

NEWS EXPRESS

Distribution

Software

Hardware

Headline

Event

iモードからイントラネットへ接続する みたi君

NTT-MEは、iモード携帯電話からイントラネット内の情報を閲覧できるモバイル・イントラネットサービス「みたi君」(みたいくん)を、9月7日より発売した。

システムは、本体にデルコンピュータ製サーバ、OSにはTurboLinux Server日本語版6.1、そしてNTT-MEが開発したソフトウェアを搭載している。サーバの設置形態として、ユーザーのLAN内に設置する場合と、NTT-MEデータセンター内にサーバを設置するハウジング、NTT-MEのサーバを使ってサービスを受けるホスティングの3種類が用意されている。

みたi君は、NTT-MEが独自に開発した、Webコンテンツをiモード向けに自動変換する機能やユーザー認証の機能を持っている。

Webコンテンツの変換機能は、WebページのHTMLをiモード用のコンパクトHTML向けに最適化することで行われ、コンテンツ作成者にとってはiモード専用ページを別に作成する必要がないというメリットがある。

変換機能の仕組みは、まずロボットプログラムによってWebページを収集し、iモード専用のページを自動的に作成しておくことで行う。そのため、みたi君サーバの管理者が、iモードユーザーに見せるページを制限することが可能だ。変換によってバナー広告を省くことが許可されないWebページへのアクセスを回避することで、著作権上の問題をクリア可能としている。

なお、ロボットの更新頻度、探索階層数指定や、iモード用に変換する際のページ分割数などを設定できる。

POP3メールユーザー認証、UNIXパスワード認証、RADIUS、その他の独自認証システムと接続できるため、イントラネット内にあるメールサーバに接続し、会社や学校のアカウント宛でのメールをiモードから読むことが可能だ。



発売日
2000年9月7日
発売
株式会社エヌ・ティ・ティ・エムイー
TEL
03-5200-4420
価格
500万円～(ユーザー設置タイプ)
URL
<http://www.ntt-me.co.jp/>

Hardware

発売日

インターネットアプライアンスサーバ
Internet All in One ServerシリーズURL <http://www.proside.co.jp/>

プロサイドは、インターネットアプライアンスサーバ「Internet All in One Serverシリーズ」を8月25日より発売した。ブックサイズの「コンパクトタイプ」とラックマウント1Uサイズの「スタンダードタイプ」、同2Uサイズの「高性能タイプ」の3モデルが用意されている。

ブックタイプは、メモリ64Mバイト、20Gバイトハードディスク、10/100BASE-TXネットワーク

2000年8月25日

発売	プロサイド株式会社
TEL	043-279-9280
価格	10万8000円～

で価格は10万8000円。高性能タイプは、Pentiumデュアルプロセッサ、18GバイトのRAID 5対応ハードディスクを搭載している。

Webサーバ、メール、FTP、ファイル共有（Windows / Mac）、DNSなどの機能を、Webブラウザから設定が可能だ。

そのほか、Webアクセス認証サーバ、データバックアップサーバも9月中旬に発売された。



スタンダードタイプ

Hardware

発売日

TurboLinux 日本語版 6.1 をプレインストール
TNS Micro Server 2000URL <http://www.tnsservice.co.jp/>

トータルネットワークサービスは、コンパクトなオールインワンサーバ「TNS Micro Server 2000」を9月1日より発売した。CPUにPentium 667MHz、メモリ128Mバイト、20Gバイトハードディスク2台を搭載し、RAID 1でミラーリングを行うことで、ハードディスク故障時のデータ損失を防いでいる。価格は32万8000円から。

TurboLinux Server日本語版6.1と各種ソフトウェアをプレインストールしており、設置してすぐに使うことができる。インターネット接続時の各種設定

2000年9月1日

発売	株式会社トータルネットワークサービス
TEL	03-5792-5301
価格	32万8000円～

は、HDE Linux Controllerツールを利用してWebブラウザから設定が可能。

また、NTTのフレッツISDNでInfoSphereのIP常時接続サービスを利用して、独自ドメインでインターネットサーバを運用するサービスを、すべてセットにした「TNS Micro ServerIP Set」も用意されている。Micro Server 2000本体とISDNルータ、リモート監視、オンサイトサポート保証、手続き代行までが含まれていて、価格は4年間のリース契約で月額2万8500円からとなっている。



Hardware

発売日

Pentium Xeon 1GHz搭載中型PCサーバ
PowerEdge 4400URL <http://www.dell.com/jp/>

デルコンピュータは9月19日より、部門サーバ「PowerEdge 4400」にPentium Xeon 1GHz搭載モデルを追加した。128MバイトECC 133MHzメモリ、オンボードUltra3 SCSIコントローラ×2（ハードディスク用）、オンボードUltra/Narrow SCSI-3コントローラ（フロントベイ用）、9Gバイト10,000rpm Ultra3 SCSIハードディスク、40倍速SCSI CD-ROMドライブ、10/100BASE-Tネッ

2000年9月19日

発売	デルコンピュータ株式会社
TEL	044-556-6190
価格	58万8000円～

トワーク、Dell Server Assistant CD-ROM、OSなしの構成で58万8000円。

プレインストール可能なOSは、Red Hat Linux 6.2J、Windows NT 4.0 / 2000 Server / 2000 Advanced Serverとなっている。

そのほか、PowerEdge2400、PowerEdge2450シリーズには、Pentium 1GHz搭載モデルが追加された。



Hardware

発売日

Linuxでフォールト・トレラントシステムを構築
PC FT System Kit for LINUXURL <http://www.duaxes.com/>

デュアキシズは、フォールト・トレラントシステムキット「PC FT System Kit for LINUX」を、8月21日から発売した。PCIスロット用のPC-DPUインターフェイスボード2枚と、FT-SWUコントロールユニット、FT-SKTソフトウェアCD-ROMで構成され、価格は98万円。

2台のLinuxサーバのPCIスロットにPC-DPUを装着し、FT-SWUに接続してセットアップを行うと、

2000年8月21日

発売	デュアキシズ株式会社
TEL	03-3523-6933
価格	98万円

2台のサーバのTCP/IPデータ転送を二重化し、フォールト・トレラントシステムを構築できる。

マスタサーバで障害が発生した時には、約100msという短時間で、もう1台のバックアップサーバに切り替えて処理を継続する。完全に同一スベックのマシンで、同時に処理を実行させておき、相互の動作監視はハードウェアで行うところが、フェイルオーバークラスシステムと違う点だ。



Hardware

発売日

2000年10月

低価格インターネット・オールインワンサーバ JP-7020E

URL <http://www.jpcltd.co.jp/>

日本パーソナルコンピュータは、インターネット・オールインワンサーバ「JP-7020E」を10月より発売する。CPUにCeleron 667MHz、メモリ128Mバイト、10Gバイトハードディスク、フロッピー、CD-ROMドライブ、10/100BASE-Tネットワーク、キーボード、マウス、15インチCRTディスプレイ込みで16万8000円。

OSにLinuxを採用し、Webサーバ、メールサーバ、DNSサーバ、FTPサーバ機能を設定済みなので、すぐにインターネットサーバとして使うことができる。さらに、データベースソフトPostgreSQLを利用して、データベースと連携したWebサービスを簡単に行えるサンプルが添付されている。

また、SambaによるWindowsファイルサーバや、IPマスカレードによるルータ機能も備えている。



発売	日本パーソナルコンピュータ株式会社
TEL	0426-46-7667
価格	16万8000円～

Software

発売日

2000年9月25日

セキュリティを強化したLinuxサーバ設定ツール HDE Linux Controller 2.0 Professional Edition

URL <http://www.hde.co.jp/>

ホライズン・デジタル・エンタープライズ（HDE）は、LinuxサーバをクライアントのWebブラウザから設定、管理する「HDE Linux Controller 2.0 Professional Edition」（以下Professional）を9月25日から発売した。価格はオープンプライスで11月までは直接販売のみとなっている。

ProfessionalではSSL（Secure Sockets Layer）を搭載することで、インターネット経由でも安全な利用が可能になった。

8月に発売された同Standard Editionの機能に加え、サーバ管理の一部の利用権限を特定のユーザーに与えることや、設定を復元させるUNDO、ネットワーク経由で不正なログインの兆候があった場合に警告メールを送るswatchログ転送機能を搭載した。

そのほか、APOP、Proxy、パーチャルホスト、ディスクquota（使用量制限）、ユーザーアカウントCSVアップロードなどの設定機能が追加された。



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	オープンプライス

Software

発売日

2000年10月20日

ディスク自動分割&最適化ツール パーティションコマンダー6

URL <http://www.softboat.co.jp/>

ソフトボートは、パーティション操作ユーティリティ「パーティションコマンダー6」を10月20日から発売する。CD-ROMで提供され、価格は9800円。

パーティションコマンダー6は、Windows 95 / 98 / Me / NT / 2000で動作し、ハードディスクの空き容量を増やす / ディスクアクセスを速くする / ドライブ構成の変更 / 新しいIOSをインストール、といったパーティション操作をメニューから選ぶだけで自動的に処理することができる。

新機能として、Windows NT / 2000のNTFSパーティション、Linuxパーティションのサイズ変更、NTFS（圧縮）パーティションからFATへの変換、WindowsとLinuxパーティション間の空き領域の移動などを備えている。

パッケージには、1台のPCに複数のOSをインストールし、ブート時に起動したいIOSを選ぶことができるブートセクタ「システムコマンダー パーソナル」が含まれている。



発売	株式会社ソフトボート
TEL	03-3256-4711
価格	9800円

Software

発売日

2000年9月13日

高機能メーリングリストサーバ Lyris 4.0 日本語版

URL <http://www.awavetech.com/lyris/>

カナダのActivewave Technologiesは、高機能メーリングリストサーバLyris（ライリス）の新バージョン「Lyris 4.0日本語版」を9月13日にリリースした。動作OSは、Windows、Solaris、Linuxで、価格は5万9800円から。50メーリングリスト、200メンバー（各リスト）までの運用なら無料である。

Lyrisは、受信したメールをあらかじめ登録された複数のメールアドレスに同時配送するサーバソフト

ウェアで、Webブラウザから管理ができる。小規模な社内連絡用メーリングリストから、大規模な商用メーリングリストまでスケラブルに対応する。

メール配送エンジンの改良により、最高で1時間に50万件のメール配送が可能になったほか、運用状況分析レポート、サーバ上に保存しておいた広告用テキストを自動的にローテーションして挿入するなどの新機能を搭載した。



発売	Activewave Technologies Inc.
TEL	+1-604-893-7017
価格	5万9800円～



インプライズのJava開発環境JBuilder 4 Foundationは12月無償配布の予定 2000年9月22日

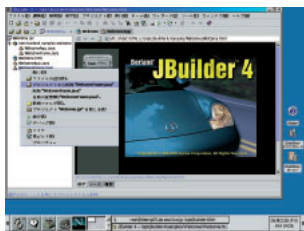
インプライズのJava統合開発環境「JBuilder 4 Foundation」の配布予定が、12月であることが判明した。雑誌添付を中心になる予定。WebダウンロードやCD-ROM送付サービスも提供するという。

JBuilder 4は、それ自身が100%Javaで記述されたビジュアル開発ツールで、Java2 v1.3に対応、Javaアプリケーションやアプレット、JSP/Servlet、JavaBeansなどの開発をサポートする製品だ。「ビジュアルデバッグ」や「クラスブラウザ」、「コンパイラ」、「デザイナ」、「エディタ」などが1つにまとめられており、プログラマの生産性を高めることができるという。対応プラットフォームはWindows、Solaris、Linuxの3つだ。

JBuilderの製品ラインナップは大規模分散アプリケーション開発向けの「Enterprise」、サーバアプリケーション開発向けの「Professional」、Javaの学習に適した「Foundation」の3つに分かれおり、そのうち「Foundation」は無償で提供されている。JBuilder 4 Foundationも無償で提供される。

ServletやJSPの開発を行えるのは「Professional」以上の製品ラインである。JBuilder 3.5におけるラインナップごとの機能一覧はインプライズのWebサイトで参照することが可能だ。

インプライズ (<http://www.inprise.co.jp/>)



Mozillaのロードマップを公開。 Mozilla 1.0は2001年第2四半期に登場 2000年9月21日

Mozillaの今後の開発方針とスケジュールを示したロードマップが公開された。これによるとMozillaは、Netscape PR3リリースのタイミングでメインのMozillaとNetscape 6に技分かれして開発が進み、Mozilla 1.0のリリースは2001年第2四半期の予定だ。

Mozillaのコードのツリーは、まずNetscape PR3の「ブランチ」(枝)とMozillaの「幹」に分かれる。「幹」はそのまま開発が続けられ、IBMのBiDi(左から右へ書く言語のサポート)やActiveState DOMスクリプティングサポートなどが統合される。一方、NetscapeのブランチはNetscape Communicationsの管理方針のもとに開発が進められ、Netscape 6出荷のタイミングで、“Mozilla”の名がつくブランチとNetscape 6.0xとのブランチの2つに分けられる。このバージョンは“Mozilla 0.9”と呼ばれる予定だ。

それぞれのブランチの開発者は、変更したコードをメインの幹にも反映することが義務づけられている。

Mozilla 0.9リリース以降、マイルストーンは「mozilla0.90」のようにバージョン名で呼ばれることになる。2001年第1四半期にmozilla0.90/0.91、第2四半期にmozilla1.0/1.01がリリース予定で、第4四半期のmozilla1.21までがロードマップに示されている。

Mozilla Organization
(<http://www.mozilla.org/>)

Sun、Cobalt Networksを 20億ドルで買収

2000年9月20日

Sun Microsystemsは9月19日、LinuxサーバアプライアンスメーカーのCobalt Networksを20億ドル相当の株式交換により買収すると発表した。成長しつつあるサーバアプライアンス市場におけるSunの参入を加速することが目的だという。

Cobalt Networksは、立方体形の小型サーバ「Cobalt Qube」やWebホスティング用途のラックマウントサーバ「Cobalt

RaQ」、Webキャッシングサーバ「Cobalt CacheRaQ」、ファイルサーバ「Cobalt NasRaQ」などを販売している企業。これらのサーバは特定用途に特化しているので、汎用サーバと比較して安価で扱いやすいものとなっている。

Cobaltの製品は、Sunの製品ラインにローエンドのサーバアプライアンスを追加するものだとSunは述べている。

買収は株式交換で行われ、Cobaltの株主は1株あたりSunの株式0.5株を受けとる。買収は2000年12月31日までに完了する予定だ。

Sunは最近、Linuxへの接近を示しているが、製品ラインナップを支えているのは自社のUNIX「Solaris」だ。Linuxのスケラビリティは着々と向上しつつあるが、Solarisほどには到っておらず、ハイエンドでSolarisを代替するのは難しい。しかし価格の安いIntel互換チップを搭載したサーバアプライアンスでは、コストパフォーマンスとカスタマイズ性のよい選択肢であり、さまざまな企業がこの市場に参入している。今回の買収により、SunもLinuxサーバアプライアンス市場に参入することになる。

Sun Microsystems (<http://www.sun.com/>)
Cobalt Networks (<http://www.cobalt.com/>)

いよいよLinux 2.4に近づいたカーネル、 2.4.0-test9-pre2がリリース 2000年9月18日

Linus Torvalds氏は9月17日、Linuxカーネル開発版の最新となるLinux 2.4.0-test9-pre2をリリースした。このバージョンはLinux 2.4.0-test9の前身となるもので、さらに開発が進めばLinux 2.4.0として正式にリリースされる。

Linus氏はtest9-pre2リリースのメールの中で、「主要なバグがないところまで来たと思う」と述べて、Ted Ts'oのリストで“critical”(致命的)に分類されていないバグに対するパッチは、test9リリースをもって受け付けない方針を示した。

Linux 2.4は、1999年にはリリースされるはずだったが、ずるずると遅れて2000年の夏が終わった時点でもリリースされていない。しかし、今回のアナウンスのように確実にプロジェクトが進んでいることは確かで、

今秋にLinux 2.4が出る可能性は高いだろう。
Linux 2.4.0-test9-pre2のダウンロード
(<http://www.jp.kernel.org/pub/linux/kernel/testing/>)

Trolltech、QtをQPLとGPLのデュアルライセンスに KDEのライセンス問題解決へ
2000年9月4日

KDEの配布が合法と確信できる時が来るようだ。ノルウェーのTrolltechは、ツールキットの次期バージョン「Qt/Unix 2.2.0」を、従来のQPL (Q Public License) に加えて、GNU GPL (General Public License) も選択可能なデュアルライセンスでリリースする。QPLでライセンスされるQt Free Editionと、GPLなKDEデスクトップ環境との組み合わせは、ライセンスについての議論を巻き起こしていた。QPLは「オープンソース」として認められてはいるが、GPLと互換性がないという懸念である。

問題となるのは、GPLが要求する配布条件にQPLが適合しない可能性があるという点だ。GPLなプログラムを配布するためには、ライブラリを含むプログラム全体がGPLでなければならないが、OSの主要な要素(カーネルなど)と一緒に配布される構成要素(libcなど)については、例外的にGPL以外のライセンスが許されている。ただし、その構成要素をGPLなプログラムに付随して配布することは許されない。

QtはKDEに付随して配布されると見なすことができるので、KDEの配布はGPL侵害に当たる可能性があった。そのため、Debian GNU/LinuxなどはKDEを配布してこなかった。

QtのライセンスにGPLが選択可能になることで、この問題は解決する。

なお、これは無料でQtを商用ソフトに使用できることを意味するわけではない。QPLがリンクを認めるのはフリーソフトに限られ、またGPLがリンクを認めるのはGPLに限られるからだ。従って、商用ソフトウェアの開発にはQt Professional Editionを購入する必要がある。

Qtのほかに、StarOfficeやMozillaが、独自ライセンスとGNU GPLのデュアルライセンスでソフトウェアを配布することを決めている。

ライセンス変更のアナウンス

(<http://www.trolltech.com/company/announce/generalpl.html>)

Red Hat、Linuxカーネル内で動作する高速Webサーバをリリース
2000年9月4日

米Red Hatは9月1日、高速なWebサーバ「TUX」の開発者向けプレビュー「Hawaii」リリースを公開した。TUXはCPUの特権モードで動作するLinux専用のWebサーバで、Webサーバのベンチマーク「SpecWeb99」でDellのLinuxマシンが飛び抜けた値をマークしたときにも使用された。

TUXはカーネルモジュールとして動作し、スタティックなコンテンツのみをサービスする。ダイナミックなコンテンツについては、80番ポート以外で待機しているApacheなどのユーザープロセスにリクエストをそのままダイレクトし、それらに処理を任せることが可能だ。最近のWebでは動的なコンテンツが多用されるようになったとはいえ、画像などはほとんど静的なので、コンテキストスイッチの必要がないカーネルモジュール内でHTTPを処理するメリットは大きいという。

また、TUXの持つコンテンツのキャッシュ機能も、できる限り高速にサービスできるように最適化されている。キャッシュされたデータは、あらかじめ計算されたTCPチェックサムと共にページキャッシュに保持され、DMAによりコピーなしにネットワークカードに送られるため、非常に高速に処理される。新たなシステムコール“tux(2)”を利用すれば、ダイナミックなコンテンツもキャッシュすることが可能だ。さらに、動的に生成されたデータとキャッシュされたデータの混在するページを処理することもできる。

TUXの設定は、/proc/sys/net/http/*に置かれた仮想ファイルに書き込むことにより行う。

TUXのライセンスは、カーネルと同じGPL。正式なリリースは2000年9月末の予定だが、プレビュー版はRed HatのFTPサイトまたはミラーサイトからRPMソースパッケージをダウンロードすることができる。

TUXのダウンロード

(<ftp://ftp.redhat.com/redhat/tux/>)

HP、Intel、IBM、NECが共同でLinux開発支援ラボを設立

2000年8月31日

米Hewlett-Packard、米Intel、米IBM、NECの4社は共同で、Linuxの開発支援を行う非営利の研究所「Open Source Development Lab」を設立する。研究所は、2000年末に米国Oregon州Portlandでオープンする予定。

4社は今後数年間にわたり、研究所に対して資金と機材の提供を行う。

この研究所の目的は、エンタープライズに必要とされる機能をLinuxに付け加えることだ。その目的のために、研究所が新たなプロジェクトを立ち上げることはないが、その代わりに、オープンソースコミュニティによる開発を加速する支援を行う。

研究所の運営は、オープンソースコミュニティとスポンサー企業 上記の4社と、Caldera、Dell、Linuxcare、LynuxWorks、Red Hat、SGI、SuSE、TurboLinux、VA Linuxの各企業 のメンバーから構成される独立した委員会により行われるという。

プレスリリース

(<http://www.intel.com/pressroom/archive/releases/cn083000.htm>)

Sun、Solarisの国際化テクノロジーをX.orgに提供

2000年8月31日

米Sun Microsystemsは8月29日、Solarisの国際化のためのソースコードをX.orgに提供すると発表した。X.orgは、X Window Systemの標準を定める非営利団体。

このソースコードの提供によって、アプリケーション開発者は国際化されたソフトウェアの開発が容易になる。サポート対象となるのは37言語、123ロケールで、右から左に書くアラビア語や結合文字を利用するタイ語のように、レイアウトの複雑な言語を表示することも可能になるという。

ソースコードは、X Window Systemと同じオープンソースなライセンス(Xライセンス)のもと、2000年9月15日に提供される予定。

米Sun Microsystems (<http://www.sun.com/>)

NECのLinux戦略

PCサーバ分野で国内1位のシェアを誇るNECは、Linuxへの取り組みも早くから行っていた。外資系メーカーはLinuxへの取り組みを本格化させ、Linuxプレインストールマシンをフルラインナップで揃えてきている。そんななか、NECはLinuxサーバでも強さを見せられるのか、Linuxマーケティング戦略を、NECニュービジネス企画部マーケティングマネージャーの川井俊弥氏に伺った。

interview

Linuxの市場について

国内のLinuxのシェアはどの位でしょうか？

川井：今年7月に発表されたIDCのレポートでは、'99年の日本の国内サーバシェアでLinuxは4%です。その前の年は、0.7%でしたので、非常に伸びています。これが2004年には12%になると予測されています。

しかし、このデータ自体はOSとして売っている本数を数えていて、我々から見るとすごく少なく見えます。その4%というのは約1万7000本ですけど、

Linuxはダウンロードすればだれでも使えますし、ネットワーク・アプライアンスサーバという組み込み機器が入っていないらしいんですよ。ほかのOSの数は合っているようですけど。

ですからこの数字は、実際に使われているのと合っていないと認識しています。

そこで、ある調査会社のデータを元に、ハード、アプリケーション、SIなどを含めて、本当の国内のLinux市場規模を算出してみたら、'99年度100億円、2000年度200億円、2001年度以降は、300、500、700億円と平均62.3%の伸びと見積もっています。

金額としては商用UNIXやNTと比べて小さいですけど、これから急成長していくと思っています。

NECのユーザー事例

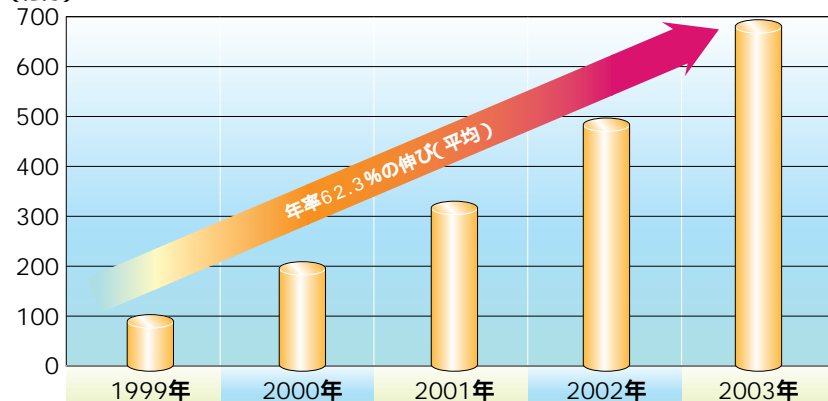
川井：次に、どういうところに適用されているのかというと、メールやWebのネットワークサーバ（インターネットサーバ）やデータベース、それとあまり知られていないが業種専用のシステムに使われています。

たとえば大学などの教育機関向けや、医療用の専用端末。これは、もともとシステムをUNIXで作っていて、アプリケーションはお客様が開発していたのを、PC+Linuxベースで作ったわけです。その結果、300万円だったのが200万円ぐらいに価格が抑えられました。リース切れでリプレースする時期だったこともあって、日本全国で3000台ぐらい使われています。

そういう見えないマーケットが最近結構あってですね、それらのデータは先ほどの統計には含まれていないわけです。

それは、Linuxを見える形で販売していないからですか？

(億円)



国内のLinux市場規模（NEC推定）

PCサーバにおけるLinuxの出荷数量／金額をベースに、ハードウェア／SI比率を加味した市場規模

川井：我々はハードとドライバを開発して納めただけです。お客様がインテグレーションを全部やっていて、そこが薬局などに出荷しています。そういう点を含めて200億円とか300億円といった広がりを見せていくのではないのでしょうか。

NECは昨年3月から本格的にLinuxに取り組み始めました。1年やってきて納入実績はどれくらいあるのかというと、箱売りだけではなく、インテグレーションしたり、インストールサービスしたりで100社以上の実績があります。

その中の代表的な事例を2つ紹介しましょう。

まず、稚内信用金庫の場合ですが、こちらはお客様が「Linuxで」と言われたのではないのですよ。「インターネットサーバをやりたい、内部からも透過的にアクセスしたい」ということでしたので、インテグレートしたNECソフトウェア北海道（DNES）が、「それならLinuxでできますよ」と言って使ってもらったら非常に好評でした。システムはOCNルータ、インターネットサーバ、イントラネットサーバの3階層のファイアウォールを持っていて、その2台のサーバはLinuxで動作するExpress 5800です。

外からアタックが来ても中までは入ってこられなかったとか、中からシームレスにアクセスできるとかいう、監査レポートを定期的にキチンと出すことによって信頼が得られました。

札幌（DNES）と稚内というように離れていても、リモート保守ができて、何かあったときにもメールでやりとりします。いちいち行かなくて済むため、トータルのコストを削減できるのも好評の理由です。

NTではリモート保守はできないのですか？

川井：NTの場合はNECの保守センターで一括してやっています。そして、エクスプレス通報サービスという保守契約によって、リモート保守/監視を依頼する必要があります。ですが、この場合はその契約をしなくても、NECソフトウェア北海道だけでできます。なお、そのサービスはNTでしかできなかったのですが、去年の夏ごろからExpressベースのハードウェア検知機能がLinux上で全部動作するようになっていきます。

次に、甲府信用金庫の場合は全然逆で、情報システム部門がしっかりしていて、ACOS（汎用機）やEWS4800（UNIX）を導入していたこともあってテクニカルに明るいわけです。

最初は、新しいことをやるときに予算が抑えられていて、ハードを買って、DB買ってというのは非常につらかったために、「それだったらLinuxは安いハードでできるし、ちょっとやってみようか」ということから始めたらしいのです。

その結果、Webサーバでスタートしたのですが安定して使えることがわかったので、業務も載せてみようというDBを入れはじめたりして、Linuxをいろいろなところで使っています。最近では、モバイルバンキングのコンテンツサーバもLinuxで構築しています。

コンテンツサーバを立ち上げたり、インターネットサーバ、基幹業務のデータを落とす「SimpWright」というWeb情報管理パッケージで見られるようにすると、情報管理をやるとかのベース

にLinuxを使っていきたいと言っておられました。

このお客様は、Linuxを指名していますし、Linuxをだいたいわかっています。我々、特にNECソフトウェア北海道では、テクニカルな支援とか全体のソリューションをアドバイスしています。

そのほか、中部電力ではネットワーク監視にLinuxベースでOracle 8iを入れてやっていますし、跡見女子学園大学では、PC98-NXで70台ほどNTとLinuxのデュアルブートシステムを導入しています。主にUNIXの講義に使うようです。ここでは、リカバリーCD-Rを用意して障害時に簡単に初期状態に戻せるようにしています。

あとは、セキュアWebを活用したBtoBシステムや、Solarisの置き換えでネットワークを再構築したり、同報メールシステムを構築した例などがあります。

先ほどの、医療機関向けのシステムでは、ソフトウェアRAIDのセットアップと、LinuxでIBMフォーマットのフロッピーディスクに対応するドライバの開発を、NECで行いました。

IBMフォーマットのフロッピーをLinuxでアクセスしたいのですか？



川井 俊弥氏
日本電気株式会社
NECソリューションズ マーケティング本部
ニュービジネス企画部マーケティングマネージャー

川井：従来のシステムとの互換性のためでしょうね。探したのですが、見つからなかったので一から作りました。

Linuxを勧める理由

Linuxを使いたいというのは、お客様から？ それともNECから勧めているのですか？

川井：UNIXに明るい情報システム部門にいる人たちは、Linuxに興味ありありで、だいたいみんなやっています。そして本格導入するときに、ハードウェアを含めてサポートできるのかを聞かれます。マクロ的に見ると、いままでそういう方たちが多かったですね。

お客様の要望は、Linuxでも商用UNIXでも、NTでも同じようにやってほしいと思っているわけです。

ただし、我々はお客様に対して「Linux自体は保証できるものではないですよ。オープンソースのもの全部そうなのですが保証はできません」と話しています。

けれどOSが何であっても、ハードウェアが壊れれば修理しますし、保守も行います。サポートについても契約を結ぶことで、わからないことにもテクノロジーを含めて対応します。それは、ソースコードを追いかけることができるので可能です。

それでも「全部やって欲しい」というお客様には、覚え書きとか契約するときに、「ここまではできますけど、これ以上はできません」というのを交わしたうえで行っています。

全然違う観点では、お客様がそういうOSに関しては「なんでもいいよ」という場合、よく言われる「丸投げ」という形があります。Linuxであろうが、なかろうが関係ないんです。このシステムをキチンと動かすという契約なの

で、Linuxで何か起こっても我々が責任を持って対処することになります。

インターネット/イントラネットサーバには、Linuxを特に勧めているのでしょうか？

川井：Linuxで安定して、それなりに使えますから。NTでもできますが、SI業者としてやりやすいからLinuxを選んだということでしょう。また、お客様のトータルのコストで収まるようにする必要もあります。

NECのLinuxへの取り組み

川井：現在Linuxへの対応は、ハードウェア、ソリューションサービス、ソフトウェア、教育コース、インストールサービスを提供しています。

最終的には、お客様にLinuxに対して安心感を持ってもらわないといけないと考えています。いろいろと不安だとか、コミュニティが開発しているとか、保証とかに関して言われるので、そのためにもいろいろなものを提供しています。

NECは、'99年3月にWebで情報発信することから始めたのですが、ハードウェアを作っているベンダーが、Linuxに対して「動く、動かない」という情報を発信したのは、実はNECが世界初でした。今ではほかの会社もやっています。当たり前になっていますけど。

他社がやっていないことで特徴的なのは、Linux関連企業へ出資していることでしょう。現在、米TurboLinux、テンアート二、ミラクルリナックス、VA Linux日本法人へ出資しています。

商用OSと何が違うかといいますが、Linuxのビジネスというものは全部自分でできるものではないと思っています。パートナーシップを組んで協業しながらやっていくものです。

Linux対応製品とサポート

川井：また、58Linux対応モデルに関しては、4Way以下のすべてのPCサーバはLinuxに対応します。動作確認もキチンと行っていますし、要望があったのでLinuxインストール、基本サポート込みのLinuxパックモデルも用意しています。

7月には、「Express5800インターネットプライアンスサーバ」という必要なものを全部入れて、Webやメールといった機能に特化したモデルを発売しました。この製品は、それぞれ組み合わせる時に統合的に管理することができるツールを組み込んであるのが特徴です。

そして、それをさらに組み合わせて、ルータやスイッチなどのネットワーク機器と導入管理などの総合的なサービスをセットにした「Express5800インターネットサーバパック」も提供します。業種向け、ECサイト、データセンター、学校向けといった用途を用意してあって、すぐに使えるようになっています。

また、5月から「Linuxサポートセンター」を設置し、販売会社やISP、ユーザーへのサポート体制を強化しました。社内でLinuxに関するノウハウを集めていた「Linux技術センター」もそこに統合しました。

以前、マルチCPU環境でLinuxがうまく動かないことを発見して、NECソフトウェア神戸で、Multiple IO-APICをサポートするカーネルパッチを作成したことがあります。それはコミュニティに提供し、開発版カーネルに取り込まれましたので、カーネルソースに神戸NESの文字が載っているそうです。

Linuxに対応したアプリケーションソ

ソフトウェアも多数提供しています。新聞の組み版機能を実現した「NEPCELL-UX for Linux」は、X上で動作するクライアントソフトで、2社から受注しています。

Linuxでは最近クラスタが流行っていますが、「CLUSTERPRO for Linux」は、NT用のクラスタソフトを移植したものです。他のLinux用クラスタソフトはロードバランシング中心で、フェイルオーバーに関してはサポートが弱い。そのため、最大16台のフェイルオーバーと最大128台のロードバランスクラスタを実現するCLUSTERPROは、ISPの人たちなどから期待されています。また、アプリケーションのフェイルオーバーにも対応します。

今後について

クライアント向けLinuxに関しては？

川井：現在のLinuxでは、使えるオフィスソフトがないので当面はダメだと思っています。Windowsで十分だし、困っていない人たちは、Linuxに移行しないでしょう。苦労してまでLinuxを使わせようとは考えていません。

いわゆるオフコン分野はどうでしょうか？

川井：以前のアプリケーションは、みんなNTへ行ってしまいました。パッケージソフトをNTに移植したら、最初はそうでもなかったけれど、だんだんNTの標準のAPIやGUIの機能を使うようになってしまい、ほかへはいけなくなってきてしまいました。まあ、それがMSの作戦でもあるのですが、NTで安定して使っているものをLinuxへ持っていくかという、そうはならないでしょう。ユーザーにとっては、OSは関係なくてそのアプリケーションがキチン

と動けばよいのだから。

NECのLinuxマーケットのシェア目標はどれくらいでしょうか？

川井：統計が取りにくいのでわかりにくいのですが、いまは10%くらいでしょう。現在のPCサーバでのマーケットシェアは3割以上ありますので、それに匹敵するくらいは取りたいと思っています。特にサーバ分野に力を入れていきます。

現在のライバルはどこでしょうか？

川井：99年にハードウェアで一番売れたのはコバルトでしょう。IBMもLinuxに力を入れていますよね。しかし、SIベースでは今でも国内No.1だと思っています。

ディストリビューションがたくさんありますが、サポートはたいへんではないですか？

川井：ディストリビューションとのアライアンスをどうするかという方針は非常にクリアです。「日本法人があってキチンとサポートできること」と「サポート契約を結んでビジネスtoビジネスのやりとりができること」が大前提です。

それが今可能なのは、レッドハットとターボリナックスだけなのですが、PCサーバ分野では、ユーザーもほとんどその2社の製品を使っていますし、十分でしょう。

Linux技術者育成が求められていますが、どう思われますか？

川井：Linuxの専門知識がないと、SIできないかという、そんなことはないと思います。当社では、UNIXのエンジニアがたくさんいるので困っていません。

確かに、1人でサーバを立ててファイアウォールを構築できるような、なんでもできる人は不足していると思いますが、全体のインテグレーションや部

分部分をとってみると、NTでもあまり変わらないでしょう。

そんなに技術者がいなくて困るほど、マーケットが広がっていないと思います。もっと商談を増やさないといけません。

そのために、どういう方針があるのでしょうか？

川井：2つのアプローチがあって、ひとつは、「NECがLinuxをちゃんとやっている」ことをアピールすることです。それに関しては、広告ではなく展示会などでLinuxに興味を持っている人に情報を流していきます。

もうひとつは、Linuxと言わなくても「Linuxを使うようにしてしまう」ことです。たとえば、先ほどのExpress 5800サーバはLinuxとは、ほとんど言っていないませんが、インターネットサーバとか用途に合わせたパックを用意していますので、そういうお客様に「ばんばん売れる」ということです。

(聞き手：編集部 木下)



Express 5800 インターネットサーバパック

Distribution

新着ディストリビューション

TurboLinux Workstation 日本語版 6.0 Limited Edition

廉価な日本語デスクトップ環境を。4月に発売されたTurboLinux Workstation 日本語版 6.0 に付属するバンドルソフトをシェイプアップし、低価格を実現した Limited Edition が新たにリリースされた。果たしてコストパフォーマンスはいかに？

Debian GNU/Linux 2.2

RPMってなんですか？ やっと出た新しいDebianの収録パッケージ数は約4000。これらのパッケージを有効に使うため、新しくaptとdebconfが標準で採用された。ユーザーにインストールメディアを意識させないaptはLinuxユーザーを墮落の道へと誘う一歩進んだパッケージ管理ツールである。今回はこのaptツール群を中心に新しいDebianへアプローチしてみた。

LASER5 Linux 6.5 Secure Server Edition

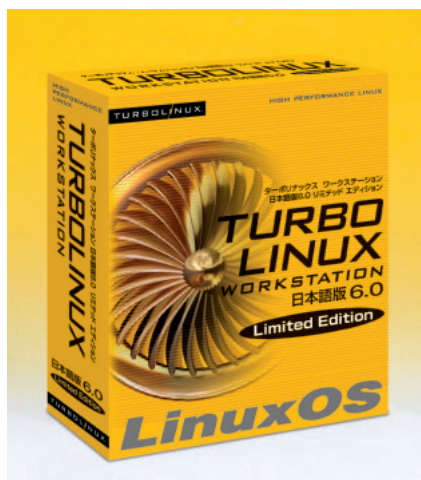
Xなんてステ。インターネットやイントラネットなど、各種サーバ用に開発されたLASER5 Linux 6.5 Secure Server Editionは、セキュリティホールになり得るものをすべて消し去っている。デスクトップ環境では定番のXも例外でないその一徹さ、ちょっとだけお手並みを拝見させていただこう。

TurboLinux Workstation日本語版6.0 Limited Edition

TurboLinux Workstation日本語版6.0 Limited Edition (以下、TurboLinux LE)は、ターボリナックスジャパンが新しく発売したディストリビューションだ。インストールCD、Netscape用各種プラグインなどを含むコンパニオンCD、アップデートパッケージを含むアップデートCDなど4枚のCD-ROMと、ユーザーリファレンスガイドなどのドキュメントが収録され、10月6日から店頭およびオンラインでの販売が始まっている。

Limited Editionとは？

TurboLinux LEは、4月に発売されたTurboLinux Workstation日本語版6.0 (以下、TurboLinux WS) にバンドルされていた日本語入力プログラム のATOK12 SE、日英翻訳ソフトの翻訳魂とVMware Expressが付属しな



製品名 TurboLinux Workstation日本語版6.0 Limited Edition
価格 6800円 (税別)
問い合わせ先 ターボリナックス ジャパン株式会社
 03-5766-1660
<http://www.turbolinux.co.jp/>

い点が異なり、そのぶん価格が6800円と低めに設定されている。商用日本語入力プログラムとしてはWnn6がバンドルされるので、日本語デスクトップ環境として使うには、さほど問題ないといえよう。

また、次世代Webテクノロジーとして期待されるXMLの開発エンジンiPEX 2.0が、ほかのTurboLinuxラインナップにさきがけバンドルされることも、TurboLinux LEの特徴である。

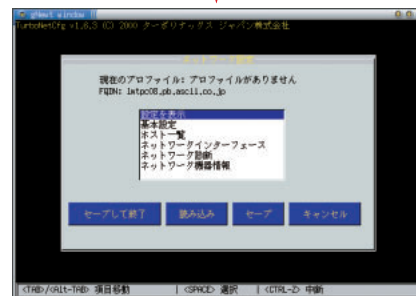
TurboLinux LEの基本システムはTurboLinux WSと同じで、カーネルにバージョン2.2.13が、Cライブラリにglibc2.1.2が、XにXF86 3.3.6が採用されている。

なお、同梱されるアップデートCDに、カーネル2.2.16やglibc2.1.3などとセキュリティを修正した最新パッケージが含まれており、付属スクリプトを使って簡単にアップデート可能だ。

使いやすい独自設定ツール

Linuxシステムは、基本的に/etcディレクトリや各ユーザーのホームディレクトリに配置されたテキストファイルで設定され、これらの設定ファイルはUNIX系OSに不慣れなユーザーにとっての鬼門になっている。

そこで、TurboLinuxは設定ファイルに不案内なユーザーのために、GUIで操作できる独自の設定ツールを装備している。この設定ツールはコマンド



画面1 TurboLinux独自の設定ツール
 TurboLinux独自の設定ツールは、TurboCentroで一括管理される。

ラインやGNOMEの下段にあるアイコンでTurboCentroを起動し、そこから各種設定メニューを呼び出して使う仕組みになっている (画面1)。

このほかにも表1のような商用ソフトがバンドルされている。廉価にオフィスでの使用にも十分耐える日本語環境が提供されるTurboLinux LEは、デスクトップ用途にLinuxを試してみたい入門者へとくにお勧めのディストリビューションである。

本誌付録 CD-ROM Disk1に TurboLinux Workstation 日本語版6.0 (FTP版) 0915を収録しています。非商用のソフトだけが含まれますので、本文の表1に掲載されている商用ソフトは含まれていません。また、ターボリナックス ジャパンのサポートも受けることはできません。

System Commander Lite	マルチブート用のアプリケーション
RYOBI日本語TrueTypeフォント5書体	美しい日本語フォント
Wnn6 Ver3	UNIX系OSで定番の日本語入力システム

表1 バンドルされる商用アプリケーションのリスト
 TurboLinux LEは、付属する商用アプリケーションの数をおさえることで手頃な価格を実現している。

Debian GNU/Linux 2.2

膨大なパッケージ数を誇るDebian GNU/Linux (以下Debian) の新バージョン2.2がリリースされた。前バージョン2.1のリリースから約1年半経過しており、「やっと出たか」という感である。

Debian 2.2は、基本システムとしてカーネルにバージョン2.2.17、Cライブラリにglibc2.1.3、XにXFree86 3.3.6を採用し、今回から新たにPowerPCとARMアーキテクチャに対応したため、以前からサポートされているIntel、Alpha、SPARC、Motorola 68kとあわせて、合計6つのプラットフォームで利用可能となった。

収録パッケージはさらに増えてその数約4000、これらのバイナリパッケージは、ひとつのアーキテクチャにつき3枚のCD-ROMに収録される量である。

Debianってなんなのさ？

Debianをひとことで言うと「Red Hat LinuxベースでないLinuxディス

トリビューション」だ。現在多くのLinuxディストリビューションは、RPM (Red Hat Package Manager) というパッケージ管理方式を採用している。RPMを採用するディストリビューションの目印は、.rpmという拡張子のついたファイルである。

一方のDebianは、.debという拡張子のDebianパッケージを、rpmコマンドに相当するdpkgコマンドや、dpkgのフロントエンドであるapt (画面1)、dselect (画面2) といったツールを使って管理する。これらのツールを使ったパッケージ管理は非常に強力で、Debianが「一度インストールしてしまえば再インストール不要」とまで言われるゆえんである。

また、以前のバージョンには、日本語環境構築のためにDebian JPというパッケージ集が存在したが、今回のバージョン2.2からそれらの日本語パッケージは本家Debianに統合され、ユーザーはDebian JPパッケージなのかどう

かを意識することなくパッケージ管理ができるようになった。

aptを使って墮落する

これまでのDebianは、dselect (画面2) というツールを使ってパッケージのインストールやアンインストールを行ってきた。

しかし、このdselectは直感的でないインターフェイスのためか、使いづらいというユーザーが多く、「Debianは難しい」というイメージを広める大きな要因となっていた。

これを改善するため、新バージョンの2.2では、標準のパッケージ管理ツールとして、dselectに代わりにapt (画面1、3、4) が採用された。ユーザーはaptを使う前に、あらかじめapt-setup (画面3) を使って、

- FTPサイト
- HTTPサイト

```

kterm
lmtpc08:~# apt-cache search --names-only apache
libapache-session-perl - Perl modules for keeping persistent user data across ht
tp requests.
libapache-mod-ruby - Embedding Ruby in the Apache web server
libapache-dbilogger-perl - Tracks what's being transferred in a DBI database
apache-ssl - Versatile, high-performance HTTP server with SSL support
libapache-filter-perl - perl Apache::Filter - Alter the output of previous handl
ers.
apache-common - Support files for all Apache webserver
libapache-mod-dtcl - Allows the use of Tcl as a server parsed language, similar
to PHP.
libapache-asp-perl - perl Apache::ASP - Active Server Pages for Apache with mod_
perl.
libapache-mod-ssl - Strong cryptography for Apache
apache - Versatile, high-performance HTTP server
apache-perl - Versatile, high-performance HTTP server with added Perl support
libapache-mod-auth-pam - Authenticate web access using PAM
libapache-mod-ssl-doc - Documentation for Apache module mod_ssl
libapache-ssi-perl - perl Apache::SSI - Implement Server Side Includes in Perl.
libapache-mod-perl - Integration of perl with the Apache web server
libapache-dbi-perl - Connect apache server to database via perl's DBI
apache-dev - Apache webserver development kit
apache-doc - Apache webserver docs
lmtpc08:~#

```

画面1 aptツール群のひとつapt-cache

Debianには約4000という膨大なパッケージが存在する。その中から使いたいパッケージを探すのは大変な作業だが、このapt-cacheを使うと簡単にパッケージを検索できる。画面はWebサーバApacheを検索しているところ。

```

kterm
Debian GNU/Linux: 'dselect' package handling frontend.
* 0. [A]ccess      Choose the access method to use.
1. [U]pdate      Update list of available packages, if possible.
2. [S]elect      Request which packages you want on your system.
3. [I]nstall     Install and upgrade wanted packages.
4. [C]onfig     Configure any packages that are unconfigured.
5. [R]emove     Remove unwanted software.
6. [Q]uit       Quit dselect.

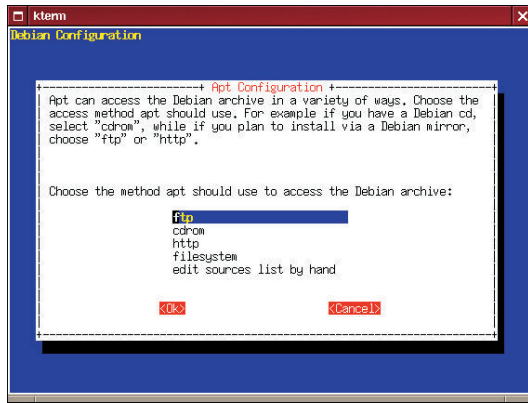
Move around with ^P and ^N, cursor keys, initial letters, or digits;
Press <enter> to confirm selection. ^L redraws screen.

Version 1.6.14 (1386). Copyright (C) 1994-1996 Ian Jackson. This is
free software; see the GNU General Public Licence version 2 or later for
copying conditions. There is NO warranty. See dselect --licence for details.

```

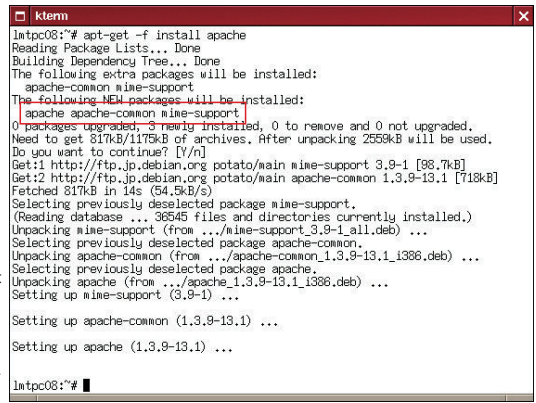
画面2 悪名高きパッケージ管理ツールdselect

前バージョンまではこのdselectが標準のパッケージ管理ツールであったが、今回からその座をaptにゆずった。Debian 2.2でもdselectは利用可能だ。



画面3 aptの設定ファイルを生成するapt-setup
aptを利用するには、パッケージ収録場所の設定が必要だ。aptの設定ファイルは手書きでもよいが、このapt-setupを使えば簡単に設定できる。

画面4 パッケージを簡単にインストールできるapt-get
aptのバックエンドツールapt-getを使って、WebサーバApacheをインストールしているところ。apt-getはApacheだけでなく、Apacheに必要なパッケージも一緒にインストールしてくれる。



- CD-ROM
- ハードディスク

をパッケージ収録場所として登録する。パッケージの収録場所は複数指定でき、FTPなどネットワークを利用したのパッケージ管理は、ファイアウォール内でも可能だ。

ひととおりaptの設定を終えると、ユーザーはとても簡単にアプリケーションのインストールなどが行えるようになる。たとえば、WebサーバApacheのインストールは、root権限で、

```
# apt-get -f install apache
```

とするだけだ(画面4)。この際aptは、Apacheの使用に必要なパッケージも一緒にインストールしてくれるので、ユーザーはアプリケーションのインストール時に、わずらわしい依存関係で悩まされることがない。「4000もパッケージがあると、探すだけでひと苦労で

はないか!」というユーザーは、
apt-cache search --names-only
apache (実際は1行)

などとして、必要なパッケージを探せばよい(画面1)。



debconfを利用してさらに墮落する

debconfは、aptとともにDebian 2.2から標準で採用されたテクノロジーである。今までのDebianでは、パッケージをインストールする際、個別に基本設定に関する質問に答える必要があった。特にシステム全体をインストールする際は、この方式だとユーザーが質問に答えるためにずっと席を離れられないという事態になる。そこで、debconfであらかじめ質問に対するおおよっぱな回答を設定しておけば、ユーザー不在でもパッケージをインストールできるというわけだ。

ただ、約4000あるすべてのDebianパ

ッケージがdebconfに対応しているわけではないので、debconfの恩恵をフルに受けるには、今しばらくの時間がかかりそうである。



Debianベースのディストリビューション

Debian 2.2のインストーラは、前述のaptやdebconfの採用により、インストールするパッケージ群をおおざっぱに選択できるので、以前と比べてインストール作業は若干だが簡単になっている。とはいうものの、GUIを採用するRed Hat Linuxベースのディストリビューションのインストーラと比べて見劣りする感があるのは否めない。さしいわいDebianをベースに開発されたディストリビューションがいくつかあるので、簡単にDebianをインストールしたいユーザーは、それらのディストリビューション利用するのもよいだろう(表1)。

	ベースのDebian	URL	概要
Omoikane GNU/Linux	2.2	http://www.omoikane.co.jp/	日本のベンダーが開発し本家Debianとほぼ完全な互換性を有する
Storm Linux	2.1	http://www.stormix.com/	英語版がメインだが、今秋日本語版がリリースされる予定
Corel LINUX	2.1	http://linux.corel.com/	インストール直後からWindowsドメインに参加可能
プロサーバLinux	2.2	http://www.fujifilm.co.jp/fmd/linux/lintop.html	サーバ用途に特化され、独自ツールを使った集中管理が可能
Debian GNU/Linux	-	http://www.debian.org/	本家のDebian

表1 Debianベースのディストリビューション

上記はDebianをベースにしたLinuxディストリビューションである。OmoikaneとプロサーバはDebian 2.2をベースにしているが、ベースのDebianはfrozen potatoというリリース前のものである。

LASER5 Linux 6.5 Secure Server Edition

LASER5 Linux 6.5 Secure Server Edition(以下、LASER5 Secure Server)は、インターネット、イントラネットをはじめ各種サーバ用途に特化されたディストリビューションである。

開発はレーザーファイブとLinuxのSIベンダー鹿嶋コンピュータサービスが共同で行い、9月1日から店頭販売が始まっている。

LASER5 Secure ServerはRed Hat Linuxをベースにしており、カーネルにバージョン2.2.14、Cライブラリにglibc2.1.2を基本システムとして採用している。

この製品には、インストール用CD-ROM、サーバソフトの設定ファイルリファレンス、サーバ用の商用アプリケーションの体験版などが同梱され、ユーザーは180日間で3件までのサポートと、年4回のセキュリティアップデートサービスを受けられる。

必要なものは始めから

LASER5 Secure Serverには、WebサーバApacheやメール転送プログラムのSendmailなどのほかに、標準でメーリングリストサーバFMLや、全文検索エンジンNamazu、Kakasiといったツールが収録されている。メーリングリ

ストの全文検索エンジン作成を考慮したようなパッケージ構成は、インターネットサーバ用をうたうだけあって非常にユニークである。

また、収録されるApacheには標準でSSL、Perl、PHPのモジュールが適用されており、データベースと連携させたeコマース用Webサイトの構築も可能だ。

さらにテキストベースのLASER5 Secure Serverのインストーラは、構築するサーバタイプ(画面1)や、インストールするサーバ用アプリケーションの選択が可能で、マシンのリソースを圧迫し、セキュリティホールにもなり得るX Windows Systemはインストールされず、デフォルトの状態では不要なデーモンプログラムが起動しないように設定されている。



充実のサポート

LASER5 Secure Serverは、SendmailやBINDをはじめとするサーバアプリケーション(表1)の設定、運用までのサポートが提供されるのも大きな特徴である。

これらの運用は、ネットワークに関する豊富な経験や、多くの知識が必要とされものであり、長年にわたり

Apache	多くのサイトで採用されているオープンソースWebサーバ
BIND	ドメイン管理に必須のDNSサーバ
FML	高速さが特徴のメーリングリストサーバ
IMAP	ユーザーの居場所を選ばないメール管理サーバ
PHP	Webサイトで動的コンテンツを提供するためのソフト
Qpopper	広く普及しているPOPサーバ
Samba	Windows、Linux混在環境で使われるファイル/プリントサーバ
Sendmail	定番メール配送サーバ
Wu-ftpd	公開FTPサイトなどで広く使われているFTPサーバ

表1 設定、運用のサポート対象になるサーバ用アプリケーション

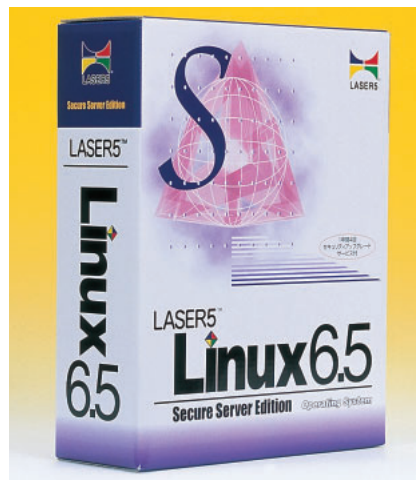
ネットワークサーバの運用は一筋縄ではいかないが、LASER5 Secure Serverではこれらの運用についてもサポート対象になる。



画面1 サーバ用に工夫されたインストーラ
ユーザーはサーバタイプとサーバアプリケーションを選択してインストールする。作業中に迷う箇所はほとんどない。

Linuxの開発にかかわってきたレーザーファイブが、その技術に自信を持っていることの現れだろう。

LASER5 Secure Serverは定価が4万9800円とかなり高価だが、SIベンダーによるサポート料がさらに高額なことを考えると、低コストで自前のサーバを構築したいユーザーには魅力的な選択肢といえよう。



製品面 LASER5 Linux 6.5 Secure Server Edition
価格 4万9800円(税別)
問い合わせ先 レーザーファイブ株式会社
03-5818-6626
http://www.laser5.co.jp/

Distribution ▶▶▶

緊急速報 Red Hat Linux 7J (日本語版)、Red Hat Linux 7 (英語版) リリース

レッドハットは、9月26日に「Red Hat Linux 7J」のリリースを発表した。今回発売される製品パッケージは2種類で、「Official Red Hat Linux 7J Deluxe」は、OS用3枚、ドキュメント、ワークステーション向け商用アプリケーション、Power Tools、IBMソフトウェア（評価版）が各1枚の合計7枚のCD-ROMが付属し、1万2800円。10月6日より発売される。

「Official Red Hat Linux 7J Professional」は、同Deluxeに、サーバ向け商用アプリケーションCDと、CPAN (Perlスクリプト) CDが追加され、2万9800円。こちらは10月13日より発売される。

Red Hat Linux 7Jは、カーネル2.2.16、glibc 2.1.92を採用しているが、近日リリース予定のカーネル2.4へのアップグレードも容易になっている。X Windows Systemには、最新のXFree86 4.0.1を採用し、従来のXFree86 3.3.6よりグラフィックス表示のパフォーマンスを向上させている。また、OpenGL互換のMesa 3Dグラフィックスライブラリも採用している。デスクトップ環境にはGNOME 1.2.1 + Sawfish 0.30とKDE 1.1.2

を搭載し、ユーザーがカスタマイズ可能になっている。

新機能として、Webサーバでの128ビット暗号化通信をサポートするOpenSSL、MySQLデータベース、GUIによるカーネル/ファイアウォール・コンフィギュレーションツール、グラフィックスサポートツールの追加、USBマウス/キーボードのサポートが行われた。

また、gccコンパイラの最新バージョン2.96によって、C、C++、Objective-C、Fortran 77、CHILL、Javaなどをサポートしているなど、開発者向けの機能も拡張されている。

なお、米Red Hatから9月25日に英語版の「Red Hat Linux 7」がリリースされている。

レッドハット株式会社 (<http://www.redhat.com/jp/>)



NECのクラスタソフトをバンドルしたTurboLinux CLUSTERPRO Server 6 発売

ターボリナックス ジャパンは、クラスタシステムの新製品「TurboLinux CLUSTERPRO Server 6」を10月4日より発売する。この製品は「TurboLinux Server日本語版6.1」に、NECのクラスタミドルウェア「CLUSTERPRO for Linux」をバンドルしたもので、可用性・信頼性の高いサーバシステムの構築が可能となっている。価格は2ノード80万円から。

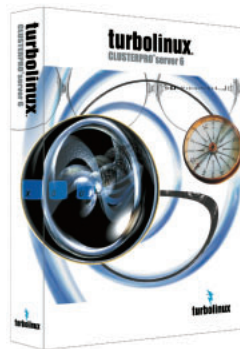
TurboLinux CLUSTERPRO Server 6は、クラスタを構成するCLUSTERPROサーバと、GUIによってクラスタを管理できるCLUSTERPROマネージャの2つ機能を備えている。CLUSTERPROサーバ同士は共有ディスクを介して相互に接続し、ノードの障害を検出すると、自動的に正常なサーバでデータと処理を引き継いで行うフェイルオーバー機能を提供する。

障害時や各種資源の引き継ぎなどの復旧処理は、スクリプトによりカスタマイズが可能だ。

また、複数のマシンでサーバの負荷分散を行うロードバランシング機能を持っており、フェイルオーバーと組み合わせることもできる。

フェイルオーバークラスタでは最大16ノード、ロードバランシングクラスタで最大128ノードまでサポートしている。

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



TurboLinux Server日本語版6.1メンテナンスキット #1 を無償提供開始

ターボリナックス ジャパンは、10月11日より「TurboLinux Server日本語版6.1メンテナンスキット #1」(以下メンテナンスキット #1) を無償提供する。これは、今年6月に発売されたサーバ用ディストリビューション「TurboLinux Server日本語版6.1」の登録ユーザーに対して、年4回行われるものの第1回目で、CD-ROM3枚組で提供される。

メンテナンスキット #1には、各種アップデートモジュール、LVM (Logical Volume Manager) やkparamといったサーバを拡張する新機能ツール、サーバ向け商用アプリケーションと評価版ソフトウェアが収録されている。

アップデートモジュールは、カーネル2.2.16、glibc 2.1.3、wu-ftpd 2.6.05、samba 2.0.7-4jaJPなどで、セキュリティ対策

が講じられている。

LVMを利用すると、複数のディスクにまたがるパーティションを、ひとつのパーティションとして管理でき、パーティションサイズの伸長も可能だ。kparamは、共有メモリのサイズやカーネルパニック時の動作などの、カーネルパラメータの設定を行うツールで、Oracle8iなどのデータベースに最適なパフォーマンスを得られるように変更できる。

そして、IBM DB2やLotus Notesなどのトライアル版(機能制限なし)、Oracle8i Workgroup Server R8.1.6 for Linuxの120日間トライアル版、HDE Linux Controller 2.0 Expressなどが含まれる。

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

Products

- 40 CPUにUltra SPARC i300MHzを搭載したラックマウントサーバ
XtraNet 1U 300-IW
- 42 ホスティングサービスに最適なWebアプリケーションサーバ
Cobalt RaQ4r

CPUにUltra SPARC i300MHzを搭載したラックマウントサーバ



XtraNet 1U 300-IW

サードパーティから低価格なSPARCサーバが発売された。UNIXサーバ分野のトップブランドであるSunの純正マザーボードを使用し、1Uサイズ(高さ44mm)のラックマウントシャーシへの搭載を実現している。

製品名	XtraNet 1U 300-IW
価格	29万9700円
問い合わせ先	ロジカルイフェクト株式会社 TEL 03-5822-3322 http://www.logicaleffect.com/

ロジカルイフェクトから、SPARC CPUを搭載した低価格なラックマウントサーバ「XtraNet 1Uシリーズ」が発売された。



Sun純正マザーボードを使用

XtraNet 1Uシリーズは、SunからOEMで供給されたマザーボード(写真1)を採用し、Ultra SPARC i300MHzを搭載している。メモリはECC対応のDIMMスロットが4本用意されていて、512Mバイトまで拡張できる。スリムCD-ROMとフロッピー

ドライブを上下にセットし、2台のハードディスクと共に本体前面部分に配置されている(写真2)。

ハードディスクの構成の違いによって3モデル用意されていて、エントリータイプであるIDE標準モデルは、20GバイトのIDEハードディスクを1台内蔵している。

上位モデルとして、IDE標準モデルと同じハードディスクを2台内蔵し、ハードウェアRAIDカードを内蔵したIDE RAID標準モデルと、SCSI標準モデルがあり、SCSIモデルは、9.1GバイトのSCSIハードディスクを1台内

蔵し、Sun純正のUltraWide SCSIカードを装着することで、Sunの製品と高い互換性を得ている。

なお、IDE RAID標準モデルに装着されるRAIDカードは、PCIバスからは電源しか利用しないハードウェアRAIDコントローラで、ドライバソフトを必要としないため、OSを選ぶことなく利用できる。

ケースは、金属製のフロントカバーを装備し、カラーもブラックとオフ・ホワイトの2色を選べるようになっている。オプションのユニバーサルスライドレールを、ケース横面に装

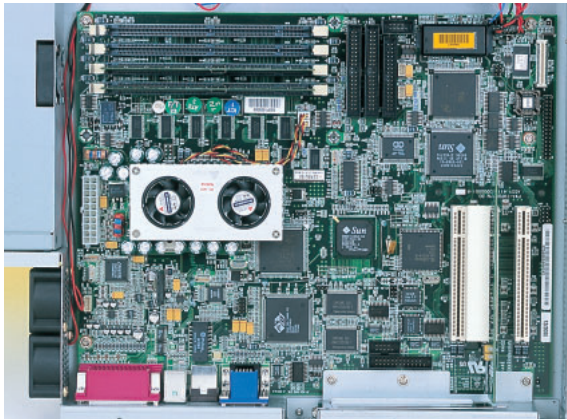


写真1 Sun純正マザーボードを使用
中央のファンが2個付いている部分の下に、Ultra SPARC i300MHzが搭載されている。

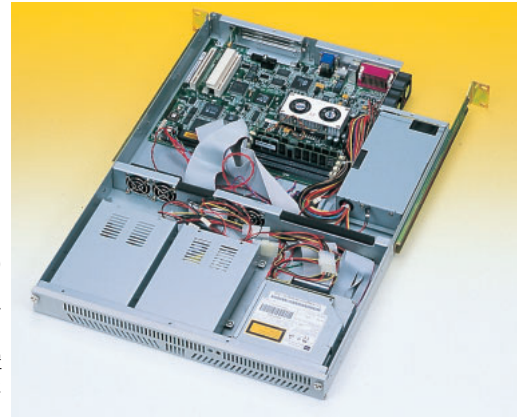


写真2 XtraNet 1U 300の内部
ケースが1Uサイズ（高さ44mm）と超薄型のため、小型のファンを各所に配置して効率よく放熱できるようにしている。

備すれば、ラックにマウントした状態で前面に引き出すことが可能になり、高いメンテナンス性を実現する。

Red Hat Linux/SPARCをインストール

今回はIDE標準モデルのXtraNet 1U 300-IWを試用した。SPARCといえば、OSにはSolarisを連想するところだ。XtraNet 1Uシリーズは、オプションでSolaris 7またはSolaris 8を選ぶことができるが、今回はOSなしを選び、Linuxをインストールしてみた。

64ビットのUltra SPARC用のLinuxは、UltraLinuxとも呼ばれていて、Red Hat、Debian、Caldera、SuSE、Mandrakeといったディストリビューションが対応している。一番入手しやすいのは、Red Hat Linux/SPARC

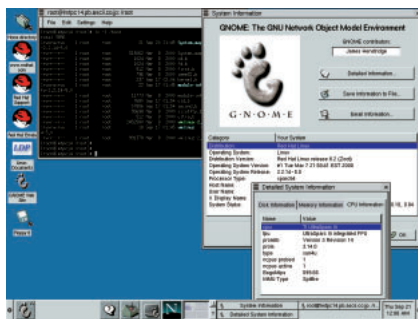
6.2英語版だろう。秋葉原のショップやWeb通販などで購入できる。Vine Linuxが次のバージョン2.1でSPARC版のリリースを予定しているので、今後は日本語環境も整はずだ。

インストールは簡単で、CD-ROMからブートしてインストーラを起動すればいい。ハードディスクのドライブ名やブートロードがLILOではなくSILOであるなどの違いはあるが、Intel版とほぼ同様の手順である。本機に搭載されているグラフィックスチップは、ATIの3D RAGE PROなのでX Window Systemも、XFree86を利用してその通りに設定すればよい。

本機には、ウエスタンデジタルのWD205BAという、ディスクの回転数が7200rpmと高速なハードディスクが使われていた。編集部内のLANに接続し、WebサーバやFTPサーバとして試用してみたところ十分に高速で、特に不満もなく快適に使用できた。

サーバ機なので本来の使い方ではないのだが、GNOMEを起動してみたのが画面1である。ラックにマウントして使う性質上、当然だがキーボードやマウスは標準では付属していない。基本的に運用中はネットワーク経由で管理する使い方がふつうだろう。そのため手持ちのPC用の日本語106キーボードとマウスを利用したのだが、そのせいかCPUがSPARCだということを忘れてしまうぐらい違和感はなかった。

IntelやAMDの1GHzを超えたCPUと比較すると、64ビットとはいえUltra SPARC i300MHzでは、速度的には不利がある。しかし、個人用途とは違い、サーバに一番要求されるのは信頼性である。多くのサーバ機で実績を積んだSunの技術を搭載した本機は、長時間安定した運用が要求されるインターネットサーバなどに向いているだろう。



画面1 Red Hat Linux 6.2/SPARCのGNOMEデスクトップ
Detailed System InformationにUltraSparc iと表示されている。

CPU	Ultra SPARC i300MHz
RAM	128MバイトECC SDRAM (最大256Mバイト)
ハードディスク	20Gバイト7200rpm IDEハードディスク (最大2台)
CD-ROM	24倍速ATAPIスリムCD-ROM
フロッピードライブ	3.5インチ1.44Mバイト
グラフィックス	ATI 3D RAGE PRO (4MバイトSGRAM) オンボード
ネットワーク	10/100BASE-T x 1ポート
インターフェイス	RS-232C x 1、パラレル x 1ポート、PS/2 x 2ポート
PCIスロット	1スロット
サイズ (mm)	425 (W) x 610 (D) x 44 (H)
電源	200W ATX (最大300W)

表1 XtraNet 1U 300-IWの主な仕様

Cobalt RaQ4r



WebブラウザからGUIで操作でき、専門的な知識がなくてもサーバの管理ができるというコンセプトを最初に製品化したCobalt社の、第4世代ISP向けラックmountサーバは、インターネットのオンラインサービスに必要な機能をすべて備えている。

製品名	Cobalt RaQ4r
価格	オープンプライス
問い合わせ先	コバルト・ネットワークス株式会社 TEL 03-3599-0722 http://japan.cobalt.com/

コバルト・ネットワークスから、ラックmountタイプのサーバプラットフォームRaQファミリーの新製品として「RaQ4シリーズ」が発売された。「RaQ4i」と「RaQ4r」の2モデルあり、RaQ4iは、IDEハードディスクを1台、RaQ4rはIDEハードディスクを2台内蔵し、RAID 1（ミラーリング）に対応している。

コンパクトなケースに機能を凝縮

RaQ4はCPUにK6-2 450MHzを採用し、メモリは128M～512Mバイト、15.2G～30GバイトのIDEハードディスクを最大2台まで内蔵可能で、10/100BASE-TXネットワークインターフェイスを、RaQ4iは1ポート、RaQ4rは2ポート備えている。

本体は、1Uサイズ（高さ44mm）のラックmountタイプで、フロッピーやCD-ROMドライブを内蔵しない分、奥行きが短くまとめられている。

また、UPS（無停電電源装置）とシリアルインターフェイスで接続することで、停電時の自動シャットダウンに対応する。UPSに接続したRaQ4と、他のRaQ4がネットワーク経由で通信することで連動してシャットダウンすることも可能だ。

コバルト・ネットワークスの製品は、キーボードやディスプレイを直接接続せずに、PCなどからWebブラウザでアクセスし、すべての操作を行うようになっている。

ただし、最初にネットワークに接続するときには、本体に付いているLCDパネルを見ながら、4方向の矢印

ボタン、S（選択）、E（実行）の合計6個のボタンを操作して、IPアドレスなどの設定を行う（写真2）。なお、再起動やシャットダウンもこのボタンで行えるようになっていて、メンテナンスの時に非常に便利である。

今回試用したRaQ4rは、2台の20GバイトIDEハードディスクを内蔵しており、ソフトウェアRAIDでミラーリングを行っている。2台のハードディスクに同一のデータを書き込むミラーリングなので、利用できるディスク容量は1台の時と同じ20Gバイトだが、万一ハードディスクが故障しても大事なデータを失うことがなく、そしてもう1台のハードディスクによってサービスを継続して行うことが可能だ。

ハードディスクはパーティションで

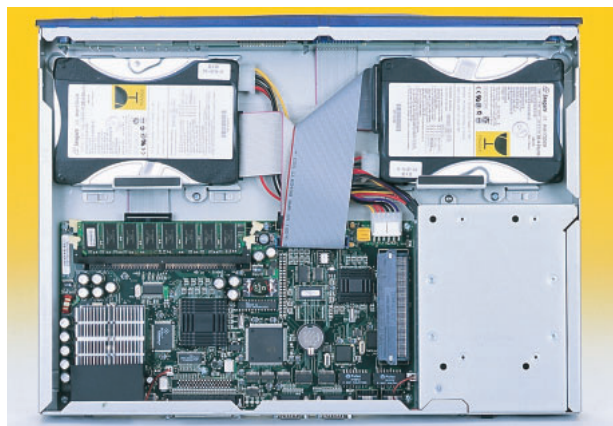


写真1 Cobalt RaQ4rの内部
1Uサイズのケースにハードディスクを2台内蔵しているが、ケーブルやパーツが適切に配置され、スッキリとまとめられている。



写真2 LCDコンソール
16文字×2行のLCDディスプレイと矢印ボタンを使って、ネットワーク設定やRaQ4rのシャットダウンなどの操作を行う。



写真3 背面パネル
USB、外部SCSI、シリアル×2、ネットワーク×2、PCI拡張スロットといったインターフェイスが用意されている。



画面1 サイト管理画面
ユーザーの追加やメンバーリストの管理、仮想サイト、SSLの設定、サイトの使用状況の確認、データのバックアップ/リストアをWebブラウザから設定できる。



画面2 サーバ管理画面
複数台分のWebサーバ機能を1台のRaQ4で実現する、仮想サイト（パッチャルホスト）を定義することもこのように簡単に行える。そのほか、コントロールパネルから各種サービスの設定が可能だ。

区切られており、OSとプログラム、ログ記録用のエリアとして1Gバイト弱が確保されていた。それ以外のエリアがユーザー用で18Gバイト弱だった。

すぐに使えるWebサーバ機能

OSにLinuxを採用し、Apacheを中心としたWebサーバ機能があらかじめ設定してあり、ネットワークに接続して電源を入れるとすぐにサービスを開始することができる。

最初に、WebブラウザでRaQ4サーバに接続すると、セットアップウィザード画面になるので、ホスト名やドメイン名、DNSサーバなどのネットワークと管理者パスワードを設定する。次に、/admin/ディレクトリを指定すると、IDとパスワードを尋ねられるので、adminと管理者パスワードを設定すれば、管理画面になって各種設定ができる（画面1）。

Webサーバの機能を拡張するものとして、CGI（Perl）やPHP 4.0スクリプトのサポートや、Chili! Soft ASPを使ったActive Server Pageの実現、データベースソフトのInterBase 6.0がインストール済みといった特徴があり、ダイナミックWebコンテンツ

を構築することが可能になっている。

また、管理者はtelnetで接続することも可能なので、直接使う必要はほとんどないが、WebブラウザよりもLinuxのコマンドラインでの操作のほうが使いやすい人も満足できるだろう。

パッチャルホストが簡単に設定可能

RaQ4には、Apacheのパッチャルホスト機能を利用して、複数の仮想サイト（IPアドレス）を1台のサーバでホスティングする機能を標準で備えている（画面2）。管理画面のメニューも、サーバ管理とサイト管理の2種類に分けて操作しやすくしている。

仮想サイトごとにユーザーを作成でき、そのユーザーはメールの送受信、Webサーバ、FTPサーバの機能を利用できる。そして、ユーザーご

とに使用可能なディスク容量を制限できる。

RaQ4内部では、仮想サイトごとにディレクトリを作り、仮想サイトへのアクセスがあると、そのディレクトリをホームディレクトリとしてレスポンスするという仕組みだ。

なお、IPアドレスごとに帯域幅の制限を設定できるので、仮想サイト機能と組み合わせると、特定のサイトにアクセスが集中して他のサイトへのレスポンスが落ちることを防ぐことができる。

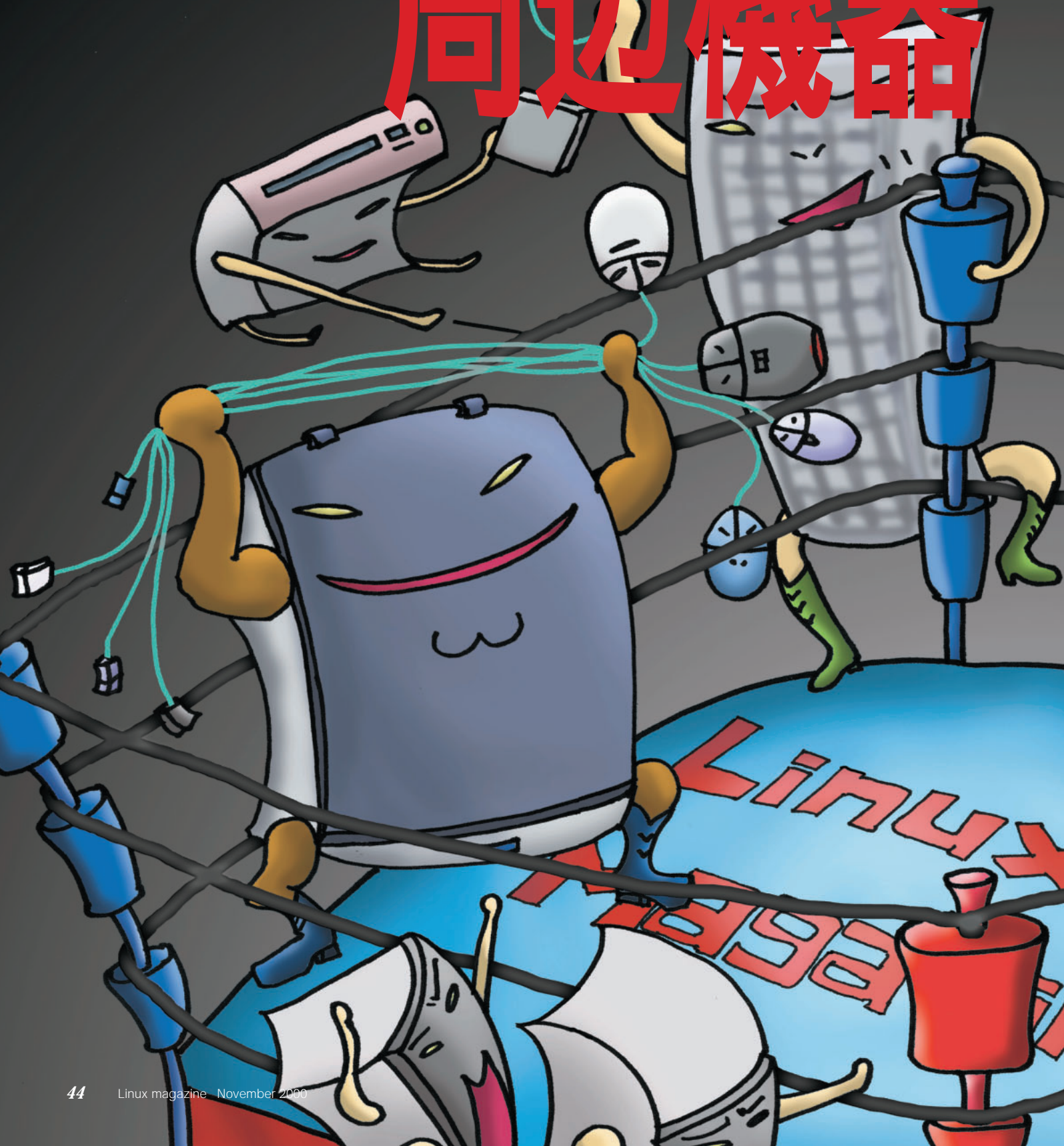
最近では、インターネットのドメイン名を個人で取得している人も多いが、1台のRaQ4で複数のユーザーのドメイン管理を簡単に実現することができる。ISPなどのホスティングサービスにRaQ4を利用すれば、ユーザー管理の煩雑さから解放されるだろう。

CPU	K6-2 450MHz
RAM	256M ~ 512Mバイト
ハードディスク	20Gバイト ~ 30Gバイト (RAID 1)
インターフェイス	RS-232C x 2ポート、USB x 1ポート
SCSIインターフェイス	UltraWide SCSI
PCIスロット	1スロット
ネットワーク	10/100BASE-T x 2ポート
LCDパネル	16文字 x 2行
サイズ (mm)	432 (W) x 318 (D) x 45 (H)
重量	5kg
最大消費電力	60W

表1 Cobalt RaQ4rの主な仕様

動く? 動かない?

周辺機器



バトルロイヤル

マニュアルにWindowsやMacしか載っていないからって、あきらめるのはまだ早い。やってみなけりゃワカラナイ！ やらずに動くワケがない！！

文：山岸典将、のりぞう、編集部
Text : Norimasa Yamagishi , Norizoh , Linux magazine
illustration : Aki



USBとLinuxにまつわるアレコレ

USBの普及で様変わりを見せている周辺機器にまつわる事情。手始めとして、Linuxでの「そこらへん」を探ってみよう。

文：編集部

Text：Linux magazine

インターネット接続にはモデムやTA、画像データの処理にはスキャナ、音楽再生を楽しむにはサウンドカードが必要だ。ゲームをバリバリやりたいならジョイスティックやMIDI音源もほしくなる。コンピュータでやりたいことが増えるほど、増えてくるのが周辺機器。そこで気になるLinuxでのサポート。というワケで、お贈りするのが今回の特集なのである。



企画趣旨説明 (いいわけ)

今回の企画を実現するにあたって担当の頭を悩ましたのが、「いったいどこまで取り上げるのか」ということであった。周辺機器なんてそれこそ星の数ほどあるし、その多くはWindowsやMac OSに対応した製品だ。メーカーがLinuxでの動作を保証していないので、ひとつずつ動作を検証していかなければならない。思いついたはいいいけれど、いきなり山本リダ状態である。

しかしそこはお気楽&フレキシブルが魅力の我が編集部のこと、いきなり

開き直ることにした。決定された方針は「各人が気になってる周辺機器を適当にピックアップすること。今どきUSBモノが中心でしょ」という非常にアバウトなものである。読者のみなさまには、どうか広い心でご了承願いたい。

なお先ほども触れたように、ほとんどの製品は、Linuxに関してメーカーの動作保証はされていない。今回取り上げた機種もそうである。限られた時間内でのテストなので、編集部で太鼓判を押すわけにもいかない。この点も、どうかご了承いただきたい。



USBは覇権を握るのか

周辺機器について語るとき、必ずついてまわるのが接続インターフェイスがらみのお話だ。

周辺機器接続の王道ともいえるSCSIをはじめ、今やAT互換機のデフォルトとなったATAPI、プリンタとモデムにはパラレルとシリアル、カード挿すならPCI、忘れちゃいけないIPS/2とい

った具合に、これまではさまざまな接続インターフェイスとその規格が1つのPCにてんこ盛り状態であった。

こうした状態を回避して、1つのインターフェイスですべての周辺機器を接続しちやおうという意気込みのもとに登場したのがUSB (Universal Serial Bus) である。「ゆにばーさる」の部分にその意気込みが表されているのだ。USBの特徴をまとめると以下のようなになる。

ホストとノードのツリー構造により

127台までの周辺機器が接続可能

最大データ転送速度は12Mbps

ホットプラグ機能

バスからの電源供給

これらに加えて、ハブ、入力デバイス、オーディオなどの「クラス」に周辺機器を分類し、それぞれのクラスの機能を規定することによって、クラスごとに共通のデバイスドライバで動作を制御できるようになっている(実際には機種固有の機能があるため完全な



写真1 USB接続ケーブル

USB規格ではケーブルに方向性があるが、ホスト側とノード側でコネクタ形状が異なるので間違えることはない。



写真2 ホスト側のコネクタ部分(オス)

ホスト側のコネクタ形状は「USB-Aタイプ」と呼ばれる。延長ケーブルなどではオス/メスの違いにも注意。



写真3 ノード側のコネクタ部分(オス)

ノード側のコネクタ形状は「USB-Bタイプ」と呼ばれる。こちらもオス/メスがあるので注意しよう。

共通化はむずかしい)。

フツー127台もつながないとか、12Mbpsじゃ用途に限られるなどの意見もあるだろうが、従来のSCSIやATAPIなどに比べて柔軟なシステム構成が可能になっていることは確かだ。

LinuxのUSBサポート

LinuxにおけるUSBサポートの総本山は「Linux USB Project」である。その成果はカーネル2.4.0に採用される予定で、最新の2.4.0-test8にも組み込まれている。また、2.2系カーネル用のバックポートパッチもあり、プロジェクトのWebサイト (<http://www.linux-usb.org/>) からダウンロードできる。このサイトには『Linux USB Guide』というドキュメントも用意されている。ぜひ目を通しておこう。

USB製品のサポートに関する情報も、このサイトで検索可能だ。もちろんメーカーオフィシャルではないのだが、世界中のLinuxユーザーから送られてきた動作報告をまとめたリストが掲載されている。

USBサポートパッチ

今回のUSBデバイスのテストでは基本的に、2.4.0ではなく2.2系にパッチをあてたカーネルを使用している。これは安定版カーネルで動作させたほうがシステム全体への影響が少ないと考えたからだ。

カーネルパッチは、さきほど紹介したバックポートパッチではなく、より新しいpre-patch-2.2.18-9を使用した。このパッチは、Linuxカーネルの中心的なメンテナーのひとりであるAlan Cox氏がまとめたもの。カーネルがアーカイブされているサイトの/linux/kernel/people/alan以下で見つかるは

ずだ。The Linux Kernel Archives (<http://www.kernel.org/>) か、Ring Server Project (<http://www.ring.gr.jp/>) などのミラーサイトで入手しよう。

と言いつつ、両パッチとも付録CD-ROMに密かに収録してあるので利用してほしい。適用方法は、まずパッチファイルのコピーから。USBプロジェクトのバックポートパッチは/usr/srcに、18pre9は/usr/src/linuxにコピーする。次にパッチをコピーしたディレクトリに移って、以下のコマンドを実行する。<filename>にそれぞれのパッチのファイル名を指定すればOKだ。

```
# zgrep -cd <filename> | patch -p1
```

USBサポートの導入

コンパイルの手順も説明しておこう。実行するコマンドは次のようになる。

```
# cd /usr/src/linux
# make mrproper
# make xconfig
# make dep
# make clean
# make install
# make modules
# make modules_install
```

2行めはカーネルの設定を初期化するコマンドなので、パッチ適用後の初回コンパイル時にのみ行えばよい。

カーネルのUSBサポート設定は3行めの「make xconfig」のところで行う。なお、これはX上で動作するコンフィグレーションツールを使用する場合の指定だ。「make menuconfig」とすればテキスト版のツールが起動する。[USB] メニューの表示は同じで、[Support for USB]と、[USB Controllers] セクションにある3つのUSBホス



写真4 USBハブのポート部分
ハブからも電源は供給されるが、デバイスによってはACアダプタが必要な場合もある。ちなみに、このコネクタ形状がUSB-Aタイプのメス。

トコントローラ用ドライバのうちの1つを選択すれば、USBサポートのコア部分が有効になる。ホストコントローラを選択は、ハードウェア環境に依存するので、システムに搭載されているI/Oコントローラチップの種類をあらかじめマニュアルなどで調べておこう。

今回のテストではモジュール化したケースもあったが、コアのサポート部分はUSBデバイスを使用する場合には常に必要なので、カーネルに組み込んでおくほうがよいだろう。ちなみに、モジュール化した場合のファイル名はusbcore.oとusb-uhci.o (またはuhci.o / usb-ohci) だ。

接続テストを行う場合には、[USB verbose debug messages] を有効にしておく効果的。USBに関するデバッグメッセージが出力されるようになるので、ポートやデバイスの状況を分析するのに役立つ。メッセージは、/var/log/dmesgとmessagesで確認できる。

コンパイルが完了すると、カーネルイメージが/bootにvmlinuzとして置かれる (実際にはvmlinuz-2.x.xのシンボリックリンク)。必要に応じてliloを再設定しよう。コンパイルの詳細については、カーネル付属のドキュメントを参照のこと。

USB HID

まずはキホンの基本から。人間様がPCへ指令を伝える入力デバイスをテストした。PS/2モノは動いてあたりまえ。で、USBはどうよ？

文：にやー@編集部

Text : Nyaa@Linux magazine

キーボード&マウス

HID (Human Interface Device) はUSBのクラスカテゴリのひとつで、キーボード、マウス、ジョイスティックなどの入力デバイスの総称だ。LinuxのUSBサポートでは、hid.oというデバイスドライバとして実装されている。

キーボードとマウスは、AT互換機では長らくPS/2接続のものが主流であったし、それで十分といえはいる。今後、急速にUSB接続にとって代わられることもないだろう。ただ、iMacの例もあることだし、将来的にはPS/2コネクタやシリアルポートを持たないAT互換機（もはやATとは呼べないかも）が登場することも考えられる。「ワシ知らんもんね」というのでは、オペレーティングシステム的にはかなりやばい。というワケでもないんだらうけど、LinuxにおいてもUSBキーボード&マウスのサポートは比較的早くから進められてきた。安定版カーネルでのサポートは2.4.0からになるが、開発版の2.3系や2.2系用のバックポートパッチでは、すでにサポートされている。今回のUSB HID関連のテストは、LASER5

Linux 6.2に2.2.17カーネル + 2.2.18-pre9パッチという環境で行った。



パイプできません

さて設定である。USBキーボードについては、カーネルのコンフィグレーションでHIDとキーボードの項目を有効にすればいいだけ（詳細は表1を参照）。カーネルをコンパイルして必要なドライバモジュールをロードすれば、すぐに使えるようになるはずだ。もちろん、ドライバはカーネルに組み込んでもいい。

コンフィグレーションに関して注意点がひとつ。2.4.0-test8では、2.2.18-pre9と異なり、入力デバイスの設定が [Input Devices] としてUSBとは独立した項目になっている。2.4.0で試す場合は、うっかりチェックを忘れていて動かない、てなことがないように気をつけよう。これはマウス、ジョイスティックにも共通するポイントだ。

設定自体はサクサク進んだのだが、動作テストで問題が発生！「|」と「¥」、「\」と「_」が入力できない

のである。テストに使用したのはLogitechの109キーボード（写真1）。Linux側の設定は106配列になっている。2.4.0-test8でも確認したが、やはりダメ。ほかのキーは全然問題なく入力できるのだが、やはりLinuxでパイプが使えないのは相当イタイ。

Linux側のキーボード設定をus101配列にすれば入力できるかも、と思って試してみたところ、これがうまくいった。しかし、キートップの文字と実際の入力が違うのは、やはり問題だ（101配列にも慣れてないし……）。



マウスは好調！

続いてUSBマウス。キーボードと同様に、まずはカーネルコンフィグレーションで必要項目を設定する。なお、USBマウスのサポートを有効にすると表示される [Horizontal screen resolution] と [Vertical screen resolution] の数値は、マウスドライバで同時にサポートされているデジタイザ用なので特に気にしなくてかまわない。

マウスの場合、さらにひと手間必要となる。デバイスドライバが認識/制御する/dev以下のデバイスファイルが従来のpsaux（PS/2マウスの場合）などと異なるため、新たに作成する必要があるのだ。手順は以下のとおり。

オプション	説明	モジュール名
USB Human Interface Device (HID) support	HICサポートの中核。input.txtによると「デカくて複雑」らしい	hid.o
Keyboard support	USBキーボードのサポート	keybdev.o
Mouse support	USBマウスのサポート	mousedev.o
Joystick support	USBジョイスティック/ゲームパッドのサポート	joydev.o
USB HIDBP Keyboard support	HICドライバの機能限定版。キーボードのみをサポート	usbkbd.o
USB HIDBP Mouse support	HICドライバの機能限定版。マウスのみをサポート	usbmouse.o

表1 USB HID関連のカーネルオプション

```
# mkdir /dev/input
# mknod /dev/input/mouse0 c 13 32
# mknod /dev/input/mice c 13 63
# ln -sf /dev/input/mice /dev/mouse
```


「mouse0」は実際のデバイスに対応するノードで、mouse1 (13 33) 以下、mouse30 (13 62) まで作成できる。「mice」は各デバイスが共有するノード。マウスが接続されていない場合のダミーとしても使われる。マウスを1つだけ接続するのであれば(普通はそうだろう)、miceだけを作成してもかまわない。最後の2行は、XFree86やgpmなどマウスを使うソフトウェアに対応するためのもの。こうしておけば、ソフトウェア側の設定を変更しなくてすむ。



写真1 Logitech iTouch Keyboard
ベーシックなつくりのUSBキーボードだ。写真では見えないが拡張USBポートを2つ備えている。



写真2 ノンブランドUSBマウス
編集部にごろがっていたという以外はナゾ。これが動くんなら何でも動きそうではある。

動作テストはノンブランドのUSBモノにしては地味なマウス(写真2)で行った。やや怪しげなブツだったが、

問題なく動作。キーボードの拡張ポートでも快調に動作したし、USBマウスのサポートは二重丸なのであった。

ジョイスティック(ゲームパッド)

HIDのもうひとつの目玉であるジョイスティックのサポートもテストした。今回使用したのは、GRAVIS製のUSBゲームパッド「GAMEPAD PRO USB」だ(写真3)。そのままなネーミングがナイスセンスであるが、海外のWebサイトでの評価もまずまずで、わりとメジャーな製品であるようだ。



**とりあえず
動きそうな感じ**

テスト前にLinux USB ProjectのWebサイト(<http://www.linux-usb.org/>)で確認したところ、標準のjoydev.oドライバで動作したとの報告を発見。LogitechのWingManシリーズやMicrosoftのSidewinderシリーズなどの動作も報告されていた。USBモノも結構動くのだ。

カーネルコンフィグレーションでは、USBのHIDサポートとジョイスティックサポートをモジュール化するように指定した。マウスの場合と同じくデバイスファイルの作成とリンクの再作成

も必要だ。

```
# mkknod /dev/input/js0 c 13 0
# ln -sf /dev/input/js0 /dev/js0
```

コンパイル完了後、リポートしmodprobeコマンドを使ってモジュールをロードしても、とりあえずは何の変化も起こらない。「ホントに動いてんのか?」といった感じだ。

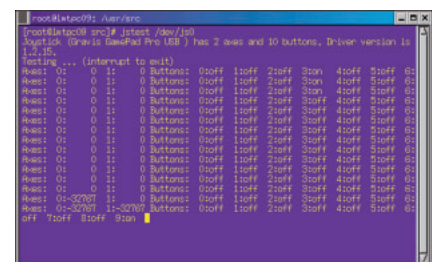
このままではラチがあかないので、動作を確認しよう。動作テストにはjstestコマンドを使う。このコマンドは、The Linux Input Driver Projectが提供している汎用のジョイスティック



写真3 GRAVIS GAMEPAD PRO USB
見てのとおり、某ゲーム機のコントローラにソックリ。名前といい形状といいイラストレータなポリシーがいさぎよし。

クドライバに付属している。LASER5 6.2には標準でインストールされていた。ほかでは、Red Hat 6.2Jにも標準で含まれている。手元にない場合は、RingServer Project (<http://www.ring.gr.jp/>)のWebサイトなどで探してみしてほしい(RPMのパッケージ名は「joystik-1.2.15-2.i386.rpm」)。

テスト方法は簡単で、「jstest /dev/js0」とすればいい。正しく認識されていれば、x軸、y軸、それに各ボタンを示す文字列が1ラインに表示される(画面1)。操作に合わせて、「Axes: 0:32767」とか「Buttons: 0:on」というふうに表示が切り替わればOK。なお、テスト機は問題なく動作した。



画面1 jstestコマンドを実行したところ
1回のボタン操作でオン状態とオフ状態の2ラインが表示されることに注意。

CD-R / RWドライブでオリジナルCD-ROM / CDを作る

CD-R / RWドライブは、メディアのコストパフォーマンスの良さや読み出しに特別なドライブがいらないなど、非常に便利な外部記憶デバイスだ。LinuxでCD-R / RWドライブを使う方法を解説しよう。

文：山岸典将

Text : Norimasa Yamagishi

メルコ CRWI-B1210FBをLinuxで使う

現在普及しているCD-Rドライブは、SCSI接続のものとATAPI (IDE) 接続のものが多い。それ以外にもパラレル接続やUSB接続のものもあるが、こちらに関しては、普及しているとはいえないため、ソフトウェアや関連資料も豊富とはいえない。困難に打ち勝つのが無上の喜びという、チャレンジ精神旺盛な方以外は避けたほうがよいだろう。

今回はATAPI接続のCD-R / RWドライブでCD-R作成に挑戦しよう。テスト機として選んだのは、メルコのCRWI-B1210FBだ (写真1)。CD-R / RWドライブメーカーとして定評のあるPlextorのPX-1210Aをドライブユニットに採用しており、CD-R 12倍速、CD-RW 10倍速、CD-ROM 32倍速のスペックに加え、バッファアンダーラ

ンによる書き込みエラーを防止する「BURN proof」に対応した最新のモデルだ。

なお、CD-Rを10倍速以上で使用するときには専用のメディアが必要となる (当然、LinuxでもWindowsでも変わらない)。メディアを購入するときにはパッケージを確認しよう。



CDのフォーマット

CDのフォーマットにはいくつかの種類がある。Linuxで読み書きできるCD-ROMを焼く (作成する) ことさえできればよいのであれば、フォーマットの違いについてはあまり気にする必要はない。基本的には「ISO9660のRockRidge拡張」で焼けばよい。ただし、作成したCD-ROMをWindowsや

Macintoshでも使いたいとなると、話は変わってくる。

そこで、現在使われている主なフォーマットについて簡単に説明しよう。これらは、今回紹介するソフトウェアですべて作成できる。ただし誌面の都合上、すべてについて詳しく取り上げることはできないので、必要な方はマニュアルを参照して作成に挑戦してみてください。

ISO9660 Level1

- 作成できるサブディレクトリは8階層まで
- ファイル名の長さは8文字 + 拡張子3文字まで (いわゆる8.3形式)
- ファイル名に使えるのはアルファベット大文字、数字、_ (アンダースコア) のみ

ISO9660 Level2

- 作成できるサブディレクトリは8階層まで
- ファイル名の長さは31文字まで
- ファイル名に使えるのはアルファベット大文字、数字、_ (アンダースコア) のみ

ISO9660 RockRidge拡張

- 主にUNIX系OSで使われる
- ISO9660を長いファイル名と、深いディレクトリ構造を扱えるように拡張したもの
- 所有者、グループ、パーミッション、シンボリックリンクといったファイル属性も保存することができる



写真1 メルコ CRWI-B1210FB
ATAPI接続の内蔵モデル。BURN proof対応でバッファアンダーラン知らずだ。

Joliet

- 主にWindowsで使われる
- Windows上で長いファイル名を扱えるようにISO9660を拡張したもの
- ファイル名の長さは64文字まで(非対応OSでは8+3文字で表示される)

HFS

- Macintosh専用
- リソースフォークやアイコンデータなども保存することができる

CD-DA (Compact Disc Digital Audio)

- 一般的な音楽用CD

CD Extra

- 音楽用CDのデータと、CD-ROMデータが両方入っている
- 音楽CDプレーヤで再生したときに、CD-ROMデータが再生されないよう、先に音楽データを書き込む必要がある

て、それを書き込むという方法と、イメージファイルを作らずに、データを参照しながら書いていくオンザフライという方法がある。今回は、イメージファイルを作成してからCD-Rに焼くやり方を紹介しよう。

イメージファイルを作成するときにはCDのフォーマットを指定する。イメージファイルを作ってしまう、フォーマットに関係なく書き込みの手順は同じだ。

LinuxでCD-Rを作成する場合、cdrecordというソフトウェアパッケージを使うことが多い(画面1)。このパッケージには、イメージファイル作成ツールのmkisofs、ハイブリッドイメージの作成ツールmkhybrid、イメージ書き込みツールcdrecord、そして音楽用CDから音を抜き出してWAV形式にするcdda2wavなどのソフトウェアが含まれている。従って、このパッケージをインストールしておけば、とりあえず不自由は感じないはずだ。

cdrecordの一次配布サイトでは、ソースが配布されている。tarボールからのインストールは、「make」「make install」という手順で可能だ。makeする際に「./configure」も自動実行される。

ただし、そのままでは「/opt/schily/」以下のディレクトリにファイルがインストールされる。これを変更するには、「DEFAULTS/Defaults.linux」ファイル中に3カ所ある「/opt/schily」の部分を「/usr/local」などに変更してしてから「make install」すればよい(リスト1)。

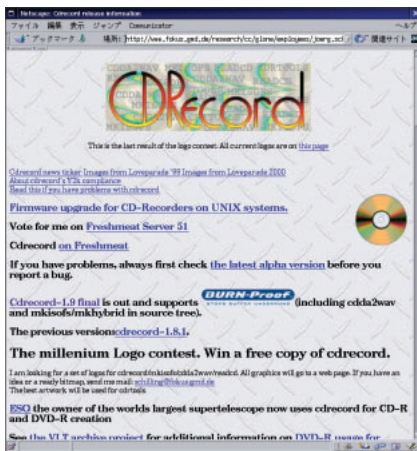
```
$ make
$ su
# make install
```

これで、cdrecord、mkisofs、mkhybrid、cdda2wavといった、今回使うツールのほとんどがインストールされる。



ソフトウェアの準備

CD-Rを焼くには、焼きたいデータをひとまとまりのイメージファイルにし



画面1 cdrecordのWebページ
http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html

リスト1 DEFAULTS/Defaults.linuxを書き換える

```

:
:
DEFCCOM= cc
:
:
CWARNOPTS=

DEFINCDIRS= $(SRCROOT)/include /usr/src/linux/include
LDPATH= -L/usr/local/lib
RUNPATH= -R $(INS_BASE)/lib -R /usr/local/lib -R $(OLIBSDIR)
:
:
INS_BASE= /usr/local
INS_KBASE= /
:
:
DEFUMASK= 002
:
:
DEFINSMODEF= 444
DEFINSMODEX= 755
DEFINSUSR= bin
DEFINSGRP= bin

```



IDEドライブのSCSIエミュレーション

ソフトウェアをインストールできた
ら、CD-Rドライブがcdrecordから使
えることをチェックをしよう。

```
# cdrecord -scanbus
```

画面2のようにCD-Rドライブらしき
デバイスが表示されれば、CD-Rドライ
ブを認識しているの、この節は飛ば
して「イメージファイルを作ろう」か
ら読んでかまわない。SCSIドライブ
であれば、ほとんどの場合表示され
ずだ。しかし、ATAPIドライブの場
合は、このままでは表示されないだ
ろう。

実はcdrecordのコマンド群は、SCSI
ドライブにしか対応していないのだ。
でも、安心してほしい。Linuxには
ATAPIドライブにSCSIのふりをさせ

るエミュレーション機能がある。た
だしこの機能は、標準のインストール
状態では組み込まれていない。

ATAPIドライブのSCSIエミュレ
ーションを行うには、ドライバが必要
になる。ドライバは、カーネル自体に
組み込まれているか、モジュールとし
て用意しておかなければならない。必
要なドライバは表1のとおりだ。

今回使用したRed Hat Linux 6.2J
では、SCSI CD-ROM用のドライバが
用意されていなかったの、カーネルソ
ースをインストールし、sr_mod.oのみ
を作成、インストールした。ただし、
追加が必要なドライバの種類や、以
下を書く組み込み方法については、
環境によって異なってしまう。自分
の環境を調査して、必要なドライバ
を組み込んでほしい。

ドライバの準備ができたら、SCSI
エミュレーションを試してみよう。SCSI

エミュレーションをする場合、起動時
にATAPIドライブとして認識させては
いけない。そのためには起動時にLILO
の画面で「linux hdc = ide-scsi」と
タイプして起動する(106/109日本語キ
ーボードの場合“=”は“^”キーで
入力する)。ただし、デバイス名「hdc」
の部分は、CD-RドライブをIDEインタ
ーフェイスに接続する方法によって異
なるので、表2を参考にしながら自分
の環境に合わせて変更しよう。

起動したら、手動でドライバモジ
ュールを組み込んでいく。ここで組み
込むべきモジュールは、前述のように
環境によって異なる。

```
# insmod sr_mod
```

```
# insmod ide-scsci
```

必要なモジュールを組み込んだら、
再度「cdrecord -scanbus」を実行し
てCD-Rドライブをチェックしよう。

なお、上記の方法は一時的に、SCSI
エミュレーションを使用するための設
定で、再起動したときには再度設定し
直さなければならない。常にSCSIエ
ミュレーションを使用したい場合には、
/etc/lilo.confや/etc/conf.modules(ま
たは/etc/modules.conf)にモジュール
の設定をする必要がある。



イメージファイルを作ろう

システムがドライブを認識したら、
いよいよCD-Rの作成だ。まず最初にや
ることは、元になるデータの準備だ。
適当な名前のディレクトリを作り、そ

接続先	デバイス名
プライマリマスタ	hda
プライマリスレーブ	hdb
セカンダリマスタ	hdc
セカンダリスレーブ	hdd

表2 IDEインターフェイスへの接続方法とデバイス名

```
# cdrecord -scanbus
Cdrecord 1.9 (i686-pc-linux-gnu) Copyright (C) 1995-2000 Jg Schilling
Linux sg driver version: 2.1.36
Using libscg version 'schily-0.1'
scsibus0:
  0,0,0 0) 'PLEXTOR ' 'CD-R PX-W1210A' '1.01' Removable CD-ROM
  0,1,0 1) *
  0,2,0 2) *
  0,3,0 3) *
  0,4,0 4) *
  0,5,0 5) *
  0,6,0 6) *
  0,7,0 7) *
```

画面2 cdrecordがCD-Rドライブを認識した場合の表示

BLOCK devices設定

設定項目	モジュール名
Enhanced IDE/MFM/RLL disk/cdrom...	ide
Include IDE/ATAPI CDROM support	ide-cd
SCSI emulation support	ide-scsci
SCSI設定	
設定項目	モジュール名
SCSI support	scsi_mod
SCSI CD-ROM support	sr_mod
SCSI generic support	sg

表1 SCSIエミュレーションに必要なドライバ

の下にCD-Rに焼きたいファイルやディレクトリを配置しよう。データを置くために作ったディレクトリがCD-Rのルートディレクトリとなる。

キチンとファイルを配置できたら、mkisofsコマンドでイメージファイルを作成する。焼きたいデータを配置したディレクトリがtestで、作るイメージファイルの名前がcdimageなら以下のようにする。

```
# mkdir test
(ファイルを配置する)
# mkisofs -R -o cdimage test
```

これでイメージファイルcdimageが作られる。mkisofsコマンドにはさまざまなオプションがある。比較的よく使われるオプションを表3にまとめた。

RockRidge拡張を指定する場合に、-Rオプションの代わりに-rオプションを指定すれば、すべてのファイルの所有者、グループがrootとなり、どのファイルもすべてのユーザーが読み出せるようになる。データをCD-ROMで配布する場合などに使うとよいだろう。

なお、Joliet拡張を行った場合、Windowsから読むことのできる長いファイル名を付けることができるが、2バイト文字(全角文字)のファイル名はサポートされていないので気を付けてほしい。

また、起動可能なCD-ROMを作る場合は、-bオプションに続けてブートイメージファイルを指定する。ブートイメージファイルは、1.2Mバイト、1.44Mバイト、2.88Mバイトのうち、いずれかのフロッピーディスクイメージでなければならない。

イメージが作成できたら、そのイメージをloopbackデバイスとしてマウントし、確認することもできる。ただし、

loopbackデバイス用のドライバが読み込まれていなければならないことに注意しよう。

```
# mkdir /mnt/testcd
# mount -t iso9660 -r -o loop
cdimage /mnt/testcd
```

このようにすると、/mnt/testcdにイメージファイルがマウントされ、内容を確認できる。確認が済んだら、イメージをアンマウントする。

```
# umount /mnt/testcd
```



イメージをCDに書き込もう

いきなりイメージをCDに書き込んでもかまわないが、安くなったとはいえCD-Rメディアを無駄にするのは避けたい。安全を重視するなら、書き込みのテストをしてみよう。

-dummy オプションを付けて

cdrecordを実行すると、書き込み用のレーザーは照射せずに、実際の書き込みと同じ動作を行う。バッファアンダランが起きるかどうかを確認できるわけだ。オプションの意味については、表4を参照してほしい。

```
# cdrecord -dummy -v -dev=/dev/sga
speed=6 -pad -dao -eject cdimage
```

なお、書き込み先のデバイス名は、「-scanbus」オプションを付けて実行したときに、上から表示される順番に「/dev/sga」「dev/sgb」……になると思えばよい。今回はCD-Rドライブが一番最初に表示されたのでデバイス名は、「/dev/sga」となる。

テストでエラーが出なかったら、今度は「-dummy」オプションを付けずに実行すれば、書き込みを行うことができる。

CD-RドライブがWindowsマシンにしかつなげていなくても、Linuxで作成したイメージファイルをLANなど

オプション	説明
-R	RockRidge拡張を行う
-J	Joliet拡張を行う
-hfs	ISO9660とHFSとのハイブリッドCDを作成する
-o filename	イメージファイル名を指定する
-r	RockRidge拡張を行い、かつuid、gidを0にする。さらに、すべてのユーザーに対し、読み出し可、書き込み不可にする
-a	ファイル名に や#が含まれるものも含める
-d	ピリオドのないファイルにはピリオドを付けない
-L	ピリオドで始まるファイル名を許す
-b filename	ブートイメージファイルを指定して起動可能なCDを作る

表3 mkisofsでよく使われるオプション

オプション	説明
-dummy	書き込みテスト(実際には書き込まない)
-v	処理内容の詳細を表示する
-dev=device	書き込み先のデバイスの指定
-dao	ディスクアットワンス
-pad	データを確実に読めるように最後に空のデータを付け加える
-multi	マルチセッションで書き込む
-speed=number	書き込み速度の指定
-driveropts=burnproof	BURN proof機能を有効にする
-eject	書き込みが終了したらCD-Rを排出する

表4 cdrecordのよく使われるオプション

でWindowsマシンに転送し、CD-R作成ソフトウェアで焼くことが可能だ。この場合、CD-R作成ソフトウェアでイメージファイル(ISOイメージファイルなどと呼ばれることが多い)を読み込んで焼くことになる。



オーディオCDを作ろう

LinuxでもオーディオCD(音楽用CD)を焼くことができる。しかし、ここまでで解説した手順で焼いてはいけない。なぜなら、前述の手順で作成できるのはCD-ROMであり、CD-DAではないからだ(音楽データの入ったCD-ROMを作りたいなら話は別だ)。オーディオCDはCD-DAというフォーマットで記録されており、これはCD-ROMとは違うフォーマットだ。

まず、オーディオCDから、音のデータを抜き出してみよう。データの抜き出しにはcdda2wavコマンドを使う。たとえば、CDの2曲目のデータをWAVフォーマットのデータファイルにするには、以下のようにする。

```
# cdda2wav -Owav -D /dev/sga -t1
```

こうすると、audio.wavというファイルが作成される。CDに含まれるすべての曲をWAVデータにしたいならば、

```
# cdda2wav -Owav -B -D /dev/sga
```

とすれば、audio_01.wav、audio_02.wav、.....というファイルが1曲ずつ作成される。

いくつかのデータを作成したら、今度は焼いてみよう。

```
# cdrecord dev=/dev/sga -speed=6 -audio *.wav
```

これで、そのディレクトリにあるすべてのWAVファイルが音楽として焼かれることになる。

いうまでもないことだが、音楽に限らずデータをコピーする際には、著作権を侵害しないよう十分に気を付けてほしい。



GUIで焼こう

これまではコマンドラインでCD-Rを焼くツールについて説明してきたが、コマンドラインツールをX Window System上でGUI操作するためのフロントエンドもある。これらのフロントエンドを使えば、mkisofsやcdrecordのコマンドラインオプションを意識することなく、イメージファイル作成やCD-Rへの書き込みを行うことができるようになる。

数あるフロントエンドの中から、ここではX-CD-RoastとCDR Toasterを紹介しよう。

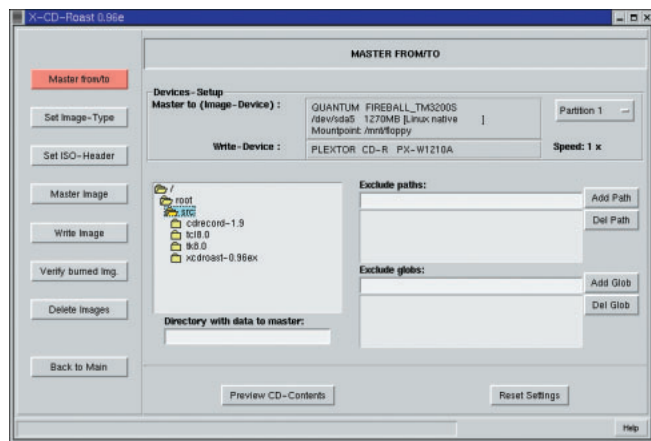
X-CD-Roast

X-CD-RoastはTcl/Tkで作成されたGUIフロントエンドだ(画面4)。インストールには、TCL 8.0p2、Tk 8.0p2、Tix 4.1.0.006が必要だが、これらはすべてX-CD-RoastのWebページで入手できる(画面5)。これらのライブラリなどを事前にインストールした上で、X-CD-Roastをインストールしよう。また、WebページからはRed Hat Linux 6.1用のバイナリRPMファイルもダウンロードできる。

インストールは、次の手順で行う。

```
$ ./configure
$ make
$ su
# make install
```

ところが、Red Hat Linux 6.2JにX-CD-Roastをソースからインストールする場合、この手順ではうまくいかなかった。これは、TCLのライブラリがインストールされているディレクトリ名が、想定されているものと異なっているためだ。「configure」ファイルをエディタで開き、「\$BASEDIR / tcl8.0」



画面4 X-CD-Roastの画面
コマンドラインが苦手な人はX-CD-Roastを使うとよいだろう。



画面5 X-CD-RoastのWebページ
http://www.fh-muenchen.de/home/ze/rz/services/projects/xcdroast/e_overview.html

と記述されている部分をすべて「\$BASEDIR / tcl8.0.jp」に修正したところ、インストールが無事完了した。

インストール作業によって、フロントエンドであるX-CD-Roastだけではなく、mkisofsやcdrecordなどもあわせてインストールされる。

なお、日本語訳されたマニュアルが「<http://www.bekkoame.ne.jp/bero/docj/perl/xcdroast.html>」で公開されている。英語が苦手な方は参考になるだろう。

CDR Toaster

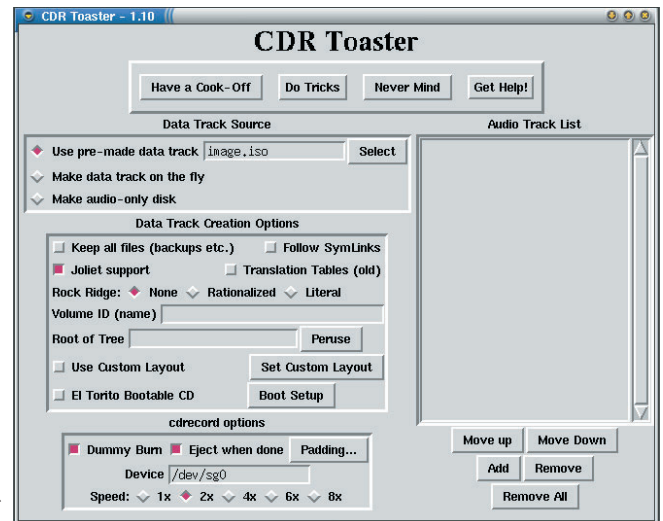
CDR Toasterは、ファイルサイズがおよそ40Kバイトというとても小さなソフトウェアだ(画面6)。実行するにはTkが必要となるが、ほとんどのディ

ストリビューションには含まれているのでそのまま実行できるだろう。原稿執筆時点での最新版は1.10。Webページ(<http://www.jump.net/brooke/cdrtoast/>)からダウンロードしたら、

```
$ gunzip cdrtoaster-1.10.gz
```

```
$ chmod +x cdrtoaster-1.10
```

とすることでインストールは終了だ。もう焼くしかない。



画面6 CDR Toaster
山椒は小粒でぴりりと辛い、
珠玉のフロントエンド。

Column

バッファアンダーランと BURN proof

CD-Rドライブは、PCから送られる書き込みデータをいったんバッファにたくわえ、それを書き込むようにしている。ところが、従来のCD-Rドライブは、何らかの原因により書き込み速度よりもCPUからデータが送られる速度が遅くなるとバッファの中身がなくなってしまう、その時点で書き込みが失敗してしまっていた。この現象をバッファアンダーランという。もったいないことに、書き込みに失敗したメディアはもう使えなくなってしまう。この現象は、低CPUパワーのマシンでWindows 95 / 98を使っている場合によくみられた。Windows 95 / 98に比べマルチタスク性能の高いLinuxでは、バッファアンダーランは起きにくい、CPUパワーの低いマシンでATAPI接続のハードディスクとCD-Rドライブを使っていると、まれに起きることもあるようだ。

バッファアンダーランに対する一番の対策は、単純だが書き込み速度を下げることだ。次は、書き込み時には、CPUやハードディスクに負荷のかかるようなソフトウェアを走ら

せないこと。もちろん、ハードウェアを入れ替えることができるなら、高速なCPUと高速なハードディスクを使うのもよいだろう。数年前までは、CPUに負担がかからないSCSIのハードディスクが良いとされていたが、最近ではATAPIコントローラとハードディスクの性能が向上し、CPUへの負担も小さくなっている、よほど古いマシンでなければ問題はないだろう。

BURN proof機能は、バッファの中身がなくなりそうになると、レーザーの照射をやめて書き込みを停止し、データがたまった位置からデータを書き足していく機能だ。これにより理論上はバッファアンダーランによるエラーは起きなくなった。なお、BURN proofはソフトウェアの対応も必要だが、今回使ったcdrecord-1.9ではすでに対応している。

今回はBurn proof機能のテストを行うために、Pentium 90MHz、Western digitalのかなり古いHIDEドライブCaviar 21600というマシン構成で300Mバイトのイメージファイルを焼いてみた。cdrecordでは -v オプションを付けて実行すると、現在のバッファ状況などをリアルタイムに見ることができる。

このマシン構成でBURN Proof機能を使用せずにテストしたのだが、バッファアンダーランはまったく発生しなかった。うむ、さすがはLinuxだ。と感心している場合ではない。ここではBURN Proofのテストにはならないではないか。そこで、MP3データのエンコードをしながら、同時にファイルの読み書きを続けるという簡単なシェルスクリプトを作成し、CPUとハードディスクの両方に負荷をかけてCD-Rを焼いてみた。すると、さすがに耐えきれず、50Mバイト程度書き込んだ時点でバッファが空になり、書き込みは失敗した。書き込みに失敗して喜ぶのもヘンだが、まずはひと安心だ。

今度は、cdrecordに - driveropts = burnproof オプションを付け、BURN proofを有効にして再度CD-Rを焼いてみると、バッファが空になっていくのに耐えながら書き込みは成功した。実際にBURN proofが必要になるほどLinuxマシンを酷使しながらCD-Rを焼くことがあるのかどうかは、人によりけりだが、安心してCD-Rを焼きたいのであれば、BURN proof機能を持つドライブはお勧めといえるだろう。

リムーバブルメディア

バックアップやデータの交換に、あれば便利なリムーバブルメディア。Linuxでも使いたいよね

文 : Tux Heaven@LM Lab.

Text : Tux Heaven@LM Lab.

ATAPI MOドライブ

容量が数100Mバイトオーダーのリムーバブルメディアは、データのやり取りや、お手軽なバックアップの手段として、以前から利用されてきた。

MO (Magneto Optical) ディスクは、日本国内での人気が高いメディアである。PC-9801シリーズの頃から使われていたため、今でも持っているユーザーが多い。最初のモデルは、容量が128Mバイトだったが、以後230、540 / 650と増えて、最近では“GIGAMO”と呼ばれる1.3Gバイトの製品が登場している。

SCSI接続の製品がほとんどだったMOドライブにも、手軽なATAPI接続の製品が登場しており、SCSIカードを買わずに済む分、お買得といえる。

せっかくATAPI接続のMOを買うなら、Windowsだけでなく、Linuxからも使いたいというのがLinuxerの心情だ。早速「世界の秋葉原」に出撃して、

富士通のMCC3064APという機種を購入してきた(写真1)。650MバイトのMOディスクに対応した製品だ。ATAPI接続なので、もちろん内蔵用のベアドライブだ。



SCSIエミュレーション

まず、MOドライブを取り付けて、PCを起動する。マスター/スレーブなどの設定が間違っていなければ、起動時のメッセージに、MOドライブの名前が見えるはずだ。

新しいデバイスを使うには、まずドライバを手に入れる必要がある。早速、ATAPI接続MOドライブのドライバを探してみると、これがないのだ。せっかく買ってきたのに、Linuxでは使えないの？いきなり企画倒れか？と心配したが、そうではなかった。

Linuxには、SCSIカード用ドライバ

の一種として、ATAPIデバイスをSCSI機器に見せかけて利用するための、「SCSIアダプタエミュレーションドライバ」が用意されている。SCSIインターフェイスは、多彩な機器を接続するために利用されているので、このような仕組みを用意しておけば、ATAPI接続のデバイスも利用できるというわけだ。

エミュレーションドライバは、ロードダブルモジュールとして用意されており、以下のようにしてロードする。

```
# modprobe ide-scsi
```

メッセージが表示されて、その中にデバイスファイル名があるはずだ。筆者の環境では、MOドライブは/dev/sdbという名でアクセスするようになっていた。そこで、

```
# mount -t vfat /dev/sdb /mnt/mo
```

のようにマウントすることで、MOを利用できる。

上記のmountコマンドで、デバイス全体を指定していることからわかるように、MOディスクは、パーティションなしの巨大なフロッピーディスクのように扱われている。ハードディスクのようにfdiskコマンドでパーティションを切ろうとしたり、ext2ファイルシステムを作成しようとしても、エラーが出てしまった。もっとも、データの交換に使うことを考えれば、あえてフォーマットを変更する必要はないだろう。



写真1 MOドライブ
富士通 MCC3064AP。実はこの写真、上下逆さまである。ハードディスクと違って、基板が上にくるのが正しい。ディスクを挿入しようとして、はじめて気がついたのだが、後の祭りだった。

Zipドライブ

Omega社のZipは、アメリカを中心に普及しているリムーバブルメディアだ。初期のメディアは100Mバイトだったが、後に250Mバイトのメディアが追加された。

編集部には、なぜかSCSI接続のZipドライブがある。今まで使われているところを見たこともなかったのだが、周辺機器特集の実験材料(?)として、筆者の手元に届けられた。

筆者の実験用PCは、ハードディスクもCD-ROMドライブもIDEで接続されているので、このZipドライブを使うためには、まずSCSIカードを接続しなければならない。今回利用したのは、少し古めのアダプテックAHA-2940だが、比較的低速なZipドライブなら十分に使える製品だ。

あらかじめSCSIカードを挿した状態で、Linuxのインストールを開始すると、インストーラがAHA-2940を検出し、適切な構成でインストールが行われる。だからといって、SCSIカードを追加するためにインストールのし直しでは、定期的に再インストールするのがお約束の某有名OSと変わりがない。こんなやり方は、もちろん却下だ。



SCSIカードの追加

Linuxerなら、このようにしてみよう。まず、実験用PCには、SCSIカー

ドなしの状態、Vine Linux 2.0をフルインストールした。その後、SCSIカードを挿し、Zipドライブを接続した。

この状態で、SCSIカードを含めた、ほとんどデバイス用のドライバが、ロードブルモジュールとして用意されている。AHA-2940用のドライバは、

```
/lib/modules/2.2.14-1vl6/scsi/
```

内のaic7xxx.oだ。このドライバをロードするには、modprobeコマンドを用いて、

```
# modprobe aic7xxx
```

のようにする。何もエラーが表示されなかったら、lsmodコマンドでロードされているモジュールを確認しよう。リスト1のようにaic7xxxが入っていればOKだ。あとはドライブにZipディスクを入れ、適当なマウントポイント、

たとえば/mnt/zipを作成し、

```
# mount -t vfat /dev/sda4 /mnt/zip
```

とすれば、めでたくZipドライブがLinuxから使えるというわけだ。まず、新品のZipディスクに入っている、README.TXTという名の宣伝を読んで、「うーん、アメリカンテイスト」と感心してみよう。

上述のmountの引数を見ればわかるように、なぜかZipディスクはパーティション番号が4になっている。筆者も研究熱心なLinuxerの一人として、fdiskでパーティション番号の変更を試してみたが、変更した情報を書き込むところでエラーになるようだ。

あまり実用的ではないが、Zipディスクをext2ファイルシステムでフォーマットすることもできる。調子に乗って、Reiserファイルシステムのディスクも作ってみたが、こちらはマウントできなかった。

ZipとMOドライブを追加して、動くことを確認したら、リスト2を参考にしておこう



写真2 Zipドライブ

日立マクセル ZIP-D100SC1。チープな中にも、アメリカンテイストを感じさせるZipドライブだ。日本では、あまり普及しなかったのに、なぜ編集部にあったのだろうか?

リスト1 lsmodの出力例

Module	Size	Used by
ide-scsi	7408	0
vfat	11584	0 (autoclean)
fat	32512	0 (autoclean) [vfat]
eeepro100	12512	1 (autoclean)
aic7xxx	112960	0

リスト2 お勧め設定

```
/dev/sda4 /mnt/zip vfat noauto,user 0 0 — /etc/fstabに追加
/dev/sdb /mnt/mo vfat noauto,user 0 0

modprobe aic7xxx ————— /etc/rc.d/rc.localに追加
modprobe ide-scsi
```

サウンド系デバイス

Linuxerだって音楽は必要です。いろんなデバイス、Linuxで鳴らしてみせましょう

文：にゃー@編集部、Tux Heaven@LM Lab.
Text：Linux magazine

サウンドカード

最近のPCは、低価格化の要求に押されて、多くの機能がチップセットに統合されつつある。また、ベアボーンキットでよくあるような省スペースPCでは、何でもマザーボードに搭載されていて、拡張スロットに何も挿さなくても、ひととおりのおりができてしまうのが一般的だ。

だがその昔、PCは、さまざまな拡張カードを付け加えることで、用途に合った機能を実現するものだった。多くのベンダーが拡張カードをリリースし、シェアを競っていたが、各分野とも事実上の標準と呼べるような製品が存在した。

サウンドカードの分野では、Creative Technology社（以下Creativeと表記）のサウンドブラスター（以下SBと表記）が事実上の標準であった。ほかの製品もたいていは、「SB互換モード」を持っていた。SBにしか対応してないソフ

トもあったためだ。

以前のSBは、設定が大変なことで有名だった。割り込み（IRQ）やI/Oアドレスといったハードウェアの設定を、ほかの拡張カードと衝突しないように、ユーザーが調整する必要があったのだ。SBの名誉のために付け加えると、当時のISAバス用拡張カードはそれが普通で、SBはユーザーが多かったために、うまくできない人もおおぜいいたというのが、本当のところだ。

今どきのサウンドカードは、すべてPCIバス用であり、黙って挿せば設定は自動で行われ、ちゃんと音が出るようになっている。この機能は、OSに依存していないので、LinuxでもWindowsと同様に設定される。まったく良い時代になったものだ。あとは対応するドライバが存在していれば、サウンドカードとして利用可能ということだ。

ドライバについても、多くの製品で、

対応したドライバが作られており、ベンダーがハードウェア情報を開示していない場合や、よほどマイナーな製品以外は、Linuxで利用できるようになっている。



**とりあえず適当に
買ってみる**

以上のような状況は、知識として知ってはいたが、本当のところは、実際に試してみないとわからないのではありませんか？そこで、秋葉原に詳しい編集者に同行してもらって秋葉原に赴き、特にLinuxで動くかどうかチェックせずに、サウンドカードを買ってみることにした。残暑厳しい中を歩き回り、以下の3種類の製品を購入してきた。

Sound BLASTER Live!

Creativeの製品で、前述の昔話にも登場したサウンドブラスターの子孫といったところか。ドライバCD-ROMと拡張カード本体を、静電気防止袋に入れた状態で売っているバルク品を選んだが、7800円もした。いまどきのサウンドカードとしては、とても高価だ。

この製品は、Linuxで動作することが事前にわかっていた。しかし、3枚買ってどれも使えないという、0勝3敗のストレート負けでは、いくらチャレンジ企画でも問題ありなので、一応安全パイとして選択した、ということは公然の秘密だ。

Zoltrix Nightingale

香港に本社があるZoltrix社の製品。

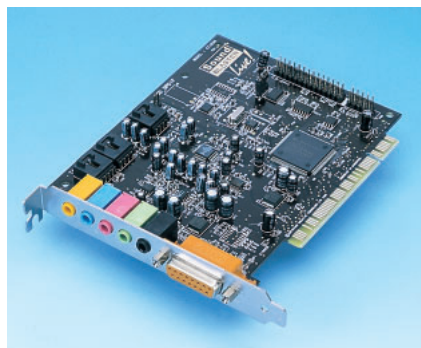


写真1 Sound BLASTER Live!
Sound BLASTERは、今でもサウンドカードの定番と言える。

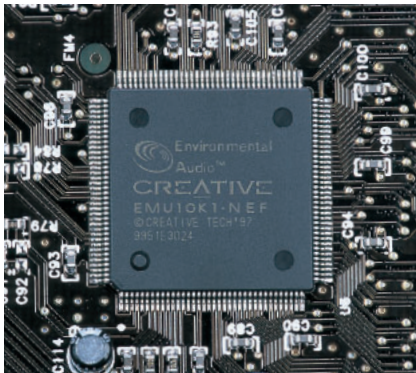


写真2 EMU10K1
Creative内製のサウンドチップだ。

なぜサウンドカードに看護婦さんの名前が? と悩みたくなるが、実はナイチンゲールというのは、和名サヨナキドリという、美しい声で鳴く鳥の名前なのであった。

これもバルクで購入したが、拡張カード本体、ドライバCD-ROM、簡易マニュアルに加えて、SPDIFのオプティカルコネクタと接続ケーブルまで付いていて3580円と、お買得感のある一品だ。

購入後、袋を開けるまでは、使用されているサウンドチップもわからなかったが、冒険も必要だということで、選んでみた。

TepWave 6200

Teppro社の製品。台湾の会社らしいのだが、いまだきの製品にしては珍しく、どこを探してもURLが見つからなかった。しかたがないので、カンで、

<http://www.teppro.com.tw/>

とWebブラウザに入れてみたら、あまりやる気のなさそうなWebサイトが表示された。それほどわけのわからない会社ではなさそうだ。

印刷された紙のパッケージに入って売られていたが、価格は2980円で、今回扱う製品の中では最も安いものだ。

パッケージの裏に、“Yamaha 754 chipset”と書かれており、おそらくヤマハのYMF744の後継品で、たぶん動くけど、ひょっとしたらダメかも? そう思って選択した。



サウンド関連のプロジェクト

Linuxでサウンドカードを利用するためのドライバを作っているプロジェクトで代表的なのが、フリーの

OSS/Free、ALSA(Advanced Linux Sound Architecture)、商用のOSS/Linuxの3つだ。

OSS/Linuxは、4Front Technologies(画面1)という会社が開発しているもので、Caldera OpenLinuxのパッケージ版に採用されている。同社のOSS(Open Sound System)は、Linuxだけでなく、*BSDやSolarisなど、各種UNIX対応版もある。

OSS/Freeのドライバは、カーネルソースにも取り込まれており、Linuxの標準的なサウンドドライバと言えるだろう。

ALSA(画面2)は、現在、とても活発に開発が進んでいるプロジェクトで、多くのサウンドカードに対応した、ドライバを提供している。また、OSS/Freeとの互換性を持たせることも可能で、OSS/Freeを念頭において作成されたアプリケーションからも利用できる。

メジャーなディストリビューションでは、Kondara MNU/Linux、Linux Mandrake、SuSE Linux、そしてTurboLinuxが、ALSAの成果物を取り込んでいる。使いたいサウンドカードが、OSS/Freeは未対応だが、ALSAなら動くことがわかっているなら、これらのディストリビューションを選択するといいたいだろう。

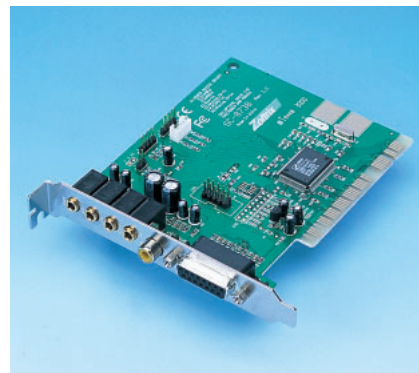


写真3 Zoltrix Nightingale オプティカルコネクタまで付属しているお買得な製品。

実験は、まず標準的なOSS/Freeで試し、ALSAが必要な場合は、手動でインストールことにしたので、Vine Linux 2.0(FTP版)をフルインストールした環境で行った。



Sound BLASTER Live!

Sound BLASTER Live!は、Creative内製のEMU10K1サウンドチップを採用している。このチップは、ALSAでもサポートされているが、Creative自らもドライバを開発し、オープンソースとしてソースをコミュニティに公開している。

業界トップメーカーのオープンソースへの貢献を評価して、今回はこちらを利用する。最新版は、



画面1 4Front Technologies URL <http://www.opensound.com/>



写真4 CMI-8738 C-Media社のサウンドチップ。モデム機能が統合されている。

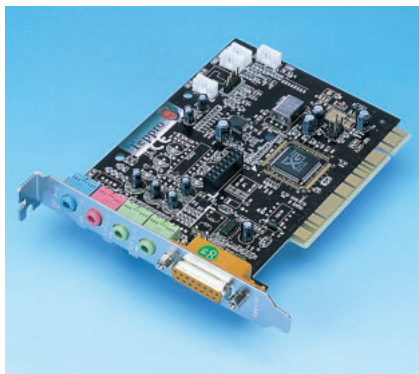
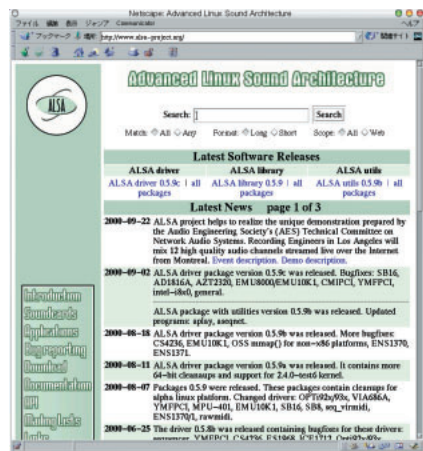


写真5 TepWave 6200
安価な製品だが、一般的な使用には十分だ。

<http://opensource.creative.com/>

から取得できる。最新版のtarボールをダウンロード・解凍して、Vine Linux 2.0に含まれていたソースと比較してみたが、特に大きな変更や機能の追加はなく、カーネル2.4に対応した修正が施されているくらいで、ほとんど違いはなかった。通常は、ディストリビューションに付属しているソースやモジュールを、そのまま使用しても問題はないだろう。

Red Hat系ディストリビューションには、sndconfigというコマンドが用意されており、対応しているサウンドカードなら、機種を判別して適切な設定を行ってくれる(画面3)。自動判別できなかった場合でも、リストから選択



画面2 ALSAプロジェクト
URL <http://www.alsa-project.org/>



写真6 YMF-754
定番チップYMF-744の後継機種。

すればOK(画面4)だ。

Sound BLASTER Live!は、もちろん正しく認識され、/etc/conf.modulesに、

```
alias sound emu10k1
```

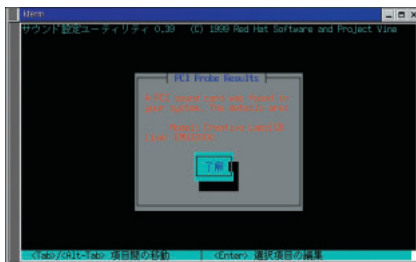
という行が追加された。マシンを再起動したところ、emu10k1.oとOSS/Freeのメインモジュールであるsoundcore.oがロードされた。XMMSでMP3ファイルを再生してみたら、きちんとスピーカーから音が流れてきた。どうやら緒戦は、勝てたようだ。



Zoltrix Nightingale

Zoltrix Nightingaleには、“CM18738”と記されたサウンドチップが載っていた。編集部内には、このチップを知っている人はいなかった。おいおい、本当にこのサウンドカード使えるのか？

ちょっと心配になってきたが、あら



画面3 sndconfig
対応しているサウンドカードなら自動設定が可能

ためて調べたところ、このチップは、台湾のC-Media社が出しているCMI-8738という製品だとわかった。サウンド系の機能だけでなく、ソフトウェアモデムのインターフェイスまで内蔵している、欲ばりなチップである。その多機能さゆえに、面積が限られているベアボーンPCに搭載されていることが多いそうだ。

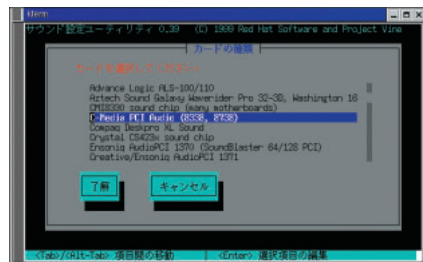
OSS/Freeでサポートされているかどうか確認するため、下記のWebサイトを参照したところ、

<http://www.linux.org.uk/OSS/>

に“CMI-8338/8378”という記述があった。もしかして書き間違い？念のため、カーネルソースのディレクトリ内を探したところ、linux/drivers/soundにcmpci.cというファイルがあり、その中に“CM8738”という記述もあったので、とりあえずOSS/Freeで動かしてみることにした。

Sound BLASTER Live!と同じように、sndconfigコマンドで設定すると、“C-Media CM8738”と認識されており、テスト用の音声ファイルも正しく再生された。

しかし、XMMSでMP3ファイルの再生をさせると、なぜか途中でストップしてしまうという現象が起きてしまった。まったく音が出ないならあきらめもつくだが、途中までは鳴るといのがよくわからない。



画面4 sndconfig
自動設定できなかったときは、リストから選択しよう

実験マシンは、ちょっと古めの Aladdin 5 というチップセットを用いたマザーボードに、K6-2 350MHz を搭載したものだ。ひょっとしたら相性問題なのかと考えて、440BX + Celeron という王道の組み合わせのマシンでも確認したが、状況は変わらなかった。

Zoltrix のボードが特別な作りになっているのか、それともサウンドチップがマイナーチェンジしたのか理由は不明だが、OSS/Free では Nightingale を正しく動作させられなかった。だが、あきらめるのは早い。ALSA があるさ(教育的指導!)。ALSA の設定については、次に紹介する TepWave 6200 のところであわせて説明しよう。



TepWave 6200

TepWave 6200 は、ヤマハの YMF754 というサウンドチップを採用している。型番から想像できるように、おそらく YMF744 の後継機種である。YMF744 は、ALSA でサポートされているので、YMF754 もドライバが流用できるのではないかと甘い期待を抱いて、ALSA プロジェクトの Web サイトを見にいったら、すでに YMF754 にも対応していた。おそるべし、ALSA。

OSS/Free の対応状況だが、Vine Linux 2.0 に含まれているソースには、YMF744 / 754 用ドライバは見つからなかった。しかし、テスト版のカーネル 2.4.0-test8 には、サウンドブラスター互換デバイスとして YMF744 / 754 などを利用するドライバが含まれているので、カーネル 2.4 環境が一般的になれば、OSS/Free でも利用可能になるだろう。サウンドブラスター互換モードでも、BGM を聞くといった用途には十分である。

ALSA ドライバを手に入れる

では早速、ALSA 関係のドライバなどを手に入れよう。

<ftp://ftp.alsa-project.org/pub>

上記のプロジェクトの FTP サイトから、“alsa-driver-0.x.x.tar.bz2”、“alsa-lib-0.x.x.tar.bz2”、“alsa-utils-0.x.x.tar.bz2”、“alsaconf-0.x.x.tar.gz” をダウンロードする。“0.x.x” の部分は、バージョンを表しており、記事作成時の最新バージョンは、リスト1を参照してほしい。

インストールの手順は、アーカイブファイルの解凍とコンパイル、デバイスファイルの作成、そして conf.modules の作成だ。リスト1の順番で行う(行頭の# はプロンプト)。

conf.modules の作成

alsaconf が正しく設定してくれれば、conf.modules を自力で書く必要はないが、うまくいかなかった場合には、自分で作成しなければならない。ドキュメントを見ると、非常に多くの設定項目があるが、ほとんどの項目はデフォルトの値でかまわないので、conf.modules に書かなくても大丈夫だ。必要なのは、ドライバとデバイスファイルの対応などを記した alias 文だ。

リスト2は、ALSA を使う場合の一般的な conf.modules のサンプルだ。もし、alsaconf がうまく働かなかった場合は、リスト2の内容を conf.modules に書き足して、/etc/rc.d/rc.local の末尾に、

```
modprobe snd-card-0
```

の行を書き加えておけばよい。再起動後は、サウンドカードが利用できるようになっているはずだ。

OSS/Free でうまく動作しなかった Nightingale も、ALSA のドライバを使用すると、問題なく利用できた。

リスト1 ALSA のインストール

```
# tar xIf alsa-driver-0.5.9c.tar.bz2
# tar xIf alsa-lib-0.5.9.tar.bz2
# tar xIf alsa-utils-0.5.9b.tar.bz2
# tar xzf alsaconf-0.4.3b.tar.gz
# cd alsa-driver-0.5.9c

# ./configure
# make install
# cd ../alsa-lib-0.5.9
# ./configure
# make install
# cd ../alsa-utils-0.5.9b
# ./configure
# make install

# ./snddevices

# cd ../alsaconf-0.4.3b
# ./alsaconf
```

リスト2 conf.modules の例

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

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

YMH-754 の場合

CMI-8738 の場合

Rio 500

アスキー本社の3階にあるLinux magazine編集部は、ほかの編集部としくられていないため、いつでも賑やかだ。というか、はっきり言うてうるさいのだ(一番うるさいのは、おまえたちだ! と、編集部から苦情が来そうだが)。

そのため仕事が佳境に入ると、集中するために、各自ヘッドフォンを着用することになる。PCのCD-ROMドライブを使っている編集者もいるが、筆者はダイヤモンド・マルチメディアのMP3プレーヤ、Rio 500(写真1)を使っている。写真を見れば分かるように、トランスルーセント(半透明)おたく好みの製品だ。オンボードで64Mバイトのメモリを搭載し、スマートメディアで32Mバイト(ファームウェアを最新版に更新すれば64Mバイト)までのメモリを追加できる。

Windows 98とMac OSに対応したユーティリティが付属しており、PCやMacとはUSBポートを介して接続し、MP3ファイルをやりとりするようにな



写真7 ダイアモンド Rio 500
トランスルーセントがイカサシリコンオーディオプレーヤだ。

っている。今のところ、Windows 2000では動作しないようだ。まして、LinuxのUSB対応は始まったばかりなので、LinuxマシンでRio 500を使うなんて考えもしなかった。

ところが、今回の特集の下調べのため、カーネル2.4.0-test8のアーカイブを展開し、含まれているファイルを調べていたら、なんとlinux/drivers/usb以下に“rio500.c”というファイルがあるではないか! キーボードやマウス、USBハブといった需要の多いデバイスに続くサポートリストに、Rio 500が入っているというのも、何か唐突な気がする。だがRio 500ユーザーとしては、ドライバの作者であるCesar Miquel氏に感謝しつつ、この幸運を享受するだけだ(ラッキー)。



転送ソフトはあるの?

だが喜ぶのはまだ早い。ドライバがあって、Rio 500がLinuxカーネルからなんらかのUSBデバイスとして見えたとしても、それだけでは実用にならない。MP3ファイルの転送や、Rio 500内のメモリのフォーマットを行うユーティリティが必要だ。

ぬか喜びだったかと思っていたら、ほかの編集者から、

“rio500.sourceforge.net”がある

というタレコミがあった。それによく見たら、linux/Documentation/usb以下に、その名もズバリrio.txtというドキュメントがあり、rio500.sourceforge.netも紹介されていた。ド

キュメントの重要性を再認識させられてしまった。

sourceforge.netは、オープンソースの開発者に対して、Web/FTPサーバなどを提供してくれるサイトだ。米VA Linux Systems社によって運営されている。Linux magazineの人気コーナーである、Free Application Showcaseでも“なんとか.sourceforge.net/”というURLをよく見るはずだ。

rio500.sourceforge.netは、Rio 500用ドライバと同じく、Cesar Miquel氏がメンテナンスをしている。Webサイトの一番上に、“*nix support”とうたっていることから分かるように、ここで公開されているRio 500用ユーティリティは、Linuxだけでなく、FreeBSDをはじめとする*BSD系のOSでも動作する。またx86 PCだけでなく、PowerPCやAlpha用のLinuxにも対応しているようだ。



試すしかないでしょ

マウスやキーボードと違って、Rio 500は、誰もが持っているデバイスではない。だけどカーネルソースのアーカイブにドライバが含まれているくらいだから、「きっと世界的に見ればメジャーな製品なんだ、うん、そうに違いない、これはもう試すしかない!」と個人的に盛り上がっていたら、またまたほかの編集者から忠告があった。それは、

このユーティリティで、Rio 500が再起不能になった例があるらしい

というものだった。再起不能? そりゃまずいだろ。

今回の特集で取り上げているほかのデバイスは、編集部で買ったものだから

ら、たとえ壊したところで、個人的にふところは痛まない。でもこのRio 500は私物だ。万が一壊れたら、物笑いのネタになったうえに、約2万7000円の損失だ。

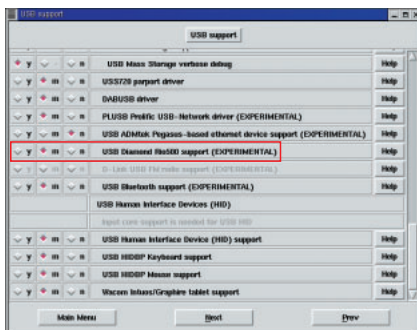
半日ほど悩んだが、Rioと言え、ブラジルの元首都リオデジャネイロ、ブラジルと言え、ラテン系、ラテン系なら細かいことに悩まず、とりあえず試してみるしかないという結論に達し、チャレンジしてみることにした。



ちゃんと動く
じゃないか

ベースにしたシステムは、Vine Linux 2.0をフルインストール後、カーネルを2.4.0-test8に、modutilsを2.3.16にアップデートしたものだ。カーネルをコンパイルする際には、もちろんUSBサポートやRio 500サポートなどを「あり」にしておく(画面1)。USB関連のモジュールは、modprobeコマンドなどで必要なときに組み込むようにしてもいいが、今回はusbcore、usb-uhciといった基本的なモジュールは最初からカーネルに組み込んでおいた。

準備ができたなら、早速Webサイトから必要なソフトをダウンロードだ。tarボールと、x86用のバイナリRPMが提供されている。今回のシステムでは、バイナリRPMからのインストールでも



画面1 カーネルコンパイル時の設定
USBサポート、Rio 500サポートは、yまたはmにしておく。

問題なく動作した。

インストールされるのは、ユーティリティプログラム(表1)、ドキュメント、開発用のライブラリとヘッダファイル、そしてフォントだ。MP3ファイルを転送する際に、フォントファイルをあわせて指定することで、Rio 500のディスプレイ上のフォントを変更することができる。

カーネルが適切に作成されていれば、あとはデバイスファイルを作って、必要なモジュールをロードするだけだ。suコマンドでroot権限を得て、

```
# mknod /dev/usb/rio500 c 180 64
# chmod 666 /dev/usb/rio500
```

としてデバイスファイルを作成し、一般ユーザーがRio 500を利用できるようにしておく。そして、

```
# modprobe rio500
```

とすれば、Okだ。付属ケーブルでRio 500をLinuxマシンに接続し、おそろおそろ、

```
# rio_stat
```

と入力してみよう。リスト1のようにRio 500内のMP3ファイル名が表示されれば、成功だ。あとはWindows版と同じように、フォルダを作って、MP3ファイルを転送すればいい。

なお、GTK+を用いたGUIフロントエンドも存在するようだが、現状では使い勝手に問題ありだったので、紹介は見送った。興味があればWebサイトをのぞいてみよう。

Windows 2000よりも先に、LinuxでRio 500が使えるようになるなんて、ちょっといい話だ。

rio_add_directory	ディレクトリ内のMP3ファイルをすべてRio 500に転送する
rio_add_folder	Rio 500のメモリ内にフォルダを作成する
rio_add_song	Rio 500にMP3ファイルを転送する
rio_del_song	Rio 500内のファイル/フォルダを削除する
rio_font_info	フォント情報を取得する
rio_format	Rio 500のメモリを初期化する
rio_get_song	Rio 500からPCにMP3ファイルを転送する
rio_stat	Rio 500内のMP3ファイルのリストなどを表示する

表1 インストールされるコマンド一覧

リスト3 rio_statコマンドで動作を確認する

```
# rio_stat
Your Rio500 has firmware revision 2.12
Card 0 reports 21 Mb free (22085632 bytes) out of 64 Mb (67108864 bytes).
Command 0x57 returned:
first_free_block = 0x00000aae
sl = 0x00000544

-----
N  offset  num songs  Folder Name
-----
( 0) 0x0017 (10 items)  hekiru

(num) offset      size      song name
( 0) 0x0019 ( 3126752 bytes) rightbesideyou_01.mp3
( 1) 0x00d8 ( 4269869 bytes) rightbesideyou_02.mp3
:
:
```

USBスピーカ

デジタル伝送で高音質！ 事前の調査で判明したUSBスピーカのアドバンテージは、機種選定の段階で脆くも崩れ去ったのであった。その顛末をここで語ろう。



USB接続のメリット

USBオーディオ製品のテストということで、まず思いついたのがスピーカだった。というか、それしか思い浮かばなかったのである。しかも、ふつうスピーカは一度設置してしまえば、あまり動かすものでもないし、特にUSB接続にするメリットもないと思っていた(ホットプラグの甲斐がないってこと)。ところが今回のテストのために調査を行ったところ、次のようなメリットが浮かび上がったのである。

USBスピーカはエライ - その1
USBインターフェイスを介して直接デジタルデータを送出し、スピーカ側のD/Aコンバータでアナログ変換するので、パソコン内部のノイズの影響を軽

減できる。高音質な再生が可能に！

USBスピーカはエライ - その2
「その1」と同じ理由で、PC側のD/Aコンバータ、つまりはサウンドカードが不要になる。拡張スロットが1つ空く！

USBスピーカはエライ - その3
USBオーディオクラスの規格に準拠した汎用ドライバで動作するので、機種を替えた場合でも、ドライバの再インストールが不要。楽チンだ！

「その1」についてももう少し詳しく考えてみよう。PC側のライン出力に接続してスピーカを使う場合、PCで再生しているデジタルデータをサウンドカードまたはオンボードのサウンドチップでD/A変換する。PC内部には、CPUファンやケースのファン、そして電源装置などノイズ源となりそうなものがたくさんある。まず、ここでノイズを拾う可能性があるのだ。

また、アナログの信号を伝えるため、そのインターフェイスであるライン出力のコネクタ、スピーカケーブルでも

ノイズを拾ってくる可能性がある。USB接続の場合、どちらの問題も回避されるため高品質なサウンドの再生が可能になる。もちろんスピーカに内蔵されているD/Aコンバータや、スピーカそのものの品質が高くなければ意味はないのだが.....。



安物買いのってヤツ？

この懸念されていた問題に、今回のテストでモロにぶつかってしまった。秋葉原への買い物につき合わなかったのが悪かったのだ。買い物担当は変なものマニアで、名の知れたメーカーと怪しげなバルク品が同じ値段であれば、間違いなくバルクのほうを選ぶ男なのであった。その担当がお買い上げになったのが、写真8のスピーカ。Sung Forn社製「TURANDO DS-102」、2480円也である。なんでこんな安いのにしたのかと文句を言うと、「いや、USBオーディオのドライバの出来を試すなら、少々怪しいくらいの製品のほうがいいんですヨ」などとノタマウのであった。むむっ、確かに一理ある。

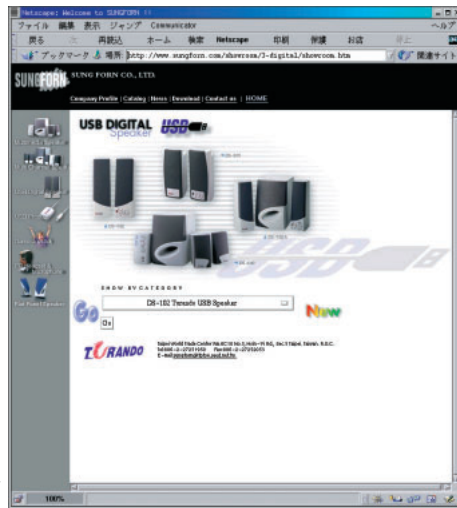
見かけ上はスリムなデザインでなかなかイカす感じであるが、カバーを取ってスピーカ部分を確認してみると、



写真8 Sung Forn TURANDO DS-102

直線的なフォルムのシャープなデザイン。これで音が良ければ言うことなし。

画面6 Sun FornのWebサイト
安価でそれなりに優れた製品を出すというも立派なことである。いろいろ文句言ってしまうせん。ちなみに画面中央にあるのがDS-600。





画面7 USBスピーカ接続時のGMIXイコライザの設定は結構面倒なので、トーンコントロールを備えているのはうれしいところ。

画面8 オンボードサウンド有効時のGMIX PCM以外に、CD、マイク、ラインの入出力などのコントロールがある。ヘッドフォンでは、こちらのほうが音質がよかったのが悲しかった。



コーンの材質などはやはり値段相応なものであった。一応、製造元である Sung Forn の名誉のために言っておこう。TURANDO シリーズの USB スピーカには、DS-301 と DS-600 という上位機種が用意されている（同社の Web サイトで発見）。実際に手にしていないので不明だが、DS-102 よりは、きっといい製品であるに違いない（と思うぞ）。

前置きが長くなった。Linux での USB オーディオの設定に話を進めよう。テストは、LASER5 Linux 6.2 にカーネル 2.2.17 を導入し、2.2.18 の pre9 パッチを適用した環境で行っている。



鳴れば合格ってことで

USB サポートを有効にするためにカーネルを再構築しなければならないが、設定自体はそれほどむずかしくはない。カーネルコンフィグレーションでは、まず [Sound] メニューの [Sound card support] を有効にする。このチェックを忘れると [USB Support] メニューの [USB Audio Support] が有効にならない。もちろん [USB Audio Support] は有効にしておく。モジュール化した場合、それぞれ soundcore.o と audio.o というモジュールがコンパイルされる。

USB の基本モジュール、usbcore と usb-uhci をロードしたところで、メッセージが表示された。/var/log/

messages を確認すると、「DS-102 は HID クラスのデバイスです」と言い切っている。幸先の悪いスタートである。大丈夫か？

メッセージを気にしつつ、audio.o をロードする。今度はコンソールにメッセージが出力された。「アウトプット用のオーディオストリーミングインターフェイスが1つあります」ということなので認識はされているようだ。しかし、「エンドポイントがありません」というメッセージも同時に出力されているのが気になる。

不安を残しつつも、mpg123 で MP3 データを再生してみる。「ゴゴツ」というやたらとデカイ音が一瞬だけして、あとは無音状態になった。ソフトウェアのほうは、再生を続けているようだ。本当に正しくオーディオとして認識されているのか不安になったので、X を起動してオーディオミキサー（GMIX）を起動してみると、[USB Audio Class Driver] と表示されている（画面7）。PCM のボリュームだけしかないのは寂しいが、USB オーディオではデジタルデータをそのまま送出すので、アナログのラインやマイクなどの制御は必要ないのだ。

どうやら大丈夫そうなので、今度は xmms を使って MP3 データを再生する。またもや「ゴゴツ」とデカイ音がして再生が止まった。ボリュームが最大になっていたの、調整してもう一度...

...、鳴ったー！ 鳴りました。音質は予想通り、あまり良くないけど、ちゃんと鳴っているではないか。確認したところ、トーンコントロールも機能しているようだ。再生が終わると自動的にスタンドバイモードに切り替わる機能も働いている。

音が出なかったケースから変更したのはボリュームのみなので、これが原因としか考えられない。ボリューム調整後にもう一度 mpg123 で再生すると、今度は音が出たのだ。さっきは、あまりにデカイ音にビビったのか？ 確かに人間様にも快適な音ではないので、まずボリュームを調整してから再生するよう心がけることにしよう。

その後、継続して行ったテストの結果、大量データのコピーなどシステムの負荷が高くなると、音跳びが起こることがわかった。また、ノイズがやたらと発生することがあったので、気になって別のマシンで試してみた。同じデータを同じく xmms で再生したにもかかわらず、こちらはノイズがしない。PC の USB ポートの問題であるようだ。

夢のデジタルハイクオリティサウンドとはいかなかったが、とにかく音が出たということで、まずはめでたしといったところか。本音を言えば、YAMAHA の YST-MS55D とか Roland の MA-150U とかのハイグレードな機種でテストしたかったなあ。

イメージスキャナで画像を取り込む

スキャナの価格も下がり、個人でも気軽に購入できるようになってきた。そこで、Linuxでスキャナを使うにはどうすればよいのか実験した。

文：山岸典将
Text: Norimasa Yamagishi

エプソン GT-8700をLinuxで使うには

スキャナにはSCSI接続、USB接続、パラレル接続のものがあり、最近ではUSB接続のスキャナが増えている。しかし、現在の安定版カーネルである2.2系統ではUSBを公式にサポートしていない。LinuxでUSBを使用するには、開発版のカーネル2.3、テスト版の2.4、あるいは2.2にUSBサポートパッチをあてたものが必要になる。

今回は、エプソンのGT-8700 (写真1)を使い、まずSCSI接続に挑戦し、その後にUSB接続を試すことにする。



数多くのスキャナに対応するSANE

Linuxでスキャナを使うためのソフトウェアは多いが、今回は最も多くの機種に対応しているSANE (Scanner Access Now Easy) を使うことにする。SANEは、GPLに従って配布されているフリーソフトウェアで、スキャ

ナ以外にもデジタルカメラ、ビデオカメラなどを扱うことができる (表1)。さらに、別のマシンにつないであるスキャナをネットワーク経由で利用することもできる。接続インターフェイスも、SCSIだけでなくUSBやパラレル接続にも対応している。詳しくは公式Webページを参照してほしい (画面1)。

なお、SANEはLinux専用ではなく、FreeBSD、Solarisなど、多くのプラットフォームで動作する。Linuxでも、Intel x86系のほか68000系、PowerPC、Alpha、SPARCなどにも対応している。

SANEの大きな特徴は、プログラムのドライバとユーザーインターフェイス部分が完全に独立していることだ。

一般に、ハードウェアに関するソフトウェアは、ハードウェアを制御するためのドライバ部分と、ユーザーが操作するためのユーザーインターフェイス部分の2つから構成される。SANE

は、この2つが独立しており、規格化されたAPI (Application Program Interface) で結ばれて動作する (図1)。なお、SANEではドライバ部分のことを「バックエンド」、ユーザーインターフェイス部分のことを「フロントエンド」と呼んでいる。

このようなしくみで動作するため、新しいスキャナに対応させるにも、SANEのAPIに従ったバックエンドを開発するだけでよい。APIに従っていれば、既存のフロントエンドから操作が可能だからだ。逆に新たなフロントエンドを作りたい場合も、各スキャナ固有の部分はバックエンドが吸収してくれるため、スキャナの仕様については考える必要がない。



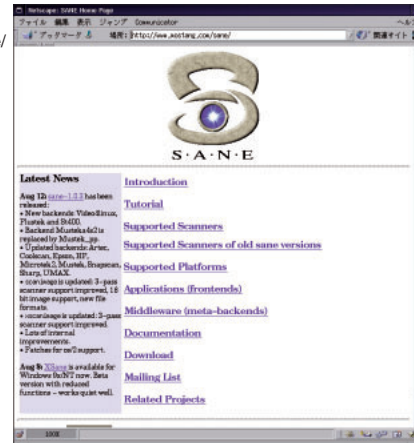
ビルドとインストール

執筆時点で最新のSANEは、バージョン1.0.3だ。しかし、このバージョンでは「カラーのスキャンがうまくいかない」、「Gimpのプラグインとして動作しない」という問題があったため、バ



画面1 SANEのWebページ
<http://www.mostang.com/sane/>

写真1 エプソン GT-8700
今回使うGT-8700はSCSI / USBの両方に対応するスキャナだ。



ージョン1.0.2を使用した。

SANEは、一次配布サイトでは、tarボールによるソースで配布されている。ソースを配布している主なサイトは表2のとおりだ。tarボールからのインストールは、「./configure」「make」「make install」という手順で可能だ。

なお、X上のGUIツールであるxscanimageやxcamを使う場合には、SANEをビルドするときにGTKライブラリ(libgtk、libgdk、libglib)と関連するヘッダファイルがインストールされている必要がある。また、GimpからSANEを利用したい場合には、さらにlibgimp(バージョン0.99.13以上)と関連するヘッダファイルもインストールされている必要がある。

バイナリの配布については、表3に示したサイトで配布されている。

インストールが完了したら、まず、バックエンドの選択を行う。今回の例では、エプソン用の設定ファイル“/usr/local/etc/sane.d/epson.conf”を書き換える。エディタでファイルを開き、「# /dev/scanner」という行のコメントマーク「#」をはずすと、エプソン用のバックエンドがSCSIスキャナを認識できるようになるはずだ。ほかのメーカー製スキャナでも書き換えるファイルが違うだけで手順は同じだ。どの設定ファイルを利用すればよいのかは、ドキュメントを参照してほしい。

次に、スキャナの電源を入れてからリポートし、SANEがスキャナを認識することをチェックしよう。コマンドラインから、次ページ画面2のようにタイプすれば、認識しているスキャナが表示されるはずだ。この作業はスーパーユーザー(root)で行う。

認識されない場合は、さらに画面3のようにタイプして、LinuxがSCSI経由でスキャナを認識しているかどうか

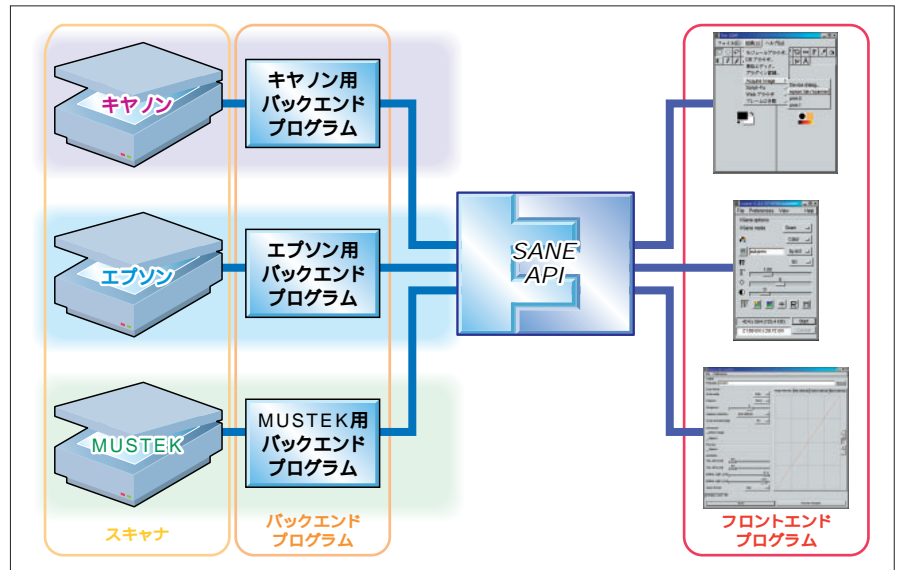


図1 SANEの動作概要
ドライバは各機種ごとに、ユーザーインターフェイスは必要なものをそれぞれ別途作成する。

を確認してみよう。

なお、スキャナがつながっているデバイスファイル/dev/sgbなどは、通常スーパーユーザー以外はアクセスできないようになっている。そのためスーパーユーザー以外のユーザーがスキャナを使う場合は、デバイスファイルのパーミッションを変更しておく必要がある。スキャナが/dev/sgbにつながっている場合は次のようにする。

```
# chmod 666 /dev/sgb
```

この後、/dev/scannerというデバイスファイルから、実際にスキャナがつながっているデバイスへのリンクを作成しよう。

```
# ln -s /dev/sgb /dev/scanner
```

メーカー	機種
キヤノン	CanoScan 300、CanoScan 600、CanoScan 2700F
エプソン	GT-5500、GT-7000
ヒューレットパッカード	HP ScanJet Plus、II、3、4、5、6200Cほか
日本電気	PC-IN500
コダック	DC210、DC25、DC20
Connectix	QuickCam

表1 SANEが対応している代表的な機種

サイト	URL
公式サイト	ftp://ftp.mostang.com/pub/sane/
会津大学	ftp://ftp.u-aizu.ac.jp/pub/misc/device/scsi/scanner/sane/

表2 SANEのソースを配布している主なサイト

サイト	URL
Red Hat	ftp://ftp.redhat.com/pub/redhat/beta/pinstrip/i386/en/RedHat/RPMS/ ftp://ftp.redhat.com/contrib/libc6/i586/
KDD研究所	ftp://ftp.kddlabs.co.jp/pub/Linux/packages/RedHat/contrib/libc6/i586/
TurboLinux	ftp://ftp.turbolinux.co.jp/pub/TurboLinux/unstable/RPMS/
deb形式	http://www.debian.org/Packages/unstable/graphics/sane.html

表3 SANEのバイナリを配布している主なサイト

```
# scanimage -l
device `epson:/dev/sgb' is a Epson GT-8700 flatbed scanner
```

画面2 SANEでスキャナが認識されていることを確認する

```
# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 02 Lun: 00
Vendor: EPSON Model: Scanner GT-8700 Rev: 1.03
Type: Processor ANSI SCSI revision: 02
```

画面3 SCSIデバイスとして認識していることを確認する

これで/dev/scannerを指定してスキャナを利用できるようになる。



SANEの豊富な フロントエンド

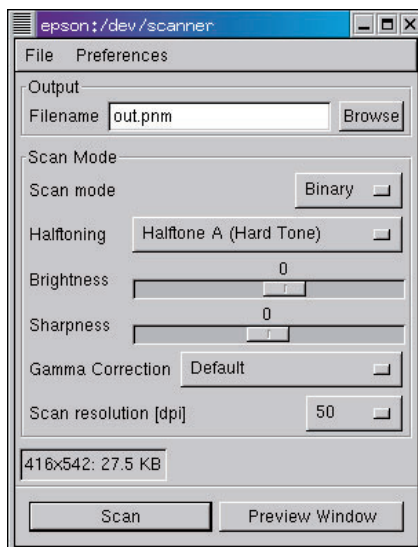
次はフロントエンドを見てみよう。

xscanimage

xscanimageは、X上でGUIによってスキャナを操作するフロントエンドだ(画面4)。次のようにして実行する。

```
$ /usr/local/bin/xscanimage
```

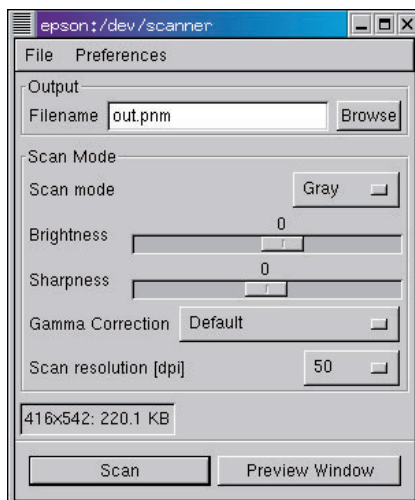
GT-8700の場合は、白黒2値(Binary)、グレイスケール(Gray)、カラー(Color)の3つのモードでスキ

画面5 Binaryモードでのxscanimage
白黒2値で画像を取り込むときのモード。

ヤンできる。スキャンするモードに応じて設定項目が変化する(画面5~7)。さすがに、WindowsのTWAINドライバのようなオートモードといったものはない。スキャンしてからGimpを使って補正するとよいだろう。

では、実際にスキャンする際の手順を追っていこう。

- 1.まず、スキャンするモードを設定する。
- 2.次に、保存するファイル名を「file name」欄に入力しておこう。すでにあるファイルは警告なしに上書きされるので、注意してほしい。
- 3.右下の「Preview window」ボタンをクリックしてプレビュー画面を開く。
- 4.プレビュー画面左下の「Acquire Preview」をクリックすると、プレビュースキャンが行われる。このプ

画面6 Grayモードでのxscanimage
グレイスケールで画像を取り込むときのモード。

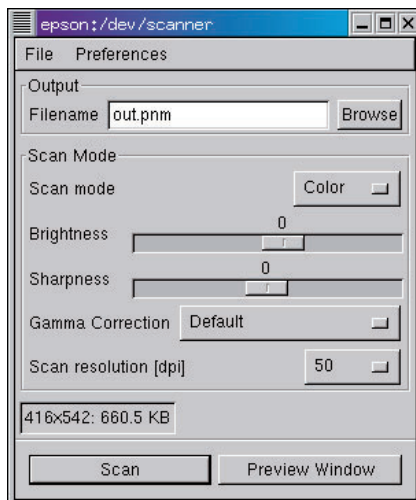
画面4 xscanimage

SANEのパッケージに含まれるGUIフロントエンドプログラム。

レビューでは、「Brightness」などの補正オプションが反映されるので、オプションを変更しながらプレビューを繰り返すこともできる。

- 5.補正オプションが決定したら、マウスを使ってプレビュー画面上でスキャンする部分を選択しよう。そして、解像度(Scan Resolution)を設定し、「Scan」ボタンをクリックする。

WindowsやMacintoshと比べても、遜色のないユーザーインターフェイスだといえる。さらに、「Preferences」メニューから、「Show advanced option」を選択することにより、細かいオプション設定を行うことができる(画面8)。

画面7 Colorモードでのxscanimage
カラー画像を取り込むときのモード。

動く? 動かない?

周辺機器バトルロイヤル

WindowsとLinuxで実際にスキャンしたデータを見てほしい(画面9~11)。それぞれ、若干色合いが違うものの、Gimpなどで十分に補正可能な範囲だ。

なお、xscanimageでスキャンした画像はPNM形式(Portable aNY Map)で保存される。Linuxではしばしば使われる画像形式だが、ファイルサイズを小さくしたり、Webで使用したりするためにJPEGやGIF形式で保存したいことも多いだろう。GimpではPNM形式の画像を開くこともできるので、さまざまな画像形式に変換して保存することが可能だ。しかし、変換するファイルが多いときなどは、Gimpで1つずつ保存しなおすのは面倒だ。そのようなときは、convertコマンドでコマンドラインから簡単に形式を変換することができる。

convertコマンドは変換先ファイル名のピリオドよりあとの文字を判断して、変換する画像形式を決定する。

scanimage

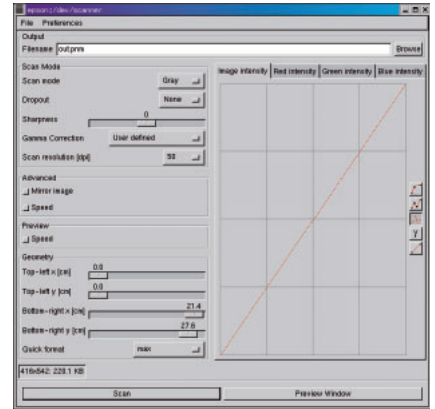
コマンドラインでスキャナを操作するのが、scanimageコマンドだ。レビューしながらの範囲指定などはでき

ないが、基本的にはX上で動作するxscanimageとほぼ同等の機能を持っている。コマンドラインツールゆえに、XやGTKがインストールされていない場合にも使用できる。また、大量の文書をスキャナで画像変換するなど、シェルスクリプトを活用して一括処理したいときには便利だろう。

scanimageコマンドは、読み込んだ画像を標準出力に出力する。標準出力をリダイレクトしておかないと、画面がぐちゃぐちゃになってしまうので注意してほしい。なお、対応する画像形式はPNMのほかにTIFFも選べる。

次の例では、スキャンしたイメージをoutput.tiffというファイルにTIFF形式で出力する。

```
$ scanimage --format tiff --device-name=epson:/dev/scanner > output.tiff
```



画面8 Colorモードのadvanced option 細かいガンマ補正も行うことができる。

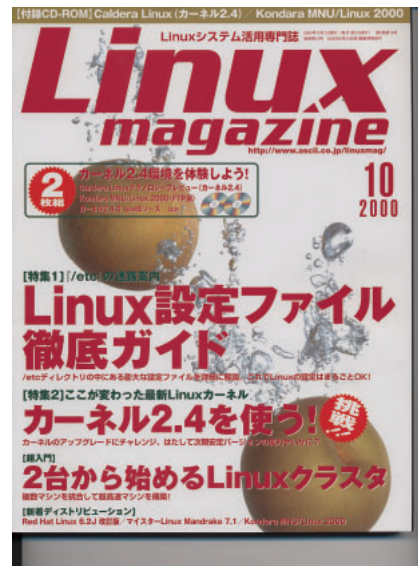
主なオプションは表4に示すとおりだ。これ以外にもスキャナによって使用できるさまざまなオプションがある。これらのオプションはmanコマンドで調べてほしい(エプソンのスキャナなら「man sane-epson」でマニュアルが表示される)。また「-help」オプションを使っても、各スキャナ用の詳しいオプションを調べることができる。

オプション	説明
--device-name=device (-d)	デバイス名がdeviceのスキャナから読み込む
--format format	出力する画像形式を指定する(pnmもしくはtiff)
--list-device (-L)	使用できるデバイスをリストアップする
--test	動作テスト
--help	ヘルプの表示

表4 scanimageの主なオプション



画面9 Windowsでスキャンした例(オート)



画面10 Windowsでスキャンした例(マニュアル)



画面11 SANE 1.0.2を使い、Linuxでスキャンした例

XSane

SANEのtarボールには入っていないが、xscanimageの上位互換ツールといえるツールがXSaneだ(画面12)。

XSaneは、一次配布サイトでは、tarボールによるソースで配布されている(画面13)。tarボールからのインストールは、「./configure」「make」「make install」という手順で可能だ。ただし、XSaneを使用するには、libgtk(1.0.xもしくは1.2.x)とlibglibがインストールされている必要がある。また、GimpからXSaneを利用する場合は、libgtk(1.2.0以降)がインストールされていること、Gimpのバージョンが1.0.4以降であることが推奨されている。

なお環境によっては、XSaneを起動しようとすると、「ライブラリlibsane.so.1が見つからない」というエラーが出ることもある。この場合は、libsane.so.1がインストールされた“/usr/local/lib/sane”をライブラリのサーチパスに加えてしまおう。サーチパスは“/etc/ld.so.conf”ファイルに書かれているので、スーパーユーザーになって、このファイルをエディタで開き“/usr/local/lib”という1行を追加する。追加したらldconfigコマンドを実行してサーチパスを認識させればよい。

```
# vi /etc/ld.so.conf (パスを追加)
# ldconfig
```

XSaneとxscanimageの違いで、特に重要なのが次の3点だ。

1つめは、保存することのできる画像形式が多いこと。JPEG、PNG、PNM、PS、TIFF、RAWなどに対応している。これらは起動時に「by ext」となっているボタンをクリックすることにより選択することができる。

2つめは、ガンマ値や明度、コントラストの補正が、プレビュー画像にリアルタイムに反映されることだ。実際にはプレビュー画像に対して補正をかけているので、本当にスキャンされる画像とは異なることもあるが、xscanimageでは、補正の効果を見るためにはスキャンが必要だったのに比べると、とても便利になっている。

3つめが、スキャンモードのほかに、コピーモードとFAXモードが用意されている点だ。コピーモードを使うとスキャンした画像をPostScriptに変換してプリンタに出力する、まさしくコピー機として使えるものだ。FAXモードはスキャンした画像を、hylafaxやsendfaxといったLinux用のFAXソフトに出力して送ってくれる。

このほか、SANEのパッケージにはConnectix QuickCamのようなビデオカメラを扱うためのxcam、ネットワーク経由でリモートスキャンをするためのデーモンsanedなどのソフトが含まれている。



Gimpでの利用

フロントエンドでスキャンした画像を、フォトタッチソフトのGimpで修

正することも多いだろう。xscanimageやXSaneはGimpのプラグインとして使うこともできる。実際に使ってみると、Gimpにスキャナ制御機能がついたように感じる。取り込んだ画像を修正することが多い場合、これは便利だろう。

この機能を使うためには、Gimpのplug-insディレクトリの中にxscanimage(もしくはXSane)のシンボリックリンクを作ればよい。xscanimageが/usr/local/binにあるのなら、次のようにする。

```
$ ln -s /usr/local/bin/xscanimage
~/gimp-1.0/plug-ins/
```

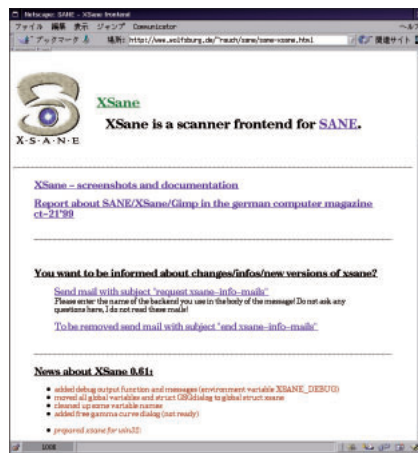
または、

```
$ ln -s /usr/local/bin/xscanimage
~/gimp/plug-ins/
```

その後、Gimpを立ち上げると、メニューの[Xtns][Acquire Image][Device dialog]でxscanimageが起動するようになる(画面14)。xscanimageが起動したらそのままスキャンすれば、画像はGimpの1ウィンドウとして表示される(画面15)。これは便利だ。



画面12 xscanimageよりも高機能なXSaneソフトの名前はXSaneだが起動コマンドは「xsane」だ。



画面13 XSaneのWebページ
http://www.wolfsburg.de/rauch/sane/sane-xsane.html



市販のドライソフト

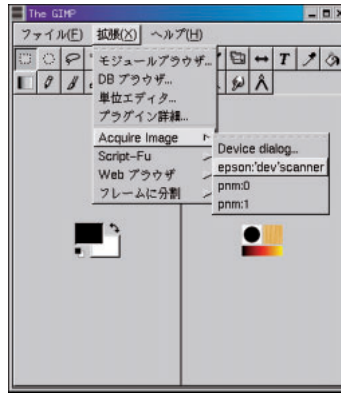
SANEは優秀なフリーソフトウェアだが、「インストールが面倒」あるいは「動作保証がほしい」という向きには、エプソン、キヤノンなどのスキャナに対応する市販スキャナソフトを利用する方法もある。

そこで今回は、Webページで評価用プログラムのダウンロードができるアドバック・システムのGTscanという製品を試してみた。GTscanは、エプソンやシャープ、リコーなどのスキャナに対応している。対応ディストリビューションは、Red Hat Linux 6.2やTurboLinux 6.0、Vine Linux 1.0、OpenLinux 2.3などとなっている。メーカーの推奨する環境であれば、ユーザーサポートを受けることができるので、ビジネスユーザーには心強い。

またGTscanには、輪郭抽出や画像のデータベース化などといった、SANEではサポートしていない機能があるのも魅力的だ。

評価用プログラムはスキャンだけに機能が限定されているものの、最も重要となる動作チェックには十分だ(画面16)。購入を検討する際には、まずダウンロードして試してみるとよいだろう。

なお、GT-8700は新しい製品でもあるので、GTscanの対応スキャナにはリストアップされていない。もっとも、スキャナハードウェアの内部コマンドは、そうそう変わるものではないと思いきや、まったく問題なく動作した。ただし、GT-8700での動作に関しては、メーカーはもちろん編集部でも保証するものではなく、あくまでも参考ということでご了承ください。



画面14 Gimpでスキャン
Gimpの中にスキャナ用のメニューが組み込まれる。



画面15 取り込んだ画像をGimpで表示
スキャンした画像はGimpでそのまま自由に編集できる。



USB接続

では、最後にUSB接続にチャレンジしてみよう。

今回はカーネル2.4.0-test8をインストールしてUSBを使えるようにした。カーネルをインストールする際にUSBスキャナのサポートを有効にするのを忘れないこと。

システムが起動したら、mknodコマンドでUSBデバイスノードを作成する。デバイスノードを作成したら、chmodコマンドでパーミッションを変更し、一般ユーザーでもスキャナデバイスを使えるようにしよう。

```
# mknod /dev/usbscanner c 180 48
# chmod 666 /dev/usbscanner
```

「/dev/usbscanner c 180 48」という設定については、カーネルソースの“Documentation/devices.txt”に詳しく記載されている。

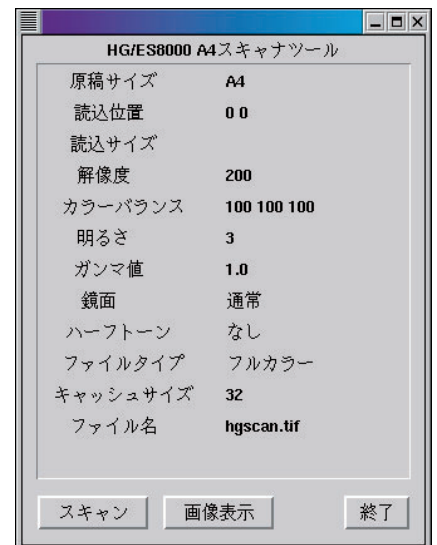
次に、USB用のモジュールを組み込もう。まず、USBのコントローラモジュールを組み込んで、それからスキャナ用のモジュールを組み込む。

```
# /sbin/modprobe usb-uhci
```

```
# /sbin/insmod scanner
```

USBのコントローラモジュールは、チップセットの種類によって「usb-ohci」になることもある。

モジュールの組み込みが終わったら、最後はSANEの設定だ。エプソン用の設定ファイル“/usr/local/etc/sane.d/epson.conf”の中にある「# usb/dev/usbscanner」という行の先頭にあるコメントマーク「#」をはずすと、エプソン用のバックエンドがusbスキャナを認識できるようになる。これで、USB接続したGT-8700を利用できるようになる。



画面16 GT scanの試用プログラム
アドバック・システム <http://www.advac.co.jp/>

メモリメディア

デジカメのデータをLinuxマシンに保管したい。でも、gPhotやカーネル2.4でサポートされていない場合どうすればいい？

文：にゃー@編集部

Text：Nyaa@Linux magazine

フラッシュパス

デジタルカメラやMP3プレーヤなどを通じて、スマートメディアやコンパクトフラッシュなどの小型のフラッシュメモリメディアを使用する機会が増えてきている。コンパクトで大容量、そのうえハードディスクと同じように記録されたデータを操作できるメモリメディアは、確かに優れた記憶媒体だ。これからも利用分野を広げながら、ますます普及していくだろう。

そうなってくるとLinuxでも使いたいというのが人情だ。デジカメに関しては、gPhotで多くの機種がサポートされているが、新しい機種やUSB接続のものはサポート待ちというのが現状だ。また、カーネル2.4のUSBサポートには、Kodakのデジカメ「DC-2XX」シリーズやDiamond MultimediaのPM3プレーヤ「Rio500」のドライバが含まれているが、こちらは機種が限定される。専用のカードリーダーやアダプ

タがLinuxで動作すれば問題は解決する。そこでサポートの現状をテストしてみることにした。

今回は数あるメモリメディアの中から（メモリメディアの規格については75ページのコラムを参照）、普及率の高いスマートメディアとコンパクトフラッシュカード（写真1）をテストした。まずはスマートメディアから。



Linux Ready ?

スマートメディアにアクセスするために利用されるデバイスのうちメジャーなもののひとつにフラッシュパスがある。フラッシュパスは、写真2からもわかるようにフロッピーディスクと似た外観を持っていて、サイズもまったく同じ。それもそのはず、このデバイスはスマートメディアを装着してFDドライブにセットすることでアダプタ

として機能するのだ。そのままズバリ「フロッピーディスクアダプタ」と呼ばれることもある。あとはデバイスドライバさえあれば、フロッピーにアクセスすると同様にスマートメディアに記録されたデータを操作できる。

では、Linux用のドライバはあるのかということ、これがあったのだ。Linux用ドライバは、SmartDisk CorporationのWebサイト（<http://www.smartdisk.com/>）から入手できる。トップページの「Download」からリンクをたどっていけば見つかるはずだ。

ドライバのダウンロードページから入手できるPDF形式のドキュメントによると、このドライバは2.2系のカーネルに対応しており、サポートするディストリビューションはRed Hat 6.1となっている。また、カーネルソースとCコンパイラ以外にも、以下のRPMパッケージが必要となるようだ。確認しておこう。

```
gnome-lib-devel
xpm-devel
```



写真1 今回テストしたメモリメディア
右側がスマートメディアで、左コンパクトフラッシュカード。ほぼ同じサイズだが、コンパクトフラッシュのほうが厚みがある。

写真2 テストに使用したフラッシュパス
写真3のデジカメと一緒に弊社の備品を拝借してテストを行った。スマートメディアを取り上げたのは、たまたま手近にあったからでは決していない。




```
esound-devel
gtk+-devel
XFree86-devel
ORBit-devel
```

インストール方法は定石どおり。tarボールを展開して、「./configure」、「make」、「make install」で完了だ。問題がなければ、/lib/modules/2.2.XX/block/に「flashpath.o」が、/usr/local/binに「fpmonitor」がインストールされる。fpmonitorはその名が示すようにフラッシュパスのモニタリングソフトである。バックグラウンドで動作し、ステータスに変化があればポップアップメッセージを表示してくれるらしい。GNOMEにも対応しているようなので、あとで動作テストのときに確認してみよう。

テストには、LASER5 Linux 6.2を使用することにした。カーネルのバージョンは2.2.14である。サポート対象のRed Hat 6.1で動くのは当然として、ほぼ互換性があると思われる日本語ディストリビューションでの動作を試したかったからだ。ハードウェアにも触れておくと、フラッシュパスは富士フィルム製、スマートメディアはオリンパスのデジカメCAMEDIA C2000ZOOM(写真3)に付属のものを使用した。

コンパイルとインストールは問題なく行えた。動作はどうだろうか? まずはドライバのロードから。ついでにモニタリングソフトも起動しておこう。

```
# modprobe flashpath
# /usr/local/bin/fpmonitor
```

特にエラーなどは発生していないようなので、フラッシュパスをFDドライブとしてマウントしてみる。

```
# mount -t vfat /dev/fd0 /mnt/floppy
```

マウントも問題なくできた。ファイルのコピー、削除なども試したが問題ないようだ。スマートメディアを使用しているデジカメで撮影した画像ファイルの表示をX上で行ってみたが、これにも成功(画面1)。前出のドキュメントによると、フォーマットは危険らしいので控えることにした。



若干問題あり

しばらくは機嫌良く動いていたのだが、スマートメディアにコピーしたテキストファイルをlessしたところで、ベロベロ~っとエラーメッセージが表示され、最後に「フラッシュパスを挿

入し直してください」というメッセージが表示された。エラーが発生したのは悲しいことだが、fpmonitorの動作が確認できたのでよしとしよう。GNOMEでポップアップメッセージがどう表示されるのか見たかったので、XのコンソールやGMCでいろいろ操作してみたが、残念ながらこちらは確認できなかった。

何度か反復してテストしたところ、確かにlessやviなどでテキストファイルにアクセスする際にエラーが起こる。ところが、gimpで画像データを開いて加工してもエラーは発生しないのだ。ファイルのコピーや削除といった操作でエラーが起こることはなかった。かなりナゾな部分は残るものの、今回のテスト結果から判断する限り、データの受け渡しだけなら問題ないようだ。

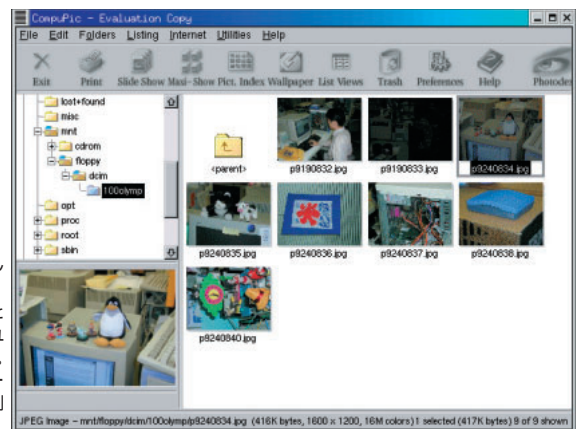
ただし、フロッピーインターフェイスを介しているせいか、リード/ライトが非常に遅い(これはLinux固有の問題ではないと思われる)。主な用途はデジカメやMP3プレーヤとのデータ交換であろうことを考えると、この点にもやや不満が残る。

まったく問題なしとはいかないまでも、総合的には、何とか「及第点」といったところだ。どうしても必要な場合は、試してみる価値があるかも。



写真3 オリンパスCAMEDIA C2000ZOOM 備品その2。ファイルのコピーや削除のテストには、このカメラで実際に撮影したデータを使用した。

画面1 サムネイル表示した画像データ
LinuxからはFDドライブとして見えるので、専用のユーティリティは必要ない。使い慣れたコマンドやツールでデータを扱えるのが利点のひとつだ。



フラッシュパス

コンパクトフラッシュは、スマートメディアと並んでメモリメディアの主流となっており、デジカメをはじめとする多くの機器で記録メディアとして利用されている。

コンパクトフラッシュに記録されたデータをPCで操作する場合、PCカードタイプのアダプタを利用するのが一般的だが、今回はUSBとの合わせ技ということで、USB接続のコンパクトフラッシュリーダーに挑戦してみよう。



機種選定は慎重に

テスト機器の選択にあたっては、Linux USBプロジェクトのWebサイト (<http://www.linux-usb.org/>) に掲載されているサポート情報を参考にした。コンパクトフラッシュリーダーは、USBのクラス分けでは「Mass storage devices」に分類されるので、トップページから [Working devices list] - [devices overview] - [Mass Storage] とリンクをたどっていけば目的の情報を発見できるはずだ。

リストにはコンパクトフラッシュの

本家SanDiskの製品のほか、いくつかの機種が掲載されていた。このページは、実際に使ってみて動作した機器の情報をユーザーが登録する仕組みになっている。日本のメーカーの製品もあったので、よく見てみると登録者は日本の方であった。何だかうれしかったので、この製品に決定することにした (ハギワラ シスコム FlashGate、型番はHBC-UC10。写真4)。

機種が決まったところで秋葉原へ。量販店を見て回ったが、どの店もコンパクトフラッシュリーダーの品揃えはあまりよくないようだ。4件めでようやく目的の機種を発見。実売価格は5979円である。PCカードのアダプタは500円くらい。やはり価格からいってもこちらが主流なのだろう。

同じ店でメディアも購入した。こちらは32Mバイトで7479円。リーダーよりメディアのほうが高いのだ。コラムでも紹介したように、コンパクトフラッシュのカードにはType とType の2種類があり、厚さが若干異なる。Type 用のスロットにはType のカードは挿せないので、購入時には注意

が必要だ (Type のスロットでType を使用することは可能)。



2.2.18pre9は.....

テストするブツが揃ったところで、Linuxの設定へと移ろう。テスト環境はフラッシュパスと同じく、LASER5 6.2 + カーネル2.2.17 + 2.2.18pre9パッチとした。

前述したように、コンパクトフラッシュリーダーはUSB Mass Storageクラスに分類されている。デバイスドライバモジュールはusb-storage.oである。このドライバはUSB接続された機器をSCSIデバイスとしてエミュレートする。OS側では、コンパクトフラッシュがSCSIハードディスクとして認識されるわけだ。よってカーネルコンフィグレーションでは、USBまわりの設定のほかにSCSIサポートの設定も必要になる。テスト結果をもとに先にバラしておくとして、SCSIサポートでは [SCSI support] と [SCSI disk support] の2つを有効にしておけばOKである。

カーネルを再構築してモジュールをロードしてみよう。テストということで、すべてのドライバをモジュール化しており、ロードするだけでも結構たいへんだったりする。

```
# modprobe scsi_mod
# modprobe sd_mod
# modprobe usbcore
# modprobe usb-uhci
# modprobe usb-storage
```

カーネルオプションの [USB verbose debug messages] を有効にしているので、USB接続に問題があればエラーメッセージがコンソールに表示されるはず.....。されました。usb-uhci.oの



写真4 ハギワラシステム FlashGate USB接続デバイスらしく(?)トランスルーセント仕様。チープな感じも味わい深くていい。

ロードでは「何かデバイスが見つかったけど、インストールされてるドライバでは認識できませんワ」と文句をつけられ、usb-storage.oのロードでは「時間かかりすぎ。バスのリセットも失敗」とのそっけないメッセージが。

一応はMass Storageクラスのデバイスとして認識しているみたいだ。デバイス側のPowerランプも点灯しているので、バスから電源も供給されていると思われる。おいしいとこまでいってるんだけどなー。



怒涛のメッセージ

この時点で「あの報告はイタズラなのか? オレってだまされてる?」と思わないでもなかったのだが、考えてみると報告にはカーネルのバージョン

などの詳しい説明は載っていなかったのだ。そこで、気を取り直して2.4.0-test8で試してみることにした。

カーネルコンフィグレーションは18pre9とまったく同じ。ただ、モジュールのロードが面倒なのでSCSIサポートはカーネルに組み込んだ。

コンパイルして、再起動して、ロードして、と同じ手順をトレースしていく。ここで本特集最大級のトラブルが発生した! usb-storage.oをロードした直後に、怒涛のメッセージの洪水がまさしく「どと〜」と表示されたのだ。コンソール画面をすさまじい勢いでメッセージが流れていく。しばらく放置しておいたが、まったく止まる気配もない。その間はキー入力も何も受け付けない。しかたがないので、リセットスイッチを「プチッ」とした。

何度繰り返しても結果は同じ。目を凝らして流れゆくメッセージを見ると、どうやら/dev以下のscsiのデバイスファイルに次々と何ごとかの処理を試みでは失敗しているようだ。SCSI関連のカーネルオプションを変えたり、/var/log/dmesgやmessagesをじっくり眺めてみたりしてみたが解決には至らず。やっぱだまされてる?

結局動かなかったのかということそうではなくて、2.2.18pre9のときと同じくSCSI関連のドライバをモジュール化したところ(ファイル名がscsi.oとsd.oに変わっていた)見事に接続できたのであった。原因は不明。メディア上のデータの操作も問題なく行うことができた。何だかスッキリしないことは確かだが、とりあえず動くことは動くというのが結論である。以上!

Column

携帯メモリメディアのあれこれ

スマートメディアとコンパクトフラッシュ以外にも、メモリスティックにマルチメディアカード(MMC)、さらにはマイクロドライブと、メモリメディアにはさまざまな規格がある。乱立という感もなきにしもあらずなので、整理する意味でも、それぞれの規格について簡単に触れておくことにしよう。

スマートメディア

東芝が開発し、富士フィルム、オリンパスなどと共同して提唱した規格。主に国内で、デジタルカメラ、MP3プレーヤの記録メディアとして広く利用されている。切手サイズで非常に薄い(45mm×37mm×0.76mm)のが特徴。PCからのアクセスには、専用のリーダー/ライター、フラッシュバス、PCカードアダプタなどが必要となる。最大記憶容量は64Mバイト。

コンパクトフラッシュ

米国のSanDiskが開発・提唱した規格。ス

マートメディアと並んで、デジタルカメラやMP3プレーヤの分野で広く普及している。また、ノートPC、ハンドヘルドPC、PDAにも専用スロットを備えているものがある。

カード自体にコントローラチップを内蔵しているため、スマートメディアとほぼ同じサイズながら、かなり厚めである(Type が3.3mm、Type が5mm。両者はスロット形状も異なるので注意が必要)。PC側からはPCカードATA規格に準拠するストレージとして認識される。最大記憶容量は、Type が192Mバイト、Type が300Mバイト。PCでの利用はPCカードアダプタが最も一般的。

マルチメディアカード(MMC)

コンパクトフラッシュの小型バージョンとして、同じくSanDiskが開発した規格。MP3プレーヤを中心に、徐々に採用されはじめている。PCでの利用には、専用のフラッシュバスが必要となる。最大記憶容量は64Mバイト。

マイクロドライブ

IBMが開発した超小型ハードディスクドライブ。他のメディアと違いフラッシュメモリ

を使用していないのが特徴。インターフェイス規格としては、コンパクトフラッシュ(CF+ Type)に準拠している。このため、厚さも含めてサイズは、Type のコンパクトフラッシュカードとまったく同一である。

最大のウリはその記憶容量で、現在1Gバイトのものが開発されている。PCでの利用には、専用のPCカードアダプタが必要。専用スロットを備えたデジタルカメラ、ノートPC、ハンドヘルドPCも登場している。

メモリスティック

SONYが開発/提唱した規格。VAIOをはじめとする同社の製品でサポートされている。先ごろ発売されたPalm OS搭載PDA「クリエ」も、専用スロットを備えている。形状は他のメディアと異なり長方形(21.5mm×50mm)で、厚さはコンパクトフラッシュのType とほぼ同じ(2.8mm)。PCでの利用には、専用のPCカードアダプタまたはフロッピーアダプタが必要。また専用のリーダー/ライターも登場している。最大記憶容量は64Mバイト。

番外編：Linuxで動かせなかったUSBデバイス

ナニも考えず、場当たりに機器調達をしたのが裏目に出て、動かないモノもあった。のりぞうが敗れ去るまでのドキュメントをお伝えする。

文：のりぞう

Text : norizoh@nought.rim.or.jp

USBデバイスとの戦い方

この特集のために機材を購入するべく、のりぞうはLinux magazine編集部員とともに秋葉原へ乗り込んだ。さすがは世界の電腦街、いろいろなものが売られていてクラクラする。秋葉原マイスターの編集部員と一緒によかったと思いつつ、USB接続デバイスを購入したのだが、のりぞうが選んだ3つに限ってLinuxで動かないのだ。

ここからは、番外編としてUSBデバイスとの戦い方を概論し、続いて個別のデバイスとの戦いの記録をお届けする（といってもさすがにいいほど見事に敗れ去ったのだが）。



情報収集が肝要

最も重要なのは己を知ること、すなわちLinuxのUSBデバイスサポート状況をおさえておかねばならない。本特集の冒頭で解説しているので参考にしてほしい。また、Linux USB ProjectのWebページ（<http://www.linux-usb.org/>）では、USBデバイスの稼働実績を調べることができるので参考にできるだろう。

次に大事なことは敵を知ることだ。前述のページで稼働実績ありとされているデバイスを購入できれば問題はないが、日本国内で販売されているUSBデバイスはあまり掲載されていない。このようなデバイスを購入してしまった

ら、実際に接続して調べるほかはないのだ。



接続されたデバイスを調べる

接続したUSBデバイスを調べるには、カーネルを構築する際に“CONFIG_USB_DEVICEFS=y”としてデバイス情報を参照できるようにする。make xconfigで設定する場合は、“USB support”メニューで“Preliminary USB device filesystem”の項目を“y”にすればよい。

新しいカーネルで起動したら次のようにして、USB device filesystemをマウントする。

```
# mount -t usbdevfs none /proc/bus/usb
```

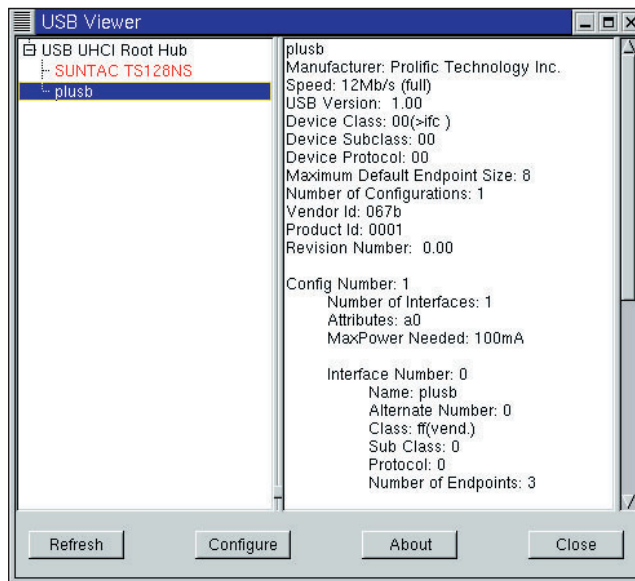
これで、/proc/bus/usbディレクトリにdriversとdevicesというファイルが作られる。

driversには、現在組み込まれているUSB関連のドライバが書かれている。また、devicesには、接続されているUSBデバイスの情報が書かれているので、catやlessで開けば認識状況がわかるはずだ。USBView（画面1）を使えば、デバイスごとにわかりやすい情報を得ることもできる。

devicesファイルにある項目の中で重要なのは、Manufacturer、Vendor、ProdIDの3つである。次ページから機器ごとに見ていくことにしよう。

そうそう、試した機器が動作したときは、Linux USB Projectで動作実績レポートを公開してほしい。あなたの報告が不幸な人を減らすかもしれない。

画面1 USBView
(<http://www.kroah.com/linux-usb/>)
適切なドライバが組み込まれていないデバイスは赤い文字で表示される。Manufacturer、Vendor Idでメーカー名を、Product Idで機器の種類を調べることができる。



USB接続 3.5 / 2.5 インチハードディスクケース

ブラネックスコミュニケーションズのRX-35Uは、USB接続用のハードディスクケースだ。パッケージにはACアダプタ、USBケーブルが含まれており、2.5インチ、あるいは3.5インチのATAPI (IDE) 接続ハードディスクのベアドライブを別途用意すれば、USB接続の外付けハードディスクが完成する。対応OSは、Windows 98 / 98 Second Edition / 2000となっており、Linuxへの対応については何も書かれていない(あたりまえ?)。従って、Linuxでの動作については、メーカーへの質問はしないようお願いしたい。

LinuxでUSB接続のハードディスクを使う場合、カーネルが“CONFIG_USB_STORAGE=m”あるいは“CONFIG_USB_STORAGE=y”で作られている必要がある。make xconfigでは、USB support”メニューで“USB Mass Storage support”の項目を“m”または“y”にする。

今回は、モジュールとして構築したので、ドライブを接続してから次のようにしてドライバを組み込んだ。

```
# modprobe usb-storage
```

lsmodコマンドでモジュールが組み込まれていることも確認できた。ところが、USBViewではデバイス名が赤字で表示されている。これは、適切なドライバがないことを示している。すなわち、usb-storageドライバではRX-35Uを使うことができないのだ。



デバイス情報を見る

USBViewでデバイス情報を調べると、ATAPI-4 Bridge Controllerとして表示される。Manufacturerは、Prolific Technology, Inc.だ。Vendor Idは067b、Product Idは2307と表示されている(画面2)。

Linux USB ProjectのWebページにある、“USB Vendor / Device IDs list”で検索したところ、これは、Prolific Technology, Inc.のUSB-ATAPI4 Bridge PL-2307となっていた。ケースを開けると、基板にはPL-2307Bと書かれたチップが載っている。なるほど、これが。

Prolific Technology, Inc.のWebページ(<http://www.prolific.com.tw/>)を見ると、PL-2307はUSBとATAPIの相互変換機能を持ったチップで、

ATAPI-4で定義されたコマンドのほとんどをサポートするという。

DMAやUltraDMAモードはサポートしておらず、PIOモードのみに対応するが、USBの転送速度がボトルネックになるから問題はないそうだ。しかし、動かないことにはどうしようもない。念のため、/proc/bus/usb/devicesをのぞいても“Driver = (none)”となっており、デバイスに対応するドライバがないことがわかる。

このコントローラチップはLinuxでサポートされていないというのが現状だ。のりぞうにはデバイスドライバを書くスキルもないので、残念だがあきらめるしかない。



のりぞうは引きが弱い?

USB接続の周辺機器に関しては、メーカーがLinuxでの動作を保証しているものはまずないだろう。また、コントローラチップに何を使っているのかはつないでみるまでわからないのがふつうだ(店頭で中を開け、チップを拝むなんてできっこない)。運のいい人なら、Linuxで動作するUSB接続ハードディスクを選ぶことができるのかもしれない。

なお、試したケースはドライバの作成に挑戦してくださる方にプレゼントする。お申し込みはのりぞうまで。



写真1 ブラネックスコミュニケーションズRX-35U
USB接続の外付けハードディスクは、ノートパソコンでは特に重宝するのだが……無念。

画面2 USBViewでデバイス情報を見る

```
ATAPI-4 Bridge Controller
Manufacturer: Prolific Technology Inc.
Serial Number: 0
Speed: 12Mb/s (full)
USB Version: 1.00
Device Class: 00(>ifc)
Device Subclass: 00
Device Protocol: 00
Maximum Default Endpoint Size: 8
Number of Configurations: 1
Vendor Id: 067b
Product Id: 2307
Revision Number: 1.00
```

USB接続ISDNターミナルアダプタ

次なるデバイスは、サン電子のUSB対応ISDNターミナルアダプタSUNTAC TS128NSだ(写真2)。購入価格1万3980円という低価格ながらも、USB接続、シリアル接続の両方に対応する。DSUを内蔵し、アナログポートを2つ備えるなど基本スペックは十分だ。

対応OSはWindows 95 / 98 / 98 Second Edition / NT 4.0 / 2000、Mac OSとなっている。Linuxでの動作保証はない。



USBモデム/TAサポートCDC ACM

LinuxのUSBサポートには、the Universal Serial Bus Communication Device Class Abstract Control Model (USB CDC ACM) 規格に対応するドライバが含まれている。

この規格に合ったモデムやTA(ターミナルアダプタ)を利用するためには、カーネル構築時に“CONFIG_USB_ACM=m”または“CONFIG_USB_ACM=y”とする。make xconfigなら、“USB Modem (CDC ACM) support”で“m”または“y”を選ぶ。



写真2
サン電子 SUNTAC TS128NS BookletTA(ブックレット)という愛称をもつ低価格TA。USBだけでなく、シリアル接続で使うことも可能なので、Linuxでも実用上困ることはない。

TS128NSをUSB接続し、モジュールを組み込んでみた。

```
# modprobe acm
```

lsmodで確認だ。

```
# lsmod
```

Module	Size	Used by
acm	6704	0 (unused)

よし、正常に組み込まれた。ところが、USBViewではまたも赤文字だ。



デバイス情報を確認

さっそくUSBViewで詳細を確認する(画面3)。Manufacturerはサン電子SCC事業部を示している。前述の“USB Vendor / Device IDs list”にはこのVendor IdとProduct Idは載っていない。おそらくサン電子のものなのだろう。

ここで注目したいのが、Device Classの項目だ。ここが“00(>ifc)”になっている。CDC ACM規格に沿うデバイスでは、“02(comm.)”になるはずなのだ。



USB シリアル変換?

今度はUSB シリアル変換ドライバで試してみよう。このドライバを使うには、カーネル構築時に、“CONFIG_USB_SERIAL=m”、“CONFIG_USB_SERIAL_GENERIC=y”を指定する(make xconfigでは、“USB Serial Converter support”と“USB Generic Serial Driver”)。Vendor Id、Product Idを指定し、USB シリアル変換ドライバに認識させることで利用可能になるモデムやTAもあるらしい。さっそくやってみよう。

```
# modprobe usbserial vendor=0x05db
product=0x0004
```

次にデバイスノードを作成する。

```
# mknod /dev/ttyUSB0 c 188 0
```

正しく動作する場合は、/dev/ttyUSB0にTAが見えるようになるので、/dev/modemなどヘシボリックリンクを張ればシリアル接続のTAと同様に使えるようになる。……のだが残念ながらUSB接続をしたTS128NSはこのドライバでも利用できなかった。USBパラダイスへの道は険しいのう。無念じゃ。

```
SUNTAC TS128NS
Manufacturer: Sun Corporation SCC div.
Speed: 12Mb/s (full)
USB Version: 1.00
Device Class: 00(>ifc)
Device Subclass: 00
Device Protocol: 00
Maximum Default Endpoint Size: 8
Number of Configurations: 1
Vendor Id: 05db
Product Id: 0004
Revision Number: 0.01

Config Number: 1
Number of Interfaces: 1
Attributes: 40
MaxPower Needed: 0mA
```

画面3 サン電子独自開発のようだ

USB接続ネットワークケーブル

トライコーポレーションのJusty UTU-02は、USBによるネットワークを構築できるケーブルだ(写真3)。USBハブを使えば最大17台までのネットワークを構築できるという。対応OSはWindows 98 / 98 Second Editionだ。Linuxでの動作保証がないのは承知のうえで挑戦してみた。

今回はいきなり接続してデバイス情報を調べる(画面4)。うげ、“Unknown Device”だと! ManufacturerはProlific Technology, Inc.? USB接続ハードディスクケースで使われていたチップのメーカーだ。Vendor Idの067b、Product Idの0001という情報をもとに“USB Vendor / Device IDs list”で調べるとPL-2302 USB-USB Bridgeだとわかった。ケーブル中間部にPL-2302が内蔵されているようだ(一体成形されており、かち割らないと中を見られない)。

なんと! このコントローラーチップはサポートされているではないか。よしよし。カーネルのコンフィグレーションで“CONFIG_USB_PLUSB=m”を指定する。make xconfigなら“PLUSB Prolific USB-Network driver (EXPERIMENTAL)”で設定

する。EXPERIMENTALというのが気になるところだが……。



**ドライバによって
認識された!**

さて、ワクワクしながらモジュールを組み込んでみよう。

```
# modprobe plusb
```

lsmodで確認すると正しく認識している(当然!)。

```
# lsmod
Module          Size Used by
plusb           6048  1
```

よし、2台のマシンをUTU-02で結び、双方でモジュールを組み込む。次はネットワークの設定だ。2台のマシンのIPアドレスをそれぞれ192.168.1.1と192.168.1.2とする場合、一方のマシンで次のように設定する。

```
# ifconfig plusb0 192.168.1.1
pointopoint 192.168.1.2
```

もう1台のマシンでは192.168.1.1を

192.168.1.2に、192.168.1.2を192.168.1.1にそれぞれ変更して設定すればよい。ifconfigコマンドを引数なしで実行し、ネットワークデバイスplusb0が作られていることを確認しよう。



**世の中そんなに
甘くない**

にんまりしながら、pingしてみると数パケットは返事があるものの、カーネルパニックでLinuxがダウンしてしまった。なぜだっ! FTPでは即座にダウンする。不可解なことに、ダウンするマシンは決まっている。そこで、別のマシンに変えてみたが、これまたダウン。落ちないほうのマシンはSMP構成なのだが、それが関係しているのだろうか? 相関関係は謎のままだ。

やっと動かせるデバイスに巡り会ったと思ったのに。もしかすると、のりぞうは呪われているのかもしれない(編集部注: 日ごろの行いが悪いからですよ)。本来なら転送速度も計測し、その結果をお伝えしなければならなかったのだが、3連敗でのりぞうはすでに満身創痍だ。バトルロイヤルなんて大っキライだ!。

さて、このケーブルを1名の読者様にプレゼントする(いいよね? > 編集部I氏)。欲しい方は、のりぞうにメールをお送りいただきたい。できるだけ、Linuxで使ってね。



写真3
トライコーポレーション Justy UTU-02
PC同士のUSB端子をこのケーブルでつ
ないでお手軽ネットワークを構築できる
……はずだったのだが。

画面4 またまたProlific
Technologyだ

```
Unknown Device
Manufacturer: Prolific Technology Inc.
Speed: 12Mb/s (full)
USB Version: 1.00
Device Class: 00(>ifc)
Device Subclass: 00
Device Protocol: 00
Maximum Default Endpoint Size: 8
Number of Configurations: 1
Vendor Id: 067b
Product Id: 0001
Revision Number: 0.00

Config Number: 1
Number of Interfaces: 1
Attributes: a0
```

複数OSで1台のマシンを有効利用！

タダタダ 無料で始める実践 マルチブート

1パーティションでも大丈夫。
LinuxもWindowsも**ドン**とこい！

新しいOSを使ってみたいけど、マシンは1台しかないしな……。他のディストリビューションを使ってみたいけど、今の環境は壊したくないしな……。などなど、OSをインストールするのは悩みのタネだ。せっかくハードディスクが大容量化しているのだから、OSの数も大容量化(?)、マルチブートしない手はないだろう。メーカー製のWindowsバンドルPCを購入して、Linuxがインストールできないとお嘆きのアナタも、ちゃんとインストールできるので大丈夫。フリーソフトを使って、お金をかけずにマルチブート。ここまでできるんです！



マルチブートのための基礎知識

マルチブートを実現する前に、まずハードディスクの構造を理解しておかなければならない。構造といっても、ハードディスクそのものの仕組みではなく、論理的な部分、つまりソフト的な部分だ。マルチブートを実現するうえで、論理的な構造は必須となるのでじっくり読んでほしい。



OSが起動する仕組み

まず、どのようにハードディスクのOSが起動するのかを簡単に説明しよう。

PCが起動してハードディスクのOSを起動する場合、まずBIOSがハードディスクの最初にあるMBR (Master Boot Record) という領域から IPL (Initial Program Loader) というプログラムを読み出して実行する。IPLはインストールしてあるOSによって書き込まれたもので、このIPLがOSそのものを呼び出すのだ (図1)。



パーティション

ハードディスクが1台しかなければ、そこに2つのOSをインストールしなければならない。そこで必要になるのがハードディスクのパーティション分割だ。

ハードディスクは、2つ以上の区画に分割することができる。パーティションに区切られたそれぞれの領域は、あたかも複数のドライブがあるかのように振る舞う。

複数のドライブがあればOSが複数インストールできそうな気がするだろう。その通り、マルチブートの基本はパー

ティションを分割して複数のOSをインストールするところから始まるのだ。

パーティションには基本パーティション (プライマリパーティション) と、拡張パーティションの2種類がある。

基本パーティションは最も標準的なもので、1つのハードディスクに最大4つまで作成することができる。

一方、拡張パーティションは基本パーティションを拡張し、さらに内部にいくつものパーティションを持つことができるものだ。このため、拡張パーティションを作成する場合は、基本パーティションは最大3つまでしか作成できない (図2)。しかし、内部にいくつものパーティションを作成できるメリットがある。



MBR

OSが起動するにはMBRに格納されているIPLが必要だ。このIPLはインストールしたOSによって書き込まれた唯一のものである。ということは、複数のOSをインストールした場合は、当然のように後からインストールされたOS

によって書き換えられてしまうのだ。このため、前にインストールしたOSは起動できなくなってしまう。

これではマルチブートが実現できない。しかし、逆に考えてみると、それぞれのOSさえ読み込めれば問題ないわけだ。ということは、IPLが両方のOSを読み込めるようになっていければいいことになる。これがブートマネージャ (ブートセクタ) と呼ばれるツール類だ。

Linuxの LILOや Windows 2000の IPLには標準でこれらの機能が組み込まれており、複数のOSを読み込めるようになっている。ただし、機能はそんなに高くないので、後述する専用のブートマネージャを使用するといいたいだろう。



OSの特性

さて、いよいよマルチブートといきたいところだが、さらなる問題がある。それはOSそれぞれの仕様である。OSによってはインストールするパーティションが限定されているなど、各OSによって制約があるのだ。そこで、それぞれのOSの注意事項を説明しよう。

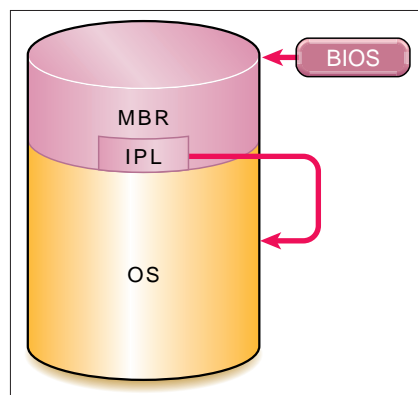


図1 ハードディスクからOSが読み込まれる仕組み

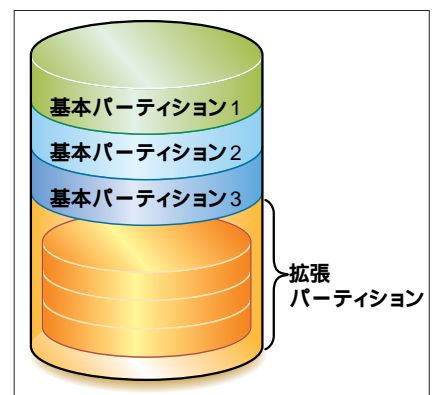


図2 拡張パーティション

Windows 95/98/Me

Windows 95/98/Meは良くも悪くもDOSからの制約を引き継いでいる。あくまでもDOSの拡張版なのである。このため、ここで紹介するOSの中で最も制約が多い。

どのドライブにインストールする場合でも、1台目のハードディスクの基本パーティションでかつ、最初のFATまたはFAT32形式のファイルシステムが必要になる。残念ながら拡張パーティションにはインストールできない。

また、Windows 95 OSR2より前のバージョンでは、2Gバイトを超えるパーティションは認識することができない。

Windows NT/2000

Windows NT/2000はWindows 95/98/Meとは異なり、新たに設計されたOSであり制約は少ない。しかし、いくつかの制約はある。Windows 95/98/Meと同様、1台目のハードディスクでかつ、Windows NT/2000で認識できるFAT/FAT32やNTFS形式のファイルシステムが必要になる。拡張パーティションにインストールすることも可能だ。

Linux

Linuxはインストールするパーティ

ションを最も選ばないOSである。2台目以降のハードディスクにインストールすることもできるし、基本パーティション、拡張パーティションを選ばずインストールできる。ただし、多くのディストリビューションに収録されているLILOの仕様上、ハードディスクの先頭から1024シリンダ(LBAアクセスの場合で約8Gバイト)以降にインストールすることができない(LBAアクセスについては215ページ『Try&Try』を参照)。ただし、TurboLinuxに収録されているLILOは拡張されており、1024シリンダ以降にインストールすることができる。



パーティションを分割する

ハードディスクに初めてOSをインストールするのであれば、インストールするOSに合わせてパーティションを分割することができるが、すでにOSがインストールされている場合はどうすればいいだろうか。特にメーカー製WindowsバンドルPCなどの場合は、パーティションが分割されておらず、すべての領域をWindowsが占めていることもある。このような場合はなんとかしてOSをインストールする領域を確保しなければならない。

インストールされているOSがWindows 95/98/Meの場合は、フリーソフトのFIPSというツールを使ってパーティション領域を縮小することができる。こうしてできた空いた部分に新しいパーティションを割り当て、Linuxをインストールすることができるようになる。

残念ながらFIPSはWindowsのFAT/FAT32以外のファイルシステムでは利用できない。NTFSなどのファイルシステムを使用している場合は、再度インストールしなおすが、市販の

パーティション操作ツールを使用する。

FIPSを使う

FIPSはハードディスクのうしろのほうの、ファイルが使っていない領域を新たなパーティションとして分割することができるツールだ。このため、ハードディスクのうしろにファイルが使っていない領域が必要になる。ハードディスクにパーティションを分割するだけの十分な領域があったとしても、ハードディスクのうしろの部分でファイルが使用していた場合は分割できないので、あらかじめファイルをハードディスクの前のほうに集めておく必要があるのだ。

なお、FIPSは重要なパーティション情報を書き換えるソフトであるため、念のためハードディスクのバックアップを取ってから作業すること。作業する前にFIPSに付属しているドキュメントに一通り目を通しておくほうがいいだろう。以下のWebサイトでもFIPSに関するドキュメントを参照することができる。

<http://www.igd.fhg.de/~aschaeffe/fips/> (英語)

<http://members.nbci.com/transl8r/FIPS20J/fips20J.html> (日本語)

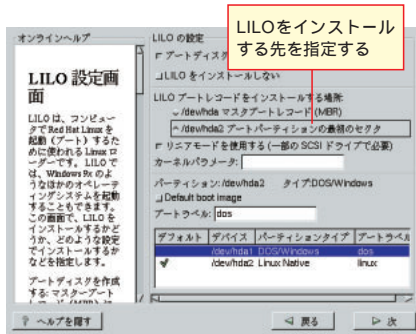
それでは、どのようにしたら新たなパーティションが作成できるかを、順を追って説明していこう。

FIPSはWindows上で使用してはならない。必ずDOSモードで使用するようにしなければならない。そこで、起動ディスクの作成から行う。

起動ディスクを作成するには、コントロールパネルの[アプリケーションの追加と削除]を実行し、[起動ディスク]タブにある[ディスクの作成]ボタンをク



画面1 起動ディスクを作成して作業する



画面2 LILOをパーティションの先頭セクタに書き込むように指定

リックする。なお、起動ディスクはほとんど空き容量がないので、もう1枚フロッピーディスクを用意し、FIPSをコピーする。

FIPSは多くのディストリビューションの/dosutils/fips20に収録されているが、今月号の付録CD-ROMのDISC 1の/Linuxmag/MultiBoot/fips20.zipにも収録してある。このアーカイブの内容を展開してフロッピーディスクにすべてコピーすればよい。

ここまで準備できたら、ハードディスクのファイルを前のほうに集める。

まず、Windowsのスキャンディスクを使って、ハードディスクに不良がないかをチェックする。もし、エラーがあれば修正しておく。

次に、Windowsのデフラグを使って、ファイルをハードディスクの前のほうに集める。ただし、デフラグではシステムファイルやリードオンリー属性のファイルの配置を変更することはできない。もし、ハードディスクのうしろのほうにこれらのファイルがあった場合は、十分な領域が確保できない可能性がある。このような場合は、これらのファイルを使用しているソフトをいったんアンインストールする必要がある。また、Windowsのスワップファイルがハードディスクのうしろのほうにある場合も同様で、この場合は、Windowsのシステムのプロパティで、

仮想メモリを使わない設定にしておく。デフラグによってハードディスクのうしろのほうに空き領域ができればよいよFIPSを使用する。

FIPSでは多くのメッセージが表示され対話式に作業を行うが、誌面の都合上、すべてを掲載することはできない。そこで、大まかな流れと操作方法を説明する。

Windowsを終了して先ほど作成したフロッピーディスクで起動し、FIPSを実行する。

A:>¥fips

FIPSが起動したら、以下の手順で操作する。

最初に実行したOSが検知され、「Press any Key」と表示される（任意のキー）

現在のパーティションテーブルが表示され、「Press any Key」と表示される（任意のキー）

ブートセクタ（MBR）の情報が表示されるので、現在のブートセクタのバックアップを保存する（Yキーを2回）

カーソルキーを使って、パーティションのサイズを設定する

新しいパーティションテーブルが表示される（Cキー）

新しいブートセクタの情報が表示される（Yキー）

以上でパーティションの分割は終了だ。あとは、システムを再起動して新しく作成した領域をインストールするOSでフォーマットするだけだ。もし、TurboLinux以外のディストリビューションをインストールするのであれば、作業のときに、新しいパーティシ

ョンが先頭から8Gバイト以降にならないように注意しよう。

なお、パーティションを元に戻す場合は、でバックアップしたパーティション情報を元にFIPSのツールであるRESTORRB.EXEを使う。



Linuxのインストール

パーティションが分割できたら、いよいよもう1つのOSのインストールだ。

マルチブートであっても、Linuxであれば普通にインストールすることができる。Linuxのインストーラが自動的にLILOを設定し、「LILO」のプロンプトが表示されているときに「DOS」と入力すればWindowsが起動できる。しかし、この方法では2つのOSしかブートできない。そこで、ブートマネージャを使用しよう。

ブートマネージャを使用するためには、LILOのインストール先について考慮する必要がある。

LILOは単なるIPLではないので、そのまま他のブートマネージャに置き換えることはできない。LILOがないと、カーネルイメージを見つけることができなくなるのだ。しかし、LILOをMBRに書いてしまうとブートマネージャを書き込む場所がなくなってしまふ。

そこで、Linuxをインストールする際に、LILOをLinuxをインストールするパーティションの先頭のセクタに書き込むようにする。これは、Linuxのインストーラでカスタムセットアップを選択することで指定できる（画面2）。これにより、ブートマネージャがLILOを読み込み、無事Linuxが起動できるのだ。また、必ず起動ディスクを作成しておくこと。起動ディスクがないと、次で紹介するブートマネージャをインストールするまでLinuxが起動できないからだ。

難しいのはイヤ。誰でも簡単マルチブート

GAG

<http://raster.cibermillennium.com/gageng.htm>

GAGはコマンドラインからの操作が不要で簡単にセットアップできるブートセクタだ。GAGを設定する前に、まず空のフロッピーディスクと本誌付属CD-ROMに収録したGAGのアーカイブファイル（ファイル名gageng31.zip）を用意しよう。gageng31.zipをWindows上の適当なディレクトリにコピーして展開すると、gageng31というディレクトリが生成される。

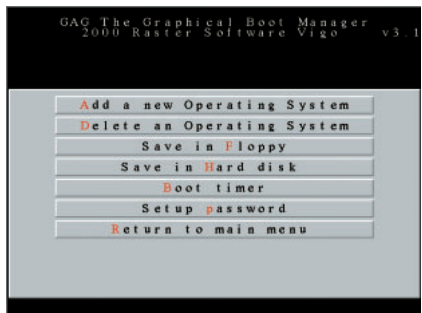
次にDOS窓を立ち上げてgageng31ディレクトリに移動する。

```
rawrite2 -f gag.ima -d a:
```



画面3 メインメニュー

GAGをインストールした直後はこの画面に「Boot from disk Key 1」と「Setup GAG Key S」の2つのメニューがある。GAGから起動するOSを追加すると、この画面にメニューが追加される。



画面4 設定メニュー

この画面でGAGから起動するOSを追加したり、設定内容を保存したりする。OSを追加する場合は「Add a new Operating System」で赤くなっている文字「A」をタイプする。

として再度Enterキーを押すと、GAGのフロッピーディスクが作成される。



GAGからWindowsを起動してみよう

作成したフロッピーディスクをセットしてマシンを再起動するとGAGのメインメニューが立ち上がる（画面3）。

ここで「S」をタイプして設定画面に進み（画面4）、「A」をタイプしてGAGから起動するOSを追加しよう。

ハードディスクのパーティション情報が表示されたら（画面5）「B」をタイプして「MS-Windows FAT32」を選択する。するとメインメニューに表示するOS名と、設定ユーザーを制限するためのパスワードを質問されるので、OS名を「Windows 98」などとし、パスワードを入力しないでEnterキーを押す。

最後に「C」をタイプして、Windows 98に割り当てるアイコンを選択し（画面6）「F」をタイプして設定内容をフロッピーに保存しよう。「R」をタイプするとWindows 98のメニューが表示されるので、「2」をタイプしてWindowsが起動すれば、GAGの設定は成功である。



画面5 ハードディスクのパーティション情報

GAGに起動するOSを追加したり削除するときに表示される画面。Windowsパーティションは「MS-Windows FAT32」、Linuxパーティションは「83h Linux EXT2」と表示される。



今度はLinuxを追加しよう

今度はGAGにLinuxを追加しよう。WindowsをGAGに登録したときと同様に、GAGのフロッピーディスクをセットした状態でマシンを再起動する。メインメニュー（画面3）から「S」「A」とタイプしてパーティション情報の画面まで進む。すると「83h Linux EXT2」が表示されるので、「C」をタイプしてこれを選択する。最後にOS名を「Linux」にして、アイコンにペンギンでも割り当てる。「F」で設定内容をフロッピーへ保存して、「R」でメインメニューに戻り、「3」をタイプすればLinuxが起動するはずだ。

毎回フロッピーディスクからOSを起動するのがわずらわしいというユーザーは、メインメニューで「H」をタイプしてGAGをMBRにインストールしよう。

GAGはフロッピーディスクでも運用できる手軽さと、グラフィカルな画面が魅力のブートマネージャだ。MBRを間違えて書き換えてしまったときなどにも役立つので、フロッピーディスクに用意しておこう。



画面6 OSに割り当てるアイコン

ここで選択したアイコンが、メインメニュー（画面1）に登録される。Linuxへは「D」を、Windowsへは「C」を割り当てるのが自然だが、Linuxに「F」を割り当てるのもアリだ。



OSを自動認識するブートマネージャ

Smart Boot Manager

<http://btmgr.sourceforge.net/>

Smart Boot Manager (以下、SBM) は、フリーでありながらインストール済みOSを自動認識してメニューに表示する優れたブートマネージャだ。また、テーマファイルを変更することで、キーバインドなどを自由に設定できる自由度の高さも魅力だ。



インストール

SBMはソースで提供されているため、インストールする前にコンパイルする必要がある。コンパイルには、gccとアセンブラNASMが必要になる。NASMは以下のサイトからダウンロードすることができる。RPM形式で提供されているのでインストールは簡単だ。

<http://www.web-sites.co.uk/nasm/what.html>

SBMのソースファイルは今月号のCD-ROMに収録されているので、これを展開してコンパイルする。コンパイルは、単にmakeするだけだ。

```
$ su -
# tar zxvf /mnt/cdrom/Linuxmag/MultiBoot/btmgr-3.6-2.tar.gz -C /tmp
```

```
# cd /tmp/btmgr-3.6-2
# make
```

makeが終了したら、releaseというディレクトリに必要なファイル一式が作成されるので、以下の手順でSBMをMBRに書き込む。なお、make installとすることもできるが、これは必要ない (make installしてもSBMがMBRに書き込まれるわけではない)。

```
# cd btmgr-3.6-2/release
# ./sbminst -t theme-us -d /dev/hda
```

ここで指定している「theme-us」というファイルはキーバインドなどが定義されたテーマファイルだ。テーマファイルはあらかじめいくつか用意されている。また、自分で変更することもできるが、アセンブラソースを変更する必要がある (manager/themes/* .asm)。

sbminstを実行すると、本当に書き込んでよいかどうかの確認を求められるので、Yキーを押してSBMをMBRに書き込む。インストール作業は以上だ。



使い方

システムを再起動すると、SBMのOS選択メニューが表示されるようになる。メニューを選択してEnterキーを押すと、そのOSが起動する (画面7)。

残念ながら、OSの名前までは自動認識できないので、わかりやすい名前に変更するといいたいだろう。

カーソルキーでパーティションを選

択し、F3キーを押すと名前が変更できる。変更後、F2キーを押して変更を保存する。

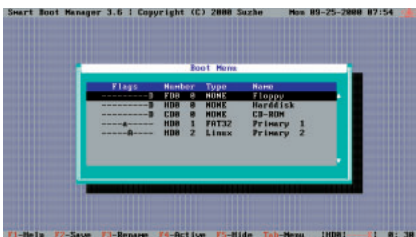
メニューの表示順はCtrl + Uキーで変更することができ、Ctrl + Dキーで削除できる。

SBMの独自のおもしろい機能にキーストロークがある。キーストロークは、OSが起動する際に自動的に文字が入力される機能で、最大13文字まで入力できる。

たとえば、同じパーティションのLinuxに起動メニューを2つ作成し、それぞれ「linux 3」「linux 5」とキーストロークを設定しておけば、コンソールとX Window Systemがメニューから選択できるようになる。キーストロークを設定するには、OSを選択し、Ctrl + Kキーを押し、入力するキーをタイプしたあと、Scroll Lockキーを押す。これで、そのOSが起動したときにタイプした文字が自動的に入力される。

ただし、標準のLILOでは起動時にキーがクリアされてしまうのでこの機能は使えない。LILOを「-DNODRAIN」オプションを付けてリコンパイルする必要がある。リコンパイルの設定の仕方はSBM付属のドキュメントに詳細があるので参照してほしい。

SBMは非常に高機能で、一定時間経過すると自動的にOSが起動するように設定したり、パスワードを設定したりできる基本的な機能から、他のOSからパーティションを隠す機能も持っている。これらの機能はTABキーを押すことで表示されるメニューから設定することができる。なお、F1キーを押すとヘルプが表示されるので参照してほしい。



画面7 SBMのOS選択メニュー。カーソルキーで選択し、Enterキーで起動する

日本製のコンパクトなIBM-IPL完全上位互換IPL

Extended-IPL

<http://www.tsden.org/takamiti/extipl/>

Extended-IPL (以下、EXT-IPL) は、日本で生まれたブートマネージャで、長い歴史がある。元々はMINIX用に作られたものだが、その後、改良が加えられ現在の形になった。



インストール

EXT-IPLもソースで公開されているため、インストールする前にコンパイルが必要だ。コンパイルには、NASMが必要になる。

EXT-IPLのソースファイルは今月号のCD-ROMに3種類のアーカイブ形式で収録(内容は同じ)されているので、これを展開してコンパイルする。なお、展開先のdocフォルダに日本語の詳しいドキュメントが収録されているので参照してほしい。

```
$ su -
# tar zxvf /mnt/cdrom/Linuxmag/
MultiBoot/extipl422.tar.gz -C /tmp
# cd /tmp/extipl/src
# make
```

makeが終了したら、srcディレクトリにextiplというバイナリが作成される。あとは、このプログラムを使ってMBRを操作する。

MBRを書き換える手順だが、EXT-IPLではフロッピーディスクでのテストのあと、MBRを書き換えるように推奨している。

フロッピーディスクでのテストは、フォーマット済みのフロッピーディスクをドライブにセットし、

```
# ./extipl fdtest /dev/fd0
```

とする。これでフロッピーディスクを使ってマルチブートが可能になる。

ここで、フロッピーディスクをセットしたまま、システムを再起動する。このとき、Shiftキーを押したまま起動するようにする。Shiftキーを押すことによって、EXT-IPLが有効になるのだ(画面8)。Shiftキーを押さなかった場合は、アクティブパーティションのOSが起動する。

ここで、パーティション番号を数字で入力し、Enterキーを押すことでそのパーティションのOSが起動する。

フロッピーディスクで問題がなければ、MBRの書き換えを行う。以下のようにしてMBRにEXT-IPLを書き込む。

```
# ./extipl install /dev/hda
Enter file name to save:
```

ここでは、現在のMBRのバックアップを保存するファイル名を入力する。

あとは、ハードディスクから起動する際にShiftキーを押すようにすれば、先ほどのメニューが出るようになる。



使い方

拡張パーティションにインストールされたOSを起動する場合は、拡張パーティションの番号を入力する。これによって拡張パーティションの内容が表示されるようになる。2台目のハードディスクから起動するには0を入力する。

ここまで読むと「使いにくいのではないか」と思われるかもしれない。実は、EXT-IPLは他のブートマネージャとは指向が違うのである。

EXT-IPLでは、起動するOSを固定することができる。つまり、いつもはLinuxを使っているが、たまにWindowsを使うというような場合に便利なのだ。Shiftキーを押して毎回OSを選択するのは本来の使い方ではない。

EXT-IPLのパーティション一覧が表示されているとき、いつも使うOSの番号を入れ、Endキーを押す。これにより、通常起動するOSが固定される。次回からはそのOSがパーティションの選択なしで起動することになる。

別のOSを起動したいときは、Shiftキーを押しながら起動すればいい。

Part	Sys
1	S0B
2	S83
3	S05
4	

Boot#0:0:1

パーティションの形式

- S0B FAT32
- S05 拡張領域
- S07 HPFS / NTFS
- S82 Linux SWAP
- S83 ext2
- S86 NTFSボリューム
- S87 NTFSボリューム

パーティションの番号と
パーティションの形式

ブートするパーティション番号
対象パーティションテーブル番号
ハードディスクユニット番号

画面8 EXT-IPLのOS選択メニュー



カーネルのブート、一手に引き受けさせてもらいます GRUB

<http://www.gnu.org/software/grub/>

ここまで紹介した3種類のツールがOSごとのブートローダを呼び出すブートマネージャであるのに対し、このGRUBはOSのカーネルを直接読み込むOSブートローダである。

ブートローダにLILOを使う場合、Linuxカーネルを再構築する際にLILOの再インストールが必要だが、GRUBを使う場合はこれが不要で、8Gバイトを超えたパーティションにあるOSも起動可能だ。

さらにGRUBのフロッピーディスクを作成しておけば、起動しなくなったLinuxマシンの復旧にも使えるなど、ほかのマルチブートマネージャにない柔軟性を備えている。

現時点ではLinux Mandrake 7.1とCaldera Linux Technology Previewの標準ブートローダとして採用されており、今後多くのディストリビューションでGRUBの採用が予想される。



GRUBのインストール

まずLinuxを起動して付録CD-ROMに収録したGRUBのアーカイブファイルをコピーする。

```
$ su -
# mount /mnt/cdrom
# cd /tmp
# cp /mnt/cdrom/Linuxmag/MultiBoot/grub-0.5.95.tar.gz .
```

次にコピーしたファイルを展開してコンパイルしMBRにインストールする。

```
# tar zxvf grub-0.5.95.tar.gz
# cd grub-0.5.95
# ./configure && make
# make install
# grub-install /dev/hda
```



GRUBの設定

設定ファイルのサンプルがGRUBのアーカイブファイルに含まれているので、/boot/grubにコピーして編集しよう。

```
# cd /boot/grub/
# cp /tmp/grub-0.5.95/docs/menu.lst .
```

設定ファイル(リスト1)では「# For booting Linux」以下がLinuxに関する設定箇所だ。「title」にはGRUB

のメニュー(画面9)に表示するOS名を記述する。ここではテスト環境に合わせ「Red Hat Linux 6.2」とした。

次に「root」の欄を設定する。GRUBではプライマリマスタディスクに0を、プライマリスレーブディスクに1を割り当てる。またパーティションを0から数え始めるので、/dev/hda2のLinuxパーティションは(hd0,1)となる。

さらに「kernel」の欄を「/boot/vmlinuz root=/dev/hda2」と記述する。Windowsに関する設定はデフォルトのままでもよいだろう。

最後に「# For booting the GNU Hurd」や「# For booting FreeBSD」などで始まるLinuxとWindows以外のOSの設定項目を削除するか、行の先頭に「#」を入れて無効にする。

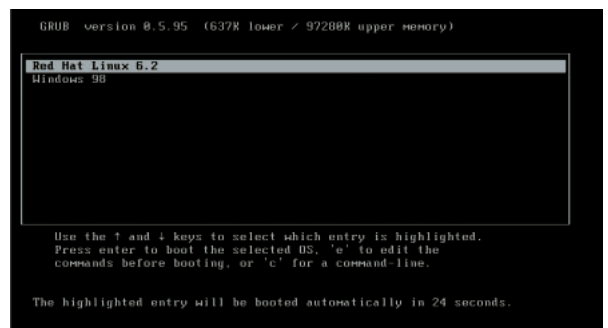
GRUBのセットアップはこれで終わりだ。システムを再起動すれば、GRUBのメニュー(画面9)が表示されるので矢印キーでOSを選択してEnterキーで起動しよう。

設定が難しいといわれているGRUBだが、思いのほか簡単にインストールすることができるので、8Gバイトの壁で困っている人にお勧めだ。

リスト1 /boot/grub/menu.lstの設定例

```
# For booting Linux
title Red Hat Linux 6.2
root (hd0,1)
kernel /boot/vmlinuz root=/dev/hda2

# For booting Windows NT or Windows95
#title Windows NT / Windows 95 boot menu
title Windows 98
rootnoverify (hd0,0)
makeactive
chainloader +1
```




画面9 GRUBの起動画面
OS名を「↑」キーと「↓」キーを使って選択しEnterキーでOSを起動する。

アップデート最新版をCD-ROM収録！

Vine Linux 2.0 for PowerMac

文：松林弘治

Text：MATSUBAYASHI Kohji



Vine LinuxのMacintosh対応版「Vine Linux 2.0 for PowerMac」が、今年7月にリリースされた。Vine Linux 1.0の発表当初からMacintosh版開発の計画はあったが、Red Hat LinuxをベースにしたIntel版Vine Linuxとの高い互換性を持たせるために時間がかかった。

今月号の付録CD-ROMには、7月のFTP版公開後にアップデートされたパッケージを取り込んだ最新版を収録している。インストール直後から快適な日本語環境を実現し、多くのユーザーに支持されているVine Linuxを、Macintoshで体験してみよう。

Photo：Shuichi Mito (Dee)

Vine Linux for PowerMacの特徴

Vine LinuxのPPC版、Vine Linux 2.0 for PowerMac (以下Vine 2.0/PPC)が、2000年7月3日に公開されました。Intel版とまったく同様に、各パッケージが隔々まで日本語化/国際化されており、インストール直後から快適な環境を楽しめるようにデフォルトの設定ファイルも整備されています。

つまり、PC/AT互換機ユーザーのみが享受してきた、使いやすいと評判のLinuxディストリビューションを、PowerMac/iMac/PowerBookなどで利用することができるのです。

Intel版とは一部異なるパッケージ
先行して4月に公開されたIntel版2.0より遅れてリリースされたこともあり、

アーキテクチャ	Intel (x86)	PPC
kernel	2.2.14-1v16 (2.2.14)	2.2.14-19v13 (2.2.15pre19-pmac)
glibc	2.1.2-17v12	2.1.3-4v11
gcc/egcs	egcs-1.1.2-24v11	gcc-2.95.3-0.2v11
XFree86	3.3.6-13v13	3.3.6-13v18
gimp	1.1.17-0v11	1.1.19-0v11
Wanderlust	2.2.15-1 (Beta版)	1.1.1-0v12 (正式リリース)
PostgreSQL	6.5.1_jp-1	6.5.3_jp-2

表1 Vine 2.0のIntel版 (FTP版) よりPPC版のほうが新しい主なパッケージリリース時期の違いにより、一部のパッケージはより新しくなっています。一部はPPC向け修正や、ソースが共通化できないPPC専用パッケージのための違いも含まれます。

パッケージ名	簡単な説明
fbset	フレームバッファ設定プログラム
gtk+perl	perlからGTK+を使用するライブラリ (PPC版Xインストーラで使用)
hfsutils	HFSパーティションを操作するプログラム集
jfbterm	日本語フレームバッファコンソール
mol	Mac-On-Linux (Linux上でMac OSを動作するネイティブエミュレータ)
netatalk	AppleTalkを扱うプログラム (LinuxをMac OSのファイルサーバにできる)
pdisk	Macintosh形式パーティションマップ編集ツール
pmac-utils	画面設定やADBデバイスの設定を行うプログラム集
yaboot	NewWorld ROM機用Linuxブートローダ

表2 PPC版のみに含まれる主なパッケージ
ハードウェアやコンソールの違いによるパッケージ。Mac-On-LinuxなどPowerMac専用のパッケージが収録されています。

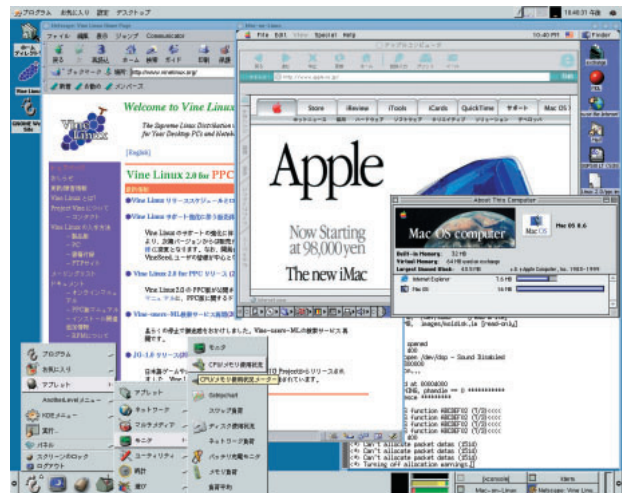
一部のパッケージはIntel版よりも新しいものが採用されています (表1)。また、PPC版には必要のない一部のパッケージ (lilo、kon2、liloなど) が外された代わりに、PPC版専用のパッケージが収録されています (表2)。

特にMOL (Mac-On-Linux) が収録されたことにより、Linux内からMac OSを起動することができるので、少々アプリケーションならば、わざわざ再起動してMac OSに切り替える必要がありません (画面1)。

動作対象機種

Vine/PPCは、CPUにPowerPCを搭載し、Open Firmwareを備えたMacintoshで動作します。6100、7100、8100といったNuBus搭載PowerMacでは (現時点では) カーネルが対応していないため、動作させることができません。また5300、1400のPowerBookも同様の理由で動作しません。

逆に最新機種は、iMac DV、G4 AGP (Sawtooth)、PowerBook 2000 (FireWire、Pismo) などまで動作確認しています。また、Vine 2.0/PPCリリース後に発売されたG4 CubeやG4 MPでも動作したと報告がありますが、



画面1 Vine 2.0/PPC上で動作するMac-On-Linux
Intel版のVMwareのように、Linux上でMac OSを動作させることが可能なネイティブエミュレータ。Vine 2.0/PPCにはバージョン0.48を収録しています。フルスクリーンモードでMac OSを動作させることもできます。



画面2 「ドライブ設定」(Drive Setup) によるパーティション設定
Mac OSに標準付属するディスクフォーマット/パーティションングツールを使って、Linux用にパーティションを抽出する必要があります。この例では4GバイトのハードディスクをMac OS用に1Gバイト、Linux Swap用に128Mバイト、Linux / (root) 用に2.9Gバイト割り当てるように再フォーマット/パーティションしようとしています。

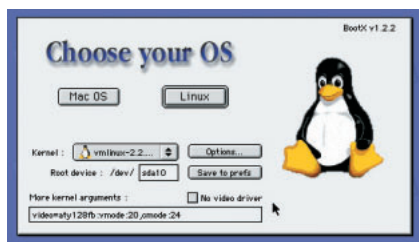
Vine 2.0/PPC付属のカーネルより新しいカーネルでないと不具合があるかもしれません。これらの機種でのテストは次期リリースまでに行いたいと思います。

Intel版とはちょっと違うインストール方法

いったんインストールしてしまえばIntel版とほぼ変わらない操作感を味わえますが、インストールの仕方はやや異なります。以下、インストーラを起動するまでに行う作業を概説します。

Linux用パーティションを確保する

Intel版ではFIPSというパーティション分割ツールを使って、既存のFAT/VFATパーティションの一部をLinux用に切り出す作業から始まりますが、Mac OS用には同等のツールは残念ながら用意されていません（売り物のソフトならそのような作業が行えるものがあるかも知れませんが……）。



画面3 BootXコントロールパネル
Mac OS起動途中に現れるこのパネルからLinuxを起動します。



画面4 Vine Linux 2.0 for PowerMacのログイン画面 (wdm)

そのため、ハードディスクが1台しかない場合には、まずハードディスク上のMac OS用ファイルを全部バックアップしたあと、フォーマットしなおす必要があります。バックアップが済んだらMac OSのインストールCD-ROMから再起動し、ドライブ設定 (Drive Setup) アプリケーションを使ってパーティションを切り直します。Swapに128Mバイト程度、/(root)に1.5Gバイト以上を割り当てるとよいでしょう (画面2)。

パーティション設定が済んだら、バックアップしておいたMac OS用ファイルを、やや小さくなったMac OS用パーティションに戻します。Macを買った直後であれば、開封直後にこれらの作業を行っておき、その後「システムリストアCD」を使ってやや小さくなったMac OS用パーティションにMac OSシステムを入れると簡単です。

もちろん、Linux専用にハードディスクを別途用意できる場合は、これらの作業は必要ありません。なお、Linux内からMac OS用パーティションを読み書きすることができませんが、HFS+ (Mac OS拡張) タイプのパーティションにはアクセスできません。

そこで、Mac OSとLinuxのファイルやりとり用に、別途小さなHFS (Mac OS標準) タイプのパーティシ

ョンを作っておくとよいでしょう。LinuxのHFSサポートは完全ではないようで、最悪の場合、Linuxから書き込み/削除を繰り返すとパーティションが壊れる可能性があることが知られていますから、ふだんMac OSで使うパーティションとは別に、小さな専用パーティションを用意しておくほうが安全です。

いざインストール開始

ここまでできたら、CD-ROMからインストーラを起動します。Vine 2.0/PPCのCD-ROMをドライブに入れておき、Macを再起動します。この時[C]キーを押し続けるとCD-ROMから起動し、直接インストーラに入ります。インストール作業の詳細については、別セクションの解説を参照して下さい。

ブートローダの設定

Intel版では、インストーラ内からLILOのインストールと設定が半自動的に行われますが、PPC版では、インストール終了後にブートローダの設定を行う必要があります。なお、LILOとは異なり、読み込むカーネルのサイズには制限がありませんので、ほとんどのPowerMac版のカーネルは、よく使うモジュールをカーネル組み込みにして

Column

他のLinux PPC ディストリビューションとの違い

現在、日本向けに公開/販売されているPPC向けディストリビューションには、Linux PPC 2000日本語版 (Amulet)、Linux for PPC Japanese Edition (マインド)、Linux2000G (ホロン) などがあります。

Amulet版のみが、Linux PPC英語版に対する日本語パッケージ集を別途追加する形態

をとるものの、そのほかは最初から日本語対応パッケージが組み込まれています。また、Amulet版はXインストーラ、そのほかはテキストベースインストーラなどの違いがありますが、Vine 2.0/PPCを含めてパッケージの細かい違いを除いては、その成り立ちからいっても大きな差はないと言ってよいと思います。逆に言えば、Intel版譲りの快適な日本語環境がインストール直後から使用できるのがVine/PPCの最大の特徴となります。

おり、そのため比較的大きなサイズ（3Mバイト程度）になっています。

BootX（OldWorld Mac向け）
ページG3までの古いPowerMacでは、BootXというコントロールパネルを使いLinuxを起動します（画面3）。Linuxの/（root）パーティション名、カーネルに渡す引数などを指定します。
システムフォルダ内に「Linux Kernels」フォルダを作り、その中にカーネルファイルを入れておけば、ポップアップメニューから起動カーネルを切り替えることができます。

yaboot（NewWorld Mac向け）
iMac以降の機種では、BootXが必ずしも動作しなくなりましたので、まったく違う方法で起動します。このyabootを使うと、Mac OSに一切入ることなくLinuxを起動させることができます。設定ファイルyaboot.confの書式もLILOに似たものとなります（リスト1）。

ただし特殊な方法で行わない限り、現時点ではOpen Firmwareの設定が必要となります。Open Firmwareに入るには、Macの再起動直後にキーボードから、Command + Option + O + Fを押し続けます。あらかじめシステ

ムフォルダ内にyabootとyaboot.confを入れておいて、

```
boot hd:,\\yaboot
```

と入力することによりyabootを起動します。

そのほかにも、起動時にスペースキーを押すとLinuxが、押さないとMac OSが起動するようにする設定方法や、Optionキーを押しておいて起動パーティションを選択する方法（最新機種のみで有効）について、付録CD-ROMに収録されたマニュアルに詳しく書かれています。

リスト1 yaboot.confのサンプル

```
init-message = "\nWelcome to Vine Linux!\n\nHit <TAB> for boot
options.\n\n"
timeout=300
default=linux-fb

image = hd:8,\\Vine_Default
label = linux
root = /dev/hda10
novideo

image = hd:8,\\Vine_Default
label = linux-fb
root = /dev/hda10
append = " video=aty128fb:vmode:20,cmode:24"

image = hd:8,\\vmlinux-2.2.17pre20-ben3
label = linux-nox
root = /dev/hda10
append = " 3 video=aty128fb:vmode:17,cmode:24"
```

Column

OldWorld Macと NewWorld Macの違い

1984年の初代Macintoshから、Mac OS用APIはMac OS Toolboxとして、オンボードのROMに納められてきました。このROM上には、電源投入時のサウンドや、起動時のハードウェアテストルーチンなども納められています。ところが1998年に発売されたiMacから、Mac OS Toolboxはシステムフォ

ルダ内のMac OS ROMファイルに納められるようになり、Mac OS起動時にRAMに読み込まれるように変更されました。これがNewWorld ROMです。これと同時にMac OSの起動シーケンスも大きく変更されたため、必ずしもNewWorld機ではBootXが動作しなくなっていました。そのため、Open Firmwareから直接Linuxカーネルを読み込み起動するブートローダが開発されました。それがyabootです。

XFree86の設定

Vine for PPCが動作するほとんどの機種では、Xpmacサーバを使うことができます（インストール直後に自動的にXpmacサーバが起動します）。設定は、/etc/X11/Xpmac.confファイルを介して行います。

Intel版のXほどではないものの、まずまずの速度で動作しますし、何より従来の複雑な設定ファイルから開放されるのが一番のメリットです。そのほか、フレームバッファ上で動作するXF86_FBDevサーバもあります。これはIntel版と同様に、Xconfiguratorプログラムを使って設定します。ホイールマウスなどを有効に使いたい場合は、こちらを選択するとよいでしょう。

ただし、iMac DVだけはXpmacサーバがうまく動作しないため、別のXサーバがインストールされるようにしてあります。

マウスだけは要注意

Macに標準付属のマウスはすべて1ボタンとなっています。しかし、Xを

使ううえでは何かと不便です。USB搭載機種の場合は3ボタンマウスが安価で入手可能ですので、そちらを利用するのがよいのですが、ADBしか搭載していない機種の場合は2ボタン/3ボタンマウスは入手困難です。

特に現在流通しているADBの2ボタンマウスの中には、右ボタンがControl + 左ボタンにハードウェア的に固定されているものが多く、改造しない限りX上で2ボタンマウスとして使用できません。

最悪1ボタンマウスしか用意できない場合を考えて、Xpmacサーバでは、Option + F1で中ボタンを、Option + F2で右ボタンをエミュレートするようにしてあります。このキーボードによるマウスボタンのエミュレートは、前述の設定ファイル(Xpmac.conf)により割り当てを変更することも可能です。

Mac周辺機器の扱い

Intel版とほぼ同様に各種周辺機器を利用することができます。ただし、Apple社製のインクジェットプリンタなどはプリンタドライバがないので、Linuxから使用することはできません。

そのほか、USB接続のフロッピードライブやMOドライブなども利用できると報告されています。また、内蔵モデムも問題なく動作するようです(幸いMacの内蔵モデムはソフトウェアモデムではありません)。

一部周辺機器は要注意

Intel版Linuxで動作が確認されているからといって、PPC Linuxから問題なく使えるとは限りません。特にPCMCIAカードは注意する必要があります。また、サードパーティのネットワークカードやSCSI-IDE変換ボードな

ども要注意です(カーネルにパッチを当てれば動くという報告もあるようです)。これらの対応状況は本当に日進月歩ですので、メーリングリストやWebの検索を有効に使って情報を集めましょう。

今後のVine for PPC

現在、10月~11月リリースを目標としてVine Linux 2.1の開発が進んでいます。Vine 2.0/PPCは、FTP版のみのリリースでしたが、2.1ではPPC版も製品版として販売される予定です。

Vine Linux 2.1では、同時にSPARC版/Alpha版もリリースされることになっており、製品版には、Intel/PPC/SPARC/Alpha各アーキテクチャ用のものが、1パッケージに同梱されることになっています。

Vine Linux 2.1の改良点

そのVine Linux 2.1では、2.0に比べて以下のような変更が行われる予定です。

- ベースをRed Hat 6.1 6.2に変更
- 最新カーネル(2.2.16または2.2.17)の採用
- より新しいglibc 2.1.3の採用(PPC版では移行済み)

- Sendmail PostFix (PPC版では移行済み)
- wu-ftpd ProFTPD
- ncftp lftp
- lynx w3m
- その他マイナーバージョンアップ、およびバグフィックス

PPC版でも、より新しい機種に対応したカーネルの標準収録やネットワーク経由でのインストール対応など、さらなるブラッシュアップが行われる予定です。

XFree86 4.0やkernel 2.4、glibc 2.2などは次期メジャーバージョンアップ版であるVine Linux 3.0で採用される予定となっています。なお、次期Vine Linuxのロードマップ(<http://vinelinux.org/roadmap.html>)が公開されています。

Vine開発協力を随時募集

Intel版に比べると、他のアーキテクチャ版の開発に携わる人数はどうしても少ないため、テストもIntel版ほど十分に行えなかったり、パッケージ作成の人手も不足しがちです。Vine Linuxの開発版であるVineSeed(<http://vinelinux.org/vineseed.html>)に、ふるってご参加ください。皆さんと一緒にVine Linuxをより良いものにしていけたらと思います。

Column

Intel版とのソースの共通化、Red Hat版とほぼ同じ環境の実現

インストーラだけはIntel版のAnacondaとは大きく異なり、Linux PPC 2000のXインストーラをベースに日本語化したものですが、パッケージのほぼすべてはIntel版と共通です。

x86系CPUとPowerPCではバイトオーダが異なる(x86はLittle Endian、PPCはBig Endian)ことに加えて、ハードウェア的にも

さまざまな違いがあります。またPPC向け基本ライブラリやコンパイラも、Vine 2.0/PPCに収録したものはx86版と若干異なります。

これらの違いをすべて加味し、パッチなどを整備することで可能な限りIntel版とソースを共通化し、その結果Intel版とほぼ同じ環境を実現することができました。これらの作業の多くは、広く国内外でPPC Linuxに携わっている幾多の開発者の成果を取り込んだものです。

Vine Linux 2.0 for PowerMac(アップデート版)のインストール

Vine 2.0/PPCのインストーラは、ix86版に比べるとステップ数が少ないため、インストール前に行うハードディスクパーティションの準備を除いては、インストール作業自体は特に問題なく終わると思います。

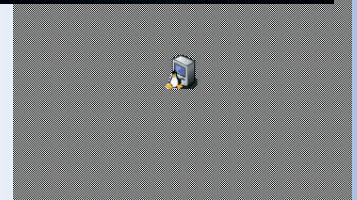
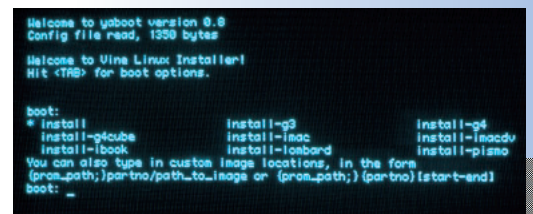
まず、CD-ROMに納めているHTMLマニュアルの「イン

ストールの準備」セクション以下にある、「ハードディスクの構成を調べる」、「ハードディスクのスペースを確保する」を熟読し、パーティションを設定しておいてください。操作を間違えると、最悪Mac OS用パーティションのデータを消失してしまうので、注意して行ってください。

CD-ROMからの起動

Mac OSが起動しているPowerMacのドライブにVine Linux 2.0 for PowerMacのCD-ROMを入れます。その後Mac OSを再起動させ、直後に[C]キーを押し続けます。

OldWorld ROM機(USBをオンボードで持たない機種、ページG3まで)ではそのままインストーラが起動します。NewWorld ROM機(iMac以降の機種)ではyabootの画面が現れますので、[TAB]キーを押してオプションを確認し、該当するキーワードを入力して[return]キーを押してください。



キーボードタイプの選択

ここではキーボードタイプを選択します。[TAB]キーを押すことでグルできます。ADB接続のキーボードを使っている場合は「ADB」を、USB接続の場合は「USB」をそれぞれ選んで[return]キーを押してください。

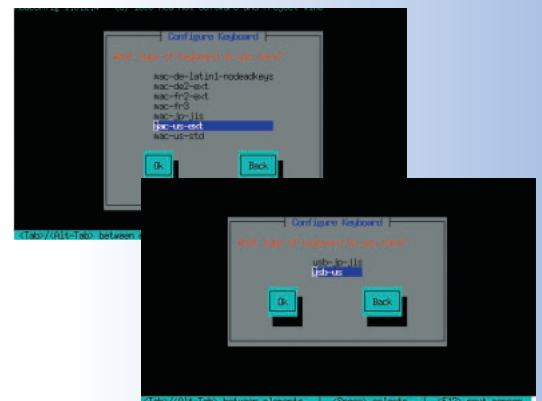


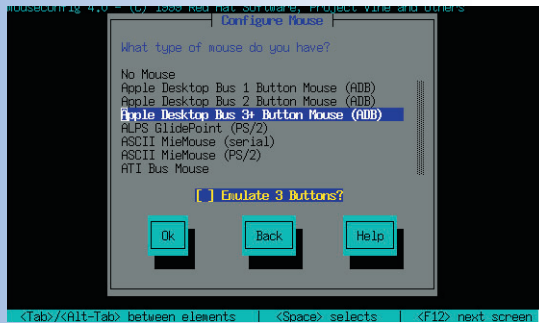
キーマップの選択

お使いのキーボードに相当するキーマップを、カーソルキーを使って一覧の中から選択してください。

- mac-jp-jis 日本語キーボード (ADB)
- mac-us-ext 英語キーボード (ADB)
- usb-jp-jis 日本語キーボード (USB)
- usb-us 英語キーボード (USB)

その後、[TAB]キーを使って「OK」ボタンに移動し、[return]キーを押してください。



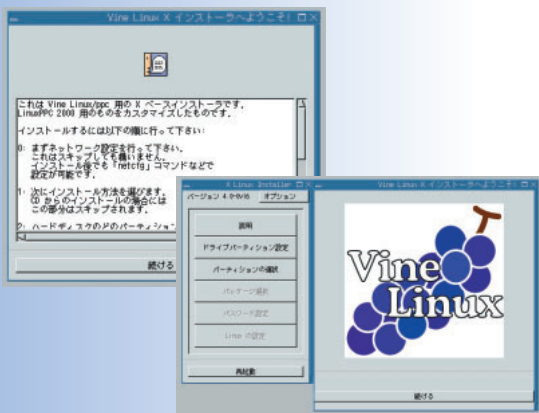


マウスの選択

お使いのマウスを、カーソルキーを使って一覧の中から選択してください。

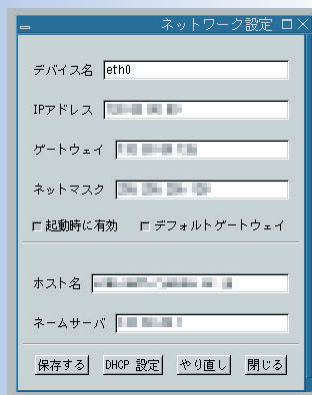
- Apple Desktop 1 Button Mouse (ADB)
- Apple Desktop 2 Button Mouse (ADB)
- Apple Desktop 3+ Button Mouse (ADB)
- Alps GlidePoint (PS/2)
- ASCII MicMouse (serial)
- ASCII MicMouse (PS/2)
- ATI Bus Mouse
- Universal Serial Bus 1 Button Mouse (USB)
- Universal Serial Bus 2 Button Mouse (USB)
- Universal Serial Bus 3+ Button Mouse (USB)

その後、[TAB] キーを使って「OK」ボタンに移動し、[return] キーを押してください。



Xインストーラの起動

しばらく待つと、Xが起動し、画面のようなインストーラが表示されます。



ネットワークの設定

インストールするマシンに固定IPアドレスを割り振る場合は、全項目を設定して「保存する」を押してください。ネットワーク接続されているがDHCPサーバからIPアドレスを取得する場合は「DHCP設定」を押し、その後ホスト名を指定して「保存する」を押してください。

ダイヤルアップ接続、あるいはスタンドアロンでの使用の場合は特に何も設定せず「閉じる」を押し、次に進みます。ダイヤルアップ接続やPCMCIAネットワークカード接続の場合は、インストール終了後に別途設定する必要があります。



パーティションの設定

Vine Linuxをインストールするためには、Swap領域用とLinux / (root) 用の最低2つのパーティションが必要です。

まず、「ドライブパーティション設定」ボタンを押し、パーティション設定ウィンドウを表示させます。

次に、あらかじめ確保しておいた各パーティションを選択し、それぞれについて「タイプ変更」ボタンを押し、Apple_UNIX_SVR2に変更し、パーティション名を指定します。

画面の場合ならば、/dev/sdaの6番目のパーティションをタイプ Apple_UNIX_SVR2、名称 swap に、7番目のパーティションをタイプ Apple_UNIX_SVR2、名称 / に変更しています。

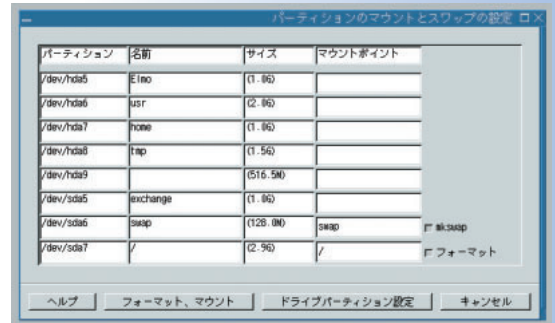
変更が済んだら「終了」ボタンを押し、次に進みます。ただし、パーティション数やパーティションサイズを変更した場合は、必ずここで再起動し、ステップ1からやり直してください。再起動しないとパーティションマップの変更が認識されません。

マウントポイントの指定

次に「パーティションの選択」ボタンを押し、マウントポイント設定のウィンドウを表示させます。画面の場合ならば、/dev/sda6をswapとして、/dev/sda7を/としてマウントするように指定しています。設定したら「フォーマット、マウント」ボタンを押します。

新規インストールの場合は、「フォーマット」や「mkswap」のチェックを入れておいてください。自動的にフォーマットされます。アップグレードインストールの場合は、swapパーティション以外について、これらのチェックを入れなくてください。チェックを入れると、そのパーティション内のデータは全部消去されます。

なお、インストール後の起動設定を行う際に、/のルートデバイス名（ここでは/dev/sda7）を指定する必要がありますので、控えておいてください。



インストールするパッケージの選択

パッケージ選択ウィンドウが現れますので、ここでインストールしたいパッケージグループを選択します。

「基本パッケージ」と「PPC用パッケージ」については必ず選択しておくようにします。ほかに（default）と書いてあるパッケージグループが最小推奨パッケージ集です。

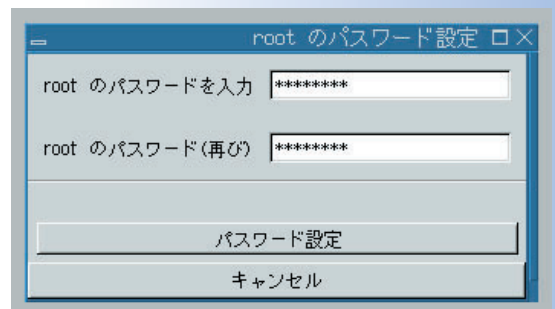
各パッケージグループの左の「+」を押すと、グループ内の個別のパッケージについて選択/非選択が可能ですが、よくわからない場合はグループだけ選択しておきます。

パッケージの選択が済んだら「インストール」ボタンを押してください。選択したパッケージの個数、CD-ROMの速度やCPUの速度などに依存しますが、10～30分程度でインストールが終了します。



rootのパスワード設定

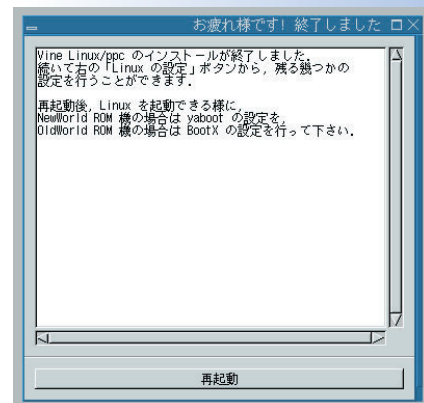
インストールが終わったら、rootパスワード設定ウィンドウが開きます。注意してパスワードを入力し、「パスワード設定」ボタンを押してください。



インストール終了

パスワード設定が正しく行われるとインストールは終了です。

このあと、Linuxを起動するための設定を行う必要があります。いったん再起動してMac OSを起動し、CD-ROMに納めているHTMLマニュアルの「Linuxが起動できるように設定する」セクションに従い、BootXあるいはyabootの設定を行ってください。



2台から始める

Linuxクラスタ

第2回 クラスタサーバを立てる（3回連載）

複数のコンピュータを使用してクラスタシステムを構成することで、クライアントからの処理を分散して計算するロードバランシングや、サーバマシンの故障が起こってもサービスを中断しないフェイルオーバーを実現できる。

今回は、TurboLinux Cluster Server 6.0と、Red Hat High Availability Serverのインストールと設定を行う。



TurboLinux Cluster Server 6.0の導入と設定

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

今回は、TurboLinux Cluster Server 6.0を使用して、**図1**のような2台構成のスケラビリティクラスタを構築してみることにする。しかも最新のバージョン6.0でチャレンジしてみることにした。ただし、執筆時点ではバージョンだったため、製品版とは若干の違いがあることを留意していただきたい。

構築上の大まかな作業の流れは、

- ・ベースOSのインストール
- ・クラスタソフトのインストール
- ・クラスタの設定
- ・クラスタの動作確認

の4つである。

本来は、このあとに負荷テストや運用テストを実施して、クラスタのチューニング作業をしなければならないが、超入門編の範囲を超えているので今回

は割愛する。

ベースOSのインストール

TurboLinux Cluster Server 6.0 (以下TLCS) は、ベースOSとして以下の2種類に対応する。

TurboLinux Server 日本語版 6.1
Red Hat Linux 6.2 (英語版)

ベースOSにTLCSをインストールすることで、フェイルオーバーとダイナミックロードバランシングを実現する。カーネルをクラスタ用の最新版2.2.16-0.4に置き換えるので、既存のマシを利用して考えている人は、トラブルに備えてデータや環境を、別の外部ディスク領域やDATなどのバックアップ媒体などに保存してから作業してほしい。

TurboLinux Cluster Server 6.0のインストール

それでは、TLCSのインストール方法を解説しよう。まず、ベースOSにrootでログインする。そして、TLCSのインストールCDをマウントして、インストーラを起動する。

```
# mount /mnt/cdrom
# cd /mnt/cdrom
# ./TLCS-install
```

正常に起動できると、コンソール画面にWelcomeメッセージが表示される。あとはTurboLinuxのインストーラと同様に、対話式に進めていけばよい。ここでは、重要なポイントだけ紹介する。

クラスタ用のカーネルに入れ替えるため「Linux kernel v2.2.16-04」を選ぶと、インストールするカーネルパッケージの選択画面が表示される。すべて選択されているので、*マークを消して不要なパッケージを外しておく(画面1)。なお、カーネルの再構築に備えて「Kernel sources」は選択しておこう。

インストール中の経過メッセージが



画面1 TurboLinux Clusterのインストーラ Installing new kernelでインストールするカーネルを選択する。

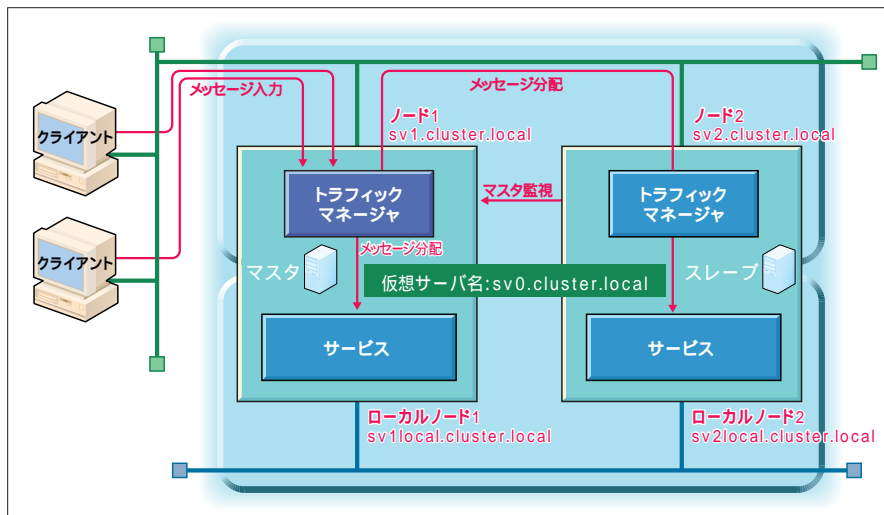
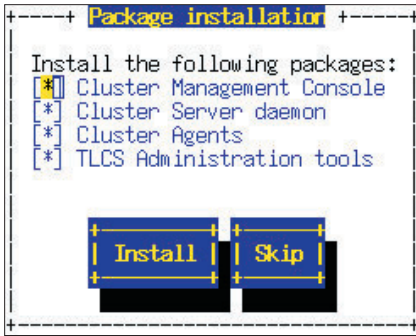


図1 2台で構成されたスケラビリティクラスタ クライアントからのWebアクセスは仮想サーバ名で行い、マスターとスレーブで処理を分担する。



画面2 Package installation
インストールするパッケージを選択する。

表示され、カーネルのインストールが完了すると、クラスタパッケージの選択画面が表示される（画面2）。ここでは、すべてを選択しておく。

「SSHの設定に必要な情報をすべて入力してください」という意味の画面が表示され、[OK] を押すと、暗号キーを生成するための基本データ入力画面になる（画面3）。「Country code」に日本を意味する“81”を入力し、その他のフィールドは英語ですべての情報を入力する。これらの情報はどこかに送信されるわけではないが、正しい情報を入力する必要がある。

LILOブートローダにTLCSの起動が



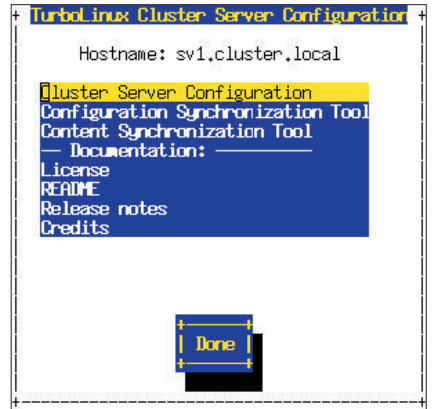
画面3 Secure Connection Certificate and Private Key
ユーザーの情報を登録する。

追加され、新カーネルがインストールされたことを示すメッセージが表示されたらTLCSのインストールは完了だ。[Reboot] を押してマシンをリポートしよう。起動時のLILOプロンプトが表示されたときに、Tabキーを押せば旧カーネルを選択して起動できる。

なお、当然のことだがクラスタを構成するマシンすべてに、同じ手順でTLCSをインストールする。

インストール後の確認

再起動したらログインして、Linuxのバージョンとクラスタプロセスが立ち上がっていることを確認する。



画面4 TurboLinux Cluster Server Configuration
TLCSの設定で表示される最初のメニュー。

```
# uname -s -r
Linux 2.2.16-0.4

# ps x |grep cluster
255 ?    S  0:00  clusterserverd
```

とクラスタデーモンの存在が表示されていれば、インストールはOKだ。

TLCSを利用するにはライセンスが必要なので、ターボリナックスジャパンより提供されるライセンスコードファイルを、/etc/clusterserver/licenses/ディレクトリにコピーしておく。

また、DNSがダウンしてもクラスタ

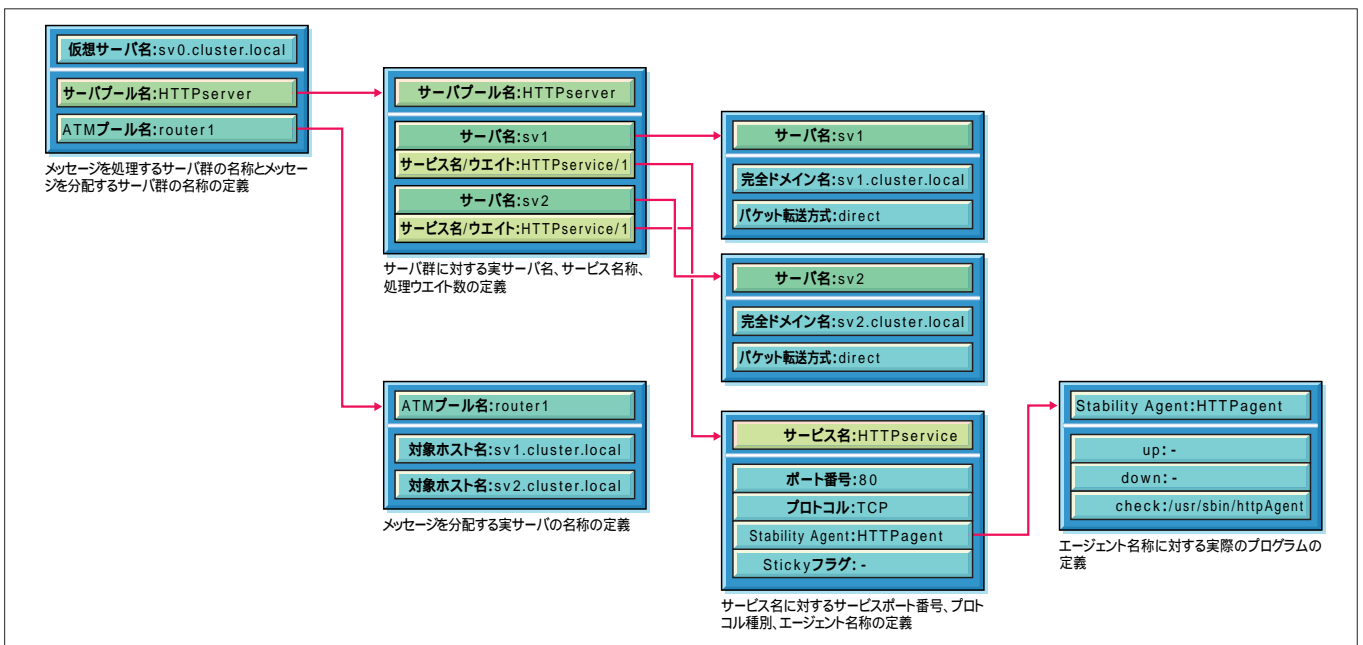


図2 クラスタ定義に設定する各パラメータの内容



サーバが影響を受けないように、クラスタ化するホストのIPアドレス情報を設定しておく。/etc/host.confを、

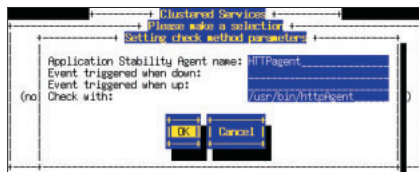
```
order hosts,bind
multi on
```

に修正しておく。こうすることで、ホスト名の名前解決を(1) /etc/hostsファイル(2) bind (DNS) の順で行うようになる。/etc/hostsにクラスタ対象ノードのアドレス、ノードのホスト名称、ホスト略称名などをすべて追記しておくこと。

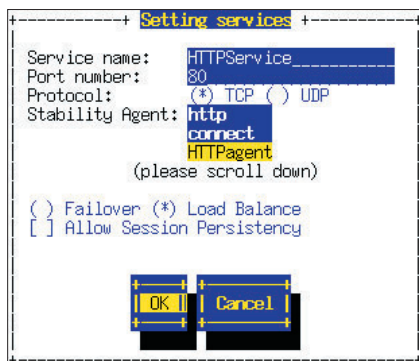
■ クラスタ定義の設定

さて、いよいよTLCSの設定だ。turboclusteradminコマンドを起動するとメニューが表示される(画面4)。

まず、一番上の [Cluster Server Configuration] を選ぶ。初回だけは、クラスタ定義を記録するclusterserver.



画面5 Setting check method parameters
アプリケーションのチェックを行うプログラムを指定する。



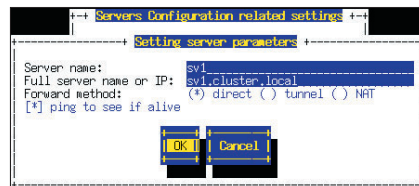
画面6 Setting services
クラスタで提供するサービスの設定。

confファイルがないため、Warning画面が表示されるが、気にせずに進もう。これから設定する各パラメータの内容を図2にまとめておく。

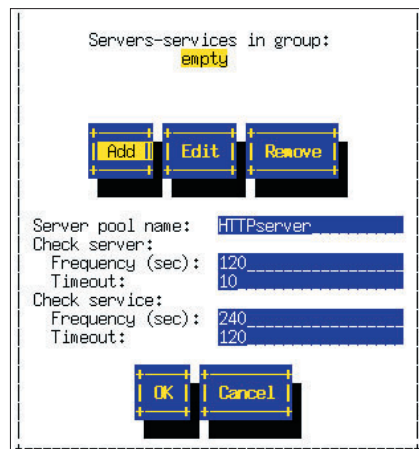
ASA (Application Stability Agent)

「Application Stability Agent」では、他のノードのサービスをチェックするエージェントを定義する。エージェント名には「HTTPAgent」、チェック用エージェントソフトには、標準で提供される「/usr/bin/httpAgent」を設定する(画面5)。

今回は話が複雑になるので詳細には説明しないが、サービスポートが異常になったり、正常に戻ったりしたタイミングで独自のエージェントソフトを起動したい場合には、「Event Triggered when down:」と「Event Triggered when up:」のところに、独自プログラム名をフルパスで入力することで可能だ。



画面7 Setting server parameters
各ノード(サーバ)の設定。



画面8 サーバプールの設定
サーバグループ名とチェック間隔を指定する。

サービス設定

「Services Settings」では、クラスタで実行するサービスとその各種パラメータをセットする(画面6)。

Service名:HTTPservice
ポート番号:80
Stability Agent:HTTPAgent

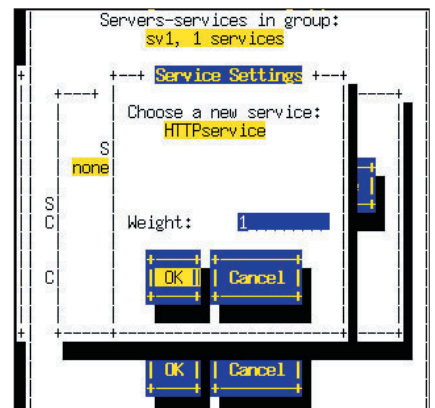
を入力し、「Load Balance」にチェックを入れておく。

サーバコンフィグレーション

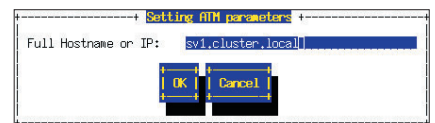
「Servers Configuration」では、Setting server parameters(画面7)で、サーバ名「sv1」とドメイン名「sv1.cluster.local」を入力する。メッセージの転送方式「Forward Method:」には、「(*) direct」を選択し、かつ「[*] ping to see if alive」にも、*のチェックが入っているのを確認する。同様にsv2についても行う。

サーバプールの設定

「Server Groups Configuration」



画面9 Service Settings
ノードごとに実行するサービスと、負荷割合を設定する。

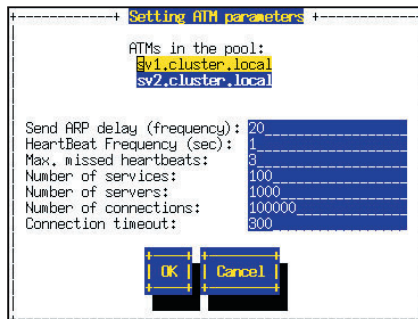


画面10 Setting ATM parameters
トラフィックマネージャ(ATM)のホストを設定する。

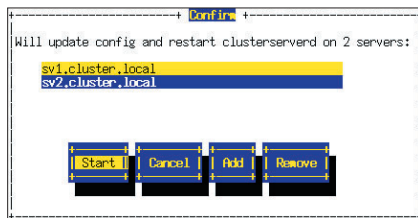
では、サーバプール（グループ）の設定を行う。「Server pool name:」には初期値として“ServerGroup1”が入っている。このままでも機能的には問題ないが、プール名称から内容が類推できるように変更しよう。今回は“HTTPserver”とする（画面8）。

次に、このHTTPserverを構成する実際のサーバを追加する。「Choose a server:」で[sv1]を選ぶと、このサーバで提供（実行）しているサービス名が表示される。前に設定した[HTTPservice]が出てくるのでこれを選択する（画面9）。

Weightパラメータは、そのノードのメッセージを処理する割合（比率）を示す。各ノードが均等に処理する場合は、デフォルト値1のままでもよい。ただ



画面11 Setting ATM parameters
ATMが監視するパラメータを設定する。



画面13 Configuration Synchronization Tool
ノード間のクラスタ定義ファイルを一致化させる。

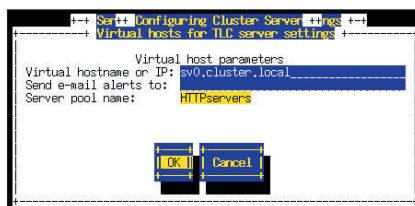
し、サーバ間で著しく能力差がある場合、たとえば能力が高いサーバに10を、能力が低いサーバには3を定義すると、10：3の割合で分散処理される。

今回は1つのサービスしか定義しなかったので間違えることはないと思う。同様にサーバsv2の定義を行う。すべてのサーバの追加が完了したら、サーバプールの設定は完了だ。

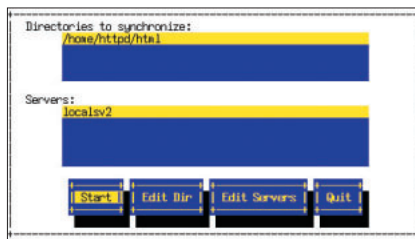
トラフィックマネージャの設定

「Advanced Traffic Manager Systems」(ATM)では、ATMになるサーバを設定する。今回は“sv1.cluster.local”と入力する（画面10）。同様にサーバsv2の情報も入力する。

[Advanced Traffic Manager Settings]では、ATMを監視するパラメータを



画面12 Virtual hosts for TLC server settings
クライアントからアクセスする仮想サーバの設定。



画面14 Content Synchronization Tool
ノード間のコンテンツを一致化させる。

入力する（画面11）。一般的にはデフォルト値で問題ないが、参考までに各パラメータの説明を表1に示す。今回はデフォルト値のまま[OK]を押す。

仮想サーバの定義

「Virtual Servers」は、初期状態では何も定義されていないので、「Virtual hostname or IP:」に、クライアントからアクセスする仮想サーバ名を入力する。今回は“sv0.linux.local”と入力する（画面12）。

ここで、「Send e-mail alerts to:」には、クラスタノードが異常のときに通知する管理者のメールアドレスをセットする。なにもセットしない場合は、このノードのrootユーザーに送信される。「Server pool name:」には、サーバプールのところで作成した名称（HTTPserver）を設定する。

クラスタ定義ファイルの一致化

ここまでで、クラスタ定義は終了である。

最初のメニューから「Configuration Synchronization Tool」を選び、クラスタ定義ファイル（clusterserver.conf）を、クラスタを構成する各ノードに転送する（画面13）。

[Start]キーを押すと、リスト上のノード間で自動的に一致化される。

コンテンツの一致化

最後に、HTMLファイルなどのコンテンツを各ノードに転送する。「Content Synchronization Tool」を選ぶと、「クラスタとして登録されたホスト名」が表示されるが、この画面を表示しているノードがsv1の場合、sv1は自分自身なのでコピーは不要である

項目	説明
Send ARP delay	ATMが送信するARPパケットのインターバル（秒）
Heart Beat Frequency	ATMマスタの生存信号をATMスレーブに送信する時間間隔（秒）
Max. missed heartbeats	ATMスレーブ（待機側）がマスタを異常だと判断する未受信のHeart Beat数
Number of services	HTTP、Telnetなどサービスの種類（Service Settingsでの登録の上限值）
Number of servers	サーバ数の上限値
Number of connections	同時接続数
Connection Timeout	クライアント無応答のタイマー値

表1 ATM Settingの設定パラメータ



ため削除しておく。

デフォルトの状態では、クライアントへのサービスポート（ネットワーク）を使用してコンテンツをコピーすることになっている。ノード間でコンテンツのコピーを行うために大量のデータが流れると、本来のサービスに支障が起る可能性がある。LANカードを2枚準備したほうがよいのはこのためである。

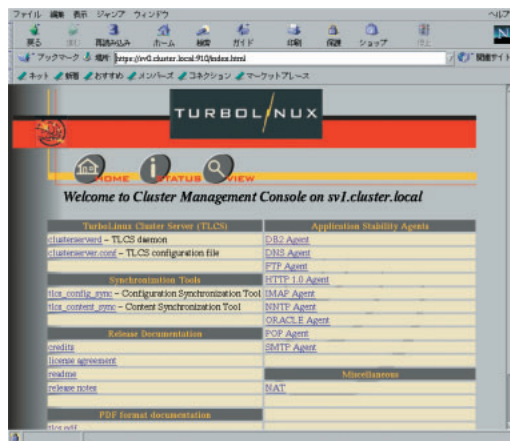
[Edit] で、ノード名を “ localsv2 ” に変更し、ローカル側のポート経由でコンテンツをコピーするようにする。コンテンツ同期化対象のディレクトリが、デフォルトのディレクトリと異なる場合も同様に修正する（画面14）。

同期化対象のディレクトリと相手ノードを変更したあと、[Start] を押すとコンテンツの一致化が始まる。動作しない場合は、両方のノードの /root/.rhosts に、相手ホスト名を記入して実施してみたい。

■ クラスタの動作確認

クラスタ管理コンソール（CMC）からクラスタの動作状態を確認することができる。Webブラウザを立ち上げ、URLに、

<https://sv0:910/>



を入力する。ユーザーIDとパスワードを聞いてくるので、入力するとCMCが表示される（画面15）。このStatusアイコンをクリックすると、クラスタのプロセス状態が表示される（画面16）。そのほか処理のロードバランスをビジュアルに検証できるツールがある。

<http://www.turbolinux.co.jp/products/turbocluster/>

にアクセスするとJavaベースで動く検証ツールが提供されている。

なお、コマンドによってクラスタノードを停止するには、

```
# /etc/rc.d/init.d/clusterserverd stop
```

と実行する。クラスタノードを起動するには、“ stop ” のところを “ start ” に置き換える。

また、正常に動作しないときは、

・クラスタ関連ログ

```
/var/log/clusterserverd.log  
/var/log/cmc.log
```

・カーネルログ

```
/var/log/messages
```

の3つのログを見れば問題点の切り分けができるはずだ。

もしこれでも不明な場合は、

```
# echo 1 > /proc/net/cluster/debug
```

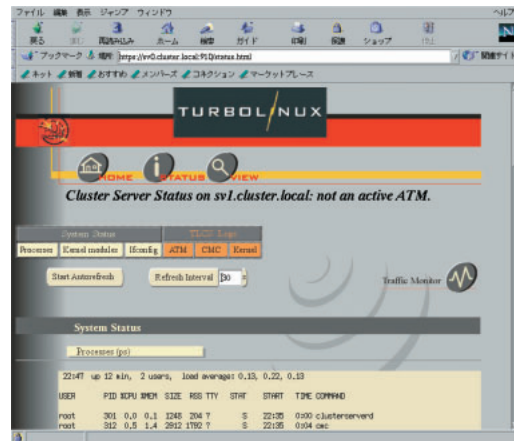
として、デバッグモードでクラスタを再起動してみよう。より多くのクラスタ関連ログを入手することができる。

さて、これでクラスタが動作し、2ノードで動作するWebサーバを構築できた。クライアントのWebブラウザから、仮想サーバ名「sv0.cluster.local」へアクセスすれば、自動的に2台のマシンに分散してアクセスされる。もし片方のマシンが停止しても、もう1台のマシンによってサービスは提供される。

クラスタシステムでは、ノード間の制御情報のやり取りが頻繁になるように定義すると、相手のクラスタを異常だと誤認識したり、本来のクライアントに対する処理能力が低下したりするので注意してほしい。

今回は、TurboLinux Cluster Serverを使った実際の設定と、動作検証までを説明をした。次回は、クラスタのユーザー事例やBeowulfクラスタのMPIの簡単な使い方、EnFuzionの内容を紹介する。

画面 15 Cluster Management Console Webブラウザでリモート管理できるクラスタ管理コンソール。



画面 16 CMCのステータス表示
各ノードの状態やログを表示することができる。

Red Hat Linux High Availability Server 1.0の概要と運用

文：濱田充男 レッドハット株式会社
Text : Mitsuo Hamada (mhamada@redhat.com)

Linuxは、安全なインターネットサービスを安価で提供できることで急速にその数を増やしてきた。また、最近ではLinux上で動作する、ロードストミノやSAP R/3のアプリケーションサーバといったビジネスアプリケーションが増えてきており、ASPのサーバとしても注目されている。

そういった世界の動きの延長線上で、BtoC、BtoBのシステムや、IPベースのビジネスアプリケーションサーバなどへの応用を考えると、常に動作し続けること、サーバの負荷に応じて負荷分散ができることといった能力がLinuxに求められるようになってきた。そういった要求に応えるために誕生したのが、Linuxのクラスタリングシステムである。

ここでは、7月に米国レッドハットからリリースされた、Red Hat High Availability Server(以下、HA Server)の概要とその運用までを解説する。



画面1 Linux Virtual Server Project
http://www.linuxvirtualserver.org/

HA Serverとは?

HA Serverは、Linux Virtual Server Project (画面1) で開発されているコードを利用し、Red Hatのクラスタリング技術Piranhaを組み込んだ高信頼性クラスタリングサーバプロダクトである。

実は、このPiranha自体はRed Hat Linux 6.2J (以下、6.2J) にも、すでに同梱されているが、Linuxのクラスタリング機能に対する期待がますます高まっており、そういったユーザーのニーズに応えるため、さまざまな改良とともに、まったく別の製品として開発された。

HA Serverの特徴を簡単に示す。

- ・インストールオプションがCluster ServerあるいはCustomの2種類のみ
- ・セキュリティの強化
- ・HA Serverの機能概要とセットアップ方法が解説されたマニュアルの同梱
- ・6.2Jで判明した各種バグの修正
- ・年間サポート付属
- ・もちろんオープンソース

Piranhaの概要

Piranhaは大きく分けて2つの機能に分けることができる。1つはFailover Service (FOS)、もう1つはLinux Virtual Server (LVS) である。

FOSは、非常に単純な2ノードにおけるサーバのフェイルオーバー機能を

提供する。したがって比較的負荷の低いWebサーバ、イントラネット、DNS、メールサーバなどに向いている。

対して、LVSは、フェイルオーバー可能なルータを介し、ロードバランシングも行えるため、大規模なWebベースシステムの構築に向いている。

FOSの概要

FOSの特徴を以下に示す。

- ・2ノードクラスタリングのみサポート
- ・ダイレクトソケット接続をサポートするIPサービスのモニタリング(ユーザー作成のプログラムも含む)が可能(HTTP、FTP、telnet、lpd、SMTP/Sendmail、ssh、LDAPなど)
- ・モニタリングしているサービスを自動で開始および停止することが可能
- ・クラスタノード用のIPアドレス以外に、もう1つIPアドレスを利用し、このIPアドレスでサービスの提供を行う

次に、FOSの現時点での制限事項を示す。

- ・フェイルオーバーさせるサービスはグループとして扱う。個々のサービスをフェイルオーバーさせることはできない
- ・NFS、NTPはフェイルオーバーできない
- ・データ共有のための仕組みが存在しないため、NFSなどを利用してデータ共有をする
- ・プライマリ、バックアップ共にLinuxサーバでなければならない

FOSのアーキテクチャ

図1のように、HTTPサービスは



Virtual IP (図では192.168.0.100) を通じて提供される。バックアップ側のサーバは、プライマリサーバのHTTPサービスを本来のIPアドレス (192.168.0.1) を通じてモニタリングする。

プライマリサーバになんらかの障害が発生し、HTTPサービスが停止するとフェイルオーバーが起き、図2のようになる。フェイルオーバーはVirtual IPの移動とバックアップサーバ側でのサービスの起動によって行われる。

FOSのコンポーネント

フェイルオーバーを行うコンポーネントは図3のようになっている。

pulseはPiranhaのメインのデーモンプロセスで、ハートビート(一定周期監視)によりクラスタノード同士で相

手のサーバが動作しているかを確認、およびfosデーモンプロセスの起動、停止を管理する。

fosプログラムは、pulseによって起動され、2つのモードを持っている。アクティブノードでは、--activeオプションで起動され、IPサービスの起動、停止を行う。

バックアップノードでは、--monitorオプションで起動され、nannyサービスモニタリングデーモンを起動する。バックアップノードがサービス停止などを検知すると、fosはフェイルオーバー動作を初期化し、pulseにより--activeモードで起動されるように仕向ける。

nannyは、サービスモニタリングデーモンで、定義されたIPサービスに対してそれぞれ1つずつ起動される。

LVSの概要

LVSは、複数の2台以上のサーバによって構成され、高可用性サービスを提供する。LVSの典型的な構成例を図4に示す。

つまり、FOSで動作するのと同時に、サービス提供サーバの負荷分散をつかさどるルータと、実際にサービスを提供する複数のサーバという構成になる。

LVSの特徴は次のとおりである。

- 複数のルーティング方法のサポート
NAT、IPトンネリング、ダイレクトルーティング
- 複数の負荷分散アルゴリズムのサポート
Round robin、Least conne

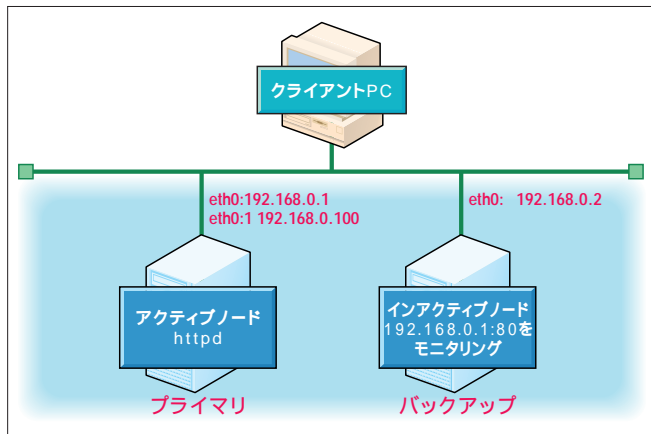


図1 フェイルオーバー前の動作状態

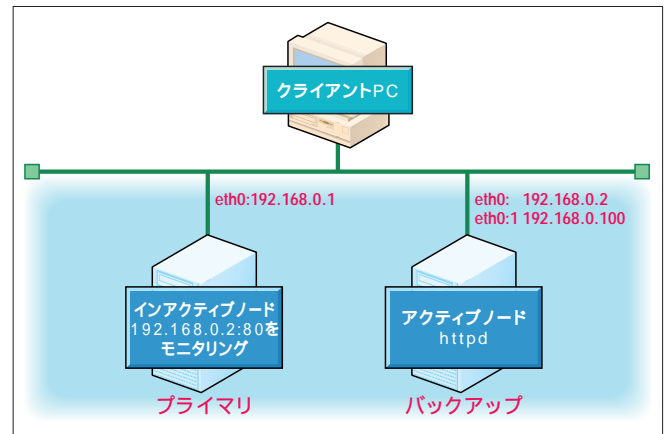


図2 フェイルオーバー後の動作状態

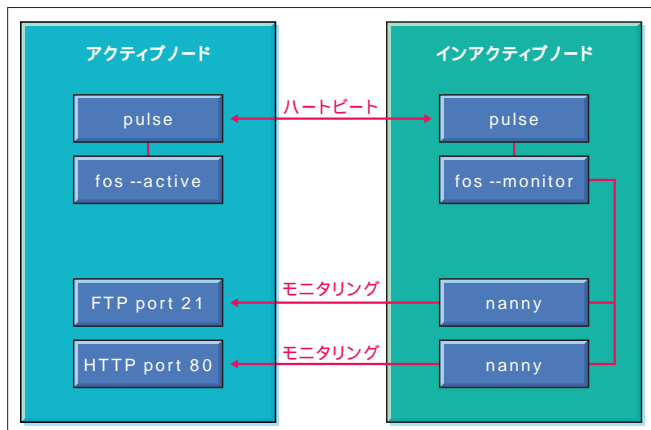


図3 FOSを構成する各コンポーネント

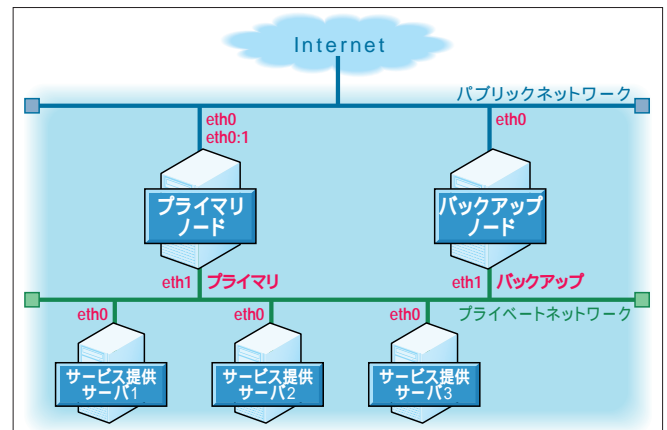


図4 LVSの構成例

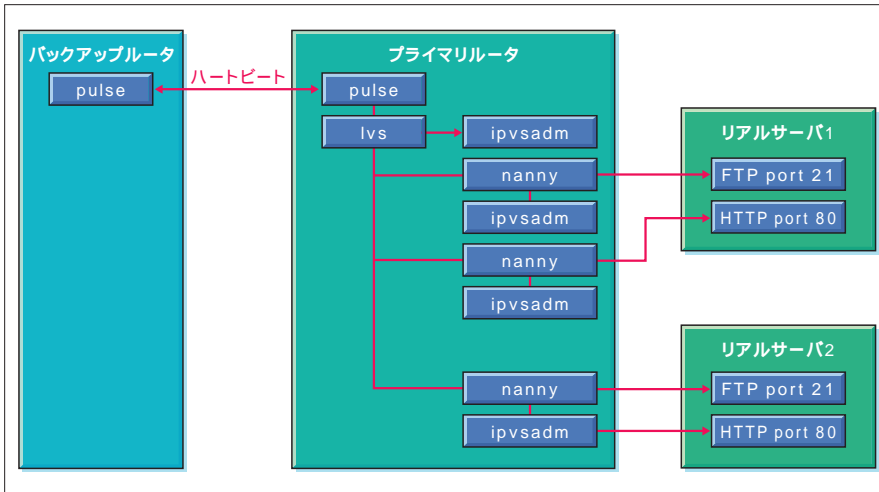
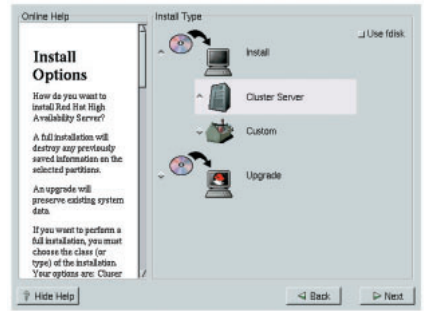


図5 LVSを構成する各コンポーネント



画面2 インストールオプション選択

lvsデーモンは、ipvsadmおよびnannyデーモンの起動・停止を行う。ipvsadmデーモンは、カーネル内のルーティングテーブルを更新するプロセスだ。

nannyデーモンは、サービス提供サーバがアクティブか否か、また負荷の状態を監視する。

HA Serverのインストールから運用まで

ここまででHA Serverの概要は理解できたと思うので、実際のインストールと設定を紹介しよう。

HA Serverのインストール

HA Serverも通常のRed Hat Linuxと同じインストーラを使用しているため、一度でもRed Hat Linuxをインス

ction、Weighted round robin、Weighted least connectionの4つをサポート)

- ・サービスを提供するサーバはLinux以外のOSもサポート

前述の4種類の負荷分散アルゴリズムについて簡単に説明しよう。

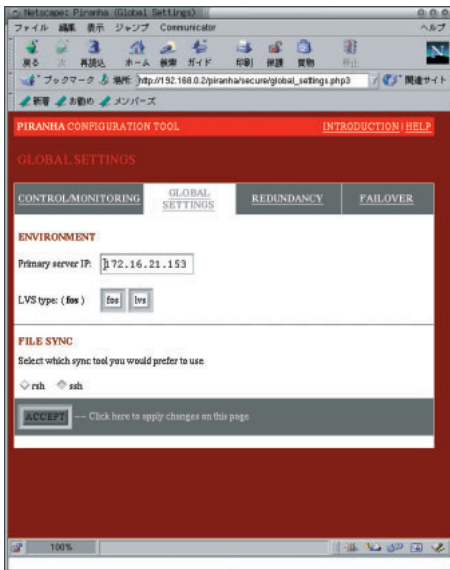
Round robinでは、文字どおりすべてのサーバを同等に扱い、順番に利用する。

Least connectionでは、その時点でもっとも接続数の少ないサーバを利用する。

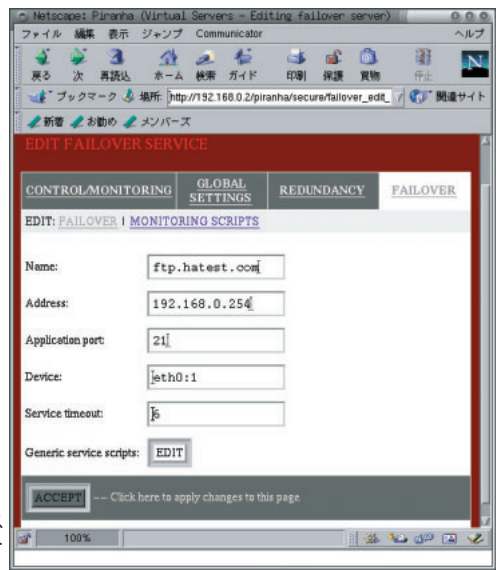
Weightedというのは、それぞれのサーバの能力に応じて重みづけをすることができるというものだ。たとえば、サーバが3台あり、1台だけ他のサーバの2倍のメモリを搭載しているとする。この場合に、2 : 1 : 1といった重みづけを設定できるということである。

LVSのアーキテクチャ

pulseは、FOSの場合と同じくlvsデーモンの起動・停止を行い、ハートビートによりプライマリルータとバックアップルータの動作状況を監視する(図5)。



画面3 FOSのプライマリサーバの設定
LVS typeはfos (フェイルオーバー) を指定し、File Syncにはクラスタノード間のファイル同期を指定する。sshを選んで暗号化するほうがセキュリティ上安全だ。



画面4 フェイルオーバーの設定
クライアントに対して行うサービスを設定する。Addressには仮想サーバのIPアドレスを入力する。



インストールしたことがある方なら簡単にインストールすることができるだろう。

インストールでの大きな変更点はインストールオプションの選択である。通常のRed Hat LinuxではGNOMEワークステーション、KDEワークステーション、サーバ、カスタムが存在するが、Cluster Serverとカスタムの2種類だけになっている(画面2)。通常は、Cluster Serverでインストールすれば問題はない。

FOSの設定

FOSを実際に設定してみよう。ここではもっとも便利なブラウザからの設定を行う。設定された値は/etc/lvs.cfファイルに保存される。

まずプライマリサーバを登録する(画面3)。LVS typeのfosボタンを押したあと、Primary server IPにプライマリノードのIPを入力し、[ACCEPT]ボタンを押す。File Syncの部分はクラスタノード間でファイルシンクに利用されるコマンドとして、好みのほうを選択すればよい。セキュリティのことを考えるとsshのほうが望ましいだろう。

次にフェイルオーバーを設定する

(画面4)。ここでAddressに入れる値は、サービスを提供するために利用されるVirtual IPだ。Deviceのところは“eth0:1”とあるのでVirtual IPであることがわかると思う。[ACCEPT]ボタンで設定完了だ。ここでGeneric service scriptsの[EDIT]ボタンを押すと、次の画面に変わる。

ここではサービスの起動・停止の方法や、そのサービスが返す文字列を設定する。この画面はhttpdを例にしている(画面5)。この情報文字列はフェイルオーバーのモニタリングに利用され、起動・停止の方法はバックアップサーバでサービスを起動する際に利用される。このように、サービスが返す文字列を入力することができるため、ユーザーが作成した特殊なサービスもクラスタリングで利用可能になっている。

そして、バックアップノードのIPアドレスを設定する(画面6)。Heartbeat Interval(ハートビートモニタリングの間隔)、Assume dead after(各ノードがダウンしたとみなすまでの時間)も秒単位で設定できるが、これらはデフォルトのままでも構わない(ただし、複数のFOSサーバ群がある場合には、

Heartbeat runs on portの値を変更する必要がある)。

最後に、同じ設定をバックアップノードでも行うか、/etc/lvs.confをバックアップノードにコピーすれば設定は完了する。プライマリ、バックアップ双方で、

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

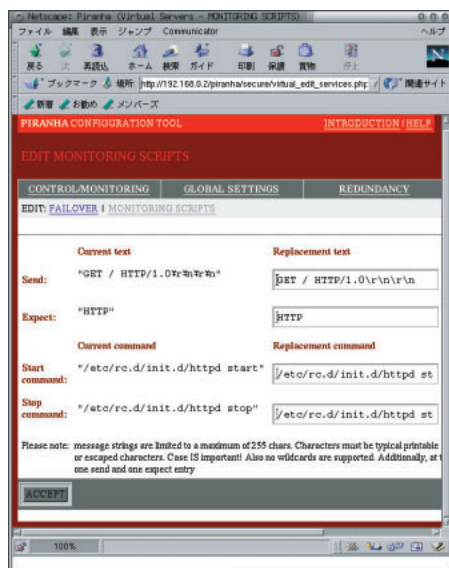
を実行すれば、FOSとして動作する。

クラスタリングサーバの構築というと難しそうなお雰囲気が漂うが、インストールもメニューで選ぶだけだし、実際の設定はブラウザから行うことができ意外に簡単であることがおわかりいただけたと思う。今回割愛させていただいたLVSの設定にも挑戦してみたい。

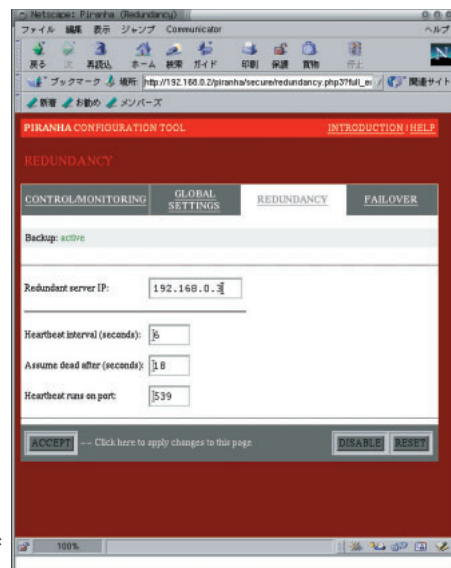
HA Serverの入手方法は、現在www.redhat.comから購入するか、あるいはftp.redhat.comからダウンロードの2つの方法がある。

参考文献：

<http://www.linuxvirtualserver.org/>
<http://www.linux-ha.org/>



画面5 ポートモニタリングの設定
サービスの起動、停止を行うプログラムの設定と、サービスをモニタするために送るメッセージと正常に動作しているときに返ってくるメッセージを登録する。



画面6 バックアップノードの設定
サービスを提供している仮想サーバ/VIPアドレスと、その監視時間間隔を設定する。

手作りサーバでMP3レコーディング ラジオ自動録音システム(後編)

Webからのアクセス機能を追加する

ラジオ番組の自動録音そのものは、前編で説明した手順ですぐに始められる。しかし、ハードディスクに蓄積された録音ファイルの中から、目的の講座、日付の番組を見つけ出すのは、なかなかやっかいである。そこで、シリーズ後編では、録音済みファイルへWebページ経由でアクセスできるように、Linuxに付属するWebサーバとCGIスクリプトを活用した、簡単なユーザーインターフェイスを追加してみる。

文：竹田善太郎
Text: Zentaro Takeda

前回構築した自動録音システムでは、録音済みのMP3形式ファイルに図1のような形式のファイル名をつけ、番組ごとのディレクトリに分けて保存するようにした。ファイル名は、日付と時刻をもとにつけられているので、lsなどでディレクトリを一覧すれば、自動的に日付順に並び、ファイルの数が少ないうちなら、目的の日時の録音ファイルを見つけ出すのはそれほど難しくはない。しかし、録音を続ける期間が長くなり、ファイルの数が100、200、……と増えてゆくと、ディレクトリ一覧も大きくなり、ファイルを見つけ出すのはかなり苦痛になってしまう。また、このシステムでファイル

につけられている名前は、日付、時刻の数値を羅列した形式になっているので、人間の目にはかなり判別しづらい。

ところが、人間の目にはわかりづらい数値の羅列も、コンピュータにとっては逆に扱いやすい。実は、ファイルにこのような名前を付けたのは、録音ファイルへのアクセスや整理なども、Linuxで自動化したいと考えていたからだ。番組、録音日時の情報がファイル名そのものに含まれているので、録音内容を管理するためのデータベースを別途作成したり、ファイルに埋め込まれているデータを読み取ったりする必要なしに、番組の検索やファイルの移動などの処理が簡単にて

ec 2000 08 01 15 45 _64K .mp3
番組コード 年号(西暦4桁) 月(2桁) 月(2桁) 時(24時間制、2桁) 分(2桁) ビットレート 拡張子

図1 自動録音されたファイルのファイル名
番組識別用コード、日付け、時刻、ビットレートなどの情報をファイル名に埋め込んである。人間にはわかりづらくなってしまうが、ファイル名から録音内容をすべて把握できるようになっているので、後々の管理が楽になるのだ。



きるのである。

今回は、「音声の処理」から離れて、大量の録音済みファイルをPerlスクリプトを使って自動的に整理し、さらにWebサーバ (Apache) のCGI機能 (こちらPerlスクリプト) を使って、クライアントマシンのWebブラウザから簡単にアクセスできるようにする。

例によって、スクリプトにPerlを使っていることに特別な意図はない。シェルスクリプト、Ruby、AWKなどのほかのスクリプト言語でも、同じようなシステムが構築できるはずである。ただ、ファイル名の処理などには、Perlの特徴である「ハッシュ」(連想配列) や「文字列置換」などの機能を積極的に使っていて、ほかのスクリプト言語に比べてより簡潔な記述ができていと思う。本記事の目的は「Perl入門」ではないのだが、Perlの「パワー」の一端を感じてもらえるかもしれない。

クライアント向けのユーザーインターフェイス提供にはCGI (Common Gateway Interface) 機能を使っている。「CGI」というと、すぐに「アクセスカウンタ」とか「掲示板システム」を思い浮かべる読者も多いかもしれないが、CGIやマイクロソフトのASP (Active Server Pages) といったオンデマンドページ生成システムの本来の目的は、サーバの上にあるさまざまな形式のデータを、クライアントからの要求に応じて最新の状態で、かつ、わかりやすく整理して表示する、というものである。そういう意味で、本記事で作成するユーザーインターフェイスは、見てくれはよくないのだが(余計な飾りはなく、使っている機能も最低限のものだけ)、CGI入門の良いサンプルになるのでは、と自負している。

本システムの構築にあたって、特別なハードウェアやソフトウェアは使っていない。すべて、Linuxのディストリビューションに付属するものばかりである。モデルとして筆者が構築したシステムでは、TurboLinux Workstation 6.0を使っているが、一般的なディストリビューションであれば、ほとんど同じ手順で同様のシステムが構築できるだろう(ただし、Webサーバのデフォルトのセキュリティ設定などは、ディストリビューションによってかなり異なるので、注意が必要だ)。また、前回構築した録音システムの基本部分に変更を加える必要は、いっさいない。

ユーザーの利便を考える

ラジオの番組をMP3形式で録音することの利点は、なんといっても、再生できる環境が広がることである。数年前は

ともかく、現在ならば、Windows、Macintosh、Linuxなどを問わず、ほとんどどんな構成のPCでも、追加投資の必要なしにすぐにMP3ファイルの再生ができる。また、ポータブルタイプのシリコンオーディオプレーヤでも、(形式変換の必要性や対応するビットレートなどの制限はあるものの)MP3ファイルの再生ができない機器はないだろう。

家庭内でLANを構築していて、家族がそれぞれPCを使っているような場合なら、録音したデータを家族で共有して「バイリンガル家族」を目指すのもよい。このような場合でも、MP3ファイルで保存しておけば、それぞれのPCやOSの種類を問わずに、ファイルを利用できるようになる。

蓄積されたファイルの共有には、SambaやNFS、netatalkなどの、Linuxに標準的に付属する各種のファイル共有サービスが使えるだろう。しかし、前述したようにファイルを共有しただけでは、ユーザー(あるいは読者自身)が目的の録音ファイルを探すのに、英数字の羅列であるファイル名と格闘しなければならぬ(画面1)。自分1人だけが使うのならともかく、家族にも使ってもらおうとする場合、長ったらしいファイル名の「解説」方法を理解してもらうのは、教わるほうも教えるほうも苦痛でしかないだろう。

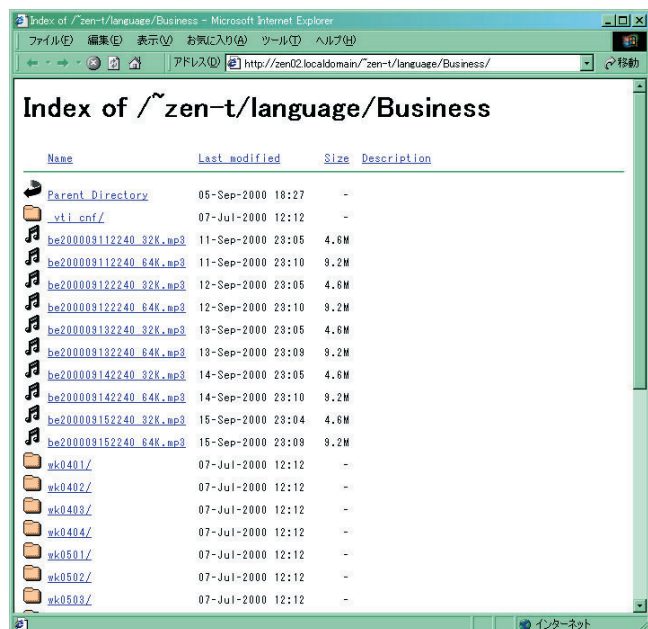
そこで、LinuxのWebサーバ(Apache)を利用して、クライアントマシンからはWebページを使ってアクセスできるようにしたい。Webブラウザなら、ほとんどのOSで似たような操作でアクセスできるし、ページの構成やレイアウトを工夫すれば、検索機能などがなくても、目的の情報にすばやく到達できるようになる。また、LAN環境がなく、Linuxマシンをスタンドアローンで使っている場合であっても、Webページを使ったインターフェイスは有効である。



画面1 ファイルを直接選択するのは面倒
録音ファイルを再生するときに、長いファイル名をコマンドラインに入力するのは苦痛である(シェルの「補完」機能を使えば多少は楽になるが)。GUIを使って一覧から選択する場合は、補完機能が使えないのでさらに苦痛が増す。

どのような機能を持たせるか

ラジオの語学番組は、1週間単位でカリキュラムが組まれていることが多い。このため、録音済みの番組は、1週間単位のページに分けて表示されるようにすれば便利である。また、もっとも新しい週（すなわち「今週」）の録音について



画面2 Webサーバの「ファイル一覧」ページ

Apacheを始め、多くのWebサーバには、サーバ上のファイル一覧をクライアントで閲覧できるようにする機能を持っている。セキュリティの設定によって、この機能が使えないようになっていることも多いが、インターネットで情報を検索しているときにこのような画面を目にすることも多いだろう。単にファイルの共有をしたいというだけなら、この機能を利用してよいのだが、今回の目的には使いづらい。

は、できるだけすぐにアクセスできるようになっていると便利である。つまり、できるだけ「浅い」ページの階層に置いておいたほうがよい（図2）。

各ディレクトリの名前には、月と週の番号が含まれるようにしておく。これで、録音ファイル自身と同様に、ディレクトリ名だけを見ればその内容がわかるようになる。

一定の期間（たとえば1週間）が経過して、録音済みのファイルがある程度たまってきた時点で、そのファイルを週ごとのディレクトリに「分類」するわけだが、このような処理も手作業ではなく、スクリプトで自動化したい。ファイル名の中の日付から、そのファイルが「何月の第何週」に属するかを判定して、必要ならディレクトリを作成し、そこに移動する。

このようにして、日々自動的に録音され、一定期間ごとに「整理」されたファイルへのアクセスを、Webページから行えるようにしたいわけだが、これにはいくつかのアプローチが考えられる。

- 1) 特別なWebページは作らずに、Webサーバの「ディレクトリ一覧」表示機能を利用して、ファイルに直接アクセスする（画面2）。
- 2) 一定期間ごとに、録音済みファイル一覧のページを自動生成して、そこからアクセスできるようにする。
- 3) CGI機能を使って、ユーザーからの要求があるたびに、動的にページを生成する。

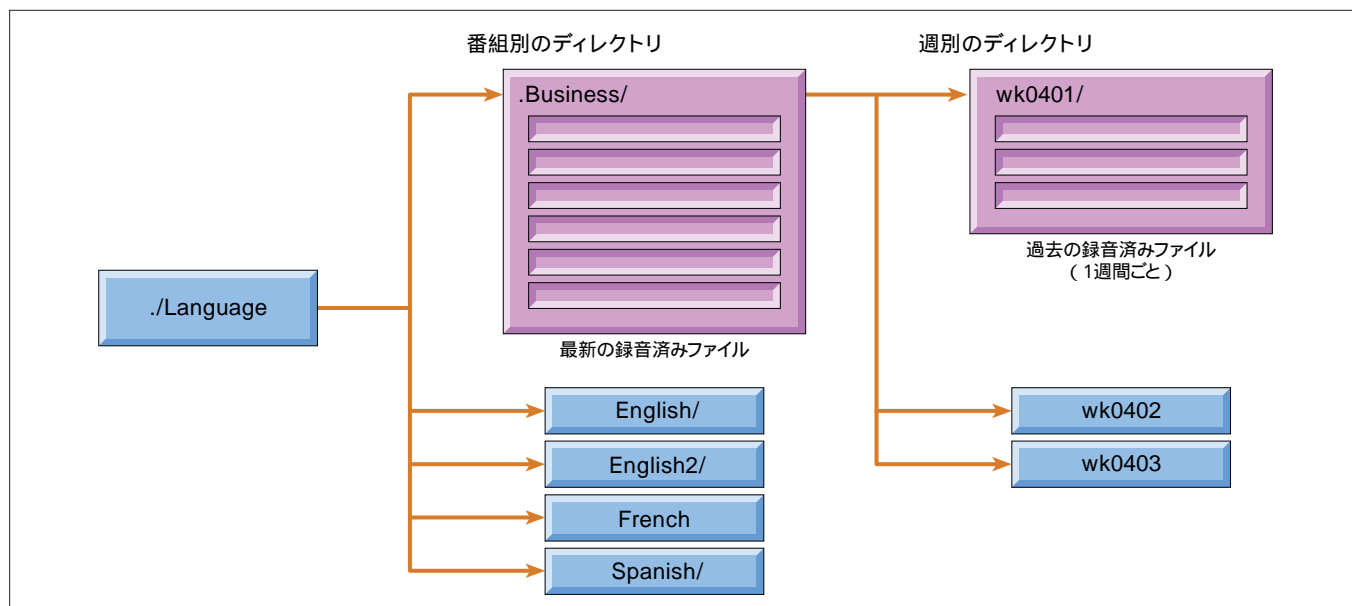


図2 ファイルを配置するディレクトリ階層

番組ごとのディレクトリの下に、さらに週ごとに録音ファイルをまとめるサブディレクトリを作成する。最新の録音（現在の週の録音）のみを番組の「ルート」ディレクトリに残し、古いファイルはすべて週ごとのディレクトリに置いておけば、ファイルの管理がしやすくなるし、CGIで目次ページを生成するのにも楽になる。



もっとも簡単なのは1)の方法なのだが、これでは数字の羅列でしかないファイル名から目的の番組を特定することになり、当初の目的には合致しない。2)の方法なら、よりわかりやすい方法でアクセスできるようになり、録音する番組の数が少ない場合などは十分なのだが、スクリプト作成の手間は3)と変わらない。また、ページを更新するまでの間、アクセスできない録音ファイルが存在することになり、どのようなタイミングでページを更新するかの判断も難しい。3)の方法については、「CGI」という難しそうなテクニックをマスターしなければならないうえ、Webサーバ自体の設定も面倒そうなので(セキュリティに関する心配ごとが多い)、最初は躊躇したのだが、実際には2)の方法と比べて、作成すべきスクリプトの内容はほとんど変わらず、Webサーバの設定についても、LAN内部での利用に限るのであればそれほど神経質になる必要はない。

正直に白状すると、筆者はCGIについては、基本的な知識はあったものの、実用的なサイトを構築した経験はなかった。しかし実際に挑戦してみたところ、完全なスクリプトの作成には半日もかからなかった。フレームやフォームを使ったり、「動く」ページを作ろうとしたりなど、複雑なことをしないのであれば、実用的なCGIスクリプトの作成はそれほど難しくはないのだ。できあがったページは、見た目はそっけないものの、動作はきわめて軽快で使いやすいものになった。もちろん、より洗練されたページをつくることもできる

だろうし、フレームやフォーム、テーブルを駆使した「カッコイイ」ページにするのも自由である。記事中のスクリプトを「叩き台」にして、読者自身の納得のいくシステムにしてほしい。

では、録音済みファイルの仕分けからWebサーバの設定、CGIスクリプトの作成と、順を追って説明しよう。

録音済みファイルの仕分け

前出の図2のようにディレクトリを分け、ファイルを配置するわけだが、ここでひとつ大きな問題がある。ある日付(たとえば9月1日金曜日)を取り扱うとき、それが「何月の第何週」に属するかを決定するのが、意外と難しいのである。どのような問題があるかについては、別掲のコラム「日付の扱いの難しさ」にまとめたが、今回は機械的に「その月の最初の月曜日が含まれている1週を第1週」と決めることにした。NHKラジオの語学番組の週分けと矛盾することもあるのだが、ここではあえて無視して、プログラムの簡潔さを重視することになっている。個々の録音ファイルへは「日付」でアクセスすることになるので、実用上は問題ないだろう。

仕分け用スクリプトの作成

Perlで記述した仕分け用のスクリプトがリスト1のweeklycleanupである(付録CD-ROMに収録)。全体で131

Column

日付の扱いの難しさ

録音した番組のファイルを管理するには、月、週、日といった「日付け」の情報を使うのがもっとも適している。しかし、この日付けをコンピュータで扱うのは、なかなか厄介である。

なにより面倒なのは、「1週間」をどのように定義するかが決まっていないうことである。カレンダーによって、「月曜日」を1週間の始まりにするか、「日曜日」をそうするかまちまちなのである。日本では「日曜日」を週の始まりにすることが多いようだが、国によって(あるいは宗教や「会社の都合」によって)変わってくる。

「第1週」をどのように定義するかも難しい。「1日」の含まれる週を第1週とすると、

たとえば、2000年9月の「第1週」は、(日曜日を週の始まりとした場合)1日と2日の合計2日間だけということになる。また、そうした場合、2000年8月の「第5週」と9月の「第1週」は同じ週を指すことにもなる(読者のお手元にあるカレンダーを見ていただければわかるだろう)。

世間一般の会話の中で「何月の第何週」という場合、上述したような方法で区別することが多いようだが、本記事で取り上げている「語学番組」の放送スケジュールのような場合は、「1週間」を単位に番組が編成されているため、これでは都合が悪い。NHKラジオの語学番組の年間スケジュールを見ると、おおざっぱには「その月の最初の月曜日のある週」を「第1週」としているようだ。このような法則性があれば、コンピュータでの処理も簡単そうに思えるのだが、よく調べてみ

ると、やっかいなことに、カレンダーの配置によって、ひと月に月曜日が4回ある場合と5回ある場合がある。そして、5回目の月曜日がある場合は、その週は翌月の第1週に組み込んでしまう場合と、その月の「第5週」として、翌月の頭の数日をその週に組み込んでしまう場合の両方があるようなのだ。

もちろん、放送の年間スケジュールがわかっているのなら、そのデータをプログラムに組み込んでしまえばよいのだが、それではプログラムの「汎用性」がなくなってしまって、ほかへの応用が利きにくくなる。

そこで、今回は放送のスケジュール(テキストでの週番号の表記)とは矛盾する場合があるのを承知のうえで、プログラムのわかりやすさを優先して「第1月曜日のある週を第1週とする」という方法で週の番号を割り当てることにした。

行（掲載したのはその一部）と、かなり長いスクリプトのように見えるが、多くの部分はファイル名から日付情報を取り出すルーチンを「汎用化」するためのコードである。ここで汎用化したコードは、後述のCGIスクリプトで利用している。

「日付」や「曜日」の処理は、「西暦2000年問題」や「うるう年」の話を持ち出すまでもなく、コンピュータでもっともデリケートな領域なのだが、ここでは、Perlの組み込み関数localtime()とライブラリ関数timelocal()を利用して、日付からの曜日の取得などの日付け関連の処理を行っている。リスト中で、

```
$theday = timelocal(@theday);
@theday = localtime($theday);
```

となっている部分は、一見無意味な処理のように見えるが、リスト形式の日付けデータ（曜日データを含まない）から、timelocal関数を使っていったん数値形式の日付けデータに変換し、それを逆にlocaltime関数でリスト形式に戻すことによって、曜日の情報を得ている。ある日付の曜日がわかれば、その週の月曜日の日付がわかり、その月曜日が「何月の第何回目の月曜日か」を計算することができるわけだ。

あとは、算出した月と週のデータをもとに、週ごとのディレクトリを（必要に応じて）作成し、ファイルを移動すればよい。以上の処理を、与えられたディレクトリの中のすべての*.mp3ファイルごとに行う。

仕分けスクリプトの自動運転

スクリプトの作成が済んだら、次のコマンドでスクリプト

リスト1 weeklycleanupスクリプト（抜粋）

```
# get the date of last monday

@theday = (0,0,0,m_day($basename), m_month($basename) - 1,
m_year($basename),,,);
$theday = timelocal(@theday);
@theday = localtime($theday);
$monday = timelocal(@theday) - (86400 * ((@theday[6]+6)%7));
@monday = localtime($monday);
$date_of_monday = @monday[3];
$month_of_monday = @monday[4] + 1;
$week_of_monday = int(($date_of_monday - 1) / 7) + 1;

# get the weekly directory name.

$weekdir = "wk" . substr("00" . $month_of_monday, -2) .
substr("00" . $week_of_monday, -2);
```

掲載のリストは前号掲載のradiorecも含め、すべて付録CD-ROMに収録されています

ファイルに実行権限を設定する。

```
$ chmod +x ./weeklycleanup
```

次に、消してしまってもかまわないテスト用のファイルを作って、実際にうまく動くかどうかを試してみる。

```
$ touch aa200009010100_32K.mp3 aa200009040100_32K.mp3
$ ./weeklycleanup ./
$ ls
weeklycleanup*      wk0804/            wk0901/
$ ls wk0804
aa200009010100_32K.mp3
$ ls wk0901
aa200009040100_32K.mp3
```

2000年9月1日は、その週の月曜日が8月28日でその月の4回目の月曜日になるため、「8月第4週」にあたる。同様に、9月4日の週は9月第1週に判定されるので、これら2つのファイルはwk0804、wk0901のディレクトリにそれぞれ移動されている。

うまく動くことが確認できたら、このスクリプトを適当なディレクトリ（たとえば、/home/zen-t/Language）にコピーして、定期的に行うように設定する。これには、前から何度も登場している「cron」機能を利用する。録音の処理を行っているのと同じ、一般ユーザーとしてログインしている状態（ルートユーザーではない）で、

```
$ crontab -e
```

コマンドを実行し、起動したテキストエディタ上で、リスト2のような行を追加する。

この例では、毎週日曜日の午前0時5分から1分ごとに、各番組のディレクトリの中の整理を行うようにしている。設定する時間は各自の判断で変えればよいが、録音を行っている最中や、ユーザーが録音済み番組を利用する可能性のある時間帯は避けるべきだろう。

Webアクセスの実現

Linuxマシン上にあるファイルを、LANで接続されたほかのPCから閲覧できるようにするには、前述のようにファイ



ル共有を使うのがもっとも簡単な方法である。しかし、それではあまりにも不親切であるし、ファイル操作になれていないユーザーが、誤って録音済みのファイルを消してしまう可能性もある。そこで、いまや事実上の世界標準になった情報共有手段であるWebを使って、ブラウザから録音済みファイルにアクセスできるようにする。Webブラウザなら、マウスでリンクをクリックするだけの操作で利用できるの、ファイルの操作に不慣れなユーザーでも使いやすい。

Linuxマシンのハードディスク上にあるファイルを、Webページ経由で閲覧できるようにするのにあたって、必要になるのはWebサーバだけである。MP3ファイルをネットワークで公開するのに、ストリーミングサーバのようなものが必要になるのでは、と考える読者もいるかもしれないが、コラム「ストリーミングシステム」で説明するように、クライアントPC側でWebページ上のリンクをクリックすると、MP3再生ソフトで再生されるようにしたいだけなら、ストリーミングサーバは必要ない。

一般的なLinuxディストリビューションなら、Apacheは初めから付属している。Linuxのインストール時に「Webサーバ」や「Apache」などと記述されたパッケージを選択していれば、あらかじめインストールする必要すらない。ただし、最近のディストリビューションの多くでは、デフォルトのApacheの設定が「きわめて厳しい」設定になっている。たとえば、TurboLinux Workstation 6.0の場合、ローカルマシンからはWebサーバにアクセスできるが、同じLAN上であっても、外部からのWebサーバへのアクセスは拒否する設定になっている。このため、Apacheの設定、およびそれに関連するLinux側のネットワークの設定を変えなければならない。

ネットワークの設定

本記事でサンプルとしているシステムでは、何度も述べているようにTurboLinux Workstation 6.0を使用している。Linuxのデフォルトのネットワーク設定の状態は、ディストリビューションによって、また、同じディストリビューションでも、インストール時のオプションの選択状況によってかなり変わってくる。このため、ネットワークの設定やWebサ

ーバの設定方法は、各自の環境によって異なる場合があることを、あらかじめご了承ください。

本記事の「ラジオ自動録音システム」では、インターネットに常時接続していないLAN環境（あるいは、ダイヤルアップルータのNAT機能などを利用して、外部からは隔離された状態でインターネットに接続されている環境）を前提としている。より平たくいえば、IPアドレスとして「192.168.xxx.xxx」や「10.xxx.xxx.xxx」などのプライベートアドレスを使っている環境、と考えてもよいだろう。

LAN環境でのネットワーク設定の詳細については、ここではすべてを説明することはできない。本誌のネットワーク関連の記事や、市販の参考書などを参考に、自分で設定してもらいたい。ラジオ自動録音システムで必要になるのは、LAN内部のクライアントに対してWebページを公開できるようにすることだけである。必要となるであろう設定項目だけを簡単に挙げると、次のようになるだろう（Apache自体の設定については後述する）。

- ・サーバ/クライアントマシンへのIPアドレスの割り当て（/etc/hostsなど）
- ・外部から接続できるようにする設定（/etc/inetd.conf、/etc/hosts.allow、/etc/hosts.denyなどの確認と設定）
- ・Webページ中で公開するディレクトリのアクセス権設定（ユーザーのホームディレクトリのpublic_htmlディレクトリ、および録音済みMP3ファイルを保存してあるディレクトリを、すべてのユーザーから読み出し可にする）

これだけでは不親切かと思うので、設定のヒントを記しておこう（あくまでも「ヒント」なので、すべての環境でこの通りに設定できるというわけではない）。

- ・IPアドレスの割り当て
家庭内LANであれば、IPアドレスは「192.168.XXX.XXX」のプライベートアドレスを割り当てる。たとえば、「192.168.1.1」、「192.168.1.2」、「192.168.1.3」……、というように、サーバやクライアント各1台に1つずつ、アドレスを割

リスト2 crontab -eで追加する行（weeklycleanupを定期的に起動する）

```
5 0 * * 0 /home/zen-t/Language/weeklycleanup /home/zen-t/Language/English
6 0 * * 0 /home/zen-t/Language/weeklycleanup /home/zen-t/Language/English2
7 0 * * 0 /home/zen-t/Language/weeklycleanup /home/zen-t/Language/Business
8 0 * * 0 /home/zen-t/Language/weeklycleanup /home/zen-t/Language/French
9 0 * * 0 /home/zen-t/Language/weeklycleanup /home/zen-t/Language/Spanish
```

り当てるようにする。

IPの世界では、「ネットワークアドレス」というものも重要な意味を持つが、この例では、ネットワークアドレスは「192.168.1.0」となる。このネットワークアドレスについては、各種設定ファイル中では「192.168.1.0/24」や「192.168.1.0/255.255.255.0」というように、アドレスの数値の中で「ネットマスク」を含めた書式で記述することが多いので、覚えておくとよいだろう。

・外部からの接続設定

Webサーバを公開するだけなら必要ないことが多いが、場合によっては必要になることがある。まず、Webサーバにしたいマシン上で、クライアントとなるマシンから接続されていない状態で、

```
$ ps ax
```

を実行してみて、表示されるプロセス一覧の中に「httpd」というプロセスが含まれているかどうかを確認する。httpdプロセスが起動している場合は、すでにWebサーバが利用できるようになっているはずなので、クライアントマシンのWebブラウザからサーバに接続してみる。たとえば、WebサーバのIPアドレスが「192.168.1.1」ならば、ブラウザのアドレス入力バーなどに、

```
http://192.168.1.1/
```

と入力してみる。画面3のようなページが表示されれば、Webサーバ（Apache）は利用可能な状態になっているので、

Column

ストリーミングシステム

インターネットのWebサーバ上にある音声ファイルをクライアントマシンで聴取する場合、「RealPlayer」や「Windows Media Player」などの「ストリーミングシステム」が使われることが多い。とくに、128kbps以下の比較的低速な接続手段を使っている場合、数Mバイトもの音声ファイルをダウンロードしてから再生する方法だと、ダウンロードだけで相当の時間がかかり、再生が始まるまで十数分間も待たされることになる。ストリーミングメディアを使えば、「少しずつダウンロードしながら再生する」ことができるので、音声ファイルへのリンクをクリックしてすぐに再生されるようになるのだ（図）。

ファイルをいったんクライアントに転送してから再生する場合、ネットワークやサーバに一時的に高い負荷がかかる。一方、ストリーミングシステムなら、データを少しずつ転送するので、ネットワークやサーバへの負荷はそれほど高くないという利点もある。

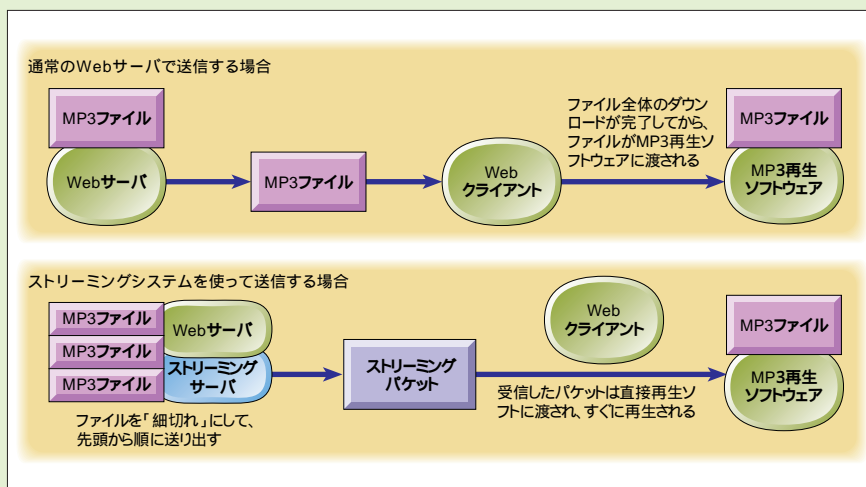
ストリーミングシステムを使えるようになるには、Webサーバとは別に「ストリーミングサーバ」と呼ばれるプログラムがサーバマシンに必要な。Linuxの場合、RealPlayer用の「Real Server」や、MP3形式のオーディオストリームを供給できる「icecast」などのサーバが利用できる。

RealAudioのサーバは、いわゆる「商用ソフトウェア」なので、正式な製品はお金を出して購入する必要があるのだが、無償で利用できる機能限定版もある。icecastはフリーで利用できるサーバなので、インターネットからソースファイルやRPM形式のバイナリファイルをダウンロードすれば、すぐにも利用できる。

ただし、Real Serverの場合、機能限定版だと送信できる音声のビットレートがかなり低い値に制限されるうえ、送信するファイルをMP3形式から別の形式に変換する必要がある。また、どちらのサーバも、あらかじめ定義された「プレイリスト」中の音声ファイルを順に再生するような用途を想定しており、

ユーザーから要求されたファイルをその都度再生するような使い方をすることは、設定がかなり複雑になる。

本記事の場合、LAN内部に限定したユーザーを想定しているので、利用できるネットワークの速度は、100Mbpsまたは10Mbpsのイーサネットということになる。数Mバイト程度のファイル転送なら、100Mbpsで2～3秒、10Mbpsでも30秒程度で完了するので、待ち時間はそれほど問題にならない。また、ネットワークやサーバにかかる負荷についても、家庭内で利用するのであれば、これも問題にはならないだろう。このため、あえて面倒なストリーミングシステムを使わなくてもよいと判断したのだ。



図



後述する公開ディレクトリの設定以外は必要ないはずだ。

httpdプロセスが見当たらない場合でも、/etc/rc.d/init.dディレクトリの中に「httpd」というコマンドファイル（シェルスクリプト）が存在するディストリビューション（RedHat系ディストリビューションなど、多くのLinuxがこれに該当する）ならば、suコマンドでrootユーザーになった状態で、

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

を実行してhttpdを起動できる。

psコマンドでhttpdが表示されない場合、かつ、/etc/rc.d/init.dディレクトリに「httpd」が存在しないような場合は、/etc/inetd.confファイルを参照して、次のような行が含まれているかどうかを調べる（行の内容はディストリビューションによって異なるかもしれない）。

```
http stream tcp nowait ....(後略)
```

もし、上のような行の先頭に「#」記号がつけられて、設定がコメントアウトされている場合は、suコマンドでrootユーザーになってから、テキストエディタなどで/etc/inetd.confファイルを開き、該当する行の先頭の「#」を取り除いてから、次のコマンドを実行してみる。

```
# kill -HUP <inetdプロセスのPID>
```

inetdプロセスのPIDは「ps ax」コマンドで調べればよい。この状態で、先ほどと同じようにクライアントマシンのWebブラウザから、サーバのWebページにアクセスできれば、Webサーバは使用可能な状態である。

以上のどれにも該当しない場合は、ディストリビューションのマニュアルなどを参照して、Webサーバを利用可能な状態に設定しなければならない。

・公開用ディレクトリのアクセス権設定

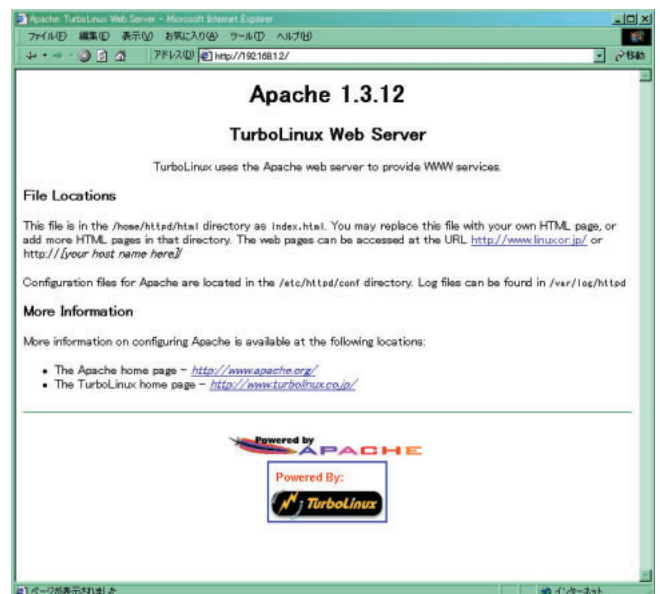
厳密には「ネットワークの設定」とはいえないが、マシン上のファイルをWebサーバを介して他のPCから閲覧できるようにするには、そのファイルやファイルの存在するディレクトリの「アクセス権」の設定を確認する必要がある。

Apacheのプログラムは、「root」でも一般ユーザーでもない「nobody」というユーザーID（ディストリビューションによっては別のユーザー名の場合もある）の権限で動作し

て、ローカルディスク上のファイルを読み出し、ネットワーク上のクライアントにデータを送信する。このため、Webページとして公開するHTMLファイルやその他のデータファイルは、「nobody」ユーザーが読み出し可能な状態になっていなければならない。今回のシステムでは、ユーザーに公開するページはすべてCGIによって自動生成するので、サーバ上にHTMLファイルは存在しないが、肝心の「録音ファイル」がnobodyユーザーに読み出し可能となっていないと、クライアントからアクセスできなくなる。

「nobody」ユーザーの読み出しを可能にするのに、ファイルの所有権までも「nobody」にする必要はまったくない。所有権はそのままにしておいて、「自分以外のユーザー」からの読み出し権限を「許可」するようにしておけばよい。また、録音済みMP3ファイルを収めているディレクトリについては、「自分以外のユーザー」が読み出し可能、かつ「実行可能」になっている必要がある。なお、ディレクトリを作成したりファイルを作成する場合に、特別な指定を行っていない限り、「その他のユーザー」に対する読み出し可能属性（およびディレクトリの場合は実行可能属性）が有効の状態になっているはずである。したがって、以下のアクセス権の変更は必要ないはずだ。

今回のサンプルでは、MP3ファイルは/home/zen-t/Languageというディレクトリ以下に収められているので、



画面3 Apacheのデフォルトページ

Apacheをインストールした状態（あるいはLinuxディストリビューションをインストールしたばかりの状態）で、そのWebサーバのルートにアクセスしてみると、サーバが正しく動作していればこのような画面が表示されるはずだ。ちなみに、このページのHTMLファイルは、サーバの/home/httpd/htmlに置かれているので、自分で好きなように変更することができる。

まずディレクトリの属性を確認する。

```
$ ls -ld /home/zen-t/Language
drwxrwxr-x  9 zen-t  zen-t  4096 Sep  5 18:27
/home/zen-t/Language/
```

このように、アクセス属性の最後の3文字が「r-x」になっていれば問題ない。このディレクトリ以下のすべてのディレクトリについても、同様の属性になっているかどうか確認する。もし読み出し属性と実行属性がついていなかったら、次のようなコマンドで属性を変更すればよい。

```
$ chmod o+rx 変更したいディレクトリ名
```

これらのディレクトリ中のMP3ファイルについても、同様に「読み出し可能」属性が有効になっているかどうかを確認する。

```
$ ls -l /home/zen-t/Language/English/*.mp3
```

Apacheの設定

Apache側で必要になる設定は、公開ディレクトリに対する外部からのアクセスを許可することと、CGIプログラムを配置したディレクトリでの「CGI実行」を許可することである。また、今回はMP3ファイルを保存しているディレクトリが、通常のWebページ公開用のディレクトリ（各ユーザーのホームディレクトリにあるpublic_htmlディレクトリ）とは異なるので、そのディレクトリに対するシンボリックリンクを公開用ディレクトリに作成することになる。したがって、シンボリックリンクのリンク先へのアクセスを許可するような設定が必要になる。

リスト3 srm.confの変更部分

```
.(前略)
.
.
# ScriptAlias: This controls which directories contain
server scripts.
# Format: ScriptAlias fakename realname

ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
.
.
.(後略)
```

・公開用ディレクトリの準備

通常、公開用のWebページを作成する場合には、自分のホームディレクトリに「public_html」というディレクトリを作り、その下にHTMLファイルを作成する。今回は、あらかじめ作成するHTMLファイルはないのだが、MP3ファイルを一般ユーザーのホームディレクトリ上に保存している関係から、「ユーザーのホームページのURL」経由でファイルにアクセスするようにしたい。このため、該当ユーザーのホームディレクトリにpublic_htmlディレクトリを作成し、そのディレクトリの中に、録音済みMP3ファイルを保存しているディレクトリへのシンボリックリンクを作成する。

```
$ cd /home/zen-t
$ mkdir public_html
$ cd public_html
$ ln -s /home/zen-t/Language ./language
```

・srm.confの設定

Apacheの各種動作の設定は、/etc/httpd/confディレクトリにある設定用のファイルsrm.conf、access.conf、

リスト4 access.confの設定例（関連するディレクトリ部分のみ抜粋）

```
<Directory />
    order deny,allow
    deny from all
    Options None
    AllowOverride All
</Directory>

<Directory /home/httpd/html>
    Options Indexes Includes FollowSymLinks
    AllowOverride None
    order allow,deny
    allow from 192.168.1.0/24
</Directory>

<Directory /home/httpd/cgi-bin>
    AllowOverride None
    allow from 192.168.1.0/24
    Options ExecCGI
</Directory>

<Directory /home/*/public_html>
    order deny,allow
    deny from all
    allow from 192.168.1.0/24
    Options All
    Options +ExecCGI
</Directory>
```



httpd.confなどの記述を変更することで行う。このうち、srm.conf、access.confが今回のシステムで設定が必要になる部分である。ただし、ディストリビューションによっては、srm.conf、access.confの内容がすべてhttpd.confに書かれていて、srm.conf、access.confはコメントだけになっていることもある。その場合はhttpd.confに書かれた同じ部分を変更すればいい。なお、以下の作業はroot権限で行うこと。

srm.confではおもにWebサーバが内部的に使用するディレクトリの定義や、ディレクトリの別名などの設定、CGIなどの特殊機能を処理する「ハンドラ」の設定などを行う。通常、Apacheをインストールすると自動的に作成されるsrm.confファイルには、設定の意味などを記した詳しいコメント文がついており、それに従って設定を行えばよい。CGIスクリプトを利用できるようにするには、リスト3のように、CGIスクリプトのハンドラを定義している行のコメント記号を外せばよい。srm.conf、および後述するaccess.confの設定を変更したら、次のコマンドでhttpdを再起動する。

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

・access.confの設定

access.confでは、クライアント（すなわちWebブラウザ）からのアクセスがあったときに、それをどのように許可したり拒否したりするかを主に設定する。Apacheの場合、アクセスの可否はディレクトリ単位で行うことができる。

TurboLinux Workstation 6.0では、デフォルトの状態では外部からのWebページへのアクセスはできないように設定されている。このままではクライアントから一切利用できないので、access.confを編集して、必要なディレクトリに関してのみ、外部からのアクセスを許すように設定する。今回は、LAN内部からのアクセスしか想定しないので、LANで利用しているネットワークアドレスからのアクセスに限って、制約なしにアクセスできるように設定する。設定例については、リスト4を参照してもらいたい。要点としては、必要なディレクトリに対して、

```
allow from 192.168.1.0/24
```

という設定を追加すること、/home/httpd/cgi-binディレクトリに対して、

```
Options +ExecCGI
```

という設定を追加すること、そして、/home/*/public_htmlディレクトリ（各ユーザーのホームディレクトリ）に対して、

```
Options +FollowSymlinks
```

の設定を追加することである。

CGIを使う

CGI（Common Gateway Interface）とは、平たくいうと、Webクライアントからの要求を受けて、Webサーバ上で別のプログラムを実行し、その実行結果をHTTPプロトコルでWebクライアントに返す仕組みである（図3）。このように書いても、まだ難しそうに思えるかもしれないが、標準出力にHTTP形式の文字列を出力するようなプログラムなら、どんなものでもCGIプログラムになるのである。使用するプログラミング言語としては、Perlの人气が高く、本記事でもPerlを使っているが、実際にはシェルスクリプトでも、コンパイル済みのC言語プログラムでもなんでもよい。

CGIプログラムの作成とインストールは簡単である。HTML文字列を出力するようなプログラムを作成したら、その実行可能なスクリプトファイル（あるいはコンパイル済み実行ファイル）を、CGIプログラム用のディレクトリにコピーするだけでよい。今回は、/home/httpd/cgi-binディレクトリをCGIプログラム用に設定してあるので、このディレクトリにプログラムファイルを配置する。クライアントからサーバのCGIプログラムにアクセスするには、URLでそのプログラムの位置を指定するだけでよい。たとえば、/home/httpd/ディレクトリは、Webサーバの「ドキュメントルート」、すなわちURLでサーバ名のみを指定した場合の「デフォルトページ」となり、/home/httpd/cgi-binは「http://サーバ名/cgi-bin」でアクセスできる。したがって、「langserv」という名前のサーバの「/home/httpd/cgi-bin/mycgi」というCGIプログラムにアクセスするには、

```
http://langserv/cgi-bin/mycgi
```

というURLをWebブラウザに入力すればよいのだ。

ページ構成

ラジオ自動録音システムでは、図4のようなページ構成をとることにした。HTMLで直接記述したページは一切使用

せずに、すべてCGIスクリプトで生成するページから構成する。録音している番組の一覧を表示する「目次」ページ、各番組ごとに、その週に録音したファイルと過去の週別のページへのインデックスを表示するページ、そして、過去の録音について、1週間単位でファイルのインデックスを表示するページの3種類である。

作成すべきCGIスクリプトは、それぞれの種類のページについて1つ、つまり、合計3つのスクリプトファイルを作成すればよいことになる。

なお、スクリプトを単純化するために、番組の識別用コードと番組名、番組の録音ファイルを保存するディレクトリ名などの情報は、別のファイルにまとめるようにした。このため、録音する番組を変更したり、数を増減させたりした場合でも、変更すべきスクリプトファイルは1つだけで済むようになっている。このため、実際に作成したスクリプトは、各ページを表示する3つのファイル (langindex、language、weeklyindex) のほかに、番組に関する定義を行っているファイル (programdef.pl) そして、日付け関連の処理を行うルーチンだけを抜き出したファイル (programdate.pl) の、合計5つである (リスト5~9)。

CGIスクリプトを作る

最初の「大目次」を表示するスクリプト「langindex」はきわめて単純なもので、番組を定義している変数の内容に従って、各番組の「タイトル画面」を表示するためのCGI (language) を呼び出す「リンク」を並べているだけだ (画面4)。

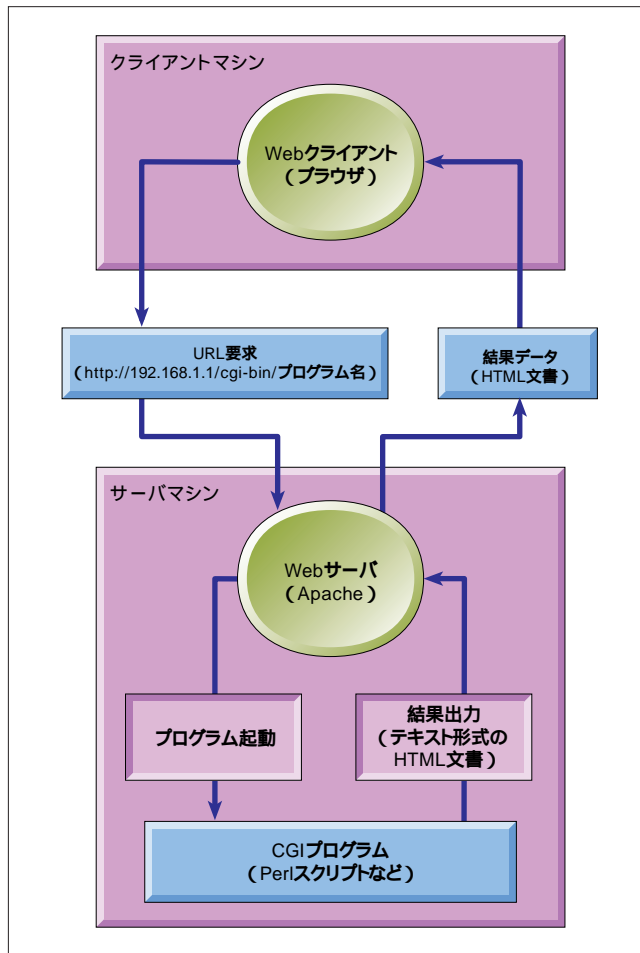


図3 CGIのしくみ
クライアントからサーバへは「URL」が渡されるが、この中にサーバに実行を依頼するプログラム (スクリプト) の名前が含まれている。サーバはクライアントから渡されたURLがCGIプログラムを示していると判断すると、そのプログラムを起動して、実行結果をHTML文書の形式で受け取ると、それをクライアントに返す。

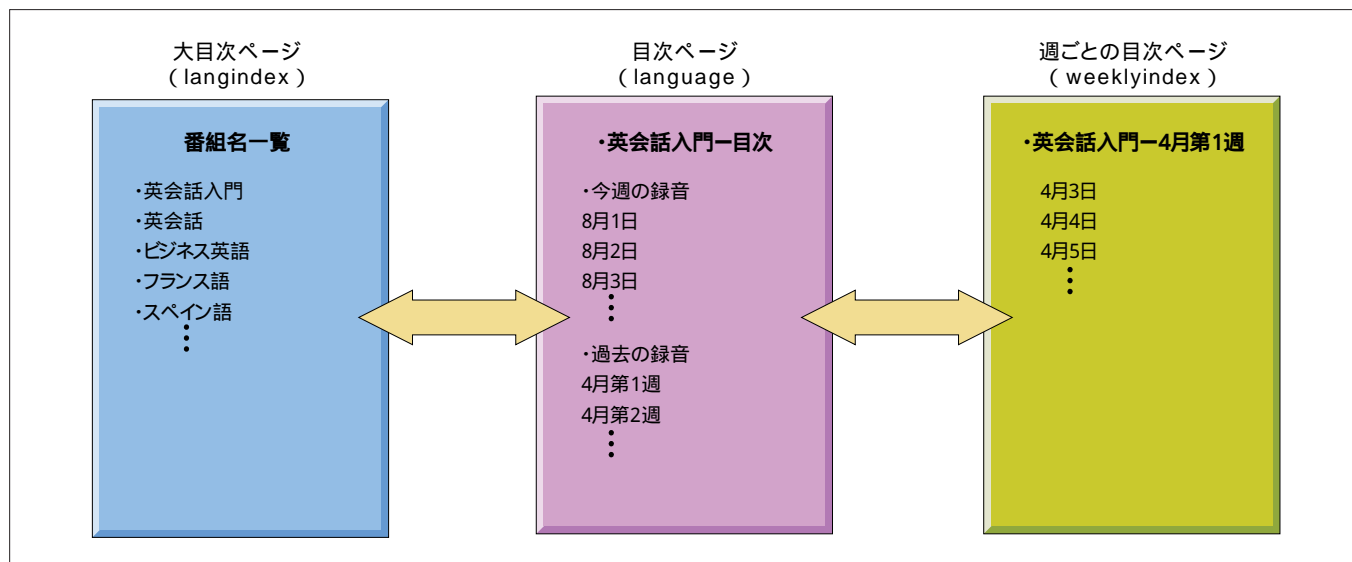


図4 Webページの構成
Webページは3段階の「目次」ページで構成する。目次中の項目をクリックすれば、次の階層の目次ページへ移ったり、特定の日付の録音ファイルが再生されるようにする。前の階層のページへ戻るための機能は特に設けず、ブラウザの「戻る」ボタンなどを使うことを前提にする (そのほうがユーザーにとって使いやすいと考えたので)。



リスト5 programdef.pl (番組タイトルやサーバなどの情報を設定)

```
# program and host definition
$webhostname = "zen02";
$librarydir = "/home/zen-t/Language";
$surldir = "http://$webhostname/~zen-t/language/";

@programid = ("ec", "ed", "be", "fr", "sp");
$programname{"ec"} = "英会話入門"; $dirname{"ec"} =
"English";
$programname{"ed"} = "英会話"; $dirname{"ed"} =
"English2";
$programname{"be"} = "ビジネス英語"; $dirname{"be"} =
"Business";
$programname{"fr"} = "フランス語講座"; $dirname{"fr"} =
"French";
$programname{"sp"} = "スペイン語講座"; $dirname{"sp"} =
"Spanish";
```

リスト7 langindex (大目次表示用CGIスクリプト)

```
#!/usr/bin/perl

# languageindex(.pl) -
# user interface cgi script for radio recording system
# Copyright (c) 2000 by Zentaro Takeda
# NOT FOR CRITICAL OR FATAL PURPOSE
# PERSONAL USE ONLY

require "programdef.pl";

# Header

print "Content-type: text/html\n\n";
print "<HTML>\n";
print "<head><title>NHKラジオ語学番組ライブラリ --- 目次
</title></head>\n";

# title
print "<H1><center>NHKラジオ語学番組ライブラリ
</center></H1>\n";
print "<HR>\n";

# list

print "<H2>番組名一覧</H2>\n";
print "<UL>\n";
foreach $id (@programid) {
    print "<LI>";
    print "<A href = http://".$webhostname."/cgi-
bin/language?". $id.
">".$programname{$id}."</A>\n";
    print "</LI>\n";
}
print "</UL>\n";

print "<HR>\n";

#footer
print "This page is provided for personal use only.\n";

#closing
print "</HTML>\n";
```

リスト6 programdate.pl (日付け処理ライブラリ)

```
require "timelocal.pl";

$fformat = "ppyyyymmddhii_bb_";

sub m_extract { # $basename, $format, $pattern
    local($bn) = @_[0];
    local($ff) = @_[1];
    local($ptn) = @_[2];
    local($ll) = index($ff,$ptn);

    if ($ll < 0) {die "format error!!!";}

    local($ret) = substr($bn, $ll, length($ptn));
    return $ret;
}

sub m_year {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "yyyy");
    return $ret;
}

sub m_month {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "mm");
    return $ret;
}

sub m_day {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "dd");
    return $ret;
}

sub m_hour {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "hh");
    return $ret;
}

sub m_minute {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "ii");
    return $ret;
}

sub m_bps {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "bb");
    return $ret;
}

sub m_program {
    local($bn) = @_[0];
    local($ret) = m_extract ($bn, $fformat, "pp");
    return $ret;
}
```

各番組ごとのタイトル画面（画面5）を呼び出すには、languageスクリプトに、引数として番組の識別コード（2文字の文字列）を与える。たとえば、「英会話入門」のページを呼び出すには、英会話入門の識別コード「ec」を追加して、

```
http://サーバ名/cgi-bin/language?ec
```

というURLを使えばよいわけだ。

番組ごとの目次画面には、その週に録音されたMP3ファイル（すなわち、前述のweeklycleanupスクリプトで週ごとのディレクトリに分配される以前のファイル）の一覧と、過去の週の一覧が表示される。MP3ファイルへのリンクは、

MP3ファイルそのものに直接リンクされているので、クライアントのWebブラウザでこのリンクをクリックすれば、ファイルのダウンロードや再生（クライアントの設定によって動作は異なる）を行うことができる。各週へのリンクは、週ごとのページを表示するスクリプト（weeklyindex）に対して、引数として番組識別コード、月、週の番号の3つを与えるようになっている。たとえば、「英会話入門」の「8月第1週」のページを呼び出すには、

```
http://サーバ名/cgi-bin/weeklyindex?ec?08?01
```

というURLが使われるわけだ。

リスト8 language（番組ごとの目次表示用CGIスクリプト）

```
#!/usr/bin/perl

# language(.pl) -
# user interface cgi script for radio recording system
# Copyright (c) 2000 by Zentaro Takeda
# NOT FOR CRITICAL OR FATAL PURPOSE
# PERSONAL USE ONLY

require "programdef.pl";
require "programdate.pl";

$id = @ARGV[0];

# Header

print "Content-type: text/html\n";
print "<HTML>\n";
print "<head><title>NHKラジオ語学番組ライブラリ ---
$.programname{$id}.
</title></head>\n";

# title
print "<H1><center>NHKラジオ語学番組ライブラリ
</center></H1>\n";
print "<HR>\n";

# get directory information
opendir (PROGDIR, $librarydir."/".$dirname{$id}) or die
"No Such Directory\n";
@thisweek = sort( grep /^$id.*\.mp3$/, readdir(PROGDIR));
closedir(PROGDIR);

print "<H2>$programname{$id} --- 今週の録音</H2>\n";
print "<UL>\n";
foreach $twprog (@thisweek) {
    $p_month = 0 + &m_month($twprog);
    $p_date = 0 + &m_day($twprog);

    $p_bitrate = 0 + &m_bps($twprog);
    print "<LI>";
    print "<A href =
$.urlldir.$dirname{$id}."/".$twprog.">".
    $p_month."月".$p_date."日(".$p_bitrate."Kbps)</A>\n";
    print "</LI>";
}
print "</UL>\n";

opendir (PROGDIR, $librarydir."/".$dirname{$id}) or die
"No Such Directory\n";
@alldir = sort( grep /^wk[0-9]*$/, readdir(PROGDIR));
closedir(PROGDIR);

print "<H2>$programname{$id} --- 過去の録音</H2>\n";

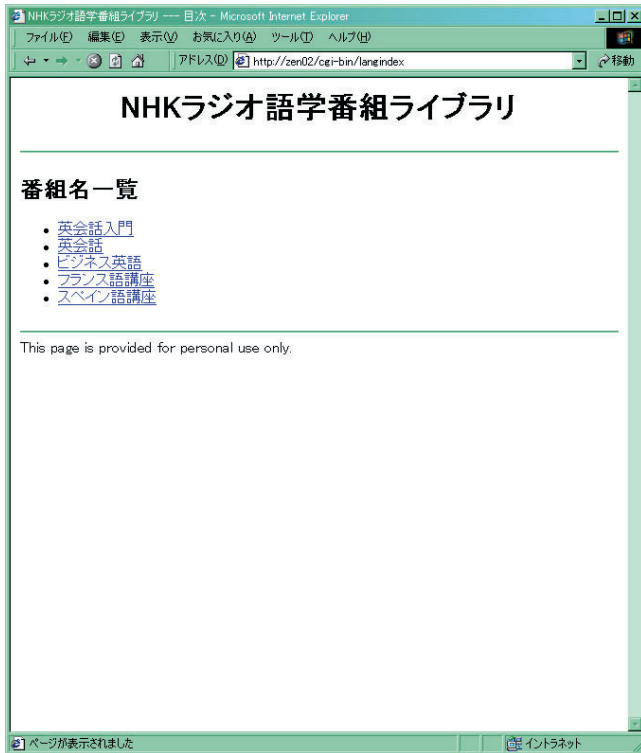
print "<UL>\n";

foreach $weeklydir (@alldir) {
    $month= substr($weeklydir,2,2);
    $week= substr($weeklydir,4,2);
    print "<LI>";
    print "<A href = http://".$webhostname."/cgi-
bin/weeklyindex?".$id.
    "+$month+$week>".($month+0)."月 第".($week+0)."週
</A>\n";
    print "</LI>";
}

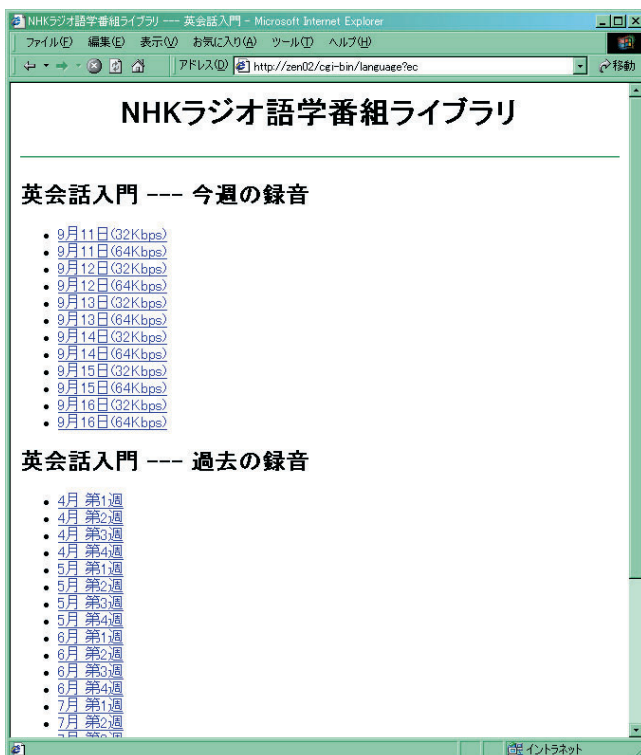
print "</UL>\n";
print "<HR>\n";

#footer
print "This page is provided for personal use only.\n";

#closing
print "</HTML>\n";
```



画面4 タイトル(大目次)のページ
録音している番組の一覧が表示される。番組の情報は、リスト5のprogramdef.plで定義しているため、録音する番組を変更した場合は、その定義を書き換えるだけで、このページの内容も自動的に変わる。



画面5 各番組ごとのタイトル(目次)ページ
最新の録音ファイル一覧と、過去の録音の一覧(週ごと)が表示される。最新の録音については、日付をクリックすればすぐに再生できる。週ごとの項目をクリックすると、画面6に移動する。

リスト9 weeklyindex (週ごとの目次表示用CGIスクリプト)

```
#!/usr/bin/perl

# weeklyindex(.pl) -
# user interface cgi script for radio recording system
# Copyright (c) 2000 by Zentaro Takeda
# NOT FOR CRITICAL OR FATAL PURPOSE
# PERSONAL USE ONLY

require "programdef.pl";
require "programdate.pl";

$id = @ARGV[0]; $mt = @ARGV[1]; $wk = @ARGV[2];

# Header

print "Content-type: text/html\n\n";
print "<HTML>\n";
print "<head><title>NHKラジオ語学番組ライブラリ ---
\".$programme{$id}.
\" -- (\".(0+$mt).\"月 第\".(0+$wk).\"週)</title></head>\n";

# title
print "<H1><center>NHKラジオ語学番組ライブラリ
</center></H1>\n";
print "<HR>\n";

# get directory information
opendir (PROGDIR,
$librarydir."/\".$dirname{$id}"/wk\".$mt.$wk."/")
or die "No Such Directory\n";
@thisweek = sort( grep /^$id.*\.mp3$/,
readdir(PROGDIR));
closedir(PROGDIR);

print "<H2>\".$programme{$id}.\" --- \".(0+$mt).\"月 第
\".(0+$wk).\"週の録音</H2>\n";
print "<UL>\n";
foreach $twprog (@thisweek) {
    $p_month = 0 + &m_month($twprog);
    $p_date = 0 + &m_day($twprog);
    $p_bitrate = 0 + &m_bps($twprog);
    print "<LI>";
    print "<A href =
\".$urldir.$dirname{$id}"/wk\".$mt.$wk."/\".$twprog.\">\".
    $p_month.\"月\".$p_date.\"日\"(\".$p_bitrate.\"Kbps)</A>\n";
    print "</LI>";
}
print "</UL>\n";
print "<HR>\n";

#footer
print "This page is provided for personal use only.\n";

#closing
print "</HTML>\n";
```

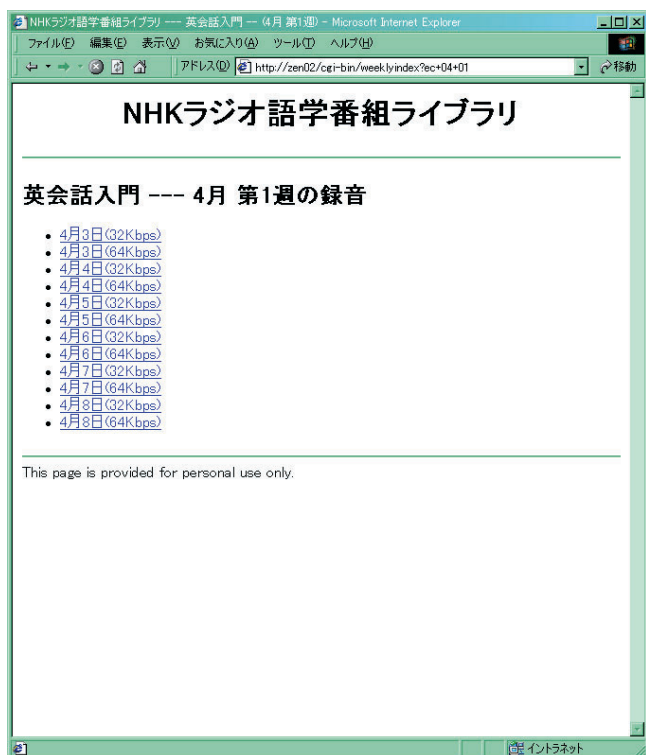
週ごとのページは画面6のように、日付け順にMP3ファイルへのリンクが並ぶようになる。ここでも、クライアント側でこのリンクをクリックすれば、MP3ファイルに直接アクセスできるようになる。

各スクリプト内部の処理の流れはほとんど同じである。Perlのopendir関数で目的のディレクトリをオープンし、readdir関数でディレクトリ中のファイル一覧を取得して名前順にソートし、その内容を元にリンクを作成しているだけだ。日付け関連の処理には、前出のweeklyindexで作成したルーチンを流用している。

いずれのスクリプトも、/home/httpd/cgi-binディレクトリ上に配置した後に、次のようにアクセス権を設定する。

```
$ su
# cd /home/httpd/cgi-bin
# chmod o-w+r+x ./langindex ./weeklyindex ./language
```

クライアントマシンのWebブラウザから「http://サーバ名/cgi-bin/langindex」にアクセスしてみて、各リンクへ正



画面6 週ごとの目次ページ

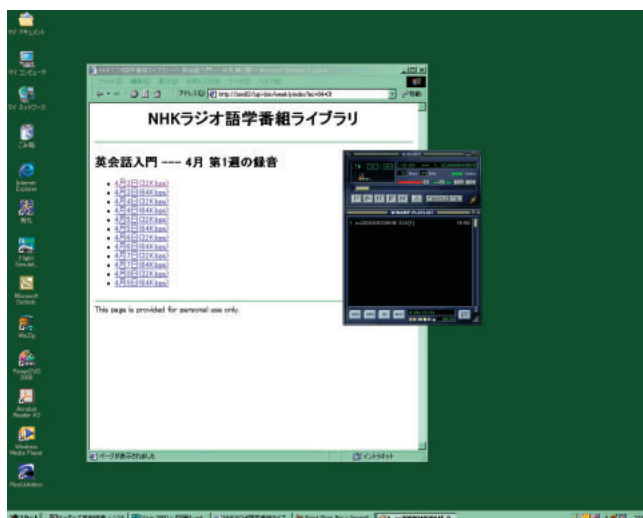
過去の録音を1週間単位に表示するページ。日付をクリックすると、再生が始まる。前の画面に戻るには、ブラウザの「戻る」機能を使う。ページ中に「戻る」リンクを作成することもできたのだが、スクリプトを少しでもシンプルにしたかったのと、筆者自身が、Webページ中のナビゲーションリンクが好きではない（ページの作者によって動作が異なるのが気に食わない）ので、あえてつけなかった。

しくジャンプできるかどうかを確認する。もし、「forbidden」や「not found」などのエラーメッセージが表示されるようなら、スクリプトファイル自体のアクセス権（読み出し可能属性と実行可能属性が必要）/etc/httpd/access.confの設定（/home/httpd/cgi-binディレクトリにExecCGIオプションと、ローカルネットワーク経由のアクセス権が必要）などが間違っていないかどうか確認すること。

クライアントの設定

クライアントマシンからのアクセスには、Internet Explorer、Netscape NavigatorなどのWebブラウザが利用できる。このほか、MP3ファイルを再生するためのアプリケーションが必要になる。たとえば、Windowsマシンでは、WinAMPやWindows Media Playerがインストールされていれば、ブラウザ上で目的のファイルのリンクをクリックすると、自動的にそのファイルが一時ファイルとしてクライアントマシンにダウンロードされ、音声の再生が開始される（画面7）。再生が終了すれば、ダウンロードされた一時ファイルは自動的に消去される。

クライアントがLinuxマシンなら、Netscape Navigatorとxmmsなどの音声再生ソフトウェアを利用することになるだろう。ただし、デフォルトの状態では、ファイルの再生を自動的に行えないことが多い。別掲のコラム「Netscape NavigatorからMP3ファイルを再生する方法」を参照して、設定する必要がある。



画面7 Windows上のWinAmpで再生している状態

Windowsをクライアントしている場合は、MP3再生用ソフトウェアさえインストールしてあれば、ブラウザ中のリンクをクリックするだけで、すぐに再生が始まる。ただし、ファイルをダウンロードするか、その場で実行するかを選択するダイアログが表示されることもある。MP3ファイルなら「この場所で実行」を選択しても問題ないだろう。



応用はアイデア次第

「CGIを使ったWebページによる録音済みMP3ファイルの公開」と表現すると、ずいぶん大げさになってしまうが、実際にはたいしたことではないのがわかってもらえただろうか。毎日決まった時刻に自動的に録音した数百ものMP3ファイルも、簡単なプログラムを使って自動的に整理して、わかりやすいWebページのインターフェイスからアクセスでき

るようになるのだ。

本記事の自動録音システムを発展させれば、たとえばWebページから新たな録音予約を追加したり、日付けから番組ファイルを「検索」したりする機能を加えることもできるだろう。また、シリアルインターフェイスからの制御が可能な無線受信機（アマチュア無線用の受信機には、そのような製品があるようだ）を使えば、複数の放送局の番組を切り替えて自動録音できるシステムも作れるはずだ。読者自身の興味と必要に応じて、チャレンジしてみてもらいたい。

Column

NetscapeからMP3ファイルを再生する方法

クライアントにWindowsマシンを使用している場合、Windows Media PlayerやWinAMPなど、MP3ファイルの再生が可能なアプリケーションがインストールされていれば、WebページからのMP3ファイルの再生を行うのに特別な設定は必要ない。場合によっては画面上にダイアログが表示されることもあるが、それに従って操作すれば自動的に再生を開始できる。

Linuxをクライアントに使っていたり、あるいは録音を行っているLinuxマシン上で、Webページからの再生を行おうとしたりする場合は、WebブラウザにはNetscape Navigatorを使うことになるのだが、Navigatorでは、インストールされたままの状態では、音声ファイルの自動再生はできないことがある。音声の再生を行う「ヘルパーアプリケーション」の場所などを、あらかじめ設定しておかなければならないからだ。

LinuxでMP3ファイルの再生を行うアプリケーションには何種類があるが、ここでは「xmms」を使用する場合を想定して、Navigatorからそのアプリケーションを起動し

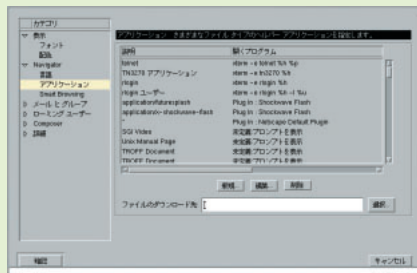
て、音声ファイルを自動再生できるようにする方法を説明しよう。

- ・X Windowを立ち上げた状態で、Netscape Navigatorを起動する。
- ・[編集]メニューの[設定]を選択する。
- ・[Netscape: 設定]ダイアログが表示されるので、左側のペインの[Navigator]という項目の左側の三角印をクリックし、開いた小項目の中の[アプリケーション]をクリックする(画面A)。
- ・右側のリストボックスの中から[MPEG Audio]という行を探してクリックし、リストボックス下にある[編集]ボタンをクリックする。
- ・開いたダイアログボックスの中の[拡張子]というテキストボックスの最後に“, mp3”を追加する(Netscapeのバージョンによって、追加が不要なこともある)。さらに、[アプリケーション]ラジオボタンをクリックしてから、右のテキストボックスに“/usr/local/bin/xmms %s”と入力する(実際にxmmsがインストールされているパス名を入力すること)(画面B)。
- ・[確認]をクリックして、ダイアログを閉じる。

- ・もう一度、画面Aの状態ですべてのボタンをクリックする。
- ・開いたダイアログボックスの各フィールドを、画面Cのように設定する。
- ・[確認]をクリックしてダイアログを閉じる。
- ・[Netscape: 設定]ダイアログの[確認]ボタンをクリックする。

Linuxのマルチメディアシステムについて知識がある読者なら、「audio/mpeg」と「audio/x-mpeg」の2つのMIMEタイプを設定するのは間違っているのではと考えるかもしれない。「audio/x-mpeg」という設定は、mpegデータの扱いが公式に決まっていなかった頃の名残なのだが、Webサーバの種類やバージョンによって、MP3データの種類の「audio/x-mpeg」として返すものと「audio/mpeg」として返すものが混在しているのが現状なので、両方の設定を行っておいたほうが都合がよいのだ。

ちなみに、現在のApacheでは「audio/mpeg」を返すようになっているが、Navigatorのデフォルトの設定では、「audio/x-mpeg」のMIMEタイプしか用意されていないので、上述のように自分で追加する必要がある。



画面A



画面B



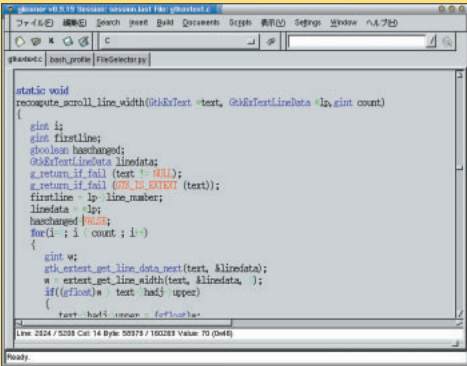
画面C

Free Application Showcase

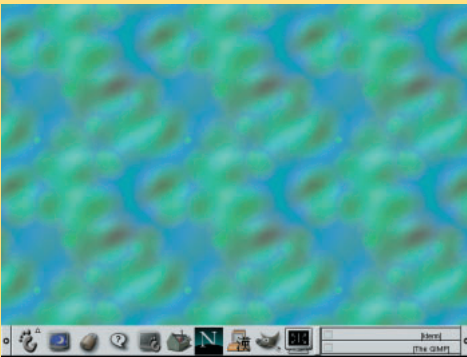
文：出井 一
Text : Hajime Dei



P.134 Asteroid3D



P.124 Glimmer



P.137 Starfish

- 画面モード (解像度) を変更するアプレット
GnomeRes  **123**
- プログラム作成に適したカラー構文エディタ
Glimmer  **124**
- GNOMEの高級科学電卓
rCalc  **126**
- 完全フリーの音声圧縮技術を体感しよう
OggEnc  **128**
- 2つのファイルの違いをカラー表示
gtkdiff  **130**
- 正規表現のマッチを目で確認
RegExplorer  **132**
- アステロイドを破壊する3Dゲーム
Asteroids3D  **134**
- サムネイル付きの軽量画像ビューア
xzgv  **135**
- 3D表示のネットワークトラフィックモニタ
WiredView  **136**
- ルートウィンドウに美しいパターンを表示
Starfish  **137**

紹介したソフトは、すべて付録CD-ROMに収録されています。

画面モード (解像度) を変更するアプレット

GnomeRes

バージョン: 0.5.0 / 0.6.0 ライセンス: GPL

<http://www.sci.fi/~syrjala/gnomeres/>

ビルドから起動まで

GnomeResは、tarボールとRPMパッケージの両方が配布されている。GNOME 1.0用の0.5.0とGNOME 1.2用の0.6.0があるので、使用しているGNOMEに合わせて選択しよう。tarボールからのビルドとインストールは、「./configure」「make」「make install」という一般的な手順だ。

コマンドラインで「gnomeres_applet&」とするか、GNOMEパネル上で右クリックし、ポップアップメニューの[アプレット] - [ユーティリティ] - [GnomeRes]として起動しよう。パネル上にGnomeResのアイコンが表示され、クリックすると切り替え可能な解像度のメニューが表示される(画面1)。

XF86Configの設定

画面モードを切り替えるには、Xの設定ファイル(/etc/X11/XF86Config)のScreenセクションで、複数の画面モ

ードの設定をあらかじめ済ませておく必要がある。

Screenセクションは、画面の色深度(Depth)ごとにDisplayサブセクションに分かれている。この中の画面モード(Modes)として、Monitorセクションで設定したモードライン名("1024x768"など)を並べて書けばいい。

たとえば、色深度16のハイカラー(65536色)で、1280x1024、1024x768、800x600、640x480の4種類の解像度を利用する場合は、

```
Subsection "Display"
    Depth 16
    Modes "1280x1024" "1024x768"
    "800x600" "640x480"
EndSubsection
```

とする。

先頭の画面モードが起動時に使われるので、記述する順番に注意しよう。

GnomeResは、GNOMEパネルから簡単にXの画面モード(解像度)を変更できるようにするGNOMEアプレットだ。Xでは、もともとCtrl - Alt - + / - (テンキー)で切り替えが可能だが、目的の解像度になるまで操作を繰り返す必要があるし、テンキーのないキーボード(HHKなど)では利用できない。GnomeResを使えば、メニューからの選択やマウスホイールの回転操作だけで解像度を簡単に変更できる。

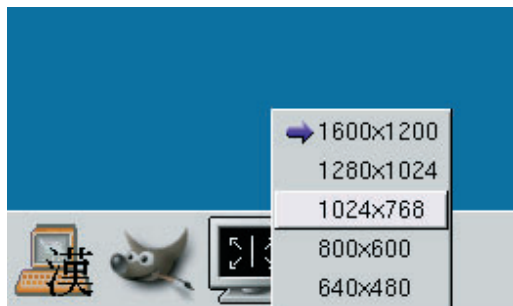
解像度が低い場合は、最高解像度のサイズの仮想画面が用意される。

マウスホイールによる切り替え

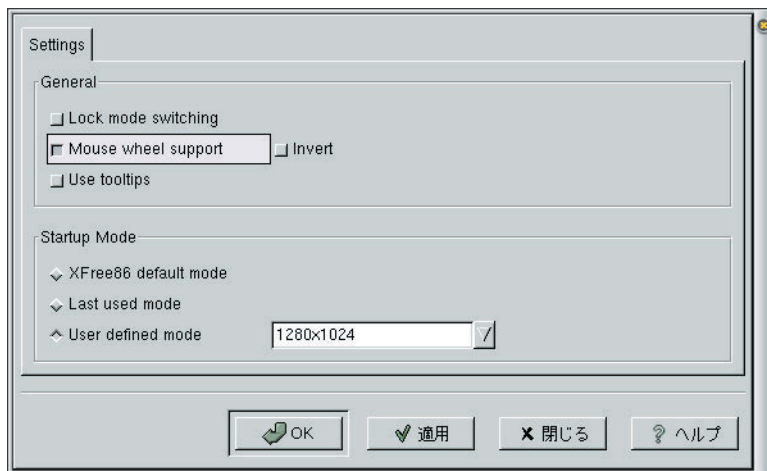
パネル上のGnomeResのアイコンを右クリックし、メニューから[Properties]を選択すると、設定ダイアログが開く(画面2)。ここでは、解像度切り替えの禁止(ロック)や、起動時の画面モードの設定などが可能だ。

たとえば、[Mouse wheel support]をチェックすると、GnomeResのアイコン上でのホイールの回転操作により解像度が切り替わるようになる。通常は、手前にホイールを回すと、Modesの設定の先頭から末尾方向に解像度が順次切り替わる(これを逆向きにすることも可能)。

また、[Startup Mode]では、X起動時の画面モードを、XFree86のデフォルトのモードや、最後に切り替えた解像度、指定した解像度の3通りから設定することができる。



画面1
利用できる解像度がメニューに一覧表示される。



画面2
設定ダイアログでは起動時の画面モードなどを設定できる。

プログラム作成に適したカラー構文エディタ

Glimmer

バージョン: 0.9.19

ライセンス: GPL

<http://glimmer.sourceforge.net/>

ビルドとインストール

Glimmerは、ソースのtarボールとRPMパッケージの両方が配布されている。日本語対応のためにソースを修正する必要があるので、tarボールを利用したほうが簡単だ。

Glimmerでは、GTK + 標準のテキストウィジェットの代わりに、独自の拡張テキストウィジェット (GtkExText) を使用する。GtkExTextは日本語EUCなどのマルチバイト文字に対応していないため、そのままだとカーソル移動や文字の削除の際に問題が生ずる。また、XIMを利用した日本語入力の際、変換前の文字列が隠れて見えないという不具合もある。

これらの修正に加え、日本語フォントを含むフォントセットを読み込むように変更を行い、日本語の表示や編集を可能にするパッチ (glimmer-euc.patch) を用意した。

Glimmerのtarボールを展開したディレクトリで、

```
$ patch -p1 < glimmer-euc.patch
```

とすると、ソースが修正される。

その後、「./configure」「make」「make install」という一般的な手順でビルドとインストールを行えばいい。

画面と初期設定

「glimmer&」として起動すると、メニューやツールバーを持つウィンドウが開く (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 (「開く」など) は日本語で表示される。

ファイルを読み込むには、ツールバーの[Open source file]ボタンを押して、独自のオープンダイアログから選択する。また、gmcなどGNOME対応ファイルマネージャからのドラッグ&ドロップにも対応している。

編集領域は上部のタブでファイルを切り替えるタイプだ (タブ位置は変更可)。1つのウィンドウで複数のファイルを同時に編集できるほか、Glimmerのウィンドウを複数開くことも可能だ。

Glimmerは、GNOME環境で動作する本格的なテキストエディタだ。C/C++やシェルスクリプト、Perl、HTML、XMLなど各種のプログラム言語やマークアップ言語のソースをカラー構文表示できる。また、マクロ言語としてPythonが採用されており、ファイルセクタなどのスクリプトが付属する。ソースの一部修正とフォントセットの設定により、日本語EUCのテキストの表示や編集にも対応できる。

GNOMEのセッション管理に対応しているため、終了時に編集していたファイルを覚えており、次回起動時に自動的にオープンする。

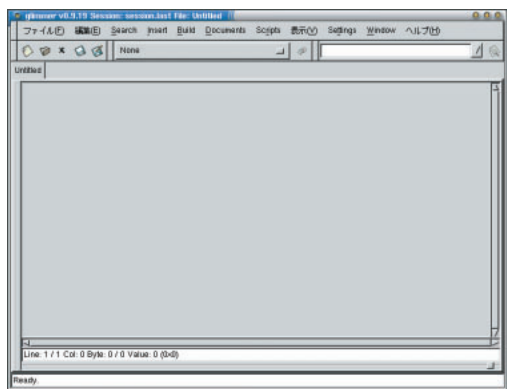
最初に、日本語を表示するためのフォントセットの設定を行おう。メニューの[Settings] - [設定]を選択すると、設定項目のジャンルがツリー表示された設定ダイアログが開く。フォントの設定は、[Style settings] - [Fonts]だ (画面2)。

[Default font]ボタンを押して、GTK + のフォント選択ダイアログを使って、既定値の「Courier」から、日本語を含むフォントセット (「Fixed」など) に変更しよう。

イタリック、ボールド、ボールドイタリックについても、同様に設定する。これらはカラー構文表示で利用されるだけなので、該当するフォントセットを用意できない場合は、初期設定のままでも構わない。

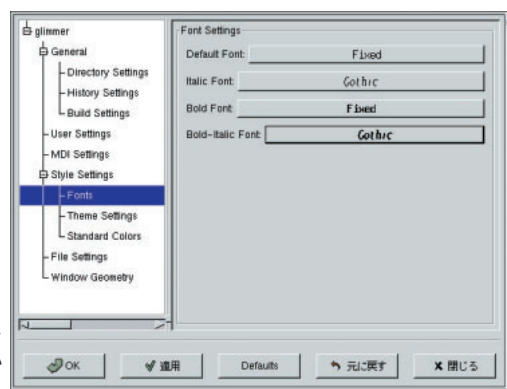
日本語EUCへの対応

パッチを当てたGlimmerを使用し、



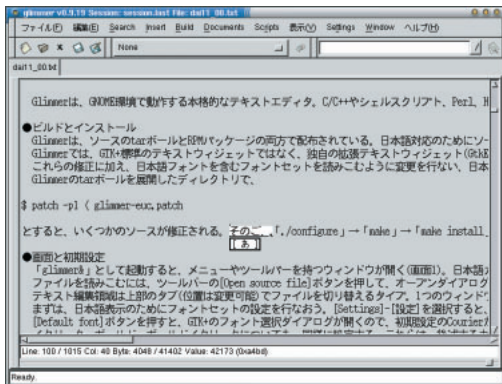
画面1

GlimmerはGNOME環境で動作する本格的なテキストエディタだ。



画面2

日本語を表示するにはフォントセットの変更が必要になる。



画面3
パッチを当てれば、日本語の表示や編集も可能だ。

日本語フォントを含むフォントセットを設定すれば、日本語EUCのテキストが正しく表示される(画面3)。

また、カーソルキーによる1文字単位のカーソル移動や、Ctrl - カーソルキーによるワード単位でのカーソル移動、Shift - カーソルキーによる範囲選択、BackSpace / Delキーによる文字の削除なども、日本語の文字を考慮した動作をするはずだ。

GlimmerはXIMに対応しているため、Cannaなどを使った日本語入力が可能だ。ただし、ウィンドウの端でテキストを折り返すことができないため、長い文章をバリバリ入力する用途には向いていない。あくまでプログラム作成が主な目的だ。

なお、今回のパッチはこの記事を書くに当たって筆者が急遽作成したもので、大規模な修正が必要な部分には対処していない。たとえば、正規表現ライブラリは日本語に未対応で、正規表現を利用した日本語の検索・置換は正常に動作しない。だが通常の検索



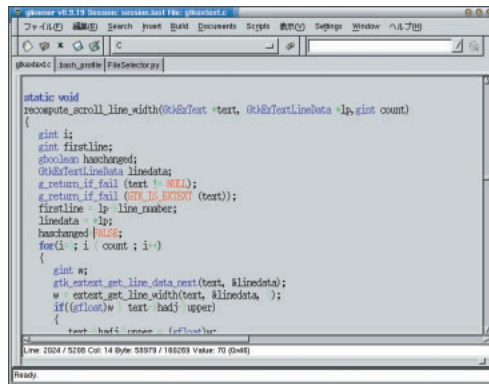
画面5
各部に対する色使いとフォント修飾をカスタマイズできる。

では日本語を検索可能だ。

カラー構文表示を利用する
Glimmerの大きな特徴は、色分けやフォント修飾により、プログラムソースの構造を理解しやすくするカラー構文表示にある(画面4)。現時点で、C / C++ / シェルスクリプト / HTML / Java / LaTeX / Perl / Python / SQL / XMLなど、計20種類ものプログラム言語やマークアップ言語に対応している。

言語の判別は、ファイルオープン時に拡張子から自動的に行われるが、ユーザーがツールバーのリストから選択することも可能だ。たとえば、編集時にカラー構文表示が必要ないなら、[None]に切り替えればよい。

また、メニューの[Settings] - [Highlight Colors]で開くダイアログで、文字列やキーワード、変数、演算子などの対象別に、使用する色とフォント



画面4
20種のプログラム、マークアップ言語に対応するカラー構文表示。

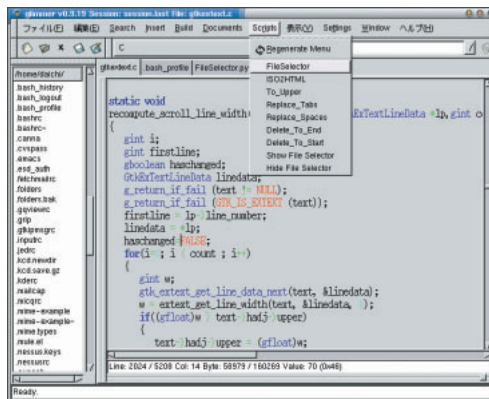
修飾を設定できる(画面5)。これらの設定は全言語で共通だ。

なお、イタリックやボールド、ボールドイタリックのフォントセットを用意できない場合は、ここで[I]や[B]などのボタンが押されていない状態に変更し、色の設定だけで区別できるようにすればよい。

マクロ言語にはPythonを採用

Glimmerでは、Pythonをマクロ言語として利用でき、ファイルのオープンを容易にする「FileSelector」(画面6)や、選択領域の英字を大文字に変換する「To_Upper」などのサンプルスクリプトが付属する。

これらのスクリプトは/usr/local/share/glimmer/scriptsに格納されており、メニューの[Scripts]以下の項目から選択する。新たなスクリプトを追加したら、[Scripts] - [Regenerate Menu]でメニューに登録しよう。



画面6
Pythonによるファイルセレクタ(左)とスクリプトメニュー。

GNOMEの高級科学電卓

rCalc

バージョン: 0.2.2

ライセンス: GPL

<http://rCalc.sourceforge.net/>

ビルドとインストール

rCalc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。

なお、ソースからのビルドの際は、「./configure --prefix=`gnome-config --prefix`」として、/usr以下へのインストールを指定する。/usr/local以下にインストールすると、ヘルプが表示されないの注意されたい。その後の手順は、「make」「make install」という一般的なものだ。

基本的な使い方

rCalcを起動すると、コマンドラインを含むウィンドウが開く(画面1)。使い方は簡単で、「rCalc>」というプロンプトに対して、計算式や代入式を書けばいい。

四則演算の演算子(+、-、*、/)は、ふだん私たちが使っている数式と

同じ優先順位(乗算・除算優先、左から右)で計算される。たとえば、「2+3/4」と入力してEnterキーを押すと、

```
rCalc> 2+3/4
      Ans = 2.75
```

と答(2.75)が表示される。優先順位を変えるには括弧を使えばいい。このほか、累乗を表わす演算子(^)も用意されており、優先順位は乗算や除算よりも上だ。

過去に入力した数式や計算結果は、100行(変更可)までさかのぼって参照できる。また、数式を入力する際は、bash風の履歴の呼び出しやコマンドラインの編集が可能だ。以前入力した数式を呼び出すにはCtrl-Pキーかキー、内容を編集するにはカーソルキーなどを使用する(画面2)。

変数と定数を活用する

rCalcでは、計算結果を保存するための変数を、宣言なしで自由に利用できる。変数名は英数字1文字以上の文字列(先頭は英字)で、大文字・小文字は区別されない。

たとえば、計算結果を変数hogeに格納するには、

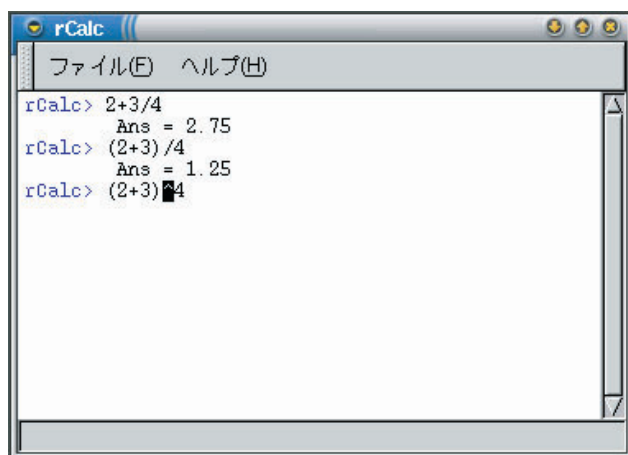
```
rCalc> hoge=(2+3)/4
      hoge = 1.25
```

とすればいい。以後、数式中でhogeを使うと、その値(この例では1.25)が計算に利用される。なお、計算結果を変数に格納しない場合は、特別な変数Ansに値が格納される。

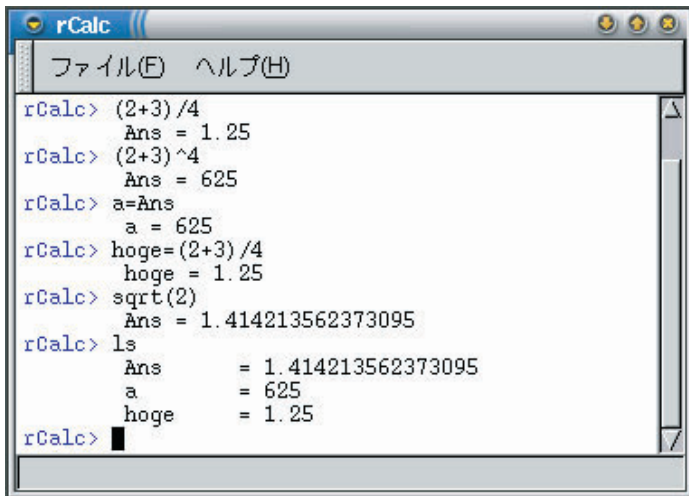
現在利用されている変数名と設定値を確認するには、組み込みコマンド(リスト1)のlsを使う(画面3)。なお、こうした変数とその値は、rCalc終了時に自動的に保存され、次回起動時に復元される。



画面1
コマンドラインに数式を記述するシンプルなインターフェイス。



画面2
bash風の履歴機能とコマンドライン編集機能で数式を修正できる。



画面3
組み込みコマンドのlsを使って、変数の一覧を表示。

変数を削除するには、組み込みコマンドのrmを利用して、

```
rCalc > rm hoge
```

などとすればいい。また、組み込みコマンドのclearを使うと、すべての変数を一括削除できる。

このほか、科学計算で頻繁に利用する円周率(π)と自然対数の底(e)は、最初から定数として定義されており、それぞれpiとeで参照できる。このため、変数名としてpiやeは使えないので注意されたい。

組み込み関数の利用

rCalcには、三角関数など基本的ないくつかの関数が最初から組み込まれている(リスト2)。これらは、

```
rCalc > sqrt(2)
Ans = 1.414213562373095
```

といった具合に、数式中で引数を指定して計算を行える。

なお、三角関数における角度の単位の既定値は度(deg)だ。これをラジアン(rad)に変更するには、組み込

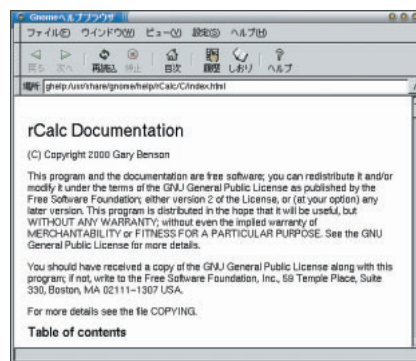
みコマンドのmodeを使って、

```
rCalc > mode rad
```

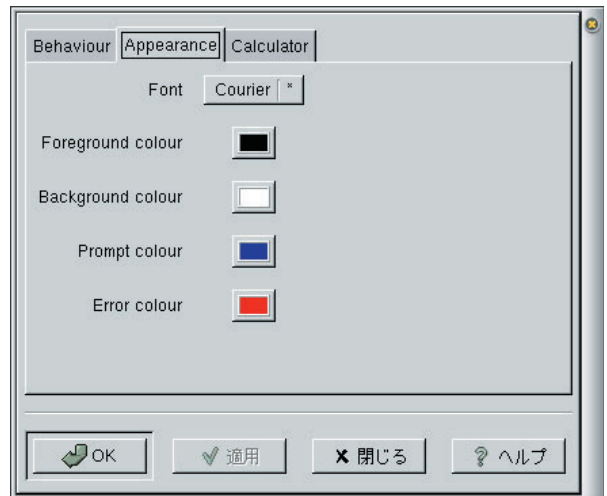
とするか、組み込みコマンドのradを使う。用途によって、適切な単位を選択すればいい。

また、[ファイル] - [設定]で表示される設定ダイアログ(画面4)の[Calculator]ページで、角度単位の既定値を変更することも可能だ。この設定ダイアログでは、ほかにも文字のフォントや色使い、過去の出力を参照できる行数などを設定できる。

なお、組み込み関数やコマンド、変数などの詳細については、GNOMEヘルプブラウザに表示されるHTMLヘル



画面5
GNOMEヘルプブラウザに表示されるrCalcのヘルプ。



画面4
設定ダイアログでは文字フォントや色使いなどを変更可能だ。

プ(画面5)を参照されたい。メニューの[ヘルプ] - [Help]や、組み込みコマンドのhelpで表示される。

exit, quit	rCalcを終了
help, man	ヘルプを表示
helpwin, ?	同上
ls, who	変数などの一覧表示
rm, clear	変数の削除
mode	角度単位(deg, rad)の設定
deg	角度単位を度(deg)に設定
rad	角度単位をラジアン(rad)に設定

リスト1 rCalcで利用できる組み込みコマンド

int(x)	xを四捨五入した整数
abs(x)	xの絶対値
log(x)	xの常用対数(10を底とする)
ln(x)	xの自然対数(eを底とする)
sqrt(x)	xの平方根
sin(x)	xの正弦(サイン)
cos(x)	xの余弦(コサイン)
tan(x)	xの正接(タンジェント)
asin(x)	xの逆正弦(アークサイン)
acos(x)	xの逆余弦(アークコサイン)
atan(x)	xの逆正接(アークタンジェント)
sinh(x)	xの双曲正弦(ハイパボリックサイン)
cosh(x)	xの双曲余弦(ハイパボリックコサイン)
tanh(x)	xの双曲正接(ハイパボリックタンジェント)
asinh(x)	xの双曲逆正弦
acosh(x)	xの双曲逆余弦
atanh(x)	xの双曲逆正接

リスト2 rCalcに用意された組み込み関数

完全フリーの音声圧縮技術を体感しよう

OggEnc

バージョン: 0.4

ライセンス: GPL

<http://www.vorbis.com/>
<http://www.xiph.org/ogg/vorbis/> (プロジェクトサイト)

ビルドとインストール

OggEncとXMMSプラグインは、いずれもバイナリ状態で配布されており、tarボール展開後に適当なディレクトリにコピーするだけでインストールが完了する。

OggEncの実行ファイル (oggenc) は、PATHに設定されたディレクトリ (/usr/local/binなど) XMMSプラグイン (libvorbis.so) は、XMMSのプラグイン格納先 (/usr/lib/xmms/Inputあるいは/usr/X11R6/lib/xmms/Input) にそれぞれコピーすればいい。

ソースからビルドする場合は、CVSを利用するか、プロジェクトサイトのダウンロードページからスナップショットのtarボールを入手する。これには、OggEncやXMMSプラグインのほか、各種のサンプルコードやWinAmpプラグインなどが含まれる。

エンコードを行う

コンソールアプリのOggEncの使い方は簡単で、「oggenc WAVEファイ

ル名」とするだけでいい (画面1)。複数のファイルを指定して、連続してエンコードすることも可能だ。

入力するWAVEファイルの形式は、「44.1kHz・16ビット・ステレオ」、つまり音楽CDから切り出した形式に限定されている。あらかじめ、cdparanoiaなどのCDリッパを利用して、WAVEファイルを切り出しておこう。出力されるOggファイルは、WAVEファイルの拡張子を除いたベース名に、拡張子「.ogg」を付けた名前になる。

曲名 (タイトル)、アルバム名、アーティスト名は、エンコード時のコマンドラインオプション (リスト1) でそれぞれ指定する。たとえば、

```
$ oggenc -t約束はிரない -lHotchpotch -a坂本真綾 track01.wav
```

とすると、曲名「約束はிரない」、アルバム名「Hotchpotch」、アーティスト名「坂本真綾」が埋め込まれた、約束はிரない.oggが作成される。タイトルを指定すると、出力ファイル名

Ogg Vorbisは、特許・特許料フリーで完全にオープンな仕様の音声圧縮フォーマットだ。可変ビットレート (VBR) を標準採用しているのが特徴だ。MP3に含まれる特許技術に基づく特許料の請求問題に伴い、一躍注目を集めるようになり、Ogg Vorbisに対応したプレーヤも増えつつある。今回は、オフィシャルサイトで配布されているエンコーダOggEncと、再生用のXMMSプラグインを紹介する。

のベース名としても利用されることに注意されたい。

なお、曲名などに日本語を埋め込む場合、再生に使うプレーヤも日本語に対応していなくてはならない。XMMSの場合、国際化パッチやjconvパッチの当たったバージョンが必要だ。

MP3エンコーダとの比較

Ogg Vorbisは可変ビットレート (VBR) が標準のため、ビットレートを指定する代わりに、「平均ビットレート」に対応するモード値 (1~6) を -mオプションで指定する。既定値は3 (平均ビットレート160kbps) だ。

そこで、モード2 (平均128kbps)、モード3 (平均160kbps)、モード4 (平均192kbps) を指定してOggEncでエンコードした結果を、代表的なMP3エンコーダの「LAME」および「午後のこ~だ」(gogo) と比較してみた (リスト2)。なお、LAMEとgogoは可変ビットレートのMP3をエンコードする機能も持っているため、固定・可変ビットレートのそれぞれの設定でエンコードした結果を示している。

圧縮後のサイズは、ビットレート128kbpsと160kbpsではそれほどの違いは見られないが、192kbpsではOggEncが他の結果より小さいファイルを生じている。Ogg Vorbisの可変ビットレート処理は、MP3のそれより優れているようだ。

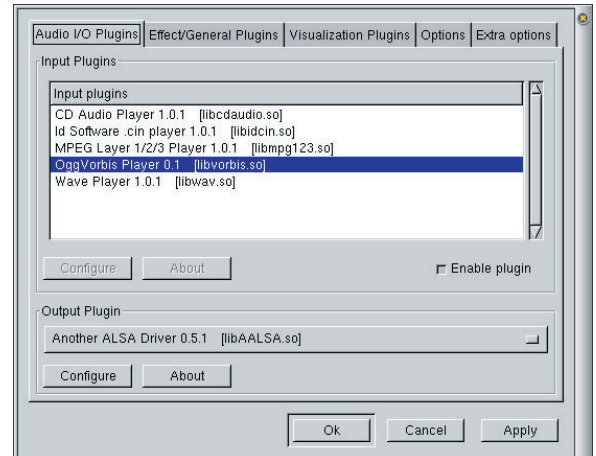
エンコード時間については、アセンブラを駆使してチューニングされた

-q	メッセージ出力を抑制
-h	ヘルプを表示
-r	Rawモード (PCMデータ直接入力)
-m	平均ビットレートのモード指定 (既定値3) 1:なし 2:128kbps 3:160kbps 4:192kbps 5:256kbps 6:350kbps
-o	出力ファイル名を指定 (単独ファイル時)
-n	曲名・アルバム名・アーティスト名を含んだ出力ファイル名を指定。
-c	コメントを付加 (複数指定可)
-t	曲名 (タイトル) を指定
-l	アルバム名を指定
-a	アーティスト名を指定

リスト1 OggEncのコマンドラインオプション一覧



画面1
OggEncを利用して、WAVEファイルをエンコード中。



画面2
「OggVorbis Player」プラグインが組み込まれていることを確認。

gogoの固定ビットレートには到底及ばないものの、LAMEの可変ビットレートとはいい勝負をしている。現在のOggEncのエンコードには、演奏時間の1割～2割増の時間がかかるようだ（Dual Celeron 450MHz PCの場合）。今後のチューニングに期待しよう。

XMMSで再生する

まずは、Ogg Vorbis再生用のXMMSプラグインが正しく組み込まれているか確認してみよう。XMMS起動後にCtrl - Pキーを押して、オプション設定ダイアログを開く。[Audio I/O Plugins]ページのリストに、「Ogg Vorbis Player 0.1 [libvorbis.so]」と表示されていればOKだ（画面2）。

あとは、MP3ファイルと同様の操作でファイルを読み込んだり、プレイリストに登録して再生を行えばいい。再生中のウィンドウを見ると、現在のビットレートが刻々と変化し、可変ビットレートであることを実感できる（画面3）。

音質については、エンコードする曲にもよるだろうが、筆者の場合は同程度のビットレートでエンコードしたMP3ファイルとの違いはほとんど感じられなかった。ぜひとも自分の耳で確かめていただきたい。



画面3
XMMSでOggファイルを再生中。ビットレートが刻々と変化する。

(1) 平均ビットレート128kbps前後		
	ファイルサイズ	エンコード時間
OggEnc	3487073バイト (9.2%)	4分02秒
LAME (可変)	3468773バイト (9.1%)	3分24秒
gogo (可変)	3460332バイト (9.1%)	1分29秒
LAME (固定)	3449417バイト (9.1%)	1分10秒
gogo (固定)	3448871バイト (8.8%)	53秒
(2) 平均ビットレート160kbps前後		
	ファイルサイズ	エンコード時間
OggEnc	4433161バイト (11.7%)	4分14秒
LAME (可変)	4319665バイト (11.4%)	3分34秒
gogo (可変)	4341234バイト (11.4%)	1分54秒
LAME (固定)	4311771バイト (11.3%)	57秒
gogo (固定)	4310812バイト (11.3%)	47秒
(3) 平均ビットレート192kbps前後		
	ファイルサイズ	エンコード時間
OggEnc	4692310バイト (12.3%)	4分13秒
LAME (可変)	4990718バイト (13.1%)	3分23秒
gogo (可変)	5175546バイト (13.6%)	2分01秒
LAME (固定)	5174125バイト (13.6%)	51秒
gogo (固定)	5173202バイト (13.6%)	36秒

リスト2 OggEncとMP3エンコーダLAME、gogoとの比較

2つのファイルの違いをカラー表示

gtkdiff

バージョン: 1.6.0

ライセンス: GPL

<http://www.ainet.or.jp/inoue/software/gtkdiff/>

ビルドと起動方法

gtkdiffはファイル一式をtar + gzipしたtarボールとRPMパッケージの両方が配布されている。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。ソースからのビルドも、「./configure」「make」「make install」という一般的な手順だ。

gtkdiffは、ktermなどのコマンドラインから、

```
$ gtkdiff ファイル1 ファイル2 &
```

と比較する2つのファイルを指定して起動する。

あるいは、ファイル名を指定せずに起動し、メニューの[ファイル] - [開く]で選択してもいい。この場合、オープンダイアログを使って、2つのファイルを順番に選択する。

相違点を色付き表示

gtkdiffのウィンドウは、初期設定では2つのファイルの内容が左右に並んで表示される2画面表示モードになって

いる(画面1)。

一目で違いを判別できるように、内容が異なる行は、左がオレンジ色、右が空色の背景で表示される。さらに、左側には相違点の位置がオレンジ色・空色のバーで、対応関係がバーを結ぶ線で示されている。

初期設定では、2つのファイルで内容が変わらない行が同期する(真横に並ぶ)ように、実際には含まれない空行が補間される(薄緑色の背景)。(ファイル表示) - [行の同期]のチェックを外すと、空行が消えて実際のファイルの内容だけが表示される。

画面が狭い環境では、[ファイル表示] - [画面表示] - [1画面表示]を選択して、1画面表示モードに切り替えといい。こちらは、2つのファイルの内容を1つにまとめ、相違点だけ背景色を変えて上下に並べて表示してくれる(画面2)。色使いは2画面表示の場合と同じだ。

こうした色使いは、[設定] - [設定]で表示される設定ダイアログで変更できる(画面3)。設定ダイアログでは、こ

gtkdiffは、2つのファイルの内容を並べて表示し、異なる部分をカラー表示するソフトだ。内容の一部が変更されることの多いプログラムのソースや設定ファイルなどに対して使うと、どの部分が変更されたのが素早く把握できる。2つのディレクトリを比較して、追加・変更されたファイルを一覧表示したり、2つのファイルを統合(マージ)する機能も用意されている。動作にはGTK+とGNOMEが必要だ。

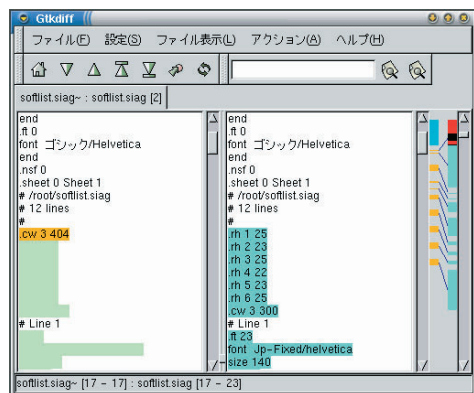
のほか、「英字の大・小文字を区別しない」などdiffに関するオプションも設定可能だ。

前後の相違点へ素早く移動

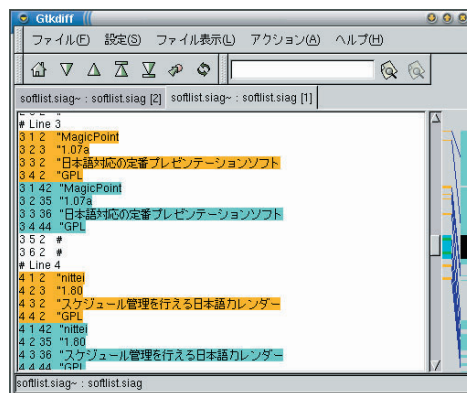
2画面表示モードでは、左端のスクロールバーを操作することで、2つのファイルの内容を同時にスクロールできる。マウスのホイールを利用したスクロールも可能だ。

また、前後の相違点に素早くジャンプする機能も用意されている。前方(ファイル末尾方向)の相違点にジャンプするには、Ctrl - Nキーを押すか、ツールバーのボタンをクリックすればいい。同様に、Ctrl - Pキーやボタンで後方(ファイル先頭方向)の相違点にジャンプする。最初や最後の相違点にジャンプすることも可能だ。

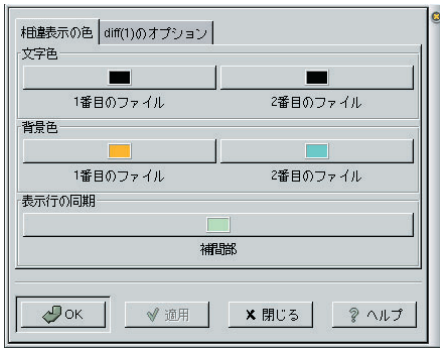
なお、[ファイル表示] - [行の折り返し]をチェックした状態では、これらの機能の動作が遅くなったり、動作しなくなったりするため、通常は行を折り返さない設定で使うことが推奨されている。



画面1
2つのファイルが左右に並んで表示される「2画面表示モード」。



画面2
両者の相違点が縦に並べて表示される「1画面表示モード」。



画面3
文字や背景の色使いはこの設定ダイアログで変更できる。

このほか、検索バーに入力した文字列をそれぞれのファイルから検索できる。ファイル1、ファイル2それぞれに対する検索ボタンが用意されており、マッチした文字列は青く反転表示される(画面4)。同じボタンを繰り返し押すことで再検索が可能だ。

2つのファイルを統合する

gtdiffには、2つのファイルの相違点を対話的な操作で統合して、新たに別のファイルを作成するマージ機能が用意されている。独自に修正を加えた2種類のプログラムソースを1つに統合したり、古い設定ファイルの設定の一部だけを再利用したりする場合に使用すると便利だ。

まずは、[ファイル表示] - [画面表示] - [マージ]を選択して、ウィンドウをマージモードに切り替える(画面5)。3分割された上部の領域にはマージ後のファイル、下部の2つの領域にはファイル1とファイル2の内容がそれぞれ表

示される。

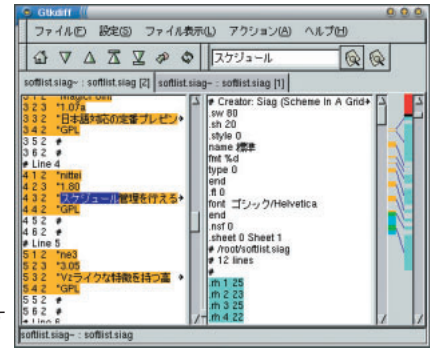
マージ後のファイルには、2つのファイルの相違点が灰色の背景色の空行で示されている。Ctrl - N / Pキーや / ボタンで前後の相違点にジャンプし、右クリックメニューを使ってファイル1とファイル2のどちらの内容を挿入するか選択しよう。

挿入した内容は、オレンジ色・空色の背景色で表示され、右クリックメニューを使って削除できる。また、[マージ]メニュー以下の項目により、いずれかのファイルの内容を一括挿入・削除することも可能だ。

ディレクトリの比較も可能

これまで2つのファイルの内容を比較してきたが、2つのディレクトリのファイル構成を比較することも可能だ。この機能を使うと、よく似た構成の2つのディレクトリの内容の違いを簡単に把握できる。

ディレクトリ表示モードに切り替え

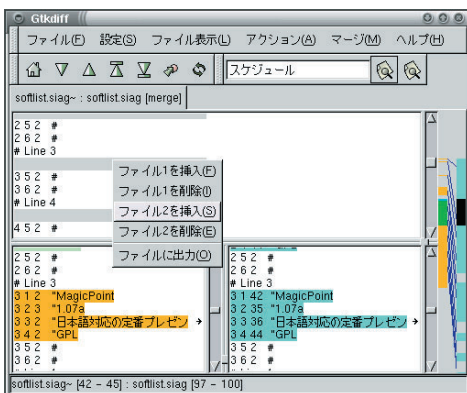


画面4
検索バーに入力した文字列を検索することも可能だ。

るには、起動時のコマンドラインで2つのディレクトリを指定するか、[ファイル] - [開く]のオープンダイアログで、[ディレクトリの比較]をチェックしてから、2つのディレクトリを順次選択する。

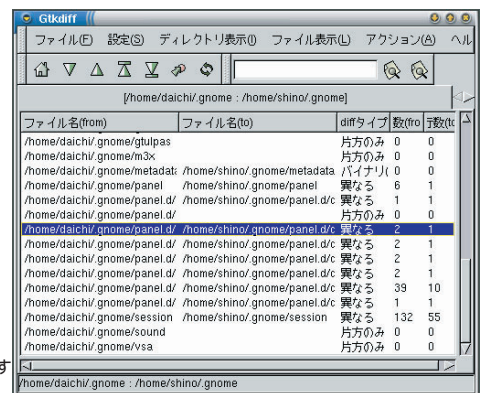
すると、指定したディレクトリ以下の全ファイルが検索され、片方のみ含まれるファイルや、内容の異なるファイルのリストが表示される(画面6)。[ディレクトリ表示]メニュー以下の項目により、バックアップファイルを隠したり、内容の異なるテキストファイルだけに絞り込むことも可能だ。プログラマーには有用な設定項目だ。

ディレクトリリスト中のファイルをクリックすると、それらのファイルが使用している画面表示モードで表示され、実際の相違点を確認できる。ディレクトリ表示モードに戻るには、ツールバー左端のボタンをクリックすればいい。



画面5
対話的な操作で相違点の統合(マージ)を行う「マージモード」

画面6
2つのディレクトリ構成の違いを示す「ディレクトリ表示モード」



正規表現のマッチを目で確認

RegExplorer

バージョン: 0.1.6

ライセンス: QPL

<http://regexplorer.sourceforge.net/>

RegExplorerは、指定した正規表現のパターンがマッチする部分を、色を変えて表示してくれるソフトだ。学習用のツールとして、またgrepやPerlなどで複雑な正規表現を実際に適用する前のテストツールとして最適だ。メニュー選択により細かな動作を変更できる。実行にはQtバージョン2ライブラリ(以下Qt 2)が必要だが、Qt 2がない環境でも動作するスタティックリンク版バイナリも用意されている。

ビルドとインストール

RegExplorerは、ソース一式をtar + gzipしたtarボールのほか、ダイナミックリンク版およびスタティックリンク版のバイナリもtarボールで配布されている。

ソースからのビルドとインストールは、「touch Makefile」「make」「make install」とする。あらかじめ、環境変数QTDIRに、Qt 2やインクルードファイルのあるディレクトリ(/usr/lib/qt-2.2.0など)を設定しておく。

一方、Qt 2のない環境では、Qt 2.2.0がスタティックリンクされたバイナリを利用するのが簡単だ。tarボールを展開後、実行ファイル

(regexplorer)を適当なディレクトリ(/usr/local/binなど)にコピーすればいい。

画面構成

「regexplorer&」として起動すると、いくつかの領域に分かれたウィンドウが開く。初期設定のままだとサイズが小さすぎるので、少し大きくしたほうがいいだろう(画面1)。なお、最初の1回に限り、自動マッチ機能(後述)が動作しないという不具合がある。これは、Ctrl - Aキーを2回押すことで回避可能だ。

ウィンドウ一番上の[Pattern]が、正規表現のパターンを記述する領域だ。正規表現ライブラリは独自のものだが、

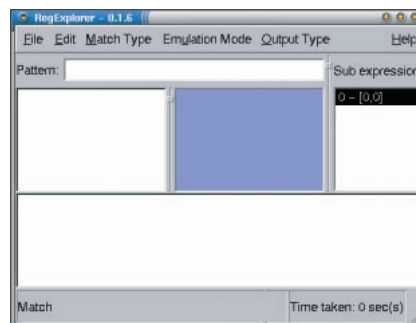
一般的な正規表現のメタ文字(リスト1)はすべて使える(日本語を含む文字列には対応していない)。

その下には、パターンマッチの対象となるテキストを記述する領域(左側)と、マッチした結果を表示する領域(中央)、マッチした行などの情報を表示する領域(右)が並んでいる。一番下は、Perlなど他のソフトで現在の正規表現パターンがどのように表現されるかを示す領域だ。

テキストとパターンの入力

簡単なパターンをマッチさせながら、RegExplorerの使い方を説明しよう。まず、左側の領域にテスト用のテキスト「This is a test.」を入力しておく。これに連動して、パターンマッチの結果を示す中央部の領域にも同じテキストが表示される。

続いて、上部の領域に「is a」というパターンを入力する。まず、「i」を入力した時点で、中央部のテキストの最初の「i」(3文字目)が黄色太字に変わり、「s」と「 」(スペース)を入



画面1

RegExplorerは正規表現の学習とテストに最適のソフトだ。

(1) 1文字にマッチ

.	(改行以外の)任意の1文字にマッチ
¥文字	メタ文字をリテラル(文字そのもの)として扱う。 (あるいは)通常文字の一部をメタ文字列にする
[...]	列挙した文字の1文字にマッチ
[^...]	列挙した文字を除く1文字にマッチ

(2) 繰り返し

+	1回以上無制限の繰り返し
*	0回以上無制限の繰り返し
?	0回または1回の繰り返し
{m,n}	m回以上n回以下の繰り返し(一方を省略可)
{m}	m回の繰り返し
(...)	文字列を繰り返しの対象とする

(3) 選択

	両側のパターンのいずれかにマッチ
(...)	選択範囲を変更

(4) 位置指定

^	行の先頭にマッチ
\$	行の末尾にマッチ
¥b	単語の境界にマッチ

(5) 前方参照

¥1,¥2,...	N番目に格納した文字列にマッチ
(...)	カッコ内パターンにマッチした文字列を格納

リスト1 RegExplorerで使える正規表現の代表的なメタ文字(列)

力すると3~5文字目の「is」が黄色太字になる。さらに、「a」を入力すると、今度は6~9文字目の「is a」が黄色太字になる(画面2)。

このように、入力したパターンにマッチする部分が即座に反応するので、正規表現の学習を行ったり、もっと複雑な正規表現をテストする際にとても便利なわけだ。

なお、テスト用のテキストの内容は自動的に保存され、起動時に復元される。また、メニューの[File] - [Load Text File]によって、ファイルから読み込むことも可能だ。

動作をカスタマイズする

英字の大小文字の区別など、パターンマッチに関するさまざまな動作は、[Match Type]メニューの項目(画面3)によりカスタマイズできる。

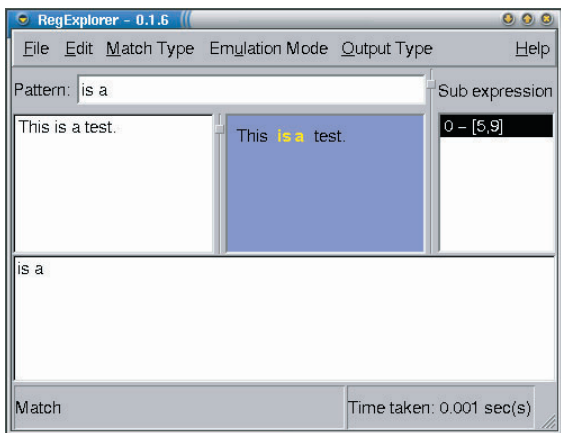
たとえば、初期設定ではパターンを入力している途中でもマッチが行われるが、複雑なパターンや長いテキストに対しては処理が重くなってしまう。こうした場合は、[Match Type] - [Automatic Matching]のチェックを外せばいい(Ctrl - Aキーでも可)。これで、パターン入力後にEnterキーを押した時点ではじめてマッチ処理が行われるようになる。

また、初期設定のノーマルモードでは、「最左最長マッチ」(最も左側から始まる最も長いパターンにマッチする)という正規表現の原則に従って、テキストの1カ所にしかマッチしない。[Match Type] - [Multiple]を選択して(Ctrl - Mキーでも可)、マルチプルモードに切り替えると、マッチする部分がすべて黄色太字で表示されるようになる(画面4)。

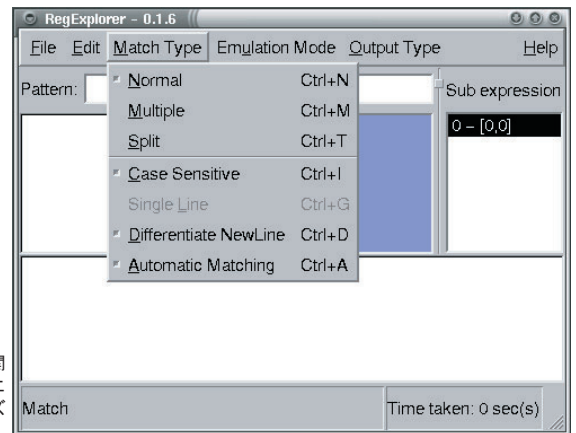
他のソフト用のパターンを生成

RegExplorerでは、Emacs LispやPerlなど、正規表現が使える他のソフトでのパターンや、検索コードの一部を生成できる。これにより、「メタ文字の前に¥が必要か」など、細かな部分の仕様の違いを気にすることなく、RegExplorerでテストしたパターンを他のソフトで利用できる。

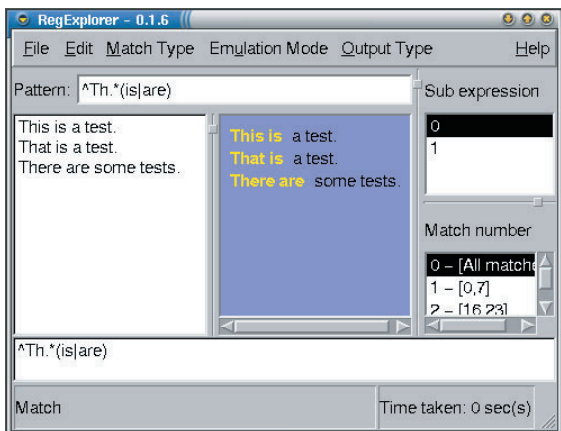
使い方は簡単で、[Output Type]以下の項目から、目的のソフトを選択すればいい。現在、Lisp (Emacs Lisp)、Perl、PHP、Qtを選択可能だ。選択したソフト用のパターンが下の領域に表示される。また、[Output Type] - [Coding output]がチェックされていると、現在のモードに対応した検索用コードが表示される(画面5)。



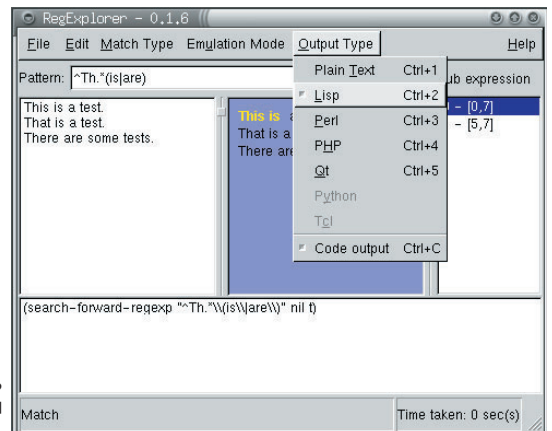
画面2
入力したパターンにマッチする部分が黄色太字で表示される。



画面3
パターンマッチに関する動作はこのメニューでカスタマイズ可能。



画面4
複数カ所のマッチを行うマルチプルモード。



画面5
Emacs Lispで使えるパターンと検索用コードを表示。

アステロイドを破壊する3Dゲーム

Asteroids3D

バージョン: 0.1.153 ライセンス: GPL

<http://www.pitt.edu/~stuart/a3d/>

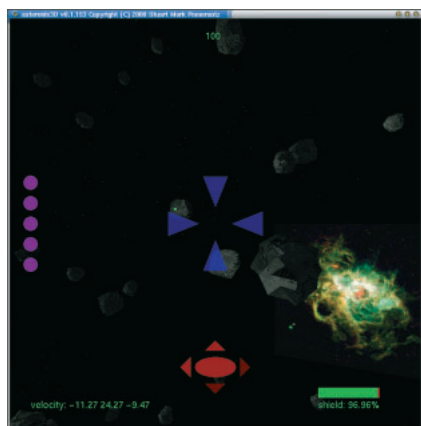
ビルドとインストール

Asteroids3Dは、ソース一式をtar + gzipしたtarボールのみ配布されている。configureスクリプトは含まれないので、必要に応じてMakefileを直接書きかえる。たとえば、Mesaライブラリが/usr以下にある場合は、Makefileの3行目に「-I/usr/include」、4行目に「-L/usr/lib」を追加する。

あとは、「make」とすればビルドが行われる。Makefileにはインストール用の設定が書かれていないので、実行ファイル(asteroids3D)を、適当なディレクトリ(/usr/local/gamesなど)に手でコピーしよう。

迫り来るアステロイドを破壊せよ

まずは、-numオプションでアステロイドの数を指定する。たとえば、5個で開始するには、「asteroids3D -num 5&」とすればいい。指定しなかった場合は100個で始まる。

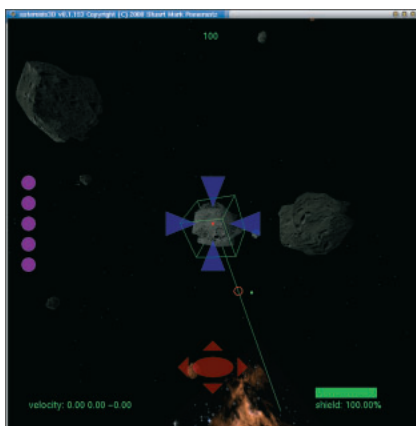


画面1
宇宙を漂うアステロイドを光子魚雷で破壊せよ。

ウィンドウには、宇宙を漂うアステロイドのほか、魚雷の照準や連続発射可能数を示すスロット、シールド残量を示すグラフ、速度計、上下左右および後方の危険を示すインジケータなどが表示されている(画面1)。

操作にはマウスとキーボードを併用する。マウスの移動で上下左右に回転し、左ボタンで魚雷発射だ。前後への加速はr/vキー、上下左右へのスライドはe/t/d/gキー、ブレーキにはfキーを使用する。テクスチャの切り替えなどのキー操作については、READMEを参照されたい。

自機とアステロイドが3D空間を移動しているため、やみくもに魚雷を発射しても命中させることは難しい。そこで、照準の中央にアステロイドを移動させてbキーを押そう。すると、そのアステロイドを囲むように緑色の立方体が表示され、黄(射程外)または赤(射程内)の円と点が表示される(画

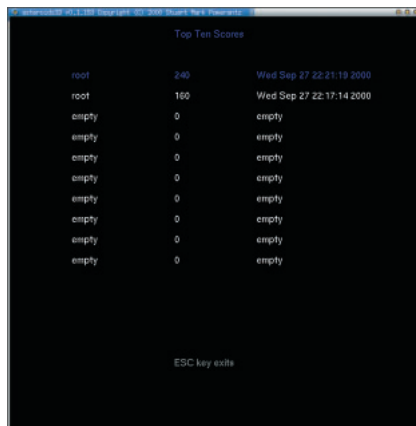


画面2
ロックオンしたアステロイドは立方体で囲まれる。

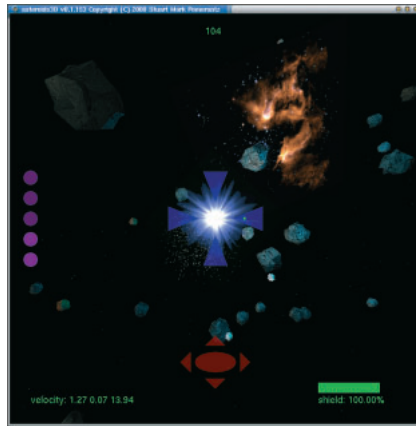
Asteroids3Dは、宇宙空間を漂うアステロイドを避けつつ、光子魚雷で破壊していく3Dゲームだ。アステロイドや背景には綺麗なテクスチャが貼られて迫力満点だ。3D空間を移動するアステロイドの破壊は難しいが、ロックオンと照準システムでサポートされる。動作には3DライブラリのMesaが必要で、テクスチャをオフにするといった工夫により、3Dアクセラレータカードのない環境でもプレイ可能だ。

画面2)。赤い円と点が重なるように移動しつつ魚雷を発射すれば、アステロイドに命中する(画面3)。

大きなアステロイドの場合は、破壊すると小さなアステロイドに分裂する。こうしてすべてのアステロイドを破壊していこう。破壊した数に応じてスコアが記録される(画面4)。



画面4
ハイスコア画面。この点数は、まだまだ甘い。



画面3
アステロイドに魚雷が命中すると、さらに細かな破片になる。

サムネイル付きの軽量画像ビューア

xzgv

バージョン : 0.5

ライセンス : GPL

<http://rus.members.beeb.net/xzgv.html>

xzgvは、サムネイル（縮小イメージ）付きの画像ビューア。JPEG / GIF / TIFF / PNGなど主要な画像形式に対応している。サムネイルの保存形式がxvやGIMPと同じなので、これらのソフトと共有できる。画像の縦横比を変えずに、ウィンドウに合わせて画像を拡大・縮小表示できるのが特徴だ。マウスやキー操作により、画像の切り替えや拡大・縮小、ファイル操作などを行える。動作にはGTK+とimlibが必要だ。

ビルドから起動まで

xzgvは、ファイル一式をtar + gzipしたtarボールのみ配布されている。configureスクリプトは付属していないが、通常はMakefileを書き換える必要はない。「make」「make install」とすると、ビルドとインストールが行われる。

「xzgv&」として起動すると、xzgvのウィンドウが開いて、xzgvのロゴが表示される（画面1）。表示したい画像のあるディレクトリや、画像ファイル自体をコマンドラインで指定することもできる。

サムネイル作成と画像の表示

ウィンドウ左側の「セクタ」には、カレントディレクトリのサブディレクトリと画像ファイルの一覧が表示されている。

最初は、それぞれの画像ファイルは同一のアイコンで表示されている。uキーを押すか、セクタ上で右クリックし、メニューから[Update Thumbnails]を選択すると、サムネイルが作成（更新）され、各画像のイメージを確認できるようになる。サブディレクトリも含めて再帰的にサムネイルを作成することも可能だ。

なお、サムネイル用の画像ファイル

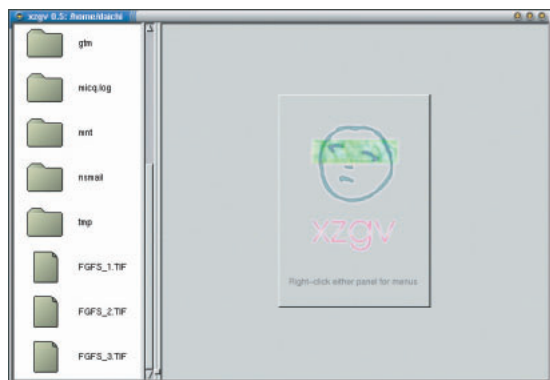
は、各ディレクトリの.xvpicsサブディレクトリに、画像ファイルと同名で作成される。これはxvやGIMPと同じ形式なので、いずれかのソフトでサムネイルを作成済みの場合は、最初からサムネイルが表示される。

セクタに表示されたサムネイルをクリックすると、その画像が右側の「ビューア」に表示される（画面2）。あとは、SPACEキーを押すか、ビューア上で左ボタンをクリックするだけで次の画像に切り替わる。前の画像に戻るにはbキーを押せばいい。

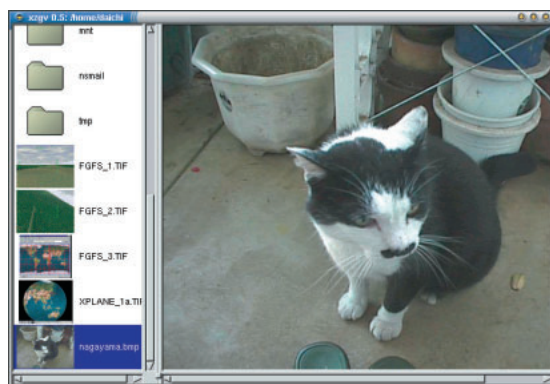
動作は軽快で、大量の画像を閲覧する際にもストレスを感じることなく利用することができる。

zキーを押すと、現在のビューアのサイズに合わせて、自動的に画像が拡大・縮小されるようになる（画面3）。画像の縦横比は変わらない。再度zキーを押すと元の状態に戻る。

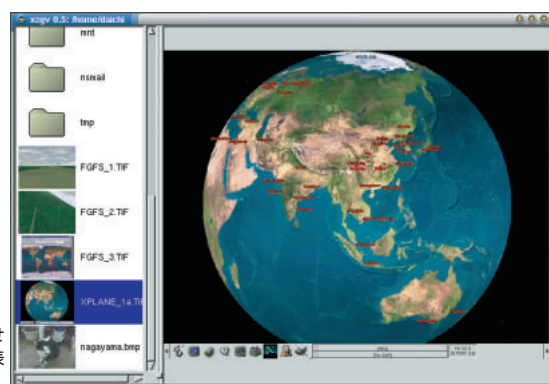
このほか、コピー・移動・削除などのファイル操作を行ったり、表示される画像の拡大・縮小・回転・反転などを行うことも可能だ。



画面1
起動時にはxzgvのロゴが表示される。サムネイルは未作成の状態。



画面2
サムネイルをクリックすると、その画像がビューアに表示される。



画面3
ビューアのサイズに合わせて画像を拡大・縮小して表示できる。

3D表示のネットワークトラフィックモニタ

WiredView

バージョン: 0.0.3

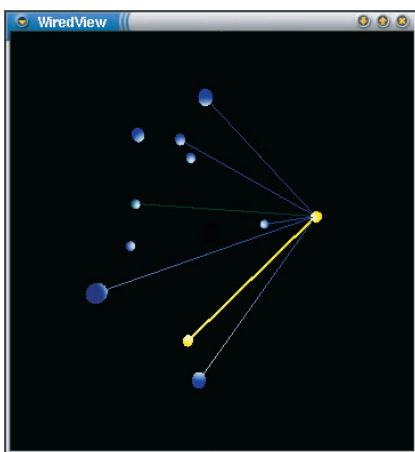
ライセンス: GPL

<http://www.redsails.org/files/>
[ftp://ftp.ee.lbl.gov/\(libpcap\)](ftp://ftp.ee.lbl.gov/(libpcap))
[http://www.student.oulu.fi/~jlof/gtkglarea/\(GtkGLArea\)](http://www.student.oulu.fi/~jlof/gtkglarea/(GtkGLArea))

ビルドとインストール

WiredView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あわせて選択しよう。必要なライブラリをあらかじめインストールしておくことも忘れずに。

なお、ソースにはconfigureスクリプトが含まれていないため、tarボールを



画面1
ネットワークトラフィックをノードやリンクの色や太さで表示。

利用するには、Makefileを書き換える必要がある。具体的には、10行目の「CFALGS = ...」の末尾に、「-l/usr/X11R6/include -l/usr/lib/glib/include」を追加する。RPMソースパッケージの場合も同様の修正を行う必要があるだろう。

回転するノードとリンク

「wiredview&」として起動すると、小さめのウィンドウが開いて、ノードとリンクが表示される。IPに対応したノードは球、ノード間のリンクは線で表現され、通信量に応じて球の色や線の太さが変化する。

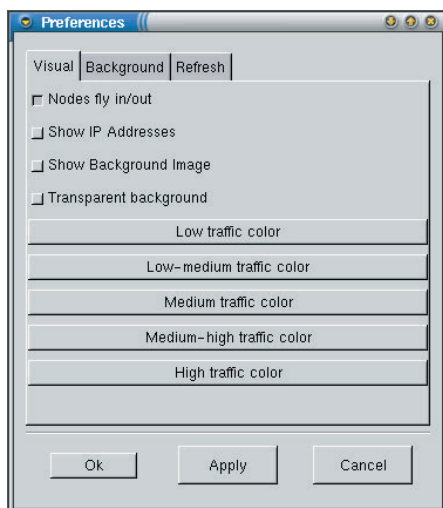
初期設定では、通信量の低いノードは自動的に消えるようになっている。ブラウザを起動してWebページを見たり、FTPサイトからファイルを受信したりすると、それぞれのサイトに対応したノードが離合集散し、刻々と色を変えながら全体が回転するのが見られ

るはずだ(画面1)。なお、回転はマウスの中ボタンクリックで停止でき、左ボタンのドラッグで回転方向や速度を変えられる。

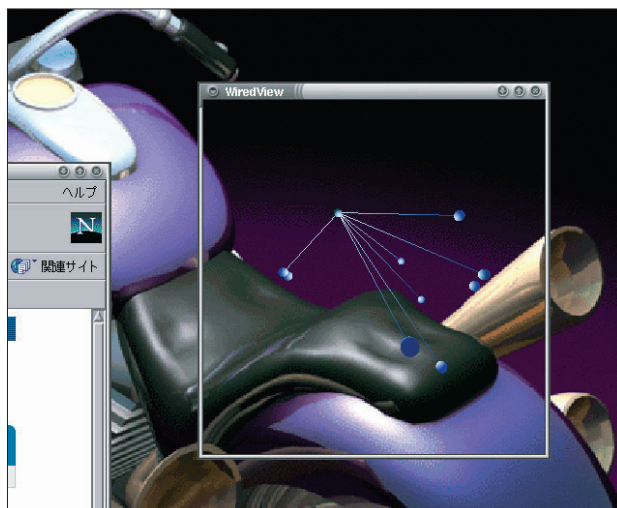
また、右ボタンクリックでメニューがポップアップし、[Preferences]で設定ダイアログが開く(画面2)。このダイアログでは、ノードやリンクの色(トラフィックに応じて5種類)の変更、ノードのIPアドレス表示のオン・オフ、各種情報の更新間隔の設定、背景の透過処理やビットマップの貼り付けなどが可能だ。

たとえば[Transparent background]をチェックすると、ルートウィンドウの画像がそのままWiredViewの背景として表示されるようになる(画面3)。ただし、こうした背景の透過やビットマップの使用は、CPUの負荷をかなり増大させるので、常用する場合はお勧めできない。

たとえば[Transparent background]をチェックすると、ルートウィンドウの画像がそのままWiredViewの背景として表示されるようになる(画面3)。ただし、こうした背景の透過やビットマップの使用は、CPUの負荷をかなり増大させるので、常用する場合はお勧めできない。



画面2
表示色などはこの設定ダイアログでカスタマイズ可能だ。



画面3
ビットマップを貼ったり、背景を透過させることもできる。

ルートウィンドウに美しいパターンを表示

Starfish

バージョン: 1.0b3

ライセンス: GPL

<http://www.redplanetsw.com/starfish/>

Starfishは、Xのルートウィンドウ(背景)に美しいパターンをタイル表示するソフトだ。パターンは数式と乱数に基づいて作られるので、起動するたびに異なった表示を楽しめる。また、上下左右が繋がるようになっており、タイルの継ぎ目が目立つことはない。タイルの大きさを指定できるほか、デーモンとして動作させて、一定時間ごとに自動的にパターンを変更する使い方も可能だ。

ビルドとインストール

Starfishは、ソース一式をtar + gzipしたtarボールのみ配布されている。configureスクリプトは用意されていないが、通常はMakefileを書きかえる必要はなく、「make」「make install」とするだけでビルドとインストールが行われる。

なお、starfishというよく似た(先頭が小文字)の別ソフトが存在する。このため、Starfishの実行ファイルは「xstarfish」という名前ですべてインストールされるので注意しよう。

表示する時間間隔も指定可能

使い方は簡単で、ktermなどのコマンドラインで、

```
$ xstarfish
```

とすればいい。しばらくすると、生成されたシームレスな(継ぎ目のない)

パターンがXのルートウィンドウにタイル表示され(画面1)、自動的にxstarfishが終了する。

また、一定時間ごとにパターンを繰り返し表示させるには、-dオプションに続けて、時間間隔(秒単位)を指定する。たとえば、

```
$ xstarfish -d 300
```

とすると、300秒ごとにパターンが更新される(画面2)。このとき、コマンドライン末尾に「&」を付けないことに注意されたい。また、「minutes」や「days」などを数値の後に書いて、分単位や日単位の時間間隔を指定することも可能だ。

このほか、表示するタイルの大きさを指定するオプションなどが用意されている。通常は指定する必要はないだろう。

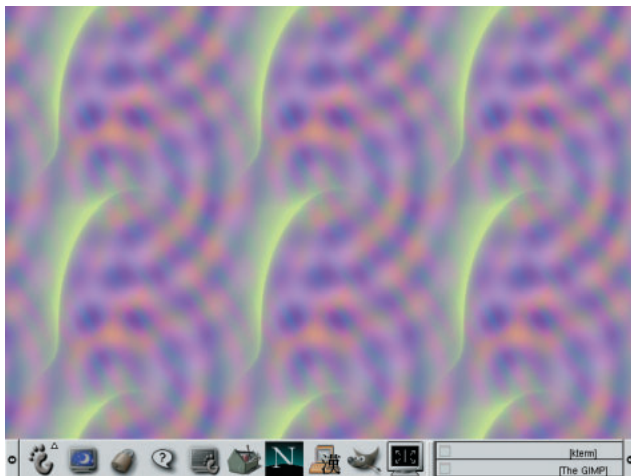
数式と乱数がパターンを決定

パターンの生成手順を簡単に説明しておこう。まず、それぞれ独自の数式に基づく「ジェネレータ」(Branch fractal、Bubble、Coswave、Flatwave、Galaxy、Range fractal、Spinflake)により、グレイスケールのテクスチャが作成される。

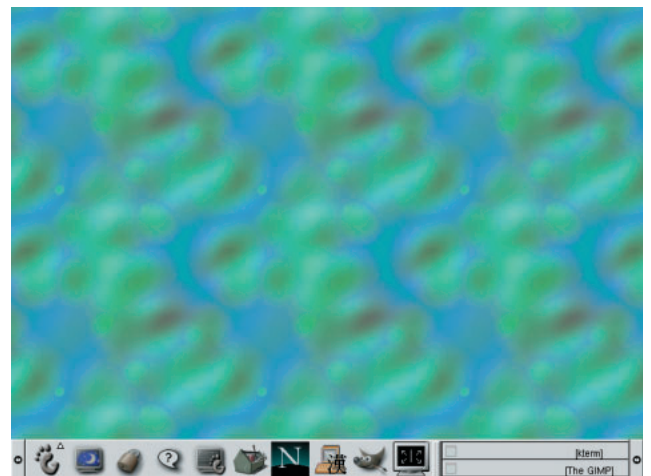
続いて、Starfishの「エンジン」がこれにカラフルなグラデーションを付ける。さらに、複数のテクスチャを重ね合わせる際に使われるアルファ(透明度)チャンネルにも、これらのテクスチャが使われる。

こうして、ある程度の数のテクスチャが重ね合わされて、最終的なパターンが作成される。

ジェネレータの選択や重ね合わせる数、色合いの選択などはランダムに決定されるので、毎回異なるパターンが得られるわけだ。



画面1
シームレスなパターンがルートウィンドウに表示される。



画面2
一定時間ごとに表示を更新させることも可能だ。

長い歴史と新しさを持つ

ATOK X & 一太郎Ark

長年愛されたかな漢字変換システムATOKのLinux版がリファインされた。また、Javaで動くワープロ一太郎ArkもLinux対応としてリパッケージされた。そこで、今回はATOK Xと一太郎Arkのレポートをお届けする。

文：塩田紳二
Text：Shinji Shioda

価格 9800円 / 9800円
問い合わせ先 ジャストシステム
03-5412-3939
<http://www.justsystem.co.jp/>

国内で販売されている多くのLinuxディストリビューションには、商用アプリケーションとして、ジャストシステムのATOK12SEが添付されている。これは、Windows用のATOK12をLinuxに移植した簡易版である。

ATOK12SEの発表時に、ジャストシステムはフル機能を持つATOKを自社パッケージとして出荷すると発表していた。そして、ようやく登場したのがATOK X for Linux(以下、ATOK X)である。このATOK Xは、単にATOK12SEに機能を追加しただけではなく、GTKを使ったGUIを実現するなど、別

バージョンといってもいい製品だ。

こうした違いがあるために、製品名もATOK+バージョン番号という、従来Windowsで踏襲してきた形式ではなく、ATOK Xという名前になっている。

今回、このATOK Xを、バージョンアップしてLinuxにも正式対応したJavaで記述されたワードプロセッサ、一太郎Arkとともにレポートしたい。

ATOK X

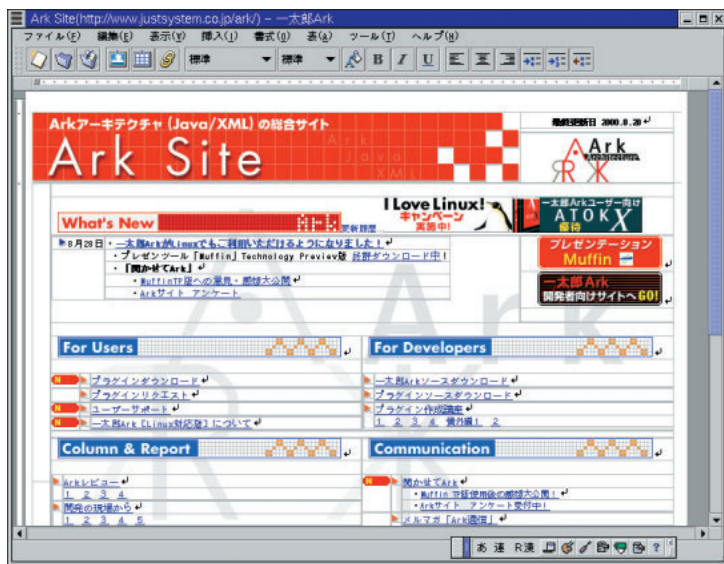
Windows用のかな漢字変換フロントエンドプロセッサ(以下、FEP。最近

ではInput Method=IMということが多いが)は、MS-DOS時代から開発され、長い歴史を持った製品が少なくない。その中でも、ジャストシステムのATOKシリーズは、NECのPC-9800シリーズのMS-DOS用ベストセラーワープロ「一太郎」のかな漢字変換エンジンとしてスタートし、単独のかな漢字変換FEPとなってからも多くのユーザーに使われてきた。一太郎ユーザーはもちろん、非一太郎ユーザーであっても、かな漢字変換FEPにはATOKを使うユーザーも少なくない。

余談だが、ATOKとは、Alpha TO Kanjiの略とも、Awa TOKushima(ジャストシステムの本社は徳島にある)の略ともいわれている。

ATOKシリーズの最大の特徴は、吟味された辞書と的確な変換候補の精度である。単に辞書登録の単語だけでなく、AI機能(変換文書中の直前の単語との関連から適切な変換候補を選択)などによる確かな変換にある。

Windowsのかな漢字変換FEPでは、カスタマイズや辞書機能などの付加機能といった部分での競争はすでに終わっており、ある意味、変換が好みに合うかどうかというレベルにまで達してい



画面1 ATOK Xと一太郎Ark
右下に見えるツールバーがATOK Xだ。表示しているのはジャストシステムのWebサイトである。

る。変換にはいろいろな方式が使われているが、ユーザーは、変換技術の原理よりも、実際の変換のフィーリングでかな漢字FEPを選んでいる。もっとも、最近では選択肢は事実上Windowsに付属するMicrosoft IMEとATOKの2つに絞られており、比較的新しいユーザーは、システムに標準で付属するIME 98/2000を、古くからのパソコンユーザーは、ATOKという形になっているようだ。逆にいうと、かな漢字変換FEPにある種のこだわりやフィーリングを要求するユーザーは、わざわざATOKを入手して使っているわけで、そのぶん厳しいユーザーと愛用者が多いともいえる。

ATOK Xは、そのATOKをベースにして開発され、従来のLinux用のかな漢字変換エンジンよりは厳しい環境で育ってきたものであるといえる。

ATOK12SEは、かな漢字変換エンジン部分は独自のプロトコルを使い、kinput2にこのATOK用のプロトコルへの対応を行ったkinput2xを使ってX Window System用のアプリケーションからの利用を可能にしていた。つまり、kinput2経由で日本語入力が行えた従来のアプリケーションのほとんどに対応できたわけである。

ATOK Xは、IIIMP(Internet-Intranet Input Method Protocol) というプロトコルに対応し、エンジン部分とアプリケーションと通信を行うATOK Xクライアントの2つのモジュールで構成されるようになった。

エンジンは、システム起動時に他のサーバなどと同じく/etc/rc.d/init.d内に置かれた「atokx」スクリプトで起動される。ここでは、変換エンジンが「/usr/lib/locale/ja/atokserver/atokmngdaemon」によって起動する。

次に、X Window Systemが起動するときに、「/etc/X11/xinit/xinitrc.d/xinput」などから、kinput2の代わりにATOK Xクライアントである「/usr/lib/locale/ja/atokserver/atokx_client」が起動される。これはシェルスクリプトになっており、実際には、「/usr/lib/im/htt_xbe」が動く。

ATOK Xを使ってみる

ATOK Xの標準の起動キーは、Ctrl + SPACEキーで、このほかShift + SPACEキーに変更することが

できる。これら以外のキーに割り当てることは残念ながらできない。

ATOK Xクライアントは、XIMを使ってクライアントと通信を行うため、たとえば、kterm内のviなどで入力を行うためには、ktermに「-xim」オプションを付けて起動する。

ATOK Xを起動すると、ATOK XのGUI部分である小さなツールバーが画面右下のほうに現れる(画面2)。この状態でかな漢字変換が可能で、たとえばローマ字などで入力していけば、一般のかな漢字変換と同じく入力が可能だ。変換候補表示は、縦に最大9つ候補が表示され(画面3)、拡大表示(画面4)や並べ替えなどが可能になっている。

付属ツールとしては、

文字パレット

環境設定

カスタマイズ

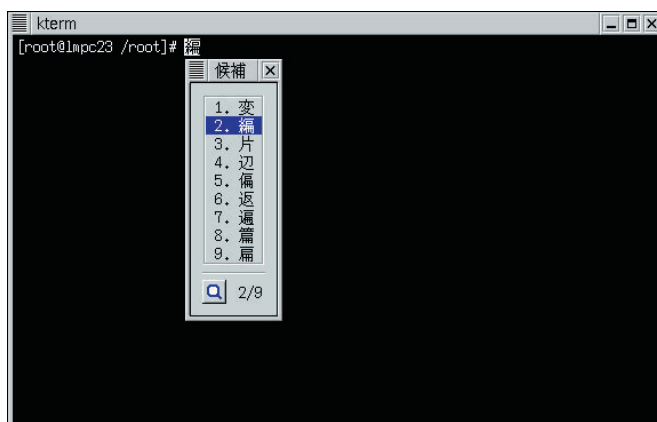
単語登録

辞書ユーティリティ



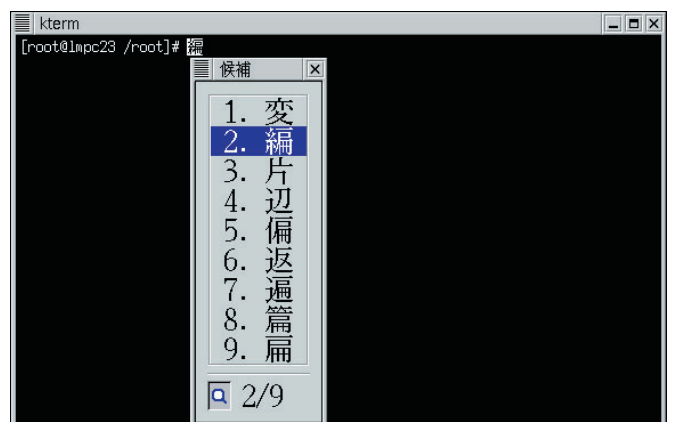
画面2 ATOK Xツールバー

Windows版のものとは若干形状異なる。横だけでなく、縦に配置することもできる。



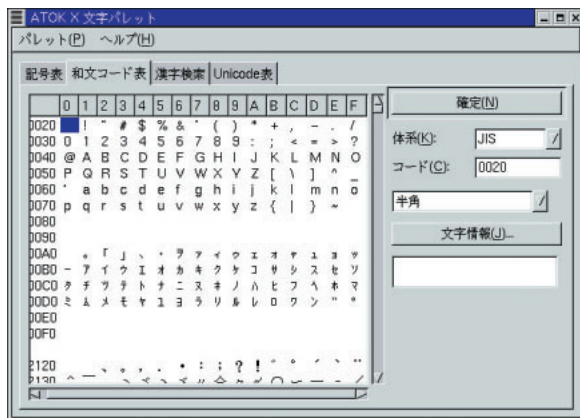
画面3 変換候補の表示

最大で9つまで表示される。左下の虫メガネアイコンをクリックすると拡大表示することができる。

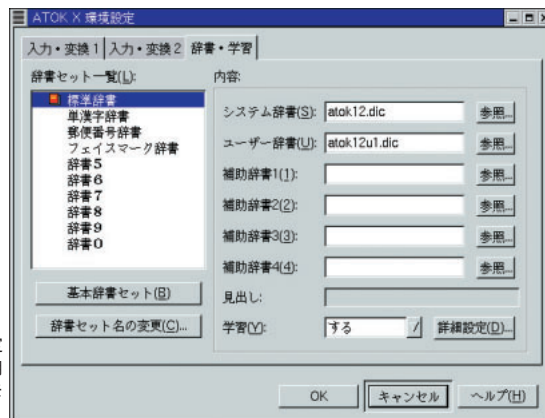


画面4 拡大表示

解像度を高くしていると変換候補が見にくくなるが、拡大表示すれば間違えない。なかなか細かい配慮だ。



画面5 文字パレット
Windowsでは当たり前前のこの機能もLinuxでは新しい。部首や画数から検索できるのは非常に便利だ。



画面6 環境設定
校正支援機能の設定や、かな漢字変換用の辞書の指定などを行う。

が用意されている。これらはATOK X ツールバーから起動可能になっている。

文字パレット

文字パレットは、読みの不明な文字や記号などを表から検索するためのもので、「記号表」、「和文コード表」、「Unicode表」に加え、部首や画数などから検索が可能な「漢字検索」機能がある(画面5)。

環境設定

環境設定は、標準の変換モードや入力文字種などを設定するためのもので、校正支援機能などの設定もここで行う(画面6)。ATOK Xの動作設定を行うためのダイアログボックスである。

カスタマイザ

カスタマイザは変換時のキーコマンドを設定するためのものだ(画面7)。キーバインドの設定は、「スタイル」と

呼ばれており、ATOK12互換のスタイルファイルが入っている。Windows版では、ここにVJEなど他のかな漢字変換FEPとキーバインドを同じにするスタイルファイルが提供されているが、ATOK XではCannaやWnnのスタイルファイルが用意されている。今までCannaやWnnを使ってきたユーザーもすぐに利用できるよになっているのが嬉しいところだ。

単語登録

単語登録はユーザーが辞書に単語を登録するためのダイアログボックスだ(画面8)。ここで、単語と読み、品詞などを指定して、登録を行う。

辞書ユーティリティ

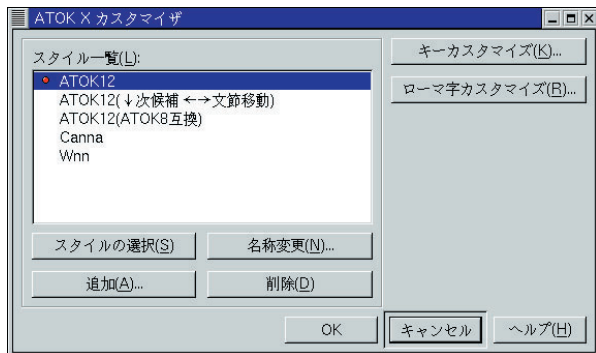
テキスト形式の単語ファイルを使って辞書に登録を行ったり、空の辞書を作成するなどの処理を行う(画面9)。他のFEPなどの辞書から単語と読みを

テキストファイルに書き出し、これを辞書ユーティリティで処理することで、他のシステムで使っていた登録単語をまとめて登録することができる。

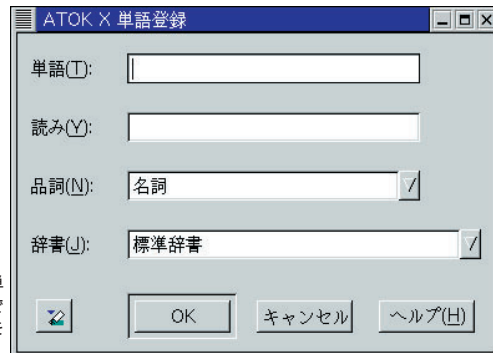
今回、多少遅めのPentium 133MHz程度のノートマシンで使ってみたが、実際の変換動作にはさほどもたつきを感じなかった。もっとも、辞書アクセスがあると、ちょっとしたタイムラグが発生するが、これも変換キーを押してから一息入れるような使い方ではあまり問題にならないだろう。

すでにWindows版でATOKシリーズを使っているのなら、使いこなれた辞書をもってきて使うことも可能である。このあたり、Windows版があるメリットだろう。

変換は、さすがに古くから鍛えられただけあって、ほとんどストレスを感じさせない。ATOKでは、文脈を見て同音異義語を判断しているため、初め



画面7 カスタマイザ
キーバインドが変更できる。CannaやWnnと同じキーバインドで使うことができる。もちろん、独自のキーバインドも可能だ。



画面8 単語登録
ここで単語の登録を行う。単語を選択して起動することでダイアログに自動的に単語を入力することができる。

て変換する単語でも、ある程度は正しい候補が出る。また、逆に、とんでもない変換結果（たとえば文節の切れ目をまったく間違えて変換してしまう）などは少ない。

たとえば、同音異義語である「選挙区」と「選曲」といった単語は、

「国会が解散した。国会議員は選挙に向けて動き出した。各議員はセンキョクに向かった」

「音楽会を開くことになった。私が責任者になり静かな音楽を選ぶことになった。今日、ようやくセンキョクが終わった」

といった2つの文章を入力する場合に、前者は「選挙区」が最初の候補となり、後者では「選曲」が第一候補となる。これは、直前に入力された単語を調査して、関連のあるほうを第一候補としているのである。なお、学習はこれに優先するため、強制的に変更することも可能だ。

同音類義語、たとえば、「変える」、「替える」、「代える」などの選択は、いわゆるAI変換によりかなりカバーできる。たとえば、

「かみそりの刃をカエる」

「車の色をカエる」

という場合、前者では「替える」が、後者では「変える」が第一候補として表示される。

前述したように、ATOKクラスのかな漢字変換では、文節の切れ目を間違えることは少ない。文節の切れ目に関しては、辞書の単語を多くするなどしてかなりカバーが可能だからだ。

次に問題となるのは同音異義語の選択である。ワープロで最も多いのは同音異義語（前述の「選曲」と「選挙区」、「選択」と「洗濯」など）や同音類義語（たとえば、「変える」「代えるなど」）の間違いだが、これも第一候補として適切なものを提示できれば少なくなるわけだ。

このあたりが、ストレスの少ないかな漢字変換の理由ともいえるだろう。

一太郎Ark

一太郎Arkは、100% Pure Javaで記述されたワードプロセッサである（画面10）。しかし、単なるワープロではなく、内部的にはXMLなどを使い、HTMLやXML出力が可能なテキストプロセッサでもある。このため、URLを指定してWebサーバ上のページを読

み込んで編集するといったことが可能となっている。

この一太郎Arkは昨年、バージョン1.0がWindows向けのパッケージとして販売されたが、LinuxでもJava Runtime Environmentなどをインストールすることで利用が可能である（ただし、Linuxでの動作は一太郎Arkパッケージの動作保証対象外ということになっていた）。

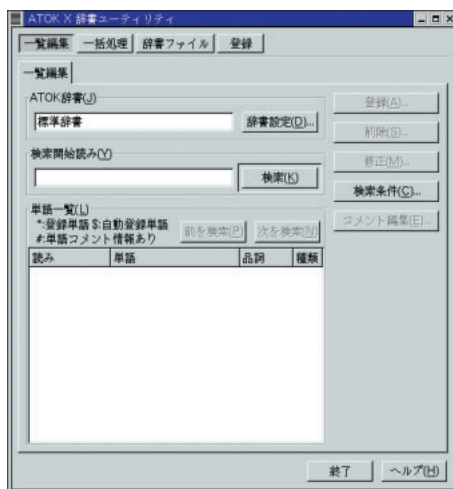
この一太郎Arkは、もちろんATOK Xと組合せての利用が可能である。日本国内で作られたワープロパッケージでもあり、同じ開発元なので安心できる組合せといえるだろう。

なお、ジャストシステムでは、この一太郎Arkに続き、プレゼンテーション作成ソフトMuffin（コード名）、表計算ソフトChoco（同）も開発予定で、現在Muffinのプレビュー版がダウンロード可能となっている（画面11）。

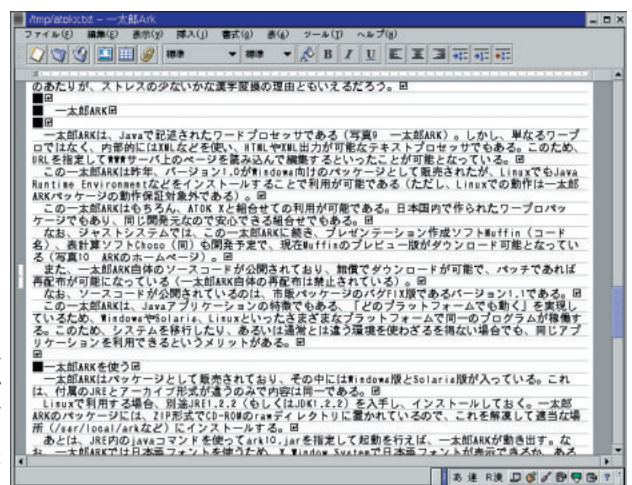
一太郎Arkのソースコードは公開されており、無償でダウンロードが可能で、パッチであれば再配布が可能になっている（一太郎Ark自体の再配布は禁止されている）。

なお、ソースコードが公開されているのは、市販パッケージのバグFIX版であるバージョン1.1である。

この一太郎Arkは、Javaアプリケー



画面9 辞書ユーティリティ
テキスト形式の辞書登録や、他のFEPの辞書をマージすることができる。登録されている単語を確認したり、削除したりすることもできる。



画面10 一太郎Arkの画面
見た目はWindowsのワープロとさほど違いはない。言わなければならないJavaで記述されていることに気が付かない。

ションの特徴でもある、「どのプラットフォームでも動く」を実現しているため、WindowsやSolaris、Linuxといったさまざまなプラットフォームで同一のプログラムが稼働する。このため、システムを移行したり、あるいは通常とは違う環境を使わざるを得ない場合でも、同じアプリケーションを利用できるというメリットがある。

Linuxに対応した一太郎Arkは9月22日に発売された。旧パッケージとの大きな違いは、Linux用のインストーラと、JREの追加である。旧パッケージでもLinuxで使用できなかったわけではないが、インストーラとJREが同梱されることで簡単にインストールできるようになった。また、Linuxに正式対応となった点も嬉しいところだ。なお、旧パッケージのユーザーであれば、ジャストシステムのホームページから無料でLinux対応版をダウンロードすることができる。

一太郎Arkを使う

あたりまえではあるが、一太郎Arkでは日本語フォントを使うため、X Window Systemで日本語フォントが

表示できるか、あるいはJava環境でTrueTypeフォントが利用できるようになっていなければならない。ただし、最近の多くのディストリビューションでは、このあたりの問題はないと思われる。

一太郎Arkでは、XMLとCSSを使ってドキュメントを作成する。最終出力は、XHTML、HTML3.2、HTML4.0（ZIP圧縮形式あり）、テキスト（gzip、ZIP圧縮形式もあり）が可能だ。また、基本的なページレイアウト機能は、HTML+CSSの範囲となる。ただし、表や画像の埋め込みのほか、フレームにも対応しているため、多くのWebサイトで使われているようなWebページの作成は可能になっている。

また、左右のマージンは、ページスタイルとして用紙サイズを決めておき、これに対して、マージンを設定するという通常のワープロと同様な形式となっている（写真12）など特殊なものではないが、実際の印刷はJava側の機能にまかされており、Javaで印刷できる環境を構築しておく必要がある。

Javaアプリケーションであるため、起動や処理などがバイナリプログラム

に比べるとちょっと遅いようだが、最近の高性能なPentium II/IIIのマシンでなら問題にはならないだろう。しかし、最低でもPentiumの166MHz程度は必要なのではないかと思われる。なお、Javaで動作する分、メモリも少々必要になるので、スワップが起こらないようにある程度のメモリ（48Mバイトではちょっと不足気味だろう。64Mバイト以上は必要だと思われる）を実装しているほうがいいようだ。一太郎Arkの動作中にスワップが起こると、CPUクロックに関わらず、ちょっと使いにくい。

全体的に日本語処理機能には問題はないのだが、禁則処理が「送り」のみであり、たとえば、行末に句読点などがきた場合、直前の文字が次の行に送られてしまう（写真13）。一般的にはこれが問題になることは少ないのだが、たとえば、20文字で2行以内などという文章を作る場合などにちょっとした問題となる（もっとも、こういうのは筆者のような仕事や編集者などが問題にするだけだろうが）。このあたり、一般のワープロは「ぶら下げ」禁則（行の最後に句読点などの記号を1文字だけはみ出させる禁則処理）を可能にしているの、なんとかしていただきたい点である。



画面 11 一太郎 Ark のホームページ (<http://www.justsystem.co.jp/ark/>)
一太郎 Ark のプラグインや、Muffin の Technology Preview 版がダウンロードできる。



画面 12 ページスタイルメニュー
ここで用紙サイズや左右マージンを設定する。ただし、実際の印刷はJava側の機能による。

Linuxの アプリケーション環境

ジャストシステムは、国産ベストセラーワープロ一太郎の開発元であるが、最近では、マイクロソフトのオフィスシリーズに押されぎみだ。その中で同社が打ち出したのが、LinuxやJavaへの対応である。少なくとも、マイクロソフトが本気でアプリケーションなどを展開することが考えにくい分野であると同時に、アンチマイクロソフト派からは、今後の成長を期待されている分野でもある。また、同社がWindowsなどで培った技術を転用することで、この分野の成長を加速することも不可能ではない。

ジャストシステムでは、昨年より国内の各ディストリビュータに対して、Linux版のATOK12SEを提供している。このため、多くのユーザーがATOK12SEを使うことができるようになった。ATOK12SEのSEとはStandard Editonの略で、いわばバンドル用の簡易版である。これに対してATOK Xは、LinuxやX Window Systemに対して、最初から対応することを目的として作られた製品バージョンである。

ジャストシステムのもうひとつの戦略がJavaである。同社は比較的早くからJavaに取り組み、Java用のワープロの開発を公表していたが、その結果として登場したのが一太郎Arkである。これは一太郎という名前を持つが、内部的にも、GUIなどについてもまったく別物である。Javaという性格上、数多くの機能を持つ一太郎そのものを移植することは難しいだろうし、一太郎だからといって必ずしも受け入れられるわけでもない。そこで、XMLやCSSといった最新技術を持ち込み、作られ

テイ) 罇

テキスト形式の単語ファイルを使って辞書に登録を行ったり、空の辞書を作成するなどの処理を行う。他のFEPなどの辞書からテキスト形式の単語と読みのファイルを作成し、これを辞書ユーティリティで処理することで、他のシステムで使っていた登録単語をまとめて登録することができる。罇

画面13 禁則処理の例

画面では区点などがぶら下げ禁則にならず、次行に送られている。このため、右側が揃っていない。

たのが一太郎Arkなのである。

このJavaへの取り組みは、ATOK Xが利用するIIIMPがJavaの入力メソッドのフレームワークからきているなど、相互に関連するものであり、ある意味、Linuxなどをベースにした標準を基準とした製品展開ともいえる。

また一太郎Arkについては、ソースコード公開により、ユーザーの力を借りて成長する方向をあえて選択している。当初より、プラグインの機構を公開して、ユーザーはJavaでプラグインを作ることができたのだが、これではワープロの補助機能部分は作ることができても、ワープロの本質的な機能を強化することはできない。また、プラグインから、編集集中のテキストを操作したいときなどは、一太郎Arkの内部API (Javaのクラス) をほとんど公開しないと、簡単なプラグインしか作ることができないということもあった。どうせ内部のAPIを公開するならば公開して、さまざまな具体的な(ユーザーのパッチなど)フィードバックを受けようという方法である。これは、バイナリのみを提供するWindowsなどのアプリケーションとはまったく違ったアプローチであり、SunやNetscapeなどが採用した方法でもある。

さて、英語環境向けには、いまや多くのオフィス用アプリケーションが登場しており、そのうちのいくつかは日本語対応がアナウンスされていたり、

実際に日本語対応しているものもある。しかし、かつてWindowsでも、英語版のワープロを日本語化した場合、さまざまな問題点が指摘され、長い時間をかけてようやく日本語ワープロとして定着した過去を考えると、これら英語版を日本語対応させたアプリケーションが十分な機能を持つと言われるまでには多少時間が必要な気もする。

また、ジャストシステムは、プレゼンテーション用ソフトや表計算ソフトもJavaアプリケーションとして開発中であり、これらがある程度の品質で登場すれば、国内でもLinux環境が一般利用に十分な能力を発揮できる可能性がある。米国などでは、Linuxを使ったインターネットアプライアンスや安価なパソコンがいくつか登場しているが、こうした製品も日本で利用できる、あるいは開発できるようになるといいだろう。

一太郎Arkのソース公開について

一太郎Arkのソースコードが公開されていると前述したが、これは一般的なオープンソースとは異なる。

あくまで開発・評価目的に限定されている。つまり、ソースコードの公開によって、無償で一太郎Arkが入手できるわけではないし、再配布することもできない。一太郎Arkを使用するには、パッケージを購入(使用ライセンス)する必要があるので注意しよう。

iモードにみる 電話会社の野蠻

文：豊福 剛
Text : Tsuyoshi Toyofuku

いまだ携帯電話を持っていない。それで生活に支障はないものの、持っててもいいかなと思わないでもない。しかしiモードとなると話は別だ。iモードには、なにかと不純なものを感じていて、それを支持する気持ちになれないのだ。

iモードの契約者数が、この8月に1000万人を突破した。これと前後して、iモードを使っただけで電話が問題になった。新聞を読んだだけでは、そのいたずらの手口はわからなかったが、iモードが使っているHTMLでは、URLに電話番号を指定できる。おそらく、``と書いてあったのだろう。

いたずら電話事件のほんとうの犯人は誰？

この事件には、iモードの抱える問題が集約されていると思う。

hrefにhttpが指定されているリンクを選択した場合、対応するページが表示されるだけですむが、mailtoのリンクは、いたずらに悪用される危険がある。本来送られてくるはずのないメールが大量に送り付けられてくるとしたら、それはメール爆弾と同じことだ。この危険性は通常のWebも同じように抱えている。だからといって、telのリンクをmailtoと同じように扱っていいものだろうか。ひっきりなしに間違い電話がかかってくるとしたら、その被害はメール爆弾による被害よりも、はるかに深刻だ。

mailtoのリンクは、メールの本文を入力する際に送信先のメールアドレスを確認することができる。しかしtelのリンクは、これからかけることになる電話番号が携帯電話の画面に表示されないために、そうとは知らずに110番にかけていたわけだ。

12月以降発売されるiモード対応の携帯電話では、リンク先の電話番号がダイアログ表示されるようになるらしい。もちろん、リンク先の電話番号が画面に表示されるとしても、それで問題が本質的に解決されるわけではない。110番であれば、すぐにいたずらとわかるだろうが、普通の電話番号だったら、判断しようがないわけだから。

それにしても、リンク先の電話番号が表示されないということは、市内なのか市外なのかもわからないし、知らずに国際電話をかけてしまうこともありえるわけだ。どうして最初から電話番号表示の機能が設計に組み込まれていなかったのだろう。iモード対応の携帯電話機の仕様が、NTTドコモと電話機のメーカーの間で、どのように取り決められているのか、よくわからない。NTTドコモ

の技術者が気づかなかったのか、メーカーの技術者がうっかりしたのか。まったく、携帯電話の技術に関しては、わからないことだらけだ。

NTTドコモは、iモード対応HTMLページにtelのリンクを記述するときは、着信先相手の了解を得るように、という注意喚起を実施するらしいが、実効性のある対応とは思えない。そもそも、URLにtelを追加することの意味を、もっと慎重に検討しておくべきではなかったのか。ボタンひとつで電話がかけられると便利だよ、くらいの安易な発想だったとしか思えない。電話をかけさせるのに便利な機能を、電話会社があえて盛り込まないはずがない。しかし、メールアドレスをWebで公開している人はいても、電話番号をWebで公開している人は少ない。それは、あきらかにメールアドレスと電話番号とでは、個人情報としての意味に大きな違いがあるからだ。

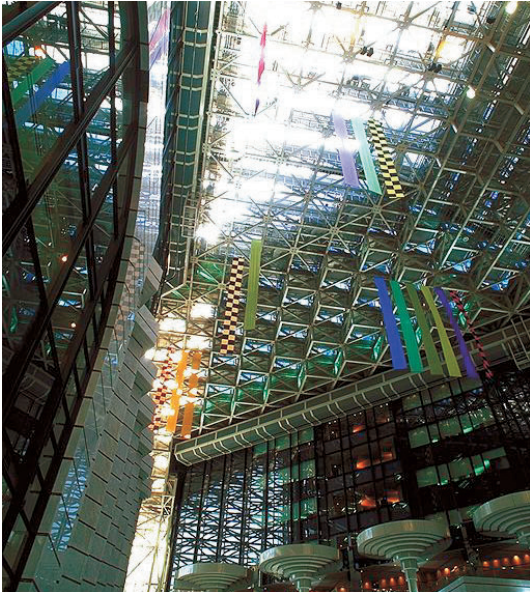
掟破りのiモードを礼賛する欺瞞

インターネットは、それが通信手段として電話回線を使うことはあっても、本質的にはコンピュータとコンピュータの通信である。そしてWebは、コンピュータでブラウザされることを前提に発展してきた技術である。そのようなWebの世界に、iモードは異質な要素を持ち込んできている。絵文字という機種依存文字や半角カタカナが使えるが、そのような機種依存文字はWebでは使わないのがインターネットのルールだったのではなかったか。

iモードが使っているCompact-HTMLは、HTMLのサブセットだったため、WebのHTMLを知っている人であれば、簡単に作ることができた。これに対して、au(DDI/IDO)のウェブサービスEZwebの記述言語であるWML/HDMLは、HTMLと異なり、カードとデッキという概念が導入されたXMLベースの言語になっている。J-PhoneのJ-SkyWebの記述言語はMML(Mobile Markup Language)で、HTMLに独自の変更を加えたもののようだが、正確な仕様は公開されていない。使用可能な画像についても各社異なり、iモードがGIFを使うのに対して、EZwebはBMP、J-SkyはPNGを使う。

iモード以外の2社がGIFを使っていないのは、GIFの圧縮アルゴリズムに対する特許が絡んでいるのだろうか。それはともかくとして、記述言語に対するアプローチとしては、技術的観点から判断した場合、EZwebが採用しているWMLが正解であると思う。しかしながら、技術的な正解がかならずしも市場でも成功するとはかぎらない。残念





ながら、そのような歴史は携帯電話においても繰り返されてしまいつつあるようだ。

EZwebもJ-SkyWebも、それぞれコンテンツ作成用のエディタをWebで公開してはいるものの、Compact-HTMLによるコンテンツ作成の手軽さと比べれば、どうしても作りにくさは否めない。Webサーバの既存のMIME設定のままでは、これらのコンテンツがそれぞれの携帯電話機で正しく扱えない場合があることも、普及をいまひとつ妨げる要因になっているのだろう。

携帯電話機とPDAの境界は今後ますます曖昧になっていくとはいえ、やはり厳然としてその区別は残ると思われる。携帯電話機は、片手で握って、耳にあて、マイクで声が拾える必要があり、そこから本体の大きさが決まる。フォン部とマイク部の間に配置される画面とボタンの大きさもそこから割り出されるのであれば、画面の解像度に限界があるのは明らかだ。この極端に小さな画面サイズをまず前提にしなければならない。

画面が小さく、受信して記憶できるデータ量にも制限がある以上、そこで扱われる情報は、あらかじめコンテキストがかなりの部分決まっっていて、極端に短いメッセージであっても意味が伝わるものに限定されるだろう。つまり、携帯電話の電子メールは、郵便的ではなく、電報的であるということだ。Webについても同様である。それはページではなく、メニューでしかない。

ポケベルに毛がはえた程度のサービスを、インターネットと同列に語るのは欺瞞的だ。携帯電話機で電子メールの送受信もできるし、Webのごくごく一部がブラウズできるとしても、それはあくまでインターネットの周縁に繁殖するローカルでちっぽけな孤島でしかない。たかだか250文字を送受信できるだけのメール、たかだか数Kバイトのページを表示できるだけのブラウザ。その程度のサービスが、なぜこれほどまでに過大に評価されているのか、理解に苦しむ。

携帯電話による電子メール利用は、パソコンに取って代わるものではありえない。パソコンは難しすぎて使えないと考えている中高年のおやじは、携帯電話だったら自分でも使えると思っているのだろうか。もしそうだとしたら、そのようなおやじこそ、まっさきにIT革命によって淘汰されるだろう。

『iモード事件』の真相

松永真理『iモード事件』(角川書店)を読むと、iモー

ドにおいてコンテンツがどういうものとして考えられているのかが、よくわかる。

iモードの基本コンセプトは、携帯電話でメールを使いたい、というところにある。「携帯を使う人同士のコミュニケーションが最大のコンテンツ」と考えられていたらしい。もしiモードが、「携帯を使う人同士」のメールのやり取りとして閉じられていたら、これほどiモードを不快に思うこともなかっただろう。しかし、不幸にして、iモードはインターネットとのゲートウェイ接続サービスとして実現されてしまった。

iモードは、電話とインターネットの野合である。「相手の電話番号がわからない限りなにもできない」という「音声コミュニケーションツールとしての携帯電話の難点」を乗り越えるために、インターネットが利用されているのだ。そこでは、インターネットは目的としてではなく手段として扱われている。

「携帯電話の場合、ユーザーはある情報を得るために電話を利用しているわけではない。電話を使うことで自然に、そこに載っている企業のサービスを見ることになる」。そのようなサービスとは、銀行口座の残高照会、チケット予約、ホテル予約（取引系）だったり、天気予報、株価情報、タウン情報（生活情報系）だったり、レストラン・ガイド、乗り換え案内（データベース系）だったり、ゲーム、占い、カラオケ情報（エンタテインメント系）である。そしてこの分類は「4つのフェーズ」と呼ばれる。その情報観・メディア観は、びあやリクルートを連想させる。「企業にとっての広告が、ユーザーにとっては不必要なものではなく必要な情報になるので、逆に料金を払って買い求めてくれる」という発想だ。そこでは、ユーザーは消費者として位置づけられている。コンテンツとは、消費のための情報なのだ。

iモード用語では、コンテンツのことを「番組」と読んでいるらしい。コンテンツを「番組」と呼ぶ発想は、テレビのチャンネルを切り替えるような操作感を連想させる。しかし、テレビのようなマスメディアでは、コンテンツを発信する側に立てるのは、少数の限られた者だけである。皮肉なことに、iモードの「番組」は、公認IP（情報プロバイダ）が提供する公式サイトだけに限定されなかった。Compact-HTMLを採用したがために、インターネット上にiモード対応のページが続々と増殖するに至ってしまった。このことの良し悪しは別にして、それはもう現実としてそうってしまった。大人だけのレストランに、子供たちが大挙して押し寄せ、お子様ランチを注文しているのだ。

Profile

とよふく つよし

1962年生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

オープンソースのデスクトップ環境は GNOMEで決まるのだろうか

文：安田幸弘
Text: Yukihiro Yasuda

GNOMEファウンデーション

GNOMEファウンデーション、なんだそうだ。

すでにご存じの通り、8月のLinuxWorldで、Sun、HP、IBM、CompaqといったUNIXの大手ベンダーを含む各社がGNOMEファウンデーションを結成した。GNOMEを核にLinuxの標準的なデスクトップ環境を構築、Windows / Officeを打倒しようという話らしい。

悪い話じゃない。大手ベンダーが足並みを揃えて打倒Windowsに立ち上がった、という意味で、これからが面白くなりそうな気がする。Linusの世界征服の野望が実現にまた一歩近づいたわけだ。でもこの話を聞いた瞬間、ぼくは何だか妙に複雑な気分になった。

なぜ複雑な気分になるんだろう

複雑な気分その1は、なんとって'80年代のOpenLook対OSFの不毛なバトルの記憶である。SunにしるHPにしるIBMにしる、大手のワークステーションメーカーが'80年代後半に「唯一のオープン」をめぐる、とってくだらない競争をしていたことは、まだ記憶に新しい。

あの競争のくだらなさ、要するにユーザーも技術も無視してひたすら肩書きをめぐる戦われたことにあった。

もっとも、当時のUNIXのユーザーなんて、1台何百万円というワークステーションを何十台もまとめて買うような組織がほとんどだったわけで、今のLinuxユーザーとは質も量も違うけれど、それでもユーザー無視のツケは結局、融通のきかない各社各様のオープンシステムとやらの蝸壺システムを生んだだけだった。

で、GNOMEファウンデーションなんだけど、GNOMEは悪くないし、各社が足並みを揃えて打倒Windowsを目指すってのは頼もしい、でも、

ぼくは当のオープンソースソフトウェアのユーザーの立場を忘れてやしませんか、と言いたいのだ。

確かにGNOMEは優秀なデスクトップ環境ではあるけれど、オープンソースソフトウェアの世界には、KDEというもうひとつの優秀なデスクトップ環境がある。そして、世界的に見れば、KDEのユーザーはあるいはGNOMEのユーザーよりも多いかもしれない。

ちなみに、最近、大手のハードウェアメーカーはこぞってLinuxをサポートするようになったが、これに疑問を感じる人は少ないだろう。FreeBSDではなくてLinuxなのは、圧倒的にLinuxのユーザーが多いからだ。つまりユーザーの支持がある。だからLinuxがスタンダードになろうとしていることに違和感はない。

Apacheにしても、圧倒的なユーザーを持つApacheだからこそ、Apacheファウンデーションを結成してゴリゴリと攻めることに誰も文句を言わないわけだ。

ところがデスクトップ環境の世界ではKDEもGNOMEも、どちらもユーザーの数に大差はない。また、技術的な面でも、どちらがズバ抜けて優秀というものでもない。全体的には互角と言っていると思う。それなのに、オープンソースのコミュニティにも大きな影響を与えかねないGNOMEファウンデーションなんてものが突然降ってくるから「これは何なんだ？」という気になるわけだ。

あるいは、KDEで使われているツールキットQtのライセンスの一部がフリーではなかった、という点が問題だったのかなとも思うけど、それも現在では解決されている。技術的な面でも、GNOMEがいいとか、いやKDEだとかという議論はあるが、使う側の末端ユーザーから見ればどっちでも同じようなものだ。オブジェクトモデルがどうこうという以前に、GNOMEもKDEも、もっとGUIを洗練させる余地はあると思うんだけどねえ。

それはともかく、先日、KDEの開発グループ

が、GNOMEファウンデーションに対抗してKDEリーグとやらを結成するのしないのというわざを見かけた。おかげでぼくの複雑な気分は、ますます複雑になってしまう。

これまで、ほとんどのユーザーは、どちらが優勢になるにしても、いずれLinuxと*BSDのように、どちらかが「ユーザーの支持」を得て自然に落ち着くだろうと思っていたはずだ。そこへ突然、一方への大企業の肩入れで流れが急変してしまうのは何とも不自然。ここで、KDEが「対抗しやむを得ず」、別の企業を集めてユーザーの分捕り合戦や、へたをしたら潰し合いのようなことが始まってしまったら……。

あのときの「オープンシステム」のタイトルをめぐるユーザー不在の抗争みたいなものにつきあいたいと思うユーザーなんて、どこにもいないだろう。

オープンソース開発グループと企業、そしてユーザー

どんな平和も永遠には続かない。まして動きの速いデジタルの世界では、技術的な変化や環境の変化で、すでに数え切れないほどのコミュニティが形成されては消滅していった。これまで、比較的初期の志向を維持しながら平和に拡大してきたオープンソースコミュニティも、そろそろ新しい局面に突入しようとしているのかもしれない。

打倒MSの悲願をオープンソースにかけるワークステーションメーカーは、慎重ながらもかなり戦略的にオープンソースの世界に足を踏み入れてきた。今回のコンソーシアム結成も、決して唐突だったわけではなく、前からGNOMEに目をつけていた各社が「あのときの二の舞はやめよう」と大同団結した結果だとも考えられる。

大企業が本気になって参入してくれば、オープンソースの普及は劇的に加速するだろう。その一方で、必然的にトップダウンにならざるを得ない大企業の製品戦略と、ボトムアップを身上とする

オープンソースの開発プロセスは、どこかで衝突する可能性がある。何も言わないユーザーが、黙ってGNOMEを選ぶ、またはKDEを選ぶという選択だって、ボトムアップの開発プロセスの重要なファクターだ。今回のGNOMEファウンデーションの結成は、何年もデスクトップ環境の熟成を待ってられない企業側の事情が先行した結果だったのかもしれない。

もちろん、単に企業とユーザーといった二項対立的な図式では、現在のオープンソースは成立しない。オープンソースの世界では、開発グループとユーザーの関係を基本に、企業がバランスの取れた関係を維持していくことがオープンソースを持続的に発展させていくポイントだろう。

その意味で、GNOMEファウンデーションは今後、きちんとユーザーの支持を集め、他の開発グループと十分な情報交換を行ってほしいものだと思う。

オープンソースは、ただ結果としてできあがったソースだけをオープンにすればいいってもんじゃないと、少なくともぼくは思っている。Linuxが成功したのも、開発のプロセスそのものが限りなく透明だったことが大きな理由だ。そのことによって、いくつかの難しい問題も切り抜けることができた。

GNOMEファウンデーションにしても、ここに参加する企業は単にGNOMEという特定のプロダクトばかりでなく、GNOMEファウンデーションを通じてオープンソースコミュニティ全体に貢献することを明確に示してほしいと思う。そうすることが、本当に「打倒Windows」につながる道なんじゃないだろうか。

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

- 9・6 RSAが暗号特許を公開
- 8・29 マイクロンがRambus特許の無効を訴える
- 8・28 インテル1.13GHz PentiumIIIをリコール
- 8・23 インテルXscaleプロセッサを発表
- 8・11 AMD Sledgehammerの詳細を公開

このところ、Linuxのサーバが調子悪くて、時々落ちてしまう。あり合わせの部品で組み立てたのが悪かったのでしょうか。いきなり落ちるんで、原因究明も難しいんだよねえ。

インテルは今月も大変な感じ

インテルは、いったん出荷した1.13GHzのPentiumIIIを回収した。どうも、予定よりも大量に熱を発生してシステムが落ちる可能性があるらしい。このところインテルにはバッドニュースが多く、端から見ていると、なんだかちょっと「ピンチ」のようにも思える。というのも、AMDのAthlonがかなり「調子いい」からである。高クロックが可能なAthlonは、大手メーカーの採用が続いており、イメージ的にもAMDのプロセッサを使ったパソコンは一般ユーザーにとっても「普通のパソコン

」という感じになってきている。かつては、AMDのプロセッサを使ったマシンを「買うべきか?」という感じだったのだが、現在は、「インテル、AMDどちらのプロセッサを使ったパソコンにするか?」というところまで来ている。

インテルのほうでは、年内に出荷させる予定の「Pentium4」でぶっちなりと思っているのだろうが、こうトラブルが続くと、ちょっと心配にもなってくる。こここのところのトラブルには、MTH (Memory Translator Hub) の設計ミス、820チップセット自体の設計ミスなどがあり、トラブルではないが、インテルが予定していたRambusメモリへの切り替えがうまくいっていない、などがある。

まあ、インテルの商売の話なので俺が心配することでもないのだが、OEMメーカーにとってみれば、かなりの不

安材料。これからどうフォローしていくかが問題だとは思いますが、ある意味、インテル自身がOEMメーカーのAMDへの傾斜を後押ししている感じもある。

インテルとしては、Pentium4が出るまで、じっと「我慢の子」という感じだが、はたしてPentium4順調に出るのでしょうか。

AMDは、64ビットもうまくいくか?

さて、かたやAMDのほうだが、独自路線の64ビットプロセッサでようやくLinuxをターゲットにしたようである。AMDの64ビットプロセッサであるHammerシリーズは、インテルが16ビットCPUである8086を32ビット化するときに使った方法を採用して、64ビット化を行う。簡単にいえば、いままでの命令の前に「後ろの命令は64ビットで動いてね」というコード(プリフィックス)を置いて、命令を64ビット化する方法である。この方法では、アーキテクチャ的に従来とほとんど同じで、演算対象が32ビットから64ビットに切り替わるだけなので、プロセッサの設計が簡単になり、その結果、開発コストが抑えられる可能性がある。また、従来のプログラムへの互換性も高い(プリフィックスがなければ、従来通り動く)ため、移行が簡単というメリットもある。

8月のLinuxWorldで、AMDはLinuxサポートに関する発表を行い、Red Hatなどからサポートの約束を取り付けたようだ。

これに対してインテルの64ビットプロセッサであるItaniumは、今までのPentiumIIIなどとはまったくアーキテクチャが違うもの。一応バイナリ互換の機能はあるが、少なくとも、PentiumIIIで同じコードを動かすより

も「効率的」ではない。しかし、すでに64ビット化したLinuxやUNIXなどが用意されつつあり、ある意味、Itaniumの最初のメインOSは、LinuxなどのUNIX系ではないかともいわれている。

AMDもようやくそこに気が付いて、遅まきながら、Linux対応を始めたようだが、さて、どうなることやら。性能によっては「速いAthlon」としての使い道もないわけではないのですが、やっぱりちゃんと64ビット対応してくれなきゃね。

可能性としては、Itaniumマシンよりもコストが低くなる可能性はあるので、Linux対応が進めば、大規模アプリケーションを動かすような場合には、ちょっとした需要があるかも。

マイクロソフトはこのところおとなしい

司法省との裁判が始まってから、マイクロソフトはマスコミ的には、おとなしくなってきた。ビル・ゲイツが他社製品について、過激な意見を吐くこともなくなったし、プレスリリースは、前から予定されていた製品のものだけ、Windows Meの出荷開始もおとなしいものであった。

マイクロソフトは、企業内で使われるパソコンをメインに考えていて、Windows 2000系を主流にしたがっている。しかし、自社内の技術的な問題もあって、Windows 2000系を主力とするのには、次期OSであるWhisperまで待たねばならない。それまでは、Windows95/98が使い続けられることになる。そこで、95/98系の後継OSであるWindows Meは、ビデオ編集プログラムなんかを付けて、「家庭用」というイメージに仕立て上げた。

これだと、多くのビジネスユーザー

は、Windows Meに乗り換えるメリットがあまりなくて、そのまま現在のOSを使い続け、この次の買い換えではWindows 2000系しか選択肢がないという状態になるわけだ。

新しいCPUがいろいろ

さて、年内にもTransMetaのCrusoeチップを採用したパソコンが登場するようである。TransMetaのプロセッサは、低消費電力をうたい文句に、ノートパソコンなどに最初に採用される。いまのところ、ソニー、日立が名乗りを上げており、富士通も製品を出すようである。

インテルはこれに対して、「もうプロセッサの消費電力はかなり小さくて、消費電力のほとんどはほかのところで使われている」というような主張を始めた。これは、CrusoeのLongRunテクノロジーへの対応なのだが、ノートパソコン用途でもパフォーマンスの追求ばかりだった方向を多少転換するようでもある。

インテルによると、ノートパソコンの電力は主に液晶とバックライト、そしてハードディスク、周辺回路などで消費されていて、CPUが使っているのは1割程度であるという。まあ、そういう部分もあるのだが、クロックを上



AMDの64ビットCPUアーキテクチャの解説はこちら
<http://www.amd.com/products/cpg/64bit/overview.html>

げ、ファンや巨大な冷却機構をノートパソコンに要求していたのもインテル自身である。

Crusoeでどれだけ長い間ノートパソコンが稼働できるようになるかは、実際にモノが登場してみないとわからないが、ノートパソコンは全体として、性能重視からバッテリーライフ重視に変わってきたような感じでもある。だけど、多くのノートパソコンは、AC電源があるところで使われることがほとんどで、ほんとにバッテリーで利用される時間はそんなに多くないという話もあるので、一時的なブームで終わってしまう可能性も。

Crusoeは、Mobile Linuxを使ったWebPadなんかのほうが気になるのだが、こちらはいつ頃出るのでしょうかね。

そういえば、インテルはStringARMの後継であるXscaleプロセッサを発表したけど、こちらLinux的にはちょっと注目。なにせIDFのデモでは、1GHzで動作してましたから。このところ、PDAなんかでLinuxを走らせるのが流行っているの、こういう低消費電力で、高性能なプロセッサがあると、ちょっと面白くなってくる。どこかWindows CEマシンなんかで採用してくれると、きっと誰かがLinux載せて、それはそれで面白いんじゃないかと思います。



RSA暗号の特許が切れて、9月から誰でも使えるようになった

<http://www.rsasecurity.com/news/pr/000906-1.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 ～インプライズのLinux製品～

インプライズは、Borland C++ Builderなどの開発ツールをはじめ、アプリケーションサーバやRDBMSを擁する一大ソフトウェアメーカーで、積極的にLinux対応を行う企業でもある。また、カナダCorelとの合併発表 (<http://www.linux24.com/linux/news/today/article/article378816-000.html>) から解消 (<http://www.linux24.com/linux/news/today/article/article458208-000.html>) といったニュースも記憶に新しい。今回は、12月に無償配布も予定されているJavaの開発環境「JBuilder 4」とLinux上で動作するGUI開発環境「Kylix」の情報についてレポートする。

(日刊アスキー編集部・吉川大郎)



JBuilder 4 Enterprise

米国ではすでに発表され、日本でもベータ版が配付され始めたのが、インプライズのJava開発環境である「JBuilder 4」である。Jbuilder 4は、それ自体がJavaのアプリケーションであり、Linux、Solaris、Windows上で動作する。製品の特徴としては、JDK 1.3対応 (JDKはユーザーが選択可能)、100% Pure Javaアプリケーション作成が可能、ビジュアル開発とソースコード開発の完全な同期 (ビジュアル画面で加えた変更は、即ソースコードに反映される。逆も同様)、ソースコードの履歴情報表示機能、などがあげられる。

製品ラインナップだが、基本機能を持ち、Javaの学習用と位置付けられた「Foundation」、ServletやJSPの開発機能を持った「Professional」、J2EE (Java2 Enterprise Edition) を利用したEJB開発など、分散アプリケーションの構築からテストまでを行える

「Enterprise」の3つとなっている。この3つの製品とも上記～までの機能を備えている。Pro～とEnterpriseは価格が付くが、Foundationのみは無償で12月頃に配付される予定である。

日刊アスキー編集部では、JBuilder 4 Enterprise (以下Enterprise) の日本語版「フィールドテスト」版を入手できた (画面1、2)。JBuilder 4 Enterpriseは、テスト環境として、WebサーバやORB (VisiBroker 4.1)、アプリケーションサーバ (Inprise Application Server 4.1) を搭載している。これにより、総合的なテスト環境の中で開発を進めることができる。

Kylix

Kylixは、LinuxのGUI上でプログラム開発を行う「RAD (Rapid Application Development) ツール」開発プロジェクトの名称である。取り扱う予定の言語は、C、C++、Delphiで使われているObject Pascalだ。Kylix最大の特徴は、CLX (Component

Development with Kylix) というライブラリを使うことにより、WindowsとLinux両方で動作するアプリケーションを作成したり、Delphiで作成したアプリケーションをLinux用にポーティングすることができるようになることだ (将来的には逆も可能になる)。CLXはまずKylixに実装されるのだが、Delphiの次のリリースにも搭載される予定だ。Kylixについてより詳しく知りたければ、KylixのWebページ (<http://www.borland.com/kylix/>) は当然のこと、<http://www.borland.nl/bww/europe/norben/array/Linusem/kylix.htm>にアクセスしていただきたい。ここには、Kylixの解説が、スライドとともに簡潔にまとめられている。

日刊アスキー編集部では、KylixについてインプライズRADツール事業部営業推進グループ・グループマネージャー大野元久氏に話を聞く機会を得た。「Kylixの情報に関しては、米国で発表している以上の情報はまだ出せないが」という前提のもとではあるが、今後の方向性や同社のLinuxに対する

姿勢などを語っていただいた。

Kylixについて

大野：基本的にはWindowsとの互換性を重視する。そこでCLXという新しいコンポーネントフレームワークを作った。これを使うとLinux上のネイティブアプリケーションをコンポーネントベースで作ることができ、さらにWindowsでCLXをサポートすることにより、同じコンポーネントアプリケーションを使うことができる。データベースエンジンについても、新しい「DBExpress」という技術により、Delphi、Borland C++ Builderからしかアクセスできないかわりに、非常に高速でコンパクトなアプリケーションが開発できるようになる。これはKylixだけではなくて、将来のDelphi、C++ Builderにも採用されていくという形になる。

Linuxというプラットフォームは、Inpriseにとってどのような位置付けなのか

大野：我々は基本的にWindowsがいい、Linuxがいいといったような見方をとっていない。ただ、Linuxが成長株であることは間違いなく、ここに我々の主要なツールを移植しようとし

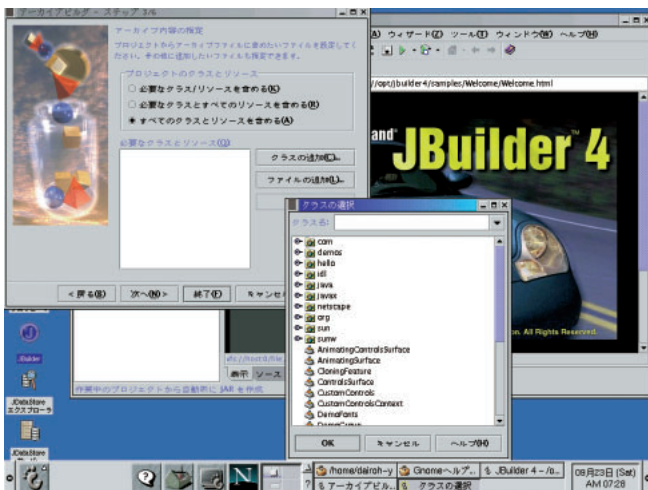
た。もちろんJBuilderもPure Javaに対応している（Linuxで動作する）のだが、ネイティブなDelphiや、C++ Builder（に相当する存在）を投入していこうということだ。それは、Windowsを捨ててLinuxに行けといっているわけではなくて、Windows上の優れたツールとして我々はC++ BuilderやDelphiを提供し続ける。我々はさまざまなプラットフォームに対して中立であり、製品の要望があって、それが我々が答えるべきだと判断すれば製品を投入していく。

アプリケーション開発はエディタで十分という方も多いと思うが、LinuxにRADツールを投入する意義は？

大野：Windowsでも5～6年前に、そう仰っていた方はいらっしゃるのではないと思う。しかし、開発の目的はアプリケーションを作ることであって、コマンドラインで苦労することではない。もちろんコマンドラインを続けていく方を批判するというわけではない。ただ、Linuxという新しい市場に、新しい技術者がどんどん入っていくと考えられる。そういう人たちは、コマンドラインでやりたいということは別に

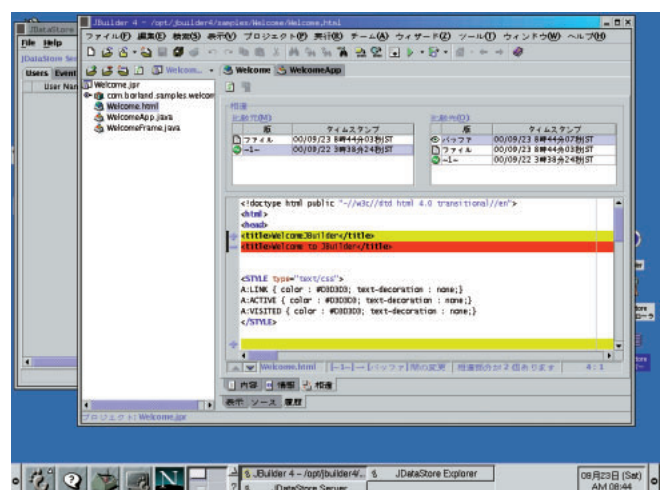
なくて、生産性の高いツールがほしいだろうし、新しいビジネスを目指すのにコマンドラインの使い方から覚えるのではなくて、Delphiに慣れ親しんでいればLinuxでもまったく同じようなスタイルで開発を続けていける。たとえばWindowsからLinuxに移行したいという人たちは、もうコマンドラインのやり方がほとんどわからないかもしれない。そういった人たちはビジュアルな開発ツールを求めているであろうと思うので、市場は今後どんどん増えていくのではないかと。つまり、我々は「Linuxのコマンドラインでまったく問題ない」という方“だけ”を見ているわけではない。もちろんLinux市場の広がりというものがあるって、初めてKylixの市場があるとは思っている。

Kylixに関しては、年内に姿を現わす予定となっており、すでに米国ではデモンストレーションも行われた模様だ。インプライズという一大メーカーからLinux用のRADツールが出てくるのも注目に値するが、CLXによってWindows / Linux間の距離が縮まるのも、見逃せない点だろう。



画面1 「アーカイブビルダ」。

作成したプログラムを、配布用パッケージとして構成できる。配付形態は、Javaアプリケーションからアプレットまで選択可能。



画面2 ソースコードの差分表示。

ソースコードのバージョン管理が容易になっている。変更後の箇所は黄色で、変更前の箇所は赤で表示される。

初級Linuxer養成講座

第14回 リダイレクトとパイプ

前回は、Linuxのコマンドラインにおける「入力と出力」について説明し、プログラムの出力をファイルに保存する「リダイレクト」の方法について説明した。今回はリダイレクトのさらに便利な使い方と、リダイレクトを「進化」させた「パイプ」について説明する。この2つをマスターすれば、Linuxのコマンドラインの「半分」は理解したといつてよいだろう。

文：竹田善太郎
Text：Zentaro Takeda

8年間以上乗りつづけてきた自家用車の走行距離が、ついに20万キロを超えてしまった（写真1）。ここまで、故障らしい故障は1回もなく、エンジン周りの消耗部品だけはまめに交換してきたが、足回りなど新車のときからまったく手を入れていない状態だ。国産のいわゆる「リッターカー」で、だれも見向きもしないようなつまらない車なのだが、正直、ここまで走ってくれるとは予想もしていなかった。あと5万キロくらいは、問題なく走ってくれるだろう。

ボディーやサスペンションにへたっているようすは感じられず、エンジンのパワーも落ちていない。燃費にいたっては新車のころよりも逆に良くなっている。もちろん、内装はそれなりに色あせてきたような気はするが、洗車をめったにしない（半年に1回くらい）せいか、逆に、たまに洗車してみると、塗装にほとんど変化がないのに驚かされる。

だからといって、とりたてて丁寧に運転しているというわけではない。それどころか、一般道の発進では、交通の流れを滞らせてはならぬとアクセル全開だし、高速道路の合流など

ではレッドゾーンまでエンジンを回すのに躊躇することはない。悪路や雪道では、大きな車体を持って余すオフロード車をごぼう抜きしたり（狭い日本の公道では、軽トラが最強の車だと思う）、峠道では、鈍重なくせにむやみに煽ってくる高級車を置き去りにしたりなど、とにかく誉められたような運転はしていないのである。

自動車雑誌のちょうちん記事や、ちょっと車にウルサイ人の言うことを聞くと、必ずといってよいほど「国産車はつまらないし、長く乗れない。やっぱり外車だよ」などと得意げに述べるのが鼻につくが、はっきりいってでたらめだと思う。外車も悪くはないのだが、性能、耐久性、扱いやすさのどの点をとっても、日本車、特に「エコノミー車」は、ずば抜けていると思う。そして、安心して乗れるからこそ、運転も思いきり楽しめるのである。

自宅近辺の駐車場を覗いてみても、10年以上乗っていると思われる国産車は多いし、外見から判断する限り、みな調子はよさそうである。不景気で車が売れないせいだという見方

もあるようだが、10年前の車も今の車も、性能にそれほどの違いはないし、ガソリン車なら排出ガスの性状や燃費に関して、カタログ値ほどの違いがあるとは思えない。買い替える必要のないものを、プロバガンダに踊らされて買い替えるほど、民衆は愚かではないのだ。

環境保護を名目に、車齢の高い自動車の税金を上げて、無理やり排斥しようとする動きがあるようだが、そんなことをするくらいなら、整備不良車や不法改造車をなんとかしたほうがましだと思う。古い車に乗りつづけるには、同じ車を1台くらい買えるだけの維持コストがかかり、部品メーカーや修理工場などの自動車産業にも貢献しているはずなのに、そういったユーザーを裏切るような措置を行政に求める自動車産業は卑怯だと思う。

ところで、PCの世界でも、やたらと重いOSやアプリケーションを普及させて、古いハードウェアを駆逐しようとする動きが常にあるが、それでもPentium以降のPCならば、まだしばらくは十分に使える。ここで、古いマシンを使うならやっぱりLinux... といいたいところだが、

Linuxの世界でも重いGUIやアプリケーションが増えつつあって、ちょっと心配ではある。でも、OS本体は昔と変わらず軽く動いてくれるし、家庭内のファイルサーバやWebサーバとして使うのであれば、重いGUIを動かす必要もない。実際、筆者宅ではPentium 90MHzのマシンが、現役のサーバマシンとして24時間運転を続けている。こちら、あと2~3年は使えそうだ。「オンボロ」国産車とPentiumマシンのどちらが長生きしてくれるだろうか。

リダイレクトのいろいろ

今回は、プログラムの入出力を切り替える「リダイレクト」について説明した。リダイレクトを利用することで、Linuxのコマンドにデータを与えたり、結果をファイルに直接保存することができる。リダイレクトの使えないコマンドも一部にはあるが、リダイレクトの使い方を知らなければ、Linuxのコマンドラインを十分に使いこなすことはできない。

簡単におさらいすると、コマンドの「入力」として、とあるファイルの内容を渡したいときには、リダイレクト文字<を、コマンドの「出力」をファイルなどに保存したい場合は、リダイレクト文字>を使って、次のようにコマンドを入力すればよい。

```
$ コマンド名 < 入力ファイル名
$ コマンド名 > 出力ファイル名
$ コマンド名 < 入力ファイル名 > 出力ファイル名
```

入力ファイルや出力ファイルには、通常のファイルだけでなく、デバイスファイルを指定できることがある。たとえば、

```
$ cat < /dev/cua0 > serialinput.log
```

と入力すると、シリアルポート（COMポート）に受信したデータを、そのままserialinput.logというファイルに保存できる。

リダイレクトには、このような単純なファイルへの入出力以外にも、さまざまな使い方があり。その中から、とくに覚えておいたほうがよい使い方を紹介しよう。

新規書き込みと追加書き込み

標準出力からファイルへのリダイレクトでは、リダイレクト文字「>」を使用すると、指定したファイルがすでに存在している場合もそうでない場合も、ファイルが新たに作成されて、その先頭から出力データが書き込まれる。同じ名前のファイルがすでに存在していた場合は、以前の内容は破棄されて、新しいデータがファイルの先頭から書き込まれることになる。たとえば、file1.txtとfile2.txtの2つのテキストファイルの中から、ある文字列を検索して、その結果を保存しようとしたとき、次のように実行したとしよう。

```
$ grep "検索文字列" file1.txt > result.txt
```

```
$ grep "検索文字列" file2.txt > result.txt
```

1行目のコマンドを実行すると、result.txtというファイルが作成されて、その中に検索結果が保存される。ところが、そのまま2行目のコマンドを実行すると、1行目で保存したはずのresult.txtの内容は消去され、2行目の検索結果だけがresult.txtに残ることになってしまう。これでは困る場合、すなわち、もとのデータは残したまま、新しい内容をファイルの末尾に追加したいようなときには、>>というリダイレクト文字を使う。

```
$ grep "検索文字列" file1.txt > result.txt
```

```
$ grep "検索文字列" file2.txt >> result.txt
```

上の2つのコマンドラインを実行すると、file1.txtの中から「検索文字列」という文字列を含んだ行がresult.txtに書き込まれ、さらにその内容を残したまま、file2.txtで同じ文字列を含んだ行

写真1 20万キロの瞬間
20万キロ達成の証拠写真。オドメーターの「枠」が狭くて、10万の桁の「2」の数字の左側が一部欠けて見える。20万キロ以上は走ることを想定していない「仕様」なのだろうか？ いずれにせよ、ゼロが5つ並ぶのはなかなか壮観だ。



を検索し、result.txtの後方にその結果が追加される。このように、既存のファイルの末尾に出力を付け足すようなリダイレクト機能のことを、追加書き込みリダイレクトと呼ぶ(図1)。

ところで、grepで複数のファイルの内容を検索したいのであれば、

```
$ grep -h "検索文字列" file1.txt  
file2.txt > result.txt
```

のように実行することもできて、こちらのほうがスマートに見える。しかし、「-h」などのオプションを覚えていないと、目的どおりの結果が得られなかったりする。Linuxのコマンドラインを使うとき、各コマンドの細かいオプションを覚えているに越したことはないのだが、いきなり複雑なオプションを使おうとするのではなく、最初は愚直と思えるような使い方から始めて、徐々に高度なオプションを習得していったほうがよいだろう。この場合も、たとえばgrepコマンドの-hオプションを覚えるより先に、リダイレクト文字「>」の使い方を先に覚えてほう

が、ほかのコマンドでの応用が利く。

標準エラー出力

リダイレクトの使い方に慣れてきて、いろいろなコマンドの出力をファイルに保存するようになってくると、リダイレクトしているはずなのに、ファイルに保存されないで画面に表示されてしまうという場合に遭遇するだろう。たとえば、コマンドの使い方をコマンド自体のヘルプ機能(--helpオプションや-hオプションなど)を使って表示させ、その内容をファイルに保存しておいて、あとでゆっくり読みたいことがある。

```
$ nkf --help > nkfhelp.txt
```

ちなみに「nkf」とは、テキストデータの日本語文字コードを変換するコマンドのことであるが、このコマンドについての詳細は抜きにして、上のコマンドラインを実行しても、コマンドの使用法のテキストは、nkfhelp.txtファイルに保存されるのではなく、画面上に表示されてしまう(画面1)。

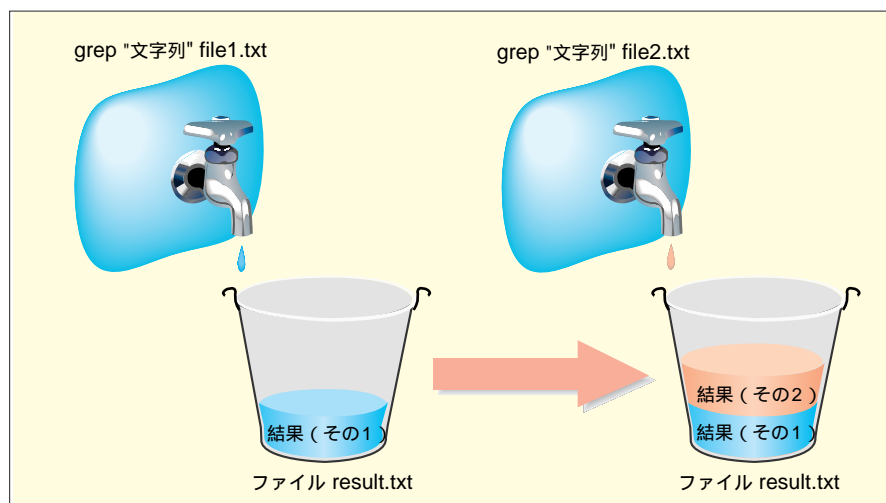


図1 追加書き込みリダイレクト
追加書き込みリダイレクトは、同じ「バケツ」に別々の「蛇口」からデータを継ぎ足していくようなものと思えばよい。ただし、追加される情報は、ファイルの「末尾」に継ぎ足される。

nkfhelp.txtファイルは作成されるものの、中身は空である。

実は、Linuxのコマンドからの出力には「標準出力」とは別に、標準エラー出力というものがあるのだ。どちらも、通常はコンソール画面に一緒に出力されるし、色分けされて表示されるわけでもないのだから、ユーザーにはまったく区別はつかない(図2)。しかし、「>」や「>>」などのリダイレクト文字を使って出力のリダイレクトを行う場合は、「標準出力」に出力された文字列だけがリダイレクトされ、「標準エラー出力」の文字列はリダイレクトされないで画面に表示されるのだ。これは、「エラー出力」という言葉が示すように、プログラムの実行中に発生したエラーや警告などのメッセージ、あるいはヘルプメッセージなどを、処理結果そのものと区別するための仕組みなのだが、「画面上に表示されるものはすべてリダイレクトできる」と信じていると、このように混乱させられることになるのだ。

このような場合に便利なのが、標準出力と標準エラー出力をまとめてリダイレクトするリダイレクト文字&>である。

```
$ nkf --help &> nkfhelp.txt
```

上のコマンドラインを実行すれば、晴れてnkfコマンドの使い方をnkfhelp.txtに保存できるようになる。

標準出力はそのままにして、標準エラー出力だけをファイルにリダイレクトすることもできる。リダイレクト文字として2>を使えばよいのだ。

```
$ nkf --help 2> nkfhelp.txt
```

ところで、なぜ唐突に数字の「2」

が出てきたのかと不思議に思うかもしれない。これは忘れてしまってもよいことだが、Linuxのコマンドの出力は、標準出力、標準エラー出力のほかに、プログラマーが必要に応じて、好きなだけの種類の「出力」を定義することができるのだ。プログラムの内部では、これらの「出力」には、「標準出力は1番」、「標準エラー出力は2番」という具合に数字で番号がつけられている。この数字を「>」記号の前につければ、何番の出力をリダイレクトするかを指定できるのだ。だから、

§ コマンド名 1> ファイル名

とやると、1番の出力、すなわち標準出力をリダイレクトできることになる。つまり、

§ コマンド名 > ファイル名

とやったのと同じことである。プログラムからの出力を、標準出力、標準エラー出力に分類して別々のファイルに保存したい場合は、

§ コマンド名 1> ファイル1 2> ファイル2

のように、続けてリダイレクト先を指定することもできる。まあ、これはかなり特殊な使い方になるので、参考程度に覚えておくだけでよいだろう。通常は、画面に表示されるテキスト全部を一度に保存する&>だけで十分だ。

パイプ

入出力のリダイレクトを使っていると、複数のコマンドを組み合わせでデータを処理するような際に、途中にいくつかのファイルを作るの

```
zen02
[zen-t@zen02 zen-t]$ nkf --help > nkfhelp.txt
USAGE: nkf(nkf32,wnkf,nkf2) [-[flags] [in file] .. [out file for -O flag]
Flags:
b,u      Output is buffered (DEFAULT),Output is unbuffered
j,s,e    Outout code is JIS 7 bit (DEFAULT), Shift JIS, AT&T JIS (EUC)
J,S,E    Input assumption is JIS 7 bit , Shift JIS, AT&T JIS (EUC)
t        no conversion
i_/o_    Output sequence to designate JIS-kanji/ASCII (DEFAULT B)
r        (de/en)crypt ROT13/47
v        Show this usage, V: show version
m[BQSO] MIME decode [B:base64,Q:quoted,N:non-strict,O:no decode]
M[BQ]    MIME encode [B:base64 Q:quoted]
l        ISO8859-1 (Latin-1) support
f        Folding: -f60 or -f
Z[0-2]   Convert X0208 alphabet to ASCII 1: Kankaku to space,2: 2 spaces
X,x      Assume X0201 kana in MS-Kanji, -x preserves X0201
B[0-2]   Broken input 0: missing ESC,1: any X on ESC-[(X)-X,2: ASCII on NL
O        Output to File (DEFAULT 'nkf.out')
d,c      Delete \r in line feed, Add \r in line feed
-L[um]   line mode u:LF w:CRLF m:CR (DEFAULT noconversion)
long name options
--fj,--unix,--mac,--windows      convert for the system
--jis,--euc,--sjis,--mime,--base64  convert for the code
--help,--version
Network Kanji Filter Version 1.9 (2/0002/Shinji Kono)
Copyright (C) 1987, FUJITSU LTD. (I.Ichikawa),2000 S. Kono, COW
[zen-t@zen02 zen-t]$
```

画面1 nkfのヘルプ画面をリダイレクトしたい.....

コマンドの出力をファイルにリダイレクトしたはずなのに、画面にテキストが表示されてしまうことがときどきある。このとき作成されるファイルは、「空」のままである。これは、コマンドの出力が標準出力ではなく、標準エラー出力に送られているからだ。

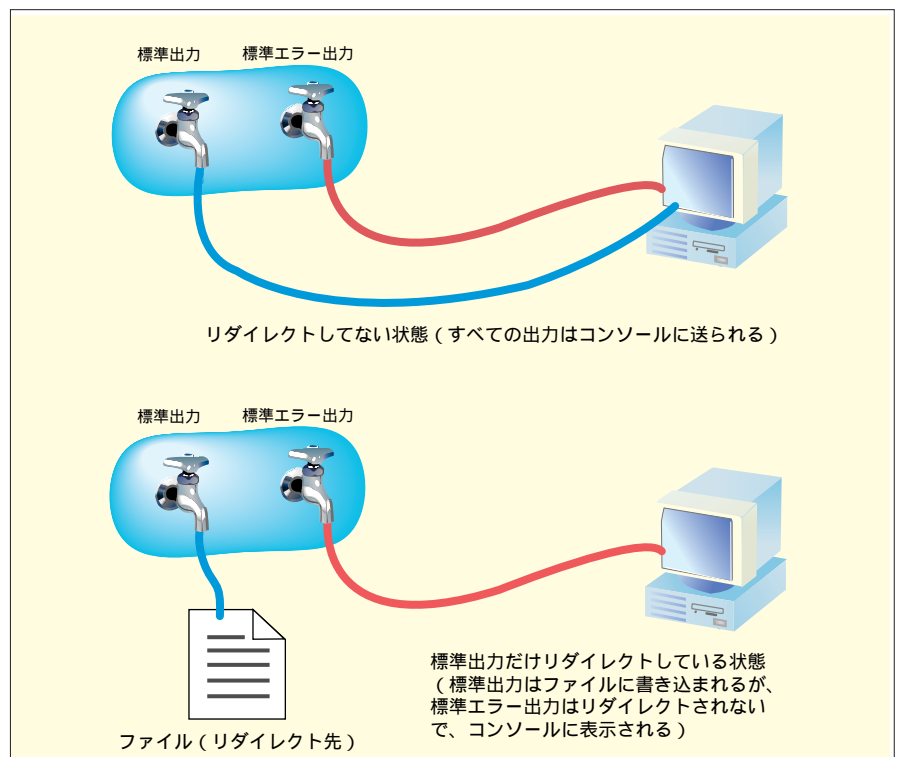


図2 標準出力と標準エラー出力

標準出力と標準エラー出力は、1つの「プロセス」にいくつもの「蛇口」がくっついている様子を考えればよい。リダイレクトしていない状態では、すべての「蛇口」からの出力は同じバケツ（コンソール）に注ぎ込まれるが、一方の蛇口に「ホース」を接続すると（すなわちリダイレクトすると）その蛇口に対する出力だけが、別のバケツ（ファイル）に送られるようになる。

は面倒と感じるようになるだろう。このような場合、2つのコマンドの標準出力と標準入力を直接つなげることができる。これがパイプと呼ばれる機能で、本連載の中でも何度か断りなしに使ってきている。grepコマンドの結果を前出のnkfコマンドで漢字コード変換する場合、

```
$ grep "検索文字列" file1.txt > temp.txt
$ nkf -e < temp.txt > result.txt
```

とする代わりに、

```
$ grep "検索文字列" file1.txt | nkf -e > result.txt
```

というように、コマンドラインを「|」（縦棒）で区切ってつなげることができるのだ（図3）。

パイプを使って複数のコマンドを組み合わせる場合、単に、あるコマンドの出力が別のコマンドの入力として使われるだけでなく、これらの複数のコマンドは同時に実行されることに

なる。ちょっと難しい話になってしまうが、これこそがLinuxの特徴のひとつであるマルチタスクの特徴なのだ。

パイプは、マシンの性能が許す限りいくらでも長くつなげることができる。たとえば、

```
$ cat data1.txt data2.txt | nkf -e | grep "シロクマ" | sort | cut -f 1-3 | nkf -s > result.txt
```

などという長大なコマンドラインもありうるが、通常は2、3個のコマンドをつなげるような使い方がほとんどだろう。



パイプの便利な使い方

「パイプ」という用語は、前回リダイレクトの説明をするのに「浄水フィルタ」を例に取ったのと同じく、水道設備や化学プラントなどで、構成要素を「パイプ」を使って接続する様子にたとえてつけられている。パイプラインと呼ぶこともよくある。

現実の水道パイプでは、中に水が流

れているわけだが、Linuxのパイプの場合にはデータが流れることになる。このデータの流れと変化を頭の中に想像しながら、コマンドを一列に配列すれば、パイプラインを完成させることができるのだ。

たとえば、さっきから何回も登場した「nkfコマンド」を使う場面としてよくあるのが、シフトJIS形式のテキストファイルをLinuxマシン上で加工して、再びシフトJISのテキストファイルにしたい場合である。ここで日本語文字コードの詳しい話はしないが、現在のLinuxディストリビューションのほとんどは、日本語文字コードとしてEUCを使うのを前提に設定されていることが多い。このため、シフトJISのファイルをそのまま処理しようとしても、うまくいかないことがあるのだ。

このようなときに便利なのがnkfコマンドであり、このコマンドをパイプラインの中で使うようにすれば、いちいちファイルの中身をEUCに変換しなくても、1つのコマンドラインで目的の処理ができることが多いのだ。

```
$ nkf -e file1.sjis | コマンド1 | コマンド2 ... | nkf -s > result.sjis
```

このように、パイプラインの先頭と末尾でnkfコマンドを使い、最初にEUCに、最後にシフトJISに文字コードを変換してやればよいのである。

パイプを利用するさらに一般的な例としては、ファイルの内容を1ページごとに区切って表示してくれるlessコマンドを忘れてはならないだろう。lessコマンドについては、本連載中でも何度か使っているが、このコマンドなしに快適なコマンドライン生活はないと言い切ってしまう。

lessコマンドを使う場面はいろいろ

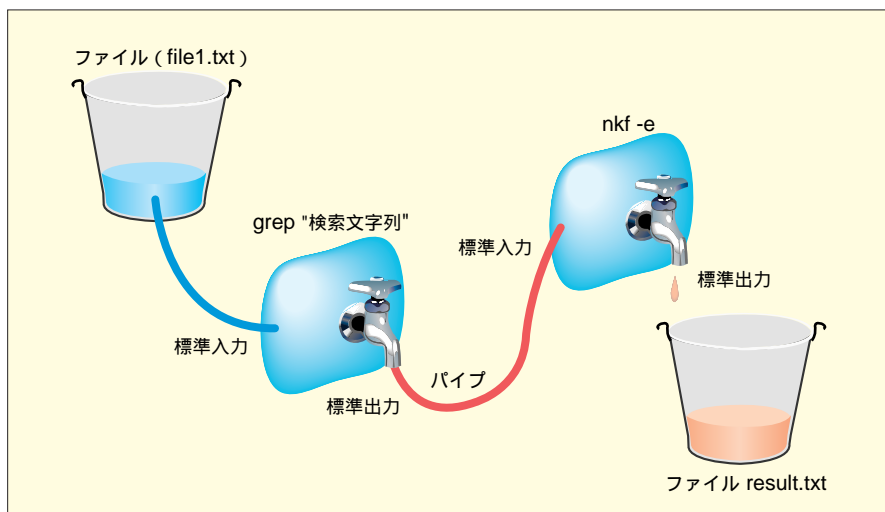


図3 パイプの概念

grepコマンドの出力（標準出力）は「パイプ」を通じてnkfコマンドの標準入力に「直接」渡される。このようにコマンド同士を組み合わせることを「パイプラインを作る」ということもある。大元のファイル（file1.txt）と結果ファイル（result.txt）は「リダイレクト」によってコマンドと結ばれるのだが、図でわかるように、これらもパイプラインの要素である。

ある。たとえば、単にテキストファイルの内容を閲覧したい場合には、

```
$ less ファイル名
```

というように、直接ファイルを指定してやるだけでよい。lessコマンドは、標準入力から受け取ったデータもページごとに表示してくれるので、

```
$ less < ファイル名
```

のように、ファイルをリダイレクトしてやってもよい。そして本題であるパイプと組み合わせて使う場合には、別のコマンドの最後にlessコマンドをつなげてやればよい。

```
$ nkf -e file1.sjs | grep "アナグマ" | less
```

lessコマンド自身の使い方は、lessコマンド実行中に「h」キーを押せば操作方法が表示されるが、スペースキーで次のページ、bキーで前のページ、qキーで終了ということだけを覚えておけば、当面はよいだろう。lessコマンドは、単にファイルの表示だけでなく、文字列の検索や、テキストエディタの起動などでもできるのだが、これについては今回は触れない。とにかく、「パイプの最後にless」という使い方を覚えておくだけで、コマンドライン生活がとても快適になるということだけは保証しよう。

ところで、パイプでもリダイレクトと同じように「標準出力」と「標準エラー出力」を切り替えられないかと思えるかもしれないが、残念ながら、Linuxで標準的に使われているbashでは、「&|」とか「2|」というような使い方はできない。ちょっと複雑になる

のだが、

```
$ コマンド名 2>&1 | less
```

というように、パイプ記号の直前に2>&1というリダイレクト指示をつけてやると、標準出力と標準エラー出力の両方がパイプに出力されるようになる。リダイレクト指示は、使い方がちょっと難しいので、ここではあえて覚えなくてもよいが、なにかと便利なので丸暗記しておいても損はないだろう。

ちなみに、bashと並んでよく使われているシェル「tcsh」では、

```
% コマンド名 |& less
```

という、より簡単な記法が使える。

パイプは「分岐」できる！

最後に、パイプを使ううえでちょっと便利なコマンドも紹介しておこう。水道管では、パイプの途中にT字型をした分岐用の部品をつけることで、水

の流れを2つに分けることができるが、Linuxのパイプラインでも同じようなことができる。その名もずばり、tee (ティー) というコマンドを使えばよいのだ。

teeコマンドは、標準入力の内容を指定したファイルに保存すると同時に、標準出力へそのまま出力するという、一見無意味な動作をする。しかし、たとえば、コマンドパイプラインの途中結果を保存しておきたいような場合に、次のように利用するととても便利である。

```
$ nkf -e file1.sjs | tee file1.euc | grep "アナグマ" | tee result.euc | less
```

上のコマンドラインを実行すると、file1.sjsをEUCに変換しただけのfile1.euc、そして、その中から「アナグマ」という文字列の含まれている行だけを抜き出したresult.eucという2つの途中結果ファイルを残しながら、コマンドライン全体の結果をlessコマンドで見ることができる(図4)。

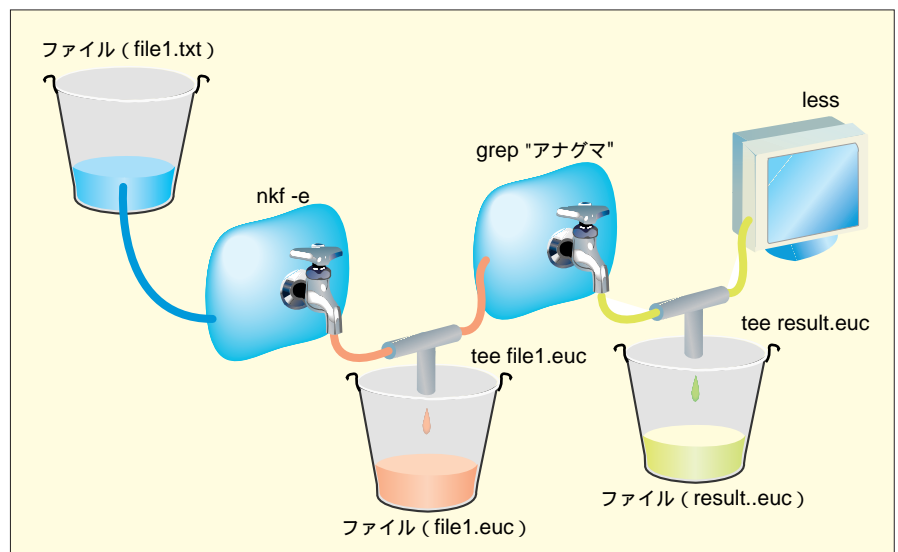


図4 teeコマンド

teeコマンドは、水道管に接続するT字型の「分岐パイプ」のようなものである。入力されたデータをそのままファイルに保存すると同時に、同じデータを標準出力に流す。

InterBase 6.0

今回は駆け足でInterBase 6を紹介してきましたが、今回はもう少し詳しく説明していきましょう。まず、データベースを使用するうえで必要になる重要な概念であるトランザクションについて説明します。

第2回トランザクションを理解しよう

文：加藤大受

Text: Daiju Kato kato@ms.tokyo.jcom.ne.jp

前回、InterBaseの概要とインストールについて説明しました。先月号のCD-ROMに製品が収録されていましたが、すでにWindows版については8月31日の時点でアップデートされています。アップデートされたのはクライアントツールであるIBConsoleがメインで、IBConsoleのバージョンが1.0.0.309に変更されています。Linuxをデータベースサーバとして使用し、WindowsでIBConsoleを使用されている方は必ずアップデートを行ってください。また、さらに新しいIBConsoleを使用したい方はIBDI (InterBase Developer Initiative) のページからIBConsole Build 319をダウンロードしてください。

InterBaseのソース、バイナリのダウンロードサイト
(**メインプライズのWebサイト**)

[http://www.borland.com/interbase/downloads/IBConsole Build 319のダウンロードサイト](http://www.borland.com/interbase/downloads/IBConsole_Build_319のダウンロードサイト)

(**IBDIのWebサイト**)

<http://www.interbase2000.org/binaries.htm>

トランザクションを理解しよう

データベースがデータ処理を行うとき、必ずトランザクションが発生します。トランザクション処理ではすべてのデータ処理が必ず成功するか、あるいは元の状態に戻るかのどちらかとなります。このトランザクションの概念を特

性を示す言葉の頭文字をとって、ACID特性と呼びます。

Atomicity (原子性)

すべてが完全に実行されるか、処理が完結しない場合には元の状態に戻る

Consistency (一貫性)

処理の順番に関わらず結果が同じになる

Isolation (独立性)

他のトランザクションの影響を受けない

Durability (耐久性)

いったんトランザクションが完結したら障害が発生してもデータの状態が変化しない

失敗した場合の復元作業も含め、トランザクションが正しく完了することがデータベースソフトの大前提です。このトランザクションをどのように管理し、どの程度のパフォーマンスで処理できるかでデータベースソフトの質が決定されます。どんなに速いパフォーマンスを提供しても、トランザクションが正しく処理されなければデータベースソフトとしての価値を持ちません。

簡単な例を説明しましょう。トランザクションを開始し、テーブルAに1行データを追加し、フィールドBに100を格納します。続いて、1レコード目のフィールドBのデータを50に書き換え、トランザクションを終了します。このとき、1レコード目のフィールドBは50という値が格納さ

れています。通常、トランザクションを開始するときには START TRANSACTION などの命令を発行し、トランザクションを完了するときは COMMIT などの命令を発行します。トランザクション処理中に問題が発生した場合はトランザクション処理前の状態に戻します。これを ROLLBACK 処理といいます。トランザクションは必ず COMMIT または ROLLBACK で終了します。中途半端な状態で終了することはありません。トランザクションは必ず一貫性を持っており、他のトランザクションに影響されず独立性を持っています。この原理が守られているので複数のクライアントからデータが更新されてもトラブルなく処理が進むのです。

InterBase などのリレーショナルデータベースでは、通常、データの追加・更新・削除などの処理は SQL 文を利用して操作します。SQL は Structured Query Language の略で、「エス、キュー、エル」あるいは「シーケル」と呼びます。海外では後者で呼ぶことが多いようです。

SQL はデータベースとの対話を効率よく実現するために開発された言語で、1974年にIBMのサンノゼ研究所で開発されました。

SQL を最初に採用した商用データベースエンジンは Oracle で、これ以降ほとんどのリレーショナルデータベースが SQL 言語に対応していきました。SQL 言語は ANSI によって標準化され、現在ではほとんどのデータベースで採用されています。しかし、製品ごとに拡張されており、たとえば、Oracle の場合は PL/SQL と呼ばれる SQL に手続き型のインターフェイスが提供されたものが装備されています。

InterBase の場合は、Dynamic SQL (DSQL) と呼ばれる SQL 文にストアードプロシージャ / トリガー用に条件文

や繰り返し処理などが追加された SQL 言語が提供されています。InterBase は他のリレーショナルデータベースに比べ、SQL 言語の拡張が少なく、他のデータベースへの移植性が高くなっています。

これは別な面から見た場合、拡張が少ない = 機能が少ないとよく判断されがちですが、基本的な機能がきちんと備わっていて、かつ高速な処理が提供されているほうが使いやすいデータベースといえるでしょう。

一般的にこの SQL 言語を利用してトランザクション処理を行います。トランザクション処理を速く行うために API (Application Programming Interface) が用意されており、この API をカプセル化しているのが ODBC、BDE/SQL-LINK、JDBC など、これらのミドルウェアを使用することで簡単にクライアント / サーバ型のアプリケーションを開発することができます。

InterBase のトランザクション機能

トランザクションの概念について簡単に説明しました。それでは InterBase のトランザクション機能を見ていきましょう。前号で InterBase は他のリレーショナルデータベースとは異なるアーキテクチャを使用していると説明しました。このアーキテクチャの違いを説明しながら、InterBase のトランザクション機能について見てみましょう。

トランザクションを確実に行うため、トランザクション中に他クライアントが同時に同じデータを使用しないように排他処理を行います。排他処理はテーブルロックまたはレコードロックを伴います。テーブルロックとは、あるトランザクションが実行中に、更新されているテーブルをトランザクションの間、文字通りロックしてしまい、他のクライアントがアクセスできないようにすることです。レコードロックは現在のトランザクションがレコードをロックし、他のクライアントがそのレコードを読めないようにすることを指します。この排他処理があることで、トランザクション中に他のクライアントからの操作、つまり他のトランザクションの影響を受けません。

では、異なるクライアントがそれぞれトランザクションを実行し、同一データを更新した場合はどうなるのでしょうか？ InterBase の場合、先に実行されたトランザクションが優先されます。たとえば図1のように、クライアント A とクライアント B が、テーブル AAA のレコード 1 のデータを更新しようとした場合、先にトランザクション処理を開始したクライアント A のトランザクションが優先さ

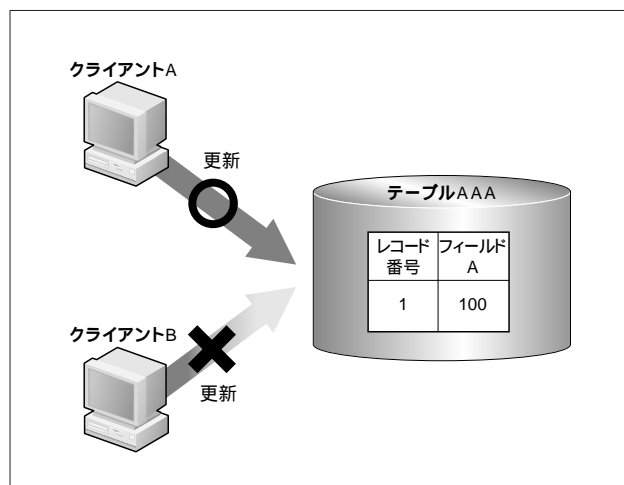


図1 同一データの更新によるトランザクションの失敗

れ、クライアントBのトランザクションは失敗します。このとき、クライアントBのトランザクションはROLL BACK処理を行うこととなります。

InterBaseにはトランザクション処理が衝突した場合の方法がモードとして用意されています。デフォルトはWAITモードで、トランザクションが衝突したときに実行されているトランザクションが終了するまで待機し、その後でトランザクション処理を行います。待機せずにエラーを返すモードがNO WAITモードです。このモードをうまく使いこなすことで処理したいトランザクションを確実に実行することができます。

排他処理と同じようにトランザクション処理を確実に行うために提供されている機能として、排他レベル(アイソレーションレベル)の設定があります。排他レベルとは、あるトランザクションがテーブルにアクセスする場合、トランザクションの独立性を指定することです。もう少し簡単に説明すると、あるトランザクションと他のトランザクションの関係を決定する要因となります。InterBaseには次の3つの排他レベルのオプションが用意されています。

SNAPSHOT

デフォルトの排他レベル。この設定では、トランザクションが起動したときのデータベースのビュー(ユーザーが望む形のデータベースのサブセット)が用意されます。このビューは、COMMIT済みのビューで変更されることはありません。データベースの行に対しては、他のトランザクションからも更新と挿入が可能ですが、その変更はビューに一切影響しません。この設定で開始されたトランザクションにとっては、トランザクションの開始時に保存されていた行が現在見える行、すなわち読み取り可能な行ということになります。また、この設定で開始されたトランザクションによって行の更新や削除が試みられた場合、その行が別のトランザクションによってすでに変更が加えられていた場合には、更新衝突が報告されます。

SNAPSHOT TABLE STABILITY

この設定では、この設定で開始されたトランザクション以外のトランザクションは、同一のテーブルに対して挿入や更新、削除ができなくなります。ただし、テーブルの行の読み取りだけは、他のトランザクションからも可能です。

READ COMMITTED

この設定では、データベース中のCOMMIT済みのデータ全部に対して読み取りが可能になります。

また、他のトランザクションによって更新、COMMITされた行に対しても更新が可能になります。この場合、更新はCOMMIT済みデータに対して行われるため、更新喪失は発生しません。

この3つの排他レベルを使い分けることで、他のトランザクションがどのように処理されるかが決定されます。この排他レベルとWAIT / NO WAITモードの組み合わせで非常にバラエティに富んだトランザクション処理が可能になっています。

前号からInterBaseは他のリレーショナルデータベースとは異なるアーキテクチャを採用していることを書いてき

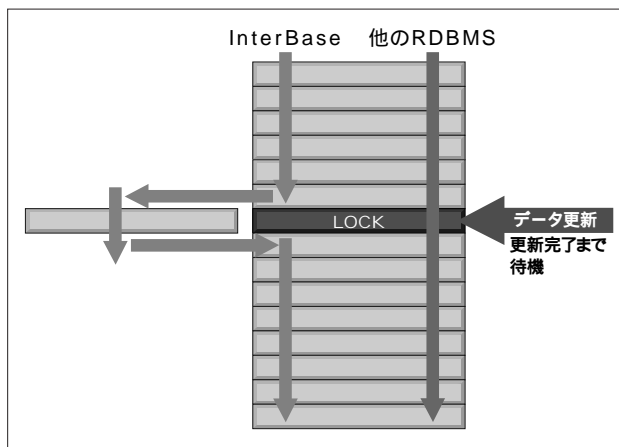


図2 書き込みトランザクションと読み取りトランザクションが重なった場合

SNAPSHOT/READ COMMITTED			SNAPSHOT TABLE STABILITY	
	UPDATE	SELECT	UPDATE	SELECT
SNAPSHOT/ READ COMMITTED	UPDATE SELECT	(注)	x	x
SNAPSHOT TABLE STABILITY	UPDATE SELECT	x x	x x	x

: 衝突することなしにデータ処理を実行
 x : データ処理の衝突が起こり、エラーとなる
 注 : 同一範囲をアクセスするときのみ衝突
 表1 InterBaseのトランザクションのパターン一覧

ました。すでに何らかのデータベースを使用している経験がある方は、表1のパターン一覧を見て不思議に思うかもしれません。InterBaseは他のクライアントが書き込みのトランザクションを行っているときに、同じデータを読み取ることでできる排他レベルを持っているのです。

InterBase以外の製品では、トランザクション（書き込みトランザクション）がデータを書き込んでいる場合、排他処理が行われロックが働いているため、他のトランザクション（読み取りトランザクション）が同一データを読むことはできません。しかし、InterBaseの場合、データを履歴として管理しているため、最新の履歴を読み取ることで、書き込み中でも読み取りが可能になります（図2）。

	読み込み & 読み込み	書き込み & 読み込み	書き込み & 書き込み
InterBase	OK	OK	NG
他のRDBMS	OK	NG	NG

図3 トランザクションの共存

もちろん、他のリレーショナルデータベースのように排他処理を行い、読み取りができないようにすることも可能です。SNAPSHOT TABLE STABILITYの排他モードが他の製品と同一の動きを示し、デフォルトのSNAPSHOTの排他モードが、書き込みと読み取りトランザクションの共存を可能にしています（図3）。

それでは、実際に行ってみましょう。WindowsからInterBaseがインストールされているLinuxにtelnetで接続できる人はtelnetを2つ、Linux上で行いたい人はKDEまたはGNOME上でTerminal(端末)を2つ開いてください。InterBaseのコマンドラインユーティリティであるisqlを使用します。isqlはSQL文ベースでデータベースの管理ができるユーティリティです。-Uはユーザー名、-Pはパスワードを指定するオプションです。ここではTerminal AとTerminal Bで同じ処理を行います。ただし、Terminal Aが先に実行するものとします。isqlが実行されたらデータベースを開きます。

まず、デフォルトの状態でも両方で更新のトランザクションを実行してみましょ。デフォルト状態では、WAITモードで排他レベルはSANAPSHOTです。そのため、同じテーブルの同じレコードに書き込もうとするとトランザク

Terminal A	Terminal B
<pre>\$ cd /opt/interbase/bin \$./isql -U SYSDBA -P masterkey Use CONNECT or CREATE DATABASE to specify a database SQL> CONNECT /opt/interbase/examples/employee.gdb; Database: /opt/interbase/examples/employee.gdb, User: SYSDBA SQL> UPDATE EMPLOYEE SET FIRST_NAME='DAIJU'; SQL> SQL> ROLLBACK;</pre>	<pre>\$ cd /opt/interbase/bin \$./isql -U SYSDBA -P masterkey Use CONNECT or CREATE DATABASE to specify a database SQL> CONNECT /opt/interbase/examples/employee.gdb; Database: /opt/interbase/examples/employee.gdb, User: SYSDBA SQL> UPDATE EMPLOYEE SET FIRST_NAME='DAIJU'; SQL> SQL> ROLLBACK;</pre>

画面1 デフォルトでのトランザクションの衝突

Terminal A	Terminal B
<pre>SQL>SET TRANSACTION NO WAIT SNAPSHOT TABLE STABILITY; SQL>UPDATE EMPLOYEE SET FIRST_NAME='DAIJU'; SQL> SQL>ROLLBACK; SQL>QUIT;</pre>	<pre>SQL>SET TRANSACTION NO WAIT SNAPSHOT TABLE STABILITY; SQL>SELECT * FROM EMPLOYEE; Statement failed, SQLCODE = -901 lock conflict on no wait transaction SQL>ROLLBACK; SQL>QUIT;</pre>

画面2 トランザクションの衝突によるエラー

ションの衝突が発生し、Terminal BはTerminal Aのトランザクションが終了するまで待機します。待機中はプロンプトが点滅しています。

Terminal AのトランザクションをROLLBACKするとTerminal Aのトランザクション処理が終了するので、Terminal Bにコンソールが表示され、待機していたトランザクションが処理され、コマンド待ち状態となります。WAITモードが正しく動作していることを確認できたとします(画面1)。

ここではサンプルデータベースを更新したくないので両方ROLLBACK処理をしています。

それでは続いて排他レベルを変更してみましょう。排他レベルを変更するには、SET TRANSACTION コマンドを指定してトランザクションを開始します。デフォルトの場合は、明示的にSET TRANSACTION コマンドを実行しなくても問題ありませんが、排他レベルやモードを指定したいときは必ずSET TRANSACTION コマンドを実行します(表2)。

NO WAIT SNAPSHOT TABLE STABILITYでトランザクションを実行した場合、書き込みトランザクションと読み取りトランザクションが衝突し、すぐにエラーが発生することが確認できるでしょう(画面2)。表1のパターンをいろいろと確かめてみることで、InterBaseの提供している多彩なトランザクションの機能を実感することができます。

isqlはこのように簡単なテストを行ったり、SQL文ベースで処理する場合に非常に便利なツールです。isqlをうまく使いこなすことでtelnetを利用したデータベースサーバ

のリモートメンテナンスを実現することができます。

isqlにはさまざまなコマンドラインオプションがあり、SQL文が書かれたテキストを指定することで、複数の処理を一度に行うようなことも可能です。この機能はデータベースやスキーマ設計を行うときに非常に便利な機能で、SQL文が書かれたテキストファイルを保存しておくことでいつでもスキーマ構造を持つデータベースを作成することができます。

コマンドラインツール

InterBaseにはisql以外にもデータベースのバックアップ/リストアを行うためのgbak、データベースの検査を行うgfix、CやPASCALなどの言語で埋め込みSQLを使用するときのプリプロセッサであるgpre、ユーザーやグループの追加・管理を行うgsec、データベースの分割と復元を行うgsplitなどが用意されています。

InterBase 3.3時代までに付属していたQLIやGDEFなどのツールも付属していますが、これらのツールはまったく改良されていないため、現時点ではほとんど使えません。これらのコマンドラインツールの使い方の詳細は、IBDIのWebサイトからInterBase 6.0の版のドキュメント(英語)を参照するか、インプライズのWebサイト(<http://www.inprise.co.jp/>)にある、InterBase 5.6 for Windows 95/NTのトライアル版に付属するPDF形式のドキュメントを参照するとよいでしょう。

InterBase 6.0の版のドキュメント
(英語、IBDIのWebサイト)

```
SET TRANSACTION [NAME transaction]
    [READ WRITE | READ ONLY]
    [WAIT | NO WAIT]
    [[ISOLATION LEVEL] {SNAPSHOT [TABLE STABILITY]; READ COMMITTED [[NO] RECORD_VERSION]]]
    [RESERVING <reserving_clause>! USING dbhandle [, dbhandle ...]];
    <reserving_clause> = table [, table ...]
    [FOR [SHARED | PROTECTED] {READ | WRITE}] [, <reserving_clause>]
```

引数	説明
NAME transaction	トランザクション名を指定する(SQLのみ)。transactionは初期化された宣言済みホスト言語変数
READ WRITE	テーブルの読み込みと書き込みができるトランザクションであることを指定する(デフォルト)
READ ONLY	テーブルの読み取りだけが可能なトランザクションであることを指定する
WAIT	他のトランザクションとロックが衝突した場合、アクセスできるまで待機する(デフォルト)
NO WAIT	ロックが衝突した場合、即座にエラーを返す
ISOLATION LEVEL	同一のテーブルを、他の並列トランザクションと同時にアクセスした時の排他レベルを指定する(デフォルトはSNAPSHOT)
RESERVING	アクセス範囲を一部のデータベースだけに限定する(SQLのみ)
reserving_clause	
USING dbhandle [, dbhandle ...]	

表2 SET TRANSACTION コマンドの構文

http://www.interbase2000.org/ib_doc.htm

InterBase 5.6 for Windows 95/NT Trial Editionのダウンロードサイト(インプライズのWebサイト)

<http://www.inprise.co.jp/download/ibase.html>

データベースの新規作成

InterBaseなどのリレーショナルデータベースを使用するときは、一般的にSQL言語を使用することを説明しました。それでは実際にSQL言語を使って新規にデータベースを作成してみましょう。

SQL言語にはDDL(Data Definition Language)文とDML(Data Manipulation Language)文があります。DDL文はテーブルの作成などに使用し、DML文はデータの追加・削除・編集に使用します。

データベースの作成はCREATE DATABASE文で行います。CREATE DATABASE文にはたくさんの引数が指定できますが(表3)、よく使う引数がファイル名、ページサイズ、デフォルトキャラクタセットです。キャラクタセットについてはのちほど詳しく説明しますが、ここではEUCを設定しています(画面3)。

データベースはデータ、テーブル、インデックス、プロシージャなどが格納されている器のようなものです。ページとはサーバが扱う最少の入出力のI/Oの単位です。ページサイズは1024バイト(デフォルト)、2048バイト、4096

バイト、8192バイトの4種類が指定できます。

ページを大きくするとデータベースのサイズは大きくなりますが、一度にアクセスできるバイト数が多くなるのでパフォーマンスが若干良くなります。デフォルトの1024バイトはInterBaseの最初のバージョン(プロトタイプ)であるPDP 11というマシン上で使用されていたときの名残です。現在は一度にたくさんのバイト数を読み取ることができますので、データベースを作成するときはページサイズを大きく指定するといいでしょう。

SHOW DATABASE文でデータベースの属性を見た場合、ページサイズやキャラクタセットの情報だけでなく、スイープ(SWEEP)についても表示されます。InterBaseはトランザクションが発生するとトランザクション番号を設定し、トランザクションを実行します。COMMITもROLLBACKもされていないトランザクション番号と最新のトランザクション番号の差が一定の数値を超えたとき、InterBaseはデータベースのサイズが大きくなるのを防ぐためガーベージコレクションを実行して、古いトランザクション番号を持つデータの履歴を削除します。このガーベージコレクションを行うことをスイープといい、デフォルトで2万トランザクションの差がスイープ間隔となっています。データが削除されるのではなく、履歴がクリアになることに注意してください。

スイープはデータベースの拡大を防ぐのと履歴が多すぎてデータの取り出しに時間がかかるのを防ぐために実行さ

```
CREATE {DATABASE | SCHEMA} '<filespec>'
  [USER 'username' [PASSWORD 'password']]
  [PAGE_SIZE [=] int]
  [LENGTH [=] int [PAGE[S]]]
  [DEFAULT CHARACTER SET charset]
  [<secondary_file>];
<secondary_file> = FILE '<filespec>' [<fileinfo>] [<secondary_file>]
<fileinfo> = LENGTH [=] int [PAGE[S]] | STARTING [AT [PAGE]] int [<fileinfo>]
```

引数	説明
'filespec'	新しいデータベースのファイルを指定する。ファイル名の表記規則はプラットフォームによって異なる
USER 'username'	任意項目。ユーザー名を入力すると、データベースが常駐するサーバのセキュリティデータベースに格納されている有効なユーザー名とパスワードの組み合わせと照合される
PASSWORD 'password'	最大8文字のパスワードを入力すると、データベースが常駐するサーバのセキュリティデータベースに格納されている有効なユーザー名とパスワードの組み合わせと照合される
PAGE_SIZE [=] int	データベースページのサイズをバイト単位で指定する。intには1024(デフォルト)、2048、4096、8192のいずれかを指定できる
DEFAULT CHARACTER SET charset	データベースのデフォルトキャラクタセットを指定
FILE 'filespec'	charsetにはキャラクタセット名を指定し、省略するとデフォルトのNONEに設定される
STARTING	1次ファイルの容量が限界になった時にデータベースページを格納する2次ファイル名を指定する。リモートサーバでデータベースを作成する場合は2次ファイルの指定にノード名を使用することはできない
[AT [PAGE]] int LENGTH [=] int [PAGE[S]]	2次ファイルの開始ページ番号を指定する
	1次ファイルまたは2次ファイルの長さを指定する。同一の文で2次ファイルを設定する場合は1次ファイルの長さを指定する

表3 CREATE DATABASE構文

れるものです。このスイープ間隔は CREATE DATABASE 文を実行するときに設定することもできますし、IBConsoleなどで簡単に変更することができます。

InterBase のサポートする キャラクタセット

InterBase は日本語、英語、フランス語、ドイツ語、中国語など、非常にたくさんの言語をサポートしています。

日本語には Windows で使用されているシフト JIS、Linux や UNIX で使用されている EUC、メールなどで使用されている JIS の 3 つのコード体系がありますが、InterBase はシフト JIS と EUC をサポートしています。

コード体系が異なると文字コードが異なりますので、正しいコード体系を使うことが必要であり、データを格納するときに指定されたコードに変換してやる必要があります (データベースで使用しているコードに変換)。

たとえば、データベースが EUC ベースでできているとき、Windows であればシフト JIS ベースのコードを EUC ベースのコードに変換してから、データの格納を行う必要があります。

何の言語を格納するかを決定するために、データベースにはデフォルトキャラクタセットという概念があります。CREATE DATABASE 文を実行したとき、何も指定しないと「NONE」というキャラクタセットになります。NONE は文字コードの変換を一切しません。単に送られてきたデータを格納し、格納されているデータを返すだけです。日本語を扱わない場合はこれでも OK です。

日本語を扱う場合は、必ずシフト JIS (SJIS) か EUC (EUCJ) を指定する必要があります。Windows のクライアントからアクセスされる場合なら、使用するデータベースのデフォルトキャラクタセットはシフト JIS を、Linux で使用する場合は EUC を指定すると扱いやすいでしょう。

デフォルトキャラクタセットをシフト JIS または EUC にすると、InterBase は 1 文字を 2 バイトベースで扱います。そのため、30 文字のフィールドには 60 バイトのデータを格納できることとなります。キャラクタセットが正しく設定されていないと、文字列の検索、並べ替え、順番の判定などが正しく行われません。特に日本語を扱う場合、NONE の状態で使用すると検索時に間違った文字列がヒットするなどの不具合が発生しますので、データベースを作成する場合は、必ずシフト JIS または EUC のどちらかをデフォルトキャラクタセットに指定してください。

InterBase はデフォルトキャラクタセット以外に、フィールドごとにキャラクタセットを設定することができます。先ほど、データベースのキャラクタセットをシフト JIS か EUC にした場合、1 文字が 2 バイトベースで扱われると説明しました。日本語を格納する場合はこれでいいのですが、商品コードなどを格納するフィールドの場合、文字数の 2 倍のデータが格納できるのはムダといえます。こういふときにはフィールドごとにキャラクタセットを指定します。たとえば表 4 のような商品マスタテーブルを作成する場合、商品コードや単価のフィールドのキャラクタセットに ASCII を使用すれば、バイト数とフィールドに格納できる最大文字数が一致します。

コレーションオーダー

キャラクタセットには必ず文字の順番があります。文字順をコレーションオーダーと呼びます。シフト JIS のデータベースの場合はシフト JIS の文字コード順、EUC の場合は EUC の文字コード順が使用できます。表 5 は InterBase のサポートしているキャラクタセットとコレーションオーダーの一部です。全世界で使用できるように、ほとんどの国の文字を扱うことができるようになっています。国際化が進むにつれてさらに多くのコレーションオーダーがサポー

```
$ ./isql -U SYSDBA -P masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> CREATE DATABASE '~/sampledb.gdb'
CON> PAGE_SIZE 8192 DEFAULT CHARACTER SET EUCJ;
SQL> SHOW DATABASE;
Database/home/db/sampledb.gdb
      Owner: SYSDBA
PAGE_SIZE 8192
Number of DB pages allocated = 131
Sweep interval = 20000
Default Character set: EUCJ
SQL> QUIT;
```

isqlの実行

作られたデータベースを確認

isqlの終了

画面 3 CREATE DATABASE 文による新規データベースの作成

トされていくでしょう。

また、辞書データベースなどを作成したい場合、フィールドごとのキャラクタセット機能を使用して英語のフィールド、フランス語のフィールド、日本語のフィールドなどを1つのテーブルに格納できるだけでなく、その言語のコード順で並べ替えることができます。多国語対応に優れているのもInterBaseの特徴のひとつです。ただし、同じデータベース内にシフトJISとEUCの両方のテーブルを格納することはお勧めできません。同一言語の場合は必ず1つのコード体系を使用するように指定してください。

また、このフィールドごとにキャラクタセットを設定できる機能はBDE/SQL-LINK、ODBC、InterBase Expressなどのミドルウェアでは使用できません。Windowsでこの機能を使用したい場合はAPIベースでクライアントを作成するか、あるいはBorland DelphiやBorland C++ Builder用のコンポーネントであるIBObjectsを使用するようにしてください。Linuxでこの機能を使用するためには、埋め込みSQLを使用するかAPIベースでクライアントを作成する必要があります。

オープンソース化によって、ツールのマルチキャラクタ対応も進んでくると思います。ヨーロッパ圏では複数の言

フィールド	文字数	キャラクタセット
商品コード	10	ASCII
商品名	50	EUCJIS
単価	10	ASCII
商品説明	200	EUCJIS

表4 商品マスタテーブル

フィールド	文字数	キャラクタセット	フィールド	文字数	キャラクタセット
BIG_5	56	2バイト	BIG_5	56	2バイト
CYRL	50	1バイト	CYRL、DB_RUS、PDOX_CYRL	50	1バイト
OS437	10	1バイト	DOS437、DB_DEU437、DB_ESP437、DB_FIN437、DB_FRA437、DB_ITA437、DB_NLD437、DB_SVE437、DB_UK437、DB_US437、PDOX_ASCII、PDOX_INTL、PDOX_SWEDFIN	10	1バイト
ISO8859_1	21	1バイト	DA_DA、DE_DE、DU_NL、EN_UK、EN_US、ES_ES、FI_FI、FR_CA、FR_FR、IS_ISIT_IT、NO_NO、PT_PT、SV_SV	21	1バイト
EUCJ_0208	6	2バイト	EUCJ_0208	6	2バイト
NONE	0	1バイト	NONE	0	1バイト
OCTETS	1	1バイト	OCTETS	1	1バイト
SJIS_0208	5	2バイト	SJIS_0208	5	2バイト
UNICODE_FSS	3	3バイト	UNICODE_FSS	3	3バイト
WIN1250	51	1バイト	WIN1250、PXW_CSYS、PXW_HUNDC、PXW_PLK、PXW_SLO	51	1バイト
WIN1251	52	1バイト	WIN1251、PXW_CYRL	52	1バイト
WIN1252	53	1バイト	WIN1252、PXW_INTL、PXW_INTL850、PXW_NORDAN4、PXW_SPAN、PXW_SWEDFIN	53	1バイト
WIN1253	54	1バイト	WIN1253、PXW_GREEK	54	1バイト

表5 InterBaseのサポートするキャラクターコードとコレクションオーダー(抜粋)

語を格納したいという要望が強いからです。

次回に向けて

InterBase 6.0からSQL-99(旧SQL-3)をサポートしました。今回は、InterBase 6.0からサポートされたデータ型を中心に、InterBaseのサポートするデータ型について解説し、リレーショナルデータベースを使用する場合に一番重要な要素となるスキーマ設計について解説します。スキーマ設計を説明するとき、必ずといっていいほど正規化という用語を使用します。正規化に関する知識がない方は正規化に関する文献を一読されることをお勧めします。また、SQL言語についても同様に、筆者が推薦する参考文献を記載しておきます。

参考文献

- ・「標準SQL」C.J.デイト、ヒュー・ダウエン著 アスキー ISBN 4-7561-2047-4
- ・「標準SQL-JIS/ANSI/ISO 準拠(改訂第2版)」C.J.デイト著 トッパン ISBN 4-8101-8019-0
- ・「C/S データベース設計 実際編」鷲崎 早雄著 日経BP ISBN 4-8222-8024-1
- ・「データベースちょー入門」栗林 誠也著 広文社 ISBN 4-8777-8016^5
- ・「実践!! データベース設計バイブル」鈴木 昭男著 ソフト・リサーチ・センター ISBN 4-8837-3117-0

Javaプログラミング入門

Javaで3Dゲームを作ろう

今回はクラス/オブジェクトの参照について解説します。クラスからクラスをどのようにアクセスするのか。これが、前回の宿題のツボでもありました。また、クラスの変数にアクセスするメソッドについても解説します。

第5回 クラス/オブジェクトの参照

文: おもてじゅんいち / かざぐるま
Text: Junichi Omote / Kazaguruma

最近、再びJavaが取り沙汰されるようになってきました。大手企業が「Javaプログラマー求む」といった募集を行っているという話もよく聞きます。話題のXMLなどもJavaと相性がよく、XMLのシステムをJavaで構築するのがトレンド(?)のようです。もちろん、Java本来の強味でもある組み込み用途でも人気は高く、携帯電話にJavaが搭載されるという発表もありました(コラム参照)。

こうなると本連載も、もはや趣味の日曜プログラマー養成ではなく、最先端のプログラミング技術講座ということになるのでしょうか。もちろん、本連載が最先端! と大声で言っているわけではなく、あくまでもJavaがということですから……。

クラスとオブジェクト再び

前回、クラスとインスタンスの話をしました。厳密に考えると、ちょっとややこしい話なのですが、アプレットクラスとアプレットオブジェクト(イ

ンスタンス)とは、変身前、変身後みたいなもので、つまり、状態の違いといったようなものです。プログラム上ではクラスといていたものが、実行時にはオブジェクトとなって実在するだけのことで、どちらも同じものの裏表です。

また、インスタンスとオブジェクトは単なる呼び方の違いで、どちらもクラスが実体化したものです。

プログラム言語によって、インスタンスと呼んだり、オブジェクトと呼んだりしますし、「実体化」というイメージからは、「インスタンス化」と呼ぶほうがしっくりいくような気もしますが、「オブジェクト」のほうが、今後いろいろな局面で耳にすることが多いと思われるので、本連載ではインスタンスのことを「オブジェクト」と呼ぶことにします。

クラス間のアクセス

プログラムを見てもわかるように、各クラスは基本的に独立しています。クラスの記述は、

```
class ..... {  
    ...  
    ...  
}
```

というように、クラスごとに{ }で括られているので、アプレットクラスの中に他のクラスを記述することはありません。

とはいえ、クラス同士がお互いにまったくアクセスできないとなると、ただ単にいくつかのクラスがオブジェクトとして存在するだけで、なんの意味もないことになって困ります。

クラス間のアクセスとは、たとえば、

```
ball.process();
```

のように、アプレットからBallクラスのメソッドを呼び出して実行することをいいます。ですから、アプレットからなんらかの方法で、process()というメソッドを持つBallクラスにつながる必要があるのです。

オブジェクトの状態で考えてみまし

よう。まず、アプレットクラスのオブジェクトはアプレットの起動時に、システム（ブラウザ）によって自動的に生成されます。次に、Ballクラスですが、前回に出てきた、

```
new Ball(...);
```

のnewというキーワードによって、オブジェクトが生成されます。しかし、これだけでは単にそれぞれのオブジェクトが存在するだけで、アプレットオブジェクトとBallオブジェクトがつながっていません。そこで、

```
ball = new Ball(...);
```

と、ballという変数にとりあえず代入します。このballは「参照」と呼ばれるもので、newによって生成されたオブジェクトを、文字通り「参照」するための取っ手みたいなものです。このballがBallオブジェクトへの参照になっているために、アプレットオブジェクトは、

```
ball.process();
```

というように、Ballオブジェクトにア

クセスすることができるのです。

図1のように、Racketオブジェクトに対しても同様で、BallオブジェクトとRacketオブジェクトは、アプレットオブジェクト内でそれぞれball、rcという参照のための変数が用意されているため、アプレットオブジェクトから自由にアクセスすることができます。ただしこれは、各クラスでpublic宣言されているメソッドに限られることに注意してください。

ちょっとまぎらわしい名前をつけてしまったので申し訳ないのですが、ここでも、Ballというのは、あくまでもクラスの種類のことで、実在のオブジェクトを参照するものではありません。ですから、最初が大文字のBallを使って、

```
Ball.process();
```

ということではできません。メソッドを実行するときは、そのオブジェクトの参照である小文字のballを使うこととなります。

BallクラスとRacketクラス

さて、ここでやっと前回の宿題についてです。宿題の内容は、「Ballクラス

内部でラケットの座標を知る」というものでした。つまり、Ballクラス内部からRacketクラスにアクセスできるかどうかということになります。

賢明な読者諸氏はもうお気づきかとは思いますが、「参照」がこの問題の鍵なのです。

図1ではアプレットクラスからBallクラスやRacketクラスへはアクセス可能ですが、BallクラスからRacketクラスにはアクセスできません。なぜでしょう？ そうです「参照」がないからなのです。つまり、BallクラスからRacketクラスのメソッドを呼び出すためには、Ballクラス内にもRacketクラスへの参照が必要になるのです（図2）。

「わかった、アプレットクラスと同じように、Racketクラスへの参照を作ればいいんだ」とばかりに、Ballクラスに、

```
rc = new Racket(...);
```

と考えてしまった方、残念ながら大間違いなのです。

Ballクラスにこのような記述をしまうと、悲しいことに図3のようなオブジェクト構成になってしまいます。つまり、Ballオブジェクト内に、また

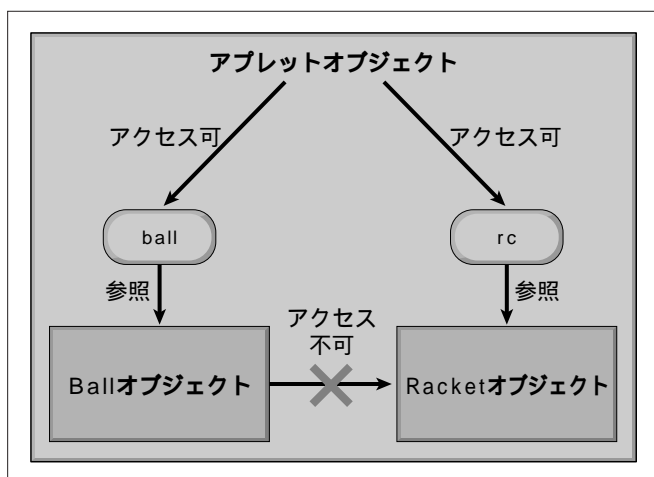


図1 オブジェクト（クラス）間のアクセス

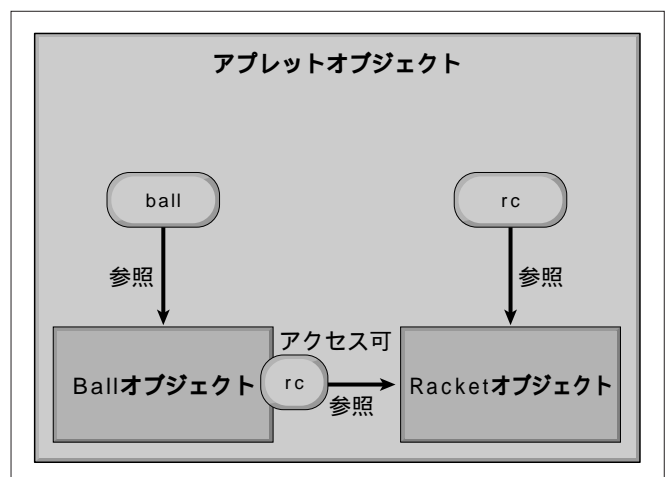


図2 参照があればアクセス可能

別のRacketオブジェクトが生成されてしまうのです。

確かに、rcという変数で参照することができますが、それは本来参照したいはずのアプレット内のRacketオブジェクトではなく、新たに生成された、まったく別のRacketオブジェクトへの参照になってしまうのです。

newというキーワードに気をつけましょう。本ゲームにラケットは1つで十分ですから、Racketオブジェクトに対するnewは全プログラムを通して1カ所だけなければなりません。

前回、マウスの動きをRacketクラスにつなげて、ラケットを自由に動かせる部分をプログラミングしましたが、その際にマウスと連動しているラケットは図3の右側にあるRacketオブジェクトですから、Ballオブジェクト内部に生成されたもうひとつのRacketオブジェクトは、出現するものの、まったく動かないオブジェクトになってしまいます。

ようするに、Ballオブジェクトが必要としているのは、図3の右側にあるRacketオブジェクトへの参照なのです。ということは、新しく作るのではなく、すでにあるものを誰かに教えてもらってこなければならぬのです。

では、そのRacketクラスへの参照はどこで手に入れればよいのでしょうか。

面白いことというか当然というか、当のRacketクラス(オブジェクト)は、自分自身の「参照」を持っていません。たとえ持っていたとしても、まだBallクラスとRacketクラス同士でアクセスができない状態ですから、手渡すことができません。ということは、アプレットクラスから渡してもらえばいいのです。アプレットクラスなら、すでにRacketクラスへの参照をrcという変数に持っていますし、Ballクラスへのア

クセスも可能ですから、簡単に受け渡すことができます。

そのためのプログラムがリスト1です(前回のリスト4)。Ballクラス内での

Racketクラスへの参照は、今後いつでも使えるようにBallクラス全体で使える変数として用意します。Ballクラスの最初のほうの、

```

リスト1 Racketクラスへの参照を渡す
//-----
// Applet Class
//-----
public class ball3d_racket extends Applet
    implements Runnable,MouseMotionListener {

    (省略)
    .....

    public void init() {
        addMouseMotionListener(this);

        buffer = createImage(400,400);

        rc = new Racket(400,400);
        ball = new Ball(400,400,rc);
        bg = new Background(400,400);

        t = new Thread(this);
        t.start();
    }

    (省略)
    .....

//-----
// Ball Class
//-----
class Ball extends Disp3D {
    int vx,vy,vz;
    int x,y,z;
    int gx,gy,gr;
    int sgx,sgy,sgr;
    Racket rc; // ラケットへの参照

    Ball(int w, int h, Racket rc0) {
        super(w,h);
        rc = rc0;
        init();
    }

    (省略)
    .....
}
    
```

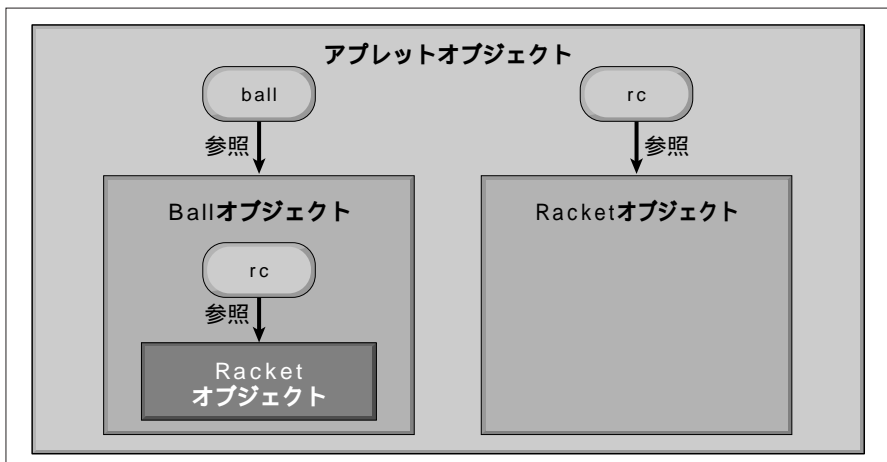


図3 Ballオブジェクト内のRacketオブジェクト

```
Racket rc;
```

がそれにあたります。しかしこの時点では、まだrcに参照は代入されていませんので、

```
Ball(int w, int h, Racket rc0)
```

という、Ballクラスのコンストラクタのパラメータとして、アプレットクラスから渡してもらうようにします。

コンストラクタは、

```
new Ball(....);
```

のように、Ballクラスのオブジェクトが生成されるときに、同時に呼び出されるメソッドです。コンストラクタで受け取れば、Ballオブジェクトが生成されたときに、一緒にRacketオブジェクトへの参照を受け取ることができ、以後、Ballオブジェクト内でいつでもRacketオブジェクトへの参照を使うことができます。

コンストラクタBall(...)のパラメータでは、rc0として参照を受け取ったのですが、パラメータとして使われる変数はそのメソッド内だけでしか有効ではないので、先ほど宣言したオブジェクトが存在する限りクラス全体で有効な変数rcに、

```
rc = rc0;
```

と、移しておきます。これで、Ballメソッドから抜け出たときに、rc0が無効になっても、rcにRacketオブジェクトへの参照が保存されています。

参照を渡す側、すなわちアプレットクラスからは、Ballクラスをオブジェクト化するとき、

```
ball = new Ball(400,400,rc);
```

というように、パラメータとしてRacketオブジェクトへの参照を渡してやるだけです。注意しなければならないのは、このときにすでにrcにRacketオブジェクトへの参照が格納されていなければならないので、

```
rc = new Racket(400,400);
```

という、Racketクラスのオブジェクト化を先に行っておかなければならないということです。

今までも、paint(Graphics g)などのように、外部（システム）からのパラメータとして、オブジェクトへの参照を受け取る場面がありました。これまでは、受け取るばかりで、参照を渡すという場面がなかったのですが、ここで受け渡しの両方ができるようになりましたから、今後どんどんクラスを増やしていった連携をとることができますね。

メンバ変数のメソッド化

めでたくBallクラスからRacketクラスにアクセスすることができるようになりました。あとは、Racketクラスから現在のラケットの座標を取得すればいいだけです。

マウスの動きに合わせて動いたラケットの座標は、Racketクラスの最初にある、

```
int gx,gy,gr;
```

といった変数に格納されています（リスト2）。このような変数をクラスのメンバ変数と呼びます。

Ballクラスからこれらの値を取り出すには、

```
rc.gx
```

などと記述することでもできるのですが、メソッドと同様にクラス外部（ここではRacketクラス外）からアクセスするためには、変数でもメソッドでも宣言時にpublic宣言しておかなければならないのです。

変数が、

```
public int gx,gy,gr;
```

とpublic宣言されていれば、クラス外部からアクセスできますが、これはあまりよくないやり方です。

本連載のこれまでも触れてきたことなのですが、クラスというのは別のプログラムなどでも再利用ができる、いわば部品のようなものなのです。Javaのような言語では、このクラスを中心としてプログラミングするのですが、クラス単位でうまく部品化してデバッグや再利用に役立てるためには、できあがったクラスの内部機構と外部とのインターフェイスをできるだけ切り離しておくほうが賢明です。あるクラスの内部機構を変更する場合でも、外部とのインターフェイスが変わらなければ、そのプログラム全体のほかの部分は変更しなくて済むからです。ま

リスト2 Racketクラスのメンバ変数

```
//-----
// Racket Class
//-----
class Racket extends Disp3D {
    int    x,y,r;
    int    gx,gy,gr;

    .....
    (省略)
    .....
```

た、クラスを再利用する場合には、内部の構造がどうであれ、外部とのインターフェイスだけがわかっていれば、別のプログラムで使うことができます。

このような観点からすると、クラス内部のメンバ変数を外部から直接アクセスするようにプログラミングしてしまうと、内部機構を修正する場合に変数の構成が変わったりすると不都合が発生しますし、うっかりしてクラス外部でその変数を直接書き換えたりしてしまつて、不具合が生じたりするかもしれないために危険なのです。

ですから、クラス中心のプログラミング（オブジェクト指向と呼ばれる）では、メンバ変数を直接参照したり書き換えたりせずに、そのためのメソッドを用意します。外部インターフェイスとしてメソッドが決まっていれば、クラス内部の変数の構成が変わっても、外部からは変化がないように見え、メソッド内で値のチェックなどを行えば、メンバ変数が予想外の値に書き換えられることを防止することができます。

ここでは、ラケットのX座標、Y座標、直径を受け取るために、メンバ変数を直接参照せず

```
get_gx();
get_gy();
get_gr();
```

というメンバ変数を用意します。これらのメンバ変数がそれぞれの値を返しますから、Ballクラスからは、

```
x = rc.get_gx();
y = rc.get_gy();
r = rc.get_gr();
```

と、メソッドを呼び出せばいいことに

リスト3 ボールとラケットの当たり判定

```
//-----
// 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

    (省略)
    .....

    int attack() {
        int gx,gy,gr;
        int dx,dy,d;
        int dvx,dvy;

        gx = rc.get_gx(); // ラケットの2D座標
        gy = rc.get_gy();
        gr = rc.get_gr();

        dx = d2x(x,y,150) - gx; // ボールの2D座標
        dy = d2y(x,y,150) - gy;

        d = (int)Math.sqrt(dx * dx + dy * dy);
        if ( d > (gr/2) ) return 1;

        dvx = (int)(Math.abs(vz) * 0.577 * dx / (gr/2));
        dvy = (int)(Math.abs(vz) * 0.577 * dy / (gr/2));

        vx += dvx;
        vy += dvy;

        vz = -vz;

        return 0;
    }
}

//-----
// Racket Class
//-----
class Racket extends Disp3D {
    int x,y,r;
    int gx,gy,gr;

    Racket(int w, int h) {
        super(w,h);

        x = 0;
        y = 0;
        gx = d2x(x,y,0);
        gy = d2y(x,y,0);
        r = 200;
        gr = d2r(r,0);
    }

    public int get_gx() { return(gx); } // 座標を返す
    public int get_gy() { return(gy); }
    public int get_gr() { return(gr); }

    public void set(int x, int y) { // 座標取得
        gx = x;
        gy = y;
    }

    public void disp(Graphics g) { // 表示
        (省略)
        .....
    }
}
```

なります。

これらのメソッドを利用してできた、ボールとラケットの当たり判定を行う attack()メソッドがリスト3です。

Racketクラスの網掛け部分が、メンバ変数の数値を返すだけのメソッドです。このメソッドをpublic宣言して、メンバ変数はpublicにしないというところがミソです。

反射に角度をつける

ひとくちにボールをラケットで打つといっても、壁での反射とは違って、いろいろと処理することがあります。attack()メソッドが「ラケットでボールを打つ」処理を一手に引き受けているのですが、

- ・ボールがラケットに当たったか
- ・ボールの反射
- ・ラケット上のどの位置かによって反射角度を変える

といった3つの処理を行っています。

まず、ボールがうまくラケットに当たったかどうかはどのように判定すればいいのでしょうか。

ラケットが四角形であれば、X座標、Y座標のそれぞれがある範囲内であればよいということで判定できますが、ラケットは円ですからそうもいきません。ここでちょっと昔に(今?)習った数学を思い出してください。

円とは、ある点(中心)から等距離にある点の集まりです。そしてこの距離が円の半径になります。ということは、円の内部にある点はすべて、中心点との距離が半径より短いということになるのです。逆に中心点との距離が半径より長い点は、すべて円の外部にあるといえます。

つまり、ボールの座標とラケットの中心の座標との距離が、ラケットの半径より短ければ、そのボールはめでたくラケットに当たったことになります。

リスト3を見てください。その処理が、

```
d = (int)Math.sqrt(dx * dx + dy * dy);
if ( d > (gr/2) )    return 1;
```

になります。その前の処理で、dxとdyにそれぞれ、ボールの座標とラケットの中心座標との差が計算されています。横方向にx、縦方向にyだけ離れた2点間の距離は、三平方(ピタゴラス)の定理から、

$$\sqrt{x^2 + y^2}$$

となります。Math.sqrt()は、ルートの計算を行うメソッドで、Javaに用意されています。この結果をdに格納し、それがラケットの半径gr/2よりも大きければ、ボールはラケットに当たらなかった(空振り)として、メソッドから抜けます。

当たったと判定されたら、次はボールを反射させるのですが、このとき単に反射するだけではおもしろくないので、ラケットの当たった位置によって反射する角度を少し変えるようにしています。

```
dvx = (int)(Math.abs(vz) * 0.577
           * dx / (gr/2));
```

これはX方向ですが、ラケットの中心点から離れるにつれてdvxの絶対値は大きくなります。ただしその変化は、最大でも30度の角度になるように、

$$vz * \tan(30^\circ)$$

を掛けています(tan(30)=0.577)。

つまり、このdvxをvxに加算することにより、vzのスピードに対して最大30度分の横方向の加速を受けることになります。vzそのものは変化しませんから、一定のvzに対してvxが加速されるので、結果として角度が変わることになります。

ラケットの中心に当たったときは

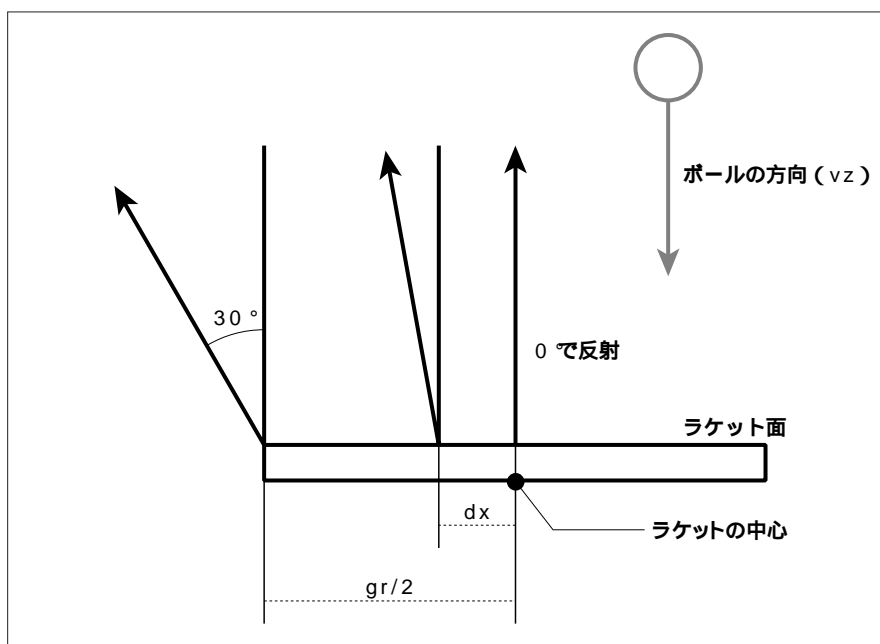


図4 ラケットの位置による反射角度

dvxは0になるため、角度の変化はありません。

このあたりは、数学の「ベクトル」を復習してみると理解が深まります。などとえらそうに言っていますが、実はこの式では中心から遠ざかるにつれても、あまりリニアに角度は変化しません。とりあえず0~30度で変化するように、擬似的に作った式です。

実際に動かしてみると、角度は変化するものの、ちょっと自然ではないような気がします。皆さんもいろいろ試してみてください。

とりあえず遊ぼう

これで、ボールとラケットの当たり判定ができ、とりあえずはボールを打って遊ぶことができそうです。まだ、くずすべきブロックがありませんが、まずはこのあたりで自分のJavaプログラミング習熟の度合いを実感するためにも、また、きたるべきゲーム完成の日のためにもこの辺で少し遊んでみてください。

全プログラムリストは掲載できませんが、前回までの(先月号の付録CD-ROMに収録)ソースリストに、リスト3のattack()メソッドとRacketクラスの3つのメソッドを追加して、リスト4の網掛け部分を修正すれば、とりあえず遊べるアプレットができあがります。

これだけでもラケットさばきの練習用として遊べますが、できることならスピードや反射の角度などを自分なりに調整して、プログラムとアプレットの動きをじっくり理解してみてください。

今回はいよいよ「ブロック」をくずします。プログラムもいよいよ完成に近付いてきました。次回までに、ブロックをくずすプログラムをあれこれ考えてみておいてください。

リスト4 遊べるボール&ラケット

```
//-----
// Applet Class
//-----
public class ball3d_attack extends Applet
    implements Runnable,MouseMotionListener {
    .....
    (省略)
    .....
    public void init() {
        addMouseMotionListener(this);

        buffer = createImage(400,400);

        rc = new Racket(400,400);
        ball = new Ball(400,400,rc);
        bg = new Background(400,400);

        t = new Thread(this);
        t.start();
    }

    (省略)
    .....
}

//-----
// Ball Class
//-----
class Ball extends Disp3D {
    int vx,vy,vz;
    int x,y,z;
    int gx,gy,gr;
    int sgx,sgy,sgr;
    Racket rc; // ラケットへの参照

    Ball(int w, int h, Racket rc0) {
        super(w,h);
        rc = rc0;
        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 ) vz = -vz; // 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 attack() ..... (省略 リスト3参照)

    (省略)
}
}
```

Column

Javaと携帯電話

もうすぐ、iモード端末(携帯電話)にJavaが搭載されます。つまり、Javaプログラムが携帯電話上で実行できるようになるのです。元来、組み込み型に適していて、ネットワークに強いJavaですから、携帯電話に採用されるのは自然なことかもしれません。

いよいよJavaの利用範囲が広まって、皆さんの開発するプログラムが身近なところで実用的に利用できるようになりますね。

PCがどんどん性能アップして、OSがGUIなどを搭載してどんどん複雑になってくるにつれて、手軽にプログラミングできて、なおかつできたアプリケーションを実際に使うといったことは、もはやPC上では少なくなってきました。しかし一方、携帯電話のような今まではユーザーが手を加えるようなことができなかった製品が、かえってプログラミング可能になるというもおもしろいかもしれません。

このような多種のプラットフォームへの対応と、コンパクトな実行環境という面では、現在のところJavaが一番適しているため、iモードだけでなく他の製品でも採用されることになると思います。

今こそ、Javaをマスターしておけば、あなたもいちやく「家電製品プログラマー」になれるかもしれません。いろいろなユーティリティやゲームを自分の携帯電話に搭載したり、iモードでサービスを行ったりすることができることでしょう。

J2SEとJ2ME

現在リリースされているJavaの開発/実行環境はバージョン1.3ですが、これはJava2と呼ばれ、Java2からは各種環境に合った本格的な環境が提供されています。

本連載のように、PCベースで実行させるアプリ/アプリケーションの開発環境は、J2SE (Java 2 Standard Edition) ですが、前述の携帯電話などでの開発と実行環境はJ2ME (Java 2 Micro Edition) と呼ばれるものです。そのほか、サーバ環境としてのJ2EE (Java 2 Enterprise Edition) などがあります。

PC上でJavaアプレットを実行する環境は、JVM (Java Virtual Machine) ですが、J2MEで

は、KVM (K Virtual Machine) といって、頭文字のKは、Kバイト単位ということを意味します。すなわち、このJava Virtual Machineは、携帯電話などの限られたメモリ容量内に組み込まれる必要があるため、非常にコンパクトにまとめられていて、実際のところ数十Kバイト内に収められています。最近のPCアプリケーションが数Mバイトの容量を必要とすることに比べれば、もの凄いい集積技術だということがわかります。

J2ME、J2SE、J2EEいずれの環境においても、同じJavaのソースプログラムが実行可能であるということは、Javaの根本思想であり、事実そのようになるはずで、もちろん、ディスプレイ画面の大きさやその他のデバイス類によって、プログラムの動きは当然のことながら変える必要はありますが、基本的にはPCであろうが携帯電話であろうがJavaのプログラミング自体は同じです。プラットフォームの違いを、JVMやKVMが吸収してくれるのです。

サンマイクロシステムズでは、J2MEを携帯電話だけではなくその他の携帯端末類、双方向ページャ(ポケットベル)、各種セットトップボックスなどへの組み込み用途向けにと提唱しています。

もちろん、携帯電話のような実用的な用途もおもしろいのですが、最近はやりの電子犬(?) などのように、プログラミング可能な玩具がいろいろと発売されてくれば、そしてそれらにKVMが搭載されていけばプログラミングが自由自在になります。

また、Java搭載だからこそ生まれてくる発想としては「ネットワーク対応」です。電子犬同士がネットワークで通信したり、ビデオの予約を電子犬に頼んだり.....?

コンピュータゲームの様相も変わってくるかもしれません。今でこそ1台のPCやゲーム機で複数のソフトを動かしていますが、KVMが搭載されたゲーム盤が安価に発売されれば、野球盤にサッカーゲームにボーリング.....。銃的を狙うおもちゃなどもJava対応なんてことになれば、子供部屋は再び昔のようにおもちゃの山になるでしょう。「テレビゲーム機ばかりに熱中して...」と嘆く大人たちも、ゲーム盤になれば懐かしがって文句も言わなくなるのでは?

子供達の創作意欲も、「組み込みJavaをいじる」ことで育っていくのかもしれない。学校教育でも、そんな勉強なら面白くて役立ちますよ

ね。先生方、ぜひお願いします。

さて、ちょっと脱線してしまいましたが、J2ME、KVMについての詳しい情報は、サンマイクロシステムズのサイトで知ることができます。日本語のサイトも用意されていますのでぜひご覧になってください。現在はまだ、実行できる携帯電話は発売されていませんが、とりあえず事前準備としてどうぞ。

J2ME、KVMについて

<http://www.sun.co.jp/software/consumer-embedded/kvm/>

Java搭載iモードについて

Java搭載iモードに関しては、9月20日と21日に開催されたゲームデベロッパー向けのカンファレンスCEDEC 2000で、今年12月に発売されると発表されました。

Javaのアプリケーションは10Kバイトで、アプリケーション用のデータ領域として各5Kバイトづつ用意されています。アプリケーションはJARファイルで圧縮されるとのことです。

また、携帯でのJavaの利用という点で、セキュリティが強化されているようです。Javaの実行環境と、メモリダイヤルなどのプライバシー情報が分離され、Javaを使ってこれらの情報にアクセスすることはできません。そのほかにも、さまざまなセキュリティ対策が行われているようです。

まだあまり詳細は明らかにされていませんが、本誌が発売されているころには詳しい仕様が公表されるということなので、興味のある方はNTTドコモのホームページをチェックしてください。



画面1 J2ME、KVMについてのWebサイト(サン)

プログラミング工房

先月号では、USBのデバイスドライバプログラミングの概要を説明した。今月号は、デバイスドライバのソースコードを実際にコンパイルしながら、最新のlinux-2.4.0カーネルプログラミングの世界を楽しんでみる。

第11回 カーネルのプログラミング(2)

文：藤沢敏喜
Text: Toshiki Fujisawa

カーネルソースの準備

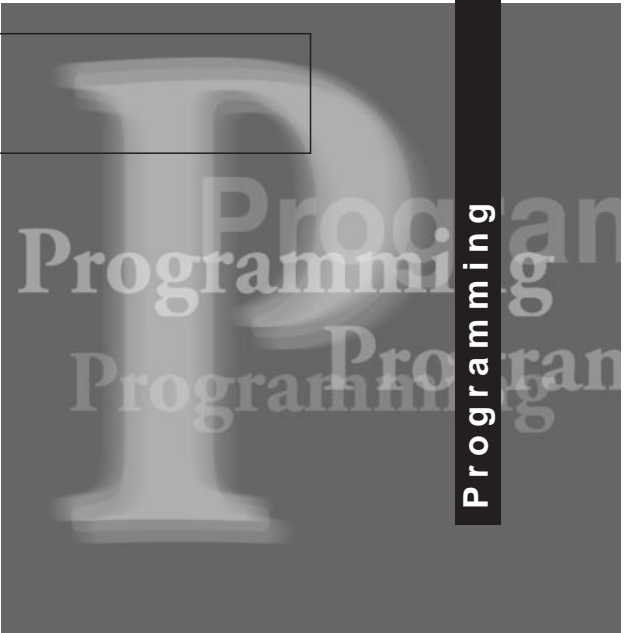
最初にカーネルのコンパイル方法について説明しよう。ここでは、linux-2.4.0-test5.tar.gzをVine Linux 2.0のフルインストール環境でコンパイルすることにする。

まず初めに、付録CD-ROMのプログラミング工房ディレクトリ(Linuxmag/Programming/)から、linux-2.4.0-test5.tar.gz、usb-xscsi-2.4.0-test5.patch(パッチファイル)、および今回のデバイスドライバ本体であるxscsi.cを、/usr/srcディレクトリにコピーする。

コピーが終わったら、ルート権限で次のコマンドを実行する。

```
# cd /usr/src
# rm linux
# tar zxvf linux-2.4.0-test5.tar.gz
# mv linux linux-2.4.0-test5
# ln -s linux-2.4.0-test5 linux
# patch -p0 < usb-xscsi-2.4.0-test5.patch
# cp xscsi.c linux/drivers/usb
```

ちなみに、今回のデバイスはUSBからSCSIへの変換(Xchange)を行うものであるため、デバイスドライバを「usb-xscsi」という名前にすることにした。



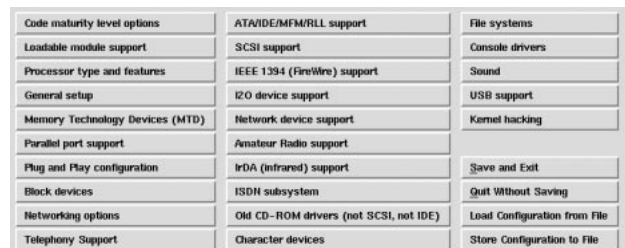
カーネルの設定

Linuxではカーネルのサイズをむやみに大きくしないため、必要でないデバイスはカーネルに組み込まないようにしている。そのため、どのデバイスを使うかの設定を行う必要がある。この設定を行うには、X上のktermなどから、

```
# cd /usr/src/linux ; make xconfig
```

を実行することにより、設定メニューを表示することができる(画面1)。

デフォルトでは、実験的なデバイスについては表示されないようになっているが、左上の「Code maturity level options」の中で、実験的なデバイスも選択できるように設定可能だ。今回作成したドライバも実験的なものなので、



画面1 make xconfigによるカーネル設定画面

これを有効にしておく必要がある。

次に、設定メニューの [USB support] を選択し、[support for USB] を有効にしてから、[UHCI(Intel PIX4,VIA, ...) support] を有効にする。そして、下のほうへスクロールして、先ほどのパッチファイルにより追加された [USB Xchange SCSI Adapter support (EXPERIMENTAL)] を有効にする。

今回のデバイスを使うには、デフォルトの設定のままにしておくほうがいいので、自分の環境で絶対に必要な部分だけを設定して、[Save and Exit] を選んで終了する。

LILOの設定

新しいカーネルの実験時には、カーネルパニックを起こしてOSが立ち上がらなくなることがよくある。したがって、必ず起動できる安全なカーネルを保存しておき、そのカーネルで立ち上げられるようにしておく必要がある。

このためには、/etc/lilo.conf をリスト1のように設定するとよい。

カーネルのコンパイルとインストール

カーネルのコンパイルは、上記の/etc/lilo.confの設定を

リスト1 /etc/lilo.conf

```
boot=/dev/hda2
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
append="apm=on"
default=linux

image=/boot/vmlinuz-2.4.0-test5
    label=linux
    initrd=/boot/initrd-2.2.14-1vl6.img
    read-only
    root=/dev/hda2

image=/boot/vmlinuz-2.2.14-1vl6
    label=safe
    initrd=/boot/initrd-2.2.14-1vl6.img
    read-only
    root=/dev/hda2

other=/dev/hda3
    label=freebsd

other=/dev/hda1
    label=dos
```

したあとで、

```
# cd /usr/src/linux ; make dep ; make install
```

を実行する。なお、上記のカーネル設定ではモジュールも組み込まれるので、

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

も行っておこう。あとは、リブートするだけで新しいカーネルが立ち上がるはずだ。

今回のデバイスドライバの実現方法

スキャナの場合、/dev/sg? という名前の SCSI Generic デバイスを用いてアクセスされることが多い。したがって、USB-SCSI の変換デバイスの場合は、この /dev/sg? をエミュレーションする形で実装するのがベストである。

このような SCSI のエミュレーションを行うデバイスの例としては、先月説明した USB ストレージクラスドライバがあげられる。

しかしながら、このドライバではエミュレーションを行うためにカーネルスレッドやセマフォなどをたくさん使っており、雑誌の記事で解説するには難しすぎる。

そこで、今回は SCSI の処理はドライバ内だけで行い、アプリケーションとは単純なデータの Read/Write だけを行うという、最も簡単な構成をとることにした。

その場合、スキャナをアクセスするアプリケーション側でも、このような Read/Write での入出力を行うことが要求されることになるが、今回実験に使ったフィルムスキャナ Konica Qscan 用の最新アプリケーション (xqscan-2.51) では、この入出力 (/dev/qscan-io) がサポートされている。しかし、これ以外のスキャナでは、SANE などのアプリケーション側の改造が必要である。

なお、このような構成ではアプリケーション側から SCSI-ID を受け渡すことができないので、SCSI-ID はカーネル内で固定的に決めることとした。今回は、2番の SCSI-ID を設定することにしたので、スキャナ側も SCSI-ID を2番に設定する必要がある。

メジャー番号とマイナー番号

linux-2.4.0からは、/devの構造が大きく変更されている

が、互換性のため伝統的なメジャー番号とマイナー番号によるアクセスもできるようになっている。たとえば、

```
% ls -l /dev/hda
```

を行うと、

```
brw-rw---- 1 root disk 3, 0 May 6 1998 /dev/hda
```

と表示される。この表示から、IDEディスクはブロックデバイス (b) であり、メジャー番号が3、マイナー番号が0であることがわかる。USBのメジャー番号は180に決められていて、マイナー番号は各デバイスごとに定められるようになっているが、ここでは実験のため、暫定的に128番をマイナー番号として使うことにした。

また、このデバイスはキャラクタデバイス (c) なので、デバイスファイルを作成するためには、たとえば、

```
# /bin/mknod /dev/qscan-io c 180 128
```

というコマンドを実行することになる。

USB デバイス接続と動作について

ここまで準備ができたなら、フィルムスキャナにUSB-SCSI変換ケーブルを接続したあとに、PCにUSBケーブルを差し込む。そして、dmesgコマンドを実行してみれば、usb-xscsiデバイスが認識されているはずだ。その状態でxqscan (ver2.51以上) を起動すると、フィルムのスキャンを行うことができる。

このデバイスドライバは、UHCIチップのUSBコントローラを用いた下記の3台のPCで、USBハブを使わずに直接PCのUSBコネクタに接続して、動作を確認してある。

- SONY VAIO PCG-Z505GR/K (RAM256Mバイト)
- SONY VAIO R-50 (RAM192Mバイト)
- 富士通 FMV DESKPOWER S-II 167 (RAM96Mバイト)

一方、OHCIチップを用いたマザーボードのPCと、PCIのOHCIチップUSBカードでも実験してみたが、カーネルのOHCIの処理部分に問題があるせいか動作しなかった。

なお、現在のドライバでは、先月紹介したロジテック社製以外のUSB-SCSI変換ケーブルを使うことはできない

Column

USB 開発用マシンの購入

USB開発用にもう1台マシンが必要になったが、自宅にある5台のマシンはどれもひどく古く、肝心のUSBコネクタはどのマシンにも付いていない。そこで、PCIスロットに増設するUSBカードを買ってきたのだが、購入したカードはチップがOHCIであるため、linux-2.4.0-test5と今回のデバイスの組み合わせではうまく動作しなかった (UHCIチップを搭載しているPCIカードを探して、秋葉原を徘徊したが見つからなかった)。

これらのマシンは古すぎて、マザーボードだけを置き換えるわけにはいかないので、この際新しいマシンを買うことに決めた。ライカのレンズなら何十万円出しても惜しくはないが、すぐにモデルチェンジしてしまうPCには、あまりお金をかけたくない。そこで、型落ちのVAIO-R50を友人経由で安く買うことにした。

このVAIO-R50には空きPCIスロットが1つしかなく、ネットワークカードを差すとSCSIカードを差すスペースがなくなってしまう。つまり、SCSI-USB変換アダプタのドライバを開発する動機がまたひとつ増えたことになるのだ。

ちょっと余談になるが、このPCにはMPEGエンコーダボードが内蔵されていて、テレビドラマなどの録画ができる。標準モードでも、テレビの画面に表示するには十分満足の画質で、オプションの赤外線リモコンを使うとまさに家電感覚で使える。

あまりに感激したため、速攻でPCショップへ走り、80Gバイトのディスクを買ってきて増設した。AWARD BIOSの33.8Gバイトの壁にぶちあたって、ハードディスクが認識できないというトラブルがあったが、BIOSをごまかすソフトをMBR (マスターブートレコード) へ入れて回避し、無事50時間の録画ができる状態になった。

読者のなかには流量の多いメーリングリ

ストやニュースをとりあえず購読して、あとで興味のある記事だけ見るという人も多いと思う。50時間の録画ができると、まさにこのような視聴形態になり、いままでとは世界が変わる。

こうなると、「FreeBSD / Linuxで全チャンネル24時間」を実現したくなるが、VAIOのMPEGボード解析は特許問題などがあり難しそうだ。

一方、NECからはUSB接続のMPEGエンコーダ装置が数万円で発売されている。USBバスアナライザがあれば、こちらのほうが近道かもしれない。USBの帯域はたかだか1Mバイト/秒なので、この装置を5個買って、PCIのUSBカードを4枚増設する。そして、複数のSCSIディスクを用意し、Linuxのような高性能OSを利用すれば、もしかしたら1台のPCで5チャンネル同時録画が可能ではないか……、と夢想している今日のごろである。

(コラムで紹介した3社の製品はすべてプロトコルが異なる)。

また、このドライバは、あくまでもデバイスドライバの解説を行う記事のサンプルプログラムとして書かれたものであり、その動作を保証することはできない。

したがって、USB-SCSI変換ケーブルやフィルムスキャナを入手する場合は、動かないことがあることを覚悟のうえで購入する必要がある。もちろん、このデバイスドライバについては、ケーブルやスキャナのメーカーもまったく保証していないので、本記事に関してメーカーに問い合わせを行うことは避けてほしい。

なお、kernel-2.4.0-test7での実験も行ったが、このバージョンのカーネルでは下層(UHCI)の通信がうまくいかないようで、正常に動作しなかった。USBはまだ開発中のデバイスであり、安定した動作はなかなか難しいようである。

デバイスドライバを追加する方法

さて、いよいよプログラムの解説であるが、まずはカーネルにデバイスドライバを組み込む方法について述べる。

自分が作成したドライバを、先ほどのカーネル設定画面で選択するようにするためには、`/usr/src/linux/drivers/usb/Config.in`というファイルに1行加えるだけである。

このファイルの書式はたいへん簡単であり、たとえば、73行目にある、USB FM radio デバイスは、以下のように定義されている。

```
dep_tristate ' D-Link USB FM radio support
```

```
(EXPERIMENTAL)' CONFIG_USB_DSBR $CONFIG_USB
$CONFIG_VIDEO_DEV
```

この行でシングルクォートで囲まれた文字列がカーネルコンフィグ (`make xconfig`) 時に表示される。また、`CONFIG_USB_DSBR` という文字列は後述のヘルプや Makefile での指定に用いられる。

さらに、このデバイスを使用するには、USBサポートと、VIDEO デバイスのサポートが有効になっている必要があるため、`$CONFIG_USB` と `$CONFIG_VIDEO_DEV` が定義されている。

以上の例から類推されるように、今回のUSB-SCSI変換デバイスの場合は、以下の行を追加すればよい。

```
dep_tristate ' USB Xchange SCSI Adapter support
(EXPERIMENTAL)' CONFIG_USB_XSCSI $CONFIG_USB
$CONFIG_SCSI
```

なお、今月号ではSCSIエミュレーションを行わないので、SCSIサポートが有効になっている必要はない。しかし後での拡張を考え、SCSIサポートが有効になっていないと、このデバイスが選択できないようにするため `$CONFIG_SCSI` も追加してある。

この1行を追加するだけで、カーネル設定時にこのデバイスの選択ができるようになるが、設定時のヘルプ画面に適切な情報を表示するためには、`/usr/src/linux/Documentation/Configure.help` というファイルのUSB Bluetoothの説明の次(10418行目)にリスト2のような数行の説明を追加するとよい。なお、このリストにある`CONFIG_USB`

リスト2 Configure.help

```
USB Bluetooth support
CONFIG_USB_BLUETOOTH
Say Y here if you want to connect a USB Bluetooth device to your
computer's USB port. You will need the Bluetooth stack (available
at http://developer.axis.com/software/index.shtml) to fully use
the device.

This code is also available as a module ( = code which can be
inserted in and removed from the running kernel whenever you want).
The module will be called bluetooth.o. If you want to compile it as
a module, say M here and read Documentation/modules.txt.

USB Xchange SCSI ADAPTER support
CONFIG_USB_XSCSI
ここに「USB-SCSI変換アダプタ」に関する説明を記述する(複数行でもかまわない)
```

_XSCSIは、先ほど指定した文字列である。

そして、この文字列を /usr/src/linux/drivers /usb/Makefileの95行目に、

```
obj-$(CONFIG_USB_XSCSI) += xscsi.o
```

という形で、加えることによりカーネルのメイクができるようになる。最後にデバイスドライバ本体(xscsi.c)を、/usr/src/linux/drivers/usb/ディレクトリに置けば完成である。

これらの、具体的な記述については、付録CD-ROMに収録したパッチファイルを参照してほしい。

デバイスドライバの構造

今回のデバイスドライバは、カーネルの中でどのようにして認識されるのだろうか？このあたりを理解するために

は、xscsi.cの最後のほうから順に見ていくとよい。xscsi.cにある最後の2行は、

```
module_init(usb_xscsi_module_init);
module_exit(usb_xscsi_module_cleanup);
```

となっていて、module_init、module_exitというマクロが使われている。ここで指定するのが、モジュールの登録と削除のときに呼ばれる関数である。ここでは登録時に、

```
int __init
usb_xscsi_module_init(void)
{
    return usb_register(&driver_table)<0 ? -1 : 0;
}
```

が呼ばれる。また、削除時には、

Column

USB-SCSI変換ケーブル

USBとSCSIの変換アダプタには、先月紹介したロジテック社製以外にも、アダプテック社から新しく発売されたUSB-Xchangeという製品がある。

この製品はSCSIのトップメーカーであるアダプテック社製ということもあり、どのような仕様であるかを調べてみたいという誘惑には勝てず、ついつい買いものカゴに入れてしまった(写真1)。

別のショップへ行くと、今度はMicrotech社のUSB-SCSI変換ケーブルが隅に1個だけ置いてあった。9000円近い価格でちょっと躊躇したが、秋葉原では物があるときに買わないと、売り切れてしまい後悔することが多い。これまでに何度もそういう経験をしたので、思いきってこれも買うことにした(写真2)。

さらに別のショップへ行くと、今度はXircom社のUSB-SCSI変換ケーブルが置いてあった。これは3900円と安かったこともあり、「毒食わば皿まで」の心境でレジに持

って行ってしまった(写真3)。

結局、気がついてみると、4種類のUSB-SCSI変換ケーブルを買ったことになり、財布がずいぶん軽くなってしまった。しばらくのあいだは胃袋を満たせない生活になったものの、この4種の製品を比べてみるといろいろと興味深く、知的好奇心のほうは十分に満たすことができた。

今月号ではこれらについて詳しく説明するスペースがないので、来月号でまた取り上げてみたい。

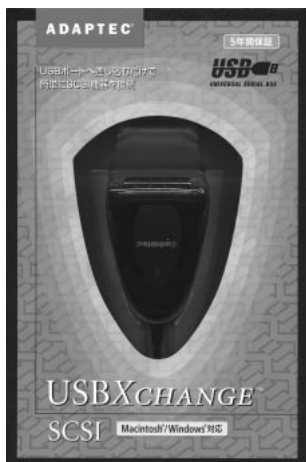


写真1 アダプテック製 USB-XChange SCSI



写真2 マイクロテック製 USB-SCSI変換ケーブル



写真3 ザーコム製 USB-SCSI変換ケーブル

```

void __exit
usb_xscsi_module_cleanup(void)
{
    usb_deregister(&driver_table);
}

```

が呼ばれることになる。このとき、usb_register 関数に渡される引数は、下記のような構造体となっている。

```

static struct usb_driver
driver_table = {
    "usb-xscsi",

```

リスト 3-1 xscsicのscsi_read関数

```

static unsigned int
scsi_read(struct file *file, unsigned char *buf, int len)
{
    unsigned char cdb_read[0x0c] = {
        /* 0x00 */ 0x08, /* SCSI READ_COMMAND */
        /* 0x01 */ 0x00,
        /* 0x02 */ 0x00, /* len[2] */
        /* 0x03 */ 0x00, /* len[1] */
        /* 0x04 */ 0x00, /* len[0] */
        /* 0x05 */ 0x00,
        /* 0x06 */ 0x00,
        /* 0x07 */ 0x00,
        /* 0x08 */ 0x00,
        /* 0x09 */ 0x00,
        /* 0x0a */ 0x00,
        /* 0x0b */ 0x00
    };

    int r;
    int sense_key;
    pdata_t *pdata;
    int rest_len, cur_len;

    cdb_read[0x02] = (unsigned char)( 0xff & (len >> 16));
    cdb_read[0x03] = (unsigned char)( 0xff & (len >> 8));
    cdb_read[0x04] = (unsigned char)( 0xff & (len >> 0));

    while( (r = send_cdb(file, cdb_read)) < 0 ){
        if( r != -EPIPE ){
            err_msg(_("ERR send_cdb EPIPE\n"));
            return r;
        }
        /* Now Endpoint was STALL, send clear feature */
        if( (r = send_clear_feature(file)) < 0 ){
            err_msg(_("ERR send_clear_feature\n"));
            return r;
        }
        if( recv_err_status(file) != err_status_read_retry ){
            err_msg(_("ERR!! not READ_RETRY\n"));
            return -EFAULT;
        }
        sense_key = scsi_request_sense(file);
        if( sense_key != SENSE_KEY_READ_RETRY ){
            err_msg(_("sense_key NG 0x%02x\n", sense_key));
            return -EFAULT;
        }
        err_msg(_("now retry scsi_read\n"));
    }
    /* リスト 3-2 へ続く */
}

```

ファイルアクセスのための構造体
 読み込むバッファ (ユーザー空間メモリ)
 読み込むバイト数
 読み込みを指示するCDBパケット
 読み込みを意味するコマンド番号
 デバイス情報を保持するプライベートデータエリア
 1回で読み込めるバイト数をセット
 読み込むべき残りのバイト数
 CDBをUSBへ送信する関数
 読み込むべきバイト数 (24ビット)
 USBエンドポイントがSTALL以外はエラー
 エンドポイントのエラーを回復
 送ったCDBコマンドの結果を取得
 SCSIプロトコルにおけるセンスキーを取得

```

device_probe,
device_disconnect,
{ NULL, NULL },
&usb_xscsi_file_operations,
USB_XSCSI_MINOR_BASE
};

```

この構造体では、このドライバの名称“usb-xscsi”や、USB ケーブルが抜き差しされたときに呼ばれる device_probe関数およびdevice_disconnect関数が定義されている。

また、アプリケーションからファイル関係のシステムコールが発行された場合に、それら进行处理するための関数群を定義した、usb_xscsi_file_operationsという構造体へのポインタも定義されている。そして、この構造体の最後のメンバ(USB_XSCSI_MINOR_BASE)が、上で説明したデバイスのマイナー番号である。

ファイルオペレーションの構造

上記の構造体中にあるusb_xscsi_file_operationsの中身は、下記のように定義されている。

```

static struct file_operations
usb_xscsi_file_operations = {
    owner:          THIS_MODULE,

```

```

read:             file_operation_read,
write:            file_operation_write,
open:             file_operation_open,
release:          file_operation_release,
};

```

たとえば、アプリケーションからopenシステムコールが発行されると、file_operation_open関数が呼ばれ、readシステムコールが発行されると、デバイスからの読み込みを行うためにfile_operation_read関数が呼ばれる。

open()システムコールの処理

アプリケーションプログラム側では、ファイルにアクセスする際に、

```
int fd = open("/dev/qscan-io", O_RDWR);
```

という記述でデバイスをオープンする必要がある。このopenシステムコール(関数)は、“/dev/qscan-io”のメジャー番号とマイナー番号から、適切なデバイスドライバを選択し、最終的に上で説明した構造体のfile_operation_open関数を呼ぶことになる。

そして、file_operation_open関数では、そのデバイスドライバで必要とするデータ領域の確保などを行う。

リスト3-2 xscsicのscsi_read関数(リスト3-1の続き)

```

pdata = (pdata_t *) file->private_data;
rest_len = len;
while( rest_len > 0 ){
    cur_len = ( MAX_TRANS_BYTES < rest_len )
              ? MAX_TRANS_BYTES : rest_len ;
    if( (r=recv_data(file, pdata->buf, cur_len)) < 0 ){
        err_msg(_("ERR recv_data r=%d\n", r));
        return r;
    }
    if( copy_to_user(buf, pdata->buf, cur_len) ){
        err_msg(_("ERR copy_to_user\n"));
        return -EFAULT;
    }
    buf += cur_len;
    rest_len -= cur_len;
}
return len;
}

```

デバイスにおけるプライベートデータエリア

アプリケーションから要求されたバイト数

残りの転送バイト数がある限りループする

1回に転送する量をMAX_TRANS_BYTES以下にする

USBからデータを受信する

カーネル空間からユーザー空間へコピー

転送した分だけポインタを進める

転送した分だけ減らす

今回転送が終わったバイト数

read()システムコールの処理

アプリケーションプログラムでは、デバイスからデータを読み込むときは、openシステムコールから返されたファイル識別子fdを用いて、

```
read(fd, buffer, length );
```

というようにして読み出しを行う。このreadシステムコールでは、fdの値によって適切なデバイスドライバを選択し、そのデバイスドライバのread関数を呼ぶ。つまり、今回の場合はfile_operation_read関数が呼ばれることになる。この関数の中では、リスト3-1に示すscsi_readという関数を呼んでいる。

USB-SCSI変換アダプタでは、PCからのSCSIパケットをそのままSCSIデバイスへ中継する。ここで送られるSCSIパケットはCDBと呼ばれ、中身はリスト3-1で定義されているcdb_readという12バイトの配列である。これを、send_cdb関数を使って、PCからUSB-SCSI変換アダプタへ送信している。

send_cdb関数はリスト4で定義されているが、この関数ではusb_control_msgというUSBの下層関数を用いて、コントロールエンドポイントへ送信される。

ここで、もし送信時にエラーが起きたらsend_cdb関数の戻り値が負になるが、この場合はUSBのエンドポイントがSTALLしているので、CLEAR_FEATUREを送ってエラーから回復するなどの処理が必要になる。

これを行っているのが、リスト3-1にあるsend_clear_feature関数である。このあと、SCSIのREQUEST SENSE処理を行い、再度send_cdbを試みている。

無事にCDBが送出されると、次はデータの読み込みだが、ここではrecv_data関数を使って、アプリケーションから指定されたバイト数のデータの読み込みを行う。なお、ここで読み込むのはカーネル空間内のメモリであるので、copy_to_user関数を用いて、ユーザー空間内のメモリにコピーしている。

次号について

今回はカーネルのデバイスドライバのしくみとその動作について解説したが、デバイスドライバを作成することは、open / close / read / writeの関数を作成するという単純なことであるのがご理解いただけたのではないかなと思う。

USBデバイスドライバについては、引き続き次号でもとりあげてみたい。

リスト4 xscsi.cのsend_cdb関数

```
static int
send_cdb(struct file *file, unsigned char *cdb)
{
    struct usb_device *udev;
    unsigned int pipe;
    __u8 requesttype = 0x21;
    __u8 request = 0x00;
    __u16 value = 0x0000;
    __u16 index = 0x0000;
    __u16 size;
    int timeout = 60*HZ; /* 5sec */
    int r;

    udev = ((pdata_t *) (file->private_data))->dev;
    pipe = usb_sndctrlpipe(udev, 0);
    size = 0x0c; /* always 12bytes */
    for(;;){
        r = usb_control_msg(udev, pipe, request, requesttype,
                           value, index, cdb, size, timeout);
        if( r >= 0 ){
            return r;
        }
        if( r != -EPIPE ){
            err_msg(_("ERR send_cdb r=%d\n", r));
            return r;
        }
        if( send_clear_feature(file) != 0 ){
            err_msg(_("ERR send_cdb clear feature\n"));
            return -1;
        }
    }
}
```

USBケーブルの情報が入っている構造体

パイプの識別番号

送信用コントロールパイプの識別番号を取得

コントロールパイプ(エンドポイント0)ヘデータを送信

このUSBデバイスでは、CDBは必ず12バイトでやり取りされる

送るデータの先頭アドレス

送るデータのバイト数

エンドポイントのSTALL状態を解除する

Ruby で行こう

とうとう出ました、1.6.0。なんだかすごく待たされたような気がします。今回は、この1.6.0について、ダウンロードからインストールの方法、そしてプログラミング例、バージョンアップによる変更点まで、詳細に解説します。

第11回 新たなる未知へ

文：赤松智也

Text: Tomoya Akamatsu

インストール

Debian、Red Hat、SuSE、FreeBSD portなどには、Rubyのパッケージが用意されていますが、Perlのようにほとんどのディストリビューションに最初から付属するようになるまでにはしばらくかかりそうです。

最新のRuby 1.6.0は、付録CD-ROMに収録されています。また、以下のURLからソースアーカイブを入手することもできます。

```
ftp://ftp.netlab.co.jp/pub/lang/ruby/1.6/ruby-1.6.0.tar.gz
```

netlab.co.jpは、Rubyの作者まつもとさんの勤務先です。以下のミラーサイトも利用できます。

```
ftp://ftp.iiij.ad.jp/pub/lang/ruby/  
ftp://ftp.TokyoNet.AD.JP/pub/misc/ruby/  
ftp://ftp.krnet.ne.jp/pub/ruby/  
ftp://ftp.nctu.edu.tw/computer-languages/ruby/
```

ソースを入手すれば、以下のステップによりインストールできます。

```
$ tar zxvf ruby-1.6.0.tar.gz  
$ cd ruby-1.6.0  
$ ./configure  
$ make  
<rootになる>  
# make install
```

標準では、rubyは/usr/local/bin/rubyにインストールされます。

```
$ type ruby  
ruby is /usr/local/bin/ruby  
$ ruby -v  
ruby 1.6.0 (2000-09-11) [i686-linux]
```

各ステップについてもう少し解説しましょう。

configure
configureはautoconfツールが生成するシェルスクリプトで、各プラットフォームでのコンパイルに必要な情報を収集します。configureの実行経過を画面1に示します。いろいろな項目をチェックしているのがわかります。

コンパイル時の設定を変更するためには、configureスクリプトにオプションを指定します。configureに指定するオプションのうち、重要なものは以下のとおりです。

--prefix=PREFIX

インストール先のディレクトリを指定します。デフォルトは/usr/localです。プログラムはPREFIX/bin、ライブラリはPREFIX/lib/ruby/1.6以下にインストールされます。多くのパッケージでは/usrが指定されてコンパイルされています。root権限でインストールできない場合には、PREFIXとして自分のホームディレクトリを指定するワザが使えます。その場合、\$HOME/binをPATHに含めることを忘れずに。

--with-default-kcode=CODE

デフォルトの文字エンコーディングを指定します。CODEの部分にはutf8、euc、sjis、noneのいずれかを指定します。デフォルトはnoneですが、日本語をよく使うならeucを指定するのが便利かもしれません。

--enable-shared

ダイナミックリンクライブラリlibruby.soを用意します。ダイナミックリンクライブラリを使うと、インタプリタのバイナリサイズが劇的に小さくなり、メモリ効率が若干向上します。

--with-sitedir=DIR

非標準のライブラリをインストールするディレクトリを指定します。デフォルトは、PREFIX/lib/ruby/site_rubyです。

--with-search-path=DIR

ライブラリのロード時の検索パスに追加するディレクトリを指定します。

そのほかの "--with" で始まるオプションは、拡張ライ

```
$ ./configure
creating cache ./config.cache
checking host system type... i586-pc-linux
checking target system type... i586-pc-linux
(中略)
checking whether OS depend dynamic link works... yes
creating config.h
updating cache ./config.cache
creating ./config.status
creating Makefile
creating ext/extmk.rb
```

画面1 configure 実行経過

ブラリのコンパイル時に使われます。標準添付の拡張ライブラリが受け付ける主なオプションは、以下のとおりです。

--with-xxx-include=DIR

拡張ライブラリxxxのコンパイル時にヘッダファイルを検索するディレクトリを指定します。DIRがコンパイラの-Iオプションに指定されます。

--with-xxx-lib=DIR

拡張ライブラリxxxのリンク時に、ライブラリを検索するディレクトリを指定します。DIRがリンカの-Lオプションに指定されます。

--with-xxx-dir=DIR

拡張ライブラリxxxがコンパイル・リンク時にアクセスするディレクトリを指定します。DIR/includeがコンパイラの-Iオプションに、DIR/libがリンカの-Lオプションに指定されます。

Debian や Kondara では、“ --with-dbm-include=/usr/include/db1 ” を指定する必要があるようです。これはndbm.hのあるディレクトリを指定します。

また、インストールされているTclのバージョンによっては、“ --with-tcl-include=/usr/include/tcl8.3 ” のようにヘッダファイルtcl.hの存在するディレクトリを指定する必要がある場合もあります。そのほか、ディストリビューションによっては、gdbmやreadlineについても同様の指定が必要な場合もあります。

拡張ライブラリのコンパイルがうまくいかないようでしたら、Rubyのインストール後に、

```
$ cd ext/dbm
$ ruby extconf.rb
```

とすると、ちゃんとヘッダやライブラリを検出できているか試してみることができます。拡張ライブラリに対する "--with" で始まるオプションはextconf.rbに渡すこともできます。たとえば、

```
$ ruby extconf.rb --with-dbm-include=/usr/include/db1
```

のように指定します。

```
make
```

実際のコンパイルとリンクを行います。念のためテストもしておきます。

```
$ make test
```

いろいろな試験をして、問題がなければ、“test succeeded”と出力されます。

```
make install
```

標準のインストール先にインストールするには、root権限が必要です。configureでprefixを自分の書き込み権のあるディレクトリに指定していれば、root権限は不要です。必要な権限が用意できたら、makeを使ってインストールします。

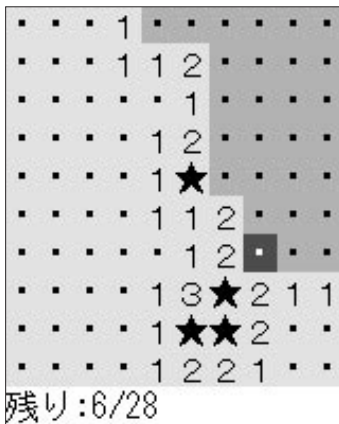
これでインストールは完了です。sampleディレクトリの中のキャラクタ版マインスイーパー(mine.rb)を実行してみます(画面2)。

hjklのvi風キーバインドでカーソルを動かし、mキーで爆弾を置き、スペースキーでマスを開きます。終了するためにはqキーを使います。mine.rbはわずか175行のプログラムですが、なかなか燃えますね。

プログラムしてみよう

例題としてURLチェッカを作ってみます。このプログラムは、コマンドライン引数で指定したURLのページが存在するかどうかをチェックします。なんらかの理由でページにアクセスできなかった場合には、理由も含めて表示します(リスト1)。

実行例は以下のようになります。



画面2 マインスイーパー

```

$ ruby urlcheck.rb http://localhost/index.html
http://localhost/index.html does exist
$ ruby urlcheck.rb http://localhost/noexist.html
http://localhost/noexist.html does not exist
404 Not Found
$ ruby urlcheck.rb http://www.ruby-lang.org/ja
http://www.ruby-lang.org/ja does not exist
301 Moved Permanently

```

ブラウザで表示できる Moved Permanently なども、存在しないことにしてしまうのはダメですが、とっかかりとしてはこんなものでしょう。HTTPクライアントがこれだけ(アクセスは実質1行)で書けるのは、気分が良いですね。net/httpライブラリについては来月(か、その次)にももう少し詳しく解説します。

変更点はどこ?

1.4系から1.6系へは、それほど影響の大きな変化はないようです。1.4系で動いていたプログラムのほとんどは、1.6系でも問題なく動くでしょう。

しかし、1.4.0が1999年8月に出てから1年以上、1.5系として並行に開発されてきたので、細かな変化はたくさんあります。1.5系の開発中のバグ修正などは1.4系にも取り込まれましたが、1.6.0にしか含まれない修正点も数多くあります。この連載では、1999年5月号から「今月のRuby

リスト1 簡易URLチェッカ(urlcheck.rb)

```

#!/usr/bin/env ruby

require 'net/http'

# URLを解析(正規表現による簡易版)
url = ARGV[0] || "http://localhost/"
if %r!http://(.*)?(?::(\d+))?(/*.*)! =~ url
  host = $1
  port = Integer($2) if $2
  path = $3
end

begin
  Net::HTTP::new(host, port).get(path)
rescue => err
  printf "%s does not exist\n", url
  print err.message, "\n"
else
  printf "%s does exist\n", url
end

```

1.5」で、1.5系の変化をウォッチしてきました。1.5系での変化を示すまとまった文書はいまだに出ていませんから、価値あるコラムであったと自画自賛しています。

無事1.6.0もリリースされたことですし、少々重複しますが、ここで1.6.0の変更点をまとめてみることにしましょう。1.6.0のドキュメントやソースを1.4.6のものと比較すると、かなり差があります。見落としがありそうですが、それでも80項目はありました。本文中では主なものだけを説明します。それ以外のものは表1に簡単にまとめておきました。数は多いですが、ほとんどは上位互換か、ささいなものです。

非互換なもの

1.6.0の変更点のうち、場合によってはスクリプトの修正が必要になる非互換なものは以下のとおりです。

Objectのinitializeが0個の引数を取る

以前は任意個の引数を受け付けていました。これにより、newが引数を受け取る場合には、initializeをきちんと再定義する必要があります。いかげんにしていたプログラムは、書き換えが必要になります。

変更点	変更点
クラス変数	initializeの仕様の明確化
Objectのinitializeが0個の引数を取る	多重代入がto_aを使わない
block_given?(iterator?)の新名称	エラーメッセージがちょっと親切に
例外メッセージで非表示文字をエスケープ	RangeError例外
rescue => 変数	rescue修飾子
メソッド本体のrescue節	より柔軟な文法
ScriptError例外	全オブジェクトがfreeze可能
-eで一時ファイルを使わない	-Xが-Cに変更
環境変数RUBYOPT	環境変数RUBYLIB_PREFIX
セキュリティ機能のデバッグ	getrlimitでスタックの最大値を計算する
GCの頻度の変更	nil,true,falseに特異メソッドが定義できる(FixnumとSymbolには相変わらずできない)
cloneとdupの仕様の統一	dyna_var構造体をリサイクル
require 'final'が組み込み	Thread.joinがなくなった
ThreadGroupができた	Thread.startが引数を取るようになった
Threadのpriority	スレッド対応デバッガ
selectがbusy waitしないように	waitpidがbusy waitしないように
新正規表現“%G”、“(?>...)”	regexの“%<”と“%>”が正式になくなった
regexが割り込みを理解する	正規表現オプション“/p” “/m”
\$KCODEでNONEがデフォルトに(以前はEUC)	offsetがマルチバイト文字の途中で検索する
コンパイル時文字列結合	文字列操作のパフォーマンス向上
slice、slice!	kconvがString#kconv,tojis,toeuc,tosjisを定義する
String#index == で\$=の値を使う	String#dupのlazy copy
Kernel#scan	pack/unpackでw(BER)が使えるように
Bignumがpackできるように	pack指定子が増えた
Array#concatが配列のみを受け取る	Array#+が配列のみを受け取る
Array#replaceがto_aを使わない	Array#at、first、last
Array#filter -> Array#collect!	%w()の実装でエスケープできるようになった
ブロックなしのcollect	grepの戻り値
Numeric#truncateが追加	Float nan? infinite? finite?
moduloが導入された	%, divmodの挙動が明確化された
Hashの値としてのnil	再帰的ワイルドカード展開
fnmatch(わたなべ版)の採用	Dir::globがブロックを受け取る
newはブロックを取らず、openは取る	printfで\$による順番の指定
バッファリングされたIOはsysreadがエラー	IO#pid
IO#seekの引数が省略可能(SEEK_SET)	IO#rewindで行番号が0に
Proc#callがnilと引数なしを区別する	statの戻り値は構造体からFile::Statオブジェクトに
rand関数の引数が省略可能	シンボルのオブジェクト化
dateライブラリがdate2と統合	Time::local、Time::gmでマイクロ秒が指定できる
Time::utc、Time#utc、Time#utc?	Process#wait2、waitpid2
rb_scan_argsでブロックを取り出せる	

表1 1.6.0での変更点

多重代入がto_aを使わない

以前は多重代入は右辺が配列でなければ、to_aメソッドで配列化してから代入を行っていました。1.6.0では、

- ・ to_aryメソッドがあればそれを使う
- ・ なければそのまま先頭の左辺式に代入

というルールになりました。右辺として構造体などを使っていた場合には修正が必要です。

ScriptError例外

LoadError、NameError、SyntaxErrorが、StandardErrorのサブクラスではなくなりました。これらは、クラス名を指定しないrescueでは捕捉されません。

Thread.joinがなくなった

以前から、使うと警告が出ていました。

Hashの値としてのnil

ハッシュの値としてnilが使えるようになりました。特別扱いが減ったのは良いことですが、nilをセットすることで削除していたプログラムは書き換えが必要です。

Arrayクラスの型変換

Arrayクラスの暗黙の型変換が減りました。暗黙の型変換はわかりづらいエラーの原因になります。

- ・ Array#concatが配列のみを受ける
 - ・ Array#+が配列のみを受ける
 - ・ Array#replaceがto_aを使わない
 - ・ Array#filter -> Array#collect!
- filterメソッドはまだ使えますが、警告が出ます。

Enumerable#indexがなくなった

Enumerableは順序がなく、indexメソッドに意味がないので削除されました。Arrayクラス以外のindexメソッドを使っているプログラムはないと思われるので、この変更はほとんど影響はないでしょう。

文法の変更と強化

クラス変数

定数をクラス変数代わりに使うというのが、長らく

Rubyのイディオムでしたが、とうとうクラス変数が導入されました。これでプログラムがもっと自然に記述できます。クラス変数は名前が@@で始まる変数です。ただ、クラス変数はアクセス範囲が限定されるだけで、本質的にはグローバル変数のようなものですから、使い過ぎは禁物です。

rescue => 変数

rescue節の末尾に「=>変数」という形で、\$!で得られる例外情報を変数に格納できるようになりました。これで、記号変数の使用頻度がさらに減ります。

```
begin
  ...
rescue ScriptError => err
  STDERR.print err, "\n"
  exit
end
```

この「=>」の部分に何が来るかは、メーリングリストでinはどうだとか、intoだとか、かなりモメたのですが、結局は「=>」に決まったようです。

rescue修飾子

rescueの修飾子形式が使えるようになりました。

codeA rescue codeB

1.4系では必ず、

```
begin
  codeA
rescue
  codeB
end
```

と書く必要があったので、例外処理がかなり手軽にできるようになります。ただ、修飾子形式では例外種別を指定できませんので、いつもStandardErrorのサブクラスすべてを捕捉してしまいますし、ensureを指定することもできません。そういうことが必要な場合には、やはり以前からの形式を使ってください。

メソッド本体のrescue節

メソッドの定義本体は、全体としてbegin..endでくくられているとみなして、直接rescueやensureが書けるようになりました。

```
def foo(a, b, c)
  ...
rescue NameError
  STDERR.print "unknown name\n"
  exit
ensure
  a.close
end
```

より柔軟な文法

以下のような表現が許されるようになりました。

```
a,b = File::split "foo/bar"
a = yield b
[1,2,3].sort do|a,b| b<=>a end.reverse
[File::dirname "foo/bar"]
```

これらはいずれも1.4系ではエラーになります。あらゆるパターンを網羅して、だんだんシンタックスエラーが出ていくような気がします。

コンパイル時文字列結合

文字列リテラルを複数並べると、コンパイル時に結合されるようになりました。

```
print "foo" "bar" "\n"
```

ANSI Cなどにもある機能です。マクロ機能のないRubyでこれができる何が嬉しいのかはさっぱりわかりませんが、とにかくできるようになりました。長い文字列を複数行に渡って記述する場合には、文がそこで切れないことを明示するため、行末にバックスラッシュが必要です。

```
print "this is a " \
      "very long " \
      "line which " \
      "spans many " \
      "lines\n"
```

“%w()”の実装でエスケープできるようになった

1.4系では、%wはsplitを使って空白で分割していたので、分割結果に空白を含むことはできませんでした。1.6.0では、バックスラッシュを使って空白をエスケープできます。

```
%w(foo\ bar baz) # => ["foo bar", "baz"] 1.6.0
                  # => ["foo\\", "bar", "baz"] 1.4系
```

スレッド機能強化

Thread.startが引数を取るようになった

スレッドに値を引き渡すため、startクラスメソッドに引数を与えることができるようになりました。引数はそのままスレッドのブロックパラメータとして渡されます。以前の仕様で、ローカル変数を使って値を引き渡すと、スレッドが値を取り出す前に別スレッドが変数の値を書き換える場合がありましたが、引数を使って値を引き渡すことにより、この問題を避けることができるようになりました。

スレッド対応デバッガ

デバッガdebug.rbが、スレッドを操作することができるようになりました。

親切設計

たとえば、メソッドの引数に間違えてローカル変数以外のもものを指定した場合、以前なら単なる文法エラーですが、1.6.0では“formal argument cannot be a global variable”のように、より明確なエラーを報告します。このようなメッセージの変更がほかにも数カ所あります。

initializeの仕様の明確化

1.6.0では、すべてのクラスがnewメソッドを使ってオブジェクトを作るときに、initializeメソッドを呼びます。以前は、組み込みクラスはほとんどinitializeメソッドを呼びませんでした。このことにより組み込みクラスのサブクラスを作りやすくなります。

バッファリングされたIOはsysreadがエラー

IOクラスのsysreadメソッドはreadシステムコールを

直接呼ぶため、そのIOがすでにバッファリングを行う読み出しを行っている、思ったような入力を得られません。1.6.0では、sysreadの呼び出し時にバッファをチェックし、誤ってバッファリング呼び出しがすでに行われている場合にはエラーにします。

再帰的ワイルドカード展開

zshで使えるような再帰的ワイルドカードが使えるようになりました。これにより「カレントディレクトリ以下の拡張子がrbのファイル」を「**/*.rb」のようなパターンで指定できます。

printfで\$による順番の指定

printfのフォーマット指定子の%とフォーマット文字の間にn\$ (nは数字) を挟み込むことによって、引数の順番どおりでなく、n番目の引数をフォーマットします。これは、gettextなどで引数の順番と出力の順番が変わってしまう場合などに活用します。

cloneとdupの仕様の統一

1.4系では「cloneはオブジェクトのできるだけ完全な複製を作り、dupは完全でない可能性のある簡易な複製を作る (cloneと同じものを返す可能性もある)」という比較的いいかげんな仕様でしたが、1.6.0では「cloneは特異メソッドも含めたできるだけ完全な複製を返し (従来どおり) dupは特異メソッドなどは含まない内容だけの複製を返す」という仕様に統一されました。

Proc#callがnilと引数なしを区別する

1.4系ではProc#callで手続きオブジェクトを呼び出すとき、

```
proc = Proc.new{|a,|...}
proc.call(nil)
proc.call()
```

を区別できませんでした (両方とも引数なしとみなされてエラーになる)。1.6.0では、両者を区別しますから、前者はエラーになりません。

pack指定子が増えた

必ずしも「親切」ではないかもしれませんが、pack/unpackで指定できるフォーマット文字が増えました。増えたものは表2のとおりです。

環境変数RUBYOPT

環境変数RUBYOPTにオプションを指定しておくと、インタプリタ実行時にそのオプションも指定したとみなしてくれるようになりました。たとえば、インタプリタのデフォルト文字コードをいつもシフトJISにしたい場合には、

```
export RUBYOPT="Ks"
```

とします。-Tオプションでセキュリティレベルを1以上に指定すると、RUBYOPTからの読み込みは行いません。

パフォーマンスの改善

1.6.0では、仕様には影響を与えないものの、パフォーマンスが改善されています。全体としてはさほど変化したようには思えませんが、極端に遅かったいくつかの状況に対応したようです。

String#dupのlazy copy

文字列をdupするとき内容のコピーを行わず、本当にコピーが必要なときまで実際にはコピーしません。文字列のdupは内部的にもかなり数多く行われますので、コピーの削減は実行速度に影響を与えます。

sub、gsub、trなど!の付かないメソッドは、内部でdupを使って文字列の複製を作っています。これらのメソッドを使って、かつ実際の更新がさほど行われない場合には、かなり実行速度が速くなります。しかし、複製のほとんどが更新されるような場合には、結局内容のコピーも行われるので、さほど変わらないようです。

文字列操作のパフォーマンス向上

dupのほかにも、gsubなどの文字列操作が若干高速化

文字	意味
D	d(double)の別名
e	リトルエンディアンのfloat
E	リトルエンディアンのdouble
F	f(float)の別名
g	ビッグエンディアンのfloat
G	ビッグエンディアンのdouble
M	Q-encodingされた文字列
U	UTF-8文字列
w	BER (Basic Encoding Rule) 圧縮された整数
Z	ASCII文字列 (nullを詰める/unpackでは末尾のnullを削除する)
-	sSILLに続く場合には機種固有のフォーマットで格納

表2 pack/unpackで新たに指定可能になったフォーマット文字

されているようです。ただ、普通の使い方ではほとんど差は出ないようです。

dyna_var 構造体をリサイクル

ブロック内ローカル変数を実装しているdyna_var構造体を、明示的に開放するようになりました。これにより、GCの呼び出し頻度が下がって、パフォーマンスが改善されます。

GCの頻度が変更

1.4系では、大量のオブジェクトを割り当てる場合などに、GCが必要以上に頻繁に呼び出されることがありましたが、GCの起動条件の変更によりこの問題は回避されました。

getrlimitでスタックの最大値を計算する

1.6.0では、getrlimitシステムコールを使って得たプロセスが使えるスタックの最大値を基に、再帰呼び出しの限界チェックを行います。

正規表現の強化

新正規表現 “¥G”、“(?>...)”

正規表現が若干増えました。

“¥G”は、前回のマッチの末尾にマッチします。連続してマッチを行うgsubやscanにおいて有効です。

“(?>...)”は、独立正規表現です。この範囲内でマッチしたパターンはバックトラックしません。ですから、“(?>a*)ab”は決してマッチしません。というのも“a*”がすべてのaの並びを「飲み込んで」しまうので、abにマッチできないからです。“(?>)”で囲まれていなければ、バックトラックによりaを1つ戻った地点でマッチできます。

regexの“¥<”と“¥>”が正式になくなった

Emacs由来の“¥<”(単語の始まりにマッチ)と“¥>”(単語の終わりにマッチ)がなくなりました。ドキュメントにこれが使えると載ったことはないと思うので、影響はないでしょう。この変更により、Perlと同じ「記号(非アルファベットの)メタ文字にはバックスラッシュが付かない」という統一ルールができました。

正規表現オプション“/p” “/m”

“/p”を使っていると警告が出ます。ほとんどの場合は

“/m”への置き換えだけで済むはずですが。ただ、“/m”は“^”や“\$”が改行を無視しないので、それぞれ“¥A”、“¥Z”に書き換えてください。以前のpオプションでは、“^”は文字列の先頭に、“\$”は文字列の末尾にマッチしていました。mオプションでは、これらは文字列中の改行ともマッチします。

regexが割り込みを理解する

1.4系では正規表現エンジンが割り込み禁止状態で動作していたので、複雑にバックトラッキングするパターンマッチでは、Ctrl + Cキーを押しても停止せず、困ったことになっていました。1.6.0では、そのような場合でもCtrl + Cキーなどが有効です。

KCODE_NONEがデフォルトに(以前はEUC)

configureで指定する\$KCODEのデフォルトがNONEになりました。海外でのRubyユーザーが増えてきたことが理由のようです。必要に応じて明示的に-Kオプションか\$KCODEを指定するか、コンパイル時に--with-default-kcodeで指定してください。

バグフィックス

バージョン1.5を開発している間に発見されたバグのほとんどは、同時に1.4系にも反映されていました。そのため、1.4.6から1.6.0に移行することで減るバグはそれほどありません。しかし、\$SAFEを使ったセキュリティ関係のバグは、1.4系には一切フィードバックされませんでしたから、セキュリティ機能を使いたい場合には移行を考えたほうがよいでしょう。まつもとさんによると「1.4系のセキュリティ機能は穴が多くて使いものになりません。1.6.0なら大分マシではないでしょうか」とのことでした。

まとめ

今回は1.6.0リリースを受けて、インストールと移行についてまとめてみました。自画自賛ではありますが、バージョン間の差分については、Rubyの歴史上最も詳しい文章ではないでしょうか。これらの調査はたいへんでした。皆さんが活用して下さることを望みます。

次回からは、1.6.0を使って、具体的なプログラムを開発して行きたいと思います。

(編集部注：原稿執筆後に1.6.1がリリースされました。)

[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。初回である今回は、シェルスクリプトとは何かといった知識や、基本的なスクリプトの書き方、スクリプトに実行パーミッションを与える方法などについて説明した後、ktermを利用して別ウィンドウでlessを実行するスクリプトを作成する。

第1回 シェルスクリプトの基本

文：大池浩一
Text:Koichi Oike

Linuxを含むUNIX系OSでは、ユーザーの入力を解釈してコマンドを実行するプログラム「シェル」が特別な位置を占めている。UNIXの標準シェルであるsh（Bourneシェルとも呼ばれる）をはじめ、csh、tcsh、ksh、zshなどさまざまなシェルが作られており、各ユーザーの好みに応じて選択できる。Linuxでは、shとの互換性を保ちつつ、コマンドライン編集などの便利な機能を追加したbash（Bourne Again Shell）が標準シェルとして使われている。

GNOMEやKDEなどのグラフィカルな環境が普通になった現在でも、最低限のshの知識は必要だ。というのも、個人のPCでLinuxを使用する場合、管理者としての仕事もある程度こなさなければならず、そうした場面ではshの知識が求められるからだ。たとえば、LinuxやXの起動時の処理は、shの「シェルスクリプト」に記述されている。何らかの問題が起こった場合、シェルスクリプトを読めれば、問題解決の糸口を見つけられるはずだ。

シェルスクリプト（単に「スクリプト」と書くこともある）は、コマンドラインにキー入力する代わりに、実行する内容をあらかじめファイルに列挙したものだ。長いコマンドラインや、複数のコマンドの組み合わせなどをいちいちキー入力することなく実行できる。

DOSの知識がある人は「バッチファイルに似ている」と思われるかもしれない。基本的には同じ概念と言ってもよいが、シェルスクリプトの場合、ファイルの有無など各種の条件判断や、複数のファイルを順番に処理する繰り返し

処理、数値や文字列を保存するシェル変数など、バッチよりプログラム言語としての側面が強く、それだけ柔軟で強力な処理が可能だ。

また、自分で作成したスクリプトを、他のコマンドと同様に実行できるのも大きな特長だ。複数のコマンドを組み合わせたり、条件判断などのシェルの機能を利用するだけで、新たな機能を持つコマンドを簡単に作成できる。つまり、C言語などの知識がなくてもプログラミングを楽しめるわけだ。

この連載ではbashを利用して、いろいろなシェルスクリプトを作成する。最初はごく簡単なスクリプトから始め、しだいにプログラムの処理を行うスクリプトへと発展させて行こう。基本的にはsh互換の機能のみを利用し、bashの拡張機能を利用すると便利な場合は、その後で説明を加えることにする。

最初のシェルスクリプトを作る

シェルスクリプトには、コマンドラインには出てこない注意すべき点がある。また、作成したスクリプトを他のコマンドと同じように実行するには、ファイルのパーミッションやコマンド検索パスの変更が必要だ。

今回は、わずか3行の短いスクリプトを作成し、スクリプト特有の注意点と、他のコマンドと同じように実行できるようにするための操作を順を追って説明する。操作に関

しては、次回以降は特に説明しないので、今回しっかりマスターしておきたい。

シェルスクリプトに何を書くか

基本的にはコマンドラインと同じように、実行するコマンドをそのまま書けばいい。スクリプトの1行がコマンドラインの1行に対応し、複数のコマンドを列挙すると上の行から順に実行される。

ここで作成するスクリプトは、ktermなどの端末画面に「最初のスクリプト」という文字列と、現在の日時の2行を表示するという簡単なものだ。文字列の表示には、bashの組み込みコマンドであるecho、日時の表示にはコマンドdateを利用する。コマンドラインなら、それぞれ、

```
$ echo '最初のシェルスクリプト'
$ date
```

とすればいい(「\$」はシェルのプロンプト)。これをシェルスクリプトにすると、

```
#!/bin/sh
echo '最初のスクリプト'
date
```

となる。Emacsなどのテキストエディタを使って上記の内容のテキストファイルを作成し、「hoge」というファイル名で保存しよう。

スクリプトhogeの2～3行目は、コマンドラインとまったく同じなので説明の必要はないだろう。問題は、1行目に「#!/bin/sh」という見なれない行があることだ。これは、「このファイルは/bin/shにより実行されますよ」ということを表しており、スクリプトの1行目に必ず書かなければならない。また、「#!」の前後や間にはスペースやタブを入れてはいけない。

スクリプトを利用するコマンドは、シェル以外にもPerlやAWK、Rubyなど数多くあるが、どのコマンドのスクリプトでも1行目は「#!」で始まり、それに続けて実行するコマンドが書かれている。たとえば、Perlの場合は「#!/usr/bin/perl」といった具合だ。

シェルスクリプトの場合、実行するコマンドは/bin/sh(つまり、ディレクトリ/binにあるsh)である。lsの-lオプションを使って、

```
$ ls -l /bin/sh
```

とすると、ファイル名の部分が「/bin/sh ->bash」と表示されるはずだ。これは、/bin/shがbashのシンボリックリンクであることを意味する。つまり、/bin/shには実体はなく、その代わりにbashが使われるということだ。もちろん、シェルスクリプトを実行する際にもbashが使われる。なお、shとして起動されたbashの動作は、通常の場合と細かな点で違いがあるのだが、ここでは触れない。

この節での要点をまとめると、

- ・シェルスクリプトの1行目には「#!/bin/sh」と書く
- ・2行目以降には実行するコマンドを列挙する

ということになる。

シェルスクリプトを実行するには
作成したシェルスクリプトを実行するには、

- (a) bash (sh) の引数としてスクリプトを指定する
- (b) スクリプトを実行する組み込みコマンドを使う

という2種類の方法がある。

(a)の方法は、コマンドラインから新たにbash(あるいはsh)自体をコマンドとして起動し、その引数としてス

Column

「#!」を解釈するのはシェル?

スクリプトの1行目に書かれた「#!」は、bashなどのシェルが解釈して、それに続くコマンドを実行しているように思えるが、答えは「No」だ。

実際には、Linuxカーネル内でコマンド実行処理を司る部分(execveシステムコール)が処理している。

このシステムコールは、ファイル先頭の数バイト(「マジックナンバー」と呼ばれる部分)からファイルの種類を判断する。「#!」で始まる場合はスクリプトであると見なし

て、それに続くコマンドを実行するわけだ。それでは、bashなどのシェルはこの行をどのように解釈しているのだろうか。シェルに限らず、多くのスクリプト言語では、「#」で始まる行はコメントとして扱われる。つまり、bashから見れば「#!/bin/sh」は単なるコメントに過ぎない。

クリプトを指定するというものだ。たとえば、カレントディレクトリにあるスクリプト hoge を実行するには、

```
$ bash hoge
```

とすればいい(「sh hoge」でも可)。その結果、

最初のスクリプト

```
Sun Oct 7 14:00:48 JST 2000
```

のように、「最初のスクリプト」という文字列と、現在の日時が2行にわたって表示される。

一方、(b)の方法は、スクリプトの内容を読み込んで実行する組み込みコマンド source (あるいは「.」) を利用し、引数としてスクリプトを指定するというもの。たとえば、カレントディレクトリにある hoge を実行するには、

```
$ source hoge
```

とする(「. hoge」でも可)。このスクリプトでは、得られる結果は(a)の場合とまったく同じだ。

2つの方法の違いは、スクリプト内に記述されたコマンドを実行するために新たなシェル(「サブシェル」と呼ばれる)が起動されるか、現在のシェルがそのまま使われるかという点にある(図1)。この違いが異なる結果を生む場合もあるのだが、今回は取り上げない。

bash にしる source にしる、いちいちコマンド名を指定するのは面倒なので、実際のシェルスクリプトでは、(a)の応用として、

(c) 通常のコマンドと同じようにスクリプトを起動する

という方法をとることが多い。X を起動する際に使う startx や、Netscape Communicator を起動する際に使う netscape などのコマンドも実はこうしたシェルスクリプトだ。

たとえば、シェルスクリプト hoge を実行するには、ただ単に、

```
$ hoge
```

とだけ入力すればいい。

ただし、この方法を利用するには、シェルスクリプトに

「実行」パーミッション(使用許可)を付け、なおかつコマンド検索パス(PATH)に含まれるディレクトリにシェルスクリプトを置く必要がある。続いては、その手順について説明しよう。

UNIX系OSでは、各ファイルに対して、

- ・読み込み(ファイルの内容を見る)
- ・書き込み(ファイルの内容を変更する)
- ・実行(ファイルをプログラムとして起動する)

という3種類のパーミッション(使用許可)情報を、

- ・所有者(オーナー)であるユーザー
- ・所有者と同じグループに属するユーザー
- ・その他のユーザー

という3カテゴリーのユーザー別に設定できる。

シェルスクリプトを他のコマンドと同じように直接コマンドラインで実行できるかどうかは、そのユーザーに対する「実行」パーミッションがスクリプトに付けられている

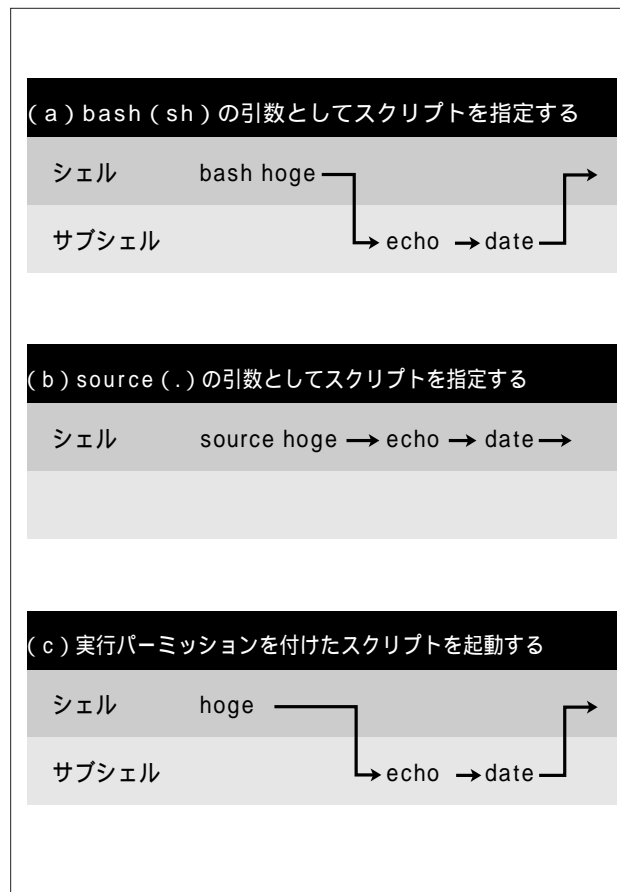


図1 シェルスクリプトの実行方法

かどうかによる。

ファイルのパーミッション情報は、lsを-lオプション付きで実行したときの先頭の記号列で確認できる。たとえば、先ほど作成したhogeの場合は以下ようになる。

```
$ ls -l hoge
-rw-r--r-- 1 daichi daichi 45 Oct 07 14:00 hoge
```

「rw-r--r--」のうち、最初の「-」は通常のファイルであることを示し、残りの記号は3個ずつに分かれてユーザーカテゴリーごとのパーミッションを表す。初めの「rw-」は所有者に対して「読み込み、書き込み可」であること、次の「r--」は所有者と同じグループのユーザー、最後の「r--」はその他のユーザーに対して、いずれも「読み込みのみ可」であることを意味する。

このように、エディタなどで設定したテキストファイルには、「実行」パーミッションは付いていない。これが付いていると、中身が単なるテキストの場合でも実行できてしまうからだ（当然、起動時にエラーになる）。

シェルスクリプトの場合は、ファイルのパーミッション情報を変更するコマンドchmodを使って、「実行」パーミッションを付ける必要がある。chmodによるパーミッショ

ンの付加や削除は、数字や記号列を使った面倒な指定が必要だが、この場合はごく単純に、

```
$ chmod +x hoge
```

とすればいい。「+x」は、3カテゴリーのユーザー全てに対して「実行」パーミッションを付けることを意味する。この操作を行ったあとで、先ほどと同じように、

```
$ ls -l hoge
-rwxr-xr-x 1 daichi daichi 45 Oct 07 14:00 hoge
```

とすると、パーミッション情報が「rwx」「r-x」「r-x」に変わり、どのカテゴリーのユーザーに対しても「実行可」に変更されていることがわかる（図2）。

まとめると、

- ・シェルスクリプトを直接実行可能にするには「chmod +x」を使って実行パーミッションを付ける

ということになる。

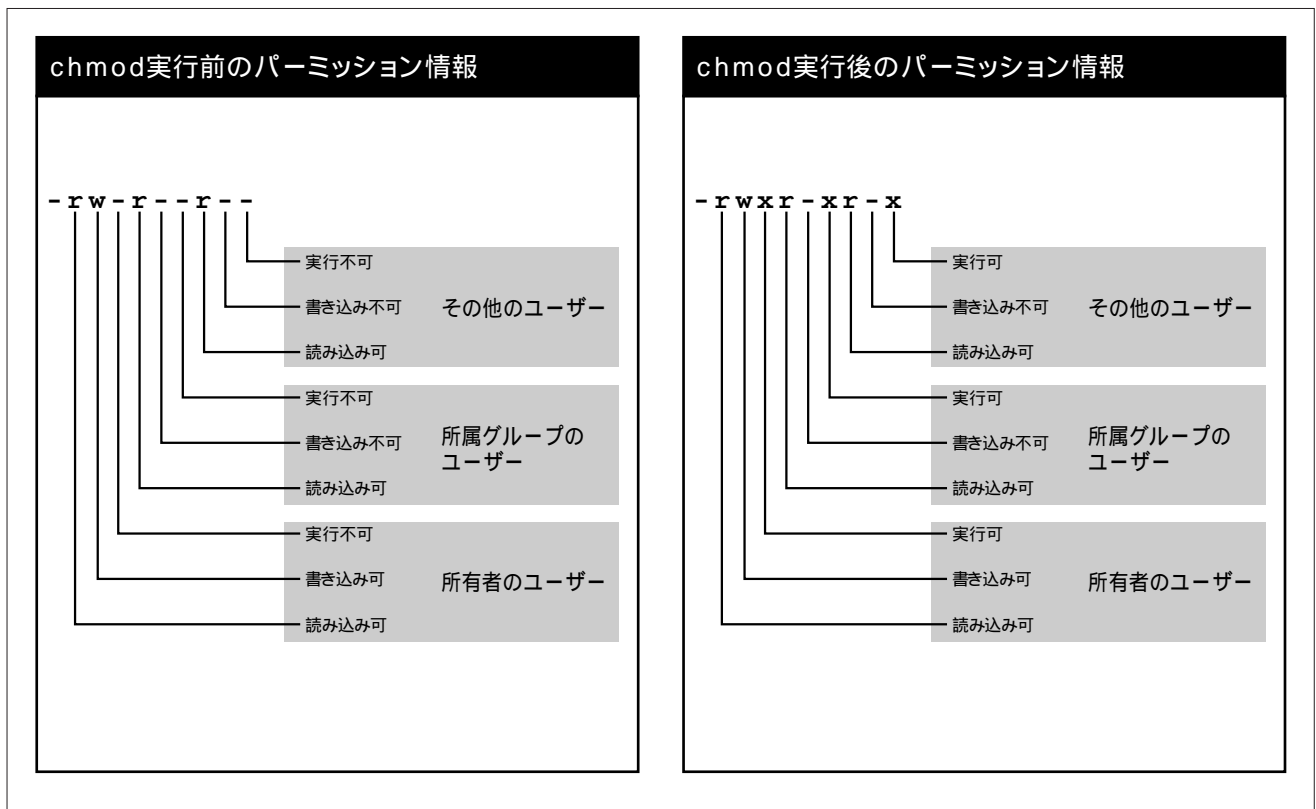


図2 シェルスクリプトを直接実行可能にする

シェルスクリプトをどこに置くか
 気の早い読者の中には、直接実行可能になったシェルスクリプトを実行しようとして、

```
$ hoge
bash: hoge: command not found
```

とエラーをくらった人もいるのではないだろうか。

実は、bashを含むUNIX系OSのシェルでは、カレントディレクトリのコマンドを、コマンド名だけ指定して起動することはできないのだ。コマンド名だけ指定した場合には、「コマンド検索パス」と呼ばれるディレクトリの集合からコマンドが検索される。通常、一般ユーザーが作業するようなディレクトリはコマンド検索パスには含まれていないので、上の例のように「コマンドが見つからない」というエラーが表示されるわけだ。

それでは、カレントディレクトリのコマンドを起動するにはどうするかというと、ディレクトリも含めてコマンドを指定すればいい。コマンドにディレクトリ部分が含まれる場合は、コマンド検索パスは利用されないからだ。たとえば、カレントディレクトリが/home/daichiなら、

```
$ /home/daichi/hoge
```

とすることでhogeを実行できる。

また、UNIX系OSのファイルシステムでは、「.」でカレントディレクトリを表すから、これを利用して、

```
$ ./hoge
```

としてもいい。カレントディレクトリのファイルを実行する方法はよく使われるので覚えておこう。

しかし、いちいち「./」を付けなければならないのでは、「他のコマンドと同じように直接実行できる」と言うには語弊がある。次の作業は、「コマンド検索パスに含まれるディレクトリにhogeを格納する」ことだ。

まずは、現在のコマンド検索パスを確認してみよう。コマンド検索パスは、PATHという環境変数（詳しくは次回以降で取り上げる）に設定されている。PATHの内容を表示するには、組み込みコマンドechoを使って、

```
$ echo $PATH
/bin:/usr/bin:/usr/bin/X11:/usr/local/bin
```

とすればいい。PATHの前に「\$」を付けると、「PATHの内容」という意味になる。

PATHの内容は、「:」（コロン）を区切りとしたディレクトリの集合だ。上の例では、「/bin」「/usr/bin」「/usr/bin/X11」「/usr/local/bin」がコマンド検索パスに含まれており、これらのディレクトリにあるコマンドは、コマンド名だけの指定で実行できる。検索は先頭のディレクトリから順に行われ、たとえ同名のコマンドが複数のディレクトリに存在するとしても、先に見つかったものだけが実行される。

なお、PATHの内容は、一般ユーザーとスーパーユーザー（root）では異なる。また、ディレクトリの順番などは、Linuxのディストリビューションによって異なることがあるので、自分の結果が上の例と違っていても気にすることはない。

今回作成したスクリプトを、コマンド検索パスに含まれるディレクトリのいずれかに移動（あるいはコピー）すれば、コマンド名だけを指定して実行できるようになるわけだが、ひとつ問題がある。一般ユーザーでは、上の例で挙げたようなディレクトリにファイルを書き込むことができないのだ。

これは、lsにldオプションを付けて、それぞれのディレクトリのパーミッション情報を見ればわかる。たとえば、/binの場合、

```
$ ls -ld /bin
drwxr-xr-x  2 root  root  2048 Jul 26 04:26 /bin
```

のようになる。なお、-dオプションを付け忘れると、/bin自身の代わりに、/binに含まれるファイルの一覧が表示されてしまうので気を付けよう。

「drwxr-xr-x」のうち、最初の「d」はディレクトリであることを示す。ユーザーカテゴリーごとのパーミッションを見ると、初めの「rwx」は所有者（root）に対して「読み込み、書き込み、実行可」であること、次の「r-x」は所有者と同じグループのユーザー、最後の「r-x」はその他のユーザーに対して、いずれも「読み出し、実行可」であることを意味する。

このようなディレクトリのパーミッション情報は、ファイルの場合と違って、そのディレクトリに格納されているファイルへのアクセスを制御する。

・読み込み（ディレクトリ内のファイル情報を調べる）

- ・書き込み(ディレクトリ内でファイルの作成・削除を行う)
- ・実行(ディレクトリ内の「検索」を行う)

実行パーミッションの「検索」とは、そのディレクトリにcdしたり、ファイルの一覧を取得したり、ディレクトリ内のファイルを実行することを意味する。

一般ユーザーは、/binに対しては「その他」のカテゴリに属するので、パーティションは「読み出し、実行可」だ。つまり、/binに含まれるファイルをlsで表示したり、/binに含まれるコマンドを実行したりすることはできるものの、/binにファイルを作成したり、削除したりすることはできないのだ。

自分のPCでLinuxを使っている場合は、スーパーユーザー (root) のパスワードを当然知っているわけだから、いったんrootになって/binや/usr/binにシェルスクリプトを作成することは可能だ。しかし、こうしたディレクトリにシェルスクリプトを置くということは、すべてのユーザーがそのシェルスクリプトを実行できるということにほかならない。未完成のスクリプトや、今回のようにあまり意味のないスクリプト、あるいは注意して使わないと危険なスクリプト(たとえば、「カレントディレクトリを丸ごと消去する」といったもの)などは、こうした共通のディレクトリに置くべきではない。

もちろん、みんなの役に立つスクリプトはこの限りではない。ただし、Red Hat系ディストリビューションでは、/binや/usr/binはRPMパッケージが利用するディレクトリなので、独自に作成したスクリプトは置かないほうがいい。こうした場合は、(rootになった状態で)/usr/local/binに移動・コピーするといいたいだろう。

一方、一般ユーザーが自分でしか使わないようなスクリプトは、各ユーザーの「ホームディレクトリ」に適当なサブディレクトリ(binなど)を作って、そこに置くといい。ホームディレクトリは、ユーザーがログインした直後のカレントディレクトリで、Linuxの場合は「/home/ユーザー名」であることが多い。

ディレクトリの作成にはコマンドmkdirを使用する。たとえば、binというディレクトリを作るには、カレントディレクトリがホームディレクトリ状態で、

```
$ mkdir bin
```

とすればいい。

続いて、スクリプトhogeをbinに移動しよう。hogeは

ホームディレクトリにあるだろうから、ファイルの移動を行うコマンドmvを利用して、

```
$ mv hoge bin
```

とする。

最後の作業は、コマンド検索パスに、このディレクトリを追加することだ。ただし、Red Hat系ディストリビューションでは、最初からこのディレクトリがコマンド検索パスに含まれていることがあるので、その場合は以下の作業をする必要はない。

コマンド検索パスにホームディレクトリのbinを追加するには、

```
$ PATH=$PATH:~/bin
```

とすればいい。「」(チルダ)は、各ユーザーのホームディレクトリに置換される記号だ。たとえば、ホームディレクトリが「/home/daichi」の場合、「/bin」は「/home/daichi/bin」になる。

なお、こうしたPATHの変更は一時的なもので、現在のシェルを終了すると失われてしまう。恒久的な変更には、ログイン時に実行されるスクリプト「.bash_profile」で設定する必要がある(.bash_profileについては次回以降で取り上げることにしよう)。

以上の作業により、シェルスクリプトhogeは、

```
$ hoge
```

として、直接起動できるようになった。

この節での要点をまとめると、

- ・カレントディレクトリのシェルスクリプトを実行するには、「./ファイル名」とする
- ・個人で使用するスクリプトは、ホームディレクトリのbinに格納するといいい
- ・コマンド検索パスの末尾に、ホームディレクトリ下のbinを追加する

ということになる。

今月のスクリプト

後半は、既存のスクリプトの内容を解説する「スクリプトを読む」と、与えられたテーマを実現するスクリプトの作成手順を説明する「スクリプトを書く」で構成する。今月の「スクリプトを読む」は、

gzip で圧縮されたファイルの内容を less で閲覧するスクリプト「zless」

で、スクリプト実行時のコマンドラインの内容（引数）をスクリプト内で参照する方法を説明する。

一方、「スクリプトを書く」のほうは、

指定したファイルを別ウィンドウ（kterm）のless で閲覧するスクリプト「wless」

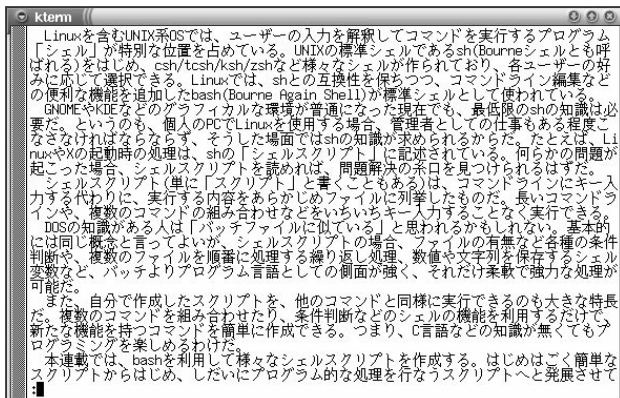
を作成する。どちらのスクリプトも、内容は2行だけなので、理解するのはそれほど難しくないだろう。

スクリプトを読む

zlessは、gzipのパッケージに付属するシェルスクリプトで、gzipで圧縮されたテキストの内容をlessで閲覧するために使われる。

たとえば、hoge hoge.txt というテキストファイルを、

```
$ gzip hoge hoge.txt
```



画面1 圧縮されたテキストの内容をlessで閲覧

として圧縮すると、hoge hoge.txt.gz というファイルが作られる。圧縮されたhoge hoge.txt.gzはテキストではなくなるわけだが、zlessを使うと、

```
$ zless hoge hoge.txt.gz
```

として、圧縮前の内容をそのまま閲覧できる（画面1）。

zlessの内容はわずか2行からなる（リスト1）。1行目の「#!/bin/sh」は、前半で説明したようにシェルスクリプトに必須の1行だ。実際に実行されるのは2行目のみということになる。

2行目の内容は、パイプ（|）によって2つのコマンドが接続されている。前半では、圧縮されたテキストの内容を出力するコマンドzcat、後半では、そのテキストの内容を閲覧するためのlessがそれぞれ起動される。どちらのコマンドもフルパス（/bin/zcat、/usr/bin/less）で記述されているのは、実行時のシェルのPATHの内容に影響されずに、確実にコマンドを起動するためだ。

ここで注意すべき点は、zcatが表示するファイル名の部分に「\$@」と書かれていることだ。これは、「スクリプト実行時のコマンドライン引数を、それぞれ二重引用符（"）で囲って並べたもの」に置換される。たとえば、

```
$ zless foo.gz 'hoge hoge.gz'
```

とすると、スクリプト中の「\$@」は、実行時に「"foo.gz" "hoge hoge.gz"」に置換される。両側の二重引用符は、スペースをファイル名の一部に含むファイルを正しく処理するために必要だ。これがないと、ファイル名の一部であるスペースを、シェルが区切り文字と間違えてエラーの原因となる。上の例だと、「hoge」と「hoge.gz」が別々のファイルとして処理されてしまうのだ。

要点をまとめると、

- ・スクリプト中で、実行時のコマンドライン引数すべてを参照する場合は「\$@」を使う

となる。

リスト1 スクリプト「zless」の内容

```
#!/bin/sh
/bin/zcat "$@" | /usr/bin/less
```

スクリプトを書く

zless を読んで経験を生かして、今度は指定したファイルを別ウィンドウ (kterm) の less で閲覧するスクリプト wless を書いてみよう。

まず、kterm のウィンドウを開くと同時に、指定したコマンドを実行するには、-e オプションに続けて実行したいコマンドとその引数を指定する。たとえば、

```
$ kterm -e less hogehoge.txt
```

とすると、kterm のウィンドウが開き、そこで less が起動されて hogehoge.txt の内容がウィンドウ内に表示される。less を終了すると、kterm のウィンドウも自動的に閉じる。なお、kterm のオプションをさらに指定する場合は、必ず -e オプションより前で指定すること。さもないと、起動するコマンド (上の例では less) のオプションと区別できなくなってしまうからだ。

ところで、上の例だと、別ウィンドウが開いている間は、元のコマンドラインはプロンプトが現れず、次のコマンドを実行できない状態になっている。そこで、

```
$ kterm -e less hogehoge.txt &
```

と末尾に「&」を付けて、バックグラウンドジョブとして kterm を起動しよう。これで、元のコマンドラインでもすぐにプロンプトが表示されて、続けてコマンドを実行できるようになる。

kterm のウィンドウのサイズは、起動時に -geometry オプションで指定可能だ。具体的には、ウィンドウ内部の表示領域の桁数、小文字のエックス (x)、行数の順番で指定する。たとえば、

```
$ kterm -geometry 80x25 &
```

とすると、80 桁 × 25 行の表示領域を持つ kterm のウィンドウが開く。

また、kterm のウィンドウの文字や背景の色は、それぞれ -fg、-bg オプションで指定できる。色は、RGB の数値による指定も可能だが、代表的な色の名前による指定を使っ

たほうが簡単だ。色の名前は、/usr/X11R6/lib/X11/rgb.txt に書かれている。たとえば、

```
$ kterm -fg black -bg white &
```

とすると、文字が黒、背景が白の表示の kterm のウィンドウが開く。

以上の内容を踏まえて、作成したスクリプト wless をリスト 2 に示す。1 行目は例によって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 行目だけだ。

まず、kterm のオプションにより、表示領域が 80 文字 × 25 行、文字が黒、背景が白のウィンドウが開き、less が起動される (画面 2)。kterm や less をフルパスで指定するのは、zless と同じ理由からだ。

また、less が表示対象とするファイル名の部分には、zless の場合と同様に「\$@」を記述して、スクリプト起動時のコマンドライン引数すべてを less で表示するようにしている。もちろん、bash にはコマンドライン引数を個別に参照する方法も用意されているのだが、それについては次回以降で説明することにしよう。

最後に、末尾に「&」を付けてバックグラウンドジョブとして kterm を起動していることに注意。これを忘れると、wless を実行中は元のコマンドラインにプロンプトが現れなくなってしまう。

ちなみに、「/usr/bin/less」の部分を変えると、指定したコマンドのマニュアルを別ウィンドウの man で参照するスクリプトになる。



画面 2 テキストの内容を別ウィンドウの less で閲覧

リスト 2 スクリプト「wless」の内容

```
#!/bin/sh
/usr/X11R6/bin/kterm -geometry 80x25 -fg white -bg black -e /usr/bin/less "$@" &
```


知ると知らぬでは大違い

Emacs はじめました

第8回 アプリケーション アラカルト

都合3回にわたって紹介してきた電子メールというテーマの主旨は、ほかのツールでできることをEmacsを使うことでより便利にしようというものでした。今回はそれをさらに推し進め、Emacsの中からテキストの編集以外のいろいろなことをしてみます。「えっ、Emacsでそんなこともできるの？」とびっくりするかも。

文：佐々木太良

Text：Taroh Sasaki

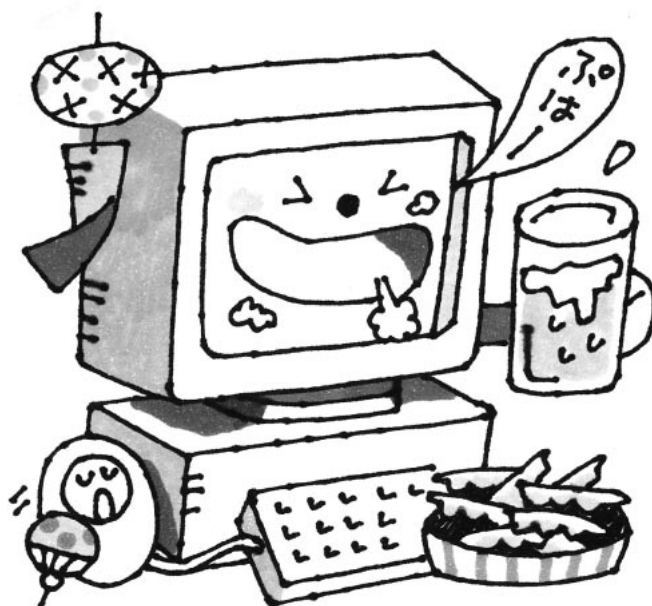


Illustration : Manami Kato

はじめに

Emacsには、外部のプログラムを駆動する能力、つまりEmacsからプログラムになにかデータを渡したり、処理結果を受け取る能力があります。また、離れた場所にあるコンピュータとのあいだでTCP/IPを使った通信機能を利用すれば、Emacsにいながらにしてほかの機能を使うことができます。

今回紹介する機能のなかには、「なに？ どうしてこれをEmacsから使わなきゃならんの？」と思われるものもあるでしょう。しかし、メリットも多いのです。

まず第一に、新しい端末(xtermなど)を開かなくてよいということがあげられます。きょうびX Window Systemが普及しているので、まさかコンソールでEmacsを使っているからほかのコマンドが使えないということはないでしょうが(Xがなくなつて、C-zで一時停止すればいいんですよね.....)新しいktermを開くためにコマンドを入力したりマウスを操作するのが嫌になっちゃうくらい仕事に脂が乗っていることもあるでしょう。EmacsのウィンドウにXのカーソルを置いたままで1日を過ごすつもりでなければ、真のEmacserとはいえませんが、というのは冗談ですが、なるべくウィンドウを切り替える手間を減らして、同じような操作でいろいろなことができれば仕事はかどると思いませんか。

次に、Emacsは元々テキストエディタですから、実行結果を加工して次のコマンドに渡したり、あるいは文書の一部として利用したり(もちろんメールでやり取りしたり)ということも楽勝なわけです。Xを利用しているとマウスによるコピー(左ボタンドラッグ)&ペースト(中ボタンクリック)も使えますが、これによる機能は限られていたり、時として意図しないところで行が分断されることがあります。

最後に日本人にとっては何よりも、日本語入力が簡単で賢く、自分に合ったカスタマイズができるEmacsのかな漢字変換(Wnnやかんな)が使えることは大きな魅力であり、この点だけでもEmacsを外部のツールのフロントエンドとして使う価値があります。

何はともあれ、一度はこれらの機能を試してみて、自分のライフスタイルに合っているかどうか確認してみる価値はあるでしょう。たまに便利な機能を知らずに苦労している若者を見かけて「こーゆーことができるんだよ」と教えてあげると「知らなかった.....」(以下絶句128秒、頭の中は走馬灯)とあぜんとしてしまう人もいますよ。ですから。

Emacsでシェル

まずは、Emacsからシェルを試してみましょう。これはEmacsがバッファ内にタイプした文字をふだんお使いのシェルに渡して、結果をまたバッファ内に返すものです。



画面1 シェルモードの利用

最近のシェル (bash、tcsh など) は履歴機能が充実しているので、以前の入力を再利用するだけならならずともこういった機能は必要ありません。ただし古典的なシェルでは履歴機能がほしいことがあります。筆者はいまだにシェルスクリプトはshで書くことが多いのですが (どんなOSでも動くように.....ってほんとかな) shの動作を確かめていると履歴機能が使えなくていららざる場合があります。しかし、標準のシェルが古典的なshとなっているOSや、Windows環境などでも、EmacsさえあればEmacs流の履歴編集ができて楽です。

さて、Emacsからシェルを利用するのに使うバッファはシェルモードと呼ばれますが、このモードはEmacsに標準で装備されているので、インストールの必要はありません。M-x shellとしてみましよう。* shell *というバッファが開いて、タイプしたシェルコマンドが実行されます。Emacsにはできないことをシェルコマンドで実行させて、結果をテキストに貼り込むのも楽勝です (画面1)。

履歴を遡って再実行するときには、C-c C-p (次に行くのはC-c C-n) で編集もできますし、Enter (C-m) をタイプすると該当行のコマンドがシェルに渡されて、再実行した結果がバッファの末尾に追加されます。バッファの末尾にコマンドをコピーするだけならC-c Enterとしてからゆっくり編集することもできます。tcshやbashのように入力行だけが遡って表示されるのではないので、場合によってはそれより快適です.....といいたいところですが、その前にちょっとした工夫が必要です。ここまで述べたことを試してみるとわかりますが、プロンプトが再入力の一部とみなされるので、プロンプトが悪さをしないように、変数shell-prompt-patternにプロンプトのパターンを入れておいてください。この設定値には正規表現も

コマンド	説明
C-c C-c	コマンドの実行を中断 (シェル環境での^Cと同等)
C-c C-o	1回ぶんの実行結果を消去

表1 シェルモードでのキー操作 (抜粋)

使えます。emacsに、

```
(setq shell-prompt-pattern "tcsh> ")
```

などと書いておけば、ふだんのプロンプトがそのまま使えます。Emacsのシェルモードでだけ実行させたいシェルの初期設定ファイルは、shの場合は /.emacs_sh (通常のshの初期設定ファイル.profileの後ろに読まれる) cshの場合は /.emacs_csh (.cshrcの後ろに読まれる)のようになります。したがって「ふだんのプロンプトじゃ、Emacsの中から使ったときうとうしい」っていうんで、.emacs_cshの中に、

```
set prompt=""
```

と書いておくと、これはこれでC-c C-pが誤動作したり、人間が表示を見誤ったりする原因になります。

```
set prompt="tcsh> "
```

くらいにしておくのが「吉」でしょう。ただし最初1回だけは、通常使用しているプロンプトが表示されてしまうため、リターンを空打ちしてから使い始めるとよいでしょう。このほか、シェルモードでよく使われるキー操作を表1にまとめておきます。

GNUのツール群のなかには、tcsh以外にもこれと同じような編集機能を備えたものがたくさんあります。GNUでは、コマンド行から面倒なコマンドを打ち込むようなツールが共通に使えるように、入力行の編集や履歴機能をサポートするありがたいライブラリ (readline) を公開しています。そのため、GNUのツール群の恩恵を多分に受けているLinuxでは、かなり古典的なコマンドでも「^Pをタイプすれば直前に入力したものを呼び出せる」のです。Emacsとは直接関係はありませんが、Emacsなら覚えておいて損はない操作でしょう。



次は遠くのコンピュータを使ってみます。「TELNETが

あるじゃないの」ってそのとおりなんです(泣) 要するにローカルコンピュータのシェルに対してシェルモードがあったように、TELNET (rlogin) に対しても telnet モード (rlogin モード) があるのです (画面2)。

この副産物として、遠方のコンピュータのシェルがろくろくセットアップされていなくても、Emacs 流のヒストリ編集のお世話になることが.....というのは、もう言わなくてもわかりますね。

使い方は M-x telnet または M-x rlogin とするだけです。リモート側のコンピュータがパスワードを聞いてくる局面では、ちゃんとかぼくバックなしになり、パスワードが表示されないようになるという気の効きよう。ただし、ダイヤルアップなどで使用してやり取りに時間がかかる場合、画面がにやにや動くのがちょっと気になるかもしれません。使用できるコマンドとキー操作はおおむねシェルモードと同じですので、いろいろ試してみてください。

ファイルを転送する

遠くのコンピュータにあるファイルをやり取りしたいとき、どうしますか? Emacs には FTP に相当する ftp-mode というコマンドもあるにはあったのですが (最近の Emacs では使えないものもあるようです) 転送したファイルをあとから編集することも多いもの。こんなとき、ange-ftp (Emacs19/Emacs20) や efs (XEmacs21) というマクロを使えば便利です。これらのモードも、Emacs19 以上ならデフォルトで組み込まれているはずで

ので、あとは皆さんが使い方を覚えるだけです。

ange-ftp や efs は、ローカルにあるファイルシステムとリモートにあるファイルシステムを「透過的に」使えるようにという発想で誕生した Emacs のマクロです。

たとえば find-file (C-x C-f) のときミニバッファに、

```
Find file: /hoge:~/moge.txt
```

とか、

```
Find file: /hoge@oresama:~/moge.txt
```

と入力してやることで、ホスト hoge の moge.txt (後者の場合は、hoge 上のユーザー oresama として) を編集できるようになります。

パスワードの入力を求めたりログインする過程がユーザーからなるべく見えないように行われるので、

1. リモートホストからファイルを一時ファイルに FTP して
くる
2. 一時ファイルを編集する
3. セーブするときは一時的ファイルをリモートホストに FTP
で転送する

という操作が自動的に行われることを意識しておかないと戸惑うでしょう。

たとえば上の例で /hoge@oresama: /mo までタイプし

Column

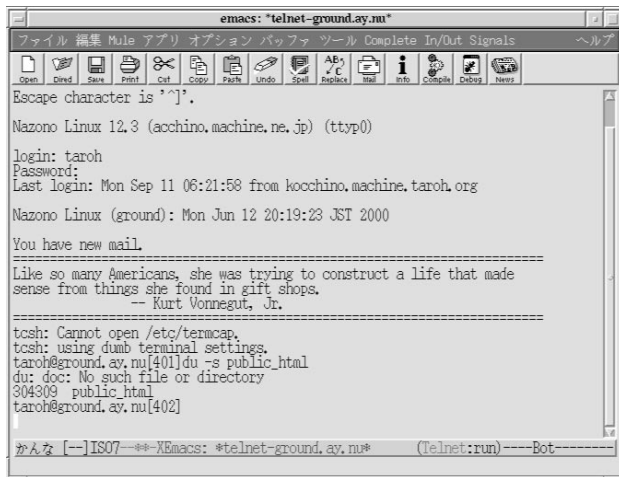
Emacs のシェルモードと tcsh の emacs モード

シェルモードは使われなくなってきたと書きましたが、これは bash や tcsh などの近代的なシェルが、Emacs の真似をしてヒストリ編集ができるようになってきたことが大きな理由でしょう。

tcsh のヒストリ編集機能はカスタマイズができますが、デフォルトでは下の表に示すような Emacs のコマンドと一致していません。また複数のキーカスタマイズを Emacs もどきにセットするだけなら bindkey -e 一発です (vi もどきにセットするのは bindkey -v)。

コマンド	説明
complete-word (TAB)	ファイル名を補完する
delete-char-or-list-or-eof (^D)	1文字削除または補完可能なファイル一覧
down-history (/ ^N)	ヒストリの次へ移動
up-history (/ ^P)	ヒストリの前へ移動
history-search-backward (M-p)	入力行に一致するヒストリを検索 (前方)
history-search-forward (M-n)	入力行に一致するヒストリを検索 (後方)
i-search-back (割り当てなし)	インクリメンタル検索 (前方)
i-search-fwd (割り当てなし)	インクリメンタル検索 (後方)
self-insert-command (文字入力)	文字を入力する
forward-char (^F)	ヒストリの右の文字へ移動
backward-char (^B)	ヒストリの左の文字へ移動
kill-line (^K)	カーソルより左をカット
yank (^Y)	カットした文字列をペーストする

*1 () 内はデフォルト (bindkey -e) でのキー割り当て
tcsh の Emacs モード



画面2 TELNETの利用

て TAB で補完しようとする、その時点ではじめて “ Password for oresama@hoge: ” とパスワードの入力を求められ、ファイルの一覧がFTPによって得られた時点で該当ファイル名を補完（あるいは表示）してくれます。

リモートファイルシステムでなくても、ローカルのMS-DOSファイルシステムをマウントしているときには、UNIXとMS-DOSのファイル名の規則の違いを吸収してくれて、ファイル名をよきに計らって変換してくれるので便利です。

さらに、プログラマーにはvcモードが便利でしょう。これはバージョン管理システム（RCSまたは集団開発に適したCVS）をEmacsが自動的に駆動して、あるバージョンのソースプログラムを取り出して編集可能にし（必要ならロックをかけ）、編集が終わったら新バージョンとして登録、という一連の手続きを代行してくれます。ユーザーは、この手順を意識することなくバージョン管理されているファイルを更新できます。プログラマーの皆さん向けの話題はまた回を改めてまとめたいと思いますが、Emacsを単なるテキストエディタとして使っていたら損をします。

ディレクトリの編集

すでにバリバリEmacsを使っている人は、気づいて使っているかもしれませんが、Emacsの使用中にどこかのディレクトリのファイルの一覧が見たくなったらどうしましょうか。シェルモードでls.....でもいいんですが、実はファイル同様、ディレクトリの編集もできるのがEmacsの強いところです。

ホームディレクトリにディレクトリdocがあるとして、find-fileするときに /docなどとしてみてください。この

場合 /doc/とスラッシュを付けないのがミソです。ディレクトリ名を補完するときは、自動的にスラッシュが付いてしまいが1文字消しましょう。このモードはdiredモードと呼ばれ、lsの出力に似ていますが、ファイルの移動やコピー機能が強力です（図1）。

ファイルにカーソルを合わせて（C-n / C-p や / のほか、 n / p も使えます） e をタイプすれば、そのままファイルが編集できますし（find-file同様）、v をタイプすれば、読み出し専用モードで開きます。v で開いたときは SPC と BS でスクロールでき、q で抜けられますので、わざわざページャ（lessやmore）を開かなくてもOKです。

ディレクトリ名の上で e をタイプしたときは、ファイル同様そのディレクトリがdiredモードで新たに開きます。また、親ディレクトリは ^ で新たに開きます。

移動や名前の変更は R で、コピーは C で可能です。これらは、カーソル行のファイル（ディレクトリ）名を第1引数として、またミニバッファで求められた入力を第2引数として、mvコマンドやcpコマンドを実行しているだけなので、mvとcpの複雑な規則（“ mv FILE DIR ”は単なる移動だが“ mv FILE1 FILE2 ”とすると名前の変更、“ mv FILE DIR/FILE ”ならディレクトリを移動しつつ名前を変更できる）を理解したうえで使ってください。

削除する場合は、d で削除マークを付けてから x で削除を実行します。複数のファイルを移動/コピー/消去するときは、m でマークを付け、その後 R / C / D でそれぞれ移動/コピー/消去を実行します。削除マークの取り消しには u を使用します。

ユニークなのは、ディレクトリ中の で終わっているファイル（Emacsなどのバックアップファイルですね）がいなかったら、 とタイプしてまとめて削除マークを付けられることです。また緊急セーブ用ファイル（#.....# というファイル名になっている）には同様にして # とタイプして削除マークを付けられます。もっとも、このファイルが残っているのにはなにが理由があるはずなので、手作業で消すのはあまりお勧めできません。さらに% d ¥.dvi\$（“ ¥.dvi\$ ”は正規表現です）とすると、.dviで終わるファイルすべてに削除マークを付けられます。

MS-DOSから持ってきた（持っていく）ファイルは大文字/小文字の区別がいい加減で悩まされますが、これも % SPC u（大文字化）や % SPC l（小文字化）で変更することができます。

以上のうち大半のコマンドには引数（C-u 数字）を指

定可能で、カーソル行から指定した数だけまとめて操作できます。同時に使用しているほかのアプリケーションでファイル进行操作して、ディレクトリ中のファイルが増えたり減ったりすることがありますが、`g` とタイプして実際のディレクトリと再度合わせられます。diredモードの必要がなくなったら `q` で脱出できます。

man マニュアルを参照する

これはお察しのとおり、Emacsの中からUNIXのmanコマンドを起動してマニュアルを閲覧しようというものです。Emacsにはinfoという、manより一段すぐれたマニュアル閲覧方法がありますが、すべてのコマンドに完備しているわけではありません。

使い方はM-x manです。起動するとどのマニュアルを見るか聞いてきます。たとえばtcshのマニュアルなら“Manual entry (default man):”と聞いてきたときに“tcsh”と答えます。残念ながら、補完は効きません。これは、manコマンドに問い合わせるだけではじめてどんなマニュアルがあるかがわかるためでしょう。

UNIXのマニュアルは、1~8章(その他ローカルなど)に分かれています。同じ見出しのマニュアルがいくつかの章にあるときは、たとえば“crontab(1)”“crontab(5)”のように章を指定します。表示はmanコマンドと変わりませんが、コピー&ペーストを使った再利用が楽なこと、いくつかお手軽コマンドが用意されていることが利点です。

`n / p` で、次 / 前の見出しに飛ぶことができます。

manコマンドを利用したことがあればわかりでしょうが、マニュアルにはNAME、SYNOPSIS、DESCRIPTIONなどの見出しが付けられていて、それらのあいだを行き来できるのです。m コマンドは別のマニュアルを見るのに使えます。m とタイプした時点でカーソル位置の単語をデフォルトで拾ってくれるので、マニュアル中でほかのコマンドに言及している場合は、そこにカーソルを合わせて m Enter とタイプするだけです。

r は m コマンドと似ていますが、特にSEE ALSO(関連して参照すべきマニュアル)にリストアップされているマニュアルに一発でジャンプできて便利です。マニュアルを読んでいると、往々にして別のマニュアルを参照し、その最中にまた別のマニュアルを参照して……ということが多いものです。シェル環境でマニュアルを参照するのに比べて、かなり便利だといえましょう。

Emacs で喋り

といってもEmacsが喋るわけではありません……実はEmacsが本当に喋る(テキストを読み上げさせる)Emacspeakというソフトウェア(マクロ+音声合成の駆動部分)も実在してとても興味深いのですが、まだ日本語を読めるように改造した人はいないようです。いずれにしてもあとの回でこの話題は取り上げましょうかね。

さて本章の主旨は、Emacsを使ってチャットをしようというものです。ネット上のチャットのプロトコル(方式)というとIRCがもっとも有名かつ大規模なものです。チャ

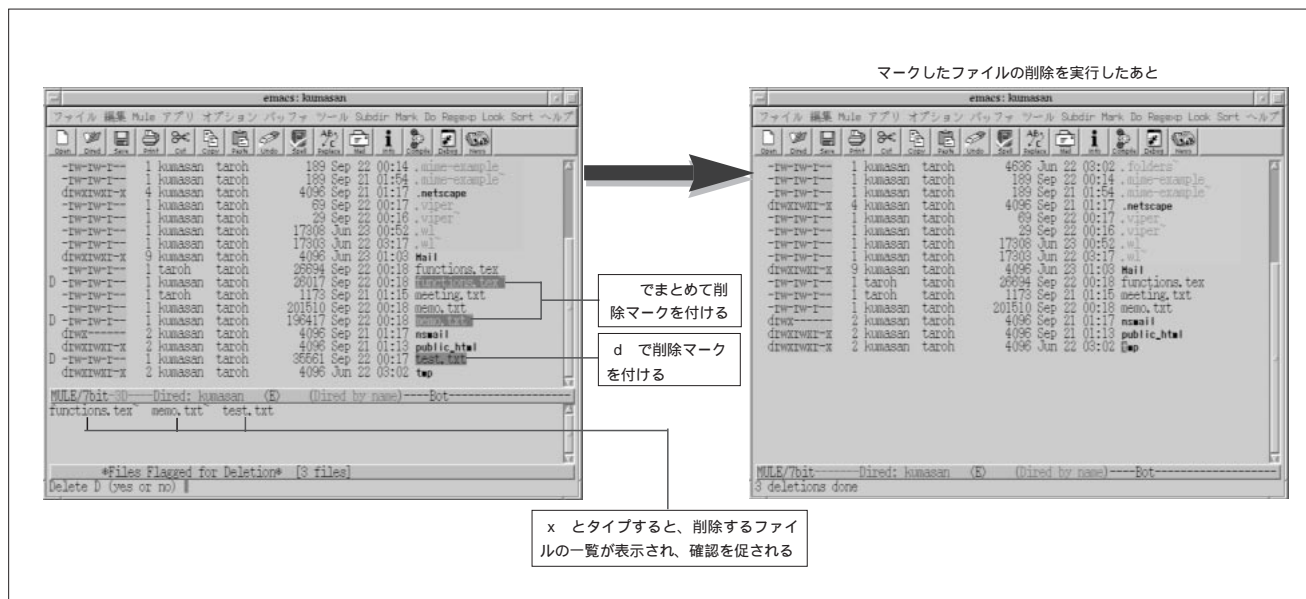
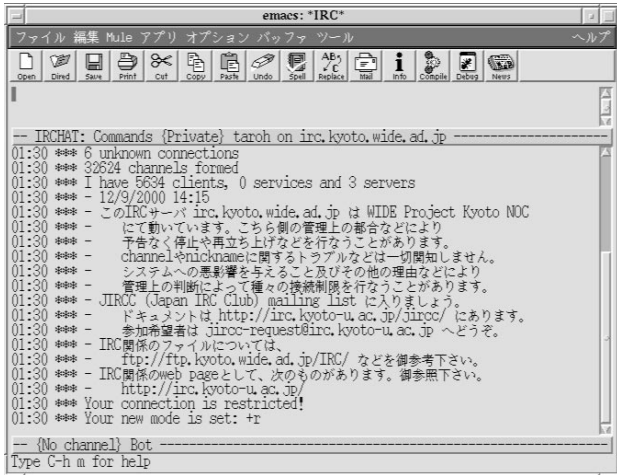
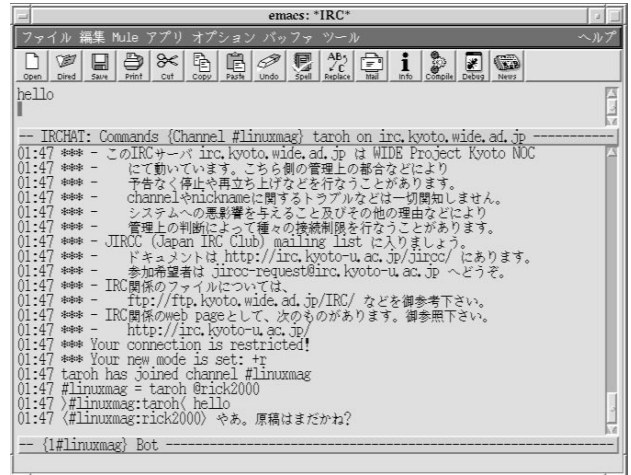


図1 diredモードでファイルを整理する



画面3 Irchatの起動直後の画面



画面4 C-cjでチャンネルにjoinしたところ

ットそのものにあまりなじみのない方もいらっしゃるかもしれないので、まずそこから説明しましょう。

IRCは、クライアント（我々のコンピュータ）がサーバ（お喋りを取り持つコンピュータ）と接続して、同じIRCネットワークに接続しているほかのクライアントとお話ができるシステムです。IRCネットワークはサーバとサーバがバケツリレー式に接続されているので、世界規模で張り巡らされたネットワーク上にいる数万人の人と同時に会話ができます。IRCネットワークの大規模なものといえば、IRCnetとEFnetでしょう。日本国内でオープンなサーバはたぶんほとんどがIRCnetなので、ネットニュースなどで「詳しい話はIRCの#hogehogeでしょうぜ」などと書いてあっても、これはIRCnetのことだったりします。

またIRCサーバのいくつかは、ネットワーク上のどこにいてもクライアントにも接続が開放されています（国名などによって接続を制限していることがあります）。日本で接続が自由に開放されているサーバとしては、irc.kyoto-u.ad.jp などがあります。

各クライアントのユーザーはニックネームで識別されます。IRC開始時に適当に付けることができますが、すでにそのとき存在する他人と同じニックネームは使えません。同じIRCネットワーク上の数万人が一度に会話をしたらわけがわからなくなってしまうので、ネットワーク全体はチャンネル（部屋と考えればよいでしょう）に分かれていて、同じチャンネルにいるユーザーだけが会話に参加できます。チャンネルは“#linux”などのように“#”で始まる名前が区別されます。チャンネルには現在の話題に応じてタイトルが付けられていて、チャンネルの一覧をながめると、結構興味のある分野が見つかったりするものです。

面白いのは、チャンネルを開いた人がチャンネルを制御する

権利をもっていて（ニックネームの前に“@”が付きます）、チャンネルのタイトルを設定したり、チャンネルに参加する人を制限したり、制御権をほかのユーザーにあげたり剥奪したり、チャンネルから追い出したりということまでできるようになっています。

Webの掲示板を頻繁に更新してメッセージを交換するチャットもどきもよく見かけますが、それに比べてはるかにスピーディで多機能、かつ大規模です。また用途がほとんど1対1の通信に限られるIPメッセージング（ICQなどが有名です）と違って、共通の話題に興味のある人みんながチャットができます。さらに、Web掲示板やIPメッセージングではホストコンピュータ1台がすべてを取り仕切っているの、そこが停止したらおしまいですが、IRCのネットワークではクライアントはどのサーバに接続してもOK（接続資格さえあれば）という利点もあります。

さて前置きが長くなりました。Emacsにはirchatというマクロがあります（外部プログラムは起動しません）。Emacsを用いずとも多数のIRCクライアントが知られていますが、それらに比べて、

日本語の入力が容易である

ログの（お喋りした）内容を加工・保存できる。逆にファイルからコピー&ペーストしてチャットの相手に示せる仕事しているふりをしてチャットができる（笑）

という利点があります。

irchatがデフォルトで組み込まれているのはVine Linux だけですので、このほかのディストリビューションをお使いの方はインストールする必要があります。

irchatはすべてEmacs Lispで書かれているので、先月

号で説明したようにマクロをしかるべきところに置くことでインストールできます。

まずはirchat一式を持ってきましょう。FTPツールやWebブラウザなどでftp://ftp.kyoto.wide.ad.jp/IRC/irchat/からirchat-jp25a12.tgzを持ってきます。インストール先は以下、/usr/local/lib/mule/site-lisp/irchat/というディレクトリを想定していますが、お使いのLinuxのディストリビューションに合わせてローカルマクロのインストール先パス名を変えてください。次にrootの権限で、

```
# mkdir -p /usr/local/lib/mule/site-lisp/irchat
# cd /usr/local/lib/mule/site-lisp/irchat
# tar xvzf 持ってきたファイルのパス・ファイル名
```

とします。muleをお使いの方は、Makefileの、

```
EMACS = emacs
```

となっているところを、

```
EMACS = mule
```

と変更します。あとは、

```
# make
```

とするだけです。

またirchatを立ち上げるユーザーのemacs(または共通の設定ファイル)には、すくなくともリスト1のような設定をしてください。

さてirchatの操作ですが、まずM-x irchatで起動します(画面3)。チャンネルに参加しなければ意味がないので、次にC-c jでjoin(参加)します(画面4)。このあと、まずは新しいチャンネルを自分で作って練習してみればよいのですが、おもな操作を表2にあげます。

このほか、特定の人と1対1で会話するモードや、特定

の人をあるチャンネルに招待する機能がありますが、これらは使っているうちに次第に覚えるでしょう。

特定の操作については、IRC上にいる人に聞くのがいちばんですが、IPメッセージャーなどと違ってIRCは「オープンなプロトコル」、つまりOSやソフトの垣根を越えて共通にお話してできるのが特徴ですから、教えるときも教わるときもどこまでがIRC共通の概念で、どこからがEmacs上のirchat特有の操作なのか区別しなければいけません。

ニュースを読む

ネットニュースは、最近でこそメーリングリストやWebの掲示板に圧倒されているようですが、地球規模で専門分野の討議ができる場(compやsciなど)としての機能は、まだまだ失われていないと思います。

ネットニュースは古代、rnewsやrn、vnといったスタンドアロンのツールで読み書きされていました(ちょうど/bin/mailでメールを読み書きするのと、MewやWanderlustで読み書きするのに相当するでしょう)。

Emacsからネットニュースを読むための手段としては、古典的なgnusというマクロがありますし、最近では前々回紹介したWanderlustなどがあります。いずれにしてもEmacsを使えばIRC同様、日本語が入力しやすいなどの利点があります。

ニュースは通常、プロバイダや学校の管理者が指定したニュースサーバ(NNTPサーバ)とEmacsが対話しながら、ニュースグループの一覧、記事のタイトルや内容を取り出していくことになります。このサーバをまず設定しないとイケません。emacsに、

コマンド	説明
C-c j	別のチャンネルに参加する。複数のチャンネルに参加してバッファを移動すれば)同時にお喋りできる
C-c n	ニックネームの変更
C-c C-n	そのチャンネルにいる人の一覧を表示
C-c f	指定したユーザーの詳細情報を表示
C-c C-p	カーソル位置のバッファのチャンネルを終了
C-c q	irchatの終了

表2 irchatのおもなコマンド

リスト1 irchat.elの設定

```
(setq load-path (cons "/usr/local/lib/mule/site-lisp/irchat" load-path)) .....パス名はirchat一式がある場所に設定
(autoload 'irchat "irchat" nil t)
(setq irchat-server "irc.tohoku.ac.jp") .....IRCサーバ。仕組みがわかってきたら近くのサーバにしよう
(setq irchat-name "SASAKI Taroh") .....本名。漢字も使えるが相手は日本人だけではないことに注意
(setq irchat-nickname "俺様") .....ニックネーム
(setq irchat-startup-channel-list '("#linux" "#linuxmag")) .....はじめからjoinしておきたいチャンネル
```

(setq gnus-select-method '(nntp "news.hoge.ne.jp"))
と書くと、NNTPサーバマシンにnews.hoge.ne.jpが指定されたこととなります。このほか、環境変数NNTPSERVERにマシン名をセットしておくという方法もあります。

さて、設定はこれだけなので起動してみましょう。gnusは標準で提供されているので、M-x gnusで起動します。X Emacsでは起動後のモード行の牛さんが可愛いんですね。

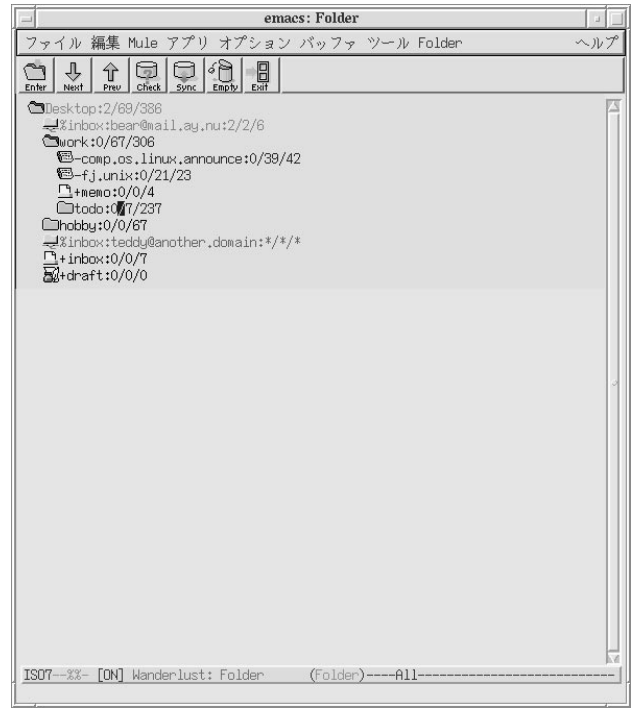
起動後は、グループバッファ（Wanderlustではフォルダバッファに該当）、サマリバッファ、アーティクルバッファ（Wanderlustのメッセージ）というバッファを行き来しながら使うこととなります（画面5）。Wanderlustをお使いの方にはおなじみですね。

グループバッファにいるとき注意しないといけないのは、グループによって読む（購読）・読まない（非購読）を指定できることです。はじめはほとんどのグループが表示されませんが、読みたいグループがあるのにみすみす見逃すのもしゃくです。かといって全部のグループを表示させるとうっとうしくなりそうです。

読めるニュースグループがどれだけあるかは、NNTPサーバがどれだけのニュースを購読可能にしているかによります。信じられないかもしれませんが、世界中の有名なニュースグループを購読可能にしているサーバでは、数万のグループが購読可能です。もし十分な時間があるか、ニュースサーバの管理者に尋ねてみて、たかだか数百程度のニュースグループしかないようであれば、思い切ってグループバッファで L または A SPC k とタイプしてみてもよいかもしれません。サーバが購読可能なすべてのリストが作られますので、あとはグループバッファで u をタイプして非購読にするか、gnus終了後にできているファイル /.newsrcのニュースグループ名の後ろの“:”を“!”に置換すれば、そのグループは非購読になります。たとえば、“hoge:12-34”となっていたら“hoge!12-34”としてみましょう。次回からは、グループモードにいるとき l とタイプすれば、自分が読みたいグループのみを表示してくれます。

Wanderlustを使ってニュースを購読する場合も、まずサーバの設定をしましょう。/.wlに、

```
(setq elmo-default-nntp-server "news.hoge.ne.jp")
(setq wl-nntp-posting-server elmo-default-nntp-server)
```



画面5 gnusでニュースを読む

のように書きます。ここでは、投稿先と購読元のサーバを同じnews.hoge.ne.jpとしています。

あとは単純に、フォルダ一覧に読みたいニュースグループを追加していただくだけです。たとえばニュースグループfj.unixを追加したいときは、フォルダモードの追加したい位置で m SPC a をタイプし、-fj.unixを指定します。または、.foldersを手で編集するのが好きな方は、

```
unix{
    %#mh/linux
    -fj.unix
}
```

などとしてもかまいません。



Emacsから何かしようという例は、このほかにもいろいろあるのですが、次回はおもにEmacsでWebを見るというテーマでお送りいたします。WebページをEmacsで楽々と作る方法を紹介するのでお楽しみに。あわせて今回紹介しきれなかった、小物の数々も取り上げたいと思います。

Linux 日記

第14回 メール配送(1)

今回からはメール配送のしくみを解説します。基本的なインターネットサービスであり、利用者も多いメールはどのように配送されるのでしょうか。

文： 榊 正憲

Text : Masanori Sakaki



この記事を読者が読むのは秋だろうが、これを書いているのは8月下旬、連日気温が30度を超えている日々である。筆者は神奈川の海沿いに住んでいるので、都心ほどの気温にならないのがせめてもの救いだ。SETI絡みでエアコンをどうするかという問題は、なしくずし的に準24時間運転という形になっている。風の通る日は窓を開けて、エアコンを止めるという運用である。

そんなある日、寝る前に新作のプレステのゲームをやっていた時のことである(ここしばらく建設機械の運転にはまっている)。エアコンからドボドボと水が落ちてきた。急遽エアコンを止め、真夜中に雑巾やらタオルやらで後始末をする羽目になった(このどたばたで、パワーショベルの運転は時間切れゲームオーバーになってしまった)。24時間ほど前にも少し水が漏り、フィルタまわりの掃除をして試運転中のことだった。もう10年も使っているエアコンなので(しかも稼働率は一般家庭やオフィスよりはるかに高い)、そろそ

ろ買い換えかなとも思っていたのだが、とりあえずどうにかしなければならぬ。コンピュータを使っている場合、冬場はエアコンが止まってもどうにでもなるが(人間はちと寒いが)、夏場のエアコン停止は致命的である。不幸中の幸いは、春にやった模様替えにより、エアコンの下にコンピュータ類がなかったこと(かつては、真下にメールサーバがあった)、そして、エアコンにたどり着くための空間があった(かつてはたどりつけなかった)ことである。

前日の水漏れでフィルタを掃除したと書いたが、フィルタ掃除はエアコンメンテの基礎である(まめにすればなおよい)。フィルタが目詰まりするとエアコン内部の圧力が低くなる。内部からファンで空気が吸い出されているのに、外から十分な空気が流入しないからだ。その結果どうなるか? 外部に通じている排水管からも空気を吸い込もうとするのである。これにより水の排出が悪くなり、最悪の場合、水受け皿が溢れて水が落ちてくる。特に、排

水管の水平配管部が長いとこの症状が出やすい。最初の水漏れの際はまずこれを疑った。実際、フィルタには目一杯ホコリが詰まっていた。フィルタをきれいにし、内部のフィンを掃除して直るかなと思ったのである。

しかし、運転を再開するとまた水漏れである。しかも前回よりひどい。最初の水漏れはボタン、ボタンという感じだったが、今回はドボドボきた。フィルタがきれいになり、冷却能力が高まったぶん、水もいっぱい取れたのだろう。こうなると原因はひとつ、排水管の詰まりである。一般に配水管の入口と出口は詰まりやすい。入口はホコリやカビなどが詰まり、出口は泥がたまったり虫が巣を作ったりする。あと、パイプが曲がっているところも詰まりやすい。出口のほうは取りあえず異常なし。配管途中の曲がっている部分は調べようがない。入口は室内機の内部である。かくして、真夜中のエアコン分解が始まった。

水受け皿は、熱交換フィンの下にあ

る。もちろん、表からは見えない。脚立を持ち出すと、パネルに貼ってある注意書に従ってコネクタ、ネジを外してフロントベゼルを取り外した。少し見えた水受け皿は、怪しげな物体でいっぱいだ。恐らく、ホコリとカビの集合体だろう。見ないほうが幸せといった類いのものである。こりゃ詰まるわなと思いつつ、排水管の接続部を探すが、どうやら室外機の電源接続端子の裏のほうのようだ。端子部は簡単には外れなかったが、ネジを外してちょっとずらすと排水口がのぞけるようになった。案のじょう怪しげな物体が詰まっている。角度が悪くてピンセットなどは届かない。そこで、部屋に落ちていた直径8mm、長さ数センチのスプリングをピローンと伸ばしたものを使った。先端の輪の部分で異物をこそぎ落とし、流してしまおうという作戦だ。スプリングなので、適度な弾性と剛性があり、手の届かないところでうまくゴソゴソとこすれるのである。これがうまくいって、室外機のそばの排水口から細かく砕かれた異物がわらわらと流れ出し、排水管は無事開通した。その後、手の届く範囲で水受け皿を掃除し、分解した部分を組み立て、運転を再開した頃には、すっかり外が明るくなっていた。

水漏れは直ったものの、長年の酷使で冷房能力がかなり低下していることが判明し、結局、この2週間ほど後にエアコンを交換した。

システム管理者の仕事

もし読者がシステム管理者になりたいと考えているのであれば、コンピュータ以外の知識、ノウハウもかなり知っておく必要がある。エアコンのメンテもそのひとつだ。最終的には修理業者を呼ぶにしても、とりあえずどうに

か運用を続けるための方策を講じなければならぬ。「エアコントラブルだからシステム止めるよ」でユーザーが許してくれる環境なら幸せである。「あした締め切りだから、今晚だけはどうかして」と泣きつかれるのが普通だ。サーバそのものが動かないのならあきらめもつぐが(というかどうかにもならない)、エアコンが動かないだけとなると、どうにかしてくれとなってしまうのだ。実際、小規模なサーバールームなら、真夏でなければ、窓を開けて扇風機を回すといった方法で運用することも不可能ではないし、勤められることではないが、水漏れを受けるために、大きなサーバコンピュータの上に洗面器を置いて運用したこともある。

もちろん、知らなければならぬのはエアコンのメンテだけではない。建物内の電源系統、弱電信号配線の配置や引き回しなどにも詳しくなければならぬ。システム管理には、総務部営繕課的業務がかなりあるのだ。現在のオフィスは、床を数センチかさ上げしたフリーアクセスフロアが多いが、こうっていないオフィスでは、天井裏に配線を通すことが多い。某A社が初台に引っ越して来る前にいたビルは、天井裏配線が多かった。当時システム管理者をやっていた筆者は、やれマシンの増設だとか移動だとかいって呼ばれていた。システム管理者がそういう作業を行うといえば、ノートPCだとかソフトウェアのインストールメディアだとかを持って颯爽と出かけるというイメージがあるかもしれないが、実際にはかなり違うスタイルだった。「ショムニ」の江角マキ子のように脚立をかつぎ、京野ことみのように台車を押して出かけるのである。

一般に、天井には点検口があり、天井裏をのぞけるのだが、常に作業場所

のそばに点検口があるわけではない。また、何メートルもケーブルを這わせなければならない。たとえば、点検口がない場所で作業を行うなら、天井に埋め込んである蛍光灯などの照明器具を外して、そこから天井裏にアクセスしなければならない。何メートルもケーブルを引っ張る時は、経路上の蛍光灯をすべて外すわけにもいかないし、天井裏を這いずり回るのも大変なので、ケーブルキャッチャーという特殊な道具を使う。これは伸縮式の釣り竿の先にフックが付いているような道具だ(業務用の配線工具などを扱っているお店で入手できる。インターネットで調べたら、秋葉原の愛三電機(<http://www.aisan.co.jp/>)では扱っているようだ)。縮んでいるときは1m程度の長さなのだが、ロッドを伸ばすと5mくらいの長さになる。伸ばした状態でフックにケーブルを引っ掛け、縮めればケーブルを手繰り寄せられるという仕組みだ。また、縮めた状態でケーブルをフックにとめ、ロッドを伸ばせば遠くまでケーブルを送るといった応用的な使い方もできる。慣れれば、天井裏を通るハリの貫通穴の中にケーブルを通すといった作業なども簡単にこなせるようになる。およそ読者の想像するシステム管理者の姿ではないだろう。気分はすっかり電気工事屋さんである。実際、この辺のノウハウは、出入りの工事屋さんから盗んだものだ。新米管理者は、先輩管理者の技を盗むだけでなく、各種営繕作業のノウハウも盗むとよい。

実際問題としては、オフィスの配線作業をすべて管理者が行う必要はまったくない。まとめて作業をする場合は、業者に委託したほうがはるかに簡単だし、費用も安い。電源工事や電話工事を行うには、資格も必要だ。しかし、ケーブル障害やマシンの移設などで、

数本のネットワークケーブルを急遽敷設しなければならないといったときに、その作業を自分で行えば、時間と費用を大きく節約できるだろう。

さて、本編である。DNSの話がやっと終わった。せいぜい2、3回かなと思って書き始めたのに、なんと半年にも渡ってしまった(余計な話をたくさん書いたからという説もある)。次はどうか。あえてマイナーなUUCPの話とか、王道を進んで(禁断の)Sendmailの話なんて声も編集部からちらほら聞こえてくる。

筆者は、比較的最近までUUCPを使っていた。今はOCNエコノミーの常時接続環境で、ネームサーバ、メールサーバを自宅で運用しているが、以前はIIJのUUCPサービスを使っていた。これは契約者側でUUCPとメールサーバをセットアップし、IIJと接続して使うという形である。XX.co.jpといった独自ドメインを持っていればそれを使用し、持っていなければIIJのサブドメイン名を使って、メールやニュースを運用できるというものだ(知り合いのIIJの人間は、早くこのサービスを止めたといった)。

UUCP

UUCPはUnix to Unix Copy Programの略で、いくつかのユーザプログラムやシステムプログラムの総称である。ファイルをリモートコピーするuucpというプログラムもあるが、大文字でUUCPと書いた場合は、これらを総称したシステムの名称とするのが一般的だ。この一群のプログラムを使うことによって、モデムやISDN TAを使ってダイヤルアップ接続したUNIXホスト間で(DOSなどにも互換プログラムが移植されている)、ファイルのコピー、プログラムのリモート実行ができる。また現在のバージョンでは、IP接続されたホスト間でもこれらの処理を行うことができる。これとメールサーバ、ニュースサーバを組み合わせることにより、ダイヤルアップ接続環境でメールやニュースの配信が可能になる。

UUCPのユーザーコマンドは、さまざまなコマンドリクエスト(ファイルのコピーやコマンドのリモート実行など)と必要なファイルを、UUCP用のスプールディレクトリに投入するという形で処理を行う(各種UUCPコマンドの実行により相手側に接続されるわ

けではない)。実際にリモートホストに接続を行い、スプールに投入されたジョブを実行するのは、uucicoというプログラムである。これはcronによって、ジョブの有無に関らず、定期的に行われる。

uucicoは、モデムなどを使ってリモートマシンに接続し、適当なユーザー名で受信側ホストにログインする。受信側では、このユーザー名によるログインに対して、ログインシェルとしてuucicoプログラムを実行する。これにより、発信側と受信側でuucicoが対話できるようになる。

発信側のuucicoは、自身のスプール上にあるリクエストとデータを相手側のuucicoに送る。受信側はこれを処理し、コマンドを実行したり、ファイルのコピーを行う。発信側の処理が完了すると、今度は受信側のuucicoが自身のスプール上にあるリクエストとデータを発信側に送る。つまり、発信は一方から行うが、この接続セッションで、双方にスプールされているジョブが実行されるのである(図1)。

今でこそ電子メールは、IP接続でサーバ間をリアルタイム配信するという

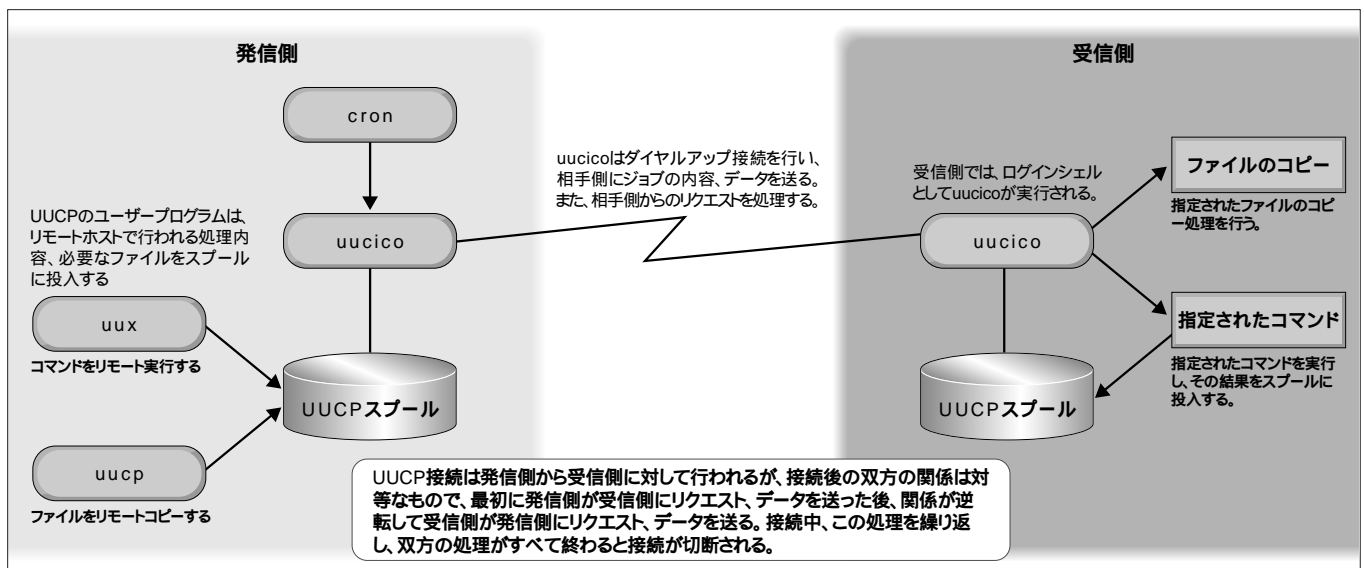


図1 UUCP

Column

JUNET

大学、研究機関、企業の間でUUCPを使ったメール、ニュースの配信実験を行うためにJUNETという組織が作られた。活動内容

からすると、Japan UUCP NetworkとかJapan Unix Networkといった意味だともうかもしれないが、当時の人間の多くは、中心メンバーだった村井 純先生のネットワークだからジュンNET、JUNETだったと信じている。

運用が当たり前だが、常時接続が一般化する前は、ダイヤルアップ接続のUUCPを使ったメール/ニュース配信環境が大学、研究機関、一部の会社組織などで運用されていた。この当時(1980年代中ごろ)は、通信事業者以外が第三者の通信を媒介するというのが法的に禁止されており、組織間のメールのやり取りはあくまで研究、実験という形で行われていた。また、運用の中心だった組織が大学や公立研究機関であったため、商用利用はできなかった。

もはや過去のものという感もあるが、UUCPが便利に使える状況というのがないわけでもない。ある程度(数十人程度)のユーザーがメールだけを使っているといったサイトなら、各人がばらばらにダイヤルアップ接続を行うよりは、UUCPを使ってバッチ形式でメール配信を行うほうが安価である(常時接続が安価になってきたので、もはやメリットはないかもしれないが)。

UUCPが先かメールサーバが先かUUCPのことがわかって、それでもメールを運用するとなると、どうしてもメールサーバ設定についての知識が必要になる。メールサーバについて知っていれば、UUCPのことは知らなくてもIP接続の電子メールを運用することができる。また、IP接続のメール転送しか行わないという設定は比較的簡単だが、UUCPが関与するとなると、多少複雑になる。というわけで、先にメールサーバの話をしておくべきだろう(はたして、あとで本当にUUCPの話をするのだろうか?)。

ここではSendmailについて解説する。qmailなど、ほかにもいくつかメールサーバソフトウェアがあるが、筆者は昔からSendmailを使っているので、ここで取り上げるのはSendmailだけである。

Sendmailの設定(特にsendmail.cfの記述)が複雑なことは有名だが、それについて解説する前に、まず電子メ

ールの基本的な動作について知っておく必要があるだろう(また長くなりそうな気がする)。例によって、Sendmailについて正確な知識を理解し、実際にSendmailを使ってシステムを運用したのであれば、オライリーの『sendmailシステム管理』と『sendmailリファレンス』を読むことを勧める。というか、これを読まないことにはどうにもならないだろう。

UUCPについては、やはりオライリーの『UUCPシステム管理』(これはアスキー発行)というクマの本が役に立つが、ちょっと古い本なので、本屋で探すのは難しいかもしれない。改訂版が出ていないということからも、もはやUUCPは世間の主流からは外れているということになるのだろう。

ローカルホスト上のメール送受

多くの人にとって、インターネット電子メールを使うということは、自分のマシン(UNIXでもMacでもWindowsでも)をなんらかの方法でプロバイダや会社、学校のネットワークに接続し、ネットワーク上のメールサーバとやり取りするという形になる。しかし、電子メールをこのような形で運用するのは結構高級なやり方なのだ。ここではもっと単純な形式の電子メールから話を始めよう。

コンピュータの端末を使って、別のユーザーにメッセージを送るという電子メールの歴史は古い。大型のホストコンピュータに多数の端末を接続し、それを複数のユーザーが対話的に使っていたTSS(Time Sharing System)環境(図2)(LAN接続のワークステーション環境に移行する前は、UNIXもこのような使い方だった)では、たいはい何らかの形の電子メールプログ

Column

通信事業者

UUCPを使った通信実験が始まったころは、第三者の通信を媒介できるのは、NTTなど、通信網を自前で持ち、免許を受けた通信事業者だけだった。その後、規制緩和により、自前の通信回線を持たない事業者も、免許を受けて第三者の通信を媒介でき

るようになった。また、事業の認可手続きも簡略化された。このような業者は、NTTなどの通信回線を使って顧客間の通信を媒介する。多くのインターネットプロバイダは、この形で業務を行っている。

自前の通信網を持つNTTのような事業者を第1種通信事業者、自前の通信網を持たない事業者を第2種通信事業者という。

ラムが提供されていた。このような電子メールプログラムの基本機能は今の電子メールと変わらない。メールを送る相手を指定し、適当なテキストメッセージを送るという形である。宛て先として指定されたユーザーが電子メールプログラムを起動すると、自分宛に届いたメールを読むことができる。1台のコンピュータ上ですべてのやり取りが行われるので、このようなメールシステムの構造は直感的に理解できるだろう。送信側のメールプログラムは、ホストマシン上の適当なディレクトリにある相手のメールボックスにメッセージを書き込むだけでいい。メールを読む側は、自分のメールボックスを見るだけでいい。多少頭を使う部分があるとしても、書き込みと読み出しの排他制御程度だ(図3)。

UNIXにも初期の時代から電子メールプログラムがあった。そのものずばり、mailというプログラムである。これは現在のmailプログラムとは違うものだ。今のUNIXでmailとタイプして起動されるプログラムは、通称ucbMail(ユーシービーメール)と呼ばれるものである。それに対して、ここでいうmailは、binMail(ピンメール)

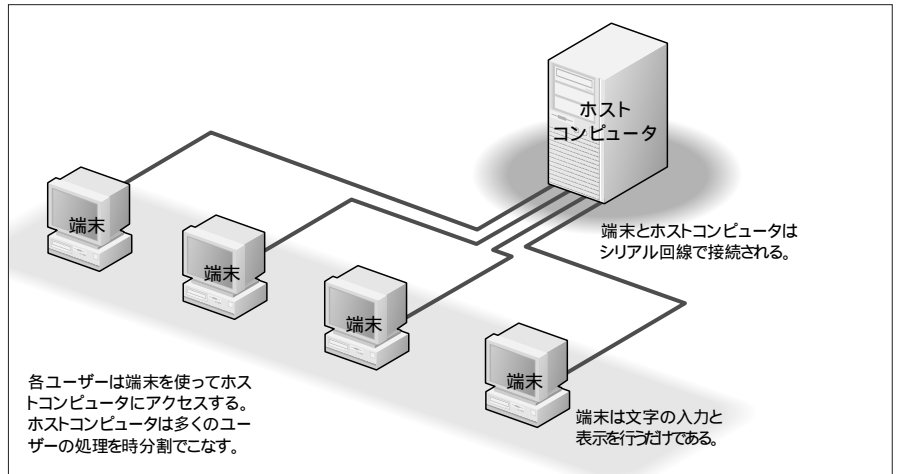


図2 TSS環境

と呼ばれるものだ。これはもともと、/usr/ucb/mailと/bin/mailというパスだったことに由来している。現在はbinMailに相当するユーザープログラムはなく、mailといえばucbMailである。しかし、内部処理において、かつてのbinMailが行っていた機能が必要なため、同等な機能を持つmail.localというプログラムが使われている。これはSendmailの内部から呼び出されるプログラムで、ユーザーが直接使うことはない(この解説記事では実際に使うが)。そのため置かれているディレクトリも、/usr/libexecなど、パスの通っていない場所である。

binMail

binMail、すなわちmail.localの働きはさほど複雑なものではない。このプログラムにメールを与えると、宛て先ユーザー名に基づいて、そのユーザーのメールボックスファイルに与えられたメールテキストを書き込むだけだ。メールに使われる宛て先の名前はUNIXのユーザーアカウント名である。メールそのものは、適当なスプールディレクトリ(/var/mailなど)の下に、ユーザー名と同じ名前のファイルとしてスプールされる。あるユーザー宛のメールが複数スプールされる場合も、ファイルは1つだけで、各メールが順につながれて1つのテキストファイルの形で保存される。

ためしに、直接mail.localを実行してみよう(画面1)。宛て先ユーザー名を指定し、標準入力でメール本文を送る。

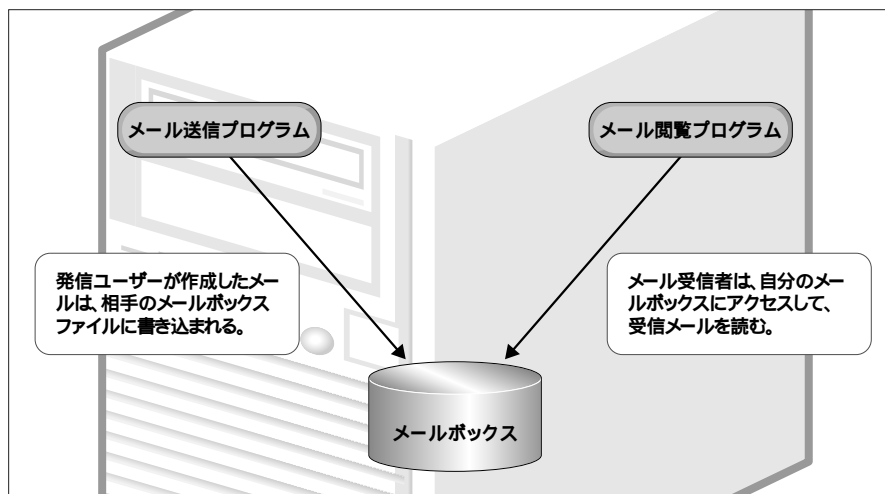


図3 ローカルホスト上でのメール送信
メール送信、受信がすべて1台のホストコンピュータ上で行われる。

```
% /usr/libexec/mail.local test
To: test
From: masa
Subject: test

test message
%
```

画面1 mail.localを直接実行してみる

宛て先はtestというユーザーである(このような実験をするときは、安全のためにテスト用のアカウントを使ったほうがよいだろう。ここではtestというユーザーアカウントを使っている)。本文には、To:やFrom:といったヘッダを手でタイプしている。リスト1に、これを実行した後のユーザーtestのスプールファイルの内容を示す(Linux magazine上の「Linux日記」という連載であるにも関わらず、以降の結果はすべてBSD 3.3で実行したものに基いている。あしからず)。

各ユーザーのスプールファイルは、そのユーザーしか読み書きできないパーミッションになっているが、mail.localはrootにSUIDされているので、他のユーザーのスプールファイルに書き込むことができる。

メールボックス

mail.localに与えたメール本文の前に、“From masa Mon Sep 4 14:02:30 2000”という行が付加されているのがわかる(mail.localはmasaというユーザーが実行した)。これはmail.localが付加した行である。また、よく見ると、メールの最後に空行が1行付加されていることもわかる。mail.localを何度も実行すると、同じファイルにこの形式でメールが追加されていく。

複数のメールは連結されて1つのファイルになるが、空行に続けて行頭から始まる“From”という文字列によって境界が判定される。これが、

mail.localが末尾に空行を追加する理由である。何らかの理由によって、メールの本文中にこれと同じパターンの文字列が現れる場合は(英文テキストなど)、メールの区切りと区別するために、“From”の前に“>”という文字が自動的に挿入される。

このスプール形式は今でもUNIXの標準のメールスプール形式であり、UNIX上で動作する各種メールプログラムは、この形式のスプールファイルを処理できるように作られている。UNIXのメールのスプールディレクトリの下は画面2のようにになっている。このファイルを見れば、自分宛のメールを読むことができる。

ここでは、この形式で複数のメールを保存したファイルをメールボックスファイルと呼び、特にシステムのメールスプールディレクトリに置かれたものを、システムメールボックスファイルと呼ぶことにする。ucbMailなどは、ユーザーのホームディレクトリなどに置いたこの形式のプライベートメールボックスを扱えるので、それと区別するために「システム」と付けている(メールアプリケーションが管理するプライベートメールボックスの形式は、本来、そのアプリケーションが自由に定義できるものだが、ここではメールアプリケーションには触れないので、あえてこの形式を指してメールボックスファイルと呼ぶ)。

システムメールボックス中にスプールされたメールをどのように扱うかは、

各ユーザーの環境しだいである。同じホスト上で各種メールプログラムを使って読む場合は、そのプログラムがこのメールボックスにアクセスすることになるし、ネットワークを介して別のマシン上から見る場合は、また別のプログラムを使ってメールボックスの内容をユーザーのマシンにダウンロードすることになる。

ヘッダも含めて、メールはすべてテキスト形式で扱われる。メールアプリケーションを使ってテキスト以外の形式のファイルをメールに添付した場合は、テキスト形式にエンコードして本文中に埋め込まれる。

プログラムの実装に依存することであるが、テキストの各行には長さの制限がある。通常は数百バイト以上の長さであるが、長い文章などを途中で改行を入れずに入力するとこの制限にひっかり、文字が化けたりすることがある。問題なくメールを送るためには、70文字程度で適宜改行することが必要である(今のメールアプリケーションなら、自動的に改行を入れる機能があるはずだ)。どうしても改行を入れない場合は、バイナリファイルとしてエンコードし、添付ファイルとして扱う必要がある。

今回は

今回は、ucbMailの使い方、メールの中身、リモート配信の基本的な考え方などについて説明するつもりだ。

リスト1 ユーザーtestのスプールファイル

```
From masa Mon Sep 4 14:02:30 2000
To: test
From: masa
Subject: test

test message
```

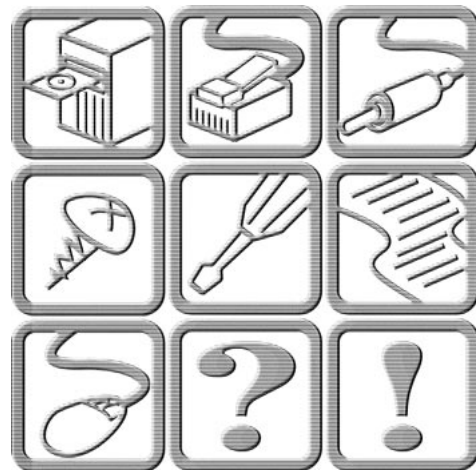
最後の空行に注意

```
% ls -l
total 131
-rw----- 1 masa user 0 Sep 4 09:47 masa
-rw----- 1 root wheel 69002 Apr 1 05:30 root
-rw----- 1 test user 85 Sep 4 14:03 test
-rw----- 1 tetsu user 63447 Sep 2 20:23 tetsu
```

画面2 メールのスプールディレクトリ

Try & Try

[trái] [trái]



ハードディスクにまつわるいろんな話

文: 政久忠由

Text: Tadayoshi Masahisa

今回は、前回に引き続き Netfilter フレームワークの詳細をやる予定だったのだけれど、オリンピックとか、個人的な諸般の事情から、ちょっとインタラプト発生ということで、ハードディスクについて見てみたいと思う。

それにしても、時差のほとんどない地域でのオリンピックというのは、生活が夜型の僕にとっては、テレビ観戦するのもひと苦勞で、サッカーの時間を何度も寝過ごしそうになってしまった。特別肩入れしている競技とか、アスリートはいないんだけど、不思議なもので、どれも見逃したくないという感覚におちいるんだよね、オリンピックって(もちろん日本人に関係なく)。

だから、だらだらとまどろみながら見ちゃいました。かなり寝不足気味で、脱力感に苛まれてしまっているけど....

しかし、それにしても、実況とインタビューはどれも酷かったね。どれだけ、ないほうがいいと思ったかわからない。競技が終わったあとには、独特のアンニュイな浮かれ気分があるわけだけど、ものすごくその気がそがれちゃうんだなあ、これが。最悪だった。それに、よどんだ空気のような嫌悪を感じさせるメダルの数を気にする連中の発する言葉。毎度のこととはいえ、あきれるばかりだ。

とにかくオリンピックは、大人たちのエゴイズムを満たすための場としてだけじゃなく、これからを担う子供たちにスポーツの本当の魅力を伝える場でもあるのだから、もう少し考えてほしいと思う。たいして純粹でもなく、無垢でもない僕でさえ、こんな風に感じちゃうんだよ。



最近のハードディスクすごいよね



さて今回、ハードディスクについてとりあげようと思ったのは、単に僕が新しいハードディスクドライブを入手したからなんだけど、ちょうどよい機会なので、それを導入して、設定する過程で考えたこと、というか、一応少しは役に立ちそうなハードディスクにまつわる話を綴っておきたいと思う。

ハードウェアにしる、ソフトウェアにしる、日進月歩、日々進歩なわけだけど、最近のハードディスクの大容量化にはすさまじいものがある。40Gバイトのハードディスクが2万円を切り、100Gバイトの製品が5万円ちょっとで入手できるのである。1GHzのCPUもそうだけど、まさにちょっと前の夢が現実となっているのだ。

今のところ、メモリは高値安定(今の最適なのかもしれないけど)で停滞気味だし、CPUは小刻みなクロックアップだけで、そもそも最低レンジの500MHzクラスと最高レンジの1GHzクラスでの性能の違いは、一般的な使用において、如実に実感できるほど効果があるものではないときている(確かに速いんだけど、ベンチマーク上はね。でもそれだけなんだよ、CPU ロードの高くないシステムでは)。

ハードディスクとて、そんなサイズが本当に必要なのか、という問題があるわけなんだけど、メモリやCPUに比べてコストパフォーマンスがよいし、利用価値も優れていると

いう面は、強調していいと思う。高価な大容量は、考え直せと指南すべきだが、手ごろな大容量は、手放しで歓迎してもよいものであろう、無理強いするものではないけれど。

で、まあ、今回の結論というか、重要な結実ポイントは、大容量ディスクを使うのにCHSアクセススペースのブートコードはないだろう、LBAベースじゃないとね、ということと、/bootのような動作中1度しかアクセスしないパーティションはディスクの後尾に配置しちゃおう、ということになるんだけど、それだけだと面白くないから、少しハードディスクをめぐる話をしておきたいと思う。



ハードディスクのインターフェイス



まず、ハードディスクのインターフェイスについて。今のところIDEの場合、基本的に互換性を重んじて、40ピンで16ビットデータバスでの接続が使い続けられている。

転送モードに関しては、ディスク性能に合わせて改良が行われ、PIO、DMA、そして現在ではUltra DMA という方式が採用されている。詳細は省略するが、PIOは転送処理をCPUが行うモード、DMAは転送処理をディスクコントローラが行うモード、Ultra DMAではDMAと同様の手法で、さらにダブルデータレート(DDR/ひとつのクロック周期にデータを2回乗せる)でもって、動作周波数をそのままに、2倍の高速データ転送を実現している。

現在は、IDEコントローラ、ハードディスクドライブともUltra DMAが主流で、UltraATA / 33(33MHz / 秒)、UltraATA / 66(66MHz / 秒)、UltraATA / 100(100MHz / 秒)がサポートされている。このUltra DMAの各モードの転送速度の違いは、動作周波数と、データをクロックに乗せるタイミング(サイクルタイム)によるもので、それぞれ120ns、60ns、40nsというサイクルタイムが採用されている。理論的な転送速度の求め方は次のようになる。

$$1/40\text{ns} \times 16 \text{ビット(データバス幅)} \times 2(\text{DDR}) = 100\text{Mバイト/秒}$$

IDEの場合、動作周波数には33MHz(1クロックあたり30ns)を使用している。UltraATA / 33からUltraATA / 66へは、データをやり取りする間隔、クロックカウントを半分にしており、UltraATA / 100では、UltraATA / 66と同じクロックカウントのまま、動作周波数を50MHz(1クロックあたり20ns)に上げることで、サイクルタイムを40nsに縮めている。

UltraATA / 66以降は、80線ケーブルを使用すること

になっているが、これはデータバス幅を広げるためではなく、高シールドのケーブルでノイズによる影響を軽減するためだ。上記のサイクルタイムを見れば納得できると思う。

とりあえず、これらインターフェイスの注意点としては、IDEコントローラとハードディスクドライブの両方が対応していなければそのモードは利用できない、また両方が対応していたとしてもケーブルがきちんとしたものでないと、UltraATA / 66やUltraATA / 100が有効にならない、といったことが挙げられる。多くの場合、80線ケーブルかどうかはBIOSレベルでも検出しているのだが、きちんと仕様を満たした80線ケーブルでないと、たとえ80線であったとしても40線ケーブルとして検出されてしまうことがある。これでは、UltraATA / 66やUltraATA / 100を有効にすることはできない。単に80線ケーブルというだけの粗悪なケーブルが多く出回っているようなので、十分に注意する必要があるだろう。

Linuxでの対応

LinuxでUltraATAを利用するには、各デバイスドライバが対応していることが必須となる。これには、カーネルのパッチレベルを上げるなどの対処が必要となる場合もある。

さらにもう一点、忘れがちなんだけど、一番重要な転送モードのネゴシエーションの問題にも気をつけたい。最近のカーネルは、DMA転送を有効にするようコンパイル時に指定できる。要するにハードディスクの検出結果に応じて、転送モードを自動的に設定できるようになるわけだが、これが意図しないモードに設定されてしまうことがある。UltraATA / 66までに対応したIDEコントローラにUltraATA / 100対応ハードディスクドライブを接続した場合、調停によってUltraATA / 66での転送モードが適用されるべきなのだが、デバイスドライバによっては、ハードディスクのUltraATA / 100対応フラグの処理が適切でなく、PIOモードを選択してしまうことがあるのだ。

そこで、新しくディスクを接続した場合は、Dmesgコマンドやhdparmコマンドで、どのモードが実際に適用されているか必ず確認するようにしたい。

たとえば、UltraATA / 100が可能な環境で、dmesgの内容が次のような場合は、UltraATA / 33モードが適用されているようなので、hdparmコマンド(-Xオプションなど)で調整してみるとよいだろう。

```
ide: Assuming 33MHz system bus speed for PIO modes;
override with idebus=xx
```



```
ide0: BM-DMA at 0xfcb0-0xfcb7, BIOS settings:
hda:DMA, hdb:pio
hda: IBM-DTLA-307030, ATA DISK drive
hda: 60036480 sectors (30739 MB) w/1916KiB Cache,
CHS=3737/255/63, UDMA(33)
```

ちなみに、UltraATA / 100は、UDMAモード5に該当するが、これをセットするには、hdparm-3.6以降が必要となる(-X 69でUDMAモード5の指定を意味する)。コマンドのバージョンを確認して、必要に応じて最新版を入手しよう(<ftp://sunsite.unc.edu/pub/Linux/system/hardware>など)。最新は3.9aだったと思う。

ざっと転送モードの説明をしたが、これはハードディスクドライブの読み書き性能をきちんと引き出せるかどうかの問題に関わってくる。現在ディスク1枚あたり10Gバイト~20Gバイトの記録密度となっているのだけれど、10Gバイトであれば、だいたいドライブ性能としては、30Mバイト/秒前後なので、UltraATA / 33でも何とか物理ドライブ性能を犠牲にすることなく操作できる。しかし、それ以上の記録密度のディスクを採用したドライブでは、性能が40Mバイト/秒前後まで向上しているのので、UltraATA / 33では、インターフェイスが完全に足を引っ張ることになる。

じゃあ、UltraATA / 100が必須かということ、単発のドライブ性能では、そこまではまだ言い切れる状況ではない。でも、同じインターフェイスのマスタとスレーブに接続して、同時に大量のアクセスが発生する場合は、UltraATA / 100でないといけないわけだが、普通のユーザーがそんな使い方をするとは思えないので、UltraATA / 66をサポートしている環境であれば、無理にUltraATA / 100にすることもないだろう。ただ、UltraATA / 33までしかサポートしていない環境では、新しいハードディスクを購入するときに、IDEコントローラもいっしょにUltraATA / 100対応にすべきであることは間違いないと思う。

ちなみに、UltraATA / 66からUltraATA / 100のように転送モードが向上すると、実際の性能がどのように変化するかは気になるところだろう。今回はテストを紹介しないが、結論だけ言っておくと、レスポンス性能は確実に向上する、しかし、ユーザーがもっとも気にするいわゆる転送速度は微々たる向上でしかない、ということになる。UltraATAの場合、各モードで異なるのは、サイクルタイムだけなので、このような結論となるわけだ。

あと、各モードの転送レートは理論値なわけで、実際にはどの程度まで性能が引き出せるかという点に関しては、PCIバスの性能によるものの、理論値の90~95%程度は実レートでも到達可能なようだ。



ハードディスクのコマンドインターフェイス



転送モードに関しては、さほどユーザーが“制限”を感じる部分ではない。では、ユーザーが“制限”を感じるのとはどんなことだろうか。たぶん、もっとも制限を感じるのは、ブート制限やサイズ制限であろう。

これらはすべてハードディスクのコマンドインターフェイスが関係している。

ハードディスクに対してアクセスを行う場合、ユーザーはIDEコントローラにコマンドを発行し、そのコントローラから該当するディスクドライブにコマンドが渡る。コマンドは、セクタとその操作を指定するものだが、ここで問題となるのはセクタの指定方法だ。

現在、ハードディスクドライブに対するセクタ指定方法には、CHSアクセスモードとLBAモードの2種類がある。

CHSアクセスモード

セクタ(C)、ヘッド(H)、シリンダ(S)の3つのパラメータで目的のセクタを指定する方法。ドライブのインターフェイスでのパラメータの上限は、それぞれ63、16、16383。つまり、アクセス可能範囲は、 $63 \times 16 \times 16383 \times 512$ (セクタサイズ) = 約8Gバイト。これを超える位置のセクタには、CHSアクセスモードではアクセスすることができない。

LBAモード

論理ブロックアクセスモード。セクタに論理的な通し番号を割り当てて、目的のセクタを指定する方法。現状は、28ビットの通しセクタ番号が使用されている。つまり、アクセス可能範囲は、 2 の28乗 $\times 512$ (セクタサイズ) = 128Gバイト (ハードディスク業界では1Kを1024ではなく、1000で計算することが多いので、137Gバイトの制限とも呼ばれる)。僕自身、少し前までLBAのパラメータは、32ビット長だと思っていたが、実はATA規格では28ビット長。容量100Gバイトのディスクドライブがすでに製品化されているというのに、かなりヤバイ状況。近々にATAでLBAのパラメータを拡張した規格が策定される模様(聞くところによると48ビット?程度にするとか)。

これらは、ハードディスク自体のコマンドインターフェイスの問題で、実際にユーザーが直接制限にぶち当たるのは、コマンドを発行するフロント部分、BIOSルーチンとデバイスドライバルーチンということになる。

CHSアクセスモードの8Gバイトの制限は、ブート制限としてよく知られているので、BIOSルーチンはCHSアクセスモード、OSのネイティブデバイスドライバルーチンはLBAモード、とと思っているユーザーも少なくないと思うが、これは間違い。BIOSとデバイスドライバに関係なく、CHSアクセスとLBAモードのどちらかを選択して利用することができる。よく勘違いされているのは、デバイスドライバのLBAモードへの対応は迅速だったにも関わらず、BIOSルーチンを利用するブートコードの対応が後回しにされたためだ。だから、1024シリンダの問題が根強く残されてしまっているのだ。

さて、先ほどのCHSアクセスモードでは、シリンダのパラメータ制限は16383なのに、パーティションを構成する際には、1024シリンダの問題と呼ぶ。これはどうしてだろう。

実は、ディスクドライブのCHSアクセスモードのコマンドパラメータと、BIOSルーチンに対して行うCHSアクセスモードのコマンドパラメータは、歴史的な経緯から異なっているのだ。

本来のCHSアクセスモードのパラメータ制限は、先ほど説明したとおりなんだけど、BIOSルーチンでは、パーティションテーブルの構成情報との整合性もあって、63セクタ、255ヘッド、1024シリンダというパラメータ制限になっている。BIOSルーチンでは、これを本来のCHSアクセスモードのパラメータに変換してコマンドを発行しているというわけだ。



とりあえず全部LBAモードを利用すれば悩まなくて済む



ディスクアクセスにLBAモードを利用することで、下らないブート制限もディスクが128Gバイトを超えるまでは気にしなくて済むようになる。古いOSはともかく、現状のOSでデバイスドライバがLBAモードに対応していないものはないので、ここではブートシーケンスにだけ注目したいと思う。

ここでいうブートシーケンスは、マシンに電源を投入してからOSのコアコンポーネントがロードされ、実行処理がそちらに移管されるまでの処理のこととする。OSに実行処理が移されれば、ディスクアクセスもネイティブなデ

バイスドライバ経由なので特に問題はない。

PCの場合、ブートシーケンスはすべて16ビットのリアルモードで行われ、ディスクアクセスにBIOSルーチンを利用することがほとんどだ。ブートコードに専用デバイスドライバを使用しているケースは希である。

BIOSルーチンにLBAモードを利用するディスクアクセスファンクション(Int13エクステンション)が追加されたのは、5年近くも前のことなので、現在使用されているPCのほとんどが対応しているといっても問題はないと思う。ただ、今でもLinuxはi386やi486でも実用的に使えることがウリという面もあるので、古いBIOSの場合は少し注意したほうがいだろう。



ブートコードの役割



Linuxでのブートコードの役割は、通常、目的の圧縮カーネルファイルをメモリにロードして実行することだ。Linuxには、いろいろなカーネルローダプログラム(ブートコード)が存在するが、僕はLILOとSYSLINUX以外は触ったことがないし、特に市販のOSローダプログラムを使いたいと思ったこともなく、またその必要性もなかったのだ。それらのLBAモードへの対応状況がどうなっているかはよくわからない。調べてみようかとも思ったのだが、今回はあきらめた。というわけで今回はLILOのみ取り上げることにする(でもまあ、Linuxユーザーの多くは、LILOを利用しているだろうから問題はないよね、きっと)。

LILO

LILOの場合、2段階のブートコードに分かれていて、まず最初のブートコードでは、次のブートコードと必要なファイルの在りかを記録したマップファイルをロードして実行する処理を行う。次のブートコードは、ブートローダ本体で、設定された情報に基づき、カーネルなどを選択する機能をユーザーに提供するとともに、最終的にそのカーネルをロードして実行する処理を行う。これらすべての段階で、ブートコードは、ファイルシステムを経由せず、その存在をいっさい意識しない。つまり、セカンドブートコード、マップファイル、カーネルなどのファイルは、ディスクのジオメトリ情報として扱われるようになっている。すなわち、ここで、CHSアクセスモードであれば、その3種のパラメータ情報、LBAモードでは論理セクタ番号での指定ということになる。

LILOでは、最初のブートコードをMBRに導入するか、

パーティションブートセクタに導入するか選択できるが、これによって上記の処理が変わるわけではない。MBRに導入した場合は、システムによって最初にロードされるのがLILOのブートコードになる。パーティションブートセクタでは、標準のMBRブートコードでパーティションテーブルの解析が行われ、ブートフラグのついたパーティションのブートセクタが読み出され実行されることになる。ブートフラグのついたパーティションのブートセクタにLILOの最初のブートコードが導入されていれば、上記の作業が継続され、そうでなければ別のブートコードが処理を行うことになる。よくLILOをMBRに導入する際には気をつけると言われるが、僕はむしろパーティションブートセクタに導入するほうが、面倒が増えて嫌だなと常々思うんだけど…。でも、実際問題として、LILOがLBAに対応したのはつい最近のことで、Windows 9xのLBAモードで管理されているディスク（パーティションテーブル）のMBRにLILOを導入するのは、トラブルの原因であった。

Windows系ブートコード

DOS / Windows系OSの場合のブートコードは、基本的に共通の処理を行う。MBRのブートコードでパーティションテーブルを解析して、ブートフラグのついたパーティションのブートセクタをロードし実行する。次にパーティションブートコードによって、Windows 9xの場合はIO.SYS、Windows 2000の場合はNTLDRをロードし実行するのである。特にNTLDRは、自身に専用のデバイスドライバを含んでいるので、この時点でBIOS経由のディスクアクセスではなくなる。

では、現状これらのブートコードがどこまでLBAモード対応になっているかということ、Windows 95 OSR2以降、Windows 2000は標準でサポート、LILOは2000年3月にリリースされたバージョン21.3以降でのサポートとなっている。逆に言うと、Windows NTやLILO 21では対応していないということだ。これらを利用する場合は、残念ながら昔のように1024シリンダのことを引きずり続けなければならないということである（ご愁傷様）。

Windows NTとWindows 9xを同居させる場合、Windows NTだけでなく、Windows 9xのインストールパーティションも8Gバイト以内に作成しなければならなかったが、これはすべてWindows NTのブートコードがLBAに対応していなかったことに起因する。一方に問題がなくても、主となるブートコードに問題があるとダメというトホホな一例だ。

LILOの場合も同様にLBAモードに対応するまでは、通常/boot以下のカーネル/ブートファイル群を8Gバイト以下に配置する必要があった。

しかし、これらのブートコードがLBAモードに対応したことで、少なくともブート制限は緩和され、パーティション設計の自由度も増したわけだ。

なお、LILOの場合、明示的にLBAモードで処理を行うことを指定しなければならないので注意したい。LILOの最新バージョンはftp://brun.dyndns.org/pub/linux/liloなどで入手可能。9月上旬時点の最新バージョンは21.5.1。



INT 13 と INT 13 エクステンション



INT 13というのは、ディスク操作のBIOSコールファンクションで、ブートコードではこれを利用してディスクにアクセスしている。基本的にはCHSアクセスモードでのパラメータ指定なのだが、LBAモードをサポートするために拡張され、それらはINT 13エクステンションと呼ばれている。

たとえば、ハードディスクの先頭セクタを読み出すには次のようなコードが実行されている。

・ CHSアクセスモード

```
MOV DL,80      #ドライブ番号
MOV DH,00      #ヘッド番号
MOV CH,00      #シリンダ番号
MOV CL,01      #セクタ番号
MOV BX,1000    #ES:BX / データバッファ
MOV AL,01      #セクタ数
MOV AH,02      #セクタリードファンクション
INT 13
```

・ LBAモード

```
MOV DL,80      #ドライブ番号
MOV SI,??      #DS:SI / ディスクアドレスバケット
MOV AH,42      #拡張リードファンクション
INT 13
ディスクアドレスバケット (開始ブロック番号 / データバッファ /
ブロック数)
```

```
00h   BYTE   10h (size of packet)
01h   BYTE   reserved (0)
02h   WORD   number of blocks to transfer
04h   DWORD  -> transfer buffer
08h   QWORD  starting absolute block number
```



LILOのLBA設定



LBAモードを使用したLILOを導入するには、設定ファイルにlba32オプションを指定するだけでよい。もちろんLBAモードに対応したバージョンであることが必須条件だが、これ以外に特別なことは必要ない。このオプションにより、インストーラであるliloコマンドを実行した際に、カーネルファイルなどのジオメトリ計算がLBAベースで行われ、その形式でマップファイルに記録されるようになる。

例：LBAを指定したlilo.conf

```
boot=/dev/hda
lba32                #この指定を行うだけ
root=/dev/hda5
image=/boot/vmlinuz
    label = linux
image=/boot/vmlinuz-2.4.0-test7
    label=old
other=/dev/hda1
    table=/dev/hda
    label=win
```



LBAにこだわる理由



僕は、基本的にひとつのマシンに複数のOSをインストールしない。だから、特定のOSローダに特別な思い入れがあるということもない。なすがままというか、標準として導入されるものをそのまま利用することが多い。

でも、テストなどでどうしてもインストールしなければならない場合は、中心にLILOを据え、MBRに導入して利用するか、Linuxはフロッピーブートで、というスタンスを取っている。一応、ある程度長い期間利用するのであれば、LinuxとWindows 2000とWindows 98の場合、LILOでLinuxカーネルとWindows系のパーティションブートセクタとの選択を行い、さらにWindows 2000のOSローダでWindows 98を選択するという、もっともオーソドックスなスタイルをとる。

ではなぜLBAにこだわるのか。

理由は3つある。パーティションを設計する際、ブート制限やパーティションサイズを気にしたくないことと、複数のOSを同居させる場合にパーティションテーブルの解析でトラブルを引き起こさないため、そして/bootなどの

アクセス頻度が低いパーティションをディスクの後尾に配置するためだ。

OSだけでなく、そのブートコードにも注意していないと、LILOから8Gバイトを超える位置にインストールしていたWindows 9xのパーティションブートセクタを読み出すことができない、なんてことになってしまう。こんな事態には、陥りたくないものだ。

また各パーティショニングツールによって、パーティションの構成方法が微妙に異なる点にも注意したい。大容量ディスクでもCHSをベースにパーティションサイズをアライメントする場合（シリンダ単位で調整する）と、LBAベースでより細かな単位でアライメントを行う場合があるのだ。Linuxのfdiskコマンドなどのデフォルトはシリンダベース、一方Windows 9xのFDISKコマンドは1Mバイト単位だったりするので、これらを適当に織り交ぜながらパーティション作成を行うと、重複する部分が生じる可能性があるのだ。

でもさあ、いい加減、ブートコードがLBAモードに対応しているかどうかなんて気にしなくてもいい時期に来ていると思うんだけど。なぜCHSが蔓延しているんだ？



大容量ディスクでのLinuxの憂鬱



僕は、Linuxで大容量ディスクを扱うにあたり、かなり憂鬱な思いを抱いている。それは標準ファイルシステムExt2があまりにも脆弱だからだ。もちろん個人的な意見なんだけど、大きなサイズのパーティションを管理するのに、ジャーナルログの実装されていないファイルシステムは使いたくないし、Ext2は構造的に見ても大量のファイルや大きなファイルの扱いが得意であると到底思えない。十分実用であることは認めるが、今となっては古臭い仕様のファイルシステムでしかない。これで100Gバイトのディスクを管理するなんて、僕はやりたくないなあ。

Linuxのファイルシステムといえば、xfsは何しているかよくわからないし、Ext3もよくわからない。現状だとReiserファイルシステムが目に見える形で実装されていて、利用することができるが、僕の中では、まだテスト段階を脱していない。困ったものである。

まあ僕の環境がテスト環境で、本来のLinuxの安定性とは関係なく、よくシステムを落としてしまっているからなんだけど、ファイルシステムがひ弱なのは、早急にかしたいと思っている。Reiserファイルシステムを実運用環境にしちゃおうかなあ。でも...なあ。

「愛をください」を録画できるのだ……。今日は自分にごほうび。西新宿でUltraDMA/100 80Gバイトのハードディスクを買っていこう。と、早々に退社。

22:30 新品のハードディスクをつなげたら、負荷がかかるとシステムがフリーズを起こすようになってしまった。「しかたないなあ」とつぶやきながらも嬉々としてソースのチェックをはじめ。そのうちに気絶（翌日もこの日と同じ行動パターンをたどる）

編注：そんなやつ、いないってばよ。

スーパーユーザー

【すーぱーゆーざー】

(1) 主に中年以上の家庭に入った女性で構成される人種。ふだんは情眠をむさぼってパワーをためているが、いざというときには驚異的な破壊力と厚顔無恥さを見せるのが特徴。毎朝、新聞に入っているチラシのチェックは欠かさず、半径25km以内のバーゲン



無敵... (サーチ中)

情報は完璧に把握している。この行動力と情報リテラシはLinuxユーザーも見習いたい。行きつけの店によって“スーパーライフユーザー”“スーパー丸正ユーザー”などに細分類される。この細分類によると、ユーザーの分布には激しい偏りがあることがわかっている。たとえば、“スーパースーパーユーザー”が多い地域もあれば、“スーパーマルエツユーザー”の多い地域もある。

(2) rootなど、管理権限をもつユーザー一般。あのスーパーマンを例に出すまでもなく、“スーパー”がつく名をもつとなにかと期待されてしまうが、大自然と運命の前には誰しも無力。スーパーマンもクリプトナイトにはかなわなかったではないか。人たるもの、大きな流れの中ではしよせん身をゆだねるしかないのだ。諦観を知れ。などといういいわけを、サーバクラッシュでデータがロストしたときにユーザーに向かって力説すると、かなり高い確率で殴られる。

(3) 日本向けにローカライズしたディストリビューションでは「ウルトラユーザー」と改名したほうが親しみやすいという主張も見受けられるが、3分間しか働かなかったり意味もなく巨大化して周辺地域に甚大な被害を与える可能性があるのですが却下された。日本のスーパーユーザーには横方向へ巨大化した人物が多いのは、この議論が尾を引いているためだとされている。

sudo

【すどー】

須藤さん。雨の戒橋で泣きじゃくりながら走り去っていった彼女。どうしているのかなあ。今なら2人でやりなおせるような気がする。連絡待ってます。

SU

【す】

(1) CH_3COOH の3～5%の水溶液。原料は融点16.6、沸点117.8。室温での密度は1.049g/cm³。「4類・引火性」に該当する。刺激臭がするので使用の際は保護手袋、眼鏡着用のこと。保存の際は密栓を忘れずに。新鮮な鶏卵、サラダオイルをまぜ、激しく攪拌するとおいしいマヨネーズになる。



酢酸

(2) Linuxで、ログイン中のユーザーが一時的に別のユーザーになりすますためのコマンド。rootになる时候にも利用する。たぶん、酢を飲むとカラダが別人のように柔らかくなる。このようなコマンド名がつけられたのだと思われる。なお“showu”というコマンドはない。

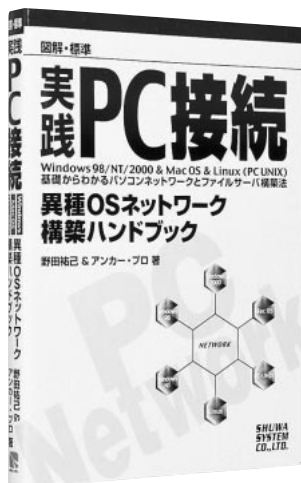
wheel

【ほいーる】

一部UNIXにビルトインされている、特別なユーザーグループ。このグループに所属するユーザーだけがsuでrootになれるしくみになっている。英語で言う「車のハンドル」が転じて（ハンドルを握ってものごとを操るような）大立者・黒幕を指すところから来ている用語だが、ハンドルを預けているのだと考えるとゾットするようなwheelユーザーが身の回りに多すぎるので、この由来は忘れたほうがよい。

wheelによる制限を利用すると、/etc/groupを見るだけで誰がrootのパスワードを知っているか、共用マシンが不調でイライラしているとき誰に向けてストレスを発散すればいいかなどがすぐにわかってしまうため、特にスーパーユーザーの労働負荷軽減の立場からはきわめて具合が悪い。そのため、wheelによる制限を採用していないシステムも多い。ただしGNU suがwheelに対応していないのは、ストールマン導師の、MITでwheelに入れてもらえず仲間はずれにされたトラウマがまだ癒えていないため。GNUではセラピストからの労働ドネーションを期待している。

Books



図解標準 実践PC接続 異種OSネットワーク構築ハンドブック

野田祐己&アンカープロ 著

秀和システム

B5判 / 340ページ

本体価格 2100円

特にマニアでなくとも、PCを複数台所有しているユーザーは結構多い。昨今のようにネットワーク機器の価格がリーズナブルなものになってくると、持っている複数のPCをLAN接続したくもなる。Linuxオンリーという場合を除いて、ファイルやプリンタを異種OS間でネットワーク共有したいというニーズも出てくるだろう。

本書はそういったニーズに応えるものだ。取り上げるOSは、Windows (98/NT/2000)、Mac OS、そしてLinux。掘り下げるといくらでも難しくできそうなテーマだが、あくまでベーシックな部分に絞っており、Samba/Netatalk/PC MACLANを使った混在ネットワーク環境作りの基礎的なノウハウを身に付けられるようになっている。ネットワークの基礎と各OSでのネットワーク設定についても解説されているので、ネットワーク構築が初めてという読者でもひと通りの設定ができるはずだ。

オープンソースソフトウェアによる 全文検索・データベースWebの作り方

西村めぐみ 著

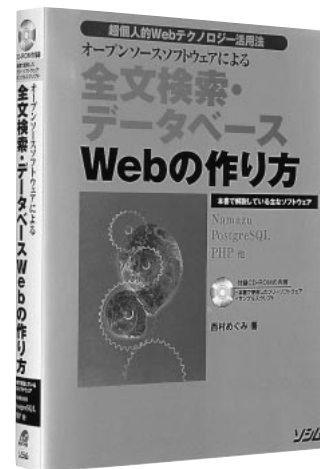
ソシム

B5変形判 / 396ページ / CD-ROM1枚付き

本体価格 2600円

Linux関連の情報をインターネット上で検索していると、メーリングリストをアーカイブしたサイトなどで「検索にはNamazuを使っています」と記述されていることがある。Namazuは日本語に対応したオープンソースの全文検索エンジンで、ローカルマシン上での検索ならtkNamazuというツールのみでOKだが、Webベースの検索を行うためにはCGIなどと組み合わせた「仕掛け」が必要となる。本書が扱うのはこの仕掛けの部分。

Namazu + CGIによる検索サイトのほかに、やはりオープンソースのPostgreSQL (データベース)とPHP (スクリプト言語)を使ったWebベースのデータベース操作についても解説している。タイトルにあるようにオープンソースソフトウェアだけを使って、検索/データベースWebを作ってしまうというのがメインテーマだ。付録CD-ROMには、上記に加えてApache、MHonArcなど本書に登場するソフトウェアとサンプルスクリプトが収録されている。



実践Linuxセキュリティー

すずきひろのぶ 著

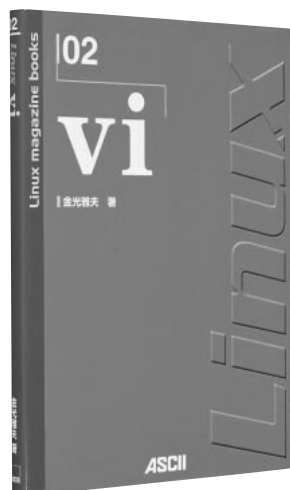
インプレス

A5判 / 224ページ

本体価格 1700円

このコーナーでもセキュリティ関連の書籍を毎月のように紹介しているし、大型書店の売り場に行っても、ずらりと「セキュリティ本」が並んでいる。それだけネットワークセキュリティへの関心が高まっているわけで、裏を返すとそれだけ何らかの問題が(犯罪も含めて)ネットワーク上で起こっているのだろう。にもかかわらず、無関心な人はやはりいるようで、本書で紹介されている「CATV経由でインターネット接続したら、よその家の共有ディレクトリが丸見えだった」というエピソードに似た話は、いまだに結構耳にする。

CATVやNTTのフレッツ・ISDNなどの常時接続サービスが普及すれば、このような無防備な人々がますます増えると思われる。本誌読者にはそうした事態はぜひとも避けてほしい。Linuxの利点を生かして、信頼性の高いセキュリティの実現を目指そう。本書は、そのためのガイドラインとして最適な1冊だ。



vi

金光雅夫 著

アスキー

A5判 / 152ページ

本体価格 1500円

タイトルそのまま、まるごと1冊viの解説書である。入門編である第1章を読めばviの操作をひと通り身に付けられる。第2章で紹介されているTipsと第3章の応用技をマスターすれば、あなたも立派な「vi使い」だ。第4章ではviクローンであるvim、第5章では日本語環境の設定を取り上げている。特に第4章で紹介されているvimの非vi互換モードでの拡張機能が興味深い。ふだんはviのふりをしているvimだが、実はGUIモード(カット&ペーストにも対応)やマルチウィンドウ処理といった機能も隠し持っているのだ。

UNIXの標準エディタとしておなじみのviだが、最近では入門書の一部を割いて解説されている程度で、専門の解説書にはあまりお目にかからなくなった。しかしながら、以前本誌で連載していた「viはじめました」がご好評をいただいたように、viの知識を必要としているLinuxユーザーは少なからず存在するのだと思う。そういう方にお勧めしたい。

Windowsユーザーのための
Linuxシステム管理ガイド株式会社リーディング・
エッジ 著
株式会社プロトスター 監修

リックテレコム

B5変形判 / 468ページ

本体価格 3300円

Apache拡張ガイド 上・下

Lincoln Stein, Doug
MacEachern 著
田辺茂也 監訳
田和 勝 訳

オライリー・ジャパン

B5変形判 / 552 & 312ページ

本体価格 4800円 / 3200円

PHPプログラミング入門

レオン・アトキンソン 著
玉川竜司 訳

ピアソン・エデュケーション

B5変形判 / 530ページ

本体価格 4700円

一晩で学ぶSQL

Ben Forta 著
篠原 慶 訳

インプレス

B5変形判 / 242ページ

本体価格 1900円

読者の声

俺にも
いわせろ!

某取締役に「なんか疲れた顔してるな。ドリンク剤を買ってきてやる」と言われ、しばらくすると本当にドリンク剤が届きました。そんなにやつれて見えたのでしょうか。ちょっとショック。

10月号特集1へのお便り

/etcの解説を.....と思っていたら、思いは通じました。これからはいろいろと念じてみますのでよろしくお願ひします(笑)。

P.S. patchの解説をお願いします。内部動作まで踏み込んだものを。

(秋田県 小松弘尚さん)

UNIXのシステムによって意外と違っている/etcの特集にひかれて購入しました。Solaris、IRIX、BSD、Linux等々コマンドラインでシステム管理を覚えた者としては、GUIで設定できてもファイルを見ないと安心できません(もう昔の人なののでしょうか?)。

(神奈川県 菅達樹さん)

/etc以下の解説特集、よかったです。次はproc以下の解説をお願い致したく御座候。

(千葉県 山田兼嗣さん)

この特集の企画は昨年うちにできあがっていたのですが、諸般の事情によりなかなか実現できなかったものです。この特集

にはたくさんのおハガキをいただき、多くの方のお役に立てたと担当者一同大喜びをしています。

/proc、/bin、/libダンジョンについてもリクエストをいただいていますし、今後もLinuxの謎に迫る記事をお届けしていきます。

カーネル2.4 リリース間近

カーネル2.4 + XFree86 4.0を採用したCaldera Linuxテクノロジープレビューの動作が軽いので、正式リリースされるのが楽しみ!!

(大阪府 川合博さん)

カーネル2.4が話題ですが、個人使用 & 小規模LAN程度なら、現在の2.2.16 + USB + ACPIで十分と思いません。機能拡張の結果、不安定になったのではどうしようもないですから。

(滋賀県 池田浩樹さん)

カーネル2.4のリリースももうすぐですね。USBなどのデバイスサポートのほか、ネットワーク周りなどもコードがリファインされているようです。

正式版が発表され、細かいバグがつぶされていけば、2.4が主流になるのは当然ですが、それまでは池田さんのおっしゃるとおり2.2系カーネルを継続使用するのも賢い選択といえるでしょう。

ラジオはLinuxでデジタル録音

「ラジオ自動録音システム」は大変参考になった。私も語学講座をタイマーでカセットテープにほぼ毎日録音している。増え続けるテープの数に頭を悩ませていた。さっそく取り入れたい。おっと、その前にALSAで音を鳴らさなければ.....。

(千葉県 hidさん)

「ラジオ自動録音システム」はよくぞやったという感じ。ここまでやれば十分でしょう。このシステムを組む過程でLinuxも覚えられて一石二鳥。すばらしい。

次は「テレビ自動録画システム」だね。楽しみにしています!

(東京都 石川信幸さん)

今回の「ラジオ自動録音システム」は参考になりました。サウンドって、スピーカーを通して音楽を聴くだけだと思っていたのに、ラジオの録音、特に英会話レッスンを記録して、管理しながら使っていくとするのは、パソコンもこんな使い方があるのだと再認識した次第です。

(千葉県 宮本邦彦さん)

自動運転はUNIX系OSの得意とするところですね。とはいえ、担当もこのシステムには驚きました。編集長がテレビの自動

録画システムを欲しがっていますので、完成の暁には記事をお届けできる……できますよね? > 編集長

クラスタで分散 コンピューティング

10月号から始まった「2台から始めるLinuxクラスタ」を楽しく読みました。続きに期待します。

(愛知県 國枝信吾さん)

㊦現在、最も手軽にクラスタを組めるOSはLinuxでしょう。大学や研究機関でもLinuxによるクラスタシステムを導入しようという気運が高まっているようです。今後さらに注目される分野ですね。

リンゴのVineをお届けします

え~、毎回同じことを書いてしまっただけなんですが、MkLinuxのCD-ROMを付録に付けてください。できたら、特集も組んでください。こんなお願いをするのは私くらいで、ごく少数だとは思いますが、その少数にも愛の手をさしのべてください。

(長野県 松村 岳さん)

近々、Mac版のVine Linuxをインストールしようと思っているのですが、PPC版ディストリビューションの記事が少ないので増やしてほしいと思っています。

特に、PPC独特のソフトに関する記述がほしいと思っています。

(東京都 安部泉典さん)

㊦Macintoshオーナーのみなさん、お待たせしました。今月はVine Linux 2.0 for PowerMacをお届けします。

MkLinuxはPowerMacで動作するマイ

クロカーネル版Linuxディストリビューションです。現在ではLinuxPPC系のモノリシックカーネルが主流になりつつあるので、MkLinuxをお届けするのは難しいかもしれません。

知らないところで海外進出

シンガポールでもLinux magazineを売っていました。

(神奈川県 西川賢一さん)

㊦わお! 海外でも売っているんですね。編集部も知らなかった秘密情報をありがとうございます。実は、韓国では提携雑誌が出版されています。名前も同じLinux magazineで、弊誌の記事も一部翻訳されて掲載されています。

Linuxでさめがめ、ゴルゴ

父がインストールしたTurboLinux Workstation 6.0の“さめがめ”で遊んでみました。

BSDのフリーソフトをLinuxにインストールする方法を詳しく教えてください。たとえば、“ゴルゴの目玉”。

(大阪府 津田桃子さん)

㊦さめがめは大変危険なゲームです。習慣性が極めて強いので、仕事をそっちのけにしてハマるのです。担当は重度の中毒患者を数名見てきましたが、社会復帰も危ぶまれるほどの入れ込みようでした。お気を付けてください。

“ゴルゴの目玉”とは、xgolgogoのことで、マウスポインタを追いかけるアクセサリxeyesはみなさんもよくご存じでしょう。xgolgogoは、ゴルゴ某の目がマウスポインタを追いかけるものです。ペンギン活用委員会 (http://www.pcnix.org/linux_soft/) で検索するとLinux用のものがあるようで

す。有名なソフトウェアはほとんどLinuxでも動作しますので、ソフトウェアアーカイブサイトで探してみましょう。

余談ですが、2000年1月号の42ページにあるデスクトップ画面でxgolgogoを見ることができます。担当がこっそりとおきました。1月号をお持ちの方はご覧ください。

他誌読者欄で発見されて

毎度!“やっぱホゲホゲ遠藤”です。とうとうチクリがあったか、うーん。他社の雑誌を読んで何が悪い! 他誌を数冊読み比べて初めて、リナマガの良さがわかるというもの(ほとんど逆ギレで開き直り状態です)。

今、これを読んでいるあなた、もしかしてあなたにも身に覚えがありませんか? あるでしょうー、ねっ? もういいわけはしません。素直に認めようではないか。

御代官様、申しわけございません。ぜひお慈悲をおかけくださいませ。ねっ、福島香西さん、一緒に謝ろう(僕らの謝罪がないと、編集部は怒ってこのハガキを捨てるかもしらん…)

(山形県 やっぱホゲホゲ遠藤さん)

㊦(^.^);; 謝っていただく必要なんてちろんありません。ハガキを捨てるなんてバチあたりなことは絶対にしません。みなさんに選んでいただけるLinux magazineとなるべく、さらに誌面を充実させてまいりますので、今後ともよろしく願いいたします。

担当はいつかこのコーナーをカラーページに昇格させたいという野望を抱いているのですが、政治力不足ゆえなかなか達成できそうにありません。

付録CD-ROMに収録した TurboLinux Workstation日本語版6.0 (FTP0915版)のインストール

本誌付録CD-ROM収録のTurboLinux Workstation 日本語版6.0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ターボリナックスジャパン株式会社からサポートを受けることはできません。

インストーラの起動

CD-ROMブートができないマシンにLinuxをインストールする場合は、WindowsマシンにCD-ROMと空のフロッピーディスクをセットして、エクスプローラでCD-ROMの中にある「DOSUTILS」フォルダを開き、「BOOT.BAT」をダブルクリックして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（画面奥）。

作成したフロッピーやCD-ROMをセットしマシンを再起動してインストーラを起動します。「boot:」プロンプトが表示されたら[Enter]を押します（画面手前）。



キーボードタイプの選択

使用するキーボードを選択します。デフォルトで「jp106」が選択されているので、日本語キーボードを使用するユーザーは「OK」を押して次へ進みます。

次はPCMCIAの設定です。デスクトップマシンを使用するユーザーは「いいえ」を、ノートPCのユーザーは「はい」を選択します。



インストールメディアの選択

ここでは付録CD-ROMを使ってインストールするので、「CD-ROMドライブ」を選択して「OK」を押します。次の場面（画面奥）では「OK」を押します。

ネットワークの設定

家庭内ネットワーク環境にLinuxマシンを参加させると想定します。家庭内でISDNルータやほかのサーバマシンでDHCPサーバを稼働させている場合は、「DHCPで設定する」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	192.168.1.1 ~ 192.168.1.254
ネットマスク	255.255.255.0
ゲートウェイ(IP)	家庭内サーバマシンのIPアドレス
DNSネームサーバ	プロバイダのDNSサーバのIPアドレス

ここでは「DHCPで設定する」を選択します。

パーティションの作成

マシンをLinux専用にする場合は、Linux用のパーティションを自動作成してくれる「オートパーティショニングモード (without Software RAID)」を選択するとよいでしょう。なお、このモードを選択すると、ハードディスク内の既存データはすべて消去され、ブートローダLILOが自動でMBRにインストールされます。LILOのインストール先を手動で選択する場合、1台のマシンにLinuxとWindowsなど複数のOSを共存させる場合は「マニュアルモード」を選択して、手動でパーティションを作成します。

「オートパーティショニングモード (Software RAID)」はRAIDの意味がわかるユーザーのみ選択してください。

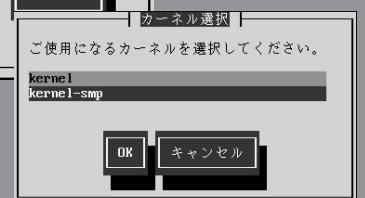
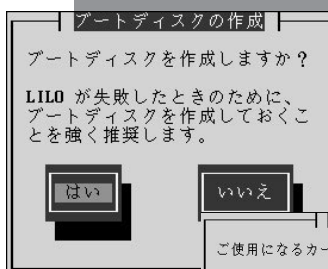
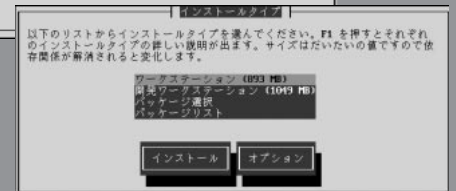
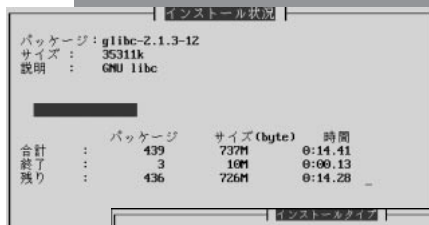
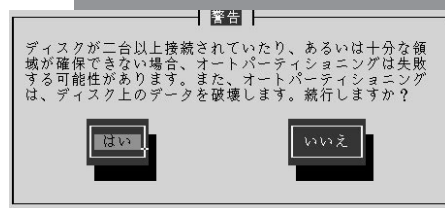
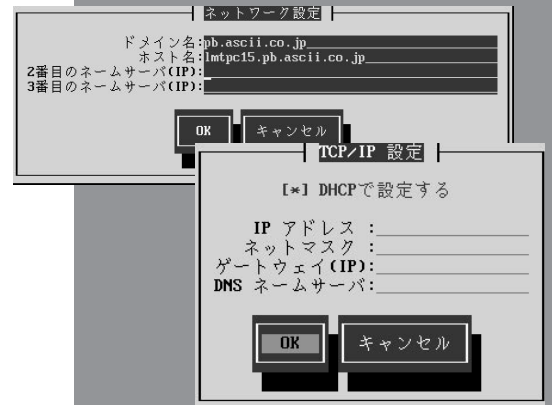
インストールタイプの選択

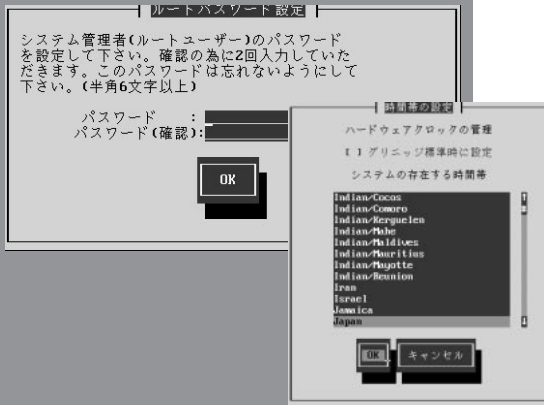
まずインストールタイプを選択します。インストールしたいパッケージ、そうでないパッケージを決めているときは「パッケージ選択」を、ある程度おまかせでパッケージをインストールするときは「ワークステーション」か「開発ワークステーション」を選択します。gccなどのコンパイラを使う場合は後者を選択するとよいでしょう。「パッケージリスト」は、ユーザーがあらかじめインストールするパッケージリストを作成している場合に選択します。いずれかのインストールタイプを選択して「インストール」を押すと、パッケージのインストールが始まります。

カーネルタイプの選択

パッケージのインストールが終わると、次はカーネルタイプを選択します。使用するマシンに搭載されているCPUが1個の場合は「kernel」を、2個以上の場合は「kernel-smp」を選択します。

次にLinuxがハードディスクから起動できなくなったときにそなえて、ブートディスクを作成します。「はい」を押してフロッピーディスクの作成を始めましょう。

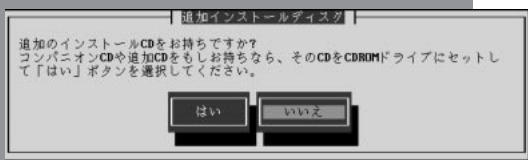




タイムゾーンとパスワードの設定

時間帯 (タイムゾーン) を設定します。Linuxを日本で使う場合はデフォルトのまま「Japan」を選択します。また1台のマシンでWindowsと共存させる場合は、画面のように「グリニッジ標準時に設定」のチェックをはずして「OK」を押します。

次にLinux管理者用のパスワードを設定します。「パスワード」、「パスワード (確認)」の欄に同じパスワードを入力して「OK」を押します。ここで設定するLinux管理者のログイン名は「root」です。パスワードの設定が終わると、いったんマシンが再起動されます。



追加インストールディスクの有無

本誌にはコンパニオンCDなどが付属しないので、「いいえ」を選択して次へ進みます。次の場面では「OK」を押します。



Xとコンソールで使用するキーボードの設定

Linuxで日本語キーボードを使う場合、ここから続く3つの場面で、

- 「コンソールキーボード設定」で「jp106」
- 「キーボードモデル設定」で「日本語 106-key」
- 「キーボード配置の設定」で「日本語」

をそれぞれ選択します。



マウスの設定

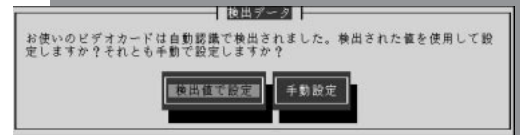
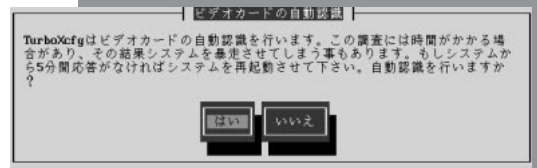
2ボタンのPS/2マウスを使う場合は、「マウス設定」で「一般的なPS/2マウス」を選択して「3ボタンのエミュレーション」をチェックします。

次の「マウスボタン」では「ボタン2つ」をチェックして「OK」を押します。マウスの設定は、ユーザーの環境に合わせて行ってください。

ビデオカードの自動認識

「はい」を押してビデオカードを自動認識させます。自動認識がうまくいかない場合は、リストの中からビデオカードを選択します。ここではビデオカードが自動認識されたとして解説します。

次の「検出データ」では「検出値で設定」を押して次へ進みます。



モニタの設定

使用するモニタのメーカーをリスト（画面手前）の中から選択します。使用するモニタがリスト（画面奥）の中に含まれていない場合は、モニタのマニュアルを参考に、周波数帯などを設定します。以後、使用するモニタがリストに含まれているとして解説します。



色数と解像度の選択

色数を設定します。Xを使う際に色数の数値が大きいくほど綺麗に表示されますが、選べる解像度が低くなる可能性があります。「16bpp」(65536色)でもカラー表示に問題はないでしょう。色数を選択したら、「OK」を押して次へ進みます。

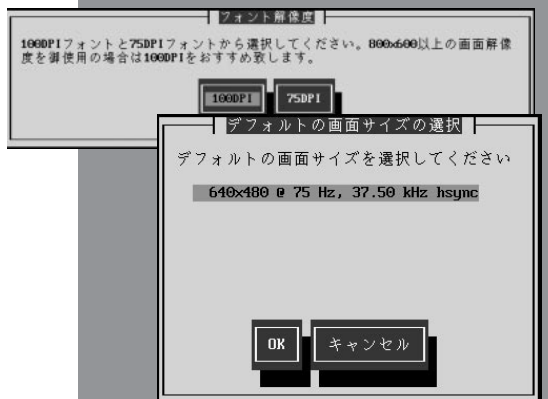
次にスペースキーで使用する解像度をチェックします。ここで複数の解像度を選択した場合、Xの使用中に、[CTRL]+[ALT]+[+]や[CTRL]+[ALT]+[-]で解像度を変更できます。

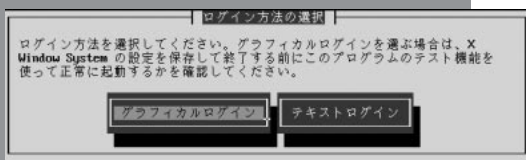
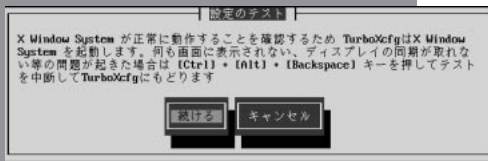
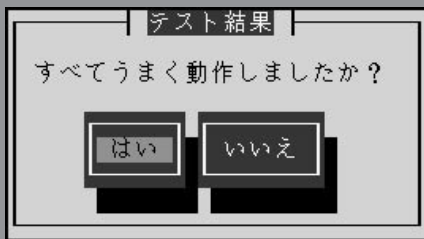


デフォルト解像度の選択

「画面サイズの設定」で複数の解像度を選択した場合は、デフォルトで使用する解像度を選択します。

次にフォントの解像度を選択します。800 × 600以上の解像度を使用する場合は「100DPI」を選択します。





Xの表示テスト

ここまで行ったXの設定をテストします。「続ける」を押して表示をテストします。Xの表示に失敗した場合は、「テスト結果」で「いいえ」を選択して、Xをはじめてから設定し直します。Xがうまく表示された場合は、「Quit」をマウスでクリックしてテストを終了し、「テスト結果」で「はい」を選択します。

ログイン方法の選択

まずログイン方法を選択します。グラフィカルな画面でLinuxにログインする場合は「グラフィカルログイン」を、テキスト画面でログインする場合は「テキストログイン」を選択します。

ウィンドウマネージャの選択

ユーザーの好みにあわせてウィンドウマネージャを選択します。どれを選択してよいかわからないユーザーは、TurboLinuxのデフォルトウィンドウマネージャである「GNOME」を選択するとよいでしょう。

インストール後でも、turbowmfcgを起動すれば、ウィンドウマネージャを変更できます。

起動サービスの設定

最後はLinux起動時に立ち上げるサービスを設定します。Linux起動時に有効にするサービスは、スペースキーを使って「*」とチェックを入れます。起動サービスを設定して「終了」を押すと、インストールは終了です。お疲れさまでした。