。

数値型については、DOUBLE PRERCISION、FLOAT、INTEGER(SMALL INTEGER)、DECIMAL、NUMERICが用意されています。このうち、DOUBLE PRECISIONとFLOATは実数が保存でき、INTEGERは整数が保存できます。DECIMALおよびNUMERICは可変長で実数が保存できるものです。InterBase 5.xまでは64ビット整数が保存できなかったため、非常に大きい数字を扱うときは苦労したのですが、ようやく64ビット整数が対応になったので安心してさまざまなシステムに使用できるようになりました。

日付や時間については、DATE型、TIME型、TIMESTAMP型が用意されています。InterBase 5.xまでは、DATE型に時間まで格納されており、他のリレーショナルデータベースでいうTIMESTAMP型と同じ動きをしていましたが、今回のバージョンから非常にすっきりした形になりました。ただし、以前のデータベースを使用するときはDialectが異なると動きが変わりますので注意する必要があります。また、InterBaseの日付型は米国型の日付形式しか格納できません。

短い形式 MM/DD/YY形式 10/15/2000
長い形式 DD-MM-YYYY形式 15-OCT-2000

長いテキストデータやグラフィックデータなどはBLOB(バイナリラージオブジェクト)として格納することができます。InterBaseのBLOBにはサブタイプという概念があり、サブタイプを指定することでいろいろなタイプのデータを格納することができます。すでに用意されているサブタイプは**表2**の通りです。

一般的に使われるのは0または1です。2以降は管理用に用意されているものです。また、マイナスの値（-1、-2など）はユーザー定義として使用することができます。このユーザー定義のサブタイプを活用することでBLOBフィルタという機能を使用することができます。

BLOBフィルタとはある特定のサブタイプのデータを、他のサブタイプのデータに変換する機能です。たとえば、-1というサブタイプを指定し、BMP形式のデータを格納します。データとして取り出すときは常にJPEG形式で取り出したいとき、-2というサブタイプを指定してデータの

BLOBサブタイプ	説明
0	構造を持たないバイナリデーター般、または型が未決定のデータに適用される
1	テキスト
2	バイナリ言語表現（BLR）
3	アクセス制御リスト
4	（予約）
5	テーブルの現在のメタデータの符号化記述
6	正常に終了しなかったマルチデータベーストランザクションの記述
マイナス値	ユーザー定義

表2 BLOBのサブタイプ

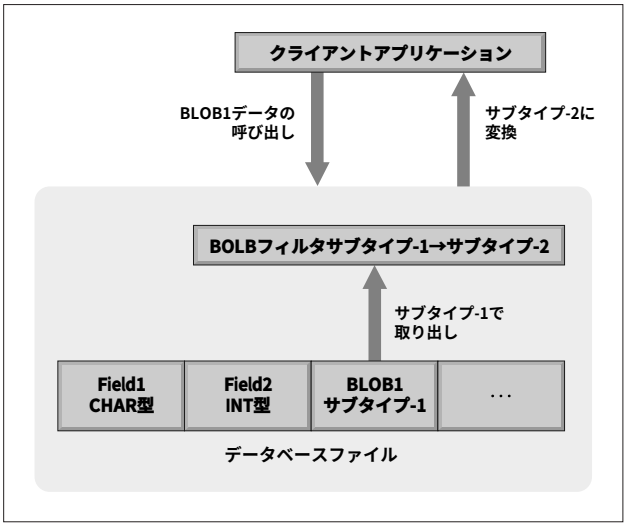


図1 BLOBフィルタ機能

取り出しが可能になります（**図1**）。BLOBフィルタを使用するには、自分で実装する必要があります。この実装方法については高度なプログラミングの知識が必要となりますので、興味がある方はInterBaseのプログラマーガイドを参照してください。BLOBフィルタはデータベースごとに設定します。設定するときはDECLARE FILTERコマンドを使用して設定します。次の例は、SAMPLEというBLOBフィルタをセットする場合です。

```
DECLARE FILTER SAMPLE
  INPUT_TYPE -1 OUTPUT_TYPE -2
  ENTRY_POINT "FilterFunction"
  MODULE_NAME "/tmp/myBlobfilter";
```

InterBaseは配列型についてもサポートしています。配列型というデータ型があるのではなく、BLOB型以外のデータ型であれば配列として定義することができます。たとえば、10桁の固定長の文字を格納するフィールドに4つの要素を持たせたい場合は、

```
CREATE TABLE TABLE1 (CHAR_ARR(10)[4]);
```

と定義することで配列を構成することができます。InterBaseは最大16次元までの配列を使用することができます。たとえば、次の文ではそれぞれ2次元、3次元、4次元の3つのINTEGER配列を定義しています。

```
CREATE TABLE TABLE1
(INT_ARR2 INTEGER[4,5],
 INT_ARR3 INTEGER[4,5,6],
 INT_ARR4 INTEGER[4,5,6,7]);
```

この例ではINT_ARR2は4行、5個の要素の幅で合計で20個、INT_ARR3は120個、INT_ARR4は840個の整数要素を割り当てます。科学技術計算などで使用する場合などでは非常に便利な機能ですが、SQL文では配列の値を取り出すことはできません。InterBaseのAPIの使用、GPREを使用した埋め込みSQL、DelphiおよびC++Builder用のIBObjectsを使用する必要があります。

データ型名	サイズ	範囲／有効桁数	説明	互換性
BLOB	可変長	なし BLOBセグメントのサイズは64Kバイトに限られる 1～32,767バイト	バイナリラージオブジェクトの略。グラフィック、文字、デジタル化した音声などの大量データを格納する。基本的な構造単位セグメント。内容はサブタイプによって記述される	5.x互換
CHAR(n)	n字	キャラクタセットの文字サイズにより、32Kバイト以内に収まる最大文字数が決まる 1～32,765バイト	固定長のCHAR型（テキスト文字列型）。CHARの代わりに、CHARACTERというキーワードも使用できる	5.x互換
VARCHAR(n)	n文字	キャラクタセットの文字サイズにより、32KB以内に収まる最大文字数が決まる	可変長のCHAR型（テキスト文字列型）。VARCHARの代わりに、CHAR VARYINGやCHARACTER VARYINGというキーワードも使用できる	5.x互換
DATE	64ビット	西暦100年1月1日～32768年2月29日	時間の情報も含まれる	5.x互換
DOUBLE PRECISION	64ビット	1.7×10^{-308}～1.7×10^{308}	倍精度:15桁の有効桁数	5.x互換
FLOAT	32ビット	3.4×10^{-38}～3.4×10^{38}	単精度:7桁の有効桁数	5.x互換
INTEGER	32ビット	－2,147,483,648～2,147,483,647	符号付き32ビット整数（ロングワード）	5.x互換
SMALLINT	16ビット	－32,768～32,767	符号付き16ビット整数（ワード）	5.x互換
DECIMAL	可変長	有効桁数=1～18 有効桁数を表す、格納可能な桁数 小数点以下の桁数=1～18 格納可能な小数点以下の桁数 （有効桁数と同じ、またはそれ未満）	有効桁数、小数点以下の桁数の順序で指定する数値。たとえば、DECIMAL(10,3)は次の形式による数値を正しく格納することを意味する ppppppp.sss	5.x互換
NUMERIC	可変長	有効桁数=1～18 有効桁数を表す、格納可能な桁数小数点以下の桁数=1～18 格納可能な小数点以下の桁数 （有効桁数と同じ、またはそれ未満）	有効桁数、小数点以下の桁数という順序で指定する数値。たとえば、NUMERIC(10,3)は次の形式による数値を正しく格納することを意味する ppppppp.sss	5.x互換
INTEGER	64ビット	－2,147,483,648～2,147,483,647 符号付き32ビット整数（ロングワード）	Dialect 3のときだけ64ビット整数まで拡張される	6.0～
TIME	64ビット	0:00 AM-23:59.9999 PM	時間の情報	6.0～
TIMESTAMP	64ビット	1 Jan 100 a.d.～29 Feb 32768 a.d.	時間の情報も含む	6.0～

表3　InterBase 6のサポートするデータ型

Javaプログラミング入門
Javaで3Dゲームを作ろう

今回はいよいよブロックを表示し、ボールが当たったらブロックを消す処理を追加します。また、ブロックを処理するためにクラスの配列についても解説します。クラスがいかに便利かを理解していただけると思います。

第6回　クラスの配列

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote／Kazaguruma

「3Dブロックくずし」も、いよいよ大詰めにさしかかってきました。できてみればそれほど複雑なものではなく、ゲームに目の肥えた方なら、これをああしてほしい、こうしてほしいと、いろいろなご要望もあるかと思いますが、本連載のねらいはこの連載で凝ったゲームを作ることではなく、ゲームを動かすためのからくり（プログラム）がどのように実現されているかを、みなさんにできるだけ理解していただくことにあります。

つまり、単純なものではありますが、ひとつのゲームがどのようにして動いているか、個々の部品のはたらきがどうやって実現されているかを理解することにより、独自の改造をほどこしたり、また新しいゲームでも作ろうかと、この連載で学んだ知識を存分に活かしてもらえることをめざしています。

ブロックの構成

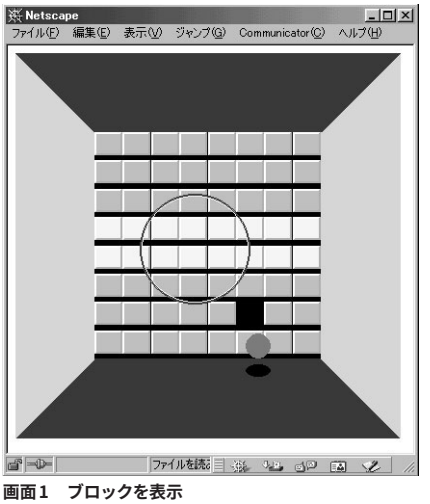
いよいよ、最後の部品であるとともに、「ブロックくずし」になくてはならない、ブロックを作ります（**画面1**）

本ゲームのブロックは、画面奥の壁にタテ8個×ヨコ8個の合計64個を配置します。白黒の画面ではわかりにくいかもしれませんが、少しゲームをおもしろくするために、3段目と4段目の2列のブロックは、ボールが2回当たらないと消えないようにしました。この2列は、最初黄色で表示されていて、1回ボールが当たれば他のブロックと同じようにグレーになり、もう1回当たれば他のブロック同様に消えます。

今まで、壁、ボール、ラケットとそれぞれクラスを作りながらここまできましたが、今回のブロックだけは、ちょっと扱いが違ってきます。

ボールもラケットもゲーム内には1つしか存在しませんから、オブジェクトも1つだけ生成すればよかったのですが、ブロックはグレーが48個と黄色が16個の合計64個必要になります（**図1**）。グレーのブロックと黄色のブロックとの違いは、ボールが何回当たれば消えるかということと、表示するときの表面の色だけですから、それはちょっとした属性の違いとして、1種類のクラスで対応することにします。ですから、64個のブロックオブジェクトを生成して、そのうちの16個には、黄色の表面と、ボールが1回当たっても消えないという属性に変えてやればよいのです。

生成するオブジェクトは64個ですが、プログラムも同じように64回書かなければならないということはありません。これまで何度も出てきた「クラス」とは、いわばオブジェクトの設計図にあたるものですから、あくまでもクラス



画面1　ブロックを表示

は1つあればよいということになります。そして、そのクラスから64個オブジェクトを生成すればよいのです。どうです、クラスって便利でしょう。

さて、1個のブロックのためのクラスをもとに、オブジェクトを64個生成すればよいことがわかりましたが、そのブロック全体を管理するのは誰なのでしょうか。個々のブロックは、自分自身の表示やボールの当たった回数などは管理できますが、ブロック全体にわたる情報、たとえば残りブロック数などは、どのクラスに聞けばわかるのでしょうか。

これまで何度もお話ししているように、クラスはできるだけ部品化して、自分の機能だけに専念すればよいように作るべきですから、個々のブロッククラスは、他のブロックのことまでまかなうべきではなく、あくまでも1個のブロックに関する属性や機能にとどめておくべきです。そこで、64個のブロック全体に関わる問題を引き受けるためのクラスを1つ用意します。

どちらもブロックと呼んだのではまぎらわしいので、1つずつのブロックをPieceクラス、ブロック全体をBlockクラスと名付けることにします。

アプレットが起動してそれぞれがオブジェクト化されると、アプレット内にはBlockオブジェクトが1つ、Pieceオブジェクトが64個存在すればよいことになります。そして、Blockオブジェクトは Pieceオブジェクトのいわばまとめ役ですから、64個のPieceオブジェクトはBlockオブジェクト内で生成します（**図2**）。

こうすることによって、アプレットやBallオブジェクト、Racketオブジェクトからは Pieceオブジェクト（個々のブロック）にアクセスすることができませんが、この場合はそれがかえって都合のよいことになります。というのも、Pieceオブジェクトのは働きは、

- **表示すること**
- **ボールが当たった回数をおぼえておくこと**
- **消えること**

となりますが、それは外部のオブジェクト（BallやRacket）からすれば、ブロック全体に対して（Blockオブジェクト）、

- **表示しなさい（残っているブロックのみ）**
- **ボールが当たったから適切に処理しなさい**

と指令し、あとはブロック内部でうまく処理してもらうほうがよいからです。プログラムもすっきりします。

BlockクラスとPieceクラスを用意したことで、このあたりをうまく処理することができます。

アプレットやBallクラスからは、ブロック全体、すなわちBlockクラスにしかアクセスしません。用途としては、

- **ブロックの表示**
- **ボールの当たり判定**
- **残りブロック数の取得**

ができれば十分です。

一方、Blockクラスは、外部からのこのような要請を受けて、それにこたえるのですが、個々のブロックの状態や動きに関しては、各Pieceにまかせながら、64個のPieceをとりまとめます。

具体的には、ブロックを表示する場合、まずアプレットがBlockクラスの

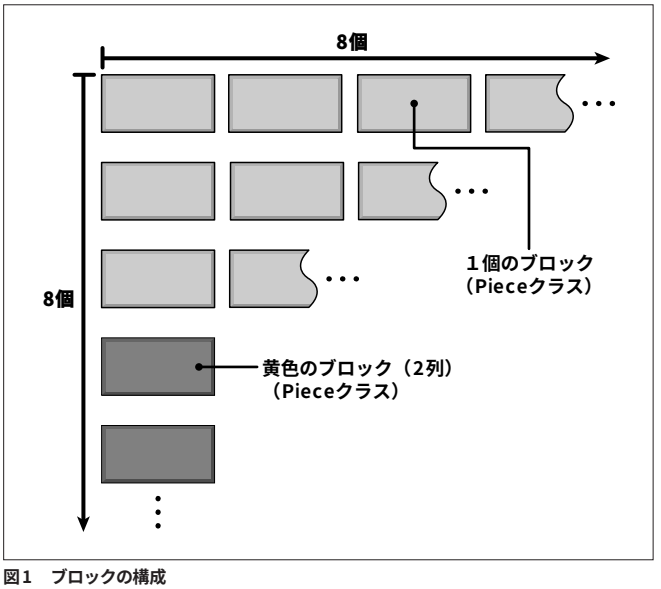


図1　ブロックの構成

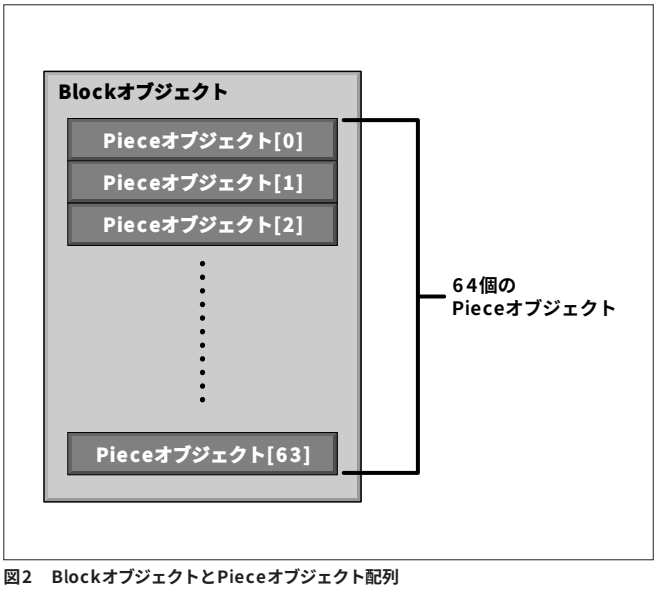


図2　BlockオブジェクトとPieceオブジェクト配列

表示メソッドを呼び出します。Blockクラスのメソッドでは、順に64個のPieceそれぞれの表示メソッドを呼び出します。もちろんこのとき、ボールが当たって消えたブロックや、中の2列のように、黄色で表示したり、1回ボールが当たったらグレーで表示したりする必要があるのですが、Blockクラスではそれぞれの Pieceの表示メソッドを呼び出すだけで、表示の方法には一切関与しません。

1つ1つのPieceクラスの表示メソッドでは、それぞれボールが当たった回数をおぼえていて、その回数に応じて、表示しない／グレーで表示／黄色で表示などを適切に処理します。

ここがクラスとオブジェクトの醍醐味で、プログラミングとしては、1種類のクラスを作るだけ、すなわち1個のブロックの表示処理を書いておくだけです。しかし、実際に起動しているときは、そのクラスが64個のオブジェクトとして増殖されますので、64個のオブジェクトが独立した表示機能を持つことになり、それぞれが独自に消えたり、適切な色で表示してくれるのです。

リスト1がPieceクラスで、**リスト2**がBlockクラスです。それぞれのdisp()メソッドを見てください。Pieceクラスのdisp()メソッドでは場合に応じて、

- **表示しない**
- **黄色で表示**
- **グレーで表示**

という処理を行っています。一方Blockクラスのdisp()メソッドでは、単にPieceクラスのdisp()メソッドを64個分呼び出しているだけです。決して、ブロック全体を表示するときに、全体を管理するBlockクラス自らが、このブロックは表示しないで、あのブロックは黄色にして……などとは行っていないのです。

クラスという考え方が導入される以前のプログラミングでは、まさに64個分を、1つ1つ管理するプログラムを書かなければなりませんでした。しかし、クラスとオブジェクトという考え方、すなわちオブジェクト指向のプログラミングによって、個々の機能は個々のクラスにまかせて、その機能（メソッド）ごとを複数に増殖することにより、あたかも複数のプログラムが動いてい

▼リスト1　Pieceクラス

```
//---------------------------
// Block piece Class
//---------------------------
class Piece extends Disp3D {
        int             id;
        int             gx,gy;
        int             glx,gly;
        int             life;

        Piece(int w, int h, int i) {
                super(w,h);
                int             x,y;
                int             lx,ly;

                id = i;

                x = (id % 8) * (1000 / 8) - 500;
                y = (id / 8) * (1000 / 8) - 500;

                lx = 1000 / 8 - 5;
                ly = 1000 / 8 - 25;

                gx = d2x(x,y,500);
                gy = d2y(x,y,500);
                glx = d2r(lx,500);
                gly = d2r(ly,500);

                life = 1;

                if ( (id / 8) == 3 || (id / 8) == 4 )       life = 2;
        }


        public void disp(Graphics g) {              // Piece の表示
                if ( life == 0 )          return;

                g.setColor(Color.white);
                g.draw3DRect(gx  ,gy  ,glx  ,gly,true);
                g.draw3DRect(gx+1,gy+1,glx-2,gly-2,true);

                if ( life == 2 )          g.setColor(Color.yellow);
                if ( life == 1 )          g.setColor(Color.lightGray);
                g.fillRect(gx+2,gy+2,glx-3,gly-3);
        }


        public int hit() {                          // ボールが当たった
                if ( life == 0 )          return(0);

                life--;
                return(1);
        }

        public int get_life() {                     // life数を返す
                return(life);
        }
}
```

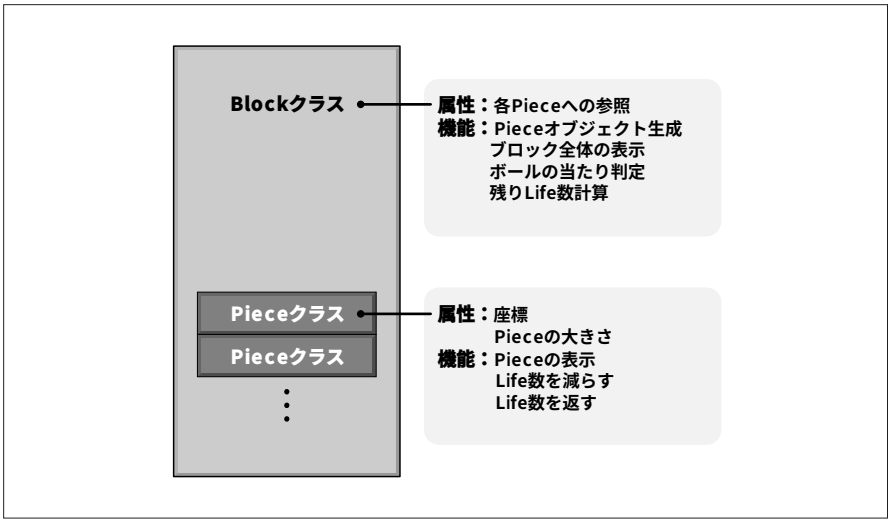


図3　BlockクラスとPieceクラスの役割

▼リスト2　Blockクラス

```
//---------------------------
// Block Class
//---------------------------
class Block extends Disp3D {
        Piece           pi[] = new Piece[64];

        Block(int w, int h) {
                super(w,h);

                for (int i=0; i<64; i++ ) {        // Piece オブジェクト生成
                        pi[i] = new Piece(w,h,i);
                }
        }

        public void disp(Graphics g) {             // Block 全体の表示
                for (int i=0; i<64; i++ ) {
                        pi[i].disp(g);
                }
        }

        public int lifecount() {                   // 残りlife数
                int             cnt = 0;

                for ( int i=0; i<64; i++ ) {
                        if ( pi[i].get_life() > 0 )        cnt++;
                }

                return(cnt);
        }

        public int hitblock(int x, int y) {// 当たり判定
                int             id;
                int             res = 0;

                id = ((y+500) / (1000 / 8)) * 8 + ((x+500) / (1000 / 8));

                res = pi[id].hit();

                return(res);
        }
}
```

るかのようなことが実現できるようになったのです。PieceクラスとBlockクラスのdisp()メソッドのように、数行のプログラムだけで、64個はおろか、1000個でも1万個でも独立して処理できるプログラムが簡単に作れるようになったのです。

図3はBlockクラスとPieceクラスの働き、**表1・表2**はそれを実現するメンバ変数とメソッド、**図4**はBlockクラスのメソッドとPieceクラスのメソッドの連携のようすです。これまでの説明を読みながら、BlockクラスとPieceクラスの役割分担をじっくりと考えてみてください。

オブジェクトの配列

64個のオブジェクトと、ひと口にいってきましたが、BallやRacketのオブジェクトでは1つのオブジェクトしか生成していませんでした。ブロック全体を意味するBlockオブジェクトはBallやRacketと同じく、1つのオブジェクトをアプレットが生成すればよいのですが、64個のPieceオブジェクトだけは生成の方法が異なります。

Pieceオブジェクトは、Blockオブジェクトからアクセスできればよく、アプレットやBall、Racketからはアクセスできる必要はありません。Blockオブジェクトが生成されると、ただちにPieceオブジェクトが生成されて、Blockオブジェクト内部にのみ、その参照があればよいのです。

Blockオブジェクトが生成されるときに自動的に実行されるメソッドは、コンストラクタと呼ばれるBlock()メソッドなので、このコンストラクタの中で、64個のPieceオブジェクトを生成します。そのとき、Pieceオブジェクトに対する参照も64個必要になりますの

メンバ変数	pi[ ]	各Pieceへの参照
メソッド	Block()	Pieceオブジェクト生成
	disp()	ブロック全体の表示
	hitblock()	ボールの当たり判定
	lifecount()	残りLife数計算

表1　Blockクラスのメンバ変数／メソッド一覧

メンバ変数	gx、gy	座標
	glx、gly	Pieceの大きさ
メソッド	disp()	Pieceの大きさ表示
	hit()	Life数を減らす
	get_life()	Life数を返す

表2　Pieceクラスのメンバ変数／メソッド一覧

で、そのために、pi[ ]という配列（変数）を用意します。

```
Piece  pi[] = new Piece[64];
```

これが、Pieceオブジェクトを64個生成したときに、それらの参照を格納する変数を同じだけ、つまり64個用意するための式です。

変数が1個のデータや参照を記憶するものであるのに対して、配列は同種のデータや参照を複数記憶しておける、変数のひとならびのようなものです。変数を1つの引き出しと考えれば、配列はタンスといえるでしょう。

配列の1つ1つの値を使うときは、pi[0]とかpi[1]……などのように、後ろに[ ]で囲んだ番号を付けます。

そして、

```
for (int i=0; i<64; i++ ) {
    pi[i] = new Piece(w,h,i);
}
```

で、実際にオブジェクトを64個生成しており、それぞれのオブジェクトの参照が、順に、

```
pi[0]、pi[1]、pi[2]……
```

に格納されます。以降、Blockオブジェクトの中では、pi[ ]を各Pieceオブジェクトへの参照と考えることができるのです。

リスト2では、すべて、piに[ ]をつけて使っています。[ ]の中には数値か、iなどのように数値を表す変数が書かれています。このように配列は、必ず[ ] を伴って使わなければなりません。[ ]のついていないpiだけを使うことはできません。

ブロックの寿命　Life

本アプレットがゲームとして成り立つためには、ブロックの消え方にルールがなくてはなりません。あたりまえのことですが、原則としてブロックにボールが当たればブロックは消えます。中央の２列のブロックはもう少ししぶとくて、1回当たると通常のブロックと同じになり、もう1回当たると消えます（2回当たると消える）。

これは、ゲームのユーザーから見た現象で、プログラミングのためには少し視点を変えて考える必要があります。

ユーザーからの視点どおりに考えると、各Pieceは自分がボールに当たった回数をおぼえておいて、それが1になれば表示しないで、中央の2列は、それが1になればグレーで表示して、2になれば表示しないと考えればいいように思えます。

今回のルールではそれでもあまり支障ありませんが、もしこのゲームに2面目、3面目といったステージを追加するとどうでしょうか。別のステージではボールが3回当たらなければ消えないブロックを追加したいとか、黄色のブロックを中央の2列ではなく、花の形のように配置したいとなったとき、プログラムそのものに、その都度手を入れなければならなくなります。

この手のゲームでは、ステージの追加がユーザーレベルで、プログラム自体には手を加えずに行えるということがよくあります。もちろん、ステージ

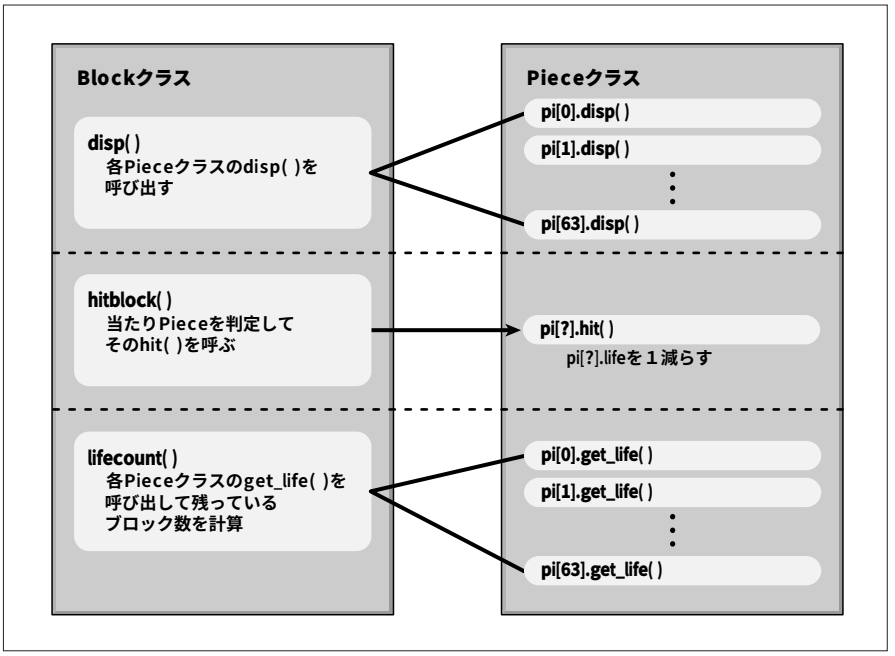


図4　各メソッドの働き

▼リスト3　Ballクラスの変更点

```
//---------------------------
// Ball Class
//---------------------------
class Ball extends Disp3D {
        int             vx,vy,vz;
        int     x,y,z;                  // 3D座標
        int             gx,gy,gr;       // 2D座標、直径
        int             sgx,sgy,sgr;    // 影2D座標
        Racket  rc;                     // ラケットへの参照
        Block   bl;                     // ブロックへの参照

        Ball(int w, int h, Racket rc0, Block bl0) {
                super(w,h);
                rc = rc0;
                bl = bl0;
                init();
        }

        void init() {
                ....
                (省略)
                ....
        }

        void process() {
                int res = 0;

                x += vx;
                y += vy;
                z += vz;

                int lim = 500 - gr / 2;

                if ( x > lim )  vx = -vx;
                if ( x < -lim ) vx = -vx;
                if ( y > lim )  vy = -vy;
                if ( y < -lim ) vy = -vy;

                if ( z > lim )  res = hit(); // ブロック面
                if ( z < 150 )  res = attack();

                if ( res == 1 ) x = y = z = -9999;

                gx = d2x(x,y,z);
                gy = d2y(x,y,z);
                gr = d2r(100,z);

                sgx = gx;
                sgy = d2y(x,500,z);
                sgr = gr / 2;
        }

        int hit() {
                int             res = 0;

                vz = -vz;

                res = bl.hitblock(x,y);

                return(10*res);
        }

        int attack() {
                ....
                (省略)
                ....
        }

        public void disp(Graphics g) {
                ....
                (省略)
                ....
        }
}
```

の追加のためには、プログラム内にある程度の用意は必要ですが、それでも、ステージやルールを変えるたびにプログラムを修正するのではなく、たとえばデータファイルを与えるだけで、新しいステージが追加できるようにしたいものです。

本連載では、ステージの追加機能までは扱いませんが、皆さんが今後そのような機能追加を行いたいときに少しでも役立つ方向で考えていきたいと思います。

また、オブジェクト指向の考え方からしても、いろいろな場面の処理に対応する場合、場面場面の処理をいちいちプログラミングするのではなく、できるだけプログラム（処理）は1つにまとめて、その都度与えるデータによって、各場面に対応できるように努めるべきなのです。いってみれば、動きをデータに変えるのです。そうすればプログラムは簡単になりますし、後に追加される場面にも、うまくいけばプログラムの修正をすることなく対応したり、ユーザーレベルでのカスタマイズが可能になったりします。

では、本ゲームの場合はどのように視点を変えればよいのでしょうか。

ここでは、ボールが当たったことと、ブロックを表示することを切り離します。そして、両者を関係づけるために個々のブロックに「Life（命）」という考え方を持たせます。つまり、ボールが当たるということは、Lifeが1つ減るということにして、そのブロックの持つLifeの数によって、表示するときの色と表示する／しないを判断します。

Life数が0の場合はもちろんそのブロックは表示しないことになり、1であればグレーのブロックを表示、2のときは黄色のブロックを表示します。ボールが当たればLife数は1ずつ減っていくの

ですから、黄色のブロックはボールが1回当たればグレーに、もう1回当たればうまいぐあいに消えてくれることになります。

なんとなくこれでは、先程のユーザーの視点での考え方と同じように思えるかもしれませんが、大きな違いは、こちらの考え方ではあらかじめ各PieceにLife数を与えておくことにあります。

つまり、中央の2列には最初にLife数を2と与えておくことにより、ボールが当たるたびに、黄色→グレー→消えるとなり、その他のPieceには1と与えておくことにより、グレー→消えるとなってくれるのです。このとき、Life数が2であろうが1であろうが、表示メソッドはまったく同じでよく、最初に与えたLife数のデータによって、それぞれのPieceの振る舞いが変わってくれます。つまり、メソッドを変更することなく、与えるデータによってステージの形を変えることができるのです。

これが、動きやはたらきをデータに変えるということです。データは修正や追加が楽ですし、別ファイルとして外部から与えることも可能です。ユーザーの視点（動き）をデータで対応するということは、ちょっとしたことに感じられるかもしれませんが、ソフトウェアのできぐあいを左右する重要なポイントです。

ブロックの当たり判定

ボールがブロックに当たったときの当たり判定は、Blockクラスのhitblock()メソッドと、Pieceクラスのhit()メソッドで処理しています。

まずBallクラスは、とにもかくにもブロック面（奥の面）にボールが当たったら、ブロックの状態には関係なく、Blockクラスのhitblock()メソッドを呼び出します（**リスト3**）。

Blockクラスでは、まずボールの位置に該当するPieceオブジェクトの番号を算出します。

本来は、個々のブロックの位置や大きさもPieceオブジェクトにはわかっていることですから、Pieceオブジェクト側で自分がボールに当たったかどうかを判定すればよいのですが、ここではちょっとズルをして、Blockオブジェクト側の計算式だけで、どのPieceオブジェクトにボールが当たったかを計算しています。いちいちメソッドを呼び出すよりも、たった1行ですむ計算式を使ったほうが、処理スピードが速くもなるからです。

どのPieceオブジェクトに当たったのかがわかれば、Blockオブジェクトはそのpieceオブジェクトのhit()メソッドを呼び出します。要するに、その時のPieceオブジェクトの状態がどうであれ、とにかく「ボールが当たったぞ」と通知するわけです。

通知されたPieceオブジェクトは、その時の自分のLife数を確認します。

もし、すでにLife数が0、すなわちそ

▼リスト4　アプレットクラスの変更点（ball3d_block）

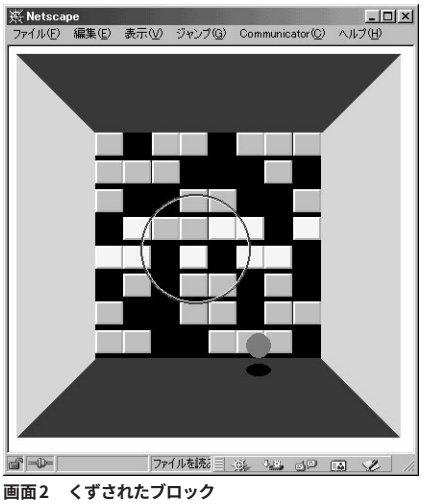
```
//---------------------------
// Applet Class
//---------------------------
public class ball3d_block extends Applet
        implements Runnable,MouseMotionListener  {

        Thread          t;
        Ball            ball;
        Background      bg;
        Racket          rc;
        Block           bl;
        Image           buffer;
        Graphics        bufferg;

        public void init() {
                addMouseMotionListener(this);

                buffer = createImage(400,400);

                rc = new Racket(400,400);
                bl = new Block(400,400);
                ball = new Ball(400,400,rc,bl);
                bg = new Background(400,400);

                t = new Thread(this);
                t.start();
        }


        ....
         (省略)
        ....


        public void paint(Graphics g) {
                if(bufferg == null)     bufferg = buffer.getGraphics();

                bg.disp(bufferg);               // 背景描画
                bl.disp(bufferg);               // ブロック描画
                ball.disp(bufferg);             // ボール描画
                rc.disp(bufferg);               // ラケット描画

                g.drawImage(buffer, 0, 0, this);
        }

        ....
         (省略)
        ....
}
```

のPieceが消えているのであれば何もしません。そうでなければLife数を1減らします。

ここでの処理はこれだけです。先に説明したように、ボールが当たったときの処理では、どのように表示するかなどには一切触れません。単にLife数を減らすだけの処理になります。Life数だけを適切に減らしておけば、あとはアプレットの管理するタイミングで、Blockオブジェクトのdisp()メソッドが呼び出され、それぞれのPieceオブジェクトのLife数により、正しい表示が行われるという寸法です。

もう1つのBlockクラスのメソッドは、残りブロック数を計算して返すlifecount()というメソッドです。今は特に必要ありませんが、最終的にブロックを全部消し終わったら「GAME CLEAR」というようなメッセージを出したいので、そのためにあとどれくらいブロックが残っているのかをアプレットから知ることができるように用意します。

lifecount()の機能は、各Pieceからそれぞれの現在のLife数を取得して、それを合計しているだけです。そして、各Pieceが自分のLife数を返すメソッドがget_life()というメソッドになります。



画面2　くずされたブロック

さて、これでブロックが用意できましたので、アプレットクラスとBallクラスを若干修正して、Blockを使えるようにします（**リスト4**）。

アプレットクラスではBlockオブジェクトを生成して、適当なタイミングで描画をするだけです。

```
bl = new Block(400,400);
```

このようにして、Blockオブジェクトを生成してその参照をblに格納しています。くり返しになりますが、アプレットクラスはPieceオブジェクトを生成しません。Blockクラスが、下請けとしてPieceオブジェクトを生成してくれるからです。

また、BallオブジェクトからもBlockオブジェクトにアクセスする必要があるため、Racketオブジェクトのときと同じように、アプレットからBlockオブジェクトへの参照を渡してもらいます。

```
ball = new Ball(400,400,rc,bl);
```

最後のパラメータがBlockオブジェクトへの参照で、Ballクラス側ではコンストラクタで、

```
Ball(int w, int h,
    Racket rc0, Block bl0)
```

として、受け取ります。

アプレットクラス内でのブロックの描画は、他の部品の描画と同じタイミングで行えばよいのですが、画面は次々と上書きされるため、奥にあるものから先に描画していかなければなりません。ですから、

- **壁**
- **ブロック**
- **ボール**
- **ラケット**

の順になります。

一方Ballクラスは、ボールがブロック面に当たったときに実行するためのhit()メソッドを用意します。とはいっても、このhit()メソッドが行っている処理は、

- **ブロック面でボールを反射させる**
- **Blockクラスのhitblockを呼び出す**

の2つだけです。

やっとブロック「くずし」

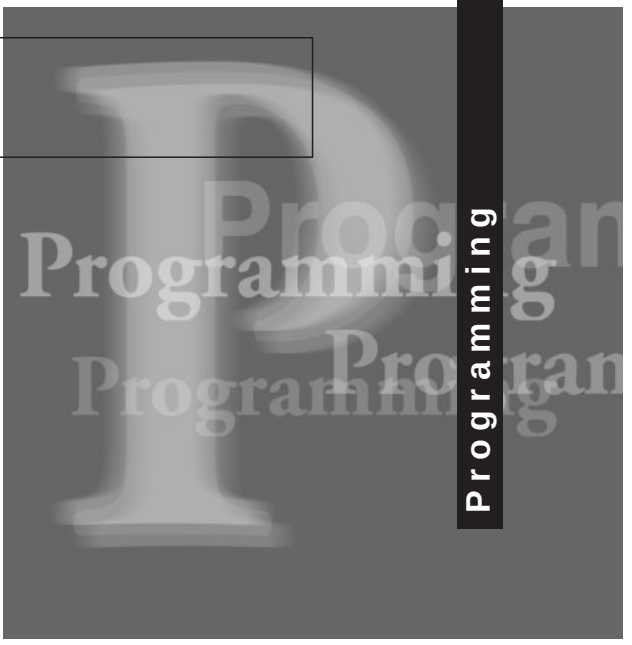
さて、BlockクラスとPieceクラス、Ballクラスとアプレットクラスの修正が済んだら、早速動かしてみましょう。さすがにここまでくるとやっとゲームらしくなってきました。

「GAME CLEAR」のようなメッセージや、手持ちのボール数の設定、ボールが当たった時のサウンドなどはまだ搭載されていませんが、肝心の「ブロックをくずす」ことは十分できますので、一度遊んでみてください。単なる「ブロックくずし」とはいいながらも、実際にやってみるとけっこう難しくて、なかなかクリアできないものです（**画面2**）。

いずれにせよ、これで3Dグラフィックスゲームとしての機能は実現しましたので、あとはゲームを盛り上げるための付加機能になります。

本連載も残すところあと1回となりました。次回は上記のようないくつかの付加機能を搭載し、皆さんオリジナルの「3Dブロックくずし」を完成させましょう。お楽しみに。

プログラミング工房



先月号では、USBデバイスドライバを作成しながら、カーネルプログラミングについて解説した。今月号では、このドライバを改良し、デバイスドライバの開発に必要なカーネルスレッドやセマフォの概念について解説する。

第12回　カーネルのプログラミング（3）

文：藤沢敏喜
Text : Toshiki Fujisawa

先月号で開発したドライバの改良

先月号では、SCSIをUSBに変換するケーブル用のデバイスドライバを作成した。このドライバは、プログラムが複雑にならないように、単純なRead/Writeを行うデバイスドライバとして構成されていた。そのため、アプリケーションからはSCSIとしてアクセスできず、デバイスドライバ専用のアプリケーションが必要になるという欠点があった。

そこで、今月号では、アプリケーションから見て、SCSIホストアダプタとまったく同じように扱うことができるような、より汎用性があるデバイスドライバへの改良を行ってみる。

先月号の記事を読んで、自分には難しすぎるし、関係ない世界だと感じた読者が多いかもしれない。しかし、カーネル開発の雰囲気を知っていると、Linuxを使ううえでもいろいろと応用が利くし、世界中にいるデバイスドライバ開発者の気分も体験できる。

また、今回のドライバでは、スレッド（タスク）とセマフォを使用している。筆者は、「タスクスイッチ」と「セマフォ」、そして「仮想記憶」がオペレーティングシステム分野での3大発明ではないかと感じている。このうち、今回説明する2つの概念は特に興味深いものなので、この機会にぜひ概要を理解していただければと思う。

デバイスドライバの開発手法

さて、先月号でも述べたように、SCSIエミュレーションを行うためには、カーネルスレッドやセマフォといった比較的難しい概念を理解する必要がある。しかし、いきなりこれらの機能を使用するとプログラムが難しくなり、デバイスドライバ入門者には理解しにくくなる。そのため、先月号では簡単な方法を実装して解説することにしたのである。

一方、開発を進めるうえでも、専用品から汎用品へと発展させるほうが、いろいろと都合がよい。なぜなら、デバイスドライバなどのカーネル開発時は、ほんの少しプログラムを間違えただけで、すぐにカーネルがパニックを起こして、リブートしてしまうからである。

したがって、階段を一段ずつ上るように慎重に開発を進めていくのが、回り道のようでも、結局は近道になるのである。

実際、SCSIエミュレーションを行うのは結構難しく、今回の開発でも何回もリブートを繰り返し、やっと原稿の締め切りに間に合わせることができた。もし、先月号のプロトタイプデバイスドライバを書かずに、最初からSCSIエミュレーションを実行するようなコードに挑戦していたら、記事にすることは当分先になっていただろう。

参考にしたソースコード

デバイスドライバを書くには、似たようなデバイスドライバのソースコードを参考にするとよい。linux-2.4.0には、USBマスストレージクラスのデバイスドライバが含まれていて、USBアクセスとSCSIエミュレーションの実装が行われている。

当然のことながら、今回のUSBとSCSIの変換を行うデバイスドライバでは、USBとSCSIへアクセスするためのコードが必要になる。したがって、このUSBマスストレージクラスのデバイスドライバのソースコードはたいへん参考になる。

このような参照は、GPLにより自由に行えるため、デバイスドライバのソースコードが公開されてないOSと比べ、開発はきわめて簡単である。

また、ソースが公開されることが前提になっているためか、開発者は綺麗なコードを書かかなければならないという意識を持つようである。そのため、ソースは非常に読みやすく、我こそはと思う全世界の腕自慢がソースコードをブラッシュアップするので、たいへん高品質である。

Linuxが急速に発展したのも、読みやすく、技術レベルが高いカーネルソースを、自由に参照して再利用できるという面が大きいと思われる。

今回の改良のポイント

さて、前回のドライバでは、アプリケーションからopen、read、write、closeの各システムコールが呼ばれたときに、その機能を実行するという単純な構造を採用していた。

一方、今月号で作成するデバイスドライバでは、アプリケーションからはSCSIとしてアクセスされるため、SCSIを処理するレイヤから間接的にread、writeが行われる。

このため、まずSCSIを処理するレイヤをSCSIホストアダプタとして登録しなければならない。

この登録時には、SCSI処理レイヤからのコマンドを別のタスクとして処理するために、カーネルスレッドを起動し、そのスレッド（タスク）で処理を行うことにする。

ここで生成するカーネルスレッドでは、セマフォを使うことにより、送られてくるSCSIコマンドの排他制御を行い、送られたSCSIコマンドを実行することになる。

文章にすると何やら難しそうな感じがするが、プログラムの行数としては、全部で800行ちょっとしかない小さなプログラムである（付録CD-ROMに収録）。

したがって、タスクやセマフォの基本的な概念がわかっていれば、プログラムを理解することはそんなに難しくはない。

▼リスト1　デバイスをSCSIホストアダプタとして登録するための構造体定義

```
static Scsi_Host_Template
xscsi_scsi_host_template = {
        name:           "usb-xscsi",
        detect:         xscsi_scsi_detect,
        release:        xscsi_scsi_release,
        command:        0,
        queuecommand:   xscsi_scsi_queuecommand,

        eh_abort_handler:       xscsi_scsi_abort,
        eh_device_reset_handler:0,
        eh_bus_reset_handler:   0,
        eh_host_reset_handler:  xscsi_scsi_host_reset,

        can_queue:              1,
        this_id:                -1,
        cmd_per_lun:            1,
        present:                0,
        unchecked_isa_dma:      FALSE,
        use_clustering:         FALSE,
        use_new_eh_code:        TRUE,
        emulated:               TRUE
};
```

SCSIホストアダプタの名前
検出時に呼ばれる関数
終了時に呼ばれる関数
SCSIコマンドをキューに入れる関数
異常処理関数（未実装）
SCSIリセット関数（未実装）

SCSIホストアダプタとして登録する方法

Linuxでは多数のSCSIホストアダプタがサポートされていて、それぞれのチップに応じた処理をしている。しかしながら、SCSIプロトコル自体は汎用の規格であるため、個々のSCSIホストアダプタに依存しない処理もたくさんある。このため、Linuxではチップに依存しない共通部分をまとめて、どのSCSIホストアダプタからも共通に使えるような構造になっている。

したがって、あるデバイスをSCSIホストアダプタとして登録するためには、**リスト1**にあるような構造体を定義する必要がある。

ここでは、ホストアダプタの名称(" usb-xscsi ")や、アダプタが検出されたときに呼ばれる関数（xscsi_detect）、そしてアダプタをリリースするときに呼ぶ関数、さらにアプリケーションからSCSIコマンドが送られたときに、そのコマンドを処理する関数（xscsi_scsi_queuecommand）などが定義されている。なお、この構造体ではタイムアウトしたときの関数や、SCSIバスリセット関数も定義しなければならないのだが、今回これらの実装は行っていない。

さて、上記で定義したSCSI構造体を、SCSIホストアダプタとして登録するには、

```
scsi_register_module(MODULE_SCSI_HA, SCSI構造体)
```

という関数を呼ぶことになる。

以上のような登録作業をすることにより、アプリケーションからSCSIコマンドが発行されると、そのコマンドが正しく実行されるようになる。

カーネルスレッドの生成

さて、コマンドを処理するxscsi_scsi_queuecommand関数であるが、この関数は呼ばれたときに直ちにコマンドを処理するわけではなく、SCSIコマンドをいったんキューと呼ばれるバッファ（待ち行列）に登録し、カーネルはあとからそれを順番に読み出して実行する。

Column

カーネルスレッド

Linuxはマルチタスクなので、ユーザー空間ではさまざまなプロセス（タスク）を同時に実行することができる。カーネルのプログラミングでも、いろいろなタスクを同時に実行すると、都合がよいことが多いのだ。この目的、つまりカーネル空間内で複数のタスクを実行するために用いられるのが、カーネルスレッドと呼ばれるものである。

スレッドは、プロセスよりも処理が軽いという特徴がある。しかし、どちらも複数のプログラムを同時に実行することを示す言葉であり、その概念はほぼ同じであるため、「タスク」という言葉を総称として使うことも多い。

なお、割り込みを巧妙に利用することにより、複数のタスクを使わずにプログラムを記述することは不可能ではない。しかしながら、割り込みを使ったプログラミングでは、わけのわからないフラグや変数を多用しなければならないことが多く、プログラムの見通しが非常に悪くなる。

このような場合にカーネルスレッドを利用すると、きれいでわかりやすいプログラムを作成できることが多い。

たとえば、カーネルスレッドの典型的な利用例は、右のようなコードである。

ここで、タスクBでは、タスクAから送られたコマンドを受信し、受信したコマンドによってそれぞれの処理を行っている。

このようなテクニックは、時間がかかる処理を行う場合によく用いられる。つまり、この場合、送信側のタスクAではコマンドを送り終えてしまえば、すぐに次の仕事を行うことができる。そして、受信側のタスクBでも、受け取ったコマンドをゆっくりと処理すればよいというわけである。

なお、今回作成したデバイスドライバでは、このカーネルスレッドを、SCSIコマンドの処理をするために用いている。カーネルスレッドを利用することにより、すっきりとしたコードで記述することができるからである。

デバイスドライバは、複雑な動作を実現しなければならないプログラムであるとともに、高い信頼性も要求される。したがって、できるだけシンプルなソースコードにしておくことが非常に重要である。

```
■タスクA
タスクBを生成
for(;;){
        イベントの発生を待つ
        スレッドBへコマンドを送信
        .....
}

■タスクB
for(;;){
        コマンド受信待ち
        switch(受信したコマンド){
        case コマンドA:
                コマンドAの処理;
                break;
        case コマンドB:
                コマンドBの処理;
                break;
        }
}
```

今回のデバイスドライバでは、このキューの処理をするために、カーネルスレッドとセマフォを利用している。

また、キューに積まれたSCSIコマンドを順番に処理するためにスレッドを起こしているが、このスレッドの起動部分は**リスト2**に示すようなコードになる。このリストは、先月号で解説したように、USBケーブルが接続されたときに呼ばれる関数の一部である。この部分は、接続されたデバイスが「USB-SCSI変換ケーブル」だと認識され、エンドポイントなどの情報を取得したあとに実行される。

ここでは、最初にこのデバイスで使用する各種の情報を保存するためのメモリエリア（desc）を確保し、ベンダーIDやエンドポイント番号をそこに保存している。そして、init_MUTEXというマクロを用いてセマフォを初期化し、SCSI構造体も確保したメモリ内に格納している。

その次は、いよいよカーネルスレッドの生成であるが、新しいスレッド（タスク）は、カーネルに用意されているkernel_thread関数を、

```
kernel_thread(thread_func, desc, CLONE_VM);
```

というように呼ぶことによって生成される。thread_funcは関数へのポインタで、これで示される関数が、現在のス

▼リスト2　キューに積まれたSCSIコマンドを順番に処理するためのスレッド起動

```
static void *
device_probe(struct usb_device *dev, unsigned int ifnum)
{
        /* ... */                          ← 先月解説したので省略
        desc = kmalloc(sizeof(desc_t), GFP_KERNEL);   ← 情報保存領域を確保
        desc->vid_pid = vid_pid;           ← ベンダーIDと製品ID
        desc->dev     = dev;               ← デバイス情報へのポインタ
        desc->ep_inp  = ep_inp;            ← 入力用USBエンドポイント
        desc->ep_out  = ep_out;            ← 出力用USBエンドポイント

        init_MUTEX_LOCKED(&(desc->sem_is_thread_start));   ← 初期値0のセマフォとして初期化

        init_MUTEX(&(desc->sem_dev));      ← 初期値1のセマフォとして初期化

        init_MUTEX(&(desc->sem_queue));    ← 初期値1のセマフォとして初期化

        memcpy(&(desc->host_template), &xscsi_scsi_host_template,
               sizeof(xscsi_scsi_host_template));   ← SCSIホストとして登録するための構造体

        (desc_t *)desc->host_template.proc_dir = desc;   ← SCSI構造体にUSB情報領域を一時保存
        thread_pid = kernel_thread(thread_func, desc, CLONE_VM);   ← 新しいスレッドを起動
        if( thread_pid < 0 ){
                err_msg("Can't start thread\n");
                kfree(desc);
                return NULL;
        }
        down(&(desc->sem_is_thread_start)); /* wait until start */   ← 起動したスレッドが初期化されるまで待つ

        desc->host_template.module = THIS_MODULE;   ← モジュールの登録
        r = scsi_register_module(MODULE_SCSI_HA, &(desc->host_template ));   ← SCSIホストアダプタとして登録
        if( r != 0 ){
                err_msg("scsi_register module fail err=%d", r );
                kfree(desc);
                return NULL;
        }
        return desc;                       ← このデバイスで使用する情報保存領域
}
```

レッドとは別のスレッドとして実行されることになる。2番目のdescは先ほど用意したメモリブロックで、新しいタスクへ各種情報を受け渡す。

カーネルスレッドの処理

kernel_thread関数で生成されるスレッドは、**リスト3**で示されるthread_funcという名称の関数である。この関数では最初に、そのスレッドで必要のないファイルに関する情報を破棄するなど、スレッドとして動作するための初期化を行う。

次に、init_waitqueue_entry、init_waitqueue_head、add_wait_queueを用いて、キューの初期化を行う。

ここまでの初期化が終わると、このタスクを生成した親タスクへ準備が整ったことをしらせるためのセマフォを、

```
up( &(desc->sem_is_thread_start) );
```

としてupし、呼び出したタスクの待ち状態を解除する。ちなみに、このupに対応するのが、**リスト2**で記述されている、

```
down( &(desc->sem_is_thread_start) );
```

である。このsem_is_thread_startというセマフォは、初期値が0となっているので、親スレッド（**リスト2**）でdownを呼び出したときに待ち状態になる。そして、子スレッド（**リスト3**）でupを呼び出したときに、その待ち状態が解除されることになる。

初期化が終わったことを親スレッドへ通知したあと、このスレッドは無限ループになり、キューに入れられたSCSIリクエストコマンドブロックを取り出し、それをprocess_srbという関数で処理するという動作を延々と繰り返すことになる。

上記の取り出し操作は、コマンドを送るスレッドと排他的に動作しなければならないためセマフォを用いている。

Column

セマフォ

セマフォとは、「腕木」を意味する言葉である。つまり、車の通行を許可したり、禁止したりするために踏み切りで使う「遮断機」（downで禁止、upで許可）に由来している。

このセマフォを理解するために、「豆腐の手渡しゲーム」というプログラムを書くことを考えてみよう。

ここで考えるゲームでは、AさんからBさんへ、安全に豆腐を受け渡すことが目的である。また、渡す際には、必ず水が入った容器に入れてから渡さなければならないというルールがあるものとする。

ここで、この容器には同時に10丁までの豆腐しか入れられない。したがって、容器が満杯の場合、Bさんが取り出すまで、Aさんは新しい豆腐を入れることを待たなければならない。

なお、容器にAさんとBさんが同時に手を入れてしまうと、中にある豆腐が壊れてしまうため、AさんとBさんの手が同時に容器に入らないように排他制御する必要がある。

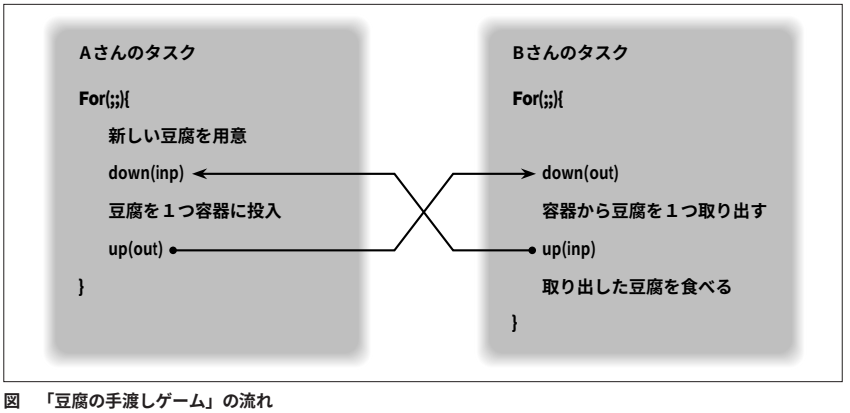


図　「豆腐の手渡しゲーム」の流れ

このようなゲームのプログラムを書く場合には、inpとoutという名前のセマフォを用意し、それぞれの初期値を10に設定したうえで、カーネルに用意されているdown関数とup関数を用いて、図のようなプログラムを記述するとよい。

ここで、down関数（遮断機を下ろして禁止する）は、引数となるセマフォの値を1つ減らし、その値が0の場合には、1になるまでタスクを待ち状態にする関数である。

一方、up関数（遮断機を上げて許可する）は、引数の値を1つ増加させ、値が0だった場合に、その値が1になるのを待っているタスクの待ち状態を解除する関数である。

このupとdownの関数を上記のように用いることにより、排他制御ができるため、AさんとBさんの手が容器に同時に入って豆腐を壊すことがなくなる。

また、容器がいっぱいになるまで、Aさんはどんどん豆腐を入れることができる。そして、Aさんがすぐに新しい豆腐が用意できないときでも、Bさんは容器に豆腐がある限り食べ続けることができ、効率的な処理が行えることになる。

この排他制御を行うためのセマフォがsem_queueである。sem_queueは、**リスト4**でSCSIコマンドをキューに入れるときに、次のように用いられる。

```
down(&(desc->sem_queue));
    desc->queue_srb = srb;
up(&(desc->sem_queue));
```

ここで、queue_srbがSCSIリクエストコマンドブロック（srb）を保存する変数である。このデバイスドライバでは、キューの数が1つだけなので（つまり、コラムの例では豆腐を入れる容器が1つ）、このセマフォの初期値は1に設定してある。

次に、このキュー（豆腐を入れる容器・コラム参照）から取り出しているのがリスト3の、

```
down(&(desc->sem_queue));
    srb = desc->queue_srb;
up(&(desc->sem_queue));
```

という部分である。このようにすることにより、2つのスレッドがこのキューに同時にアクセスしないように制御することが可能になる。

このデバイスドライバの問題点

このドライバは、SCSIカードを装着できないLinuxマシンで、ポジフィルムのスキャンをしたいという筆者の個

▼リスト3　一連の初期化

```
static int
thread_func(void *arg)
{
        desc_t          *desc = arg;
        Scsi_Cmnd       *srb;
        wait_queue_t     wait;

        /* Initialize for thread */
        lock_kernel();
                exit_files(current);
                current->files = init_task.files;
                atomic_inc(&current->files->count);
                daemonize();
        unlock_kernel();

        /*  Initialize for waitqueue */
        init_waitqueue_entry(&wait, current);
        init_waitqueue_head(&(desc->thread_wqh));
        add_wait_queue(&(desc->thread_wqh), &wait);

        up(&(desc->sem_is_thread_start));

        for(;;){
                set_current_state(TASK_INTERRUPTIBLE);
                schedule();
                /*----------------------------------*/
                down(&(desc->sem_queue));
                        srb = desc->queue_srb;
                up(&(desc->sem_queue));
                /*----------------------------------*/
                down(&(desc->sem_dev));
                        process_srb( srb, desc );
                up(&(desc->sem_dev));
                /*----------------------------------*/
        }
}
```

スレッドの初期化

waitqueueの初期化

初期化が終わったことを親スレッドへ通知

強制タスクスケジューリングをする

キューからSCSI命令を取り出す

SCSI Request Blockの内容を実行

人的な動機によって作成されたものである。この目的に対しては、先月段階のドライバでも実用上十分であり、個人的には満足していた。

しかし、プログラマーの心理としては、やはりちゃんとSCSIエミュレーションを行い、アプリケーションからはSCSIとしてアクセスできるようにしたいものである。

そこで、SCSIエミュレーションを行うように改造したわけであるが、現在のバージョンではいくつか問題がある。

まず目に付くのは、USB-STALL時に、

```
usb-uhci.c : interrupt, status 3, frame # XXXX
```

という警告メッセージが画面に出力されることである。このメッセージは、UHCIチップのUSBSTSレジスタを読み出したときに、その値が1以外のときに表示される。これが気になる場合は、usb-uhci.cの中で表示を行っている行を消すとよいだろう。

また、このドライバではSCSIコマンドのうち、今回対象としたスキャナが使用するINQUIRY、READ6、WRITE6、REQUEST_SENSEの4種類のコマンドしかサポートしていない。このため、これ以外の命令を使うSCSI機器を使う場合は、ドライバの改良が必要である。

それから、終了時の処理が適切でないため、リブート時には大量のエラーメッセージが出力されてしまう。

なお、今回テストした環境は、kernel-2.4.0-test5とVAIO-R50（UHCI）であるので、先月号で書いたようにOHCIを使用したPCでは動かないことがある。

おわりに

3回にわたり、USB-SCSI変換ケーブルという、Linuxカーネルではサポートされていないジャンルのデバイスドライバを実際に作成し、カーネルプログラミングの世界をのぞいてみた。

USBは、Linuxにとっては新しく、結構難しい分野のプログラミングである。SCSIは古くからあるものの、これも仕様を理解することはそんなにやさしくはない。

また、デバイスドライバはハードウェアが相手なので、非同期なタイミングでさまざまな事象が起こる。そのため、プログラムを書くためには、割り込み、カーネルスレッド、セマフォなどの比較的高度な概念が必要になる。

さらに、プログラムを試験するたびに、リブートを何十回も繰り返さなくてはならず、デバッガもほとんど役に立たないため、デバッグはきわめて難しい（筆者はこのデバイスドライバを書くために、週末を4回ほど潰した）。

したがって、今回の記事を難しく感じ、自分ではとてもできないと思った読者も多かったかもしれない。

しかしながら、困難であればあるほど、その達成感は大きい。特に、実際に動く「モノ」を動作させることができる「ハードウェア」を征服したときの気分は格別である。

この記事をきっかけとして、カーネルプログラミングに興味をもち、デバイスドライバをバンバン書けるような読者が増えれば、筆者としてはとても嬉しい。

▼リスト4　srbポインタの内容によるSCSIコマンドの実行

```
static int
xscsi_scsi_queuecommand(Scsi_Cmnd *srb, void (*done_func)(Scsi_Cmnd *))
{
        desc_t *desc = (desc_t *)( srb->host->hostdata[0] );
                                                    SCSI構造体に保存してあったデバイス情報を取得
        /* get exclusive access to the structures we want */
        /*-------------------------------------------------------------------*/
        down(&(desc->sem_queue));
            desc->queue_srb = srb;     /* enqueue the command */
                                                    srbをキューへ保存
            srb->scsi_done = done_func;
                                                    キューへ保存したコマンド終了時に呼ばれるコールバック関数
        up(&(desc->sem_queue));        /* release the lock on the structure */
        /*-------------------------------------------------------------------*/
        /* wake up the process task */
        wake_up(&(desc->thread_wqh));
        return 0;
}
```

Column

USB-SCSI変換ケーブル

先月紹介したように、現在秋葉原で入手可能なUSB-SCSI変換ケーブルは、筆者の知る限り次の4種類である。

- Logitec LUB-SC
- Adaptec USB Xchange SCSI
- Microtech USB Xpress SCSI
- Xircom USB to SCSI CONVERTER

この4つは、物理的形状がまったく異なる独自の製品である。Logitec製（写真1）、Adaptec製（写真2）、Microtech製（写真3）の3つは、USBの標準コネクタとケーブル、そしてハーフピッチ50ピンと呼ばれるSCSI-2コネクタが一体となっている。EPSONの少し前のフラットベッドスキャナのSCSIコネクタは、フルピッチのSCSIコネクタとMacintosh用のコネクタとなっているため、このような機器を接続するためには、2000円ほどで売られているハーフピッチ50ピンとフルピッチ50ピンの変換ケーブル（写真4）が必要になる。

一方、Xircom製は、写真5のように非常に多くの付属品が同梱されている。付属のUSBケーブルは、本体と分離できるようになっていて、一般に市販されているUSBケーブルも使用できる。

この製品のSCSIコネクタ側は、Macintoshでよく使われている50ピンDSUB-SCSIコネクタとなっている。このため、Macintosh用SCSIコネクタを持つフラットベッドスキャナに接続する場合には、変換コネクタを必要としないため非常に便利である。なお、この製品は、ほかの製品と比べて約半額であるにもかかわらず、ハーフピッチ50ピンへの変換コネクタも標準で同梱されている。

これらの4種類のケーブルは、主にMOなどのストレージデバイスを接続することしか考えられていないようで、スキャナなどの動作が確認済みの製品は少ない。

そこで、これらをSONY VAIO-R50と、KONICA Qscanの組み合わせで評価してみた。評価したOSは、VAIO-R50にプリインストールされているWindows 98（version 4.10.1998）である。

結論として、すべての製品で問題なくスキャンすることができ、各製品のスキャン速度はまったく同じであった。ちなみに、SCSI接続した場合のスキャン速度とも比較計測してみたが、このスキャナの場合、転送速度自体が律速段階とならないため、まったく変わらない速度でスキャンすることができた。高速転送を要求しないスキャナでは、SCSIホストアダプタを買うより、これらのケーブルを購入したほうが、ホットプラグができたり、コネクタが接続しやすいなど、便利な面が多いようだ。

なお、Xircomの製品は、VAIO-R50にインストールする際に、Windows 98がブルースクリーンになることがあったが、リブートしたあとは正常に動作した。また、この製品は、Windows 98 SEのマシンで評価したところ、スキャンができなかったこともあり、スキャナを使ううえでは多少不安を感じる。しかしながら、価格と物理的なコネクタの使い勝手の面ではたいへん魅力的である。

さて、Linuxを使っているユーザーとして興味深いのは、これらの製品がどのようなUSBプロトコルを使用しているかである。早速調べてはみたものの、MicrotechとXircomの製品はUSBケーブルを流れる信号の品質が悪く、信号の詳細を調べることができなかった。しかし、一部の信号を見た範囲では、同じチップを用いているようで、USBストレージクラスに似たプロトコルを使っているように見受けられる。一方、Adaptecの製品は、信号がきれいであり、ACアダプタによりSCSIバスパワーを供給できるなど、ハードウェア的な出来は4つの製品のなかでもっとも良いように感じられた。

個人的には、Logitec用のデバイスドライバを書いただけで満足しているのだが、もし要望が多ければ、ほかの製品のデバイスドライバも記事で取り上げてみたい。興味がある読者はアンケートはがき等で要望していただければと思う。

写真1　Logitec製USB-SCSI変換アダプタ

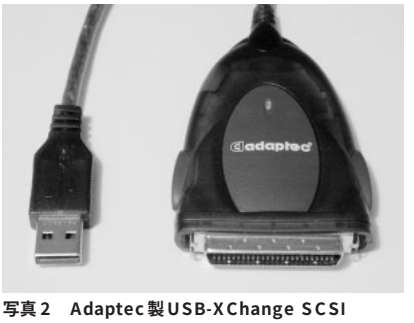


写真2　Adaptec製USB-XChange SCSI

写真3　Microtech製USB-SCSI変換ケーブル

写真4　フルピッチとハーフピッチの変換コネクタ


写真5　Xircom製USB-SCSI変換ケーブル

Rubyで行こう

Rubyプログラミングを楽しんでますか？ 1.6への移行は終わりましたか？ しばらく概念的な話が続いたので、今回は具体的なプログラムを考えてみましょう。お題は、インターネットプログラミングの中から、「電子メールプログラミング」です。

第12回　ポストマン

文：赤松智也
Text : Tomoya Akamatsu

インターネットプログラミング

「インターネットプログラミング」といえば、なんといってもCGIプログラミングでしょう。CGIはテキスト中心の処理であり、開発の手軽さもあってスクリプト言語に向いたジャンルです。世間のほとんどのCGIプログラムはPerlで開発されているでしょうが、Rubyで書けば、もっと楽に、きれいに、効率良く開発できるでしょう。

しかし、RubyによるCGIプログラミングには大きな障害が横たわっています。それは、ほとんどのプロバイダのサーバには、Rubyがインストールされていないことです。Ruby AnywhereというWebページ（http://www.jin.gr.jp/~nahi/Ruby/anywhere.html）には、Rubyをインストールしているプロバイダの一覧があります

が、正直なところまだまだ少数派と言わざるをえません。RubyでCGIしている先進ユーザーは、シェルアカウントの使えるプロバイダで、自分のホームディレクトリにRubyをインストールしているようです。Rubyのインストールは、先月説明したようにそれほど難しいプロセスではありませんが、だれでもできるとまではいきません。これでは、いくらRubyのプログラムが楽でも、CGIプログラムには使いようがありません。

本誌の読者であれば、自前でHTTPサーバを用意することは、技術的には難しくないでしょう。しかし、個人としてCGIプログラムが嬉しいかという問題があります。そこで、RubyによるCGIプログラミングは、おもしろい例題を思いつくまでしばらくお預けということにします。

そこで今回は、Rubyをクライアント側とするインターネットプログラミングにチャレンジしましょう。クライアント側であれば、もうちょっと簡単に、ある程度実用になるプログラムが用意できそうです。

Rubyのインターネットクライアントライブラリは、netディレクトリの下に分類されています。今回使うのは、この中の電子メールの受信と送信に関係するライブラリ（net/pop、net/smtp）です。

RAA（Ruby Application Archive）には、net/imapというライブラリも登録されていて、近いうちに標準添付される予定だそうですが、今回はこれは扱いません。

POP3

POP3（Post Office Protocol）は、ネットワーク経由でサーバからメールを取得するためのプロトコルです。POP3の「3」はバージョンを意味します。POPはメールの取得しか行いませんので、メールの送信にはSMTP（Simple Mail Transfer Protocol）のような別のプロトコルを用いることになります。SMTPを使ったプログラムは今月は紹介しませんが、Rubyからは標準添付のnet/smtpライブラリによって利用可能です。

POP3でのアクセスには、まずnet/popライブラリをロードする必要があります。

```
require 'net/pop'
```

それからサーバに接続します。

```
pop = Net::POP3::new("pop.mydomain.jp")
```

そして、セッションを開始します。

```
pop.start("akamatz", "passwd") do
  ...
end
```

pop.startのブロックの範囲内で、メールに対する操作を行います。 POP3接続に対応するNet::POP3クラスと、個々のメールに対応するNet::POPMailクラスの仕様の概要をそれぞれ**表1**と**表2**に示します。

まず、POPを使ったプログラムの小手調べとして、POPサーバのメール総数を調べてみましょう（**リスト1**）。ここでは、アカウントはプログラムに埋め込み、パスワードはユーザーの入力を求めることにします。

入力時にパスワードが丸見えでは困りますから、sttyを使って一時的にエコー（入力文字列が画面に出力されること）を禁止します（passreadメソッド）。禁止したエコーがそのままでは結果が表示されませんから、確実に元に戻るようにensureを指定します。このbeginなしのensureは1.6の機能ですから、1.4をお使いの方はpassreadメソッドの本体をbeginで囲んでください。passreadメソッドのようなエコーなし入力は、覚えておくと便利です。

「Net::POP3::new」の行からがPOP3処理の本体です。「Net::POP3::new」でサーバ（と必要ならポート番号）を指定し、POP3接続のためのオブジェクトを生成します。

Net::POPクラスのオブジェクトは、mailsメソッドで個々のメールに対応するオブジェクトの配列を返しますから、その配列のサイズはメールの総数になるわけです。popmails.rbの実行結果はこんな感じです（**画面1**）。

「パスワードの入力が面倒だ」という方は、ファイルに記録してもよいでしょう。たとえば、ホームディレクトリの.popmailというファイルに書き込んでおくなどの方法です。その場合、パスワードファイルの中身が他人に見られないように、パーミッションに留意してください。

ただ単にメール総数を得るだけではおもしろくないので、どんなメールがあるのかという情報を表示させてみましょう。メールの日付、発信者、サブジェクトくらいが表

クラスメソッド	
Net::POP3::new(addr="localhost",port=80)	新しいPOP3接続オブジェクトを生成する
インスタンスメソッド	
p.each{|mail|...}	個々のメールについて繰り返す
p.finish	POPセッションを終了する
p.mails	Net::POPMailオブジェクトの配列を返す
p.start(acct, passwd)	POPセッションを開始する。ブロックが与えられた場合にはブロック実行終了後にセッションを終了させる
p.start(acct, passwd){|pop|...}	

表1　Net::POP3クラス

インスタンスメソッド	
m.mail	メールの内容を返す
m.delete	メールを削除する
m.deleted?	メールが削除されているか確認する
m.header	メールヘッダーを取得する
m.size	メールサイズを返す
m.top(lines)	メールヘッダーとlinesで指定した行数のメール本体を返す

表2　Net::POPMailクラス

▼リスト1　popmails.rb

```
#! /usr/bin/env ruby

require 'net/pop'

def passread(user)
  STDOUT.printf "ユーザ %s のパスワード: ", user
  STDOUT.flush
  system "stty -echo"
  STDIN.readline
ensure
  STDOUT.print "\n"
  system "stty echo"
end

server = "pop.mydomain.jp"
user = "akamatz"
passwd = passread(user)

pop = Net::POP3::new(server)
pop.start(user, passwd) do
  printf "%s には全部で %d 通のメールがあります\n",
    server, pop.mails.size
end
```

```
% ruby popmails.rb
ユーザ akamatz のパスワード:
pop.mydomain.jp には全部で 3 通のメールがあります
%
```

画面1　popmails.rb実行結果

示できれば、便利そうです（**リスト2**）。Rubyにはfrom.rbというサンプルプログラムがあります。これは、ローカルスプールのファイルから日付やサブジェクトを表示するものですが、これをPOP対応にすればよいでしょう。

popfrom.rbは、popmails.rbを少々改造したものです。69行という行数はpopmails.rbの25行に比べれば大きくなっているといえますが、新しく増えた部分のほとんどは、実はサンプルプログラムfrom.rbの内容を切り貼りして作ったものですから、開発コストは非常に低くなっています。「プログラムのカット&ペーストは勧められない」というのはよく言われる話ですが、個人的には数行程度の枯れたコードの切り貼りは有効であると思っています。まだバグが残っているようなコードを切り貼りしてしまうと、バグの拡大再生産になってしまいますが。

popfrom.rbの実行結果を**画面2**に示します。popfrom.rbの動作は簡単です。popmails.rbと同様にPOP接続を確立し、eachコマンドで取り出したサーバ側の各メールから、ヘッダ部をheaderメソッドで取り出し、その各フィールドをハッシュに入れます。このフィールドをハッシュに格納する部分は、lib/mailread.rbを参考にしています。

あとは、ハッシュからDate、From、Subjectの各フィールドの値を取り出し、適当に整形して出力するだけです。整形部は、前述のようにsample/from.rbから切り貼りしています。日本語の切り出しを行うkjustメソッドも、取っておくと便利なメソッドでしょう。

POP3の中身

というわけで、net/popライブラリがあれば、POPのプロトコルに関するプログラミングを行う必然性はまったく

▼リスト2　popfrom.rb

```
#! /usr/bin/env ruby

require "parsedate"
require "kconv"
require 'net/pop'

class String
  def kjust(len)
    len += 1
    me = self[0, len].ljust(len)
    if me =~ /.$/ and $&.size == 2
      me[-2..-1] = '  '
      me[-2, 2] = '  '
    end
    me.chop!
  end
end

def passread(user)
  STDOUT.printf "ユーザ %s のパスワード: ", user
  STDOUT.flush
  system "stty -echo"
  STDIN.readline
ensure
  STDOUT.print "\n"
  system "stty echo"
end

server = "pop.mydomain.jp"
user = "akamatz"
passwd = passread(user)

pop = Net::POP3::new(server)
pop.start(user, passwd) do
  if pop.mails.size == 0
    print "You have no mail.\n"
    exit
  end
  pop.each do |mail|
    header = {}
    mail.header.each do |line|
      line.chop!
      break if /^$/=~line  # end of header
      if /^(\S+):\s*(.*)/=~line
        (attr = $1).capitalize!
        header[attr] = $2
      elsif attr
        line.sub!(/^\s*/, '')
        header[attr] += "\n" + line
      end
    end
    y, m, d = ParseDate::parsedate(header['Date']) if
header['Date']
    y ||= 0; m ||= 0; d ||= 0
    from = header['From'] || "sombody@somewhere"
    subj = header["Subject"] || "(nil)"
    from.gsub("\n","")
    subj.gsub("\n","")
    if ENV['LANG'] =~ /sjis/i
      lang = Kconv::SJIS
    else
      lang = Kconv::EUC
    end
    from = from.kconv(lang).kjust(28)
    subj = subj.kconv(lang).kjust(40)
    printf "%02d/%02d/%02d [%s]
%s\n",y%100,m,d,from,subj
  end
end
```

ありません。ありがたいものです。が、一応、基礎知識としてPOP3の中身について紹介しておきます。

POP3は、RFC1939で定義されています。ほかの多くのネットワークプロトコルと同様に、テキストベースのプロトコルになっています。たとえば、popmails.rbで内部的に行われるPOP3のやり取りは、**画面3**のようになります。

こうしてみると、そんなに複雑なプロトコルではありませんね。しかし、毎回このプロトコルを解析するのは少々面倒です。こうした詳細に触れずに、サーバ側の電子メールを簡単に操作できるのも、net/popライブラリのおかげです。作者のあおきさんに感謝しましょう。

spam退治

spamとは「頼みもしないのに送り付けてくる宣伝メール（またはネットニュース記事）」のことです。「ジャンクメール」という呼び名もあるようです。SPAMといえばアメリカのハムの缶詰の名称なのですが（**画面4**）、このSPAMとあのspamは実は直接は関係ないそうです。

宣伝メールのほうのspamの語源は、モンティ・パイソンというイギリスBBCのコメディのギャグに、spamを連呼するものがあるからです。噂では聞いていたのですが、先日NHK BSの深夜放送でとうとう実物にお目にかかりました。感想はというと、正直なところ、この手のギャグはちょっと私の趣味に合わないようです。ブラックジョークが嫌いというわけではないのですが。ちなみに、このモンティ・パイソンは、Rubyのライバル（？）であるPythonの名前の元にもなっています。Pythonが私の趣味に合わないのも、その辺の関係でしょうか？

なお、SPAMについては、

- **TOTALLY SPAM**
  http://www.fingers.com/spam/
- **スパム考古学**
  http://www.kt.rim.or.jp/~ksk/spam/

が詳しく取り扱っています。

日常的に電子メールを使っていると、1日に1通や2通はspamが届くのではないでしょうか？ あまりに多いと大事なメールが埋没したり、必要なメールも一緒に削除しちゃいそうですし、spamメールに「不要なら〜まで連絡してください。リストから削除します」などと書いてあっても、うっかり連絡でもしようものなら「きちんと読んでるアドレスです」と公言するようなもので、より多くのspamを

```
% ruby popfrom.rb
ユーザ akamatz のパスワード:
00/10/10 [Tomoya Akamatsu <akamatz@pdx] 先日はありがとうございました
00/10/10 [matz@zetabits.com (Yukihiro ] Re: 先日はありがとうございました
00/10/10 [schneik@us.ibm.com          ] [ruby-talk:5368] Re: Ruby 1.6.1 config+w
%
```

画面2　popfrom.rbの実行結果

```
    C - クライアント, S - サーバ

S +OK Solid POP3 server ready <30264.971191388@pop>
C USER akamatz
S +OK username accepted
C PASS xxxx
S +OK authentication successful
C LIST
S +OK scan listing follows
S 1 3153
S 2 2924
S 3 3573
S .
C QUIT
S +OK session ended
```

画面3　POP3のやり取り

呼び込みそうです。世の中には「10万メールアドレスを××ドルで。優良アドレスのみよ」などというアコギな商売をしている輩もいるのだそうです。あ〜あ。

こうなったら自衛するしかありません。先ほどのpopfrom.rbをさらに改造したpopspam.rbは、サーバ側に蓄積されたメールからSPAMと思われるメールを抽出し、一気に削除できます（**リスト3**）。本当は面倒な確認など行わず、一気に削除したいのですが、必要なメールを削除してしまってはいけませんし、今回は慎重にコマンドで指定することにしてみました。

［注意］
popspam.rbは、ある条件を満たすサーバ側のメールを削除してしまいます。条件指定を間違って重要なメールを削除してしまっても、筆者、開発者、配布元、および株式会社アスキーは責任を負えません。注意して運用してください。

それではpopspam.rbの実行例を見てみましょう。popspam.rbは対話的なプログラムなので、実行させながら見ていくのがよいでしょう。

```
%  ruby popspam.rb
ユーザ akamatz のパスワード:
    1: [Tomoya Akamatsu <akamatz@pdx] $$$$ this is spam $$$$$
    2: [Tomoya Akamatsu <akamatz@pdx] $$$$ this is spam $$$$$
    3: [Tomoya Akamatsu <akamatz@pdx] $$$$ this is spam $$$$$
```

パスワードを入力すると、SPAM_PATに指定した条件に合うメールを表示します。今回リストに出ているのは、本物のspamではなく、自分で自分に送ったニセspamです。ほしいと思うときには、本物は来ないんですよね。あとはコマンドを打って不要なメールを削除します。まずは、コマンド一覧を表示させてみましょう。

```
spam> h
q               quit
x               delete all spams
l               list spams
<digit>         delete specified spam
<digit>-<digit> delete specified spams
h               this help
```

たとえば、1番のメールを削除するには、

```
spam> 1
mail 1 deleted.
```

とします。リストを取ってみると、

```
spam> l
*   1: [Tomoya Akamatsu <akamatz@pdx] $$$$ this is spam $$$$$
    2: [Tomoya Akamatsu <akamatz@pdx] $$$$ this is spam $$$$$
    3: [Tomoya Akamatsu <akamatz@pdx] $$$$ this is spam $$$$$
```

削除したメールには、「＊」マークが付いています。一度に削除するには、範囲指定かxコマンドを使います。

今回は、Subjectに「Make Money Fast」とか「$$$」などを含むメールをspamと認定しましたが、このあたりのパターンはプログラムそのものを修正してください。この程度の個人的かつ小規模なプログラムでは、設定ファイルを用意するよりも、プログラムそのものを変更するほうがよいように思います。Rubyなら変更の手軽さも、文法の簡潔さも設定ファイルに劣りませんから。

popspam.rb（131行）は、以前のプログラムに比べれば少々大きめですが、やっているのはたいしたことではありません。本質的な部分はpopfrom.rbと大差なく、コマ

画面4　SPAMの缶詰

▼リスト3　popspam.rb

```
#! /usr/bin/env ruby

require "kconv"
require 'net/pop'

SPAM_PAT = [
  ["Subject", /Make Money Fast/],
  ["Subject", /\$\$\$/],
]

def spam?(header)
  for field, pattern in SPAM_PAT
    if header[field] =~ pattern
      return true
    end
  end
  return false
end

class String
  def kjust(len)
    len += 1
    me = self[0, len].ljust(len)
    if me =~ /.$/ and $&.size == 2
      me[-2..-1] = '  '
      me[-2, 2] = '  '
    end
    me.chop!
  end
end

def passread(user)
  STDOUT.printf "ユーザ %s のパスワード: ", user
  STDOUT.flush
  system "stty -echo"
  STDIN.readline
ensure
  STDOUT.print "\n"
  system "stty echo"
end

server = "pop.mydomain.jp"
user = "akamatz"
passwd = passread(user)

pop = Net::POP3::new(server)
pop.start(user, passwd) do
  if pop.mails.size == 0
    print "You have no mail.\n"
    exit
  end
  spams = []
  lines = []
  pop.each do |mail|
    header = {}
    mail.header.each do |line|
      line.chop!
      break if /^$/=~line  # end of header
      if /^(\S+):\s*(.*)/=~line
         (attr = $1).capitalize!
         header[attr] = $2
      elsif attr
         line.sub!(/^\s*/, '')
         header[attr] += "\n" + line
      end
    end
    from = header['From'] || "sombody@somewhere"
    subj = header["Subject"] || "(nil)"
    from.gsub("\n","")
    subj.gsub("\n","")

    unless spam?(header)
      next
    end
    if ENV['LANG'] =~ /sjis/i
      lang = Kconv::SJIS
    else
      lang = Kconv::EUC
    end
    from = from.kconv(lang).kjust(28)
    subj = subj.kconv(lang).kjust(40)
    spams.push mail
    line = sprintf("%4d: [%s]
%s\n",spams.size,from,subj)
    lines.push line
    print " ", line
  end
  while (STDOUT.print "spam> "; STDOUT.flush; line =
STDIN.gets)
    line.strip!
    case line
    when /^q(?:uit)?$/
      exit
    when /^l(?:ist)?$/
      spams.each_with_index do |mail,index|
         print(if mail.deleted? then "*" else " " end)
         print lines[index]
      end
    when /^x$/
      spams.each do |mail|
         mail.delete unless mail.deleted?
      end
      print "all mails deleted.\n"
    when /^([1-9]\d*)$/
      n = Integer($1) - 1
      if spam = spams[n] and not spam.deleted?
         spam.delete
      end
      printf "mail %d deleted.\n", n+1
    when /^([1-9]\d*)-([1-9]\d*)$/
      n = Integer($1) - 1
      m = Integer($2) - 1
      for i in n .. m
         spam = spams[i]
         next unless spams
         next if spam.deleted?
         spam.delete
      end
      printf "mail %d-%d (%d mails) deleted.\n", n+1,
m+1, m-n
    when /^\?$/, /^h(?:elp)?/
      print "q             quit\n"
      print "x             delete all spams\n"
      print "l             list spams\n"
      print "<digit>               delete specified
spam\n"
      print "<digit>-<digit>       delete specified
spams\n"
      print "h             this help\n"
    else
      print "unknown command: ", line, "\n"
    end
  end
end
```

ンドインターフェイスを付けて、POPMailオブジェクトに対する操作を可能にしただけです。増えた部分の大半は、コマンドインターフェイスによるものです。

popspam.rbは簡単なものではありますが、実用的なプログラムへのベースとしてもおもしろいのではないでしょうか？ 考えられる改善案をあげておきます。

- **spamメールのパターンマッチ対象を、複数のフィールドにできるようにする（FromまたはReply-Toが特定のアドレスだったら、など）**
- **spamメールの指定を複数条件のandで指定する（Fromが特定のアドレスで、かつSubjectが特定のパターンなど）**
- **spamメールはただ削除するのでなく、別ファイルに保存する**
- **いったん削除マークを付けたメールを、やっぱり削除しないことにする**

SMTP

POP3はメールを受け取るほうのプロトコルでしたが、メールを送るほうのプロトコルは、SMTP（Simple Mail Transfer Protocol）といいます。POP3と違って、SMTPは簡単で便利な応用例がすぐに思いつきません。ぱっと思いついたのは、数万アドレスにメールを送り付けるプログラムですが……。これは今退治しようとしたspamそのものです。やめておきましょう。

しかし、RubyからSMTPが使えて嬉しくないわけではありません。たとえば、プログラムから電子メールを送って、なんらかの通知を行いたい場合や、SGmailとかcurのようなRubyによる電子メールクライアントを実装する場合にも、このような機能は必須です。もっとも、電子メールクライアントを自作する人は少数派でしょうが。

SMTPを扱うnet/smtpライブラリが提供する、SMTP接続に対応するNet::SMTPクラスの仕様の概要を**表3**に示します。少々、人工的な例ではありますが、例題を示しておきましょう（**リスト4**）。

smtpnote.rbは、指定した時刻に指定したメッセージを自分自身に送ります。備忘録に使えるかもしれません。

```
% ruby smtpnote.rb '11/08 6:06:40' Linux Magazine 2000-12
I will send you a note at Wed Nov 08 06:06:40 JST 2000
```

ただし、smtpnote.rbはデーモンとしてメールを送る時間まで待ち続けますから、途中でリブートされたりすると消えてなくなります。悪しからず。

POP before SMTP

もともとのSMTPには認証機能がありませんでした。ですから、メールサーバはどこからの接続も受け付け、転送していたわけです。しかし、ネットワークユーザーが増えるにしたがい、このことを悪用する人物も現れるようになりました。最近のトピックの多くは、このような不法な連中に対する対処になっているような気がします。

これを禁止するためには、知らないマシンからのSMTP接続を拒否するのが一番簡単ですが、それでは「出張先のネットワークに繋いだマシンからメールを送りたい」というような状況でも、SMTP接続が弾かれてしまいます。

最近のSMTPには認証機能が含まれていますが、残念なことにSMTP認証に対応したメールクライアントはまだあまり見かけません。

仕方なく、過渡的に考えられたのが「POP before SMTP」です。すでに述べたように、POP3にはパスワードによる認証が行われます。ですから、まずPOP3認証が成功したあと、一定時間に限ってSMTP接続を認めるというものです。SendmailやPostfixなどでは、サーバ側の設定によってPOP before SMTPが可能になります。最

クラスメソッド	
Net::SMTP::new(addr="localhost",port=25)	新しいNet::SMTPオブジェクトを生成する
Net::SMTP::start(addr="localhost",port=25,...)	「Net::SMTP::new(addr,port).start(...)」と等価。ブロックが与えられた場合には生成したSMTP
Net::SMTP::start(addr="localhost",port=25,...){|smtp|...}	オブジェクトを与えて実行し、実行終了後クローズする
インスタンスメソッド	
s.finish	SMTPセッションを終了する
s.start([domain,account,password,authtype])	SMTPセッションを開始する
s.send_mail(mailsrc,from,to)	メールを送る。toは文字列または文字列の配列
s.sendmail(mailsrc,from,to)	

表3　Net::SMTPクラス

近は多くのプロバイダで採用されているようです。

POP3接続はほぼすべてのメールクライアントが実装しているでしょうから、これならクライアントを取り換えなくても認証できます。今回紹介したpopmails.rbなどはPOP3認証を行いますから、POP before SMTPに最適です。たとえば、ダイヤルアップでプロバイダに接続したら、popmails.rbのようなプログラムで自動的にPOP3認証を行えば（その場合には、パスワード入力を求めないほうが使いやすいかもしれません）、一定期間内なら自由にSMTPができるというわけです。PPP接続なら、/etc/ppp/ip-up.d/などに置いておくのがよいでしょう（Debianの場合）。

▼リスト4　smtpnote.rb

```
#! /usr/bin/env ruby

if ARGV.size == 0
  STDERR.printf "#$0 time [message...]\n"
  exit 1
end

require 'parsedate'
require 'net/smtp'

now = Time::now
year,mon,day,hour,min,sec =
ParseDate::parsedate(ARGV.shift)
year ||= now.year; mon ||= now.mon; day ||= now.day
hour ||= now.hour; min ||= now.min; sec ||= now.sec
target_time = Time::mktime(year,mon,day,hour,min,sec)

wait = target_time - now
if wait < 0
  STDERR.printf "target time %s is the past.\n",
target_time
  exit 1
end
printf "I will send you a note at %s\n", target_time

if ARGV.size > 0
  title = ARGV.join(" ")
  message = "your specified time %s has come."
else
  title = "note from #$0"
  message = "your specified time %s has come." %
target_time
end

if fork           # fork-exit-setsid はデーモンの常套手段
  exit
end
Process.setsid
sleep(wait)

user = "akamatz@pop.mydomain.jp"
from = "akamatz@pop.mydomain.jp"
server = "localhost"

smtp = Net::SMTP::new(server)
smtp.start
smtp.sendmail(<<BODY, from, user)
From: akamatz
Subject: #{title}

#{message}
BODY
smtp.finish
```

まとめ

電子メールは、おそらく最も古いインターネットアプリケーションのひとつです。時代が変わって、HTTPのような新しいプロトコルが登場しても、その重要性はまったく変わらない重要なツールです。今回はRubyのネットワーク関連ライブラリの一部を紹介し、Rubyはメールも「手軽に」、「お気楽に」プログラミングできることを示せたと思います。「Rubyでメール」、いかがですか？

Column

Rubyを256倍使うための本　邪道編

arton著
アスキー発行
本体価格1200円

待望の2冊目のRuby関連書籍もアスキーからの登場です。「邪道」という一風変わったサブタイトルは、Windowsを主眼としているからだそうです。たしかに、UNIX生まれでUNIX育ちのRubyにとって、Windowsというのは「異国の地」という印象が強くします。まつもとさんもあんまり対応してくれませんし（知らないそうですからしょうがないんですけど）。さて、内容はというと、数ある256倍書籍にひけをとらない「256倍さ加減」です。「Rubyってwindowsに（も）向いてる？」と感じさせる内容に仕上ってます。

本誌の読者でも、大半はなんらかの形でWindowsに触れていると思います。「WindowsでもRubyを使ってみようかなあ」という気分になること請け合いです。特にWordの文章の検索「DOCファイルをgrepしよう（50ページ）」というのはちょっと感動しました。ま、grepもできないフォーマットが、そもそも問題なのですけれど。

[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。2回目の今回は、シェルスクリプトを作成するうえで欠かせない「シェル変数」や、コマンドの出力をシェル変数に格納する際などに使われる「コマンド置換」について説明した後、指定した文字列をファイル名の一部に含むコマンドを表示するスクリプトを作成する。

第2回　シェル変数とコマンド置換

文：大池浩一
Text:Koichi Oike

単なるコマンドの羅列ではなく、プログラム的な処理を行うシェルスクリプトを作成する場合、必須の機能のひとつに「シェル変数」がある。これは、シェルで利用するさまざまなデータを保管する変数で、ユーザーが自由に値を設定できる。前回の記事に登場したコマンド検索パスPATHも、環境変数と呼ばれる一種のシェル変数だ。

シェル変数には、直接値を指定して内容を設定できるほか、シェルの「コマンド置換」機能を利用すれば、コマンドの標準出力をシェル変数に格納できる。設定後は、シェル変数の内容を参照する「変数展開」により、コマンドラインやシェルスクリプトの各部で自由にシェル変数の内容を利用できる。

さらに、シェル変数を条件判断や繰り返しといったシェルのフロー制御構造と組み合わせることで、よりプログラム的な処理が可能だ。フロー制御構造については次回以降に取り上げる。

なお、シェル変数はUNIX系OSのシェルに共通の仕組みだが、Linuxの標準シェルであるbashでは、変数展開の際に値を加工する演算子が豊富に用意されている。このため、sedやPerlなど他のツールの助けを借りることなく、シェル単独で文字列の置換やパターン照合を行える。ただし、一部の処理は、bash 2.0以降でないと利用できないので注意されたい。今回のスクリプトは、古いbash（1.0x）やBourneシェルでも動作するように、こうした演算子を使わないで作成している。

シェル変数を活用する

シェル変数は、bashなどのシェルが利用するデータを保管するための名前付きの格納場所で、シェル自身が値を参照・設定する変数（組み込み変数）のほか、ユーザー独自の変数を自由に定義できる。

コマンドラインでシェル変数が使われることもあるが、たいていの場合はシェルスクリプトの中で使われる。特に、プログラム的な処理を行うシェルスクリプトでは、シェル変数が重要な役割を果たす。たとえば、シェルスクリプト実行時のコマンドライン引数を参照したり（前回登場した「$@」も特殊な組み込み変数の一種）、コマンドの出力を一時保存して後で利用したりといった具合だ。また、シェルスクリプトで条件判断や繰り返し処理を行う際にもシェル変数が使われる。シェル変数によって、シェルスクリプトの柔軟性が大幅にアップするのだ。

以下では、

- **シェル変数を設定・参照（変数展開）する方法**
- **「"」で囲まれた文字列での変数展開について**
- **コマンドの実行結果をシェル変数に格納する方法**
- **変数の内容を参照する際に値を加工する方法**
- **シェルの組み込み変数・環境変数について**

などの説明を行い、最後にシェル変数を利用したスクリプトを作成する。

●シェル変数の設定と参照（変数展開）

ユーザーが新たなシェル変数を定義したり、既存のシェル変数の値を変更するには、コマンドラインやシェルスクリプトで「変数名＝設定値」とすればいい。このとき、「＝」の両側にはスペースを入れないこと。変数名には英数字と「_」（アンダーバー）が使える。英字は大文字と小文字どちらも使える（両者は区別される）が、ユーザーが定義するシェル変数の名前には英小文字を使うのが普通だ。

たとえば、

```
$ hoge=ほげほげ
```

とすると、hogeという名前のシェル変数が定義され、値として「ほげほげ」が設定される。

なお、設定値に空白文字（スペース、タブ、改行）やワイルドカードに使われる「＊」などが含まれる場合は、設定値の両端を「'」（引用符）か「"」（二重引用符）で囲む「クォーティング」を行う必要がある。クォーティングについては次の節で詳しく取り上げる。

一方、シェル変数の値を参照するには、変数名の前に「$」（ダラー）をつけて記述する。このように、シェル変数は設定時と参照時の記述形式が異なるため、他のプログラム言語の経験がある人は注意されたい。

たとえば、上の例を実行した後で、シェル変数hogeの内容を画面に表示するには、組み込みコマンドechoを使って次のようにすればいい。

```
$ echo $hoge
ほげほげ
```

まず、シェルによって「$hoge」が設定値「ほげほげ」に置き換わり（これを「変数展開」と呼ぶ）、その結果がechoに渡されて画面に出力される。変数展開はbashなどのシェルの役割で、echoなどのコマンドは関係していないことに注意されたい。

なお、「$変数名」という表記は簡略形で、正式には「${変数名}」のように変数名を中カッコ「{」と「}」で囲む。正式な書き方が使われるのは、変数名の直後に英数字やアンダーバーが続いて、どの部分までが変数名なのか判別できない場合だ。

たとえば、シェル変数hogeの設定値「ほげほげ」と文字列「hoge」を続けて表示するには、

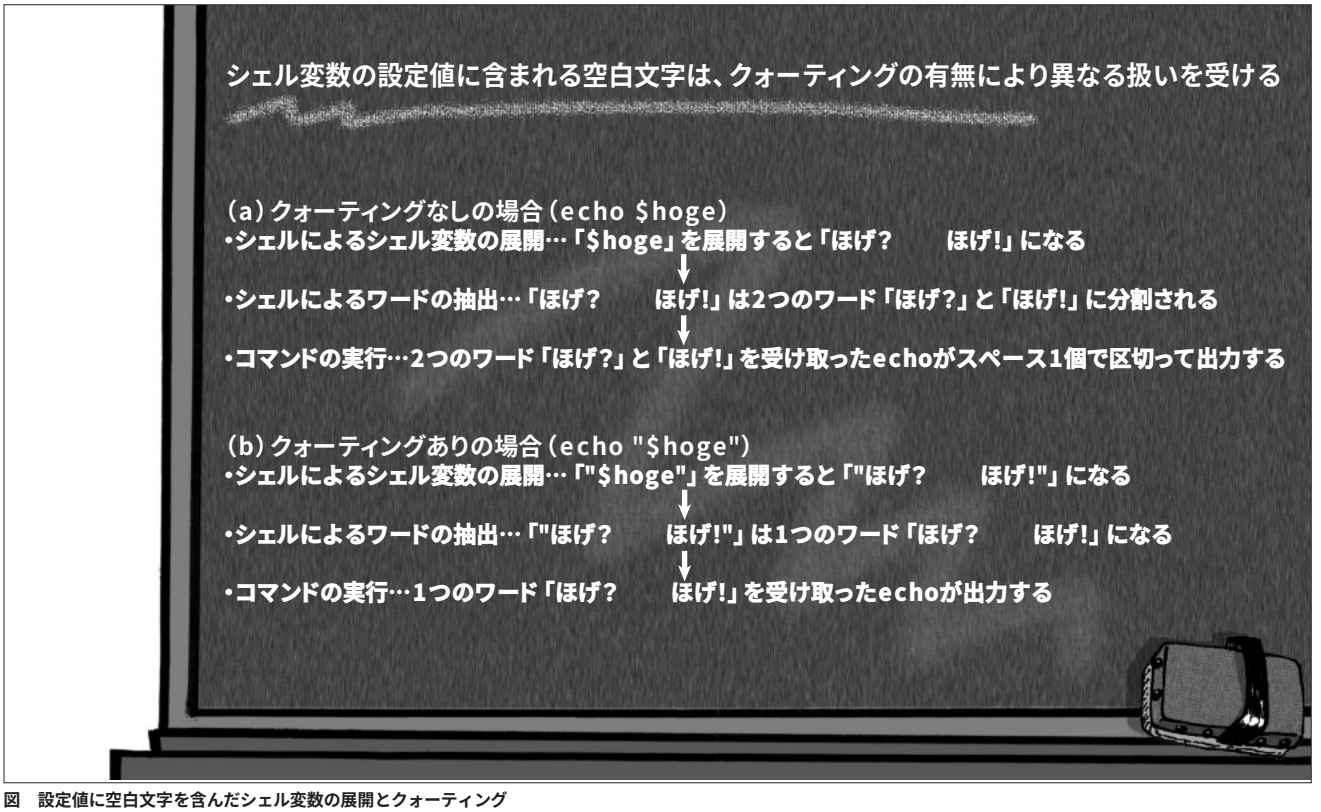


図　設定値に空白文字を含んだシェル変数の展開とクォーティング

```
$ echo ${hoge}hoge
ほげほげhoge
```

とする。

これを中カッコを省略して「$hogehoge」と書いてしまうと、参照されるシェル変数はhogeではなくhogehogeになる。シェル変数hogehogeが定義されていない場合は、長さ0の空文字列に展開されるので、引数なしのechoを実行したときと同じ出力（改行のみ）になる。

この節での要点をまとめると、

- **シェル変数は「変数名＝設定値」として設定する。**
- **シェル変数の値を参照するには「$変数名」（簡略形）あるいは「${変数名}」（正式形）と書く。**

ということになる。

●クォーティングした文字列内での変数展開

前の節でも少し触れたが、クォーティングとは、文字列を「'」（引用符）や「"」（二重引用符）で囲むことをいう。シェルスクリプトを書くうえで重要なテクニックのひとつだ。特に、変数展開と文字列のクォーティングにはいくつか注意すべき関係がある。

クォーティングを行わない場合、ファイル名展開のワイルドカード「＊」「？」や、リダイレクト「＞」「＜」、パイプ「｜」、コマンドラインを複数のワードに分割する際の空白記号（スペース、タブ、改行）など、一部の記号をシェルが特別扱いする。

たとえば、コマンドラインに「＊」を含む文字列をそのまま記述すると、シェルはそれをワイルドカードと見なしてファイル名に展開してしまう。つまり、

```
$ echo ＊
```

すると、カレントディレクトリのファイル一覧（「.」で始まるものを除く）が表示される。

こうした記号自体をコマンドに渡したい場合は、それらを含む文字列をクォーティングして、ファイル名展開の対象外にする必要がある。たとえば、echoで「＊」自体を表示したいなら、「'」でクォーティングして、

```
$ echo '＊'
＊
```

とすればいい。「"＊"」としても同じだ。

クォーティングに使われる2つの記号「'」と「"」の違いは、シェル変数の展開の有無だ。「'」では変数展開が行われず、「"」では行われる。

たとえば、シェル変数hogeに「ほげほげ」が設定されている状態で、

```
$ echo '$hogeは' "$hogeに展開されます"
$hogeは ほげほげに展開されます
```

とすると、両者の違いを確認できるる。「'」でクォーティングされた文字列中の「$hoge」はそのままechoに渡されるのに対し、「"」でクォーティングされた文字列中の「$hoge」は設定値「ほげほげ」に展開されてからechoに渡されるからだ。

なお、シェル変数の設定値に連続した空白文字（スペース、タブ、改行）が含まれている場合、そのまま「$変数名」と書くのと、「"$変数名"」とクォーティングするのとでは、得られる結果が異なるので気をつけよう。

たとえば、シェル変数hogeに「ほげ？　　ほげ！」と、間に複数の空白文字が入っている文字列が設定されている場合、クォーティングなしでは、

```
$ echo $hoge
ほげ？ ほげ！
```

のように、文字列中のスペースが1文字に省略された「ほげ？ ほげ！」が出力される。

これは、シェルが「$hoge」を設定値「ほげ？　　ほげ！」に展開した後、空白文字を区切りとする複数のワード「ほげ？」と「ほげ！」に分割してからechoに渡すからだ。echoは受け取ったそれぞれのワードをスペース1文字で区切って表示する（**図a**）。

一方、シェル変数名を「"」でクォーティングすると、クォーティングされた文字列は1つのワードとして扱われるので、

```
$ echo "$hoge"
ほげ？　　ほげ！
```

のように、echoは「ほげ？　　ほげ！」という空白文字を含んだ1つのワードを受け取ってそのまま表示する（**図b**）。

このほか、タブや改行もスペースに変換されてしまうので、シェル変数の内容に空白文字が含まれる（あるいは含まれる可能性がある）場合、その変数を参照する際には「"」でクォーティングしたほうがいい。

この節での要点をまとめると、

- **「'」で囲んだ文字列では変数展開が行われない。**
- **「"」で囲んだ文字列では変数展開が行われる。**
- **空白文字を含むシェル変数を展開する場合は「"」でクォーティングする。**

ということになる。

●コマンドの実行結果をシェル変数に格納する

コマンドが出力した内容を他のコマンドの入力として使うには、それらのコマンドをパイプ（|）で接続すればいい。それでは、コマンドが出力した内容を他のコマンドの引数として使いたい場合はどうすればいいだろうか。

たとえば、Xを起動する際に使うstartxの情報を得るには、whichを使って、

```
$ which startx
/usr/X11R6/bin/startx
```

としてフルパスを調べ、lsを使って、

```
$ ls -l /usr/X11R6/bin/startx
-rwxr-xr-x  1  root  root   1590  Mar   5   2000
/usr/X11R6/bin/startx
```

とすればいい。

もし、whichの出力「/usr/X11R6/bin/startx」を、lsのコマンドラインで利用する方法があれば、いちいちフルパスをキー入力しないで済む。このような目的のために用意されている機能が、コマンドラインの一部を他のコマンドの出力で置きかえる「コマンド置換」機能だ。

具体的には、両端を「`」（バッククォート）で囲まれたコマンドラインが別のシェル（サブシェル）で実行され、両端の「`」を含む部分が標準出力の内容で置き換えられる。なお、bashには「$(＜コマンド＞)」という別の形式も用意されている（**コラム**を参照）。

上の例をコマンド置換を使って書くと、

```
$ ls -l `which startx`
```

となる。まず、バッククォートで囲まれた「which startx」が実行され、その結果「/usr/X11R6/bin/startx」が得られる。次に、`which startx`がその結果で置換された「ls -l /usr/X11R6/bin/startx」が実行され、startxに関する情報が表示されるわけだ。

なお、バッククォートで囲まれたコマンドが正常に終了しなかった場合、置換後のコマンドラインが想定外の結果を生むことが多いので注意しよう。上の例では、whichがstartxを見つけられないと出力が空になる。その結果、置換後のコマンドラインにはファイル名に相当する部分がなくなり、「ls -l」が実行されてカレントディレクトリのファイル一覧が表示されてしまう。

このような問題を避けるには、コマンド置換の結果を直接利用する代わりに、置換結果が空でないか調べてから、それを利用するコマンドを実行すればいい。シェルスクリプトでコマンド置換が使われる場合は、置換結果をいったんシェル変数に格納し、変数の設定値が空でないか調べ、空の場合には別の内容に差し替えたりすることが多い。

シェル変数にコマンド置換の結果を格納するには、

```
$ hoge=`which startx`
```

のように、設定値の部分にコマンド置換を書けばいい。出力が複数行にわたる場合にも、シェル変数にそのまま格納される。前節で述べたように、シェル変数を参照する際に

Column

コマンド置換に$()が用意されたわけ

bashでコマンド置換に新しい記法「$(＜コマンド＞)」が追加された理由は2つある。1つめの理由は「読みやすさ」だ。従来の記法では始まりと終わりの文字が同じ「`」なので、複数のコマンド置換が使われているような場合、コマンド置換の内外が判別しにくい。この点、新しい記法は始まりと終わりをはっきり判別できる。

もうひとつの理由は、コマンド置換の中でさらにコマンド置換を利用する「ネスト」（入れ子）を容易に記述できること。従来の記法では内部の「`」の前に「\」を付ける必要があり、ネストが深くなるにつれて「\」の数が1、3、7、15、…個とみるみる増えて面倒なことになる。

「"」でクォーティングしないと、空白文字（スペース、タブ、改行）が正確に再現されないので注意しよう。

この節での要点をまとめると、

- **「」で囲んだコマンドはその実行結果で置換される。**
- **シェルスクリプトでは、コマンド置換の結果をいったんシェル変数に格納することが多い。**
- **シェル変数に格納した置換結果を参照する場合は「"」でクォーティングする。**

ということになる。

●変数展開時に値を加工する

シェル変数を展開する際には、単に設定値をそのまま取り出すのではなく、設定値が空の場合に別の値を展開したり、空の場合にはそこで処理を終了するといった仕掛けを組み込むことができる（**表1-a**）。なお、こうした操作を行う場合、「${変数名}」のように変数名を「{」と「}」で囲む正式形を使う必要がある。

よく使われるのが、「${変数名:−文字列}」という置換演算子だ。これは、指定した変数が存在していて、設定値が空でない場合はその値が展開され、指定した変数が存在しない（未定義）か、存在していても内容が空の場合には、後ろの文字列が展開されるというものだ。

たとえば、

```
$ echo "hogeの値は${hoge:−空}です"
```

とすると、変数hogeの値が「ほげほげ」の場合は「hogeの値はほげほげです」と表示され、変数hogeが未定義か設定値が空の場合は「hogeの値は空です」と表示される。

さらに、bash 2.0以降では、文字列の置換やパターン照合、部分文字列の抽出といった強力な操作を行う演算子が追加された（**表1-b**）。いずれも変数の設定値は変更されず、展開される結果だけを加工できる。

たとえば、「${変数名//パターン/文字列}」というパターン照合演算子を使うと、変数の値のうちパターン（ワイルドカード使用可）と一致した部分をすべて指定文字列で置き換えたものが展開される。

また、「${変数名##パターン}」というパターン照合演算子を使うと、変数の値のうち「パターンと一致する最長の先頭部分を取り除いた文字列」が展開されるし、「${変数名%パターン}」では、変数の値のうち「パターンと一致する最短の末尾部分を取り除いた文字列」が展開される。パターンにはワイルドカードが使用可能だ。これらの照合演算子は、フルパスが格納されたシェル変数を参照する際、ディレクトリやファイル名の部分を取り除いて使いたい場合に重宝する。

このほか、「${変数名：位置：長さ}」や「${変数名：位置}」という置換演算子を使うと、設定値の一部だけを展開できる。「位置」の指定が0以上の場合は文字列先頭から（先頭は0）、負の数の場合は文字列末尾から（末尾は-1）数える。「長さ」には切り出す文字数を指定し、省略すると指定位置から文字列末尾までが展開される。

これらは古いbash（1.x）やBorneシェルでは利用できないので、汎用性のあるシェルスクリプトを書く場合には、ほかのコマンドを使って同じ操作を行うようにする。たとえば、フルパスからファイル名部分だけ取り出すにはbasename、ディレクトリ部分だけ取り出すにはdirnameを使う。文字列の置換や部分文字列の抽出にはsedやAWKなどを利用するとよい。

この節での要点をまとめると、

- **変数展開時に値をする演算子が用意されている。**
- **一部の演算子はbash 2.0以降でしか使えない。**

ということになる。

●シェルの組み込み変数と環境変数

シェル変数の中には、シェル自身が値を参照したり、値を自動的に変更したりする「組み込み変数」が含まれている。なお、組み込み変数は一般に英大文字と数字、アンダーバーで構成される。ユーザーが定義する一般のシェル変数の名前に英小文字を使うのは、組み込み変数の名前との衝突を避けるためだ。

組み込み変数の例として、bashがコマンドラインに表示するプロンプト（この連載では左端の「$」）の制御文字列を設定するPS1を取り上げよう。PS1の現在の設定値はユーザーごとに異なるだろうが、たとえば、

```
$ PS1='\w$ '
```

と設定すると、「/home/daichi$」のように、先頭にカレントディレクトリを含むプロンプトに変化する。設定値の中の「\w」は、プロンプト定義におけるカレントディレクトリを表す特殊文字だ。

ところで、Xでktermなどの端末ウィンドウを複数開いていると、どれかひとつでPS1の設定値を変更しても、その他の端末ウィンドウのプロンプトの表示には影響しないことを確かめられる。これは、シェル変数の定義や設定値がシェルごとに独立しているからだ。つまり、あるシェルで設定した変数はそのシェル内でのみ有効で、他のシェルからは参照できない。複数のシェルで同じ名前の変数を定義しても、それらは別々に扱われる。

さらに、通常のシェル変数は、そのシェルから起動したプログラム（サブプロセス）からも参照できない。これではサブプロセスに情報を伝えたい場合に不便なので、サブプロセスに変数名と設定値がコピーされる特殊なシェル変数として「環境変数」が用意されている。

ユーザーのホームディレクトリを示す「HOME」や、実行ファイルの検索パスを格納する「PATH」、カレントディレクトリを示す「PWS」など、組み込み変数の多くは実際にはこうした環境変数だ。

組み込みコマンドのexportを引数なしで実行して、

```
$ export
```

とすると、現在のシェルで使われている環境変数の一覧が設定値とともに表示される。

これらの中には、PWDのように自動的に値が更新されたり、UIDのように読み込みのみ可能なものもあるが、たいていは通常のシェル変数と同様に「変数名＝設定値」で値を設定できる。また、「$変数名」や「${変数名}」と書けば変数展開により設定値を参照可能だ。

なお、ユーザーが定義したシェル変数も環境変数として使用できる。シェル変数を設定した後で、組み込みコマンドのexportを使って「export 変数名」とする。あるいは「export 変数名＝設定値」のように、値の設定を同時に行ってもいい。

たとえば、3Dライブラリの Mesa で3Dアクセラレータの Voodoo3 によるフルスクリーン表示を行うには、環境変数MESA_GLX_FXに「fullscreen」を設定する必要がある。この場合は、

```
$ MESA_GLX_FX=fullscreen
$ export MESA_GLX_FX
```

とするか、

```
$ export MESA_GLX_FX=fullscreen
```

とすればいい。exportを忘れると、MESA_GLX_FXが環境変数にならないため、Mesaライブラリを利用するプログラムに情報が伝わらなくなってしまう。

ところで、前回の記事で説明したように、シェルスクリプトをコマンドのように実行する場合には、元のシェルとは別のシェル（サブシェル）が起動される。サブシェルもサブプロセスの一種なので、元のシェルで設定された環境変数を参照できる。ただし、サブシェルで環境変数の設定値を変更しても、変更後の設定値が有効なのはそのサブシェルとサブシェルから起動されるコマンドだけで、元のシェルには影響しない。

この節での要点をまとめると、

- **組み込み変数の名前は英大文字（と数字、アンダーバー）で構成される。**
- **シェル変数の定義や値はシェルごとに独立している。**
- **サブプロセスに変数名と設定値がコピーされる「環境変数」にするには、「export 変数名」とする。**

(a) Bourneシェルやbash 1.xで利用できるもの	
${変数名:−文字列}	変数が存在し空でない場合はその値、それ以外なら文字列を返す
${変数名:＝文字列}	変数が存在し空でない場合はその値、それ以外なら変数に文字列を設定して返す
${変数名:？文字列}	変数が存在し空でない場合はその値、それ以外なら文字列を出力して終了する
${変数名:＋文字列}	変数が存在し空でない場合は文字列、それ以外なら空を返す
(b) bash 2.0以降で導入されたもの	
${変数名：位置}	変数の値のうち、指定位置から末尾までの部分文字列を返す
${変数名：位置：長さ}	変数の値のうち、指定位置から指定した長さの部分文字列を返す
${変数名#パターン]	変数の値の先頭部分がパターンとマッチした場合、最短マッチ部分を除いて返す
${変数名##パターン]	変数の値の先頭部分がパターンとマッチした場合、最長マッチ部分を除いて返す
${変数名%パターン]	変数の値の末尾部分がパターンとマッチした場合、最短マッチ部分を除いて返す
${変数名%%パターン]	変数の値の末尾部分がパターンとマッチした場合、最長マッチ部分を除いて返す
${変数名/パターン/文字列]	変数の値のうち、パターンに最初にマッチした部分を文字列で置換する
${変数名//パターン/文字列]	変数の値のうち、パターンにマッチしたすべての部分を文字列で置換する

表1　変数展開時に使える置換・パターン照合演算子

・シェルスクリプトで設定したシェル変数は、元のシェルには影響しない。

ということになる。

今月のスクリプト

後半は、既存のスクリプトの内容を解説する「スクリプトを読む」と、与えられたテーマを実現するスクリプトの作成手順を説明する「スクリプトを書く」で構成する。今月はページ数の都合上「スクリプトを読む」はお休みする。「スクリプトを書く」では、シェル変数とコマンド置換を利用した、

・指定した文字列をファイル名の一部に含むコマンドのフルパスを表示する「ezwhich」

を作成する。

●スクリプト「ezwhich」を作成する

指定したコマンドのフルパスを表示するコマンドwhichは、コマンド名が完全にわかっている場合にしか使えない。コマンド名の先頭何文字かわかっていれば、bashのコマンド名補完（Tabキー）を使って候補を一覧表示するという方法もあるが、コマンド名の途中や終わりの部分しかわからない場合にはお手上げだ。

そこで、指定した文字列をファイル名の一部に含むコマンドのフルパスをすべて表示する「ezwhich」を作成することにしよう。

ファイルの検索にはfindを利用する。findは、

・検索を開始するディレクトリ
・検索条件（判別式）
・検索ファイルの処理（アクション）

を「find ディレクトリ 判別式 アクション」という書式で並べると、指定したディレクトリ（複数可）の階層下のすべてのファイルとディレクトリを対象に、判別式にマッチするものだけを検索し、それらに対してアクションを実行するコマンドだ。

今回の場合、ディレクトリには環境変数PATHに設定されたコマンド検索パスのリスト、判別式には実行時に指定した文字列を含むファイル名パターン、アクションにはファイル名のフルパス表示をそれぞれ指定する。そのまま書くと、次のようなコマンドラインになる。

```
$ find $PATH -name パターン -print
```

このコマンドラインにはいくつか問題があるので、以下ではそれらをひとつひとつ解決しながら、シェルスクリプトを作っていこう。

(1) PATHの区切り文字「:」をスペースに変換する

最初の問題は環境変数PATHの展開に関するものだ。PATHの設定値は、

```
$ echo $PATH
/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin:…
```

といった具合に、「:」（コロン）を区切り文字としたディレクトリのリストなので、これをそのまま変数展開したのでは、PATHの設定値全体が1つのディレクトリと見なされ、「そのようなファイルやディレクトリはありません」というエラーになる。

これを解決するには、PATHの設定値に含まれる「:」を、シェルが区切り文字と見なすスペースに変換すればいい。方法はいくつか考えられるが、ここでは、「環境変数IFSの設定値を一時的に変更する」というテクニックを使う。

IFSは、シェルがコマンドラインの内容をワードに分割する際に利用する区切り文字が列挙された環境変数で、通常は空白文字（スペース、タブ、改行）が設定されている。IFSを「:」に変えてからPATHの内容を表示すると、

```
$ IFS=':'
$ echo $PATH
/bin /usr/bin /usr/local/bin /usr/X11R6/bin …
```

のように、ディレクトリの区切りがスペースに変換されて表示される。

これは、シェルが「$PATH」を設定値「/bin:/usr/bin:…」に展開した後、「:」を区切りとする複数のワード「/bin」「/usr/bin」…に分割してechoに渡すからだ。echoは、受け取ったそれぞれのワードをスペース1文字で区切って表示するので、結果的に「:」がスペースで置換されたように見えるわけだ。

なお、上の例のようにコマンドラインでIFSを「:」に変更してしまうと、以後のコマンドの実行に支障をきたす。

しかし、実際のスクリプトでは、「`IFS=' : ' ; echo $PATH`」のように、コマンド置換のサブシェルでIFSを変更するだけなので問題ない。変更後のIFSが影響するのは、サブシェルで実行されるechoだけだ。

(2) 指定したディレクトリのファイルだけを対象にする

次の問題は、findの基本動作に関するものだ。通常、findは、指定したディレクトリの階層下のすべてのファイルとディレクトリを対象に検索を行う。今回は、PATHのリストに含まれるディレクトリのファイルのみを対象にすればよく、サブディレクトリ以下のファイルや、サブディレクトリ自体は無視しなければならない。

findで、サブディレクトリ以下の検索を行わないようにするには、検索ディレクトリレベルを指定するオプション-maxdepthを使って「-maxdepth 1」とする。また、ファイルのみを検索対象にするには、ファイルの種類を指定する判別式-typeを使って「-type f」とすればいい。

(3) パターンの内容を考える

最後の問題は、ファイル名パターンを指定する判別式-nameの引数だ。ここでは、指定した文字列がhogeというシェル変数に格納されているものとする。

パターンに$hogeのみを指定して「-name $hoge」とすると、指定した文字列と同一の名前のファイルだけが検索対象になる。今回は、「指定した文字列をファイル名の一部に含む」ものをすべて検索したいので、これではだめだ。正しいパターンは、任意の文字列にマッチするワイルドカード「＊」を前後に付けた「＊$hoge＊」である。

ただし、そのまま「-name ＊$hoge＊」と書くと、パターンがfindに渡される前に「＊」が展開されてしまう。「＊」自体をfindに渡すには、「"」でクォーティングして、「-name "＊$hoge＊"」とする必要がある。

以上の考察を踏まえた実際のスクリプト「ezwhich」を**リスト1**に示す。1行目はシェルスクリプトに必須の1行なので、実際に実行されるのは2、3行目だけだ。

2行目では、PATHのディレクトリリストを、(1) で説明したテクニックを使って区切り文字の「:」をスペースに変換したうえで、コマンド置換によりシェル変数pathlistに格納している。

3行目では、findを実行してコマンドを検索している。ディレクトリには2行目で設定したシェル変数pathlistの設定値を利用し、「-maxdepth 1」と「-type f」によってサブディレクトリ以下のファイルとサブディレクトリ自体を対象外とし、「-name "＊$1＊"」によって、実行時の最初のコマンドライン引数（「$1」で参照できる）を名前の一部に含むファイル名パターンを指定している。なお、フルパスを表示するアクション「-print」は、アクション省略時の既定値なので、記述を省略している。

ところで、bash 2.0以降で用意されたパターン照合演算子「${変数名//パターン/文字列}」を使うと、IFSを変更することなく、PATHの変数展開時に「:」をスペースに置換できる。このため、スクリプトyawatchは**リスト2**のように書くこともできる。なお、bash 2.0の実行ファイルを「/bin/bash2」とするディストリビューションが多いので、1行目は「#!/bin/bash2」としている。

作成したスクリプト「ezwhich」に実行パーミッションを付け、PATHに含まれるディレクトリに移動すれば、「ezwhich 文字列」という形式で指定した文字列を含むコマンドを一覧表示できる。たとえば、

```
$ ezwhich ho
```

とすると、名前の一部に「ho」を持つコマンドが一覧表示される（**画面1**）。

▼リスト1　スクリプト「ezwhich」の内容

```
1: #!/bin/sh
2: pathlist=`IFS=':';echo $PATH`
3: find $pathlist -maxdepth 1 -type f -name "*$1*"
```

▼リスト2　bash 2.0以降のみで動作する「ezwhich」

```
1: #!/bin/bash2
2: find ${PATH//:/ } -maxdepth 1 -type f -name "*$1*"
```

画面1 ファイル名の一部に「ho」を持つコマンドを表示

Webもゲームも欲ばってみました

Emacsはじめました

第9回　アプリケーション アラカルト（2）

写真も音楽もついているから、さすがにWebは無理でしょうって？　でもあるんです。Emacsの中から使えるブラウザが。いろいろ意見はおありでしょうが、ま、ちょっと見てやってください。今回は、カレンダーや会議中の無聊をなぐさめる（失礼！）小さなゲームも紹介します。

文：佐々木太良
Text : Taroh Sasaki

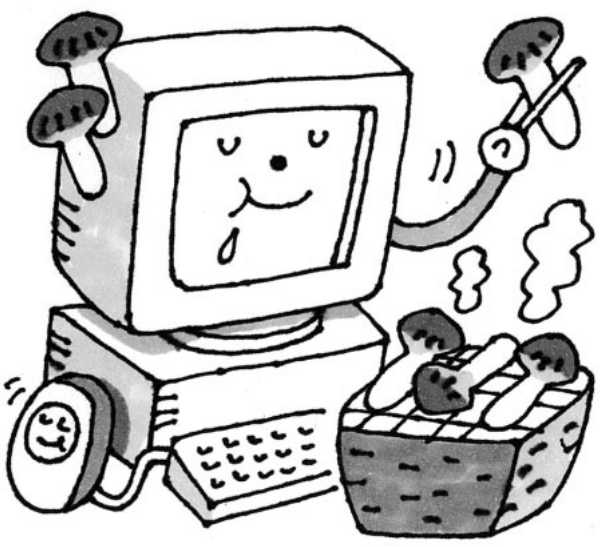


Illustration : Manami Kato

EmacsでWWW

先月は、Emacsの中からIRCとかFTPする例を紹介しました。たしかに、ネットワークサービスのクライアントというのは、GUIアプリケーションを作る（使う）のも面倒くさいし、かといってコマンドラインからアプリケーションを起動するのも面倒くさいし、ということでEmacsの中から使えるものが数々あるようです。

となるとEmacsの中からWeb （WWW）ページを見たい、というのも自然な要求（ほんとかな？）。ただし、Webは写真があって、音声がついて、という世界ですから、文字だけのエディタがブラウザ代わりになるかというと、そこは使い道しだいです。

ちなみに、Linuxのインストールの最中にも、ディストリビューションによってはLynxというWebブラウザのお世話になります。Lynxは、文字だけの表示ができるWebブラウザですが、そこで思い出していただきたいのがXEmacs。エディタの編集画面中に画像がインライン表示できて、Emacsのアプリケーションに応じた使いやすいツールバーやメニューがウィンドウ上部に表示されます。

ここで紹介するw3はEmacs19でも使えるのですが、ここではXEmacs （バージョン20・21）のためのw3を主に紹介します。

■w3のインストール

ディストリビューションに付属しているEmacsには、あらかじめw3マクロがインストールされていることがありますが、もしなかったとしても非常に簡単なので自力でインストールしてしまいましょう。

まずパッケージを入手します。最新のパッケージはw3-4.0pre.44で、ftp://ftp.iij.ad.jp/pub/network/WWW/w3/などから持ってくることができます。ダウンロードには、コマンドラインからftpコマンドを使うか、NetscapeなどのWebブラウザを使いますが、ダウンロードが終わったらrootに変身してください。

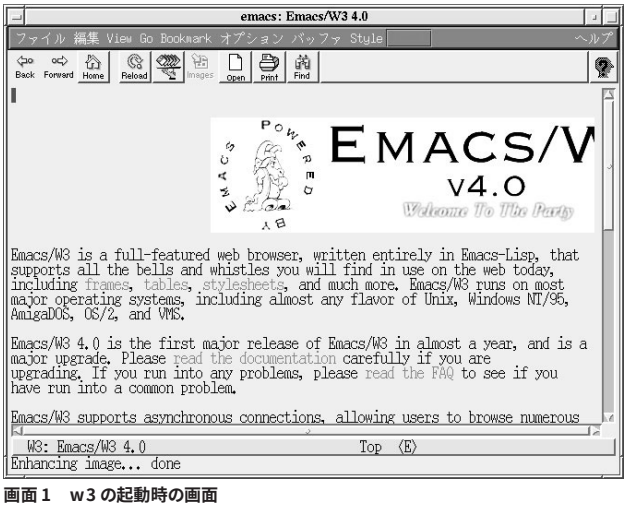
適当なディレクトリにダウンロードしたパッケージを移動して、

```
# tar xvzf w3-4.0pre.44.tar.gz
```

とします。

w3-4.0pre.44というディレクトリができるはずなので、そこに降りてconfigureというコマンド（シェルスクリプト）を実行します。

```
# cd w3-4.0pre.44/
# ./configure
（たくさんのメッセージ）
```



画面1　w3の起動時の画面

あとはmake、make installです。

```
# make
(たくさんのメッセージ)
# make install
(たくさんのメッセージ)
```

以上の操作で、適当なelispディレクトリにelispとそのコンパイル済みのファイル、infoなどが生成されます。

さらに、以下のコマンドを実行すれば、Emacsの起動ファイル.emacsに設定を自動生成してくれます。

```
# make dotemacs
```

この操作で、rootさんの~root/.emacsの最後のほうに、リスト1のような行が追加されているのがわかります。これを切り取って一般ユーザー（あなたのアカウント）のホームディレクトリの~/.emacsに追加しても結構ですし、同じ内容を直接手で書き込んでも結構です。

なお、今までの回でも断りなく使ってきましたが、Linuxのインストール時に開発者向けの選択をしていないと、make （GNU make）がインストールされていないことがあります。それが理由でmakeがうまくいかないときは、まずmakeのパッケージ（RPMなど）をインストールCD-ROMまたはディストリビューションの配布元からダウンロードしてインストールしてください。

C-o	URLを入力する（ヒストリ内補完可能）
〈TAB〉／f	次のリンクへ
M-〈TAB〉／b	前のリンクへ
他のカーソルカーソル移動	移動コマンド
〈RET〉	リンク位置ならリンクを開く
F／B	ヒストリ上の次／前のページへ（メニューバーのForward／Backと同じ）
g	再読み込み

表1 w3の基本操作

■w3の起動と基本操作

M-x w3としてみましょう（**画面1**）。はじめに表示されるのは、オンラインのw3プロジェクトのWebページです。このあとのマウスを使った操作は、他のブラウザを使った経験があれば想像がつくでしょう。また右ボタンドラッグでよりたくさんのメニューが表示されます。

キーボードによるショートカットは表1のとおりです。このなかでも、特に**C-o**は重要でしょう。URLを入力しないことには、ネットサーフィンは始まりませんからね。

■w3の利点と欠点

w3は、バックグラウンドの色も画像も表示できて、表組み、フィールドへの入力（日本語も含め）、フレームや（あまり操作性はよくないのですが）スタイルシート、SSLにも対応しています。

しかし実は、w3が開発されはじめたEmacs19用の当時以来、久しぶりに触ってみたのですが、Netscape Navigatorなどに慣れた人にはつらいかな、というのが正直な感想です。世の中のWebページのかなりが表示に支障があるからです。

その大半は、w3が悪いのではなくて、「Netscape NavigatorとInternet Explorerで見えればいいや」というページのデザインに問題があるようです。ようするに、標準規格であるはずのHTMLが、実質的に特定の2つのアプリケーションソフトでのみ快適に扱えるような使われ方になってきてしまった、といえましょうか。

そうなると、Netscape for Linuxだってあるのに、無理してまでw3を使うことはないや、という声が聞こえてきそうです。ブラウザというのは、何かを見ることが中心で、日本語の入力がたくさん必要なわけでもなく、極端に言えばほとんど情報を残さない（ブックマークやページ内容のコピー&ペーストくらいですか）アプリケーションで

▼リスト1　w3のために.emacs に追加する行

```
;;; Emacs/W3 Configuration
(setq load-path (cons "/usr/share/xemacs/site-lisp" load-path))
(condition-case () (require 'w3-auto "w3-auto") (error nil))
```

/usr/share/xemacs/site-lisp の部分はディストリビューションによって異なります

すから、かなり強引なEmacserの筆者も、今回はあまり強くお勧めしません。それでもw3のメリットというのはあるので、ちょっとご紹介しましょう。

●多国語の取り扱い

多国語、つまり英語以外での2つ以上の言語を同時に扱えるブラウザはほとんどありません。筆者も実は中国語と日本語が混在したページを作ろうとして挫折した経験があります。インライングラフィックスを多用すればそれは可能ですが、あまりにも面倒です。

「ちょっと待ってよ、私のNetscape Navigatorには［表示］－［言語］というメニューがあって、英語以外の2つの言語が表示できるよ」。おや、よくご存じですね。LinuxならばXのフォントがあってちゃんと指定されていれば、どちらか一方は表示できます。が、たとえば「日本人のためのオンラインハングル語講座」というページを作ろうとしたとき、同じページにハングルのフォントと日本語フォントを併せて表示させることは不可能なのです。また複数言語の自動判別もほとんどなされていません。

Emacsのw3では、「HTMLファイルをダウンロードして解析し、バッファに表示する」というノリなので、ブラウザとしてはあまり快適ではない代わりに、本文に書かれた言語の自動判別なんかはお手のものです。基本的には、複数言語が1バッファ（1ファイル）中に混在できるように設計されたことがその勝因でしょう。

●HTTP以外のプロトコルでの操作統一

Netscape Navigatorなどで「メールは[taroh@taroh.org]まで」などと書かれたリンクを開いたとき、付属のメーラが立ち上がって嫌な思いをしたことがありませんか？　Xの上だと満足に日本語が入力できなかったり、ふだんのシグネチャを付けるのが面倒だったりします。そもそもWebブラウザにメールサーバの設定をしていない人だっているかもしれません。

w3では、HTTP以外のいくつかのプロトコルの場合でも適当なモードを自動で呼び出すようになっています。そう、ここで、前回取り上げたネットワークアプリケーションのいろいろなクライアントとなるモードの説明が活きてくるわけです。

残念ながら、w3のメールを送るためのリンク（mailto:）からは、WanderlustやMewは立ち上がりませんが、Emacsのメールモードは比較的楽に理解できると思います。WanderlustやMewの自動立ち上げは、フックを変更することで簡単にできるでしょう。将来のw3またはメーラ側の実装では標準的にふだん使っているメーラが立ち上がることも考えられますので、今後に期待しましょう。

●将来への期待

Netscape NavigatorやInternet ExplorerがJavaを解釈・実行し、膨大な数のプラグインを利用しているWebの現在の状況からは、シンプルなアプリケーションがモンスターと化していくような印象を受けます。プラグインの大半はインラインでWWW以外のプロトコルを扱わなくていいのに、と思うものばかりです。たとえば、ブラウザにTVが貼り付いていなくてもいいのではないでしょうか。ストリーム再生ソフトを起動するなど、Emacsから別のアプリケーションを起動してもよいでしょう。そうでない「見せる」ことが主目的のページは現在も大半を占めていますし、今後も残っていくでしょう。Yahoo! などを見ると、データベースとの連携など、サーバサイドでデータを加工してシンプルなブラウザと組み合わせる応用は、まだまだ発展の可能性があると思います。

w3のinfoを眺めていると、将来への課題が山積みされたまま開発が停滞しているように見えます。誰かがムキになって開発すれば、「見る」機能はかなり強化されるのではないでしょうか。

さきほど、「見る」だけの場面ではエディタならではのメリットが少ないと書きましたが、翻訳サーバのEmacsフロントエンドとの連携などは今でもできますし、Emacspeakに音声でページ内容を読み上げさせるプランもあるようです。これについては、機会があれば紹介しましょう。

●Webページを作るときには?

現在のw3モードは、表示された内容をWYSIWYGで編集すればページの更新が簡単、というようにはできていません。最強の日本語エディタであるXEmacs/Muleには、このあたりの連携も期待されるところです。

ところで実は、w3とは別に、EmacsにはHTML記述を支援するモードがいくつかあります。筆者が愛用しているYaHTMLモードなどは日本人によって作られたもので優れたものですが、どちらかというとw3とは無関係なのであとの回で紹介することにしましょう。Emacserとしては恥ずかしいのですが、これらのモードで記述したページをNetscapeなどのブラウザ専用アプリケーションで確認しつつ修正、というのが現実的には効率がよいようです。

画面2　カレンダー

カレンダーとスケジュール

Emacsをアプリケーションとして使う話はまだまだ尽きませんが、このシリーズの最後に、標準で付属しているアクセサリー的な小物の話をしましょう。最初は小物の中でもかなり実用的と思われる、カレンダーです。ちょっとカレンダーが見たくなったとき、UNIX系OSでは、

```
% cal
```
（今月のカレンダー）

```
% cal 1 2001
```
（2001年1月のカレンダー）

```
% cal 2010
```
（2010年のカレンダー）

とかが利用できますが、Emacsでは**M-x calendar**というコマンドがあります。これを実行するとウィンドウが2つに割れ（**画面2**）、前月、今月、来月の3カ月分のカレンダーが表示されます。カレンダー操作の基本的な操作を表2に示します。

面白いのは日の出・日の入り、月齢など、各種の暦を表示できることです（コラム「いろいろな暦とcalendarのすごい機能」参照）。太公望には実用的でしょう、という話はさておいて、このカレンダーでもっとも実用的なのは、簡単なスケジュール管理ができることでしょう。もちろん専用の多機能スケジュールソフトにはかないませんが、筆者はなによりEmacsを開いているときにスケジュールを管理できるのが便利だと感じています。つまり、メールでやってきた予定（MewかWanderlustで読みますよね）、会議中に入る予定（もちろんEmacsでメモを取っています）などなど、予定を書き込んだり見たりするところにEmacsあり、という生活だからです。日本語を入力するのにももちろん、便利ですよね。

スケジュールを記録するファイルは、ホームディレクトリの~/diaryという名前のテキストファイルです（**リスト2**）。このファイルには**i d**でその日の予定を追加できますが、もちろんエディタで直接いじってもかまいません。なお〈s〉でスケジュール全体が表示され、〈d〉では該当する日付の行だけが表示されます。後者の場合、Emacsのナローイング（必要な行だけ表示する）機能を用いているだけで、残りの行が消えてなくなっているわけではありません。

所定のフォーマットの日付よりあとは、すべて表示に使われるだけなので、好きに書けばよいでしょう。たとえば“13:00”と書いてあっても、calendarコマンドが認識するのはその前の日付までです。ただし後述するように他のツールを自作して組み合わせる場合、たとえば予定時間の1時間前にメールを発信するときには、フォーマットを統一したほうがよいかもしれません。

%記号で始まっている行は、Emacs内では無視されます。毎年ある個人的な記念日や休日などは、上記の例の先頭に示すように、年を省略すると毎年のその日に表示され

▼リスト2　スケジュールファイル~/diaryの例

```
%% (diary-anniversary 3 8 1997) たりゃ誕生日
Jan 1 元日
Jan 15 成人の日
Mar 20 春分の日
....
Friday 15:00- 定例勉強会
Nov 10, 2000 Linux Magazine 12月号発売日
Nov 11, 2000 13:00- ○×会議
```

コマンド	説明
m	予定のあるすべての日をマーク
d	その日の予定を表示
C-f／C-b	翌日／前日へ
C-n／C-p	次週／前週へ
C-v／M-v	3カ月単位で次／前へ
..	今日の日付へ
g d	指定した日付へ
s	入力されているすべての予定を表示
i d	その日に予定を挿入
?	calendarのinfoを表示（M-x info からemacs→Calendar/diaryを表示したときと同じ）

表2 カレンダー操作の基本コマンド

ます。なお、アメリカの休日はあらかじめ登録されていますが、日本の休日はサポート外ですのでこのようにスケジュールファイルに記入しているわけです。それにしても、国民の休日は年によって変わることがあって困ったものですね。

毎週同じ曜日にある予定は、曜日のみの記述でOKです。

このファイルはテキストファイルなので、特にEmacsからcalendarを起動しなくても編集できます。筆者は、以前にザウルスの赤外線通信で取り込んだ予定を自作プログラムで変換し、calendarで使っていたことがありました。現在はラップトップUNIXマシンを持ち歩いているためスタンドアロンのPDAを使っていません。あえてここで実例を示すことはしませんが、赤外線ポートが/dev/cua? などに割り当てられていてPDAと同じ変調方式を利用できるマシンならば（BIOSなどで設定できます）、minicom などの通信ソフトでスケジュールの一括送受信ができるはずです。またPDA形式のスケジュールファイルとEmacsのcalendar形式のファイルとの変換ツールは、Perlを勉強した人ならわりと簡単に書くことができるでしょう。

さてEmacsのcalendarコマンドからi dで予定を挿入していると、このファイルの順番が挿入順であって日付順にならないのが気になってきます。また、1年以上前の予定はバックアップに移してしまいたいものです。**リスト3**は、筆者がふだん使っている自慢の自作スクリプトです。

- **手を加える前の状態を“~/tmp/Backups/diary.今日の日付”というファイルにバックアップする**
- **~/diaryの中身を日付順にソートして（ただしコメント、毎年ある予定はかならず先頭にもってくる）**
- **1年以上経った予定は削除**

▼リスト2　スケジュールファイル~/diaryの例

```
#! /bin/sh

DIARY=$HOME/diary
TMP=$HOME/tmp
TYEAR=`date '+%Y'`
TMONDAY=`date '+%m%d'`
BACKUP="$HOME/tmp/Backups/diary.$TYEAR$TMONDAY"
cat $DIARY |\
gawk '
BEGIN {
  tyear = '$TYEAR';
  tmonday = '$TMONDAY';
}
/^[      ]*$/ {
         next;
}
{
    mon =
index("JanFebMarAprMayJunJulAugSepOctNovDec", $1);
    if (mon == 0) {
        mon = 0;
        day = 0;
        year = 0;
    } else {
        mon = (mon + 2) / 3;
        day = $2 + 0;
        year = $3;
    }
    if (year == 0 || tyear <= year || \
    (year == tyear - 1 && tmonday <= mon * 100 + day))
{
        printf("%04d%02d%02d|%s\n", year, mon, day,
$0);
    }
}' |\
sort |\
sed -e 's/^[^|]*|//' > $DIARY.t
mv $DIARY $BACKUP
chmod 0600 $BACKUP
mv $DIARY.t $DIARY
chmod 0600 $DIARY

exit 0
```

Column

いろいろな暦とcalendarのすごい機能

ふだんは使用する機会があまりありませんが、Emacsのcalendarは本文に述べた以外にも、以下の表のようなさまざまな機能を備えています。

コマンド	説明	コマンド	説明
M-=	マークとポイント（カーソル）の間の日付を計算	p h	ヘブライ暦を表示
t m/t y	紙に出力して美麗なカレンダー（TeXのソース）を出力する	p i	イスラム暦を表示
S	その日のある地点での日の出・日の入りを表示	p f	フランス革命暦を表示
M	表示されている月の新月・満月・半月などの日付と時刻を表示	p k	コプト暦を表示
p j	ユリウス日を表示（通常のカレンダー）	p e	エチオピア暦を表示
p c	ISOカレンダーによる日付を表示	p p	ペルシャ暦を表示
p C	中国の暦を表示	p m	マヤ暦を表示

という操作をするものです。週に一度程度、cronなどで起動してもよいかもしれません。なお、シェルスクリプトを使っていることにこれといった理由はありません。Perlなどで書き直すのも簡単でしょう。

■日数の計算

日数の計算機能は、範囲指定コマンドのようにして使用します。生まれて何日経過したかを調べてみましょう。何千日目・何万日目という記念日は誕生日より希少なはずなのに、その日と気づかずに過ごしてしまいがちですよね。

まず今日の日付にマークを打ちます。ポイントが今日の日付になければまず.で今日に移動して、**C-SPC**でマークをセットします。次に**g d**をタイプし、ミニバッファに表示されるプロンプトに年、月、日を入力します。月名は数字でなく名前で入力しますが、補完も効きますので英語に弱い人も安心です（笑）。

最後に**ESC =**で経過日数が表示されます。

■太陽と月の動き

これはいわずもがなでしょう。日の出・日の入りは地点によって違いますから、緯度・経度を入力する必要があります。東経・北緯が＋、西経・南緯が－です。また分と秒の桁は小数点で表わす（分÷60と秒÷3600を緯度・経度に足す）必要があります。ほんとうは高度が違うと日の出・日の入りの時刻も違うのですが、calendar機能では高度の指定まではできないようです。山の上に初日の出を見に行く前に計算したのに高度が違ってがっかり、ということがないようにしてください。

■TeXのカレンダー

TeXはWYSIWYG（見た目どおり出力される）ワープロとは異なり、HTMLなどのようにマークアップ型の組版ソフトです。このツールを使うと、原稿となるソースファイルがcalendar.texという名前のバッファに作成されるので、あとは**C-x C-w**で名前を付けてセーブし、TeXのタイプセットを通してやれば、美麗に出力可能な最終出力ファイルのできあがり（**図1**）。あとの回で紹介するYaTeXモードが使用できれば、一度もEmacsから抜けずに（Xのカーソルを動かさずに）出力までできます。いろいろなカレンダーが作れるので、詳しくはinfoを参照してください。

■各国の暦

表3にあるような各国の暦を計算できます。通常使っている、いわゆる太陽暦による日付は**p j**で表示可能です。このほかに、日本人にとってなじみが深いのは中国の暦（いわゆる旧暦）でしょう。年・月・日ごとの十干・十二支（いわゆる干支）もカッコ内にただし書きされますが、残念ながら中国語を英語表記したものなので、日本人にはわかりにくいかもしれません。なお、Emacsバージョン19までの干支の計算は狂っているので、注意が必要です。

日本人ならこのほかに六曜（計算は難しいのですが――大安とか仏滅とか）が欲しいところです。

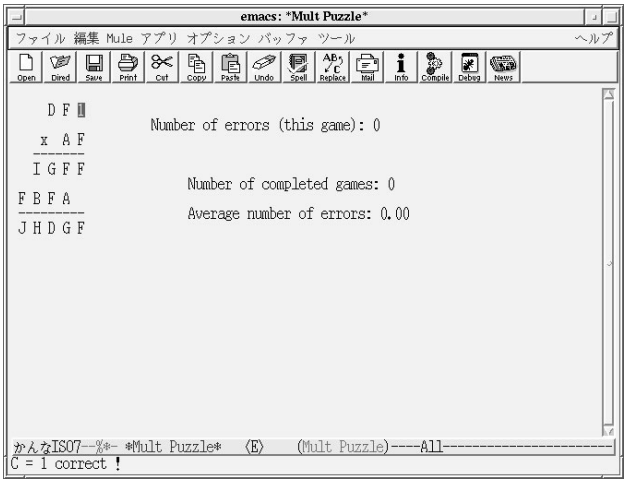
余談になりますが、イスラム暦などは完全に月の満ち欠けで日付を決めている（太陰暦）ので、月のサイクルが29.5日に固定されています。地球が太陽の周りを回る日数（365.2422日）と月のサイクルは無関係なので、16年で夏冬が逆転してしまいます（確かめてみてください）。

日本や中国の暦では、太陽太陰暦といって、月の満ち欠けで日付を決めるのに加え、適度にうるう月をはさんで夏冬が逆転しないように調整しています。古代には王朝が変わると暦のシステムも変えることがよくあったようで、太陽太陰暦のような複雑な暦を完成・運用していたのは、やはり国の歴史が長い中国や日本の特徴だといえましょう。

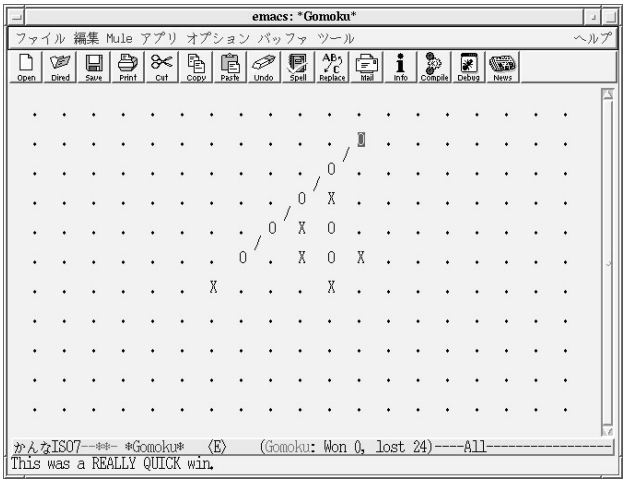
図1　Texのカレンダー

十干	十二支
甲 Jia	子 Zi
乙 Yi	丑 Chou
丙 Bing	寅 Yin
丁 Ding	卯 Mao
戊 Wu	辰 Chen
己 Ji	巳 Si
庚 Geng	午 Wu
辛 Xin	未 Wei
壬 Ren	申 Shen
癸 Gui	酉 You
	戌 Xu
	亥 Hai

表3　干支と十二支



画面3　覆面算



画面4　五目並べ

Emacsでゲーム

ちょっとまとまったツール群が続いたので、ここらで息抜きをちょっと。

EmacsはEmacs Lisp（elisp）という言語で書かれているという話は前回しましたが、プログラミング言語を見るとゲームを書きたくなるのはハッカーの性（ほんとか？）。というわけで、ごく基本のEmacsのelispのディストリビューションのセットにもはじめからちょっとしたゲームが付属しています。

■ハノイの塔

これは人間が解くパズルではなく、Emacsにパズルを解かせて喜ぼう（驚こう）、というものです。

いつの時代のことか、ベトナムはハノイにある寺院に、大きさの異なる円盤が何十枚か挿さった3本の柱がある（あった）そうです。この円盤は、空いているところか、より大きな円盤の上にしか動かせないようになっています。ハノイの寺院では、何人かの僧侶が昼夜を分かたず、休むことなく交代で円盤を動かし続けています。

このようなルールで塔のすべての円盤をほかの棒に動かすには、何手かかり、どのようにしたらよいでしょうか——というのは、プログラミング言語の初級の練習問題でよく取り扱われるテーマです。Emacsでは文字だけで、円盤を横から見た状態で表現しています。

M-x hanoiとしてみてください。7段のハノイの塔が解かれるようすがアニメーションで表示されます。**C-u 1 0 M-x hanoi**とすると、10段のハノイの塔の移動が始まります。

ところで件のハノイの塔の円盤は何十枚かあるらしいのですが、柱から柱にすべての円盤が移動し終わった瞬間、この世が終わるのだそうです。たしかに円盤が60枚あるとすれば、1秒に1枚円盤を動かし続けても365億年ほどかかりそうです（宇宙の寿命より長い？）。Emacsは僧侶よりは速く動かせるでしょうが、あまり枚数を大きくしすぎると終わりません。といっても**C-g**で止まりますが（笑）。

■覆面算

M-x mpuzとすると、覆面算の問題が表示されます（**画面3**）。紙の上で解く覆面算は、不明な文字に数字を仮定しつつ、矛盾を解消しながら答えを見つけていくのがふつうだと思いますが、Emacs上では気軽に試行錯誤をしながら解くことができます。ミニバッファに「your try?」と表示されるので、Aに当てはまる文字が“1”だと思ったら〈A〉〈1〉とタイプします。

■五目並べ

M-x gomokuを実行して先手（〈n〉）・後手（〈y〉）を選んだあとは、平均的な日本人、かつEmacs使いならば、解説は不要でしょう（**画面4**）。置き石は〈x〉です。なおEmacsのポイント移動だけではなく、viのカーソル移動（上下左右 = 〈k〉〈j〉〈h〉〈l〉）も使用できます。nethackというキャラクタベースのアドベンチャーゲームにおなじみの方は、同様の斜め移動も可能です(y/u/b/nで左上/右上/左下/右下)。

gomokuはelispで出来ていると思えないくらい（？）強くて、おまけに再勝負（またM-x gomokuを実行します）で勝ちどきに吐くメッセージがなかなか憎たらしいです。

■言葉遊び

ここからは、英語を母国語としない人にはあまり面白くないかもしれません。

M-x doctorとすると、かの有名なElizaという人工無脳？　が会話の相手をしてくれます（**画面5**）。会話文の最後では〈RET〉を2回タイプしましょう。Elizaというのは20年以上前に有名になったプログラムですが、比較的簡単なルールで入力文に応答する文を出力します。それがあたかも「悩み相談」の相談員っぽくみえるというか、実際にそれでメンタルケア効果があったとか、そういう話をきいたことがあります。が、ちょっと試してみるとあまりにバカっぽい答えを返すので、本当にメンタルケア効果があるかどうかは疑問です。

M-x yowとすると変な文章が表示されます。

M-x dissociated-pressは、「四散新聞」とか「まぜこぜ新聞」といわれる遊びを起動します。バッファ中の文字列の並びが同じところで文章を切り離し、適当にすげかえて出鱈目な内容にしようという趣向です。ところでこの言葉遊びは、日本語でも楽しめます（開発者が意図したアルゴリズムで動いているのかどうかわかりません）。原理を説明してもよくわからないかもしれませんが、適当な文書をバッファに用意してM-x dissociated-pressとしてみましょう。

⋮

原理を説明してもよくわからないかもしれませんが、
適当な文書をバッファに用意して M-x dissociated-pressociated-press
としてみましょらはそれがあたかも「悩み相談」の相談」の相談」の相談」の
相談員っぽくみえるというか

⋮

編集長！　今月の原稿は壊れてしまったので、ここまでいいですか？　え、ダメ？　うーむ、そうなんですよ。バッファ＊Dissociation＊というのは、元のバッファの四散新聞化したものを出力しただけで、元のバッファ自体はそのまま残っています。

何行か四散新聞化すると続けるかどうか聞いてきますので、この辺で許してやるか、と思ったら〈n〉をどうぞ。

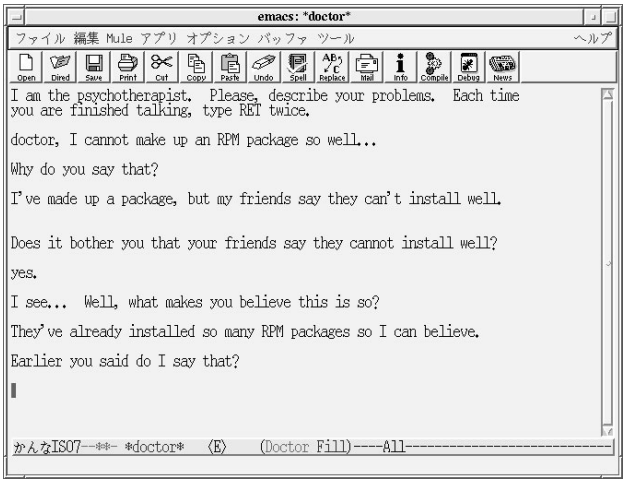
■アドベンチャーゲーム

M-x dunnetで、テキストタイプのアドベンチャーゲームが起動します。テキストタイプというのは、

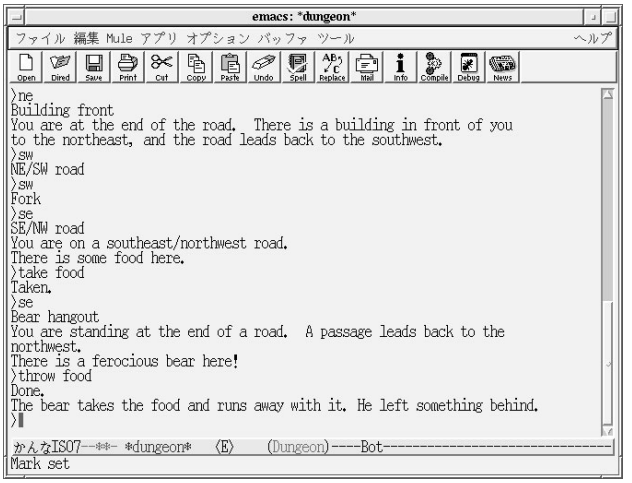
•状況が説明される
•アクションを入力する

という繰り返しで進んでいくゲームで、古くはUNIXのadventure（zork）などというゲームに端を発しています（**画面6**）。ファミコン世代の方々には、「なんだ、グラフィックスは？」といわれそうですが、実際に解き進むためには地図を書いてメモを取っていかなければなりません。UNIX系OSのrogue/hackは、テキストタイプよりはちょっとだけビジュアル系ですが、いずれにしても表示は文字を使ったもので、グラフィックスがなく、それだけ頭の中に好きな図が描けて、想像力が広がります。

さてdunnetは、前出のadventureなどよりはちょっとマニアック（？）な内容になっています。英語ができないと面白くないという向きには、あとの回で紹介しようと思っている翻訳ソフトなどとの組み合わせが便利かもしれません。コマンドなどはほとんど単語1～2つでOKですので心配はありません。



画面5　人工無脳Eliza



画面6　アドベンチャーゲーム

Linux日記

第15回　メール配送（2）

先月は同一ホスト内でのメール配送についてのお話でした。いよいよ今月からは、異なったホスト間でのメール配送について話を進めていきましょう。

illustration ; Aki

文：榊 正憲
Text : Masanori Sakaki

9月の東海地方の大雨被害の数日後、筆者の住む神奈川県でも（名古屋ほどではないが）まとまった雨が降った。浸水した地域もあったようである。我が家は浸水こそしなかったものの、停電をくらった。この時の雨は雷がすごかった。これがどこかの変電所に落ちたらしい。軽い落雷の場合は、送電系の安全装置が作動して停電になる。この場合は1分とか5分で自動復帰するはずなのだが、一向に回復しない。どうやら修理を要する停電のようだ。

停電したのは夕方4時半ごろで、まだ外が明るい時間だったのが幸いだった。いくつかある懐中電灯を難なく見つけることができた。暗くなってからの停電だと、ライターの明りで探すことになるところだった。結局、回復したのは夜7時半ごろで、3時間あまりの停電だった。途中からCDラジカセに電池を入れて、ラジオだけは聞けるようにしたのだが、大相撲とオリンピックしかやっていなかった。この停電のおかげで、YAWARAちゃんの試合を生中継で見ることができなかったのが残念である。

考えてみると、今の日本は本当に停電が少なくなったと思う。台風、地震、水害などが直撃しない限り、停電というのは完全に非日常的なできごとになった。災害による停電は、かなり地域差があるだろうが、関東地方は少ない部類だと思う。電気設備の点検などによる停電を除けば、これだけ長時間の停電というのは久しぶりだった。以前にくらったのはいつ頃だっただろうかと考えても、記憶にないくらいだ。

停電による被害はほとんどなかった。うちはUPS（Uninterruptable Power Supply：無停電電源装置）を入れていないので、当然のことながら停電中はすべてのコンピュータが停止していた。正当なシャットダウン手続きを行わなかったことによる障害は特になかったようだ。うちは、通信系には被害はでなかったが、知り合いのところでは、落雷の影響でターミナルアダプタだかDSUだかが死んだらしい。

たちが悪かったのは、この停電よりも、その前後に多発した瞬断である。リブートしたり固まってしまったりを繰り返して大変だった。また、この瞬断の嵐に耐えられなかったのか、DECのキーボードが1台死んでしまい、さらにはメールサーバマシンの調子が悪くなった。

3時間の停電により、我が家がいかに電気に依存した生活をしているかを実感した。平常通り動いていたのは、ガスコンロと電池式の時計だけというありさまだった。

■UPS

さて、コンピュータ環境を停電に強いものにするにはどうすればいいか？まずはUPSの導入である。これは、停電時にバッテリから電力を供給する無停電電源装置である。バッテリは直流電源なので、AC（交流）出力を得るために、インバータ（AC／DC入力からAC電力を出力する電源装置）を内蔵している。

UPSには、大きく分けて2種類の方式がある。1つは、非停電時は入力AC電源をそのまま出力し、停電時にのみインバータを使うというものだ。この方式はAC給電時の損失が少ないという特徴があるが、給電系統の切り換えを必要とするので、ごく短い瞬断などを検出できないことがある（あるいは検出できても、切り換えが間に合わない）。もうひとつの方式は、常時インバータを運転し、出力は常にインバータから得るものだ。通常時はAC電源でインバータを駆動し、停電時はバッテリでインバータを駆動する。この方式はインバータによる電力損失があるため、非停電時の消費電力が増えるという欠点があるが、AC系統からインバータ系統に切り換える必要がないので、極めて短い瞬断でも影響を受けない。またインバータは安定化電源としても機能するので、AC電源が不安定な環境（電圧変動が大きいとか電圧がもともと低めなど）でも機器に正常な電力を供給することができる（**図1**）。

一般に、家庭用、オフィス用などの小容量タイプは切り換え式、コンピュータ室などに使う大容量タイプは常時インバータ式が多い。

■UPSの容量

UPSを選定する時は、2つの点を考えなければならない。出力容量と、バッテリ容量である。UPS出力容量は、最大何アンペアのAC出力を供給できるか、あるいは合計何ワットの機器を運転できるかという定格である。たとえば500W、あるいは500VAであれば、簡単にいえば100V×5Aの給電能力があるということだ（WとVAの違いについては、電気工学に詳しい人に聞いてほしい）。もう1つのパラメータはバッテリ容量で、UPSのカタログなどでは、最大出力時の給電時間で表されることが多いが、細かな資料を見れば、バッテリのVAh（ボルトアンペアアワー）というパラメータを知ることができる。これは、バッテリ電圧（V）に定格放電容量（Ah）を掛けたものだ。たとえば100VAhなら、100Vで1Aの出力なら1時間、100Vで0.5Aなら2時間供給できるということになる。

ただし、バッテリ容量の数字はあくまでも目安だ。バッテリの放電性能は、簡単な割り算では求められない。一般にバッテリは、大電流を流すほど実容量が目減りするからだ。たとえば100Vで1Aを1時間供給できるバッテリで100V、2Aの放電を行うと、VAhの計算なら30分保つことになるが、実際には15分とか20分しか保たない。逆に100V、0.5Aの放電なら、2時間以上保つ。これでは実際の持続時間の求めようがないので、バッテリ容量には、N時間放電率という関連パラメータが併記されている。これは、N時間かけて放電した時のバッテリ容量で、バッテリの種類や用途にもよるが、5時間という数字が多いようだ。たとえば5時間放電率で100VAhなら、100V、0.2Aを5時間供給できるということになる。

■より頑強な電源を考える

さて、UPSを入れれば、とりあえず停電と同時にマシンが落ちるという事態は避けられる。しかし、バッテリによる駆動時間は知れたものだ。そのため、停電を検出したら（あるいはバッテリ容量低下を検出したら）、システムをシャットダウンするというのが標準的な手続きだ。これを行うために、UPSはコンピュータに通知を送るための機構を持っている。単純なものなら

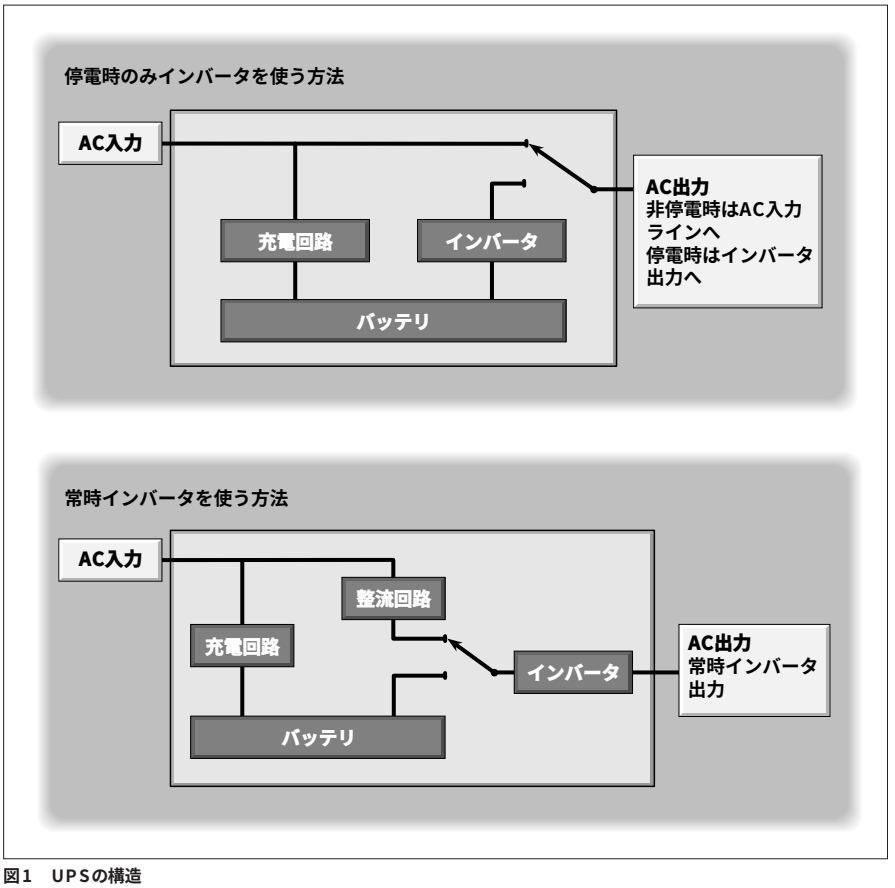


図1　UPSの構造

ただのメカニカル接点だ。この場合、この接点をシリアルポートの制御端子に接続し、このポートを監視しているプログラムがシステムをシャットダウンさせる。より高級なUPSなら、データ通信を行えるシリアルポートや各種ネットワーク経由で、ユーザーが指定したコマンドを発行することができる。

一般的な環境であれば、停電を検出して、重要なサーバマシン類を安全にシャットダウンするというのが、事実上最善の策であろう。だが、用途によっては、シャットダウンせずに、サービスを継続させたいという場合もある。大容量のUPSを導入することにより、数時間程度なら各種機器を運転することができるが、それ以上運転を続けたい場合はどうするか？　発電機の出番である。停電の検出と同時に、ディーゼル発電機など（必要なら、ガスタービン発電機という選択肢もある）を起動させる。そして、UPSのバッテリが上がる前に、発電機給電に切り換えるのである（この場合、発電機もUPSの構成モジュールになる）。秋葉原のショップのカタログには載っていないが、電源機器を扱っている会社のホームページを見れば、各種製品を見ることができる。

そんな大げさなという人が大半だろう。まったくそのとおり。だが、個人の一戸建住宅や自社ビルならまた別の選択肢もある。最近はやりの太陽光発電だ（燃料電池システムも期待しているのだが、まだ当分は無理だろう）。ソーラーパネルから得た電力をバッテリに蓄え、インバータでAC電源を供給するシステムだ。おもに省エネ用に注目されているが、筆者は非常用電源としての価値も高いと考えている。最低限の照明機器、テレビやコンピュータなどの情報通信機器を太陽光発電系統につないでおくのである。ふだんは、太陽光と商用電源を併用し、停電時は太陽光とバッテリから電力を供給するのである。バッテリ容量は、UPSに比べれば大きくできる。また、個人住宅なら1kVA程度のガソリン発電機（ホームセンターで数万円で買える）を接続できるようにしておけば文句なしだろう。筆者は、一戸建を建てる機会があれば、ぜひこれをやってみようと思っている。

■ノートPCサーバ

素朴な停電対策として、ノートPCを使うという方法がある。大規模なファイルサーバやデータベースサーバなどには不向きだが、メールサーバやネームサーバなら十分実用になる（もちろん、ルータやTA、ハブなどはUPSに接続しなければならないが、たいした電力ではないので、小型のUPSで済む）。ノートPCは、AC電源が切れれば、自動的に内蔵バッテリ動作に切り替わる。バッテリが空になれば、自動的にスタンバイモードになる。停電対策としては申し分ない。

小規模サイトでネームサーバやメールサーバを運用する場合、お古のノートPCを使うというのは案外賢いやり方だ。この種のサーバは、処理能力はさほど求められない。また、デスクトップ機に比べて、騒音や消費電力が小さいので24時間運転に向いている。場所を取らないというのも魅力だ。

■Sendmailの話の続き

さて、Sendmailの話の2回目だ。前回は、mail.localを使ってローカルホスト上でメール配信を行う実験をしてみた。ところが、前回の原稿を渡した後、編集氏からいわれた。

「今どきのLinuxはmail.localを使ってませんよ」

ぅをっと、そうきたか。確かに、手元にあるRed Hatのsendmail.cfを見ると、代わりにprocmailというプログラムが使われているようだ（筆者はFreeBSDを使っている）。このことを知った時点で、原稿はかなり進んでしまっていたので、procmailについてはおいおい調べるとして、しばらくはmail.localを使って話を進めていこう。重要なのは、mail.localかprocmailかではなく、配信エージェントという考え方なのだから。以後、一般論的な解説で、ローカル配信エージェントという意味で使う時は、総称的な意味でbinMailという用語を使うことにする。

話は変わるが、大事な注意がある。前回のmail.localの例に続けて、これからもSendmailの直接実行など、さまざまな実験を誌面で紹介することになる。読者がこれらの実験を実際に行う場合は、自分が管理権限を持っていて、他

Column

システム管理の解説といたずら

ネームサーバにしろメールシステムにしろ、その構造や動作を細かく解説していくということは、システム管理に興味を持っている人にとってはかなり有意義なことなのだと思う。しかしこのような記事は、システムの弱点の解説でもある。それまで何も知らなかった人がいたずらメールを送ることもできてしまう（本当に悪質なクラッカーは、こんな記事の内容は常識なので、実害はないはずだ）。結構悩むところだ。

ひとつの解決策は、オライリーのシリーズのように、敷居を高くしてしまうというものだ。ちょろっと調べて何かしようというには、価格も高いしページも多い。もうひとつの解決策は、そんないたずらをされる前に、システムのセキュリティを高めるということだろう。

人に迷惑をかける恐れのないマシンを使おう。この原稿を読んだ担当編集者は、手元のマシンだけでは飽き足らず、社内のメールサーバで実験をして、システム管理者に怒られたうえに、社内ブラックリストに載せられてしまったそうだ。実験の数分後には管理者からメールが来たというから、ログの自動チェックにでも引っかかったのであろう。この種の実験では、通常の使用状況とは異なるパターンのログを生成することがあるからだ。

もうひとつ注意。記事中で使っているドメインは、基本的に架空のものである。ascii.co.jpは実在するアスキーのドメイン名であるが、その下位のサブドメインは架空のものである。また、ascii.co.jp以外のメールアドレスやサーバに使っているドメインも架空のものである。いたずらメールなどを送らないように。

■ucbMail

ucbMailは、メールの送受信環境のユーザーインターフェイスを改善したフロントエンドプログラムである。これにより、受信したメール（メールボックス）の一覧表示、個々のメールの表示、保存、削除、返信などをキャラクタベースの対話環境で行える。また、ローカルホスト宛以外のメールを送ることもできる。現在は多くのCUI／GUIベースのメールツールが多数普及しているので、今さらucbMailを使う必要もないのだが、もっとも基本的なツールなので使い方を覚えておいても損はない。また、メール環境のテストには手頃なツールである。使用可能なサブコマンドは、“？”コマンドで表示される。発信、返信、引用、ローカルなエイリアス定義（アドレス帳機能）などがあるので、ひととおりのメール処理はこなせる（**画面1**）。ただし、きちんと日本語対応していないし、添付ファイルなどの自動処理も行ってくれないので、常用ツールにするのは辛いだろう。

```
% mail test                                  ← 宛て先を指定
Subject: test mail                          ← Subjectを指定

test mail                                   ← メールの本文

.                                           ← 行頭のピリオドで終わり。Ctrl＋Dでもよい
EOT
% mail                                      ← 引数を指定しないで起動すると、メールを読むことができる
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/test": 2 messages 2 new          ← 最初に一覧が表示される
>N  1 masa                  Mon Sep  4 14:02   7/84    "test"
 N  2 test@takobeya.com     Mon Sep  4 14:40  12/357   "test mail"
& 1                                         ← 表示するメールを番号で指定する
Message 1:                                  ← mail.localを直接実行して送ったメール
From masa Mon Sep  4 14:02:30 2000
To: test
From: masa
Subject: test

test message

& 2                                         ← mailコマンドで送ったメールなので、ヘッダ類がちゃんとしている
Message 2:
From test Mon Sep  4 14:40:37 2000
Date: Mon, 4 Sep 2000 14:40:37 +0900 (JST)
From: Test account <test@takobeya.com>
To: test@takobeya.com
Subject: test mail

test mail

& d 1                                       ← 1のメールを削除
& q                                         ← 終了
%
```

画面1　ucbMailの実行例

■MUAとMTA

電子メールシステムを運用するためには、ユーザーが操作する各種メールプログラム、実際にメール配信に関与するデーモン（サーバ）プログラムなどが連携して動作する必要がある。binMailを使ったローカルホスト上の配信だけなら、このような形式にすることなく、プログラムが直接相手のメールボックスにメールを書き込むことができる。しかし、より複雑なメール配信を行うために、ユーザーインターフェイス側のモジュールと、配信側のモジュールに分割して考えるのである。一般に、このような構成とする場合、サーバ／クライアントモデルになるが、電子メールの場合、単純にメールアプリケーションをクライアント、Sendmailなどをサーバと切り分けるだけでは十分ではない。メールアプリケーションから呼び出されるメール配信プログラム、リモート配信するためのサーバ間の通信、ダウンロード専用のまた別のサーバなど、いろいろ関係して

くる。

電子メールシステムでは、ユーザーが操作する部分をMUA（Mail User Agent）、背後で動作し、実際の配信を行う部分をMTA（Mail Transfer Agent）という。つまり、ユーザーが直接使う部分がMUAであり、メールの配信を行う部分がMTAなのである。ucbMailや各種メールアプリケーションはMUAであり、Sendmailやqmailなど、一般にメールサーバと呼ばれるプログラムはMTAである。binMailプログラムは、直接使えばこの両方の機能を持つことになるが、通常はSendmailから呼び出されるサブプログラムとして使われるので、MTAということになる。この用途に特化したmail.localは、完全にMTA側のプログラムである。

MUAとMTAに分割することにより、各ユーザーは自分の好きなメールソフトウェアを使用することができ、また、そのホストやサイトにおけるメール配信を、管理者が一元管理できるようになる。つまり、ユーザーはMUAだけを管理、使用し、管理者はMTAの面倒だけを見ればいいということだ。

■メールアドレスの拡張

さて、そろそろリモートホスト上のユーザー宛のメールのことを考えてみよう。まず、メールアドレスを拡張する必要がある。ローカルホスト上のユーザーへのメール送信であれば、宛て先ユーザーのユーザー名だけ指定すれば十分である。しかし、リモートホスト宛となると、ユーザー名だけでは不十分だ。相手側のホストを指定しなければならない。インターネット電子メールの場合、宛て先ホストのドメイン名を組み合わせるというのがもっとも自然なやり方だろう。そこで、宛て先のユーザー名とホストのドメイン名を“@”でつないだ形式のアドレスを使用する。アスキーのmail-svrというホスト上にユーザーアカウントを持つユーザーにメールを送る場合、メールアドレスは次のようになる。

```
user@mail-svr.ascii.co.jp
```

宛て先ホストのドメイン名

ユーザー名

簡単な仕組みだ。DNSを使って宛て先ホストの名前解決ができれば、宛て先ホストにメールを送ることができるだろう。宛て先ホスト上で各ユーザーのメールボックスにこのメールを配信するのは、相手側のサーバの仕事だ。

しかし、実際の電子メールシステムとメールアドレスはこんなに単純ではない。実際、アスキー社員のメールアドレスは、上記の形式ではなく、「user@ascii.co.jp」という形式である。別に、ascii.co.jpという名前のメールサーバホストがあるわけではない。この仕組みについては、DNSの解説中で数カ月前にタネ明かし済みだが、今回の連載の中で、再度詳しく説明することになるだろう（DNSの解説を書いていた時は、Sendmailの話をするつもりはなかったのだ）。とりあえずここでは、メールアドレスはユーザー名とドメイン名を“@”でつないだものとしておこう（これ以外の形式のメールアドレスもあるのだが、現在ではほとんど使われておらず、今回の連載ではたぶん触れないと思う）。

■異なるホスト宛のメール送信

1台のホストコンピュータ内で閉じているメール配信の仕組みは前述のとおりである。今度は、異なるホストコンピュータ間でメールを配信する場合について考えてみよう。

binMailはマシン上のメールボックスに書き込むだけなので、ローカルホスト上へのメール配信しか行えないが、ネットワーク接続やモデム接続をサポ

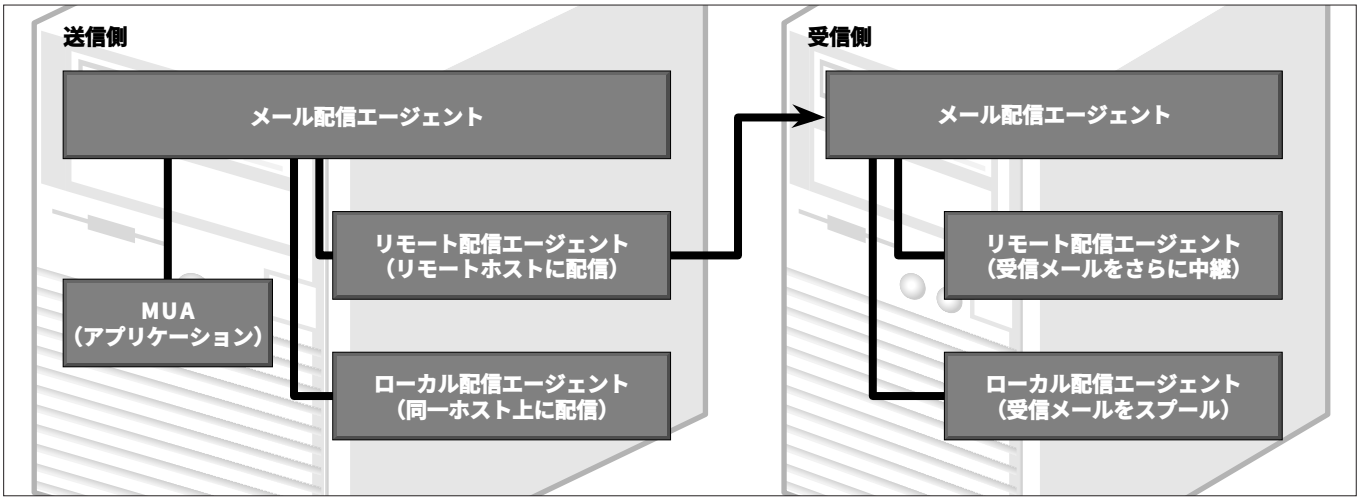


図2　ホスト間をまたがったメール配信

ートしたMTAを使うことで、メールを異なるホストに送信できるようになるのだ。

まずユーザーがMUAを使ってメールメッセージを作成し、これをリモート配信に対応したMTAに渡す。MTAは宛て先を調べて、ローカルホスト上への配信であればbinMailを呼び出し、スプールディレクトリ上のメールボックスにメールを配信する。宛て先が別のホストであることを示していれば（ユーザー名@ドメイン名という形式のアドレス）、何らかの方法で宛て先のホストを調べ、そして何らかの通信手段を使ってそのホストにメールを中継する。中継されたホスト上のMTAは、再びその宛て先を調べる。恐らくはそのホスト上のユーザーが宛て先として指定されているだろうから、MTAはbinMailを呼び出し、ユーザーのメールボックスにそのメールを追加することになるだろう（**図2**）。

このようにして異なるホストにもメールを送れるようにする場合、いくつかの問題を考えなければならない。

・宛て先の解析

指定された宛て先を解析し、どのように配信を行うかを判断しなければならない。これには、ローカルホスト宛／リモートホスト宛の判定や、複数の通信手段がある場合、どの方法を使って目的のホストと接続するかといった判断が含まれる。

・送り先の判定

ローカルホスト宛のメールでない場合、宛て先となるホストを何らかの方法で調べなければならない。

・エラー処理、システムの保護

宛て先の間違い、相手先ホストのダウンなどにより配信できない場合に、何らかのエラー処理を行う必要がある。また、悪意を持ったユーザーの使用に対して、何らかの保護策を講じなければならない。

実のところ、電子メールサービスの運用というのは、これらの問題をうまく解決できるように、Sendmailなどのプログラムを設定することにほかならない。そしてこれが簡単には済まない作業なのである。過去のシステムとの互換性を維持し、サイトごと、ホストごとのさまざまな詳細を自由に設定できるように、メールサーバは非常に自由度が高くなるように作られている。その分、設定も大変なのである。

■ヘッダ

さて、メールサーバの動作や設定を見る前に、まず電子メールそのものの構造、つまり、実際に配信されるメールの中身を見てみよう。

メールボックスにスプールされたメールには、メールの本文（ボディ）に先行してさまざまなヘッダが付加されている。ここには宛て先や発信者など、メール配信や返信のために必要な情報、サブジェクト（件名）のようなユーザーのための情報、サーバが付加する管理情報、MUAが付加する各種情報などが含まれている。これらはすべてテキスト形式なので、意味がわかれば内容も理解できるだろう。もっとも、多くのMUAアプリケーションは、一般ユーザーが必要としないヘッダ情報を表示しないようにしている。たいていはこれらを表示するオプションが用意されているが、システムによってはこれらの情報を削除してしまうものもある。メールシステムが問題なく動作している時はそれでも構わないのだが、エラーの発生時や、性悪ユーザーやサイトからのメールを分析する際には必須の情報なので、削除されると困って

Column

なぜカーボンコピーなのか？

Cc:、Bcc:はカーボンコピー（Carbon Copy）という意味だが、そもそもカーボンコピーとは何か？　カーボンコピーとは、カーボン紙を使った複写のことである。

ボールペンなどで圧力をかけて書くと、下の用紙にも文字が写る複写式の帳票は今でも広く使われている。現在は、ノーカーボン紙という圧力で色が変わる用紙が多いが、以前は圧力で転写されるインクを塗布したカーボン紙を間に挟んで複写を行っていた（あるいは上側の用紙の裏面に転写インクを塗布していた）。

余談だが、複写式帳票でカーボンコピーを作成するためには、用紙に圧力をかけて印刷しなければならない。現在でも騒がしい音をたてるドットインパクト式のプリンタが細々と生き残っているのは、この種の帳票印刷に不可欠だからである。

タイプライタで手紙などをタイプする場合も、2枚の用紙の間にカーボン紙を挟んでタイプすることで、コピーを作成することができる。このようにして作成されたコピーのことをカーボンコピーと呼ぶ。カーボンコピーは多少印字品質が低下するので、手紙の宛て先にはタイプライタで直接印刷された一番表の用紙を送り、カーボンコピーは控えや内輪への配信に使う。

電子メールの場合は、別にCc:で送っても品質が低下することはないが、この伝統に従っておくのが無難である。つまり、本来の宛て先はTo:で指定し、副次的な宛て先はCc:やBcc:を使うということである。

しまう。また、MTAのレベルでこれらの情報を削除してしまうシステムもある（方式の異なるメールシステム用のゲートウェイなど）。

代表的なヘッダ項目を簡単に紹介しよう。ここではMUAが使うフォーマット情報関連のヘッダは取り上げていない。また重要なヘッダについては、解説を進めていく際に、さらに詳しく解説していく。

・From:

メールの発信者のアドレス、本名である。binMailがスプール時に付加するFrom行（コロンが付かないもの）とは別のものである。このヘッダは、メールを作成したMUAか、最初に中継を行うMTAが付加する。

・To:

メールを送る宛て先である。ここに記載したアドレスにメールが送られる。複数の相手に送る場合は、カンマで区切って複数のアドレスを列挙することができる。このヘッダは、MUAでメールを作成する時に、ユーザーが指定する。

・Cc:

カーボンコピーという意味で、To:の宛て先のほか、ここで指定したアドレスにもメールが送られる。主要な宛て先以外に、メールのコピーを送りたい時に指定する。たとえば自分のアドレスを書いて控えとしたり、取引先にTo:で送り、自分自身や同僚などにCc:で送るといった形で使う。このヘッダは、MUAでメールを作成する時に、ユーザーが指定する。

・Bcc:

ブラインドカーボンコピーである。Cc:と同様にカーボンコピーを送る先を指定するが、Cc:の内容は受信側も認識できる（相手に送られたメールのヘッダ中にCc:が含まれている）のに対して、Bcc:は相手側に認識できない（届いたメール中にはBcc:ヘッダは存在しない）。これにより、相手に知られることなく、同じメールを別のユーザーに送ることができる。このヘッダは、MUAでメールを作成する時に、ユーザーが指定する。

・Reply-To:

このメールに対して返信を送る際の返信アドレスを明示的に指定する。これを指定していない場合、From:で指定されたアドレスに返信されるが、Reply-To:により、指定したアドレスに返信されるようになる。

通常（From:アドレスに返信すればいい場合）は指定する必要はないが、メーリングリストなど、返信を別のアドレスで受けたいといった場合に指定できる。

・Errors-To:

エラー発生時の通知メールを送る先を指定する。通常、エラー通知はFrom:で指定されたアドレスに送られるが、メーリングリストなど、From:やReply-To:に返されたくない場合にErrors-To:を使用する。

・Subject:

メールのタイトルで、ユーザーがMUA上で付加する。件名などと表記されているメールアプリケーションもある。メールに返信すると、ここで指定された文字列の先頭に、「Re: 」という文字が付加されるが、これはMUA側の機能である。

・Date:

メールを発信した日時である。MUAかMTAが自動的に付加する。

・Received:

メールがMTAで中継されるごとに、その中継記録がReceived:というヘッダの形で記録される。このヘッダは、常にメールの先頭に付加される。

・Message-ID:

電子メールを一意に識別するIDを表す。通常、この表記形式は、メールサーバのドメイン名（これは世界中で一意に決まる）と、サーバホスト上で一意なコードを組み合わせて生成する。

Column

Bcc:の用法

Bcc:で別の宛て先を指定することにより、To:やCc:で送った相手に知られることなく、メールのコピーを別のユーザーに送ることができる。さらに応用的な使い方として、To:やCc:に宛て先を指定せず（あるいはTo:に自分自身を指定して）、Bcc:だけで宛て先に配信するというテクニックがある。このようにしてメールを送ると、受け取った相手は、そのメールが誰から来たかはわかるが、同じメールがほかの誰に送られたかがまったくわからなくなる。

これは、お互いに関係のない多くの人に同じメールを送るという場合に便利なテクニックだ。To:やCc:で列挙して送ると、関係のない人に自分のメールアドレスが知られてしまい、不愉快に感じる人もいるかもしれないが、Bcc:を使えばこのようなことがなくなる。新製品情報の案内など、メーリングリストを作成するほどではないものの、お互いに関係のない不特定多数のユーザーにメールを送る場合などに便利である。

ユーザーレベルではメッセージIDはまったく必要ないが、MTAは、配信ログにこのメッセージIDを記録する。

■エンベロープ

さて、ヘッダの紹介を見たあとで、前に述べたbinMail（mail.local）の動作を思い出してほしい。メールの中には、From:、To:といった発信アドレス、受信アドレスが記述されている。にも関らず、mail.localを実行する時には、引数で宛て先を指定していた。また、mail.localがスプール時に付加するFrom行（コロンの付かないほう）は、メール内のヘッダから得られた情報ではない。実際に確かめてみよう（**画面2**）。

mail.localは、メール内のTo:ヘッダの内容を見ることなく、引数で指定された宛て先にメールを送る。そしてFrom:ヘッダではなく、どのユーザーがmail.localを実行したかに基づいてFrom行を生成している。

メールの配信には、このように2系統の発信／宛て先情報が使われている。ユーザーが指定するのはメール内に記述されたTo:ヘッダだけだが、MTAが実際に配信を行う時には、これとは異なる情報を使う場合がある。実際の配信に使われる配信情報（この例でいえば、mail.localが使っている発信者、受信者の情報）のことを、エンベロープ（封筒）という。

エンベロープの情報は、もともとはFrom:、To:に基づいたものである。そしてMUA上で指定されたFrom:、To:に基づいてエンベロープ情報を作成するのはMTAの仕事である。

通常のメール配信では、メール中のヘッダの内容とエンベロープ情報は同じものになるが、ある種の状況ではヘッダの内容とは異なるエンベロープ情報が使われる。よくあるパターンとして、プログラムからメールを送る場合がある。何らかのプログラムがメールを送信した場合、送信者のエンベロープ情報は、そのプログラムのプロセスを実行しているユーザー情報に基づいたものになる。通常のユーザープロセスであれば、実在のユーザーがメールを送ったことになるが、デーモンプログラムなどの場合は、デーモン用の特殊なユーザーアカウントになってしまう。一般にこのようなデーモンプログラムは、そのユーザーアカウントをアドレスとして指定されたメールを受信することができない。そのため、メール中のFrom:ヘッダには、エンベロープ情報とは異なる実際に受信可能なアドレスを記述しておくことになる（あるいは、そのようなアドレスを受信できるようにエイリアス（後述予定）を登録しておかなければならない）。

日常的なところでは、メーリングリスト（ML）サーバから配信されたメールがこのような形式で送られてくる。エンベロープ情報の送信者はMLサーバであるが、メール中のFrom:ヘッダは、そのメールをMLサーバに送った個人ユーザーのアドレスになっているのが普通だ（メーリングリストの場合、さらにReply-To:でMLサーバを指定し、Errors-To:でMLの管理者のアドレスを指定するようになっていることが多い）。

メール配信は、メール中に記されたFrom:とTo:、エンベロープ情報がいろいろ組み合わされて行われるということを覚えておこう。

■さて、次回は

停電の話が長くなってしまい、肝心のメールの話はあまり進まなかった。ここまでは、binMailを使ったローカル配信、受信メールのスプール、メールアドレス、電子メールの内容といったことを説明してきた。

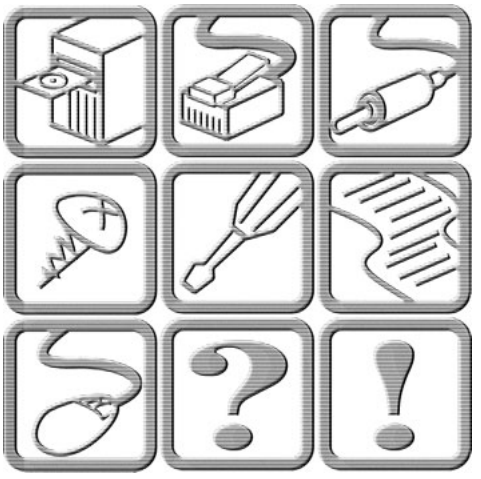
これらの要素を組み合わせれば、ローカルホスト上のメール配信はすべてできてしまう。binMailを使ったローカルメール配信、ucbMailを使った受信メールの閲覧が可能だ。

次回はいよいよSendmailがメール配信にどのように関与しているかを説明していく予定だ。

```
% whoami
test                                    ユーザーtestで実行している
% /usr/libexec/mail.local test                          宛先にtestを指定
To: you                                 To:フィールドにはyouを指定
From: me                                          From:フィールドにはmeを指定
Subject: test #2

test mail
% more /var/mail/test                  ユーザーtestのメールボックス
From test Mon Sep  4 16:22:20 2000                ユーザーtestからのメール
To: you                                 To:フィールドはyou
From: me                                          From:フィールドはme
Subject: test #2

test mail

%
```

画面2　mail.localの実行例　その2

Try & [trái] Try



今さらだけどAthlonの話

文:政久忠由
Text : Tadayoshi Masahisa

前回、ハードディスクを新調した僕のLinuxマシンは、移行にともなうトラブルも何ひとつなく、何事もなかったように起動し順調に稼動を始めていた。しかし、あまりに埃まみれになっていて、電源やCPUのファンもへたり気味の嫌な音を発生していたので、これを機にオーバーホールを行うことにした。

Destruction

オーバーホールといっても、エアスプレーとクリーナーで長年の埃を落とし、ファンモーターに潤滑油を補給するだけの簡単な作業だ。ものの数分で終わる予定だった。電源ユニットを取り外し、分解してクリーニングするのに多少時間はかかったものの、20分程度で再アセンブルは完了した。長年の経験から、ショートしている部分はないか再三確認して電源を投入した。一時的ではあるが、電源／CPUファンも活力を取り戻し、ノイズも低下していた。いつもどおりにLinuxは起動し何ら問題はなかった。

一般にPCのノイズの発生源といえば、ハードディスクと各種ファンだが、最近のハードディスクは比較的静かになってきている（その昔7200rpmなどの高回転のハードディスクは、ジェット機のエンジンのような甲高い金属音が強く生じていて、むしずが走るような冗談じゃなく体に悪影響がある感じのノイズで、ヘッド移動時の音もひどかったが）。

一方、高級な製品は別にして、相変わらず耳障りの悪い風切り音を発生しているのが、ファン、特に電源ユニットのファンだ。このノイズの原因は、ファン本体とその風の出入り口の構造にあるわけで、高級な電源ユニットでは、その対策が施されているのだが、僕が購入するクラスのユニットは、ファン自体の動作音もうるさく、取り付け先も金属板をパンチアウトしただけのしろものなので、一向にノイズは軽減されない。一時的に電源ユニットのファンを止めて運用してみたら本当に静かだったのだが、その部分の発熱を放置するわけにもいかないので、ファンを復活せざるを得なかった（自然対流、ファンレスにあこがれるよなぁ、ほんと）。そのうち、ファンの取り付け先の網を切り取りたいとは、思っているのだけれども…。

話を戻すが、オーバーホール後、しばらくしてスピーカーを接続し忘れていることに気付いた。このLinuxマシンには、サウンドはそれほど重要ではなかったが、キャプチャカードの音をモニタするためにとりあえず接続する必要があった。

このLinuxマシンは、稼動中にディスプレイケーブルを抜き挿ししたために（通電中は火花が飛んじゃうんだなぁ、これが）、過去に2枚のグラフィックスカードのRAMDACを駄目にしたことがあるいわく付きのマシンで、ケーブルを抜き差しする際には細心の注意を払っていた。しかし、このときはスピーカーケーブルということもあって、高をくくっていた。

システムで音を出力させておきながら、スピーカーアウトはどのジャックだったかなと、適当に接続して確かめていたのだが、一向に音が鳴らない。そこでミキサーかなと思い、操作しようと思ったら、画面はフリーズし、キーボードの入力も、ネットワークの反応もない状態、完全にすべてが停止していた。まったくの不覚であった。血の気が引くのを感じながら、リセットスイッチを押してみるもマシンの初期化自体が開始されない。電源を落として、再度投入しても事態に変化は見られない。

以前RAMDACを駄目にしたときは、単に画面に表示されないだけで、システム自体は稼動していた。となると今回の症状からみて、マザーボードのチップセットに不用意な過電流がフィードバックして逝ってしまったに違いない、というわけで、拡張カードやメモリ、CPUを別のシステムで検証してみたが、案の定問題はない。何度かトライしてみたが、結局、このマシンが息を吹き返すことはなかった。やはりマザーボードがいかれたようだ。

直接の原因が何なのかはよくわからないが、ケースは閉めていたし、拡張カードを固定し忘れていたわけでもない。スピーカージャックを挿そうと思ってガチャガチャしただけなのに…。よくよく考えてみるとTVアンテナケーブルを接続してから漏電のような感じが生じていた気もする。まぁ、原因はともかく、僕自身の不注意によるものだし、ケース／電源ユニットもCPUがi486DX2-66のころに購入したAT用、マザーボードもSocket7で200MHzまで対応のものなので、それほどの痛手ではなかった（十分利用したと思うしね）。

ただ、今さらAT用のマザーボードを探して購入するのはとんでもなく無駄であるし、また手持ちのこまもガラクタに近い古びたものばかりであるため、この際、すべてを一新することにした。買い換えの時期を決めあぐんでいた僕にとって、いい契機となったわけだ。

Restructure

さて今回、手持ちのシステムを一新するにあたって、まずは不要なものを徹底的にすべて処分することにした。未練は禁物ということで、72ピンSIMMを使用しているマシンと使いそうにないPCIカード、あとISAカード、そしてSCSIデバイスに完全に見切りをつけた（ハードディスクの容量が8Gバイト程度までは、SCSI至上主義を貫いてきたのだけれど、SCSI陣営には申し訳ないがもう限界）。

結果整理してみると、マシンは3台分、全部で50程度の点数になった。だが、まともな値で引き取ってもらえそうなのは、メモリとSCSI製品、ハードディスクぐらいで、あとは本当にガラクタ同然の燃えないゴミや粗大ゴミで処分するよりはタダで引き取ってもらうだけラッキーな拡張カードたちであった。

とりあえず、いくつかのジャンク屋のWebで買い取り価格をチェックしてみたところ、意外と72ピンSIMMは高値（128Mバイト分を売って、1000円強プラスすればPC100の同容量が入手できる）であったし、SCSI製品は世の中には求めている人が多いらしく、これまた思った以上にいい値であった。6万円程度にはなりそうだ。

結局、秋葉原まで両手いっぱいの重い荷物を引きずって僕が手にしたのは、約6万5000円である。これらを購入したときの合計金額を計算するのは、悲しくなるのでやめておくが、でもまぁ、PCの製品サイクルや現在の製品価格を考えると結構いい値段のような気もする。

今回処分した3台のマシンは、それぞれ欠くことのできない重要な役割を担っていたこともあり、新規購入するマシンも3台必要だったのだけれど、同時期に似たようなシステムを複数購入するのは面白くないから、1台は先送りしてとりあえず2台だけ構築することにした。今の僕にはこだわりというものはなく、そこそこ動いてくれれば十分なので、CPUは最低ランクの1万円前後、あとはそれぞれのマザーボードとメモリ（128Mバイト）を入手するだけだ。

で、結局購入したのは、Celeronの667MHzとi815Eチップセットと、Athlonの700MHzとVIA KT133チップセットの組み合わせだ（もちろん両者ともソケットタイプ）。Celeronは633MHzがちょうど売り切れ、Duronの700MHzも同じく売り切れだったので、仕方なく上記を購入する羽目になってしまった。最終的に今回はケースも必要だったので合計9万円弱にもなってしまったが、某ポイントカードの併用で、追加の資金は投入しなくて済んだ。ひとまず計画どおりである。

これら以外のデバイスは手持ちのものを流用したり、共有したりすることでなんとかなった。複数台のマシンを保有しているとフロッピードライブやCD-ROMドライブをそれぞれに装着するのって、実に馬鹿らしく思えるんだよね。フロッピードライブはBIOSアップグレードの時だけだし、CD-ROMドライブはインストールの時だけしか使わないんだから、必要なのは1セットだけで、あとはいらない。ケースを開けてケーブルを接続しないといけない点は面倒だけど、余程のことがない限り必要ないからねぇ。実際、半年に1回あるかないかだよ（ネットワークは不可

欠だけどね）。

今回、PentiumIIIではなくCeleronというのは、予定していたものなんだけど、AthlonとDuronはどちらにしようかものすごく悩んだ。なぜなら両者の違いが今ひとつピンとこなかったからである（本当に2次キャッシュメモリのサイズの違いだけ？）。700MHzのAthlonとDuronって2000円も違わなかったと思う。結果としてAthlonの700MHzを購入したわけだが、これは本当に得だったのかどうか、今でもよくわからない。まぁどっちにしても、IntelのCPUより安いからいいのだけれど。

本当は、もっと早くAthlonを手に入れたかったのだが、Slot形状には生理的に拒否反応を覚え、価格も高かったので、僕には手が出せなかった。Socket形状が登場して、安価になった今、型落ち（Athlonは今800MHz以上しか生産していないハズ、Duronも800～700MHzだけだったと思う）とはいえ、Athlonを手にすることができた。かなりうれしい。

今さらAthlonのアーキテクチャを事細かに説明しても意味がないので省略するが、とにかくL1キャッシュの命令とデータにそれぞれ64Kバイトというのが太っ腹だし、同時実行できるロジックユニット数やクロックテクノロジーなどを見ても、追撃メーカーらしく過去にとらわれ過ぎず斬新かつ新鮮で躍動感がある。一方のIntelは、悪い意味の長老体質というか、それなりでしかないんだよなぁ。巨大タンカーだからある程度は致し方ない面もあるのだが。もうすぐPentium4が出るけど、どんなもんでしょ。まぁそうはいっても、僕が個人的にそれらを入手するのは、価格的に枯れたころでしかないのだけれども。それはそうと、今CPUといえば、Crusoeも注目されているんだっけ。チャームポイントは、消費電力と何だっけなぁ？ 単に難破して無人島に漂着しただけにならないといいけど…サバイバルは大変だからねぇ。だって、そのマーケットじゃぁ、高い価格付けられないでしょ。

LinuxはAthlonで？

今回購入したマシンは、それぞれLinuxとWindows 2000ドメインのテストマシンになるのだが、どちらをLinux用にするかは、最初から考える必要はなかった。もちろんAthlonである。理由は明白、遊べそうだったからだ。僕にとってCPUの処理能力はあまり関係ない。単純にAthlonで遊んでみたかった。ただそれだけなのだ。

最終的に手持ちのマシン間で部品のシャッフルを行い、PentiumIIIとi815EマザーボードがWindows 2000のプライマリマシン、Celeronと手持ちのi810Eがテスト用のドメインコントローラになり、ファイルサーバと120日間限定のExchange ServerとISA、SQLのサーバとして稼動を始めた。

Linuxマシンのほうは、押し入れにあったRiva128とVT86C100Aを装着して稼動を始めることにした。ハードディスクの内容は、お亡くなりになった以前のマシン用設定のカーネルイメージのままだったので、ちょっと不安もあったが、何のことはない問題なく初期化されinitも順調に処理された。まぁブートデバイスだけの問題だったのだが、KT133チップセットのVT82C686はLinuxのジェネリックなコードで問題なく動作するようになっている。

とりあえず、動いてくれればこっちのもの。これからシステムにいくつかの最適化を施していくことにする。

Athlon用のカーネルとコンパイラ

まずはカーネルの再構築といきたいところだが、カーネルを作るにはコンパイラが重要ということで、少しコンパイラについて話しておこう。

■ Athlon用の最適化コード

コンパイラの役割は、ソースコードをCPUが実際に処理できるオブジェクトコードの状態に翻訳することだが、一般にユーザーが利用するリリース実行コードの場合、その過程で処理速度を向上するためにオプティマイズ（最適化）を行うようになっている。利用するユーザーにとっては、プログラムの実行過程はどうでもよくて、結果さえ合っていればいいので、処理の時間や使用するメモリサイズを短縮／縮小するために、オプティマイズによって過程を省略したり、別の手法に変換したりするわけだ。たとえば、1から100までを足す計算を行うソースコードは、コンパイル時のオプティマイズの段階ですでに結果がオブジェクトコードに埋め込まれたり、無駄なループ処理は省略されたりする。

gccの場合、最適化オプションは-fname（-fから始まるものすべてが最適化オプションではないが）が該当する。一般に最適化オプションとして利用される-O、-O2、-O3などは、適切な最適化オプションをまとめたもので、個々の最適化手段を指定するわずらわしさを軽減するために用意されている。

加えて、gccには機種／CPU依存の最適化オプション-

mが用意されている。これは機種やCPUのレジスタ構成、処理性能の特徴をコスト化したもので、ソースコードをアセンブラコードに置き換える際のレジスタの使い方、命令の選択などに影響を与える。要するに、このオプションによって新しいCPUに追加実装された命令セットを利用できるようになり、同じ処理で複数の解決手法（命令の組み合わせ）がある場合は、より効率的な方法やパイプラインを乱さないようなコード生成を行えるようになる。ただし、これによって他のCPUと互換性のない命令を使用したプログラムが生成されることがあるので、注意が必要だ。

gccでは、/usr/lib/gcc-lib/i686-pc-linux-gnu/2.97/specs（ディレクトリはバージョンによって異なる）にある程度の設定が格納されている。上記の該当する部分（抜粋）は、次のようなものだ。

```
*cpp_cpu_default:
-D__tune_i686__ -D__tune_pentiumpro__
*cpp_cpu:
%{march=athlon:-D__athlon -D__athlon__ %{!mcpu*:-
D__tune_athlon__          }}%-D__tune_pentiumpro__
}%{mcpu=athlon:-D__tune_athlon__ }
```

これはgccより最適化に強いegcs-20001016のものなんだけど、-march=athlonオプションで-D__athlon -D__athlon__、-mcpu=athlonオプションで-D__tune_athlon__がプリプロセッサの段階で付加されて処理されるということと、プリプロセッサのデフォルトはPentiumPro（II／III）用の定義になっていることがわかる。もし常時Athlon用のコード生成にしたければ、＊cpp_cpu_default:で指定しておく定義を-D__tune_athlon__にしておけばよい。

このオプションが実際にどの程度効果があるのか、紹介しておく（あくまでひとつの例、絶対視するなよ）。

```
                  sign    verify    sign/s verify/s
compiler: gcc -O3 -m486
rsa  512 bits   0.0027s   0.0002s    372.4   4011.9
rsa 1024 bits   0.0156s   0.0008s     64.2   1221.2
rsa 2048 bits   0.0986s   0.0027s     10.1    364.4
rsa 4096 bits   0.6500s   0.0094s      1.5    106.2
compiler: gcc -O3 -march=athlon -mcpu=athlon
rsa  512 bits   0.0023s   0.0002s    440.2   4450.5
rsa 1024 bits   0.0134s   0.0007s     74.8   1371.8
rsa 2048 bits   0.0870s   0.0025s     11.5    403.0
rsa 4096 bits   0.5861s   0.0086s      1.7    115.9
compiler: gcc -O3 -DSHA1_ASM -DMD5_ASM -DRMD160_ASM
rsa  512 bits   0.0015s   0.0001s    675.3   7353.0
rsa 1024 bits   0.0074s   0.0004s    134.7   2584.4
rsa 2048 bits   0.0441s   0.0013s     22.7    771.6
rsa 4096 bits   0.2968s   0.0046s      3.4    218.2
```

これは一部の人には懐かしのopensslの性能テストで、それぞれ486オプション、Athlonオプション、最後はアセンブラソースを有効にした状態でコンパイルしたコードでの結果だ。Athlonオプションにより、486オプションと比べて10%以上性能が向上していることがわかる。ただ、アセンブラコードには、程遠く及ばないのだけれど。

というわけで、問題のない範囲でこれらのオプションで遊んでみるといいだろう。-Sオプションでソースをアセンブラコードで出力できるので、ひまな時に、いろいろオプションを変えて、実際にその違いを見てみても面白いと思う。でもまぁ最適化のし過ぎは、トラブルことも多いので、実際によく利用するプログラムなどは、ほどほどにね。

■カーネルのAthlon用設定

いよいよカーネルの作成なんだけど、Athlon用に変更することとしては、プロセッサタイプの選択でAthlon/K7を選択する、これだけなんだよね。ちなみにカーネルはLinuxカーネル2.4.0-test9を前提にしている。

```
Processor type and features  --->
(Athlon/K7) Processor family
```

あとはそれぞれの環境依存だから、ここで深く説明しても意味がないのだけれど、僕の環境では、/proc/pciをcatして接続されているPCIデバイスを確認して、ATAデバイスでVIA82CXXX chipset support、USBでUHCI、サウンドでVIA 82C686 Audio Codecを有効にしたほか、Character devicesでAGPやDirect Rendering Managerを設定し直し、ネットワークカードの設定をしたぐらいかな。

そこでmakeを行う前に、少しだけプロセッサタイプでAthlonを選択すると、どのような定義が行われるのかチェックしておこう。設定ファイル（.config）は次のようになっている。

```
CONFIG_MK7=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_CMPXCHG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_X86_L1_CACHE_BYTES=64
```

```
CONFIG_X86_TSC=y
CONFIG_X86_GOOD_APIC=y
CONFIG_X86_USE_3DNOW=y
CONFIG_X86_PGE=y
CONFIG_X86_USE_PPRO_CHECKSUM=y
```

これらの中でAthlon特有といえるのが、CONFIG_X86_L1_CACHE_BYTES=64とCONFIG_X86_USE_3DNOW=yだ。

まず前者は、L1キャッシュのラインサイズを表している。Athlonの場合、64バイト、つまり512ビットのラインサイズである（命令、データそれぞれ64Kバイトだから1024本のラインがある）。ちなみにi386／486は16バイト、IntelのPentium以降は32バイト（PentiumIIIも同様）で、このラインサイズは、メインメモリとのアクセスに影響する（実際にはL2キャッシュが中間に介在しているから少し複雑なんだけど）。実はメインメモリへのアクセスはこのラインサイズを基準に行われるようになっていて、通常一度の処理でこのラインを埋めるだけのデータ転送が行われる。つまりメモリバスが64ビットの場合、PentiumIIIだと4回、Athlonだと8回の転送でキャッシュラインを埋めるためのひとつのサイクルが完了する（メモリアクセスのタイミングで5-2-2-2といった表記はこれを意味している）。

次に後者は、3D Now!の実装を表しているが、実際には、3D Now!の拡張命令を利用したいくつかの処理の有効化を意味する（これは後ほど説明）。

ちなみにPentiumIIIでは、定義はCONFIG_M686FXSR=yとなり、PentiumIIと比べてCONFIG_X86_FXSRとCONFIG_X86_XMMが追加有効になる。前者は、SSEの実装で追加されたフローティングポイントユニットの高速ロード／ストア命令を有効にするもので、後者はMMXコードを有効にするものだが、今のところ後者が利用されているのは、ソフトウェアRAID5のチェックサム生成の部分のみである。

■CONFIG_X86_USE_3DNOWの効果

では、CONFIG_X86_USE_3DNOWの効果を見てみることにしよう。この定義により有効になる関数は3つある。

```
_mmx_memcpy()
mmx_copy_page()
mmx_clear_page()
```

それぞれ何を処理するものであるかは、名前から察しがつくと思うが、最初の関数は、メモリ空間におけるデータのコピー（文字列処理にも影響する）、次はページメモリのコピー、最後はページメモリのゼロクリアを行うためのものだ。

3D Now!を応用することで、フローティングポイントユニットを複数の整数レジスタのように扱えるので、まず_mmx_memcpy()では、4つの64ビット長のFPレジスタを活用して、メモリ内でのデータのコピーを行っている。実際には4つのレジスタ（32バイト分）のそれぞれに対する出し入れの2サイクルセット（64バイト分）のループ処理なんだけど、パイプライン構造とクロック当たり4FPの操作が可能なので、操作データが多ければ、かなりの効果が期待できる（L1キャッシュのラインサイズが64バイトなのでこの単位でループ処理を行っているとも考えられるが、演算ユニットの構成上、16個のオペレーションのループに意味があるのかもしれない）。ただ、いずれにしても整数演算ユニットよりも実行するまでの手続きに比較的時間が必要なので、操作データサイズが512バイト以上でないとこの関数は利用されないようになっている（それ以下のサイズは通常の整数演算ユニットでの処理）。

Athlonの3D Now!の応用で、ピーク性能では通常の整数演算のmovslの数倍くらいは速くなると思うが、きちんとテストしたわけではないので、責任は持てない（ん〜、PentiumIIIのSSEのMMレジスタは128ビットが8本だったよな。AthlonのMMレジスタって64ビットが4本だけだっけ？ Socketタイプになったときに変更なかったかな？ レジスタ名をリネームしてどんどんやるんだったかなぁ？ これはそのうちきちんと調べることにします、悪しからず）。

次にmmx_copy_page()では、操作対象単位がページサイズ（4096バイト）である点以外は、_mmx_memcpy()と同じ処理、mmx_clear_page()では、0で埋めた64ビット長のmm0レジスタを使って、目的のページメモリをクリアしている。こちらは128バイト分（16回のmovq）を1セットにしてのループ処理だ。なお、どちらの関数もフローティングポイントユニットの利用状況によっては、通常の処理を行うようになっている。

これらが実際にどの程度効果的なのかを計測するために一応テストプログラムを作ってみたものの、今のところ、これという有意義な安定した結果が導き出せていない（要は速くないわけ。少なくとも上記の処理に限っては通常の方法のほうが速いんだなあ、これが。ありゃ）ので、またの機会にしたいと思うのだが、とりあえずカウンタを設置して4時間程度使用したシステムでこれらの関数がどの程

度利用されたかを報告しておく。

```
_mmx_memcpy()        13030回
mmx_copy_page()      100439回
mmx_clear_page()     153310回
```

copy_pageとclear_pageは、ページメモリ処理ということで、プログラムの実行／終了を始め、直接、間接を問わず、すべての処理の裏側で使用されるため、10万回を超える呼び出しが行われている。一方memcpyの回数はやけに少ない。通常ユーザーが使用するプログラムでは文字列処理が基本だからもっと多くても…、512バイトに満たない処理がほとんどだから？　いや違う。実はこの関数は、カーネル内部で利用されているだけだからだ。これはシステムコールでもないので、ユーザープログラムから利用することはできない。ユーザープログラムが利用しているのは、ランタイムライブラリ（glibcとか）のmemcpyならびに文字列処理関数であって、カーネル内部のmemcpy関数ではないのである。もしこれを利用したければ、カーネルモードで動作するカーネルスレッドなどのプログラムを書かなければならない。

glibc（ランタイムライブラリ）の文字列処理関数では、i386、486、586、686それぞれの最適化されたアセンブラコードが用意されているが、Athlon用とか、3D Now!用とか、SSE用などのルーチンは今のところ実装されていない（参考にしたのは、glibc-2.1.95）。

だから、Athlonの3D Now!を活用したLinuxカーネルのコードでユーザープログラムの文字列処理が直接高速化されることは残念ながらない。しかし、文字列処理の背景にはメモリ操作があり、間接的にそれにまつわるページメモリ操作の恩恵を授かることができるので、速くなったように感じるとは思う。

まぁ常識的に考えて、ユーザースペースのメモリ／文字列操作をカーネルモードで処理しちゃ駄目でしょ。でもその痕跡がカーネルソースにあったような気もするなぁ。とりあえず、必要ならランタイムライブラリに実装されるのを待つか、自分でどうにかしませう。

嗚呼、カーネル作成

カーネルをコンパイルする際の最適化オプションなどのフラグは、/usr/src/linux/Makefileと/usr/src/linux/arch/i386/Makefileに定義されている。

```
＃Makefile：
CFLAGS := $(CPPFLAGS) -Wall -Wstrict-prototypes -O2
-fomit-frame-pointer
（追加オプション）-fno-strict-aliasing
＃arch/i386/Makefile：
ifdef CONFIG_MK7
CFLAGS += $(shell if $(CC) -march=athlon -S -o
/dev/null -xc /dev/null >/dev/null 2>&1; then echo
"-march=athlon"; else  if $(CC) -march=i686 -S -o
/dev/null -xc /dev/null >/dev/null 2>&1; then echo
"-march=i686 -malign-functions=4";  fi fi)
endif
```

コンパイラのバージョンによって、サポートしていないものもあるので、-march=athlonなどは、動作テスト後、問題なければ設定されるようなっている。

ではメイクといきたいところだが、重大な問題がここにはある。カーネルのChangesドキュメントにもあるように推奨はgcc 2.7.2.3で、egcs 1.1.2は問題なさそうだが、gcc 2.95には問題があることが知られているなどと記述されている。実際、僕の環境でも、egcs-20001016とカーネル2.4.0-test9の組み合わせを含め、ここ最近のegcsの開発バージョンでは、生成されたカーネルは起動初期化の段階でパニックダンプを吐き、動作しない（最適化オプションを多少変更すると動作するかもしれない、と思いつつ、面倒なのでまだやっていない）。

そこで仕方なく僕は、カーネルのコンパイルの際には、gcc-2.95.2を使っている（今のところ僕の環境ではこのバージョンで問題は出ていない）。でもこのバージョンには、K6はあるもののAthlonの定義がない。そんなわけで、-marchや-mcpuにpentiumproを指定する以外の選択肢がない。トホホ。

さらに新しい環境でACPIを有効にしたカーネルは、ACPIルーチンの初期化後、ハードディスクの割り込みリソースを見失ってしまい、そのエラーメッセージが延々に繰り返され、それ以降の初期化が実行できない始末。ACPIの機能で、IDE I／Fデバイスのリソースが再配置されてしまった？と思い、BIOSの設定をあれこれ変えてみても事態は一向に改善されないので、パワーマネージメントをAPMに変更して何とか動くようにしたけど、まったく困ったものだ。まずはBIOSをアップグレードする必要がありそうだ。

さて以前のCPUがPentium 200MHz程度の性能だったので、新しい環境は何をやっても気持ちがいいなぁ〜と思えたのもつかの間、数時間で体が慣れてしまった。今回はいろいろと課題を抱えてしまっただけのような気がする…

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8回
【マルチタスク】（まるち・たすく）
【マルチスレッド】（まるち・すれっど）
【デーモン】（でーもん）
【プロセス】（ぷろせす）
【ゾンビ】（ぞんび）
【fork】（ふぉーく）

マルチタスク
【まるち・たすく】

これぢゃない…

たくさんの仕事を抱えアップアップしているさま。“タスク”という音が英語の“task”を連想させるためか、多くの入門書で誤解をまねく解説が見られるが、本来は日本語の“まるちたすき”から。かつてのおさんどんが催事などで飯炊きに忙しくなると、「なんの」と気力をふりしぼるためにたくさんの（“まるち”は東北地方の方言で“多いさま”）たすきをかけてがんばった風習に由来している。

転じて、Linuxのような OSがもつ“（ムリして）たくさんのプログラムを同時に走らせる”しくみ。どだいムリをさせるしくみなので、登場以来、コンピュータ文化にはロクな影響を与えてこなかった。アクセスカウンタが0に戻ったり、掲示板の過去ログが消えたり、プレステ2用RPGの予約サイトがダウンしたりするのは、すべてこいつのせいである。最近ではさらに病膏肓に入り、人間にまでマルチタスク機能を要求しようという動きが急である。やれ原稿を締切までに出せだの出せないだの、ラフの提出がまだだの、風呂を洗えバターを買ってこい食器洗い機を買えとかまびすしいのは、すべてマルチタスク主義者の陰謀に違いない。現在、社会の人間性回復のためにも、シングルタスクなリアルタイムOSを求めるニーズが高まりつつある。

マルチスレッド
【まるち・すれっど】

近代資本主義とOS技術の発達は、ひとりのおさんどんをこきつかうマルチタスクに飽きたらず、多数の労働力を工場に集約、道具として使用する社会体制を生み出した。当初、何台もの紡績機を並べて糸＝threadを大量生産するために利用するケースが多かったことから、このようなしくみをマルチスレッドと呼ぶ。マルチスレッドの登場により生産効率は格段に向上したが、過酷な労働条件でこきつかわれつつそれを支えてきたプログラマーの犠牲が背景にあることを忘れてはいけない。スレッド対応のプログラムをつくるため、生麦をかじりながら徹夜を続ける女工Linuxユーザーの哀史「あゝ生麦峠」や、どうしてもスレッド関連のバグがとれないことに悩むあまり、目から光線を発射する巨大カニロボットの幻影を見るようになった業界人の姿を描く「蟹光線」は、当時の社会的不正義を告発する名作プロレタリア文学である。

マルチ商法
【まるち・しょうほう】

マルチスレッドに次ぐ新世代のコンピュータ技術として注目されているもの。頂点に君臨する“トップディストリビューター”と呼ばれるプロセスが子プロセスをfork、その子プロセスがさらに子をfork……と永遠に拡大し続けることによって、生産効率を上げるしくみ。プロセス間通信に洗剤、化

粧品、フライパンといった“モノ”を使う点も、インターネット時代にマッチするとして高く評価されている。ちなみに、このような通信手順を規定しマルチ商法発展の基礎となったのが“オブジェクト（＝モノ）指向プログラミング”である。最近では、書店でもLinux magazineの隣に「ネットワークビジネス」などと題名に冠した専門誌が並び、入門者も気軽にその恩恵にあずかれるようになった。問題点としては、上位プロセス以外のパフォーマンスがよくないところと、突然ドロンと上位プロセスが消滅してしまうことがあるという点があげられる。

デーモン
【でーもん】


デーモン降臨??

Linuxの裏社会に住み、システムの動作を不安定にしようと機会をうかがっている一連のプロセス。Webサーバが不調になったり、メールの配信が遅くなったりなど、サービスに不具合が現れるときは、かならずこれらデーモンが悪さをしているのである。少なくとも編者が同様の問題を体験したときには、かならずこれらのプロセスが動作していたので、原因であることは間違いない。サービスごとに不調を引き起こす担当デーモンが決まっているようで、Webがおかしくなったらhttpd、ファイルがアップロードできなくなったらftpd、いつのまにか蝋人形にさせられていたり吉本興業に転籍したりしていたら小暮閣下を疑いたい。

万一あなたがデーモンの被害を受けた場合には、こういったプロセスを探し出してkillすれば悩む必要がなくなるだろう。ただし、目つきが悪い・服を破りながら巨大化する・必殺技がデビルカッター・名前が不動アキラであるデーモンについては、人類の味方かもしれないので殺さずにようすを見ること。

プロセス
【ぷろせす】

（1）process。過程。結果よりも大切なもの。例文「できあがったのがどんなにひどい原稿でも、それが書かれたプロセスを思えばボツにはできない」「書かれたプロセスもひどそうなのでボツにしよう」

（2）Linuxなどマルチタスクで、独立動作するプログラムを数える実行単位。プロセスがたくさん動作しているマシンほど偉い。

```
$ ps -ah | wc -l
```

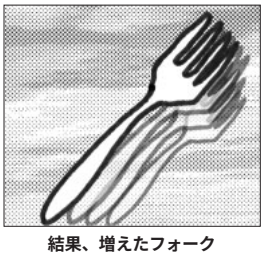
の出力は“プロセ数”と呼ばれ、過程を重んじるLinuxコミュニティにおいてLinux boxの性能を計る大切な基準と目されている。プロセ数を競うためにも、多数のプロセスを動作させておくことは重要である。もちろん、そのプロセスでなにをしていようとなにが起ころうと“結果”は問題ではない。

ゾンビ
【ぞんび】

怒念…コワイ

何らかの理由で死んでも死にきれなくなったプロセス。ハイチに伝わるブードゥ教の司祭による呪い、ふしぎな宇宙線の影響、アンブレラ社の生物兵器実験の失敗などさまざまな発生原因が考えられるが、もっとも有力なのは稚拙なコードで書かれたことをプロセス自身が恨んで、と見る怨恨の線。

fork
【ふぉーく】


結果、増えたフォーク

プロセスが子プロセスを生むこと。戦前のようにイエ意識が強かったころには、婚約、結納、三三九度を経てからforkするといった段取りが重要視されていたが、ポスト太陽族の現代では“デキたら生んじゃえ”的forkが当たり前となりつつある。いたずらなforkは限りあるコンピュータ資源の有効活用の側面からも重要な社会問題をはらんでいる。また、愛のないforkで子プロセスを抱えた若いプログラマー夫婦は虐待に走る傾向も高いようだ（例：ゾンビにしてしまうなど）。対策としては子プロセスができないよう、なるべくスレッドを使用することがすすめられているのだが、「装着時に違和感がある」「やった気がしない」などとワガママをいう男が多くて困る。

Books

Linuxプログラミングガイド

奥脇 学 著
秀和システム
B5変形判／288ページ／CD-ROM1枚付き　　本体価格 2600円

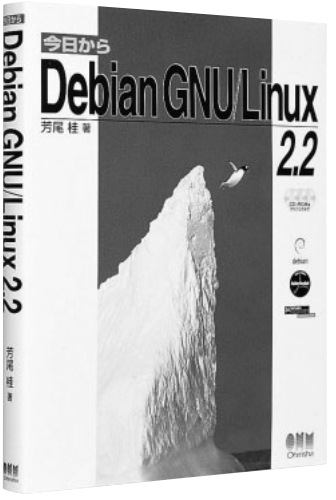
プログラミングについてまったく知らない人がよくする質問のひとつに「そもそもプログラミングって何なの？」というものがある。確かに知らない人から見れば、プログラマーがやっていることには、わかりづらい面がある。ある意味、魔術的でさえあるだろう。

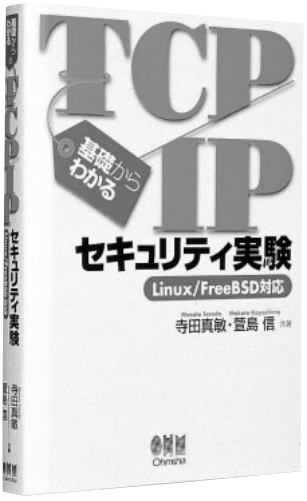
この本には、プログラミングそのものについてはほとんど書かれていない。プログラミングに際して参照する情報の集め方から、コーディングに使うエディタの紹介（Emacsが中心）、コンパイラとライブラリについて、そしてデバッグへと話は進み、最後にソースコードなどを管理するためのリビジョン管理ツールについて解説している。読者は、ソフトウェアが完成するまでの流れを追いながら、プログラム開発ではどのようなことを行うのかということをひと通り理解できる構成になっている。冒頭の「プログラミングってなに？」という疑問に応えてくれる一冊である。

今日からDebian GNU/Linux 2.2

芳尾 桂 著
オーム社
B5変形判／368ページ／CD-ROM3枚付き　　本体価格 3200円

世界で一番売れているOSだけあって、Windowsのインストーラはやはり良くできていると思う。ユーザーが判断すべき項目をできるだけ減らし、できる限り自動的に設定を行うことで、スムーズに次の段階へと進めるようにという配慮が感じられるのだ。OSを入手して最初にユーザーが触れるのがインストーラである。ヒトというのは第一印象に引きずられるものだけに、出会いが肝心なのだ。一方、Linuxはインストールの煩雑さが弱点のひとつだと言われてきたが、Debianはこの定説を打ち破る可能性を持っているのではないかと思う。現時点では、ほかのディストリビューション以上にインストールと設定が難しいと考えられているが、ユーザーインターフェイスをもう少し改良していけば、そのパッケージ管理機構の優位性を生かして、より簡単でより確実なインストールと設定が可能になるはずだ。その日が来るのを願いつつ、本書でDebianを体験してみることにしよう。



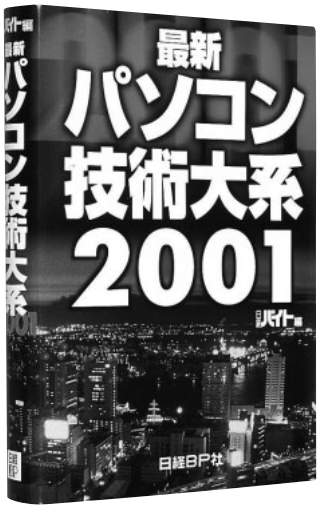


基礎からわかるTCP/IP　セキュリティ実験室

寺田真敏・萱島 信 共著
オーム社
A5判／336ページ　　本体価格 2400円

少し前に「I Love You」というウィルスが世界を騒がせた。当編集部では誰にもI Love Youメールが届かず、それはそれでなんだか悲しいという事態となった。また、先月には編集長宛に1日で800通ものメールが送られてきたこともあった。まあ、これも自分が被害者ではなかったので（日頃の感謝？を込めつつ）笑って見ていたのだが。

いくらセキュリティについてその重要性を指摘されても、実際に被害にあわないとなかなか実感できないものだ。目に見えない世界のことだけに、セキュリティのために利用されるテクノロジーも、主人を守るドーベルマンのように直接の安心感を与えてくれるわけではない。本書のねらいは、実験を通じてTPC/IPとセキュリティ技術（SSHや電子署名など）の仕組みを解き明かしていくことにある。タイトルからして「セキュリティ破りの実験をするのか！」と思われた方もいるかもしれないが、そうではないのでご安心を。



最新パソコン技術体系 2001

日経バイト 編
日経BP社
A5判／600ページ／CD-ROM1枚付き　　本体価格 3700円

コンピュータ用語の解説書というと、辞典形式のものが思い浮かぶ。これらの本は、検索性と語彙の豊富さが売りである。わからないときに引くのが辞書なのだから、それでいいわけだ。「わからないことを知りたい」というのは、これとは少しニュアンスが違っていて、辞書を頭から最後まで読み通しても知識を身に付けることはできないだろう。

とりあえず現在のコンピュータ技術について知っておきたいという向きには、本書をお勧めしたい。これだけパソコンが普及した時代であるから、まずはそれを押さえておくのは欠かせないことだ。パソコンの基礎をなすアーキテクチャから、WindowsやLinuxを含むPC-UNIXなどのOS、そしてインターネットまで幅広い分野をカバーしている。内容的にも業界の動向やその予測などもからめながら、読者が興味を持って読み進められるものになっている。CD-ROMには、本文の全文検索が可能なPDFファイルを収録。

Q&A100で学ぶ！ Perl／CGI

森下幸治 著

エーアイ出版

B5変形判／312ページ

本体価格 2600円

Pythonテクニカルライブラリ

デビッド・M・ビーズリー 著
習志野弥治郎 訳

ピアソン・エデュケーション

A5判／496ページ

本体価格 3800円

セキュアシェルリファレンス

Anne Carasik 著
トップスタジオ 訳
まえだひさこ 監修

翔泳社

B5変形判／256ページ

本体価格 2800円

インターフェースの大冒険

福富忠和 著

アスキー

四六判／244ページ

本体価格 1600円

読者の声

俺にもいわせろ！

今月の表紙はリンゴです。撮影が済んでから、モデルとなったリンゴをもらい受け、美味しくいただきました。来月はマスクメロンが食べたいなぁ。

さて、今月もたくさんのおハガキをいただきました。感謝、感謝です。

11月号特集1へのお便り

「周辺機器バトルロイヤル」よかったですネ。眠っていたUSBマウスが復活しました。この調子で対応外のネットワークプリンタの設定とかにもチャレンジしていただけると、とても参考になります。では、来月も楽しみにしています。頑張ってください。

（茨城県　大津英雄さん）

サウンドドライバの解説、ありがとうございます。ALSAのおかげで、X-Wave6000（Win用）＋Vibra128（Linux用）といった、サウンドカード2枚挿し地獄を脱出できます。

もうひとつ。Mac版Linuxの最新版が欲しくて、Vine Linux for PPCを32kbps（しかもPHS）通信でダウンロードする決意を固めた日に貴誌を発見。あと1日遅かったらと思うとぞっとします。

（宮城県　本多洋平さん）

今回の、周辺機器体当たり大検証で思い出しました。私のノートパソコンには、PCカード型のファンを付けています（ノーブランド品でした）。PCカードの端にファンがついていて、カードを通じて送風するものです。Windowsで動作すると書かれていましたが、Linuxでも動きます（たぶん、スロットからは電源を取っているだけだと思います）。

（大阪府　遠藤昌克さん）

よく、「Linuxでは最近の周辺機器を利用できない」というのを聞きます。本当にそうなのか？　現在、Linuxの周辺機器対応はどうなっているのだろうか？　という疑問からスタートした企画でした。なるべくバラエティに富んだ構成にしようということで、3人の担当者が秋葉原へ通い、気になる機器（と全然関係ない趣味の機器も）を購入したのです。ちょっとまとまりのない特集だったかもしれませんが、喜んでいただけて幸いです。

PCカードファンというのもあるんですね。5インチベイに取り付けるファンなら見たことがあるんですが。今度、Linuxで使える冷却ファン特集を組んでみましょうか（^^）。

11月号特集2へのお便り

マルチブートのための基礎知識がよかったです。もう少し、各ソフトについて説明されているともっとよかったです。GRUBを利用してみようと思いますが、MBRにインストールするのは、今の環境がつぶれるとイヤなので、ちょっと考え中です。

（福井県　服部昌博さん）

「無料で始める実践マルチブート」にひかれて購入しました。さっそくGRUBを利用して Windows 95、Linux、 COMPAQ Configuration Utility（？）をマルチブートで利用しています。

（神奈川県　佐藤貞勝さん）

マルチブートは便利ですが、PCがブートする仕組みをある程度理解していないと設定が難しいかもしれません。

11月号で紹介した「Extended-IPL」をお作りになった木村さんよりメールをいただきました。現在配布しているExtended-IPL ver.4.xxはLBAモードをサポートしていないので、OSのローダが8Gバイト以上の部分にあると起動できないのでご注意くださいとのことです。LBAモードに対応し、視覚的に操作できるメニューを備えたExtended-IPL ver 5.00を公開に向け準備されているそうです。保守ページ（http://www.tsden.org/takamiti/extipl/）にてver.5.00のベータ版をテスト公開されています。

Vine Linux for PowerMac 大反響

PPC版のVine Linuxが収録され非常

に嬉しいです。iMacで使用しておりますが、以前はグラフィック動作が遅くイライラさせられたものでした。これで快適な動作環境が構築できました。Mac OS Xのベータ版を買おうかと迷っていましたが、Vine Linux＋Mac on Linuxをメインの環境にしようかな。

（福岡県　松田 修さん）

Macなんか持っていないのにPPC版のVine Linuxにつられて買ってしまいました。目新しいディストリビューションを見るとつい欲しくなるのは、直す方法はないんでしょうか。

ところで、Macのモデムはソフトウェアモデムではないそうですね、うちのは典型的なWinモデムで、Linuxからはインターネットにつながりません。うらやましい限りです。モデムはConexant製（Rockwell HCF）のPCI接続のやつです。

（鹿児島県　松村 晃さん）

11月号の付録CD-ROMに収録したVine Linux for PowerMacに関してはたくさんのおハガキをいただきました。Macintoshユーザーの勢いを感じますね。ずいぶん昔ですが、「コンピュータをいじり倒すと、最後にはMacへたどり着くものだ」と言った人もいました。それ以来、いつかはMacと思っています。

残念ながらRockwell HCFを搭載したモデムはLinuxでは使えないようです＞松村さん

新連載［超］入門シェルスクリプト

私にとってはタイムリーな連載記事でした。1回目はまだ余裕で読んでいられましたが、しっかりついていき、シェルスクリプトを自分のものにしたいと思っております。

（千葉県　緒方宏昭さん）

11月号より連載のシェルスクリプト入門は良い。大学に入ってからLinuxを使い、学んでいる私にぴったり。ガンバロー！

（京都府　角田大輔さん）

UNIXには、単純な機能を持ついろいろなコマンドを組み合わせて複雑なことをさせるという文化があります。このような使い方とあわせ、シェルスクリプトが使えるようになると、まるで専用ツールのようなコマンドがホイホイ作れるようになっちゃいます。スクリプトマスター目指して頑張ってください。

カエルのネズミ？

USBの蛙を動かすぞ！！！

（福岡県　Samsun Baharinさん）

USBのかえる(?_?)　むむむ……。と考え込んでいたところ、プレゼント担当の編集者が、「あ、それマウスですよ。カエルの形したやつ。跳ねるんですよ」と教えてくれました。さすがはマウスフェチです。意味もなくLEDが光る300円のマウスを買ったときはとても嬉しそうでした。でも、跳ねるっていうのはウソですよねぇ。

USBマウスが動作するといいですね＞Baharinさん。

いまから活用しましょう

Linuxをパソコンにインストールして半年。苦労してインストールしたのに（LILOの8Gバイトの制約を知らずに）ほとんど活用していないよ〜。11月号から連載をスタートした［超］入門シェルスクリプトを機に、Linuxを本気で活用してみようかな？

（栃木県　桜井秀一さん）

新しいLILOは、LBA32モードに対応していますので、8Gバイト制限はありません。この問題でお困りの方は、Linuxのインストール後に最新版のLILOを入れましょう。起動フロッピーでLinuxをブートし、念のため/etc/lilo.confのバックアップを作成します。次に、http://www.ibiblio.org/pub/Linux/system/boot/lilo/からファイルを入手して、make、make installします。lilo.confに“lba32”という行を追加して、/sbin/liloを実行すれば作業は終了です（lba32は、linearとは排他オプションですので、linearという行があればlba32に書き換えます）。これでハードディスクから起動できるようになります。最新のLILOはメニュー表示がついていてちょっとイカします。

Linuxビギナーを応援します

妻がLinuxに挑戦中です。というのも、我が家にはLinuxマシンしかないからです。でも、貴誌の内容は妻には難しすぎるようです。

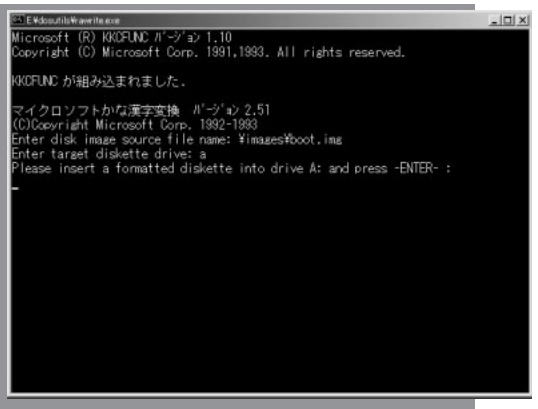
（北海道　泉 裕さん）

Linuxマシンだけという環境だと、あっという間に使いこなせるようになりそうですね。しかも、同じ屋根の下には頼もしいコーチが……。

ビギナーにもわかりやすい記事をお届けするのがLinux magazineの使命です。わかりやすいが浅くはないというのが理想なんですが、なかなか難しいものですね。みなさまからのおハガキは編集部一同、すべて拝見しておりますので、これからもご意見をお寄せください。

付録CD-ROMに収録した Official Red Hat Linux 7J（FTP版）のインストール

本誌付録CD-ROM収録のOfficial Red Hat Linux 7JはFTP版です。非商用ソフトだけが含まれています。また、製品版を販売しているレッドハット株式会社からサポートを受けることはできません。



1 インストーラ起動用フロッピーディスクの作成

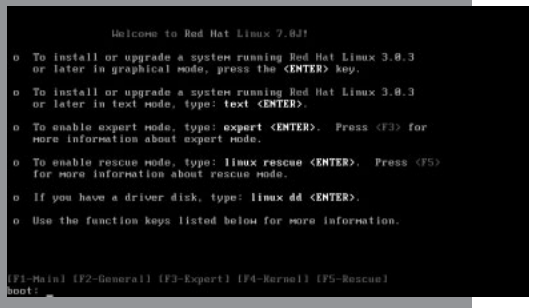
インストールするマシンがCD-ROMから起動できる場合は、CD-ROMからブートしてインストーラを起動します。CD-ROMから起動できない場合は、以下の手順で、インストーラ起動用のフロッピーディスクを作成します（ここでは、フロッピーディスクドライブがA:であるとして解説します）。

（1）Windowsのエクスプローラで、CD-ROM（Disc 1）の「dosutils」というフォルダを開き、その中にある「rawrite」をダブルクリックします。

（2）DOS窓が開き、ファイル名の入力を促してくるので、

\images\boot.img

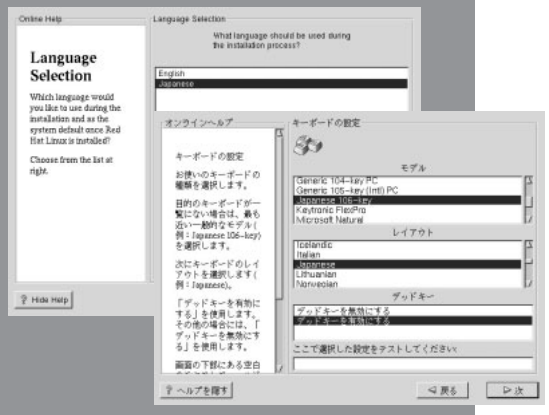
と正確にタイプして[Enter]を押します。さらに、フロッピードライブ名を求めてくるので、Aをタイプして[Enter]を押します。最後にフロッピーがセットされているかどうかを確認して[Enter]を押すとインストーラ起動用フロッピーの作成が始まります。



2 インストーラの起動

作成したフロッピーディスクや、CD-ROM（Disc 1）をドライブにセットして、マシンを再起動します。インストーラが起動し、「boot:」というプロンプトが表示されたら[Enter]を押します。

しばらくすると、Xを使ったグラフィカルな画面が表示されます。インストール画面がうまく表示されない場合は、「boot:」の箇所で「text」とタイプして[Enter]を押します。こうすると、テキスト画面のインストーラが起動します。



3 使用言語とキーボードの設定

Official Red Hat Linux 7Jでは、使用言語がデフォルトで「English」になっています。日本語環境を利用する場合は、「Japanese」を選択して、「Next」を押します。

キーボード設定のデフォルトは、モデルが「Japanese 106-key」、レイアウトが「Japanese」です。日本語キーボードを使用する場合は、このままの状態で「次」を押します。

マウスの設定

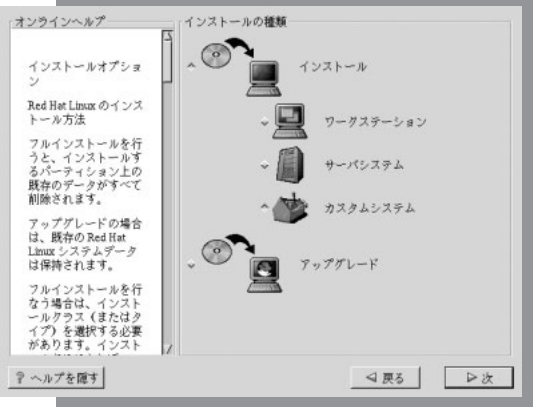
PS/2タイプの2ボタンマウスを使う場合は「2 Button Mouse(PS/2)」をチェックします。「3ボタンマウスのエミュレーションを設定する」は、2ボタンマウスの左右ボタンを同時に押すことで、3ボタンマウスの中ボタンを代用する機能です。これは便利な機能なので、チェックしたまま「次」を押します。

インストールタイプの選択

「ワークステーション」を選択すると、ブートローダLILOがMBRにインストールされます。Linuxを Windows NT/2000と共存させる場合や、 System Commanderなど、他のブートセレクタがすでにMBRへインストールされている場合は、この「ワークステーション」を選択してはいけません。

ここでは、より柔軟なオプション選択ができる「カスタムシステム」を選択して「次」を押します。

「サーバシステム」はLinuxをサーバ用途に使うユーザー用、「アップグレード」はすでにRed Hat Linuxの以前のバージョンをインストールしているユーザー用です。



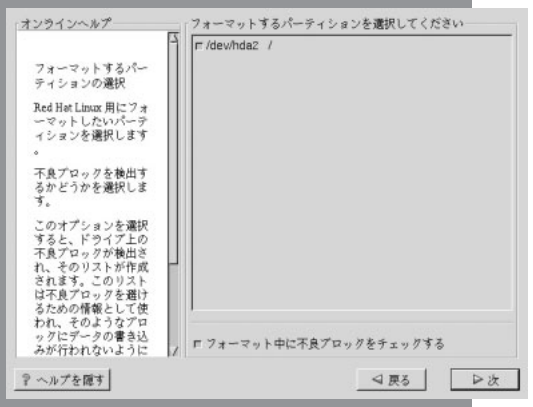
パーティション作成ツールの選択

「fdiskを使用して手動でパーティションを設定する」を選択すると、コマンドで操作するfdiskを使ったパーティション作成となります。fdiskの操作に不案内なユーザーは、「Disk Druidを使用して手動でパーティションを設定する」を選択してパーティションを作成するとよいでしょう。ここではDisk Druidを選択します。

パーティションの作成

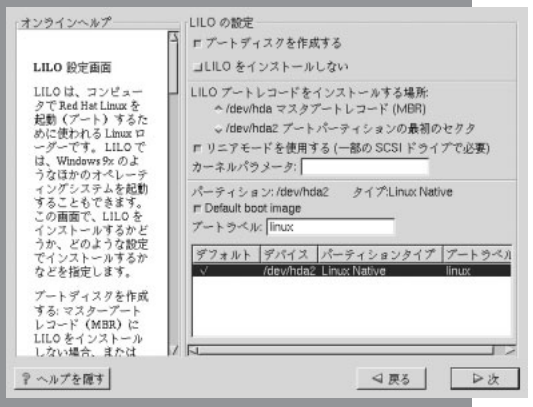
Linuxをインストールするには、最低限Linuxシステム用とSwap用の、2つのパーティションが必要です。「追加」を押すと、画面のようなダイアログが表示されます。Linuxシステム用のパーティションは、「マウントポイント」に「/」を、「パーティションタイプ」は「Linux native」を選択します。Swap用パーティションは「Linux swap」を選択します。各パーティションサイズは、以下を目安にして設定してください。

Linuxシステム用　：フルインストールの場合は3Gバイト程度以上
Swap用　：搭載メモリの1〜2倍程度



パーティションのフォーマット

前の場面で作成したパーティションをフォーマットします。チェックされている「/dev/hdax」パーティションがフォーマットされます。「フォーマット中に不良ブロックをチェックする」を選択しておくと、パーティションのフォーマット中に、ディスクの不良箇所を調べてくれます。急いでいなければ、チェックを入れておくとよいでしょう。最後にもう一度フォーマットするパーティションを確認して、「次」を押します。



LILOの設定

「LILOブートレコードをインストールする場所」は環境に応じて選択します。

/dev/hda マスターブートレコード（MBR）
- LILOを使用してWindows 95/98とLinuxを起動時に選択する
- ディスクにLinuxのみをインストールする

/dev/hda2 ブートパーティションの最初のセクタ
- System Commanderなど、LILO以外のブートマネージャを使用する
- LinuxとWindows NT/2000を共存させる

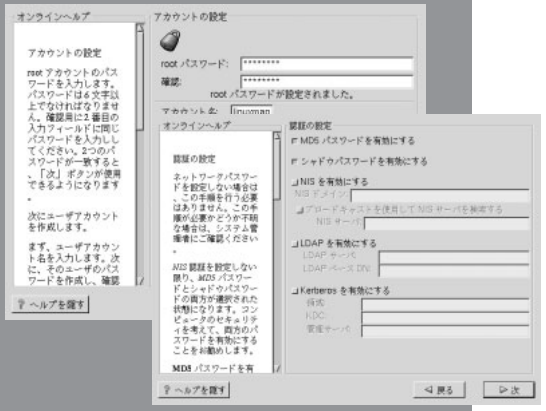
緊急時のために「ブートディスクを作成する」をチェックしておくとよいでしょう。

ネットワークとタイムゾーンの設定

家庭内でISDNルータやほかのサーバマシンでDHCPサーバを稼動させている場合は、ここで「DHCPを使用して設定する」を選択します。DHCPサーバがない環境でLinuxを使う場合は、以下のようなネットワークアドレスを入力するとよいでしょう。

IPアドレス	192.168.1.2～192.168.1.254
ネットマスク	255.255.255.0
ゲートウェイ	家庭内サーバマシンのIPアドレス
1番目のDNS	プロバイダのDNSサーバのIPアドレスなど

次のタイムゾーンの設定は、デフォルトで「アジア／東京」が選択されています。Linuxを日本時間で使用する場合は、このままの状態で「次」を押します。



ユーザーアカウントと認証の設定

Linuxを使用するユーザーを設定します。まず、Linuxシステム管理者のパスワードとして、「root パスワード」と「確認」に同じものを入力します。この管理者のログイン名は「root」です。

次に一般ユーザーのアカウントを設定します。「アカウント名」、パスワードとして「パスワード」と「パスワード（確認）」に同じものを、「フルネーム」にユーザーのフルネームを入力して、「追加」を押します。この要領で複数の一般ユーザーを作成できます。ユーザー設定を終えたら「次」を押します。

認証の設定画面は、デフォルトの状態で「次」を押します。

パッケージグループの選択

おおざっぱなパッケージグループを選択します。パッケージ群の意味がわかるユーザーは、好みのパッケージ群を選択し、どのパッケージ群を選んでよいかわからないユーザーは、最下段にある「全部」（すべてのパッケージ）を選択するとよいでしょう（全部を選択すると約2.1Gバイトのディスク領域を使用します）。パッケージグループを選択し終えたら「次」を押します。

12

Xの設定

モニタの一覧が表示されます。リストに使用するモニタがない場合は、モニタのマニュアルを参照しつつ「Generic」から無理のない解像度のものを選択します。

次にXサーバの設定です。ほとんどのビデオカードは自動で認識されます。

「デフォルトのデスクトップ」ではデフォルトのデスクトップ環境を選択します。ここではデフォルトの「GNOME」を選択します。

色数や解像度を手動で設定する場合は「X設定のカスタマイズ」をチェックし、グラフィカルなログイン画面を使う場合は「GUIログイン」をチェックして「次」を押します。

13

パッケージインストールの開始

画面（奥）で「次」を押すと、パッケージのインストールが始まります。パッケージのインストール中にディスク2を要求された場合は、CD-ROMドライブにDisc 2をセットして「OK」を押します。パッケージのインストールが終わるまでしばらく待ちましょう。

14

インストールの終了

LILOの設定場面で「ブートディスクを作成する」をチェックした場合は、手前の画面が表示されます。「次」を押すと、緊急時用のブートディスクの作成が始まります。1台のマシンにLinuxとWindows NT/2000を共存させる場合は、特にフロッピーディスクの作成をお勧めします。フロッピーの作成が終わると、奥の画面が表示されます。フロッピーディスクを抜いて「終了」を押します。

以上でインストール作業は終わりです。お疲れさまでした。

15