1台のPC上で複数のOSを稼動させる仮想化ソフトウェア VMware Workstation 3.0

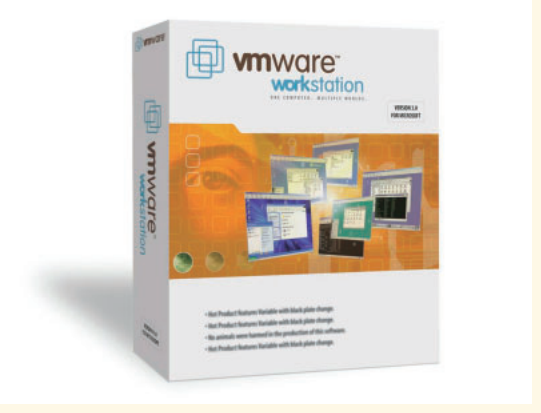
ネットワールドは仮想化ソフトウェア製品の最新バージョン「VMware Workstation 3.0」を10月22日より発売する。

VMware Workstation 3.0は、1台のPCのホストOS上で複数のOS（ゲストOS）をリブートせずに同時に稼動できるソフトウェア。ゲストOSは、VMwareプラットフォーム上に構築された、独自のネットワークアドレスを持つ仮想マシン上の独立したシステムとして扱われる。1台のPCでOSに関係なく必要なソフトウェアを利用できる。

3.0では、新たにWindows XP ProfessionalおよびHome Editionにも対応したほか、USBデバイス、DVD-ROM、CD-R／RW、CD-ROM ISOイメージ、Generic SCSIデバイスなどのサポートが加えられた。

ゲストOSをインストールするディスクは、仮想ディスクを利用することができるので、パーティションを切り直す必要はない。仮想ディスクはホストOS上のファイルを利用し、最大256Gバイトまで割り当て可能。各OSは別のゲストOSやホストOS、ほかのコンピュータなどにネットワーク接続できるため柔軟な仮想ネットワークを構築できる。Windowsスタイルの GUIを採用し操作性を向上させたほか、CPU、ネットワーク、ディスクおよびインタラクティブパフォーマンスも改善されている。

サポートされているホストOSは、Windows NT 4.0／2000／XPまたはLinux。ゲストOSは、Windows 95／98／98SE／Me／NT 4.0／2000／XP、Linuxなどに対応している。



発売日	2001年10月22日
発売	株式会社ネットワールド
TEL	—
価格	4万9000円
URL	http://www.networld.co.jp/

メディアフュージョン、XMLデータベースエンジンのLinux版を発表

2001年9月21日

メディアフュージョンは9月19日、同社のXMLデータベースエンジン「Yggdrasill（イグドラシル）」のLinux版「Yggdrasill for Linux 1.5」を12月に発売すると発表した。

「Yggdrasill」はXML文書のツリー構造を保持したまま、データベースに格納／管理／抽出することができるデータベースエンジンだ。また、同製品はDTD（Document Type Definition）の必要な検証済みXMLだけでなく、ウェルフォームドXMLにも対応しているため、拡張性、柔軟性に優れているという。今年の6月にWindows版 1.0が発売されている。

新たに発表された「Yggdrasill for Linux 1.5」では、1.0に比べて、書き込み高速化やメモリチューニング、XPathへの対応といった改良が加えられている。

動作環境は、Pentium III 600MHz以上のCPU、256Mバイト（512Mバイト推奨）以上のメモリ、ハードディスクの空き容量が1Gバイト以上。クライアントは、Pentium II 400MHz以上のCPU、128Mバイト以上のメモリ、ハードディスクの空き容量が100Mバイト以上。また、サーバ／クライアントともに、Linuxカーネル2.2以上、glibc 2.1.2以上が必要で、クライアントには、X11R6互換のXサーバが必要だ。

同製品の60日間限定のβ版配布が、9月28日より開始されている。申し込みは同社Webサイトから可能。β版ユーザーは、モニタとしてレポート提出が求められる。

メディアフュージョン
（http://www.mediafusion.co.jp/）

「KDE 2.2.1」リリース

2001年9月20日

KDE Projectは9月19日、「KDE 2.2.1」のリリースを発表した。42カ国語のサポートや、ドキュメントの改良などが行われている。

8月にリリースした「KDE 2.2」からのバージョンアップとなる今リリースは、上記の改良以外に、オブジェクトのプレリンクによるアプリケーション起動の高速化や、HTTPレンダリングエンジンの大幅な改良、多くのバグフィックスなどが行われた。

KDE Projectによると、「KDE 2.2.1」には先月リリースされた「KOffice 1.1」が統合されるという。現在のコード開発は、「KDE 3.0」に引き継がれる。「KDE 3.0」の最初のβ版リリースは、今年の12月、ファイナルリリースは来年2月末を予定している。

バイナリパッケージは、KDE.orgやミラーサイトよりダウンロードできる。

KDE Project（http://www.kde.org/）
日本KDEユーザ会
（http://www.kde.gr.jp/）

アミュレット、「セキュリティーアップデートサービス」開始

2001年9月19日

アミュレットは、同社の「Linuxサーバ導入サービス」利用者およびPCサーバ「POWERSTEP」シリーズユーザーを対象にした新サービス「セキュリティーアップデートサービス」を19日から開始した。

このサービスは、セキュリティホールになりうるプログラム一式を、最新のバージョンにアップデートし、安全性をより高めようというもの。作業はサーバの設置場所で行われる。

同社のWebサイトで紹介されている事例によると、旧バージョンシステムのバックアップをとったのち、新しいシステムをインストールし、設定、動作確認を行って再びサービスを開始するまでの作業時間は3時間程度。バックアップやインストールを省略できる作業であれば、1時間程度で終了するという。

対象となるディストリビューションは、Red Hat Linux 5.1／ 5.2／ 6.0／ 6.1／ 6.2／ 7.0／ 7.1、 Kondara MNU/Linux 2000／2.0。

価格は1件につき2万5000円。なお、同社の「サーバ年間保守サービス」を契約している場合は無料となる。

アミュレット
（http://www.amulet.co.jp/）

「Linux Kernel Conference」開催

2001年9月14日

9月14日、OSDN Japan初のイベントとなる「Linux Kernel Conference」が新宿住友ビル住友ホールで開催された。

このイベントは、今年3月に米国で行われた「Linux 2.5 Kernel Summit」を受けて、Linuxカーネルの次期開発バージョンである、カーネル2.5の方向性について探ろうというもの。当初の予定では、米VA Linux Principal Engineerであり、「Linux 2.5 Kernel Summit」主催者でもあるTed Ts'o氏が来日しプレゼンテーションを行なう予定であったが、9月11日に発生した米国でのテロ事件の影響により来日できなくなったため、プログラムの一部が変更された。

会場となった住友ホールのロビーでは、無料でインターネットに接続できるVA Linuxのマシンや、無線LAN接続環境が提供されていた。

イベントでは、現在のLinuxカーネルの特徴や、IA-64 Linuxの技術的解説、SH-4アーキテクチャや日立のメインフレーム、組み込みシステムでのLinux実装例が紹介され、最後に講師全員によるカーネル2.5の方向性や、導入される可能性の高い新技術などについてのトークセッションが行われた。参加者からは多くの質問や意見が講師に寄せられていた。

OSDN Japan（http://osdn.jp/）

テンアートニがZend TechnologiesのPHPソリューションを販売

2001年9月12日

テンアートニは9月12日、イスラエルのPHP開発企業である、Zend Technologiesと総販売代理店契約を締結し、PHPソリ

ューションの販売を開始すると発表した。

PHP（PHP:Hypertext Preprocessor）は、サーバサイドのスクリプト言語であり、JavaやPerlに比べて動作が速く、また習得が容易なため、300万以上のサイトで利用されていると言われている。日本では、日本PHPユーザ会などが中心となり普及が進められている。

イスラエルZend Technologiesは、「PHP 4.0」に搭載されているオープンソースのスクリプトエンジン「Zend Engine」の開発元で、PHP上で動作するミドルウェアや開発環境を製品として販売している。同社の主な製品ラインナップは、（1）PHP実行時のパフォーマンス改善ツール「Zend Cache」、（2）PHPコードをエンコードし非公開にする「Zend Encorder」、（3）開発環境「Zend IDE」、（4）PHP自動アップデートツール「Zend LaunchPad」など。

英語版の同社製品は、9月12日より販売が開始される。12月1日には日本語版の販売および本格的なサポートが開始され、製品はダウンロード版だけでなくパッケージ版も用意される予定。

テンアートニで行われた記者会見では、Zend Technologiesの会長兼CEOであるDoron Gerstel氏によるPHPの概要紹介と、同社の今後の販売戦略が紹介された。

同氏によると、現在実際にPHPを利用したWebサイトは300万サイトに達しており、この数は今後さらに増大するとしている。

同氏は「Zend Engineが組み込まれたPHP 4.0のユーザーは、すでにZendユーザーとしてとり込まれている」と語り、PHP開発者に対する高い知名度を武器に、企業ユーザー拡大をねらうとした。また、「日本は日本PHPユーザ会のようなユーザーコミュニティもあり、PHPに対する潜在的関心が高い市場として注目している」とし、今回の契約によりさらなる飛躍を期待すると述べた。

また、テンアートニ代表取締役社長の角田好志氏は、PHPソリューションの提供に際し、これまで日本でのPHP普及活動を行ってきた日本PHPユーザ会や、すでにPHPでの開発実績のあるテューンビズ社との積極的な協力を行うとした。

同時に、今回のZend Technologies製品の販売について「これまでJavaで培った実績を活かして、PHP製品を売り込みたい」とした。具体的な戦略として、「PHPは草の根的に、小さなソフトハウスが参画できるような」ケースで活用するとし、「官公庁や学校など、オープンソースにできる業務アプリケーション開発にPHPを利用していきたい」とした。

テンアートニ
（http://www.10art-ni.co.jp/）
Zend Technologies
（http://www.zend.com/）

米Borland、Linuxプラットフォームでの Webサービスを公開

2001年9月6日

米Borlandは、8月29日からサンフランシスコで開催されていたLinuxWorld Conference & Expo San Franciscoで、Linuxプラットフォーム向け「Webサービス」のサポートを提供すると発表した。

「Webサービス」とは、XMLやSOAP（Simple Object Access Protocol）などの技術を用い、ビジネスプロセスをインターネット上で実行できるようにしたもの。プラットフォームやハードウェアに依存しないデータのやりとりが可能になる。米Borlandの「Delphi 6」では、Webサービスのサーバサイド、クライアントサイドのアプリケーション開発機能がすでに実装されており、Microsoft.NETやSun ONEといったフレームワークにも対応しているという。

米Borlandでは、「Kylix」に現在実装されている、Apacheサーバを利用したWebアプリケーション開発環境を拡張して、Webサービスが提供できるようにするとともに、Webサービスのクライアントとなるアプリケーションの開発機能も実装するとしている。この機能が実装された「Kylix」が登場するのは、次のバージョンかそれ以降になるもよう。

Borland（http://www.borland.com/）

米テンシリカとモンタビスタ、組み込みLinuxで提携

2001年9月4日

テンシリカとモンタビスタソフトウエアジャパンの9月4日付けの発表によると、米テンシリカ（Tensilica）社と米モンタビスタソフトウエア社は同日、技術面とマーケティング面で広範にわたる提携を結んだことを発表した。

これにより、テンシリカのコンフィギュラブルマイクロプロセッサコア「Xtensa」に、モンタビスタの組み込み専用Linux「Hard Hat Linux」を移植する。

この組み合わせにより、SOC（システムオンチップ）開発者は、最初からアプリケーションに最適化されたプロセッサ上で、従来のLinuxアプリケーションソフトウェアの資産を利用できるようになる。Xtensa対応Hard Hat Linuxは、2002年第2四半期にリリースされる予定。「Hard Hat Linux Professional Edition」の中に含まれ、サブスクリプション契約の下に提供される。

テンシリカは、量産対応の組込みシステム用特定用途向けマイクロプロセッサコアと、ソフトウェア開発ツールの提供会社。独自に開発したXtensaプロセッサジェネレータにより、SOC設計者は数時間で特定のアプリケーションに適合したプロセッササブシステムのハードウェア設計と、ソフトウェア開発ツール環境を作成できるという。

米テンシリカ
（http://www.tensilica.com/）
米モンタビスタソフトウエア
（http://www.mvista.com/）

Hardware　発売日　2001年8月20日

Itanium搭載サーバ PowerEdge 7150

URL　http://www.dell.com/jp/

発売	デルコンピュータ株式会社
TEL	044-556-6190
価格	263万3000円〜

デルコンピュータは、同社「PowerEdge」シリーズで、インテルの64ビットプロセッサItaniumを搭載したハイエンドサーバ「PowerEdge 7150」を8月20日より発売した。

PowerEdge 7150はデルコンピュータのサーバとしては初めてItaniumを搭載した製品でソフトウェア開発、科学技術演算、大規模データベース向けに開発された。

主な仕様は、CPU Itanium 733MHz／800MHz（最大4個搭載可能）、メモリ 1Gバイト（最大64Gバイト）、Ultra3 SCSI HDD 18Gバイト（最大4基搭載可能）。ホットプラグ対応PCIスロット10基（64ビット／66MHz×8、64ビット／33MHz×2）、24倍速CD-ROM、100Base-TX、7Uラックマウントサイズ。電源ユニットと冷却ファンは標準で冗長化されている。対応OSは、Red Hat Linux 7.1 for Itanium。

Hardware　発売日　2001年9月末

Debian GNU／LinuxをサポートするLinuxサーバ VA Linux 1222／VA Linux 2252

URL　http://www.valinux.co.jp/

発売	VA Linux Systemsジャパン株式会社
TEL	—
価格	52万円〜

VA Linux Systemsは、1UラックマウントサイズのLinuxサーバ「VA Linux 1222」と2UラックマウントサイズのLinuxサーバ「VA Linux 2252」を9月末より発売した。

製品の主な仕様は、1222がCPU Pentium III 1.0Gバイト×1（最大2個）、メモリ256Gバイト（最大4Gバイト）、PCIスロット2基（64ビット×2）、Ultra160 SCSI HDD 18Gバイト（最大36Gバイト×2）、100Base-TX×2。2252が、CPU Pentium III 1.0Gバイト×1（最大2個）、メモリ512Mバイト（最大4Gバイト）、PCIスロット4基（32ビット×1、64ビット×3）、Ultra160 SCSI HDD 18Gバイト×2（最大73Gバイト×5）、100Base-TX×1。いずれのモデルもオプションで2チャンネルRAIDコントローラを付けられる。

OSはカーネル2.4をベースにしたVA Linux OS Version 7.1とDebian／GNU Linux、Red Hat Linuxの中から選択できる。

Hardware　発売日　2001年8月23日

最大4Tバイトのバックアップが可能なストレージ製品 PowerVault 128T

URL　http://www.dell.com/jp/

発売	デルコンピュータ株式会社
TEL	044-556-6190
価格	116万円〜

デルコンピュータは、次世代テープテクノロジーLTO（Lenear Tape Open）対応のUltriumドライブを搭載したテープオートローダ「PowerVault 128T」を8月23日より発売した。

PowerVault 128Tは、5Uラックマウントサイズで、10巻のテープを装填可能なドライブを2基まで搭載できる。ドライブ2基を搭載してデータを圧縮することで最大4Tバイトのデータバックアップ、60Mバイト／秒の転送速度を可能にした。オプションのファイバーチャネル用モジュールを使用して、SANなどの大規模システムにも対応する。

ドライブ1基、Ultra3／LVD SCSIインターフェイスの最小構成で116万円。

当日4時間以内にオンサイト保守を提供する「当日4時間プラス対応オンサイト保守サービス」を標準で提供している。

Hardware　発売日　2001年9月1日

インターネット端末機「iBOXシリーズ」の新製品「iBOX-2」

URL　http://www.jcc.co.jp/ibox2/

発売	日本電算株式会社
TEL	03-3864-5511
価格	オープンプライス

日本電算機は、インターネット端末機「iBOX」シリーズで、ブロードバンドにも対応した新製品「iBOX-2」を9月1日より発売した。

iBOX-2は、インターネット接続用のセットトップボックスで、OSにLinux、CPUにIntel互換チップを採用した。筐体はB5判以下で約600gのコンパクトサイズ。アナログ回線用モデルのiBOX-2 DLとブロードバンドに対応したiBOX-2 BBの2機種があり、ブロードバンド対応モデルでは10Base-Tをサポートしている。

Linux＋Intelアーキテクチャの採用はiBOX-2からで、これによってiBOX-1に比べて描画性能を向上させた。HTML4.01準拠、JavaScript1.3をサポートしたブラウザとメール機能を搭載。Flash、RealPlayer、PDFファイルなどほとんどのプラグインをサポートしている。ほかにも音声メール機能、リモコンが付属している。テレビで手軽にインターネットに接続できる。

Hardware

発売日 2001年9月14日

組み込みLinux開発用パッケージ Lineo uCdimm

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

発売 アミュレット株式会社
TEL 03-5295-8418
価格 14万8000円

アミュレットは、組み込みLinux開発用パッケージ「Lineo uCdimm」を9月14日より発売した。

Lineo uCdimmは、モトローラのCPU DragonBallを搭載した144pinのDIMMモジュールを、開発用ボード（uCevolution）に組み込んだ6×8インチのLinuxマシン。組み込み機器のプロトタイピング、電子回路の実験など開発用途の製品。

uCdimmモジュールには、CPU Motorola 68K 33MHz DragonBall VZ、メモリ8Mバイト、フラッシュメモリ2Mバイト、外部インターフェイス（TCP/IP、SLIP、PPP、RS-232C×2、SPI×2）、640×512のLCDドライバがあり、ファームウェアとして組み込み機器向けOSのLineo uClinuxがプリインストールされている。これを10Base-T×1、ETH／DSL切替え可能イーサネットコントローラ×1、RS-232C×2、LCDコネクタ×1などを備えた開発用ボード（uCevolution）に組み込んで、イーサネットとシリアルポートを通して利用する。

Hardware

発売日 2001年10月12日

1UサイズのWebアプライアンスサーバ eServer xSeries 135

URL http://www.ibm.com/jp/servers/

発売 日本アイ・ビー・エム株式会社
TEL 0120-04-1992
価格 22万8000円〜

日本IBMは、1Uラックマウントサイズの筐体にWeb関連のソフトウェアをプリロードしたWebアプライアンスサーバ「eServer xSeries 135」を10月12日より発売する。

xSeries 135には、コスト・パフォーマンス重視のValueモデル（8672-24X）とパフォーマンスと管理能力に優れたPerformanceモデル（8654-5CX）の2モデルがある。

主な仕様は、8672-24XがCPU Celeron 800MHz、IDE HDD 20.4Gバイト、8654-5CXがCPU Pentium III 1GHz、Ultra160 SCSI HDD 72.8Gバイトで、インターフェイスにシステム管理プロセッサーポート2個が付いている。両モデルともメモリは256Mバイトでネットワークインターフェイスは100Base-TXに対応、USB×2、シリアルポート×1を備えている。

OSにRed Hat Linux 6.2、WebサーバにIBM HTTP Server（Apache）を採用し、CGIとSSL、メーリングリスト機能（MajorDomo）をサポートする。

Hardware

発売日 2001年9月17日

Pentium 4-2GHz搭載可能なPCサーバ Standardシリーズ

URL http://www.plathome.co.jp/factory/

発売 ぷらっとホーム株式会社
TEL 03-3251-2600
価格 9万8000円〜

ぷらっとホーム株式会社は、個人・SOHO・中小企業向けPCサーバ「Standardシリーズ」の新製品4機種を9月17日より発売した。

今回発表されたモデルは、幅広いOSとの互換性を持たせるためにIntel 440BXチップを使用したStandard／BX、ServerWorksのチップセットを用いCPU Pentium IIIを2個搭載できるStandard／LE3、Intel 815チップセットを利用してPC133ベースの高速システムを構築できるStandard／815、Intel 850チップセットとRIMMをベースとしたメモリシステム（ECCもサポート）を採用し、Pentium 4を搭載したStandard／850の4種類。HDDも、経済性や拡張性など要求される状況に応じてIDE HDDとSCSI HDDのいずれかを選択できる。

筐体はタワー型のほかに、4Uラックマウント型への対応も可能。

Hardware

発売日 2001年9月上旬

全文検索ソフトをバンドルした低価格Alphaサーバ Kondara／MitakeSearch Serverモデル

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

発売 コンパックコンピュータ株式会社
TEL 0120-018589
価格 85万円

コンパックコンピュータは「Compaq Linuxソリューションパッケージシリーズ」の第3弾としてAlphaServer DS10とKondara MNU／Linux 2.0 Compaq Edition、MitakeSearch V3.0を組み合わせた「Kondara／MitakeSearch Serverモデル」を9月上旬より発売する。

Kondara／MitakeSearch Serverはネットワーク上のWeb検索、社内文書の共有と有効利用といったナレッジ・マネジメントの核となる全文検索／文書管理システムを提供する。導入後すぐ利用できるターンキーソリューションとなっている。

サーバにはCPU Alpha 21164 64ビット／600MHzプロセッサ、メモリ512Mバイト、HDD 36.4GバイトのAlphaServer DS10を使用。1年間回数無制限のLinux管理者向け電話サポート、ハードウェアの1年間オンサイトサポートが含まれる。

Hardware

6万円台のエントリーサーバ PowerEdge 500SC

発売日 2001年9月27日

発売 デルコンピュータ株式会社
TEL 044-556-6190
価格 6万9800円〜

URL http://www.dell.com/jp/

デルは、PowerEdgeシリーズのエントリーサーバ「PowerEdge 500SC」を9月27日より発売した。

PowerEdge 500SCの主な仕様は、CPU Celeron 900MHz（またはPentium III 1GHz）、メモリ64Mバイト（最大2Gバイト）、10GバイトのUltraATA/100 HDD（最大60Gバイト IDE HDD×3基搭載可能）、CD-ROM、5基のPCIスロット（64ビット／66MHz×2、32ビット／33MHz×3）を実装している。

サーバの管理は独自のサーバ管理ツール「Dell OpenManage IT Assistant」、「Dell Server Assistant」により、Webベースのブラウザを使用して行われ、メンテナンスや、ハードウェアの故障状況のモニタリングができる。

OSは現在のところWindows 2000／NT Serverに対応しているが、今後Red Hat Linuxにも対応する予定。製品には24時間電話テクニカルサポート、1年間翌営業日対応オンサイト保守サービス、3年間パーツ保証が提供される。

Hardware

ストリーミング配信専用サーバパッケージ リオサーバ・アプライアンス

発売日 2001年8月22日

発売 ぷらっとホーム株式会社
TEL 03-3251-6151
価格 54万円〜

URL http://www.plathome.co.jp/appliance/

ぷらっとホームは、トランス・コスモスと共同開発したLinuxベースのストリーミング配信専用サーバパッケージ「リオサーバ・アプライアンス」を8月22日より発売した。

リオサーバ・アプライアンスは、リアルネットワークスのストリーミングソフト「RealSystem Server 8.0」に最適化したRed Hat Linux 6.2Jとサーバ、RealSystem Server 8.0からなるパッケージ製品。

主な仕様は、エントリーモデルがCPU Pentium III 800MHz、メモリ256Mバイト、IDE HDD 40Gバイト、100Base-TX。プロフェッショナルモデルがCPU Pentium III 933MHz（最大1GHz×2）、メモリ512Mバイト、Ultra160 SCSI HDD 36.4Gバイト（最大4基搭載可能）、100Base-TX×2。両モデルにはRealSystem Server Plus for Academicライセンスとサポートがつく。

最大構成で、1台のサーバで同時配信できるストリームデータは、1Mbpsのデータで400本、450Kbpsのデータで600本、64Kbpsのデータで2000本という評価結果が得られている。

Software

Linux用コンパイラ Intel C++／Fortran Compiler for Linux バージョン5

発売日 2001年9月12日

発売 エクセルソフト株式会社
TEL 03-5440-7875
価格 6万8000円（C++）／7万5000円（Fortran）

URL http://www.xlsoft.com/jp/products/vtune/perftool.htm

インテルは、Linux用コンパイラの新製品「Intel C++ Compiler for Linux」、「Intel Fortran Compiler for Linux」のバージョン5を発表した。国内では、エクセルソフトから販売される。

Itaniumプロセッサ、Pentium 4のSSE2（Stream SIMD Extension 2）に対応したコードを生成できるほか、既存のx86プロセッサ上でもGCC（GNU Compiler Collection）と比較して高速なコードを生成可能だ。また、各プロセッサ用に最適化されたコードを複数生成し、実行する環境に合わせたコードを自動選択することもできる。

Linuxで一般的なGCCと互換性があり、出力されたコードは、GCCで作成されたオブジェクトコードとリンクできる。なお現バージョンでは、インラインアセンブラ機能をサポートしていないため、カーネルのコンパイルには対応していない。

Software

Itanium対応Linuxサーバマネジメントシステム HDE Linux Controller Itanium版

発売日 2001年10月末

発売 株式会社ホライズン・デジタル・エンタープライズ
TEL 03-5738-5410
価格 オープンプライス

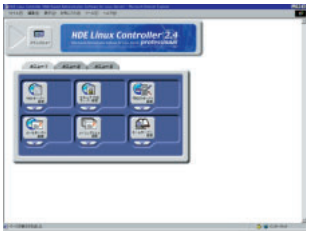
URL http://www.hde.co.jp/

ホライズン・デジタル・エンタープライズは、64ビットItaniumアーキテクチャ対応Linuxサーバマネジメントシステム「HDE Linux Controller Itanium版」を10月末より発売する。

HDE Linux Controller Itanium版はHDE Linux Controller 2.4 Professional Editionと同等の機能を有する64ビットItanium対応製品。

主な機能は、セキュリティ設定、ソフトウェア管理、Webサーバ設定、サーバステータス設定、メールサーバ設定、ユーザー／グループ管理、ファイルサーバ設定、メーリングリスト設定、ネームサーバ設定、バックアップ管理、ネットワーク設定など。

対応するOSはRed Hat Linux 7.1 for Itanium、Turbolinux Operating System 7。



Software　　発売日　2001年8月29日

Webメールサーバソフトウェア GraceMail Ver.3

URL　http://www.hitachi-ms.co.jp/gracemail/

発売	株式会社日立マイクロソフトウェアシステムズ
TEL	045-865-8160
価格	21万8000円

日立マイクロソフトウェアシステムズは、Webメールサーバソフトウェアの最新バージョン「GraceMail Ver.3」を8月29日より発売した。

GraceMailはクライアントPCのWebブラウザを利用して、メールの送受信をするメールサーバソフトウェア。Ver.3では従来の機能に加えて、新たに①メール自動振り分け、②複数署名の利用、③送信者のアドレス帳登録、④開封確認、⑤ユーザーによるカスタマイズ、⑥他のメーラとのアドレス帳共有（オプション）などの機能を追加した。

動作確認されているLinux OSは、Red Hat Linux 6.2J、Turbolinux Server 6.1／6.5、Vine Linux 2.0に対応。動作の確認されているSMTPサーバは、sendmail、qmail 1.03、POP3サーバは、imap、qpopper 4.0.3。マルチCPU上で動作させる場合には別途見積りが必要。

Software　　発売日　2001年12月初旬

WebSphere対応BtoC向けECサイト構築ソフトウェア Dynamic Personal WebShop 1.2 for WebSphere（DB2 version）

URL　http://www.azi.co.jp/

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

クラスキャットネットワークは、「Dynamic Personal WebShop」シリーズのBtoC ECサイト構築ソフトウェアの新製品「Dynamic Personal WebShop 1.2 for WebSphere（DB2 version）」を12月初旬より発売する。

Dynamic Personal WebShop 1.2 for WebSphere（DB2 version）は、JavaServer PagesおよびJavaBeansテクノロジーをベースとしたECサイト構築ソフトウェア。オンラインショップとしてのデザイン部分と運用・管理などを行うプログラム部分を分離したことにより、ECサイトのデザイン変更が簡単にできる。オンラインショップの運用・管理には専門的な知識を必要としない。顧客からの受注はデータベースに登録・管理しており、検索結果はExcelなどに出力できる。顧客アドレス帳の自動作成機能、メールによる受注通知機能などがある。

Software　　発売日　2001年9月18日

Excel／WordファイルからXML形式にデータ変換 X-大将（ばってんたいしょう）

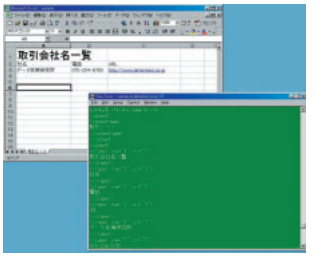
URL　http://www.dehenken.co.jp/

発売	株式会社データ変換研究所
TEL	075-254-8780
価格	40万円

データ変換研究所は、Microsoft Excel／Microsoft WordからXML形式で情報を取り出せるソフトウェア「X-大将（ばってんたいしょう）」を9月18日より発売した。

X-大将は、Excelのセルの位置情報、数値、文字情報、リンク情報、シート情報などを直接ExcelファイルからXML形式の情報に変換できる。Wordでは、「標準」の文書情報、「見出し1」「見出し2」「見出し3」の情報に分けて、WordファイルからXML形式の文書情報を生成できる。

本ソフトウェアは、Excel／Wordファイルをデータベースのフロントエンドに利用した、Webサーバ、データベースサーバでの利用を想定している。オペレータは使い慣れたExcel／Wordファイルで情報を作成・管理し、X-大将を用いてXML形式で出力してデータベースやWebコンテンツとして利用する。Linux版、Solaris版、FreeBSD版、Windows版が用意されている。



Borland Conference 2001 — Tokyo — Webアプリケーション開発の最新状況を知る　http://www.borland.co.jp/borcon2001/

「Borland Conference 2001」は、7月21日より、ロングビーチを皮切りに世界各国で開催されているソフトウェア開発者のためのコンファレンスツアーで、今回この会議が東京で開催される。

日本でのテーマは、Webアプリケーション開発技術。基調講演で、ボーランドのWebシステム開発について今後の戦略を紹介したのち、テクニカルセッションに入る。セッションは、大きくトラック（分野）に分けられ、それぞれのトラックで個別のテーマが扱われる。トラックはWeb関連技術の概要を扱う「Getting Start」、Webアプリケーション開発技術を扱う「Webアプリケーション」、システム設計を扱う「デザイン」、プラットフォームとしてのLinux利用法を扱う「Linux」の4つ。KylixやJBuilder関連のセッションも多く含まれている。展示コーナーでは、ボーランドのパートナー企業による、展示や実演も行われる。最新技術の情報交換、収集にはうってつけだろう。

場所と日時は下記のとおり。セッションの内容と申し込みの詳細はWebサイトを参照のこと。10月13日までならば、早期申し込み価格が適用される。

日時：2001年11月13日（火）～14日（水）
場所：セルリアンタワー東急 B2F、B1F
主催：ボーランド株式会社
参加費：6万5000円（税別）
　　早期申し込み価格 5万円（税別）

日刊アスキー Linux Business Report

あえて聞く「Linux景気のいい話」

http://www.linux24.com/

Gatewayの日本撤退やHPとCompaqの合併など、昨今のIT業界は生き残りを図るための大激震が続いている。日刊アスキーLinuxの読者アンケートでも、「今後、リストラやビジネスモデルの変更など、方向転換が迫られるLinux関連企業が続出すると思いますか？」という問いに対し、「思う」と回答した読者が87％に達したほか、「Linux関連企業が生き残る道は何か？」という自由回答設問に対しても、「アイデアがあれば私が教えて欲しい」といった意見が出るなど、世間一般の不況感が強まっている現実が露わになった。また、マスコミの流すニュースも、おしなべて不況関連一色といった様相だ。

そこで今回はあえて、Linux関連業界に聞いた最近の成功事例や今後の展望をお届けしたい。

「Linuxの業績は前年比2.3倍」

まず注目したいのが、Linuxの上半期業績が前年比2.3倍、という数字が上がってきているテンアートニである。同社代表取締役社長・角田好志氏によると、今後のポイントは2つあるという。1つは、IT投資の減額に悩まされている大企業情報システム部門でのSolarisからLinuxへのリプレースが進み、カーネル2.4やItaniumの出現で、ユーザーのニーズも急速に高まっている。もう1つはPHPを活用した業務アプリケーションの推進である。先月紹介した、ミロクドットコムのclear worksなど、PHP活用のソリューションは水面下での浸透という段階を終え、徐々に一般への認知度が上がってきている。テンアートニはPHP企業であるイスラエルZend社との包括的な提携をすでに行い、こうした状況に先手を打った格好となっている。

なお、スペース上の都合もあり、ここでは「Linuxは、いよいよ儲かるステージへ」と題された角田社長のコメント全文を紹介できない。同氏のコメントには、ほかにもLinux業界の展望などもあるため、本誌発売日に日刊アスキーLinux上にて全文を公開したい。

次に、具体的な数値は公表されないものの、「売り上げは確実に増えている」というのがアミュレットだ。ハードウェア、教育事業、ディストリビューション、SIなどを手がける同社だが、特にSI分野において、今まで試験的にLinuxを導入していた顧客企業から、より具体的なシステム要望を受けるようになってきたという。また、同社は約3年前からLinuxセミナーを実施しているが、その修了生からSI業務を発注されるなど、教育分野とSI分野が連動するケースもあるという。ハードウェア面でも、2001年2月に出荷を開始した1Uのラックマウントサーバ（8月はSMPの1Uも販売）が好調。同社にとっては、「ようやく実りの時期が来た」といったところだという。今後はサービス分野に注力していくとのことだ。

Linux教育の分野は動きが激しい。今年8月、ドットネットが社名変更を行い、レーザーファイブドットネットとなり、代表取締役社長にレーザーファイブの窪田敏之氏が就任している。もともとレーザーファイブとドットネットは資本関係があったが、ここにきていっそうのシナジー効果を得ようという考えだ。これによりレーザーファイブは、LASER5 Linuxの中核技術をベースに、リナックスパッケージソフトの開発・販売、教材の開発、講師・エンジニア派遣、サポートまでを総合的に提供できる体制になった。レーザーファイブドットネットの友行信氏によると、体制の移行後は資料請求が倍増し、売り上げも大幅に伸びる勢いだという。

ここまで挙げてきたのは、どちらかというと法人向けのビジネスだが、コンシューマー向けで最近目立ったのはターボリナックス ジャパンの新製品Tubolinux 7 Workstationがパソコン用ソフトウェア売り上げで1位を獲得したことである。これは、ラオックス・ザ・コンピュータ館、ぷらっとホーム、T・ZONE本店の3店舗において、発売日の9月7日から9日の3日間で、パソコン用ソフトウェア分野での売り上げ順位が1位だったというもの。Turbolinux 7の発売日に秋葉原で無料セミナーを中心としたキャンペーンの実施が功を奏したといえそうだ。小島國照氏の社長退任（現在は同社特別顧問）をきっかけに「米国でのLinux製品の販売から事実上撤退」といった誤報も飛び交った同社だが、大量のLinuxサーバを効率的に管理・運用するためのソリューションも市場に投入

予定であり、今後の動向が注目される。

法人、コンシューマー問わずまんべんなく利益がもたらされている、というのは、最大のLinuxディストリビューターであるレッドハットだ。同社では、ディストリビューションRed Hat Linux7.1、組み込み向けOSのEmbedded Linux、eCos、コンパイラ、デバッガ等のツール・GNUProの移植やボードに応じたカスタムエンジニアリングサービス、Linux管理者や開発者を養成するラーニングサービスRHCE、RHDなどを行っている。今後のレッドハットは、北アジア地域（日本、韓国、中国、香港、台湾）のヘッドクォーターとして、ディストリビューション、教育、エンジニアリング、コンサルティングなどを展開していく。

アジアという切り口では、Linux用オフィススイートを開発するハンコムリナックスが思い当たる。同社では最近、法人向けや学校向けの需要などがあるという。San Franciscoで行われたLinuxWorldでも、Hancom Officeへの関心が高かったそうで、「驚いている」とのことだ。

次に、最近はAS400における日本アイ・ビー・エムとの提携も発表したぷらっとホームだが、AS400との連携アプリケーションを使ったソリューション事例が増えてきているほか、最近では汎用機やオフコンを使うユーザーが、Web連携のビジネスを行う例も目立つという。この「Web連携」では、B2Bソリューションが中心となっている。AS400に関しては「クロスプラットフォームスクェア」というショールームも開設した。

サーバメーカーのノーザンライツは最近、アプライアンスサーバであるSOOXiに注力しているようだ。社内システム全体を、Windows NTからSOOXiで構成し直す事例などもあり、その低価格が受けている。SOOXiはLinuxカーネルを使った独自パッケージを搭載しているのだが、オープンソースとして公開を準備しているという。今後は、超並列クラスタ向けの製品を出荷予定だ。

本国ではハードウェアからの撤退が決定されたVA Linux Systemsジャパンだが、日本国内ではサーバ新製品VA Linux1222（**写真**）、VA Linux 2252を出荷するなど、ハードウェア事業が続けられる。今後は、ソフトウェア共同開発支援システムSourceForge（ソースフォージ）、Linuxシステムの販売、プロフェッショナルサービス、OSDNコミュニティサイトとイベントの運営などを中心として展開していく。

最後に、「太っ腹にやっていく」というのが、組み込み用Linux「式神」を開発するアックスだ。手書き文字入力システムを使いたいというオファーも来ているとのこと。「式神1.0はオープンソースで無償提供だが、GNOMEやKDEの開発者からも面白いと言ってもらえる。こんなに景気のいい話はないでしょう」というのが同社のコメントなのである。

Linuxの強みはやはり低コスト？

「フリーソフトの『フリー』は『タダ』という意味ではない」という話は、Linuxが広まっていく段階で何回となく語られてきた。だが現在、Linuxをはじめとしたオープンソースソフトウェアのビジネス上の最大の武器は、やはり無償であることなのではないか。

以前主張されていたオープンソースの利点は、ソースが公開されていることによって垣根のない技術共有が可能で、トラブルへの対処や開発スピードが速い、といったことだったと思う。現在も、こうした利点を土台として開発やビジネスが行われていることは変わりないが、セールストークとしては、「無償ソフトウェアを使っているので低コスト」といった面がもっと前面に出てくるのではないかと思われる。実際、clear worksにしても、オープンソース的な開発手法を用いながら、セールスはあくまでも「低コスト」である。また、ほかのLinux企業のセールストークを見ても、あからさまに「Linuxのシステムは安いです」とまでは言わないまでも、コスト面はやはり強調されている。その背景にあるのは、やはり「無償」の2文字だろう。

今まで「フリー」に対する誤解を恐れ、世間の「タダのソフトになんで値段を付けて売るのだ」という理解不足もあり、「元々はタダのソフトをベースにしていますから低コストです」とは言いにくい面があったはずだ。しかし、「フリー」に対する理解の浸透や、不況によって価格面への注目度が以前より高くなったことを追い風に、Linuxの優れた低コスト性をようやく前面に押し出せる状況となってきたのではないだろうか。「理念ではなく価格による直球勝負が可能だ」というと身も蓋もないが、そもそも「いいものをより安く」がビジネスの根本とするならば、むしろLinuxのコストパフォーマンスはもう少し強調されてもよいのかもしれない。

Linuxやオープンソースに立脚する企業だからこそ、逆に「Linuxは安い」と言える資格があるのではないだろうか。

VA Linux Systemsジャパンの1Uサーバ新製品VA Linux1222

LinuxWorld Conference & Expo

August 26-30, San Francisco

……オープンソースの危機？

文：竹田善太郎
Text : Zentaro Takeda

8月末にLinuxWorld Conference & Expoがサンフランシスコで開催された。今回は展示内容もさることながら、プレゼンテーションで暗号解除を規制するデジタルミレニアム法への危惧が語られたのが目を引いた。2週間後に起きた同時多発テロで規制強化が叫ばれるいま、市民的自由はオープンソース陣営にとっても焦眉の課題である。

2001年8月28日から30日の3日間（テクニカルカンファレンスは26日から開催）、米国カリフォルニア州サンフランシスコのモスコーニコンベンションセンターの北ホールを主会場に、IDG主催の「LinuxWorld Conference & Expo」が開催された。昨年の夏までは、同イベントはシリコンバレーの中心地であるサンノゼで開催されていたが、今年は会場をサンフランシスコに移している。

主催者の発表によると、出展企業の総数は180社、会期中の来場者数は1万8000人で、昨年8月の同イベントの200社2万人という数字に比べて、会場の利便性が増したにも関わらず、かなり控えめな数字になっている。昨今の「IT不況」の世情を反映して、Linux業界でも倒産や合併する企業が少なくなく、会場内には空いたままのブースも散見され、どうしても寂しい感じをぬぐい去ることはできなかった。

それでも、3日間の会期中、人気企業のブースやキーノートスピーチの会場には多くの参加者が詰めかけ、西海岸特有の肌寒い初秋の陽気をものともしない、独特の熱気を感じることができた。

ハードウェア関連は静か

Linux業界全体が、ハードウェアやソフトウェアの販売により利益をあげることから、インテグレーションなどのサービス分野に重点を移していることを反映して、ハードウェア関連製品の展示は、拍子抜けするほど少なかった。ラックマウント型サーバ、ミニサーバなどのサーバ製品については、各社が展示を行っていたが、Itanium搭載の大型サーバ、Athlonのマルチプロセッササーバなどの高性能サーバが中心で、デスクトップLinuxマシンについては、ソフトウェアのデモ用に用意されているものを除き、ほぼ姿を消している。IBM、HP、Compaqなどの大手PCメーカーのブースでも、ハードウェア関連の展示は意外と少なく、各社が力を入れているソフトウェアソリューションや開発者向けサポートなどの展示が中心となっていた。

カラー液晶ディスプレイを中心にして、ポータブルデバイスにLinuxを組み込んだ製品のコンセプトモデルを展示している企業も見受けられたが、これらが実際に製品として発売されるかどうかは不明だ。

IBMのブース。自社のハードウェアに関する展示はほとんどなく、中小のソリューションプロバイダの展示スペースや、同社のLinuxサポートに関する展示が中心だった。

AMDのブースに展示されていたAppro社のデュアルAthlon 1Uサーバ「Appro 1124」。1Uユニットの中に、Athlon 2個のデュアルCPUのマザーボードとHDD 4台からなるRAIDディスク装置をすべて納めた高密度、高性能サーバ。

開発環境とリモート管理ツール

ソフトウェア関連では、開発者向けのツールやサポートシステム、とくに、開発者向けにプログラミング関連情報を提供するサービスに力点を置く企業が多かった。オープンソースのコミュニティでは、「有料」のツールや情報を使うことに抵抗のある開発者が多いようだが、ボーランドのKyrixを始め、オープンソースのソフトウェアを作成する場合に限っては、無償で利用できるとす

る開発ツールやサービスは多い。IBMやHPなどのハードウェアメーカーでも、自社製品に関連する情報のみならず、Linux一般の開発者向け情報サービスに力を入れており、各社のブースで情報提供サービスのデモを行っていた。

このほか、ソフトウェア関連では、Webを中心としたインターネットアプリケーションの展示が多かったが、比較的目新しいものとして、複数のLinuxマシンをリモートで管理するソリューションが目についた。現在のLinuxディストリビューションは、RPMなどの「パッケージ」によって、ソフトウェアのインストールや管理を行うようになっている。この仕組みをリモート管理に応用して、管理用のコンソールからリモートのマシンにパッケージをインストールしたり、設定を行ったりできるようにする製品がいくつか展示されていた。

中には、ディストリビューターからリリースされるパッケージの動作を独自に検証し、それに合格したパッケージのみインストールできるようにコントロールするツールなどもあり、企業内のクライアントとしてLinuxを利用する場合など、安定性が重視される環境に対応している。

デスクトップアプリに新しい動き

一方、Corelの撤退などで、Linuxのデスクトップアプリケーションは影が薄くなってきたものの、HancomがOfficeアプリケーションの新バージョン「Hancom Office 2.0」を展示するなど、新しい動きもあった。また、GNOMEプロジェクトを母体とするソフトウェアベンダーであるXimian（旧称Helix Code）は、スケジュール管理、アドレス帳、メールクライアント、Palmとのシンクロナイズ機能などを備えた「Ximian Evolution」を展示していた。説明員によれば、日本語対応の時期については不明とのことだが、今後の展開が楽しみな製品のひとつである。また、Sunのブースでは、オフィススイートのStar Office（日本国内ではSun Office）や、オープン版の「Open Office」についての展示も行われていた。

オープンソースコミュニティ最大の敵

モスコーニセンターの北隣に位置するマリオットホテルのホールでは、キーノートプレゼンテーションが開かれ、Compaq副社長のShane Robinson氏、Red Hat社長のMathew Szulik氏が、今後の製品戦略などについてスピーチを行った。

これらのプレゼンテーションの中で異色だったのは、スタンフォード法科学校のLawrence Lessig教授による「Old vs. New: The New New Old War」というタイトルのセッションだった。同教授は、「オープンソースコミュニティ最大の敵は、私の弟子でもある、企業付きの弁護士たちだ」と断言し、「デジタルミレニアム法によって、プログラマーの『言論の自由』が侵害されているということに、もっと危機感をもってほしい」と訴えた。

今年の7月17日には、デジタルミレニアム法の適用を受けた最初の逮捕者が出ている。ロシア人のプログラマーDimitri Sklyarov氏がその人だ。おりしも、LinuxWorld Expo最終日の8月30日、サンフランシスコ市内の連邦裁判所で罪状認否の法廷が開かれ、Sklyarov氏は無罪の主張をした。この日には、Linuxコミュニティの有志による、デモンストレーションがLinuxWorld Expo会場と連邦裁判所の間で行われ、十数人の参加者が「Free Dimitri Now!」（ディミトリを即刻釈放せよ！）というシュプレヒコールをあげながら、通行人にチラシを配布していた。

Sklyarov氏の事件では、Adobeの電子書籍の暗号を解読するプログラムが、Adobe社の知的所有権を侵害したとされているが、ロシア国内の法律ではこのような行為は禁止されていない。また、他人の「アイデア」をコーディングによって具体化させるという行為自体が違法になると、プログラマーの自由が大幅に制限されてしまい、今後のオープンソース活動の大きな支障になると危惧されている。

Linux関連産業の今後だけでなく、ソフトウェア産業と全体と、そこで活動するプログラマーのありかたそのものについても、考えさせられることの多いイベントだった。

キーノートプレゼンテーション「Old vs. New: The New New Old War」のスピーカーLawrence Lessig教授。「私の弟子たちは、あなたがたよりもずっと優秀だ。もはや、勝負はついてしまっている」と、かなり挑発的な内容のスピーチだった。

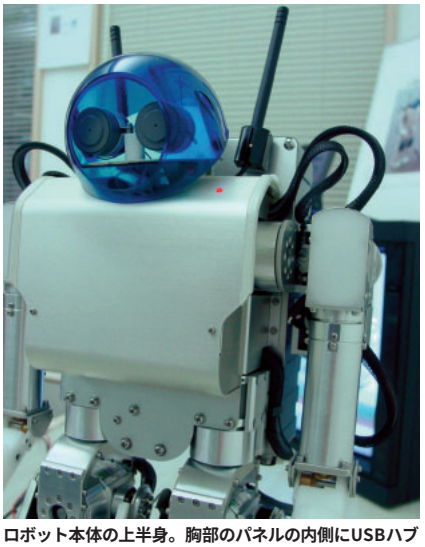
デジタルミレニアム法に反対し、Sklyarov氏の無罪放免を要求して、サンフランシスコ市内をデモ行進する人々。交通整理の警察官やパトカーがついた、本格的なデモ行進だった。

RT-Linuxで制御

小型ヒューマノイドロボット HOAP-1

ロボットの研究といえば、特化された高額の機材と企業秘密のベールに覆われた技術を利用できる、ごく限られた研究機関のものと思われてきた。HOAP-1は、個人が簡単に買える値段ではないものの、標準的な技術を採用し、オープンアーキテクチャとなっている。そしてそれは、同様のコンセプトのコンピュータが計算機科学を飛躍的に進化させたことを思い出させてくれるのである。

富士通研究所は小型ヒューマノイドロボット「HOAP-1」を開発し、製造を担当する富士通オートメーションが受注を開始した。わずかこの1年のあいだに、ホンダのASIMO、ソニーのSDR-3X、北野共生プロジェクトのPINOなど、人間に近い形と振る舞いのロボットが相次いで発表され、話題となってきた。しかし、デモンストレーションで愛嬌をふりまくこれらのロボットとはコンセプトを異にし、HOAP-1の用途は二足歩行ロボットのアルゴリズムの研究開発に据えられ、それに沿った特徴を備えている。



ロボット本体の上半身。胸部のパネルの内側にUSBハブが内蔵され、ユーザー拡張領域が用意されている。

■HOAP-1の仕様

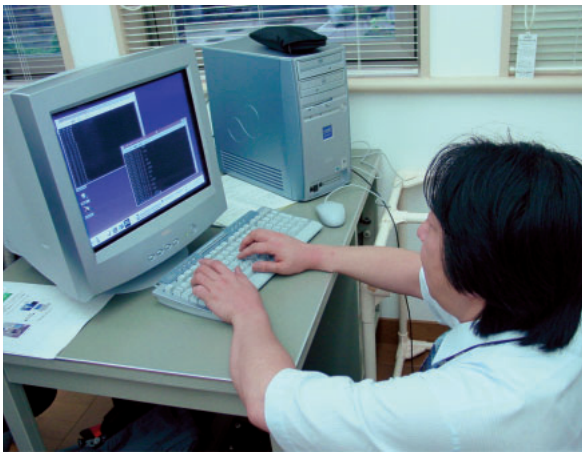
HOAP-1という名前は、“Humanoid for Open Architecture Platform-1”に由来している。その名のとおり、HOAP-1はオープンアーキテクチャとなっており、ハードウェアやソフトウェアの内部インターフェイスは仕様が公開されている。

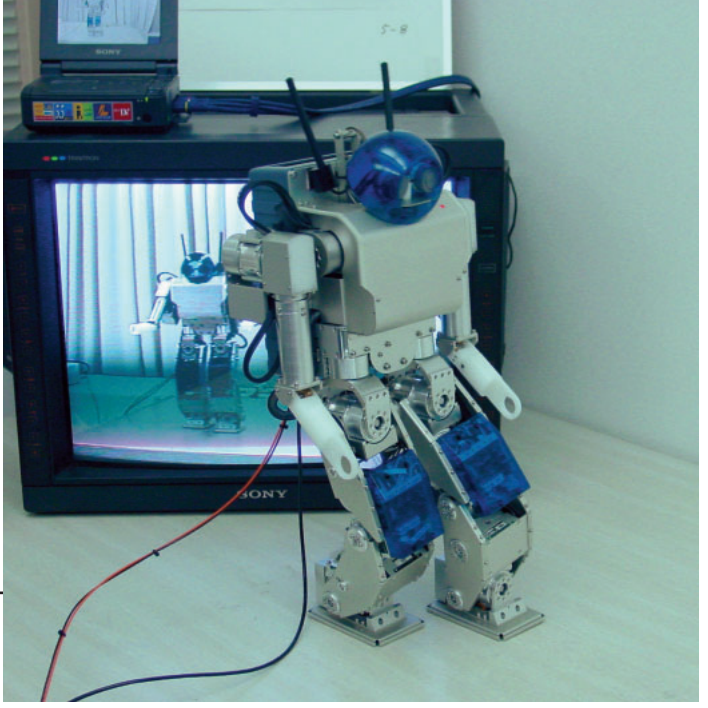
HOAP-1は、ロボット本体とロボットの制御に使用するPentium III 700MHz相当のPCから構成される。また、制御アプリケーションの開発環境のほか、シミュレータが無償で配布されるので、ユーザーが作成した制御プログラムでロボットがどのように動作するかを事前に確認できる。HOAP-1の仕様の概要を**表1**にまとめた。

HOAP-1の価格はオープンプライスで、納入時期などにより500万弱～575万円程度。販売目標は、大学や研究機関などを対象として、今後3年間で100体となっている。

HOAP-1のロボット本体は、身長48cm、体重6kgの小型で軽量なつくりになっている。このため、ロボットを吊るすクレーンなど、大掛かりな設備は必要としない。

ロボット内の各部を接続するバスにはUSBが採用され、ロボット本体内に使われているセンサやモータ類などのアクチュエータは、USBインターフェイスで揃えられている。コネクタの形状は通常よりも小さい。ロボットを購入したユーザーは、USBインターフェイスの別のデバイスを用意して入れ替え





写真左：制御に使用するPC。写真右：ロボット本体。通常の使用では、USBケーブルと電源ケーブルが接続される。

たり追加することができる。また、制御OSとしてRT-Linuxが使われている。

HOAP-1の通常の使用形態はほとんど開発作業と想定されているので、ロボット本体とPCはUSBケーブルで接続するようになっている。またロボット本体の電源はACケーブルから供給するようになっている。しかし、何体かのロボットを協調動作させたり、デモンストレーションをする場合などには、ケーブルがじゃまになるので、ロボットの背中にあるランドセルのようなボックスにオプションの中央制御ユニット（CPUボード）とバッテリーを装着して、ワイヤレスで動作させることができる。バッテリー使用時の動作時間は20分程度。ロボット頭部の背後には無線のアンテナが見えるが、ワイヤレス動作時には無線を使って背中に装着した中央制御ユニットにログインし、制御できる。また、頭部には無線カメラを2つ備えており、カメラからの映像をモニタテレビなどで見ることができそうだ。

■制御アルゴリズムは研究課題が山積

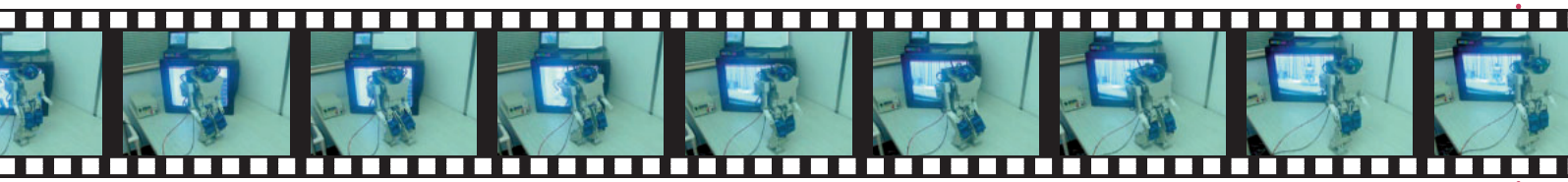
ところで、ヒューマノイド型ロボットに必要なアルゴリズムとは、いったいどんなものだろうか。たとえば、手にしている物を持ち上げたり、突然出現した水たまりをうまくよけたり、あるいは砂浜やでこぼこの道を歩いたり、といったことを人間は無意識のうちにやってのける。こうした運動をロボットにさせるためには、どのように振る舞うべきかを制御アルゴリズムとして与えなければならないが、それはまだ研究途上であり、同様の題材が山積しているのだという。

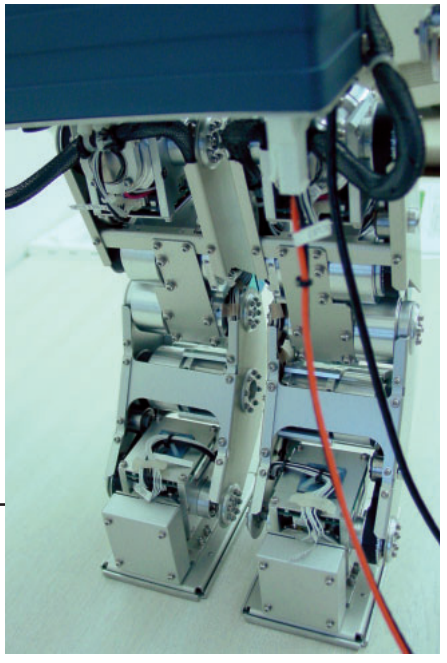
9月18日～21日に東京大学で開催された「ロボット学会」の併設展示会では、HOAP-1は生物学的アプローチと力学的アプローチによる2通りのアルゴリズム（コラム「開発者に訊く」参考）で二足歩行を披露した。HOAP-1はギミックな風貌ではあるが、力学的アプローチによる歩行アルゴリズムでは、全身を使ってしなしなと歩いているように見えた。この歩き方が力学的に合理的なのだろうか。

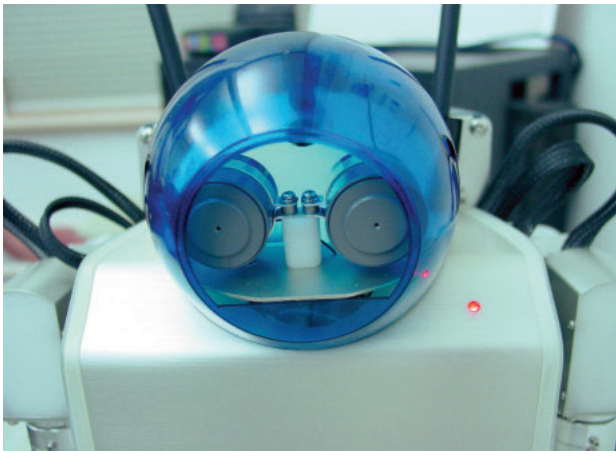
■1ミリ秒が勝負のロボット制御

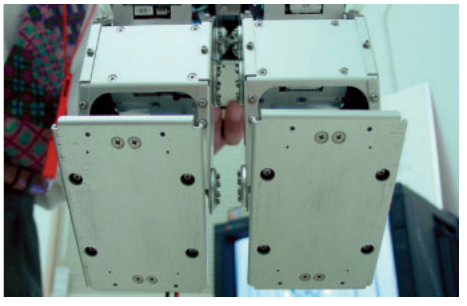
ロボットの制御では、ロボット体内のあちこちに配置されている多数のセンサから状況を集め、ロボットの現在の態勢を把握し、次にどのように振る舞うかをアクチュエータに指令する、といったことを繰り返す。

HOAP-1では、ロボット本体内にいくつかのローカルCPUが設けられ、外部の制御用PCとUSBインターフェイスで接続されている。ローカルCPUはそれぞれ、関節などに装備している3枚のセンサ制御基板からのデータの収集と、20枚のモーター制御基板への指令の伝達を担当している。いわば、USBネットワークでつながれた分散システムといったかたちになっている。ユーザーは、センサ処理基板とモーター制御基板を増設して拡張ができるようになっている。センサから値を集めたり









（左から順に）◆ロボットの背中のボックスに、オプションの中央制御ユニット、バッテリー、拡張基盤を収納する◆ロボットの脚部背面。モータが密集している◆足の裏には四隅に足底感圧センサが◆頭部の目玉には2つの無線カメラが配置されている

モーターなどに指令を出すタイミングの基礎単位となる制御周期は、1ミリ秒となっている。リアルタイム制御の世界では、制御周期が1ミリ秒であれば及第で、「使える」と判断される。

こうした「確実な」タイミングでプログラムを動作させなければならないとき、Linuxをはじめとする通常のOSのように、OSの都合でいつユーザーのプログラムが動作を開始できるか（それも1ミリ秒単位で）を保証できないのでは具合が悪い。そこで、一定時間内に必ず所定のプログラムを実行できるように、固定的な優先順位でプログラムの実行を制御するリアルタイムOSが使われる。HOAP-1では、リアルタイムOSとしてRT-Linuxが採用されているが、これは無償であることと、ユーザー人口が多く情報量の多いものを選択した結果である。

◆

「このロボットは、鉄腕アトムを作ったお茶ノ水博士のような研究者向けの製品なのですね？」と開発者に水を向けたところ、「いや、お茶ノ水博士は良心回路を組み込んだが、アトムを作ったのは天馬博士ですよ」と指摘されてしまった。こうした製品でヒューマノイドロボットの研究が進めば、いっこうに捗らない私の仕事ぶりを見て熱いコーヒーを運んできてくれるバトラーロボットが登場する日も、遠くはないかもしれない。

ロボット本体		動作指令パソコン	
身長	約48cm	CPU	PentiumIII 700MHz 相当
体重	約6kg	OS	RT-Linux
関節の自由度	6DOF/脚×2本 4DOF/脚×2本	アプリケーション	基本シミュレータ ポーザ／ビューア（CGロボットモデルの姿勢入力／表示） ロボットモデル（VRML）
センサ	関節角センサ（光学式2相インクリメンタルエンコーダ、関節角エンコーダ分解能 0.01 deg/pulse以下）	**その他**	
	3軸加速度センサ（測定範囲±2G、分解能0.005G以下（ADC分解能））	電源	DC24V×6.2A（150W）
		通信インターフェイス	USB1.0準拠 12Mbps
	3軸角速度センサ（測定範囲±60deg/s、分解能.25deg/s 以下（ADC分解能））	制御周期	1ミリ秒
		制御モード	位置／速度指令・切り換え可
	足底感圧センサ4ch／足（足底の4隅に配置）	**オプション**	
無線カメラ	2個（カラーCMOS 27万画素1/3インチ、0.8mmピンホールレンズ（最低照度：5lx）、1.2GHzマイクロ波見通伝送距離30m））	中央制御ユニット	MMX Pentium 300MHz相当 メインメモリ32Mバイト コンパクトフラッシュメモリ32Mバイト（内ユーザー領域16M）
拡張USBポート	8個	モーター制御基盤（増設用）	
制御ファームウェア	書き換え可	センサ制御基盤（増設用）	
		バッテリー充電器	

表1　HOAP-1のおもな仕様

開発者に訊く

HOAP-1の開発者を富士通研究所に訪ね、お話をうかがった。他人の芝生は青く見えるというが、ロボット開発というお仕事、なかなか楽しげだ。

（右から順に）永嶋主任研究員、植木研究員、村瀬研究員、境研究員

――HOAP-1を開発した経緯をお話いただけますか？

村瀬：富士通研究所は、新エネルギー・産業技術総合開発機構（NEDO）でやっている産学官合同のプロジェクト「ヒューマノイドロボティクス・プロジェクト」に参加していて、仮想ロボットプラットフォーム――つまり、ロボットの動作をコンピュータ上でシミュレートする、シミュレータの開発を担当していたんです。しかし、シミュレータが実世界のロボットと同じように動作するかどうかを検証するために、実際にロボットを作ったんです。作ってみた結果、ロボットの運動制御アルゴリズムの研究開発者にとっても、シミュレータだけでなくこういう小さくて小型のロボットがあれば有用だろうということになって、今年の2月から製品化の検討を開始しました。

――実機がなくてもすむようにシミュレータを作ったのに、シミュレータを検証するためにロボットを作るというと、なんだかニワトリと卵のようですが、シミュレータは実世界と合っていたでしょうか。

永嶋：合うところと、合わないところがあります。腕を振ったときの反力なんかはよく合っています。しかし、足の裏と床面が接触した瞬間の動きは合っていません。表面の効果というのは微妙で、現在のところはまだ解決できていないので、近似的な方法でやっているんです。

――現在の、ヒューマノイドロボットのアルゴリズムの研究課題とは、どのようなものなのでしょうか。

永嶋：（力学などの）理論の研究、機械の研究、それから生物学的な面からの研究。それから概念化について。日本では、100くらいのグループがこういった研究をしているのではないでしょうか。

――概念化というと、人工知能の分野とかかわる部分でしょうか？

永嶋：人間の動作から知識は芽ばえるんですよ。コンピュータの中からだけでは知識は生まれない。だから、HOAP-1のようなロボットは必要だと。

――二足歩行の具体的なアルゴリズムに関してはどうでしょう。

村瀬：突然出現した水たまりをよけたり、山登りをしたり、たくさんあります。

永嶋：ロボットの運動制御アルゴリズムは、力学的に合理的な方法を割り出すアプローチと、実世界の人間がどのように動くかを分析していく生物学的なアプローチがあります。私はこれまで、力学的アプローチでできると思っていたんですが、実際の人間はそんな歩き方をしないんですね。ロボットの作成時はそれではダメなので、生物学的なアプローチに注目しています。

――開発はどのように分担されたのでしょうか？

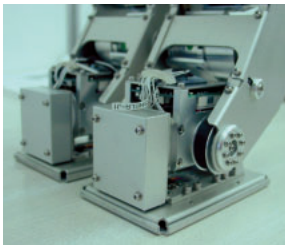
村瀬：私はハードウェアで、組み立てたり、分解したり。この席にはいませんが、境君がファームウェアを担当しました。

植木：私は、プラットフォームの担当で、無線モードで動作できるようにブート部分を作ったり、プログラム間で使用できる共有メモリを実現したり、リモートログイン操作ができるようにしたりしました。

永嶋：私はプランニングです。

――ロボットを作っていると、おもしろいこともあるのでしょうね。

永嶋：それはもう。うまく制御できなくて暴れちゃったり。脱臼もしますよ。二足歩行をうまく制御できないと、つまずいて、足の甲が捩じれて捻挫するんです。赤ん坊と同じです。



HOAP-1の足首。簡単に捻挫するようには見えないが……。

――今後にお考えになっているテーマは何ですか？

村瀬：うーん、またロボットを作れるといいですね。

永嶋：自分の老後を心配してましてね。150歳になって、朝、新聞を持ってきてくれるロボットを作りたいですね。他人の世話にならなくても、自分の力で暮らせる。実用のロボットを目指している会社の方向とも合っているし。

植木：入力手段、出力手段としてのロボット。人間のように言葉を話してくれる。

――エレベータの呼出しボタンに手が届かないから押してくれって、ロボットに頼まれたり。

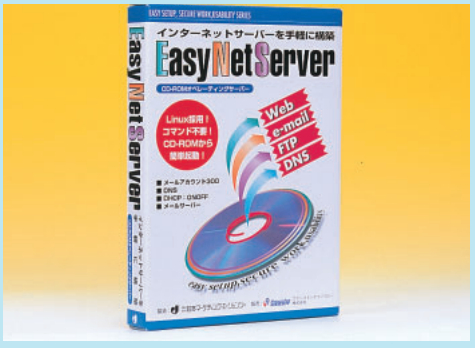
植木：人間はペットの世話をすることで癒される面がありますから、それもいいですね。

（敬称略）

RT-Linuxで制御 ヒューマノイドロボット HOAP-1

Products

CD-ROMからLinuxを起動するインターネットサーバソフト



Easy Net Server

インストールから設定まで、サーバ構築にはそれなりの労力が必要だ。Easy Net Serverを利用すれば、CD-ROMから起動して簡単な設定を行うだけで、インストールなしでインターネットサーバを立てられる。既存のWindowsパソコンを利用した小規模なサーバ構築用ソフトだ。

製品名	Easy Net Server
価格	3万8000円
問い合わせ先	プリンストンテクノロジー株式会社 TEL 03-3863-7131 http://www.princeton.co.jp/

インターネット上に公開するサーバを簡単に構築できるソフトウェア「Easy Net Server」がプリンストンテクノロジーから発売された。

WindowsパソコンにEasy Net ServerのCD-ROMを挿入し、そこから起動することで、Web／FTP／メール／DNS／DHCPサーバ機能を備えたサーバとして稼働する。

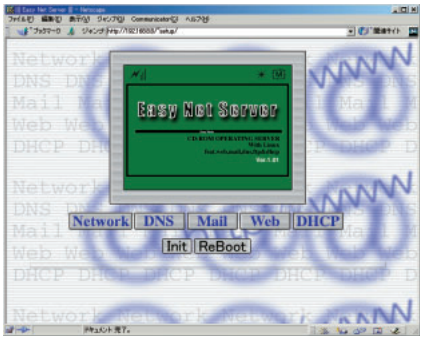
Easy Net ServerのOS自体はCD-ROM上にあり、OSやプログラムは、CD-ROMから読み込まれるため、ウィルスや事故などでファイルが壊れることはない。設定ファイルやユーザーデータをWindows上のファイルとして書き込むため、元のWindowsの環境を壊さずに運用できるというのがポイントである。すなわち、Easy Net Serverを終了し、CD-ROMを外してハードディスクから再起動すると元のWindowsが立ち上がる。

面倒なインストール作業は不要

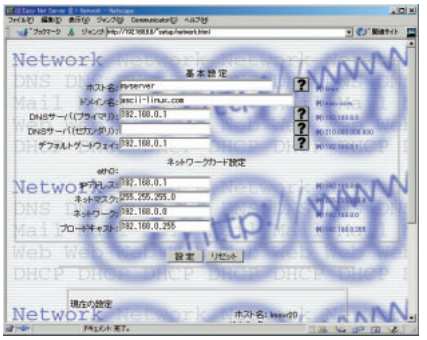
Easy Net ServerのCD-ROMから起動すると、コンソール上にLinuxの起動メッセージが表示され、セットアップを行う。少し経つと画面上に10桁のキーナンバーが表示され、パスワードの入力待ちとなる。

このキーナンバーをメモして、別のPCからサポートWebページにアクセスし、ユーザー登録画面でユーザーの情報とキーナンバーを登録する。しばらくするとメールでパスワードが送られてくるので、それをEasy Net ServerのPCに入力する。

正しいパスワードであれば先に進み、Linuxの設定ファイルをハードディスクに書き込み始める。コンソール上に「login:」が表示されれば、Easy Net Serverが起動されている。

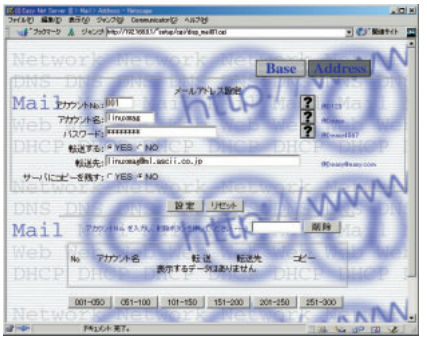


画面1　Easy Net Serverのセットアップメニュー
各サーバの基本設定を行うメニュー。タイトル部分はカーソルが機能ボタン上に乗るとそれに合わせて画面が変わる。

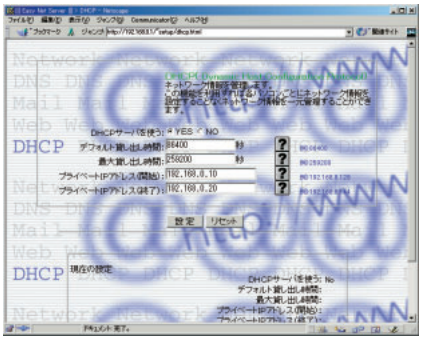


画面2　Networkメニュー
ネットワーク情報を設定する。インターネットサーバにする場合は、グローバル固定IPアドレスが必要。

画面3　DNSサーバメニュー
DNSのCNAME機能を使って、wwwやmailといった複数のホスト名を1つのサーバに割り当てることも可能。



画面4　MailサーバメニューのAddressタブ
メールアカウントは最大300個まで作成可能。ほかのメールアドレスへ転送することもできる。



画面5　DHCPサーバメニュー
クライアントPCのネットワーク設定を「自動的に取得」にすることで、サーバからIPアドレスを割り当てられる。

なお、この作業は最初の1回だけ行うもので、2回目以降の起動に際しては、パスワードは要求されない。

次に、ネットワークアドレスの設定を行う。ここからは、別のPCのWebブラウザから行う。セットアップ直後のIPアドレスは、192.168.8.8になっているので、http://192.168.8.8/~setup/index.htmlに接続する。認証用のダイアログが表示されるので、マニュアルに書かれているユーザー名とパスワードを入力すると、Easy Net Serverのトップメニューが表示される（**画面1**）。

左下の「Network」のボタンを押すと、設定画面（**画面2**）が表示されるので、実際に使用するネットワークに合わせて情報を入力する。

あとは、各メニューにあるそれぞれのサーバ機能を設定していけばいい。

DNSサーバの設定（**画面3**）では、基本設定以外に別のページで、ホスト名とIPアドレスの対応（IP Map）、ホストの別名（CNAME）の定義を行う（それぞれ16個まで）。

メールサーバの設定では、自己ドメイン名によるメールアカウントを300個まで発行することができる（**画面4**）。

クライアントにIPアドレスを自動的に割り当て、DNSのIPアドレスなどのネットワーク情報を提供するDHCPサーバの設定では、DHCPの有効期間と割り当てるIPアドレスの範囲を指定する（**画面5**）。この機能はインターネット向けではなくイントラネット内で利用する場合に用いる。

FTPサーバに関しては特に設定する項目はない。anonymous（匿名ユーザー）のFTPログインも可能だし、作成したメールアカウントを利用してFTPでログインすれば、そのユーザーのホームディレクトリにファイルを作成できる。public_htmlディレクトリにHTMLファイルを置けば、Webブラウザから「http://サーバ名/~ユーザー名/」でアクセスできる。

Webサーバは、Easy Net Serverを起動した時点で、Apacheがすでに動作している。FTPによって、/home/httpd/htmlディレクトリ以下にHTMLファイルを転送すればよい。

なお、動作しているApacheは、CGIやPHPを使ったWebページにも対応している。

Easy Net Serverでは、Windowsのファイルシステムをマウントし、LINUXDATというフォルダに各種設定ファイルやユーザー用のデータを記録している。最初のセットアップ時に、約80Mバイトほど消費するようだ。利用するWindowsマシンでは、Webサーバやメール、FTPなどでファイルを置く分を考慮して、必要な容量を空けておこう。

Easy Net Serverは、カーネル2.2.15を使用したTurbolinuxベースで開発されているようだ。メニューには、簡単に設定が行えるようにするために、あえて少ない機能しか搭載されていないが、Linuxの知識が十分あればカスタマイズして利用することも可能だろう。

InetBase IT Server

Web、メール、DNSなどのサーバ機能をインストール済みで、導入後すぐに使えるのが魅力のアプライアンスサーバのなかでも、本機は、それらに加えてWebグループウェアまで搭載した多機能サーバである。

製品名	InetBase IT Server
価格	39万8000円
問い合わせ先	株式会社AIMCOM TEL 03-5434-3412（総販売元：大明OAビジネス営業部） http://www.inetbase.to/

Webグループウェアを搭載したアプライアンスサーバ「InetBase IT Server」（以下InetBase）が、AIMCOMから発売された。

2ポートのネットワークインターフェイスを、それぞれLANとWAN（インターネット）に接続することで、Webやメールなどを提供するインターネットサーバ、ファイアウォールを備えたプロキシサーバ、グループウェアを利用したイントラネットサーバ、といった機能をすべて備えたオールインワンサーバである。

OSにRed Hat Linux 6.2Jを採用し、**表1**のような基本サーバ機能を備えている。カーネルを2.2.18にバージョンアップして、LCDディスプレイ対応などのパッチを当てているほか、必要なサービスに対応するソフトがインストール済みだ。システム管理は、Webブラウザから行えるようになっている。

CPUは、Celeron 667MHzを搭載。メモリは128Mバイト。ハードディスクは20GバイトIDEドライブを2台搭載し、ミラーリングによってデータを二重化することで信頼性を高めている。

背面には、LANとWAN用の2つの10/100BASE-Tインターフェイス、UPS接続用のRS-232C、バックアップテープ装置などを接続するSCSI-2インターフェイスが用意されている。

本体正面には、16文字×2行の表示が可能なドットマトリクスLCDディスプレイと4方向の矢印キーがあり、設置時のネットワークアドレスの設定などが行えるようになっている。

電源を入れると、本体正面のLCDディスプレイに起動時の状況が逐次表示されていく。起動が終了し「InetBaseITServer」になった時点で、矢印キーを使ってネットワークの初期設定を行う。この手法は、Cobalt QUBEなどと同様にキーボードとモニタが不要で、シャットダウン処理などのメンテナンスが行えることと、エラー発生時にLCDにステータスを表示することができる便利なものだ。

サーバ機能	ソフトウェア
DNSサーバ	bind-8.2.3
Webサーバ	apache-1.3.12
メールサーバ	Postfix
POP3サーバ	imap-4.7
IMAP4サーバ	imap-4.7
DHCPサーバ	dhcp-2.0
プロキシサーバ	squid-2.2
ファイルサーバ	samba-2.0.7-ja
FTPサーバ	wu-ftpd-2.6.0
SQLサーバ	postgresql-6.5.3

表1　InetBase IT Serverの基本サーバ機能

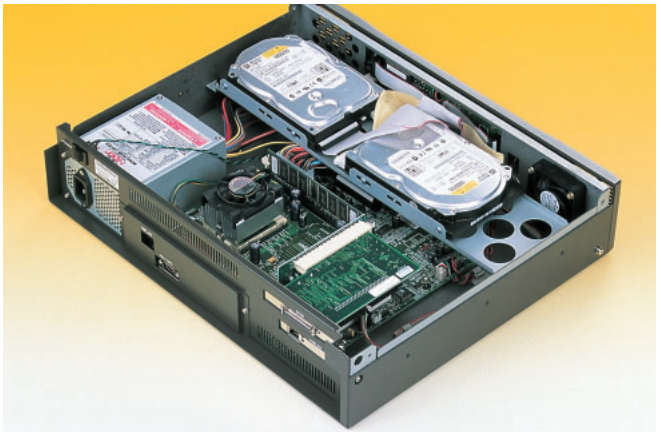


写真1　InetBase IT Serverの内部
2台の20Gバイトのハードディスクをミラーリングで使用。PCIスロットには、SCSIコントローラとLANカードが装着されている。

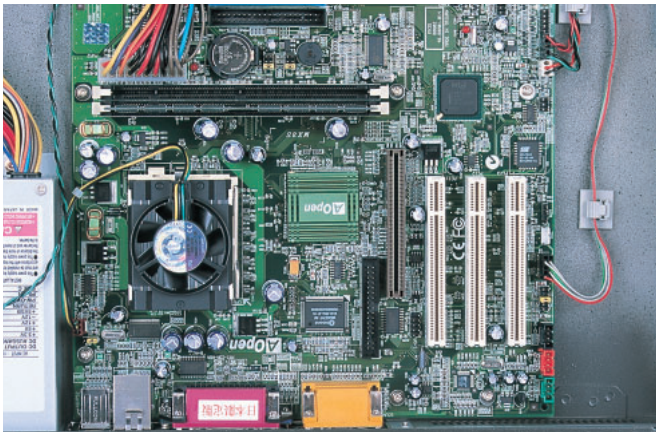
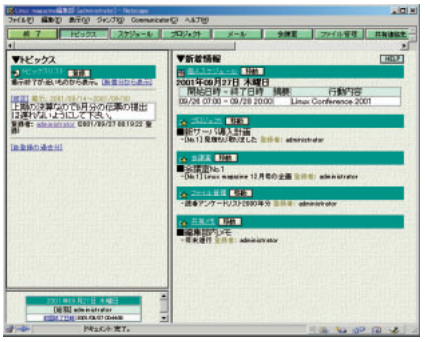
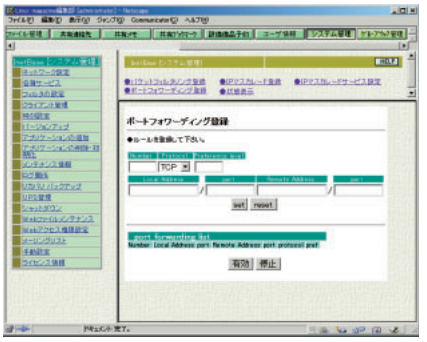


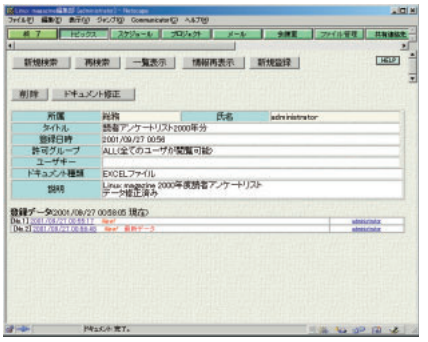
写真2　マザーボード
チップセットはIntel 815Eを使用。USBやPS/2インターフェイスなども装着されているが、本体背面で塞がれており利用できない。



画面1　Webグループウェアのトピックス
ユーザーがチェックしやすいように、掲示板やスケジュールなどの更新された新着情報が表示される。



画面2　システム管理のフィルタの設定
パケットフィルタリングやIPマスカレードなど、LANとWANの接続に必要なファイアウォール機能の設定。



画面3　ファイル管理機能の登録データ
サーバに登録されたデータファイルの検索やダウンロード者の記録が可能。再登録の履歴も明確だ。

携帯にも対応するWebグループウェア

InetBaseには、同社が開発した「GWW2000」をベースにしたWebグループウェアがインストールされており、ネットワークに接続すればすぐに利用可能になっている。PostgreSQLをDBエンジンとし、**表2**のように十分な機能を標準搭載しているが、必要に応じてアプリケーションの追加が行えるようになっている。

同社のWebページから現在12種類が提供されており、無料の追加アプリケーションとして、日報、タイムカード、チャットなどが、有料のものには、ワークフロー、問題管理、受付メール管理などがある。なお、希望ユーザーに対して、追加アプリケーションの開発仕様・開発支援ツールが提供されているので、独自機能の追加も可能だ。

iモード、J-SKY、EZwebといった携帯電話のWebブラウザから、InetBaseのグループウェアの以下の機能を利用することもできる。ただし、携帯からアクセスするにはインターネット上にグループウェアを公開することになるので、セキュリティ対策を十分に考慮する必要があるだろう。

- **トピックス閲覧・登録（一部制限あり）**
- **個人スケジュール閲覧・登録**
- **共有メモ閲覧**
- **共有連絡先閲覧**
- **ユーザー情報閲覧**

なお、同社ではダイナミックDNSサービスを有償で提供しており、フレッツ・ADSLやフレッツ・ISDNでグローバル固定IPアドレスがもらえなくても、これを利用すれば独自ドメインでの運用が可能になる。

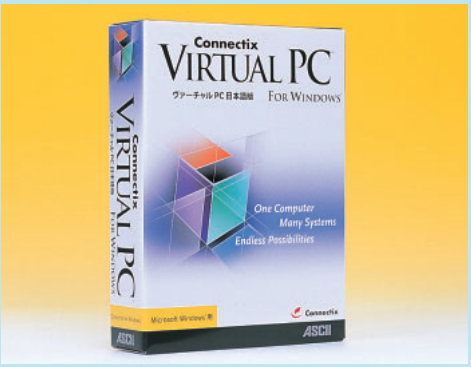
InetBaseは、1台でほぼすべてをまかなえるコストパフォーマンスの高さを考えると、これからインターネットサーバを導入する小規模の事業所などにお勧めだ。

項目	機能
トピックス	情報伝達のための掲示板と、前回終了時以降の新着情報リストの一覧表示
スケジュール	カレンダーと個人／グループ／全体のスケジュール管理
プロジェクト	各プロジェクトのToDoリストの管理と進行状況のグラフ表示
メール	IMAPにも対応するWebメール。共有連絡先のメールアドレスとの連係が可能
会議室	Webベースで議論するための電子会議室。投稿内容をスレッドで表示
ファイル管理	データベースと連動したファイル管理システム。登録された文書の検索・表示、修正履歴の記録が可能
共有連絡先	氏名、会社名、住所、電話番号、メールなどの登録、検索
共有メモ	グループごとや全体で覚えておきたい情報のメモ
共有ブックマーク	みんなで使える便利なWebページを登録しておく。カテゴリごとの分類が可能
設備備品予約	会議室や社用車などの共用設備の予約システム
ユーザー情報	各ユーザーの個人情報の登録・閲覧・検索

表2　InetBase IT Serverのグループウェア機能

CPU	Celeron 667MHz
チップセット	Intel i815E
RAM	128MバイトSDRAM（PC100、DIMMスロット×2）
ハードディスク	20GバイトIDE×2基（RAID 1ミラーリング）
SCSIコントローラ	SCSI-2対応PCIカード（AdvanSys ASC3050B）
CD-ROM、フロッピードライブ	なし
LCDディスプレイ	ドットマトリクス（16文字×2行）
ネットワーク（LAN側）	オンボード10/100BASE-T（Intel 82562）
ネットワーク（WAN側）	10/100BASE-T（Realtek RTL8139）
インターフェイス	RS-232C×1
PCIスロット	2スロット（SCSIカードとネットワークカードで使用）
キーボード／マウス	なし
サイズ（mm）、重量	428（W）×380（D）×88（H）
電源	100W

表3　InetBase IT Serverの主な仕様



Virtual PC for Windows日本語版

Linuxを勉強したいが、Linux専用PCを用意するには抵抗がある。LinuxとWindowsのデュアルブート環境では、再起動に時間がかかる。そんな方にはメインマシンのWindows環境はそのままで、仮想PC上でLinuxを稼働することができるソフトを使ってみたらどうだろうか。

製品名	Virtual PC for Windows日本語版
価格	4万2800円
問い合わせ先	株式会社アスキー ネットメディア事業部 TEL 03-5351-8652 http://win.virtualpc.jp/

1台のPC上で複数のOSを動作させることが可能な仮想PCエミュレーションソフト「Virtual PC for Windows日本語版」（以下Virtual PC）が、アスキーから発売された。

Mac上でWindowsのソフトを動かすことができるVirtual PC for Macintoshを開発しているConnectixの製品で、そのノウハウを生かして、Windowsプラットフォーム上に移植されたものだ。同種の製品として、VMwareが有名だ。

Virtual PCは、Windowsマシンにインストールして使用する。これをホストOSと呼び、Windows 2000／NT 4.0またはWindows Meが利用可能だ。

メニューから仮想PC用のハードディスクを定義し、新規PCを作成するウィザードに従って仮想PCの環境を構築する。そして起動ボタンをクリックすれば、ウィンドウが開き、その中に仮想PCが誕生する。あとは、使用したいOSのCD-ROMをセットしてインストールを行えばよい。このインストールされたOSをゲストOSと呼ぶ。ゲストOSとして**表1**のような多彩なOSが利用可能である。

新規PCは、複数個作ることが可能で、同時に実行することもできる。**画面1**では、Windows 2000のマシンに、3台の仮想PCを作成し、それぞれKondara MNU/Linux 2.0、Vine Linux 2.1.5、Windows 98SEをインストールし、実行しているところだ。

Virtual PCのメインウィンドウ（**画面2**）では、各PCのステータスと実際の動作画面が縮小されて表示されている。

MS-DOS、PC-DOS
Windows 3.1
Windows 95
Windows 98
Windows Me
Windows 2000 Professional
Windows 2000 Server
Windows 2000 Advanced Server
Windows NT 4.0 Workstation
Windows NT 4.0 Server
Linux

表1　Virtual PCが対応するゲストOS

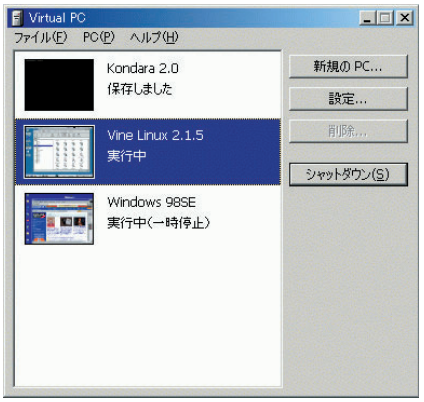
仮想PCのカスタマイズ

Virtual PCがエミュレートするPCの仕様は、**表2**のようになっている。CD-ROMやフロッピードライブ、シリアル／パラレルインターフェイスなどの、ゲストOSに割り当てられるデバイスは、基本的にホストOSの資源を共有して使用することになる。

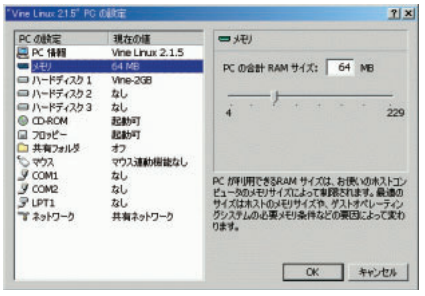
実際に仮想PCに割り当てられた情報は、**画面3**のようになっている。ホストPCには、320Mバイトのメモリを搭載しているが、この画面では、4～229Mバイトまでをゲストに割り当て可能だ。

ハードディスクは、Virtualディスクウィザードによって作成したイメ

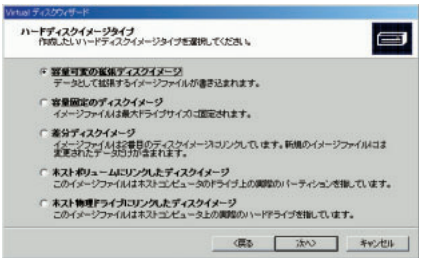
画面1　Virtual PC上で3つのOSを実行しているようす



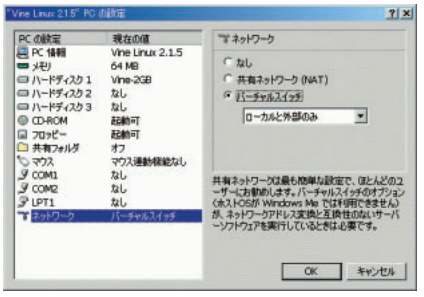
画面2　Virtual PCのメインウィンドウ
ゲストOSの実行／一時停止は自由に行える。ステータスを保存すれば、レジュームのように保存前の状態で再開できる。



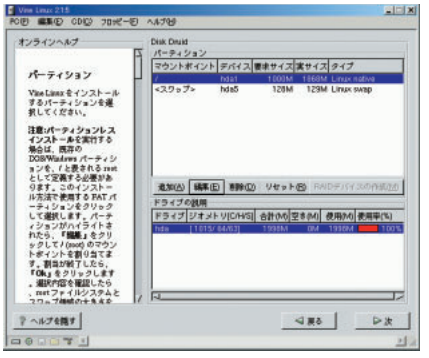
画面3　メモリサイズの設定
ゲストOSごとに割り当てるメモリを変更可能。



画面4　Virtualディスクウィザード
ゲストOSのハードディスクは、ホストOSのファイルやパーティション、物理ドライブに割り当てられる。



画面5　ネットワークの設定
バーチャルスイッチで「ローカルと外部のみ」を選ぶと、ゲストOSから直接外部と接続しているように通信可能。



画面6　Vine Linuxをインストール

ージを割り当てる。**画面4**のように5種類のタイプから選ぶことができるが、通常は容量可変の拡張ディスクイメージを使用する。このタイプは、ホストOS上に1つのファイルを作成し、それを仮想ドライブとして利用するが、ゲストOSがデータを書き込んだ部分に応じて、ファイルサイズが拡張されていくものだ。最初にドライブの容量の定義は行うが、ドライブを作成した時点では、数Mバイトの小さなサイズとなっている。

LANに関しては、共有ネットワークとバーチャルスイッチの2種類が用意されている（**画面5**）。共有ネットワークは、ゲストOSにDHCPによってプライベートアドレスが割り当てられ、NAT機能によってホストPCのネットワーク接続（LAN、ダイヤルアップ）を利用することができる。

バーチャルスイッチは、ホストOSのLANとは独立したLANのように使えるもので、**画面5**のようにオプションで「ローカルと外部のみ」を選ぶと、ホストPCのLANカードをゲストOSのもののように利用して外部と通信することが可能である。「ローカルのみ」を選ぶと仮想PC同士での通信に限られる。外部と通信する場合でも、ホストPCのLANカードとは別のバーチャルなMACアドレスが使われる。

ハードディスクイメージに2Gバイトを割り当てた仮想PCで、Vine Linuxのインストーラを起動したのが、**画面6**だ。

なお、ゲストPCの表示ウィンドウのPCメニュー、または画面2の右下のボタンから、シャットダウンを選ぶと、「PCステータスを保存」と「PCの電源を切る」が選択できる。ステータスの保存なら、現在のゲストOSの状態をそのままハードディスクに保存する。この場合は、そのゲストOSを起動すると、先ほど保存した時点に復帰することができる。

メーカーによると、仮想PCはホストマシンの80〜90%程度の性能だという。

このようにVirtual PCを利用することで、複数のOS間での通信の実験や、ソフトウェアのデバッグ環境としてなど、メインのOSには影響を与えずに安全な環境で実行が可能だ。

CPU	ホストPCと同じ
RAM	最大512Mバイト（ホストPCのメモリから割り当てる）
ハードディスク	IDEタイプを最大3台まで割り当て可能
ハードディスクイメージ	容量可変の拡張イメージ、容量固定のイメージ、差分イメージ、ホストボリュームにリンクしたイメージ、ホストドライブにリンクしたイメージを利用可能
CD-ROM	ホストPCのCD-ROMをATAPIインターフェイスとして使用
フロッピードライブ	標準のPCフロッピードライブをエミュレート
グラフィックス	S3 Trio 32/64 PCI SVGAをエミュレート（4Mバイト）
シリアルポート	COM1、COM2の2ポートをホストPCのRS-232Cに割り当て、またはテキストファイルにリダイレクトが可能
パラレルポート	LPT1をホストPCのパラレルポートにリダイレクト
サウンド	Creative Sound Blaster 16 PCIのエミュレート（DSP、FM音源）
ネットワーク	DEC（Intel）21041をエミュレート

表2　Virtual PCがエミュレートする仮想PCの仕様

Distribution

新着ディストリビューション

LASER5 Linux 7.1

「ベターRed Hat Linux」を謳う老舗ディストリビューションLASER5 Linuxの最新バージョン7.1のリリースだ。かな漢字変換ソフトにATOK XとWnn6が採用され、Red Hat Linux 7.1の日本語環境がさらに使いやすくカスタマイズされている。付属のマニュアルは約500ページという分量で質も高い。製品としてのバランスの良さはさすが老舗ディストリビューターという感がある。

Red Hat Linux β版「Roswell」

Linux業界を牽引するディストリビューションRed Hat Linuxは、次期バージョンのリリースに向けて、着々と準備を進めている。次期バージョンはマイナーバージョンアップであるにも関わらず、いくつかの部分に大きな変更がほどこされている。次期バージョンのβ版であるRoswellから、新しいRed Hat Linuxの新機能を探ってみよう。

Turbolinux 7 Workstation

Turbolinuxの新バージョンは、日本語デスクトップ環境としてのLinuxを一段高いところへ引き上げた。Windowsで使われているフォントに準拠したRicohフォントのバンドルにより、ルック＆フィールはWindows環境に伍するレベルとなっている。Windowsへ追いつき追い越すこと目指すTurbolinuxの視界は良好だ。

LASER5 Linux 7.1

レーザーファイブは、8月24日にデスクトップ用ディストリビューションLASER5 Linux 7.1（以下、LASER5 7.1）をリリースした。

LASER5 7.1には、標準デスクトップ環境向けの「デラックス」と、商用のかな漢字変換ソフト「ATOK X」と開発環境を追加した「Devel」という2つのバージョンが用意されている。

LASER5 7.1は、基本コンポーネントにカーネル2.4.3、glibc2.2.2、XFree86 4.0.3を、デスクトップ環境にGNOME 1.2.4とKDE 2.1.1を採用している。

LASER5 7.1には、かな漢字変換ソフトなどの商用ソフト（**表1**）、約500ページのユーザーガイドが収録される。また、電話、FAX、メールによるインストールからXの起動までのサポートが、90日間件数無制限で提供される。

ベターRed Hat Linux

LASER5 7.1をひとことで言うと、「ベターRed Hat Linux」となるだろう。

Red Hatベースに開発されているディストリビューションには、VineやKondaraといったものが存在する。

しかし、LASER5 7.1は、Red Hat Linuxとの互換性を保ちながら、Red Hat Linuxの日本語環境を強化しているので、多くの部分を独自拡張しているVineやKondaraよりRed Hat Linux対応を謳うアプリケーションへの対応度は高いだろう。

日本語環境についても、Wnn6とATOK X（Devel のみに収録）という代表的な商用のかな漢字変換ソフトや、XEmacs上で利用する英和／和英の辞書ソフトeWnnがバンドルされているので不満はない。また、JBuilder5 PersonalやTiny COBOLなどをバンドルして、開発環境も充実している。

Linuxをいちから学習する人に最適

LASER5 7.1には、約500ページの「インストールガイド&活用ガイド」というマニュアルが付属する。

このマニュアルのインストール解説部分は、WindowsのドライブとLinuxのパーティションの対比から説明される。

GNOMEの使い方や、ダイヤルアップでのインターネット接続方法についての説明は詳しく、コマンドの使用方法や、マルチユーザーなどUNIX系OSについての基本事項もわかりやすく解説されている。また、linuxconfを使ったシステム管理方法も詳しく説明され、さわり程度だが、CやPerlなどの開発についても触れられている。

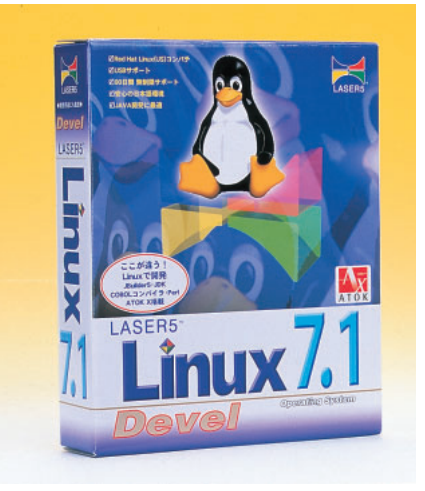
このほかにも、FAQの章では、マルチブート環境の構築方法、マルチブート環境でのTips、LILOの設定方法といった、Linux初心者が知りたい事項がしっかりと押さえられている。

アプリケーション名	Devel	デラックス
かな漢字変換ソフト「ATOK X」	○	×
かな漢字変換ソフト「Wnn6」	○	○
商用TrueTypeフォント「DynaFont 5書体」	○	○
英日辞書引きソフト「eWnn」	○	○
はがき作成ソフト「K筆」	○	○

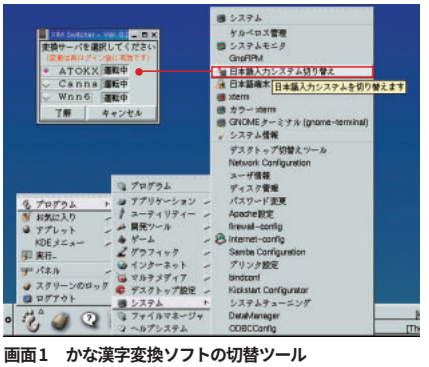
表1　バンドルされるおもな商用ソフト
LASER5 Linuxには、商用かな漢字変換ソフトや、XEmacs上で使う和英／英和辞書ソフトのeWnnがバンドルされる。

商用ディストリビューションは、とかく最新パッケージや収録される商用ソフトの数が注目されがちだが、いちからLinuxを学習しようとするユーザーには、UNIX系OSの基本から書かれたマニュアルが必要になる。

その点、LASER5 7.1付属のマニュアルは、質、量ともに充実しているので、Linuxをしっかりと身につけたい人へ、とくにお勧めできる製品である。



製品名	LASER5 Linux 7.1
価格	1万6800円（Devel） 9800円（デラックス）
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/



画面1　かな漢字変換ソフトの切替ツール
LASER5 Linuxには、複数のかな漢字変換ソフトが収録される。変換ソフトの切り替えツールをGNOMEのメニューから起動すると、簡単にそれらを切り替えられる。

Red Hat Linux β版「Roswell」

次期バージョンリリースに向けて、Red Hat Linuxのβ版（コードネームRoswell）がFTPサイトに公開された。

Roswellは、基本コンポーネントに、カーネル2.4.6、glibc2.2.3、XFree86 4.1.0が、デスクトップ環境にはGNOME 1.4.0とKDE 2.2のCVS版が採用されているほか（**表1**）、バージョンが大きく変わったSamba 2.2とSquid 2.4（**表2**）や、オプション扱いだがgcc 3.0の収録が目を引く。

ext3ファイルシステムの採用

Red Hat Linuxは、新しいコンポーネントの採用には意欲的だが、KondaraやSuSEが早くからReiserFSを採用したにも関わらず、ことデフォルトのファイルシステムに関しては、ext2のみをサポートしてきた。

しかし、Roswellのインストーラ起動すると、パーティションを新規作成する際に、ext3が選択されることに気がつく（**画面1**）。

ext3は、Linuxの標準ファイルシステムのext2にロギング機能を追加したジャーナリングファイルシステムだ。

Linuxで使えるジャーナリングファイルシステムとしては、ext3やReiserFSのほかにも、SGIのXFSやIBMのJFSが存在するが、Red Hat Linuxがext3を採用したのは、多くの既存システムで使われているext2との互換性の高さを優先したためだろう。

ブートローダGRUBの採用

ファイルシステムのほかにも、Roswellでは、標準のブートローダがLILOからGRUBに変更された（**画面2**）。GRUBはGNUが開発する高機能なブートローダで、ファイルシステムを理解できることがLILOと根本的に異なる。

LILOは、カーネルやinitrdファイルの位置をディスク上の番地のような数値で記憶しているので、カーネル再構築などの際にLILOを実行し忘れると、リブート後にLinuxが起動しなくなる場合がある。

一方、カーネルなどをファイルとして扱えるGRUBは、設定ファイルに書かれていないカーネルも、起動時にコマンドを使って読み込める。現在GRUBはext2（ext3）、ReiserFS、FATに対応していて、JFSとXFSをサポートするパッチも配布されているので、ブートローダがLILOからGRUBに変更されたことは、素直にうなずける。

基本コンポーネント	Roswell（Red Hat 7.1）
カーネル	2.4.6（2.4.2）
glibc	2.2.3（2.2.2）
XFree86	4.1.0（4.0.3）
GNOME	1.4.0（1.2.4）
KDE	2.2CVS版（2.1.1）

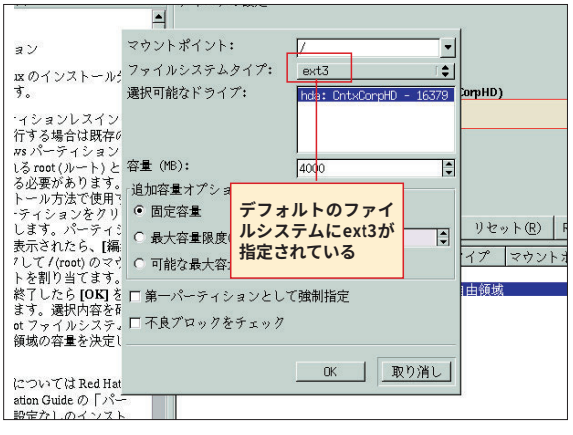
表1　Red Hat Linux β版の基本コンポーネント
Red Hat Linux 7.1と比べてGNOMEのバージョンが大きく上がっている。

サーバ名	Roswell（Red Hat 7.1）
WU-FTPD	2.6.1（2.6.1）
Samba	2.2.1a（2.0.7）
Apache	1.3.20（1.3.19）
sendmail	8.11.4（8.11.2）
PHP	4.0.6（4.0.4）
BIND	9.1.3（9.1.0）
OpenSSH	2.9p2（2.5.2p2）
Squid	2.4.STABLE1（2.3.STABLE4）
PostgreSQL	7.1.2（7.0.3）

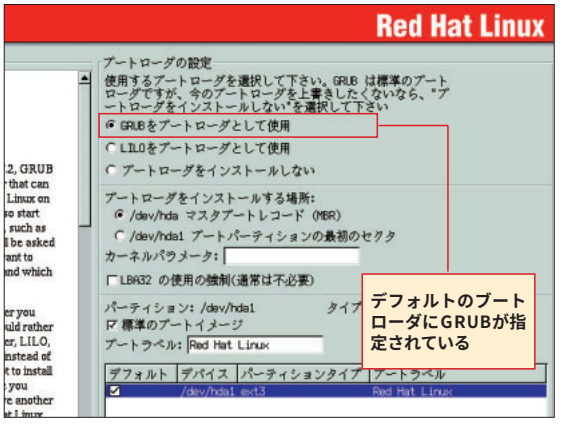
表2　Red Hat Linux β版のおもなサーバアプリケーション
PDC機能が強化されたSamba 2.2や、キャッシュの機能が拡張されたSquid 2.4などの採用が目立つ。

パーティション作成にはPartedを

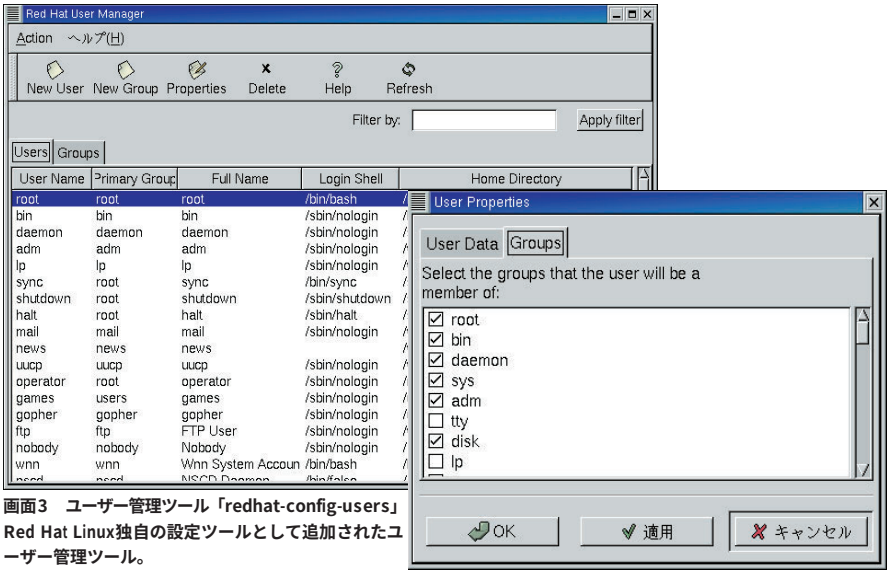
Roswellインストーラでは、パーティション作成場面のデザインが変更されたことに目を引かれるが、内部のパー



画面1　インストーラのパーティション作成場面
Red Hat Linux β版では、デフォルトのファイルシステムがext3になっている。



画面2　インストーラのブートローダ設定場面
Red Hat Linux β版では、デフォルトのブートローダがLILOからGRUBに変更されている。



画面3　ユーザー管理ツール「redhat-config-users」
Red Hat Linux独自の設定ツールとして追加されたユーザー管理ツール。

ティション編集エンジンもfdiskからPartedに変更されている。

Partedも、GRUBと同じくGNUが開発するツールだ。パーティションの作成や削除のほかに、パーティションサイズを変更できる点がfdiskと異なる。

Roswellインストーラは、インストール中のパーティションサイズの変更をサポートしていないが、内部のエンジンがPartedに変更されたことで、今後パーティションサイズの変更もサポートされると思われる。

Partedがリサイズをサポートするファイルシステムは、ext2とFATのみだが、ReiserFS、XFS、JFSのユーティリティはGPLで公開されているので、Parted用のライブラリをスムーズに開発できるだろうし、NTFSサポートの開発も始まっている。Red Hat Linuxのインストーラがpartedの機能をフルに使えるようになれば、マルチブート環境を容易に構築できるようになるだろう。

GUI設定ツールの追加

linuxconfを排除すべく、Red Hat Linuxではバージョン7.0から、apacheconfやbindconfなどRed Hatの独自設定ツールの収録が始まり、Roswellでも独自設定ツールがいくつか追加された。

redhat-config-users（**画面3**）は、ユーザー管理を行うツールで、ユーザーの新規作成や削除、ユーザーのグループへの登録などをサポートする。

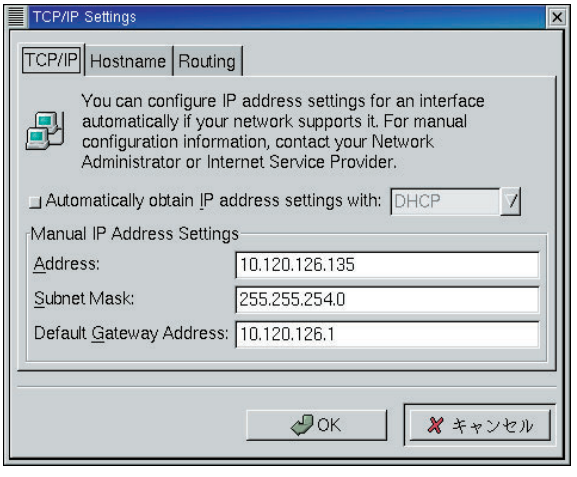
redhat-config-network（**画面4**）は、IPアドレスの設定などの基本的なネットワーク設定から、1つのネットワークアダプタへ複数のIPアドレスの割り振りなど、やや高度な設定までをサポートする。

serviceconf（**画面5**）は、システムサービスの停止や起動を行うツールで、ランレベルごとの設定もサポートする。このほかにも、システムの時刻やタイムゾーンを設定するdateconfigが新たに追加された。

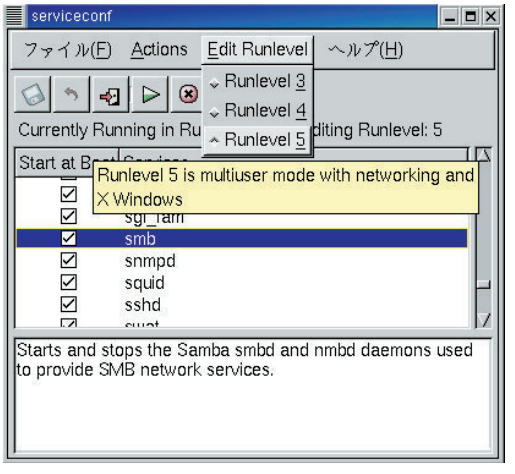
正式リリースはいつだ？

今回のバージョンアップは、Red Hat Linux 7.1からのマイナーバージョンアップのように見えるが、そうとは思えないほどインストーラは大きく拡張されている。また、デフォルトのデスクトップ環境GNOMEも、GUIシェルNautilusをサポートするGNOME 1.4にバージョンアップされている。

レッドハットによると、バグ出しの終わったRoswellは、今月中旬に正式リリース予定とのことだ。



画面4　ネットワーク設定ツール「redhat-config-network」
IPアドレスの設定や、複数のIPアドレスを1つのネットワークアダプタに割り振るといった比較的高度な設定を行う。



画面5　システムサービス設定ツール「serviceconf」
サーバアプリケーションなどのシステムサービスの起動や停止を行う。システムのランレベルに応じた設定が可能だ。

Turbolinux 7 Workstation

Turbolinuxのデスクトップ向け最新バージョンTurbolinux 7 Workstation（以下、Turbolinux 7）が、9月7日にリリースされた。

Turoblinux Workstation 6.0がリリースされたのが昨年4月のことだから、新バージョンのリリースは、約1年半ぶりのこととなる。

Turbolinux 7は、基本コンポーネントに、カーネル2.4.5、glibc2.2.3、XFree86 4.1.0を、デスクトップ環境にKDE 2.1.1とGNOME 1.4.0を採用する。

また、ATOK XやRicohのフォントを始め**表1**の収録物で構成され、インストール、サウンド設定、USB機器に関するサポートを、Webページとメールを利用して、60日間件数無制限で受けられる。

収録物	概要
Install CD	インストール用のCD-ROM（2枚）
Source CD	ソースRPMパッケージを収録するCD-ROM（2枚）
Companion CD	・かな漢字変換システム「ATOK X」 ・Ricohのフォント5書体などの商用／追加パッケージを収録
Document CD	PDF形式のFAQやコマンドガイドなどを収録
インストールガイド	GUI／CUIのインストールを詳細に解説
アプリケーションガイド	ATOK XやRicohフォントの効果的な使い方を解説
ユーザーガイド	Webブラウザやマルチメディアアプリケーションの使い方を簡潔に解説

表1　Turbolinux 7 Workstationの収録物
Turbolinux 7 Workstationは、RicohのTrueTypeフォント5書体がバンドルされるのが特徴だ。

美しいRicohのフォント

Turbolinux 7の特徴は、なんといってもRicohのTrueTypeフォント5書体がバンドルされたことだろう。

この5書体の中に含まれるTLゴシックとTL明朝は、それぞれWindowsで使われているMSゴシックとMS明朝に準拠しており、標準のデスクトップ環境KDEとあわせて、Turbolinux 7のルック＆フィールは、Windowsのそれに近づいている。

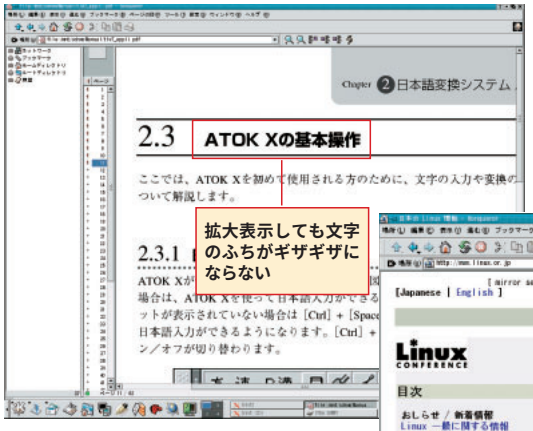
たとえば、フォントのアンチエイリアス処理を有効にしたKDE上で、ファイルマネージャKonquerorを使ってPDFファイルを開いた場合、ファイルを拡大表示しても大きな文字のふちがなめらかに表示される（**画面1**）。

また、Konquerorを使ってWebページを表示した場合も、サイズの大きな文字がきれいに表示されている（**画面2**）。なお、このWebページは前述のTLゴシックを使って表示している。

さて、小さなフォントをアンチエイリアス処理して表示すると、文字のふちがぼやけて、かえって見づらくなることがある。

Turbolinux 7にバンドルされるRicohフォントには、TrueTypeだけでなくビットマップフォントも含まれているので、小さな文字をおもに使う場合に重宝する。

たとえば、KDEのアンチエイリアス処理を無効にして、Webブラウザ Mozilla上で、ricoh-pgothicというプロポーショナルフォントを使ってWebページを表示した場合（**画面3左**）と、WindowsのIE上で、MS Pゴシックを使って表示した場合（**画面3右**）を比較しても、表示の質にほとんど差がないことがわかるだろう。



画面1　Konquerorで表示したPDFファイル
KDEのアンチエイリアス機能と、Turbolinux 7 Workstation付属のRicohフォントを使えば、ここまで文字を拡大してもきれいに表示される。画面は1280×1024の解像度の全画面スクリーンショットだ。

画面2　Konquerorで表示したWebページ
アンチエイリアスを効かせれば、PDFファイルと同様に、Webページで使われる大きな文字もきれいに表示できる。

USBデバイスのへの対応

USBデバイスへの対応が強化されたこともTurbolinux 7の特徴のひとつだ。

試しに、Panasonic KXL-RW31AN

画面3　MozillaとIEで表示したSlashdot Japanのトップページ
画面左はRicohのビットマップフォント「ricoh-pgothic」を、画面右は「MS Pゴシック」を使って表示している。2つのWebページを比べると、LinuxのWebブラウズ環境もここまできたかという感がある。

というUSB接続のCD-RWドライブを接続してマシンを起動すると、Linuxカーネルは「ピピッ」という音を鳴らしてUSBドライブを自動検出した。

ログインしてからKDEのコントロールセンターでハードウェアの情報を確認すると、このUSBドライブはLinuxのSCSIエミュレーション機能を使って、SCSI接続のドライブとして認識されている（**画面4**）。

この状態でcdrecordというCD-Rライティングソフトを使うと、あっさりとISOイメージファイルの書き込みに成功した。

次に、CD-RWメディアのフォーマットと書き込みと試してみたところ、USBドライバに不具合があるのか、CD-RWメディアのフォーマットと書き込みの最後の段階で、カーネルパニックになってしまった。

なお、KDE上でCD-RWを扱う際は、gtoasterというGUIソフトを使うようにユーザーガイドで解説されているが、収録されているgtoasterがβ版であるためか、動作が不安定でCD-Rをうまく焼けなかったことを報告しておく。

機能が強化されたインストーラ

Turbolinux Server 6.5から採用されたGUIインストーラMongooseには、パーティションのサイズを変更できるPartedが新たに組み込まれた（**画面5**）。Partedは、ext2かFATでフォーマットされたパーティションのサイズ変更に対応している。

メーカー製パソコンの多くは、ハードディスクのすべての領域を使ってWindows Meがプリインストールされているケースが多い。

このようなマシンにLinuxをインストールする場合は、既存のWindows領域を縮めてLinux用の領域を作成する必要がある。

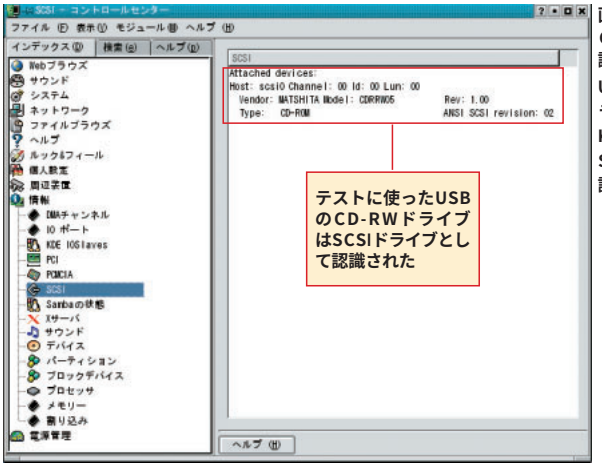
しかし、インストーラに組み込まれたPartedを使ってWindows領域をリサイズすれば、比較的簡単にWindows 9x／MeとTurbolinux 7のデュアルブート環境を構築できるだろう。

このほかにも、Turbolinux 7ではインストール中にサウンドを設定できるので、インストール直後からMP3ファイルの再生などを楽しめる。

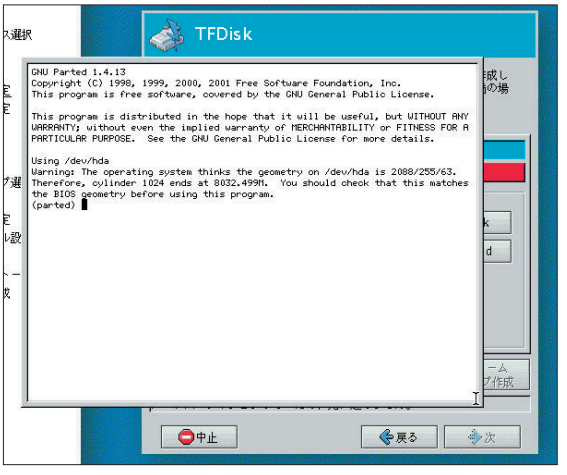
設定／管理ツールも充実

Turbolinuxの設定ツール群TurboToolsは、Turbolinux 7でさらに機能が強化された。

たとえば、パッケージ管理ツール



画面4　USB接続のCD-RWドライブの認識画面
USB接続のCD-RWドライブ「Panasonic KXL-RW31AN」はSCSIドライブとして認識されている。



画面5　GUIインストーラMongoose
Turbolinux 7のインストーラには、ext2かFATでフォーマットされたパーティションをリサイズできるPartedが組み込まれた。

画面6　パッケージ管理ツール「Zabom」
TurboToolsのひとつである「Zabom」を使えば、RPMパッケージを簡単にインストールできる。また、ネットワークを利用したパッケージのアップデートにも対応しているようだ。

画面7　人気のGUIメーラ「Sylpheed」
操作感覚はWindows用のメーラ「Becky!」に似ている。POPのほかに、IMAPやNewsも利用できる。

Zobom（**画面6**）では、起動後にRPMパッケージをCD-ROM、ハードディスク、FTPサーバのどこからインストールするかを選択する。

ここでCD-ROMを選択すると、どのRPMパッケージがインストールされているかなどのデータが読み込まれる。

このあと、画面に表示するパッケージを選択して、インストールかアンインストールのオペレーションを選択すると、依存関係をチェックして、必要なパッケージも含めてRPMパッケージをインストールしてくれる。

このツールは、「Install CD2を挿入してください」などとメッセージを出してくれるので、ユーザーは複数のCD-ROMの中からRPMパッケージを探し出す必要がない。

rpmコマンドは、パッケージの依存関係の解消をほとんどやってくれないので、Zabomのようなツールをもたないディストリビューションでは、インストール後の不便を考えて、インストール時にすべてのパッケージをインストールするという方法をとるケースが多かった。

しかし、Turbolinux 7にはZabomがついているので、もう「すべてのパッケージインストール」などという強引なオペレーションともおさらばできるだろう。

Turbolinux 7には、Zabomのほかにも、X、ネットワーク、プリンタなどの設定ツールが収録されている。

残るはオフィススイート

昨今のLinuxデスクトップ事情は、KDEやMozillaが完成度を上げ、Sylpheed（**画面7**）などを合わせて充実度が高くなっている。

加えて、Turbolinux 7にはMSゴシックやMS明朝に準拠したフォントがバンドルされたことで、これまでWindows環境に大きく水をあけられていたフォント周りの差もかなり縮まった。

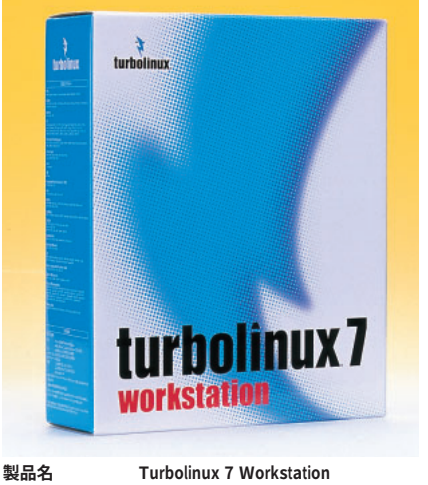
また、CanonやEPSONといったプリンタ市場で高いシェアをもつベンダーがドライバを提供し始めたことや、カーネル2.4でUSBデバイスへの対応が強化されたことで、プリンタやスキャナなど一般家庭で需要の高いハードウェアのサポートも徐々に強化されている。LinuxをWindowsの代替OSとして使うことを考えると、残りで必要なのはMS Office互換アプリケーションということになる。

Turbolinux 7にはKDE Projectが開発するKOfficeというオフィススイートが収録されているが、MS Office 2000で作成した文書を読み込めないなど、現状ではMS Officeとの互換性はまだ低い。

しかし、KOfficeを含め、商用、非商用のLinuxオフィススイートがいくつか開発されているので、将来Linuxで使えるMS Office互換ソフトが登場するかもしれない。

Turbolinux 7は、デスクトップ環境として、LinuxがWindowsをとらえてきたことを感じさせる製品だ。

本誌付録CD-ROMに収録したTurbolinux 7 WorkstationはFTP版です。非商用ソフトだけが含まれおり、製品版を販売しているターボリナックスジャパン株式会社や編集部からのサポートを受けることはできません。



製品名	Turbolinux 7 Workstation
価格	1万5800円
問い合わせ先	ターボリナックスジャパン株式会社 03-5766-1660 http://www.turbolinux.co.jp/

Distribution ▶▶▶

「Kondara MNU/Linux 2.0L（ライト）」発売開始

デジタルファクトリは「Kondara MNU/Linux 2.0L」を9月14日より発売開始した。「Kondara MNU/Linux 2.0L」は、「Kondara MNU/Linux 2.0」の基本仕様を継承する一方、商用バンドルソフトをDynaFont 5書体のみに絞り込むことによって、6800円という低価格を実現した。

基本システムには、カーネル2.4、glibc 2.2、XFree86 2.1、GNOME 1.4などを採用している。ジャーナルファイルシステム「ReiserFS」をサポートしているほか、インストールしたいパッケージを指定するだけで、依存関係をチェックし、自動ダウンロード・インストールを行うパッケージ管理ツール「mph-get」も搭載している。

デジタルファクトリ（http://www.digitalfactory.co.jp/）

Linux MLD mini用デスクトップ環境を発売開始

メディアラボは、Linux MLD miniのためのデスクトップ環境をオプションCDとして9月11日より発売開始した。

Linux MLD miniは、必要最小限のアプリケーションだけがインストールされているディストリビューションである。アプリケーションを、あとからインストールすることは可能であるが、KDEやGNOMEのような大掛かりなものをインストールするのには、手間がかかり、知識も必要になる。そこで、KDEやGNOMEなどをLinux MLD miniに簡単にインストールできるようにしたオプションCDが発売された。

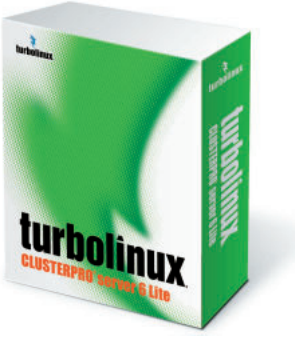
オプションCDは、KDE 2.2とKOffice 1.1などが収録された「KDE 2.2 for MLD mini」と、GNOME 1.4などが収録された「GNOME 1.4 for MLD mini」の2種類。購入の申し込みは、メールにて受付中。価格は、各1260円。

メディアラボ（http://www.mlb.co.jp/）

「Turbolinux CLUSTERPRO Server 6 Lite」発売開始

ターボリナックス ジャパンは、9月28日「Turbolinux CLUSTERPRO Server 6 Lite」を発売開始した。この製品は「Turbolinux Server 6.5」に、NECが開発した「CLUSTERPRO Lite ! for Linux」をバンドルした、2ノード構成用のクラスタシステムで、同社が2000年10月に発売した「Turbolinux CLUSTERPRO Server」に次ぐ製品である。価格は2ノード60万円。

Turbolinux CLUSTERPRO Server 6 Liteは、クラスタシステムを構成するCLUSTERPROサーバと、複数のクラスタシステムをGUIにより一括管理するCLUSTERPROマネージャ、クラスタ情報とスクリプトを事前に生成することができるトレッキングツールの3つで構成される。ネットワークで接続された各サーバの内蔵ディスク間でデータをミラーリングする機能により、サーバ間の共有ディスクを必要としないため、ハードウェア購入コストを抑えてクラスタシステムを実現することができる。さらに、トレッキングツールとCLUSTERPROマネージャを使用することで、より容易にクラスタ構成を組めるようになった。

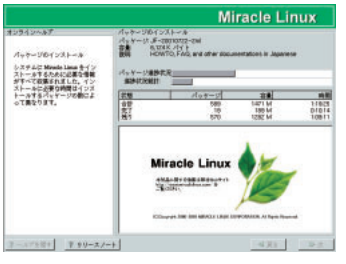


ターボリナックス ジャパン（http://www.turbolinux.co.jp/）

「Miracle Linux Standard Edition Version 2.0」10月末リリース

「Miracle Linux Standard Edition Version 2.0」（以下Miracle 2.0）は、10月下旬に「Oracle9i」と同時発売される予定。Miracle 2.0は、Oracle 9iトライアル版をバンドルし、Oracle Install Navigatorを使って、初心者にも簡単なインストールが可能だ。あらかじめOracleデータベースの使用に適したカーネルパラメータが設定してあるため、カーネルを再構築する必要はない。

基本システムには、カーネル2.4.7、glibc 2.2.3、XFree86 4.0.3、GNOME 1.2などが採用され、Windows NT/2000のドメインコントローラや分散ファイルシステムに対応したSamba 2.2.1aと更新トランザクション性能が向上したPostgreSQL 7.1.2標準搭載される。日本語の扱いで細かな問題のあったSamba 2.2系であったが、ミラクル・リナックスが日本語版を開発し、日本Sambaユーザ会に提供することとなった。



今バージョンは、Red Hat 7.1ベースに開発され、GUIインストーラが使えるようになった。また、Webブラウザでサーバをリモート管理する日本語版Webminを使って、サーバを構築・管理できる。

ミラクル・リナックス（http://www.miraclelinux.com/）

最新ディストロ対応!

/etcディレクトリの謎を解く

続々登場する最新ディストリビューションで変わったLinuxのディレクトリ構成。/etcに含まれる膨大なLinux設定ファイルを徹底解説。

Linuxカーネル2.4、XFree86 4.xを採用した新たなディストリビューションが次々に発表された。以前にも増して深く、暗い/etcディレクトリの迷宮が我々を待ち受ける。Linuxのシステム設定を極めるには、この迷宮に踏み入って無数のファイルと戦わなくてはならないのだ。すべての謎を解明するため、冒険の旅にいざ、出発しよう。

文:編集部
Text : Linux magazine
illustration : Chata Tachibana

Liunxファイルシステムの今昔

Linuxには、Kondara MNU/Linux、Red Hat Linux、Vine Linuxなど多くのディストリビューションが存在する。

昨年9月にRed Hat Linux 7がリリースされたのを皮切りに、ここ1年間で、Kondara MNU/Linux 2.0、LASER5 Linux 7.1、Turbolinux 7 Workstationなど、日本でユーザーが多いディストリビューションがメジャーバージョンアップを果たした。

そして、新しくリリースされたディストリビューションに乗り換えて、サービスを起動するスクリプトファイルの置き場所などが変更されたことや、ドキュメントの置き場所が「/usr/doc」から「/usr/share/doc」に変更されてたことなどに戸惑ったユーザーも多いはずだ。(**図1**)。

ほかにもいくつかのファイルの置き場所が変更されているが、メジャーバージョンアップしたディストリビューションは、なぜ一様にファイルの置き場所を変更したのだろうか？

FSSTND

Linuxは、もともとUNIXライクなオペレーティングシステムとして作られたので、各ディストリビューションは、UNIXに類似するディレクトリ構成をとっていた。

しかし、Linux黎明期にはディストリビューションが拠りどころにする標準のディレクトリ構成が存在せず、ディストリビューションによってコマンドの置き場所や、サービス起動用スクリプト名が異なるケースがあった。

また、当時のLinuxディストリビューションでは、BSDやSunOSなどメジャーなUNIXとディレクトリ構成が異なり、UNIXユーザーが使いづらく感じる原因となっていた。

そこで、Linuxディストリビューション間や、LinuxディストリビューションとUNIXとの間の差異を解消すべく、Slackware、Debian GNU/Linux、Red Hat Linuxなどの開発者が集い、1994年にLinux Filesystem StructureというLinux標準のディレクトリ構成が文書化された。

文書の名前はLinux Filesystem Structureだが、作成したグループ名からFSSTND(Filesystem Standard)と呼ばれるケースが多いので、本稿でもこの文書をFSSTNDと呼ぶことにする。

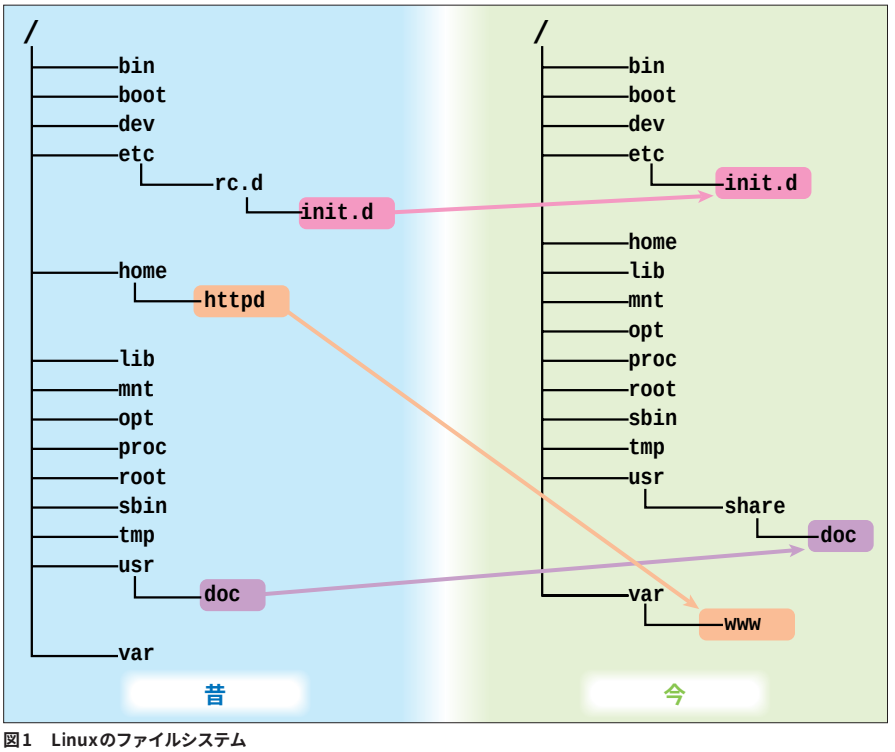


図1　Linuxのファイルシステム
左側はFHSやLSBに対応していないVine Linux 2.1.5のファイルシステム（昔）で、右側はそれらの標準規格に準拠するRed Hat 7.1のファイルシステム（今）だ。「/home/httpd」から「/var/www」への変更と「/usr/doc」から「/usr/share/doc」への変更はFHSへの準拠によるもので、「/etc/init.d」が新たに作られたのは、LSBに準拠するためだ。

FHS

FSSTNDの目的は、Linuxディストリビューションの標準ディレクトリ構成を作成することだったが、どうせ標準化するならば、Linuxだけでなく UNIX系OSすべてでディレクトリ構成を共通にしようではないかということで、FSSTND 1.2は、バージョンアップの際に、Filesystem Hierarchy Standard（FHS） 2.0と名前を変えてリリースされた。

さて、前置きが長くなったが、Red Hatなどがメジャーバージョンアップした際に、いくつかのファイルの置き場

所が変更されたのは、ディストリビューションが準拠する指針が変更されたからだ。つまり、Red Hat Linux 6.2はFSSTNDに準拠していたが、Red Hat Linux 7（7.1）はFHSに準拠することになったので、ファイルの置き場所が変わったというわけだ。

それでは、「ls -l /」を実行したときに表示されるディレクトリを、FHSがどう定義しているかを見ていこう。ディレクトリ名に「(必須)」が付いているものは、FHSがUNIX系OSに必要だと定めたもので、「(オプション)」が付いているのは、システムによってあってもなくてもよいとしたものだ。

■/bin（必須）

一般ユーザーとシステム管理者を合わせたシステム内のすべてのユーザーが実行するコマンドが配置される。配置されるのは「cat」や「sh」のようにUNIX系OSに必須のコマンドだ。

■/etc（必須）

システムの設定ファイルを配置する。ここにはサービスの起動スクリプトも含まれる。

■/home（オプション）

一般ユーザー用のディレクトを配置する。たとえば、システムに「sakura」というログイン名のユーザーがいれば、そのユーザー用のディレクトリは「/home/sakura」となる。

なお、「/home」はシステムに必須のディレクトリではなく、システム管理者が「/students」や「/staff」のように「/home」以外の名前を付けてもよい。

システム内に必ずしも存在するディレクトリではないので、アプリケーションは、「/home」の存在を前提にすべきではない。

■/lib（必須）

「/bin」と「/sbin」に配置されるコマンドが実行時に必要とする共有ライブラリを配置する。

■/mnt（必須）

システム管理者が、CD-ROMなどのローカルファイルシステムや、NFSなどを利用したネットワーク上のファイルシステムを一時的にマウントする際に使う。

■/opt（必須）

システムのインストール後に追加インストールされるアプリケーションが配置される。Linuxでは追加アプリケーションをどう定義するかが難しいところだが、商用アプリケーションがここに配置されるケースが多いのが現状のようだ。

■/root（オプション）

UNIX系OSのシステム管理者「root」用のディレクトリ。一般ユーザー用のディレクトリは「/home」以下に配置されることが多いが、管理者用「root」のものは別に用意される。

Linuxでは管理者「root」用のディレクトリが「/root」であるのが一般的だが、UNIXでは「/」になるケースが多い。

■/sbin（必須）

「shutdown」や「fdisk」などのシステム管理用のコマンドが配置される。

■/tmp（必須）

一時的に利用するファイルやディレクトリを配置する。「/tmp」にあるファイルやディレクトリは、OSの起動時に削除されることが推奨されている。なお、「/var/tmp」以下に置いたファイルやディレクトリは、OSの起動時でも削除されない。

■/usr（必須）

複数のユーザーが共有する文書、コマンド、ライブラリなどを配置する。また、配置されるファイルは読み込み専用のものに限られる。

マニュアルなどのドキュメントの置き場所は、Red Hat 6.2系のディストリビューションでは「/usr/doc」だったが、FHSは「/usr/share/doc」以下に置くよう勧告しているため、Red Hat 7系のディストリビューションは、ドキュメントを「/usr/share/doc」へ置くように変更した。

標準化プロジェクト（略称）	URL	概要
Filesystem Hierarchy Standard（FHS）	http://www.pathname.com/fhs/	UNIX系OSのディレクトリ構成の標準化を目指す
	http://www.pathname.com/fhs/1.2/fsstnd-toc.html	Linuxのディレクトリ構成の標準化を目指したFSSTND
	http://www.pathname.com/fhs/changes-2.0.html	FSSTNDがFHSに変更されたときのChangelog
Free Standards Group	http://www.freestandards.org/	オープンソースシステム全体の標準化を目指す
Linux Standard Base（LSB）	http://www.linuxbase.org/	Linuxの標準化を目指す
The Linux Internationalization Initiative（Li18nux）	http://www.li18nux.org/	Linuxの国際化標準の指針を作成する

表1　LinuxやUNIX系OSの標準化を目指すプロジェクト
FHSはUNIX系OSのファイルシステムを定める規格だが、Linuxの標準化を目指すLSBの一部分（LSBの第17章）でもある。そして、LSBはFree Standards Groupによって作業が進められている。Li18nuxは、Linuxディストリビューションの国際化対応基準を制定するプロジェクトだ。

■/var（必須）

サーバ用のファイルや、ログファイルなど更新頻度の高いファイルが配置される。Anonymous FTPサーバのデータや、Webサーバのドキュメントルートなどは、「/var」以下に配置されるのが適当だろう。Red Hat 6.2系のディストリビューションはApacheのドキュメントルートを「/home/httpd」としていたが、Red Hat 7系のディストリビューション多くは、「/var/www」に変更している。

サービス起動用のスクリプトは？

Red Hat 6.2系のディストリビューションは、サービスの起動用スクリプトを置く場所を「/etc/rc.d/init.d」としていたが、Red Hat 7系では、スクリプトを置く場所として、「/etc/init.d」が新たに追加されている。

FHSはサービス起動用のスクリプトの置き場所を定めていないが、メジャーバージョンアップ後のディストリビューションが一様に「/etc/init.d」を追加したのはなぜだろうか？

これは、Linuxディストリビューションの標準化を目指すLSB（文末のコラム参照）が、起動スクリプトを「/etc/init.d」に配置するように規定しているからだ。

ただし、「/etc/init.d」へは起動スクリプト本体を置いてもよいし、ほかのディレクトリへのシンボリックリンクでもよい、とLSBは定めている。

現状は、「/etc/init.d」へスクリプト本体を置き、「/etc/rc.d/init.d」へシンボリックリンクを張ったり、それとは逆に配置したりと、ディストリビューションによって、対応がまちまちだ。

なぜ標準化が必要なのか？

さて、Linuxのディレクトリ構成やファイルの置き場所といったファイルシステムが、FHSやLSBによってこと細かく規定されていることがわかった。では、なぜLinuxにはこのような規定が必要なのだろうか？

まず、使うディストリビューションによってファイルの配置場所が違ってはユーザーが不便だし、Linux初心者が困惑する原因にもなる。

次に、Linux用のアプリケーションを作成する開発者にとっても、ディストリビューションによって、ファイル配置に違いがあると開発の手間がかなり増える。

商用アプリケーションベンダーは、販売前に多くのディストリビューションで動作確認をとらなくてはならないが、Linuxの標準規格があれば、その規格に準じたディストリビューションで動作確認をとるだけでよい（**図2**）。

また、非商用アプリケーションの開発者にとっても、多くのユーザーが使

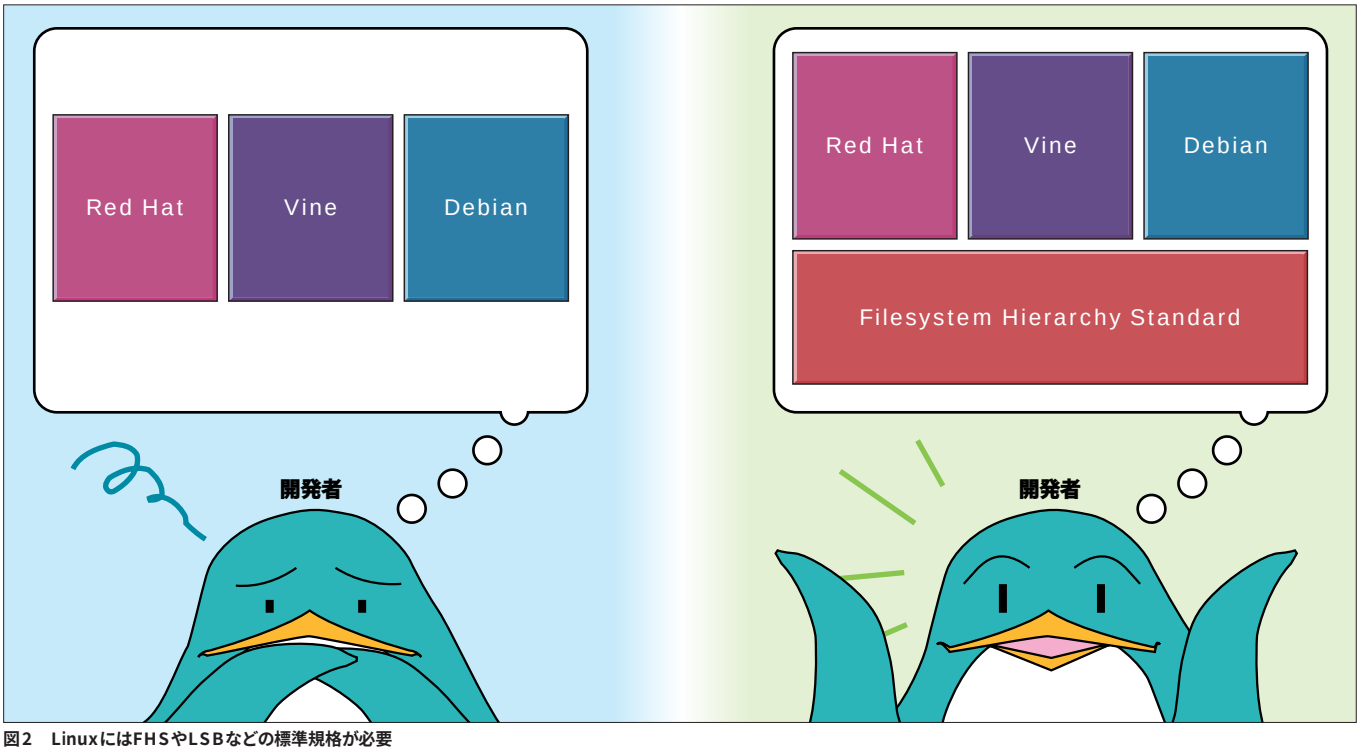


図2　LinuxにはFHSやLSBなどの標準規格が必要
Linuxは、Red Hat、Vine、Debianなど多くのディストリビューションが存在し、ディレクトリ構成などシステムの基本部分が微妙に異なる。アプリケーションの開発者はどのディストリビューションに合わせて開発すればよいのか迷うところだが、すべてのディストリビューションが準拠する標準規格があれば、それに合わせて開発すればすむ。

ってくれたほうが嬉しいだろう。

Linuxとよく引き合いに出されるFreeBSDを始めとするBSDは、このあたりに乱れがなく開発者に評判がよい。

このように、FHSやLSBが目指すUNIX系OSのファイルシステムの標準化や、Linuxシステム全体の標準化は、ユーザーや開発者がどのディストリビューションでも同じように使えることを目指して作られているのだ。

この中でも/etcは重要なのだ

「ls -l /」を実行して表示されるディレクトリの中でも、「/etc」が格納するファイルは、システムやアプリケーションの設定を司るものなので、Linuxの初心者にとっても、上級者にとっても重要だ。

Windowsではシステム設定にしても、アプリケーションの設定にしても、GUIで行うのが普通で、設定内容は人間が読めないバイナリ形式でファイルに保存されるケースが多い。

一方のLinuxは、設定内容を人間が読めるテキスト形式で保存し、設定はそれらのファイルをviやEmacsなどのエディタで編集して行うことが多い。

LinuxにもGUI設定ツールは用意されているが、ディストリビューションによって収録するツールに差異がある。

だから、結局は「/etc」に含まれる設定ファイルの書式を理解して、エディタで編集する方法がもっとも応用がきく。したがって、設定ファイルの役割や書式をしっかりと学んでおく必要があるというわけだ。

そして、いったん「/etc」に含まれるファイルを理解すれば、設定ファイルがほとんどLinuxと共通のSolarisやBSDといったUNIXシステムを使う際にもとても役に立つであろう。

/etcダンジョンを走破せよ!

本特集では、Linuxを使う際に必要になる「/etc」に含まれるファイルのひとつひとつを徹底解説する。

取りあげるのは、FHSやLSBへの対応で変更点の多い、

- **Red Hat Linux 7.1**
- **Kodnara MNU/Linux 2.0**
- **LASER5 Linux 7.1**
- **Turbolinux 7 Workstation**

の4つのディストリビューションだ。

また、設定ファイルを解説したあとで、Red Hat 6.2系ディストリビューションから、Red Hat 7系などFHSやLSBに準拠したディストリビューションへ乗り替えた際に躓きがちな事項を取りあげて、実戦的なTipsとして紹介する。なお、FHSやLSBに未対応なディストリビューションとしてVine Linux 2.1.5を取りあげた。

●

それでは、設定ファイルがうごめく/etcダンジョンの探索を始めよう。

Column

Linuxディストリビューションの標準化（LSB）

LSB（Linux Standard Base）は、「Linuxディストリビューションが最低限守るべきルール」とも言うべき規格で、Red HatやIBMといったいくつかのベンダーが参加して作成されている。

Linuxには多くのディストリビューションが存在し、それぞれの採用するライブリのバージョンが違ったり、ディレクトリ構成に微妙な違いがあったりする。

たとえば、Linuxディストリビューションは、GNU C Library（glibc）を、Cのライブラリとして採用しているが、Red Hat Linux 7.1やKondara MNU/Linuxなどはglibc2.2系、Vine Linux 2.1.5やDebian GNU/Linux 2.2などはglibc2.1系と、採用されているglibcのバージョンは、ディストリビューションによって異なる。

アプリケーションによっては動作に必要なglibcのバージョンを決めているものもある。たとえば、glibc2.1がシステムにインストールされていることを前提にして作られたアプリケーションが、glibc2.0を採用するディストリビューションで動作する可能性はとても低いし、上位バージョンのglibc2.2を採用するディストリビューションで動作する保証もない。

さて、Linuxユーザーの間でよく出る質問に、「KondaraのRPMパッケージをVineにインストールして動作しますか？」というたぐいのものがある。

このような質問に対して、「たぶん動かない」とか「異なるディストリビューションのパッケージをインストールしてはダメ」と回答されるのは、ディストリビューションによってライブラリのバージョンが違う現状があるからだ。

しかし、ユーザーからも開発者からも、ディストリビューション間の違いを意識しないでLinuxを使えることが望まれる。

そこで、LSBはライブラリなどLinuxシステムの基本部分についての標準を作り、ユーザーや開発者が、ディストリビューション間の違いを意識せずLinuxを使えることを目指しているのだ。

ここまではライブラリを例にとってLSBが必要なわけを説明してきたが、設定ファイルやサービスの起動スクリプトなどを配置する場所も、すべてのディストリビューションで同じにする必要がある。

本文で解説したFHSは、「LSBのディレクトリ構成の標準化を担当する規格」とも言うべきものなのだ。

■Linux Standard Base
http://www.linuxbase.org/

/etcディレクトリの内容を知ろう

新しいディストリビューションでは、GUIベースの設定ツールが、どんどん充実してきている。GUI設定ツールは、WindowsやMac OSのコントロールパネルに慣れたユーザーにとっては、非常に使いやすく、ありがたいツールである。しかし、各ディストリビューター独自の設定ツールに慣れてしまうと、他のUNIX系OSが使いにくいものになってしまう。また、きめ細かな設定ができない、OSの仕組みについて理解が進みにくいなど、Linuxを運用管理するうえでは、かなり不安がある。GUI設定ツールに頼ることなく設定するためにはどんな知識が必要なのだろうか？　そこで重要なのが、コマンドと/etcディレクトリである。

今回は、/etcディレクトリの内容を一挙に説明する。/etcディレクトリには、システムの中心となるソフトウェアの設定ファイルや、一部のアプリケーションが利用する重要なデータファイルなど、膨大なファイルが存在する。これらの内容を押さえることができれば、あなたのLinuxer度はぐっと上がることだろう。

なお、Linuxの運用管理上、使用頻度・重要度が極めて低いと編集部で判断したファイルや、バックアップのためのファイルなどは、今回取り上げていない。重要度の高いファイルを中心に、ジャンル別に解説したので、段階的に覚えるには最適だと思う。

名前	R	L	T	K	内容	ページ
A a2ps.cfg	○	○	○	○	テキストファイルをPostScriptプリンタに出力するa2psの設定ファイル	74
adjtime	○	○	○	○	マシンの時刻を設定するhwclockのパラメータファイル	62
alchemist/	○	○	—	—	apacheconf、bindconf、printconfなどで保存されるディレクトリ	—
aliases	○	○	—	○	メールのエイリアス設定ファイル	71
amanda/	○	○	○	—	テープバックアップツールamandaの設定ファイルを格納するディレクトリ	74
amandates	○	○	○	—	テープバックアップツールamandaがバックアップ日時を記録するファイル	74
amd.conf	○	○	—	—	ファイルシステムを自動的にマウントするamdデーモンの設定ファイル	66
amd.net	○	○	—	—	amdデーモンのネットワーク用マップファイル	66
anacrontab	○	○	—	—	24時間稼働しないシステムのためのcron、anacronの設定ファイル	66
at.deny	○	○	○	○	atコマンドの利用を禁止するユーザーを記述するファイル	67
auto.master	○	○	○	○	automountデーモンの設定ファイル	66
auto.misc	○	○	○	○	automountデーモンが利用するマップファイル	66
B bashrc	○	○	○	○	bashの起動時に実行される設定ファイル（~/.bashrc）のテンプレート	60
C cdrecord.conf	○	○	○	—	cdrecordコマンドの設定ファイル	—
cipe/	○	○	—	○	CIPE（Crypto IP Encapsulation）のスクリプトおよび設定ファイルを保存するディレクトリ	73
codepages/	—	—	○	—	Sambaのコードページが保存されたディレクトリ	72
conf.linuxconf	○	○	—	—	linuxconfのモジュールを記述したファイル	62
CORBA/	○	○	○	○	CORBA（Common Object Request Broker Architecture）に関連したファイルを格納するディレクトリ	—
cron.d/	○	○	○	○	crontabコマンドの設定ファイルを格納するディレクトリ	67
crontab	○	○	○	○	crontabコマンドの設定ファイル	67
csh.cshrc	○	○	○	○	Cシェルの起動時に実行される設定ファイル（~/.cshrc）のテンプレート	60
csh.login	○	○	○	○	Cシェルがログイン時に実行する設定ファイル（~/.login）のテンプレート	60
D default/	○	○	○	○	新規ユーザー用のデフォルト設定を記述したファイルを格納するディレクトリ	64
devfsd.conf	○	○	○	—	devfsdデバイスファイル互換性ユーティリティ（カーネル2.4系）の設定ファイル	63
dhcpc/	○	○	—	○	DHCPクライアントデーモンdhcpcdがネットワーク情報を保存するディレクトリ	68
dhcpcd/	○	○	—	○	〃	68
DIR_COLORS	○	○	—	○	カラーlsコマンドの表示色の設定ファイル	60
dumpdates	○	○	○	○	dumpコマンドでディスクのバックアップをした日にちや時刻などが記録される	75
E enscript.cfg	○	○	○	—	テキストファイルをPostScriptに変換するenscriptの設定ファイル	—
esd.conf	○	○	○	○	Enlightened Sound Daemon設定ファイル	—
exports	○	○	○	○	NFS（Network File System）サーバの設定ファイル	72
F fb.modes	○	○	—	—	フレームバッファデバイスのデータベースファイル	62
fdprm	○	○	○	○	フロッピーディスクのパラメータファイル	62
filesystems	○	○	○	○	利用できるファイルシステムのリストが記述されたファイル	63
fnrc	○	○	○	○	フォントレンダリングライブラリfnlibの設定ファイル	—
fstab	○	○	○	○	ファイルシステムとマウントポイントを記述するファイル	59
ftpaccess	○	○	—	—	FTPサーバwu-ftpdの設定ファイル	70

表見出し中のディストリビューション略称
R：Red Hat Linux 7.1／K：Kondara MNU/Linux 2.0／L：LASER5 Linux 7.1／T：Turbolinux 7 Workstation

	名前	R	L	T	K	内容	ページ
F	ftpconversions	○	○	—	—	FTPサーバで圧縮／アーカイブ転送をする際のコマンドと拡張子の設定ファイル	70
	ftpgroups	○	○	—	—	FTPサーバでのグループ名と実グループ名の対応およびパスワードを定義する	70
	ftphosts	○	○	—	—	FTPサーバにアクセスを許可／禁止するホストを設定するファイル	70
	ftpusers	○	○	○	○	FTPアクセスを禁止するユーザーを設定するファイル	70
G	gconf/	—	—	○	○	GNOME ネットワーク／コンポーネント認識構成管理システムの設定ファイルが格納されたディレクトリ	—
	gettydefs	—	—	○	○	gettyが利用できるシリアルの設定一覧	—
	gimp/	○	○	○	○	ペイントソフトGIMPの設定ファイルを格納するディレクトリ	—
	gnome/	○	○	—	○	GTK＋の国際化対応キャラクタセット定義ファイルを格納するディレクトリ	—
	gnome-vfs-mime-magic	—	—	○	○	GNOME VFSライブラリが使うMIMEのパターンファイル	—
	gnucash/	—	—	○	○	個人用財務会計ソフトGnuCashの設定ファイルを格納するディレクトリ	—
	gpm-root.conf	○	○	○	○	カット＆ペーストとマウスサーバ（gpm）のデフォルトハンドラの設定ファイル	63
	group	○	○	○	○	ユーザーグループを定義したファイル	65
	gshadow	○	○	○	○	暗号化されたグループパスワードを記述したファイル	65
	gtk/	○	○	○	○	The GIMP Tool Kit（GTK＋）の国際化対応キャラクタセット定義ファイルを格納するディレクトリ	—
	gtk-2.0/	○	○	—	—	GTK（GIMP Tool Kit）の設定ファイルを格納するディレクトリ	—
H	host.conf	○	○	○	○	名前解決の方法、優先順位の設定ファイル（libc5）	68
	hosts	○	○	○	○	ローカルで名前解決をするためのホスト名、IPアドレス定義ファイル	68
	hosts.allow	○	○	○	○	xinetd（inetd）経由で起動されるサーバのアクセス可能ホストを定義する	70
	hosts.deny	○	○	○	○	xinetd（inetd）経由で起動されるサーバのアクセス拒否ホストを定義する	70
	hotplug/	○	○	—	○	USBデバイス用ユーティリティの設定ファイルを格納するディレクトリ	63
	htdig.conf	○	○	○	—	Web索引作成システムhtdigの設定ファイル	75
	httpd/	○	○	○	○	WebサーバApacheの設定ファイルなどを格納するディレクトリ	70
	icecast/	—	—	○	○	オーディオストリーミングサーバicecastに関する設定ファイルを格納するディレクトリ	72
I	identd.conf	○	○	○	○	個人認証に使われるidentdの設定ファイル	69
	im/	○	○	○	○	MHと置き換えてMewメールリーダと併用するためのPerlスクリプトの設定ファイルを格納するディレクトリ	—
	info-dir	○	○	○	○	infoコマンドがデフォルトで表示するファイル	75
	init.d/	○	○	○	○	システム起動用スクリプトを格納するディレクトリ	59
	initlog.conf	○	○	○	○	initが出力するログに関する設定ファイル	—
	inittab	○	○	○	○	起動時のランレベルやinitの設定を記述するファイル	56
	inputrc	○	○	—	○	bashなどが利用する行入力ライブラリReadlineの設定ファイル	—
	ioctl.save	○	○	○	○	シングルユーザーモードで利用するコンソールデバイス設定（initが作成する）	—
	iproute2/	○	○	—	—	IPルーティングとネットワークデバイス用の拡張設定ツールの設定ファイルを格納するディレクトリ	-
	isapnp.gone	○	○	—	○	ISAプラグ＆プレイで認識できないリソースを記述するファイル	63
	issue	○	○	○	○	ローカルからログインする際に表示されるメッセージ	60
	issue.net	○	○	○	○	リモートからログインする際に表示されるメッセージ	60
	jfbterm.conf	—	○	—	○	フレームバッファコンソールjfbtermの設定ファイル	75
J	joe/	○	○	—	—	テキストエディタjoeに関する設定ファイルを格納するディレクトリ	—
	junkbuster/	—	—	—	○	バナー広告やクッキーデータをカットするjunkbusterデーモンの設定ファイルを格納するディレクトリ	—
K	kderc	○	○	—	○	KDEのメニューに対応したディレクトリを指定している	—
	kon.cfg	○	○	—	○	日本語コンソールエミュレータkonの設定ファイル	61
	krb5.conf	○	○	○	○	暗号化認証システムKerberosの設定ファイル	—
L	ld.so.conf	○	○	○	○	共有ライブラリを格納するディレクトリを記述するファイル	65
	ldap.conf	○	○	○	○	LDAP（Lightweight Directory Access Protocol）モジュールnss_ldapの設定ファイル	69
	ldap/	—	—	○	—	LDAP（Lightweight Directory Access Protocol）の設定ファイルを格納する	69
	lilo.conf	○	○	○	○	ブートローダLILOの設定ファイル	—
	linux-terminfo/	—	○	○	—	使われているコンソールに関するデータが格納されているディレクトリ	—
	lmhosts	—	—	○	—	SambaがNetBIOS名の解決に利用するホスト名、IPアドレスの定義ファイル	72
	locale/	○	○	—	—	起動時に利用可能なロケールデータを格納するディレクトリ	—
	localtime	○	○	○	○	マシンのタイムゾーンが書かれているバイナリファイル	—
	logcheck/	—	—	○	○	不審なアタックなどがないか調べるlogcheckの単語を保存するディレクトリ	73
	login.defs	○	○	○	○	ログインの設定	—
	logrotate.conf	○	○	○	○	ログ管理ユーティリティlogrotateの設定ファイル	76
	lpd.conf	○	○	○	○	プリンタスプーラLPRngシステムの設定ファイル	63
	lpd.perms	○	○	○	○	LPRngシステムのパーミッションを指定するファイル	63
	ltrace.conf	○	○	○	○	ライブラリトレーサの設定ファイル	—
	lynx.cfg	○	○	○	○	テキストベースWebブラウザLynxの設定ファイル	—

	名前	R	L	T	K	内容	ページ
M	mail.rc	○	○	○	○	mailコマンドの設定ファイル	—
	mail/	○	○	○	—	Sendmailが利用するデータベースファイルなどを格納するディレクトリ	71
	mailcap	○	○	○	○	metamail設定ファイル（マルチメディアメールの処理）	—
	makedev.d/	○	○	○	○	デバイスファイルの作成・管理に必要なスクリプトを格納するディレクトリ	63
	man.conf	—	—	○	—	manコマンドの設定ファイル	—
	man.config	○	○	—	○	〃	—
	mc.global	○	○	○	○	GNOMEのファイルマネージャGNU Midnight Commanderの設定ファイル	—
	medusa/	—	—	○	○	ファイル検索システムmedusaの設定ファイルを格納するディレクトリ	76
	mesa.conf	○	○	—	—	OpenGL互換3Dライブラリ Mesaの設定ファイル	—
	mgetty+sendfax/	○	○	○	○	mgetty+sendfaxパッケージの設定ファイルを格納するディレクトリ	—
	midi/	○	○	○	○	MIDIプレーヤのplaymidiなどが利用する音色ファイルを格納するディレクトリ	—
	mime.types	○	○	○	○	WebサーバApacheが利用するMIME定義ファイル	71
	mime-magic	○	○	○	○	ファイルの内容からMIME形式を判定するための定義ファイル	—
	modules.conf	○	○	○	○	modprobeが利用する、ローダブルモジュール設定ファイル	59
	modules.devfs	○	○	—	—	modules.confを元にdevfsdが自動的に生成するファイル。	—
	motd	○	○	○	○	ログイン時に表示されるメッセージを記述するファイル	60
	mph.conf	—	—	—	○	パッケージの更新を自動化するmph-getの設定ファイル	—
	mtab	○	○	○	○	マウントされているファイルシステムの一覧が書かれているファイル	-
	mtftpd.conf	○	○	—	—	リモートブートデーモンpxeで利用するMTFTPサーバの設定ファイル	73
	mtools.conf	○	○	○	○	DOSのディスクを読み書きするツールパッケージmtoolsの設定ファイル	77
	murasaki/	—	—	○	—	PCカード、USBなどホットプラグ可能デバイスに関する設定ファイルを格納するディレクトリ	—
	Muttrc	○	○	—	○	メールソフトウェアMuttの設定ファイル	—
	my.cnf	○	○	—	—	データベースパッケージMySQLの設定ファイルを格納する	—
N	namazu/	—	○	○	○	テキストサーチエンジンNamazuの設定ファイルを格納するディレクトリ	78
	named.conf	○	○	○	○	DNSサーバBINDの設定ファイル	68
	nekondara.conf	—	—	—	○	Kondara MNU/Linuxのクラスタ管理用ユーティリティの設定ファイル	—
	netatalk/	—	—	○	○	netatalk（AppleTalkファイル共有）の設定ファイルを格納するディレクトリ	72
	news/	○	○	○	○	NetNewsサーバINNの設定ファイルを格納するディレクトリ	71
	nmh/	○	○	—	○	MHをベースにしたメールツールnmhの設定ファイルを格納するディレクトリ	—
	nscd.conf	○	○	○	○	ネームサービス参照キャッシュデーモンnscdの設定ファイル	69
	nsswitch.conf	○	○	○	○	システムデータベースとネームサービススイッチ設定ファイル	73
	ntp.conf	○	○	○	○	Network Time Protocolサーバxntpdの設定ファイル	71
	nwserv.conf	○	○	—	○	LinuxをNetWareのファイル／プリンタサーバとして使用するmars_nweの設定ファイル	73
O	oaf/	—	—	○	○	GNOME 1.4以降の新しいCORBAの実装であるoafに関する設定ファイルを格納するディレクトリ	—
	odbc.ini	○	○	○	○	unixODBCに関する設定ファイル	78
	openldap/	○	○	—	○	OpenLDAP（Lightweight Directory Access Protocol）の設定ファイルなどを格納するディレクトリ	69
P	pam.d/	○	○	○	○	PAM（Pluggable Authentication Modules）の設定ファイルを格納するディレクトリ	65
	pango/	○	○	—	—	GTK+/GNOMEのフォントレンダラ。ユニコードベースで各国語に対応している	—
	passwd	○	○	○	○	全ユーザーのユーザー名、ユーザーID、グループIDなどが記述されているファイル	64
	pbm2ppa.conf	○	○	—	○	portable bitmap画像ファイルをHPのプリンタで使うPPAファイルに変換するツールの設定ファイル	—
	pcmcia/	○	○	○	○	PCカードの設定ファイルを格納するディレクトリ	63
	pgsql/	—	—	○	—	PostgreSQLの設定ファイルを格納するディレクトリ	—
	php.ini	○	○	—	—	HTML内に記述するスクリプト言語PHPの設定ファイル	—
	pine.conf	○	○	—	—	メール／NetNewsリーダpineの設定ファイル	—
	pinforc	○	○	—	—	infoファイルビューアpinfoの設定ファイル	75
	pluggerrc	○	○	—	○	Netscape/Mozilla用のマルチメディア系プラグインpluggerの設定ファイル。	—
	pnm2ppa.conf	○	○	—	○	portable anymap画像ファイルをHPのプリンタで使うPPAファイルに変換するツールの設定ファイル	—
	postfix/	—	—	—	○	MTAの一種であるPostfixの設定ファイルを格納するディレクトリ	—
	ppp/	○	○	○	○	PPPデーモンの設定ファイルを格納するディレクトリ	—
	ppxp/	—	—	—	○	PPxP（PPP接続プログラム）に関連するファイルを格納するディレクトリ	—
	printcap	○	○	○	○	プリンタの設定ファイル	—
	profile	○	○	○	○	bashがログイン時に実行する設定ファイル	61
	profile.d/	○	○	○	○	ログイン時に、profileから呼び出されて実行されるスクリプトを格納するディレクトリ	61
	proftpd.conf	—	—	—	○	ProFTPDデーモンの設定ファイル	70
	protocols	○	○	○	○	IANAで割り当てられているプロトコル番号の一覧表	—
	pwdb.conf	○	○	○	○	pwdbの設定ファイル	65

表見出し中のディストリビューション略称
R：Red Hat Linux 7.1／K：Kondara MNU/Linux 2.0／L：LASER5 Linux 7.1／T：Turbolinux 7 Workstation

名前		R	L	T	K	内容	ページ
P	pxe.conf	○	○	—	—	リモートブートデーモンpxeの設定ファイル	73
R	rc	○	○	○	○	システム起動用スクリプト	59
	rc.d/	○	○	○	○	起動時にデーモンなどを起動するスクリプトを格納するディレクトリ	59
	rc.local	○	○	○	○	システム起動時の最後に実行されるスクリプト	59
	rc.news	○	○	—	○	ネットニュースシステム（INN）の設定ファイル	71
	rc.sysinit	○	○	○	○	システム起動時にデーモンなどよりも前に実行されるスクリプト	58
	resolv.conf	○	○	○	○	名前解決手段の優先順位などリゾルバの設定ファイル	68
	rndc.conf	○	—	—	○	BIND 9用ユーティリティrndcコマンドの設定ファイル	69
	rpc	○	○	○	○	RPC（Remote Procedure Call）のサーバ名と番号の対応が記述されたファイル	65
	rpm/	○	○	○	○	RPMコマンドパッケージで利用するディレクトリ	—
	rpm2html.config	○	○	—	○	RPMリポジトリからHTMLデータベースを作成するrpm2html設定ファイル	—
	rpmfind.conf	○	○	—	○	インターネット上のRPMファイルを検索するrpmfindの設定ファイル	—
	rpmlint/	○	○	—	—	RPMファイルのエラーチェックを行うrpmlintの設定ファイルを格納するディレクトリ	—
S	samba/	○	○	—	○	Sambaに関する設定ファイルを格納するディレクトリ	72
	sane.d/	○	○	○	○	各社のスキャナを共通のAPI（SANE）で利用するための設定ファイルを格納するディレクトリ	—
	screenrc	○	○	○	○	画面管理プログラムscreenの設定ファイルテンプレート	62
	securetty	○	○	○	○	rootアカウントでログインできるターミナルを指定するファイル	61
	security/	○	○	○	○	PAM（Pluggable Authentication Modules）で利用するモジュールを格納するディレクトリ	65
	sendmail.cf	○	○	○	—	Sendmailの動作設定ファイル	71
	sensors.conf	○	○	—	○	ハードウェアモニタリングシステムlm_sensorsパッケージの設定ファイル。	—
	services	○	○	○	○	ネットワークサービス名と使用するポート、プロトコルの対応を定義するファイル	73
	sgml/	○	○	○	○	SGMLのエンティティとDTD、カタログ	—
	shadow	○	○	○	○	各ユーザーのシャドウパスワードに関する情報を記述するファイル	64
	shells	○	○	○	○	ログインシェルとして設定可能なシェルを記述するファイル	61
	skel/	○	○	○	○	ドットファイルのテンプレートを格納するディレクトリ	64
	slip/	○	○	—	—	SLIP（Serial Line Internet Protocol）ログインのための設定ファイルを格納する	—
	smb.conf	—	—	○	—	Sambaの設定ファイル	72
	smbusers	—	—	○	—	Linuxのユーザー名とSambaのユーザー名を対応づける設定ファイル	72
	smrsh/	○	○	○	—	Sendmail用制限シェルで利用できるプログラムを格納するディレクトリ	71
	snmp/	○	○	○	○	SNMP（Simple Network Management Protocol）エージェントの設定ファイルなどを格納するディレクトリ	73
	sound/	○	○	○	○	GNOMEのイベントに割り付けるサウンドファイルの設定ファイルを格納するディレクトリ	—
	squid/	○	○	○	○	Webキャッシュプロキシsquidの設定ファイルを格納するディレクトリ	71
	ssh/	○	○	○	○	セキュアシェルSSHに関する設定ファイルを格納するディレクトリ	61
	sudoers	○	○	○	○	各ユーザーが実行できるコマンドの制限を行うファイル	61
	swatch.conf	—	—	—	○	ログウォッチャswatchの設定ファイル。	—
	sysconfig/	○	○	○	○	起動時に参照されるデバイスなどの設定ファイルを格納するディレクトリ	58
	sysctl.conf	○	○	○	○	起動時にネットワークの設定を行うファイル	—
	syslog.conf	○	○	○	○	syslogデーモンの設定ファイル	—
T	termcap	○	○	○	○	以前使われていた端末定義データベースファイル	78
	timidity.cfg	○	○	—	—	ソフトウェアMIDI	—
	tripwire/	○	○	—	—	ファイルが書き換えられたかをチェックするTripwireの設定ファイルを保存するディレクトリ	79
	tux.mime.types	○	○	—	—	カーネルモードで動作するWebサーバTUXが参照するmimeタイプ一覧表	—
U	updatedb.conf	○	○	○	○	セキュアなlocate, slocateが利用する設定ファイル	—
	ups/	○	○	—	—	UPS用ユーティリティnut-clientに関する設定ファイルを格納するディレクトリ	—
V	vfontcap	○	○	○	○	Vflibで使用するTrueTypeフォントの設定ファイル	—
	vfs/	—	—	○	○	GNOME virtual file systemに関する設定ファイル群が含まれるディレクトリ	—
W	wgetrc	○	○	○	○	HTTP／FTP対応のファイルダウンロードツールwgetの設定ファイル	79
	wlan/	—	—	○	—	無線LANカード関係のカーネルモジュール用の設定ファイルを格納するディレクトリ	73
	wwwoffle.conf	—	—	—	○	Webプロキシサーバwwwoffleの設定ファイル	—
X	X11/	○	○	○	○	X Window Systemの設定ファイル群を格納するディレクトリ	62
	xinetd.conf	○	○	○	○	inetdに代わるインターネットスーパーデーモンxinetdの設定ファイルを格納するディレクトリ	69
	xinetd.d/	○	○	○	○	xinetdによって起動される各デーモンの設定ファイルを格納するディレクトリ	69
Y	yp.conf	○	○	○	○	NISの設定ファイル	—
	ypserv.conf	○	○	○	○	NISサーバのオプションを記述するファイル	—
	ytalkrc	○	○	—	—	チャットプログラムytalk設定ファイル	—
Z	zebra/	—	—	—	○	多くのプロトコルをサポートしたルーティングシステムZebra用の設定ファイルを格納するディレクトリ	—

システム起動

Linuxに限らずどんなOSでも、PCの電源を入れてからユーザーの操作を受け付けるようになるまで、何段階かの手順を経ている。ここで何が行われているかを知ることは、そのOSに関する理解を深めることでもある。

Linuxマシンの起動プロセスは、ディストリビューションによる多少の差異はあるものの、ほぼ**図1**の順序で行われている。

Linuxでは、ユーザーが実際に操作するプログラム以外にも多くのプログラム（プロセス）が動作しており、これらは、Linux本体の起動時に順次、起動されていく。その結果、Linux本体が起動する際には、とても多くの設定ファイルが参照されることになる。さらにそれらは、ディストリビューションごとに少しずつ異なっており、全貌を把握するのはなかなか大変だ。

Linuxの起動プロセスを追え

幸い、UNIX系OSの設定ファイルは、ほとんどが「/etc」ディレクトリ以下に置かれている。このセクションでは、これらの中からLinux本体の起動に関係するファイルを取り上げていこう。今回、対象とするのは、いわゆるRed Hat系ディストリビューションの最新版である。

最初のプロセスはinit

まず最初にLinuxのカーネルが読み込まれ、続いてデバイスドライバやシステムの初期化などが行われたのちに、最初のプロセスであるinitが起動される。

すべてのプロセスには、プロセスIDが振られるが、最初に起動されるinitのプロセスIDは、常に1だ。プロセスIDは、ktermなどのターミナルから、psコマンドにaxオプションを付けて、「ps ax」とすれば表示される。

プロセスは、自身の「親」にあたるプロセスによって起動されるが、initだけはカーネルによって直接起動される。プロセスの親子関係は、pstreeコマンドなどで確認できる。

initコマンド実行時に参照する設定ファイルが、inittabである（**リスト1**）。inittabの内容は、行単位に以下のフォーマットとなっている。

```
id:runlevels:action:process
```

idフィールドは、各行を区別するために付けられている。4文字まで指定できるが、昔のシステムでは2文字に制限されていたため、今でも2文字になっている。

actionフィールドには、動作に関する細かい指定を行う。

processフィールドでは、実際に実行するプログラムやシェルスクリプトをフルパスで指定する。

そしてrunlevelsフィールドは、process／actionフィールドで指定した動作を、どのランレベルで行うかを指定する。ランレベルとは、Linuxの動作状態を定義したもので、0～6までとSまたはsの8つのレベルがある。そのほかにA～Cなどを別途定義することも可能だ。これらのランレベルのうちで0、1、6は予約されており、0が停止（halt）、6が再起動（reboot）、そして1はシングルユーザーモードを表している。

そのほかのランレベルについては、ディストリビューション間で差がある

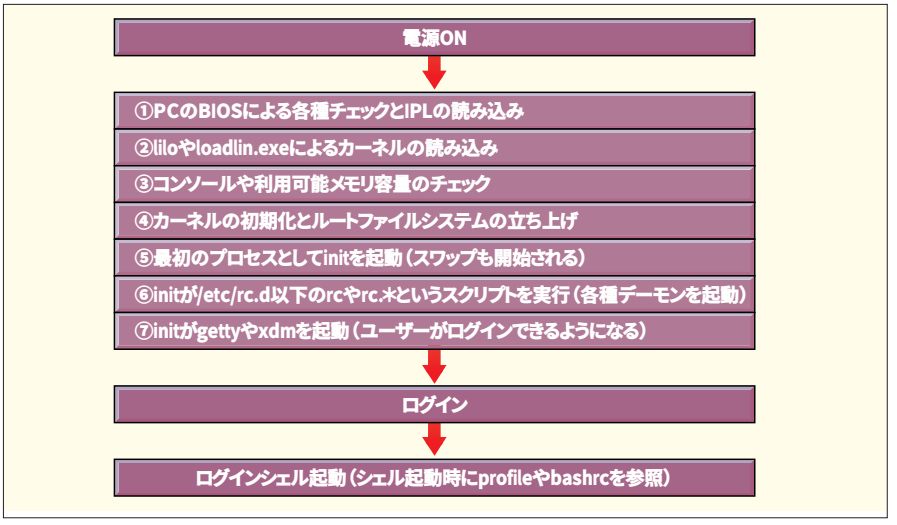


図1　電源ONからログインまでの流れ

が、Red Hat系ディストリビューションでは、3がテキストベースのマルチユーザーモード、5がX Window Systemベースのマルチユーザーモードとなっている。

inittabの内容

リスト1を例に、inittabの内容を調べていこう。2行めのidで始まる行では、actionフィールドにinitdefaultが指定されている。これはデフォルトのランレベルを指定するものであり、Red Hat系ディストリビューションでは、通常3または5を指定する。両者の違いは、「login:」のようなログインプロンプトがテキスト画面に表示されている（3の場合）か、またはX Window Systemのディスプレイマネージャ（xdmなど）によるグラフィカルなログイン画面が表示される（5の場合）かである。

5行めのsiで始まる行では、processフィールドに/etc/rc.d/rc.sysinit、actionフィールドにsysinitが指定されている。sysinitが指定されている行は、起動処理の最初に実行されるので、rc.sysinitというシェルスクリプトは、init起動後、最初に実行される設定ファイルということになる。

ランレベルごとの設定

7～13行めでは、0～6のランレベルに合わせて個別の処理が行われる。実際には、各ランレベルの数字を引数にして、/etc/rcというコマンドが実行されている。またactionフィールドにwaitが指定されているため、rcコマンドの処理が終了するまで、次の処理は開始されない。

このrcというコマンドは、シェルスクリプトであり、受け取った引数（ランレベル）をもとに、デーモンプロセスの停止／起動を行っている。

リスト2は、Red Hat Linux 7.1のrcファイルの主要部分を抜粋したものだ。ここで実行されているのは、次のとおりだ。

(1) /etc/rc?.d/ディレクトリを探す
(2) Kで始まるスクリプトを実行
(3) Sで始まるスクリプトを実行

最初に、ランレベルに対応したディレクトリ/etc/rc.?d/の存在を確認している。たとえばランレベル3の場合には、/etc/rc3.d/があるかどうかをチェックする。

前のセクションでも説明したように、最新のFHSでは、/etc/ディレクトリの直下にrc?.d/を配置することになっている。しかし、Red Hat 6.x系のディストリビューションでは、/etc/rc.d/以下にrc?.d/ディレクトリが置かれていた。

移行期間である現在は、以前と同じように/etc/rc.d/の下にrc?.d/があり、

▼リスト1　Red Hat Linux 7.1の/etc/inittab（抜粋）

```
 1: # Default runlevel.
 2: id:3:initdefault:                    ← デフォルトのrunレベル。通常は3または5
 3:
 4: # System initialization.
 5: si::sysinit:/etc/rc.d/rc.sysinit     ← システム初期化スクリプト。全runレベル共通
 6:
 7: l0:0:wait:/etc/rc.d/rc 0
 8: l1:1:wait:/etc/rc.d/rc 1
 9: l2:2:wait:/etc/rc.d/rc 2
10: l3:3:wait:/etc/rc.d/rc 3             ← runレベル0～6の処理。スクリプトrcにrunレベルの数値を渡している
11: l4:4:wait:/etc/rc.d/rc 4
12: l5:5:wait:/etc/rc.d/rc 5
13: l6:6:wait:/etc/rc.d/rc 6
14:
15: # Things to run in every runlevel.
16: ud::once:/sbin/update                ← 全runレベルで共通
17:                                      ← 電源遮断を感知し、シャットダウンを行う
18: # Trap CTRL-ALT-DELETE
19: ca::ctrlaltdel:/sbin/shutdown -t3 -r now   ← Ctrl＋Alt＋Del入力をトラップし、再起動されるようにしている
20:
21: pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
22:
23: pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
24:                                      ← シャットダウン動作のキャンセル
25: # Run gettys in standard runlevels
26: 1:2345:respawn:/sbin/mingetty tty1
27: 2:2345:respawn:/sbin/mingetty tty2
28: 3:2345:respawn:/sbin/mingetty tty3   ← runレベル2～5でmingettyを6つ起動
29: 4:2345:respawn:/sbin/mingetty tty4
30: 5:2345:respawn:/sbin/mingetty tty5
31: 6:2345:respawn:/sbin/mingetty tty6
32:
33: # Run xdm in runlevel 5
34: # xdm is now a separate service
35: x:5:respawn:/etc/X11/prefdm -nodaemon   ← runレベル5でグラフィカルなログイン画面を表示
```

それらのシンボリックリンクが/etc/の直下に置かれている。逆にKondara MNU/Linux 2.0では、FHSに準拠したディレクトリ配置をとり、/etc/rc.d/以下に/etc/rc?.d/のシンボリックリンクが存在している。本稿では、FHSに準拠したディレクトリ表記を採用している。

さらに**リスト2**の内容を追うと、目的のディレクトリがあったら、その下に配置されているファイル名がKで始まるスクリプト（4～11行）、およびSで始まるスクリプト（13～19行）を実行していることがわかる。Red Hat 6.x系のrcファイルよりも、多少込み入っているが、これはエラーが起きた際のログを記録する仕組みが付け加えられているためだ。

また、Kで始まるファイルにはstopという引数が、Sで始まるファイルにはstartという引数が付け加えられていることがわかるだろう。つまり、まずKで始まるファイルを停止（Kill）させ、続いてSで始まるファイルを起動（Start）させているのだ。

ファイル名で起動順を制御

これらのシェルスクリプトは、どちらも/etc/init.d/ディレクトリ以下にあるスクリプトのシンボリックリンクであり、引数のstart／stopによって、該当するデーモンの起動／停止を行っている。

rc?.d/ディレクトリのリストを見ればわかるように、これらのリンクは、KまたはSに続けて2桁の数字が付けられている。この数字は飾りではなく、デーモンを起動／停止する順番を制御するためのものなのだ。

リスト2の13～19行は、対象ディレクトリ内のSで始まる各ファイルに対して、14～18行の処理（startという引数を付けて実行する）を行うことを示している。「S＊」という表現を「Sで始まる各ファイル」に展開するのは、シェルの役目だ。シェルは＊のようなワイルドカードを展開する際には、ASCIIコードの順に行うため、S／Kに続けて番号をふっておくことで、ユーザーの希望どおりの順にシェルスクリプトを実行することが可能になる。

init起動後に読み出される設定ファイルを、以下にまとめておく。

▼リスト2　Red Hat Linux 7.1の/etc/rc.d/rc（抜粋）

```
 1: # Is there an rc directory for this new runlevel?
 2: if [ -d /etc/rc$runlevel.d ]; then
 3:     # First, run the KILL scripts.
 4:     for i in /etc/rc$runlevel.d/K*; do
  :         :
  :       (中略)
  :         :
 5:         # Bring the subsystem down.
 6:         if egrep -q "(killproc |action )" $i ; then
 7:             $i stop
 8:         else
 9:             action $"Stopping $subsys: " $i stop
10:         fi
11:     done
  :
12:     # Now run the START scripts.
13:     for i in /etc/rc$runlevel.d/S*; do
  :         :
  :       (中略)
  :         :
14:         # Bring the subsystem up.
15:         if egrep -q "(daemon |action )" $i ; then
16:             $i start
17:         else
  :         :
  :       (中略)
  :         :
18:         fi
19:     done
20: fi
```

変数$runlevelには、ランレベルが代入されている。-dで、ディレクトリの有無を確認している

K＊を展開し、マッチするファイル名をASCIIコード順に変数iに代入する

変数iのファイルを、stopオプションを付けて実行する

S＊を展開し、マッチするファイル名をASCIIコード順に変数iに代入する

変数iのファイルを、startオプションを付けて実行する

rc.sysinit

rc.sysinitは、init起動後、最初に実行されるシェルスクリプトである。ネットワークの起動、キーマップ、コンソールのフォントの設定、スワップパーティションの準備、ファイルシステムのチェックなどシステム全般にわたる初期化作業を行っている。

sysconfig/

システムの起動時には、ハード／ソフト両面にわたって、非常に多くの設定が必要になる。それらを1つのファイルに記述してしまうのは、メンテナンスの点からあまり好ましいことではない。

そこで、コンポーネントごとの設定項目や、起動のためのスクリプトは、個別のファイルに分けられている。それらのファイルはsysconfig/ディレクトリ以下に配置されている。

sysconfig/ディレクトリ内のスクリプトには、上述のrc.sysinitスクリプトから呼ばれるものが多く含まれている。

またそのほかに、init.d/ディレクトリ以下で起動されるデーモンプロセスが参照するための、ハードウェアの情報や、オプションが記述されているファイルも、sysconfig/ディレクトリ以下に存在している。

rc

ランレベルに応じて、デーモンプロセスを起動／停止させるシェルスクリプト。最新のFHSでは、/etc/ディレクトリ直下に置かれることになった。

rc.d/

rcが呼び出すデーモンプロセスの起動スクリプトを格納するディレクトリは、ランレベルに応じて、rc0.d/～rc6.d/まであり、それらは/etc/rc.d/ディレクトリ下に配置されていた。最新のFHSでは、rc0.d/～rc6.d/はいずれも/etc直下に置かれることになったので、現在は旧バージョンとの互換性のために置かれているディレクトリだ。

init.d/

各種デーモンプロセスをコントロールするスクリプトの格納場所。これらのスクリプトは、/etc/rc?.d/ディレクトリ以下にあるシンボリックの実体である。start／stopを引数にとり、デーモンプロセスの起動／停止を行う。そのほか、status／restartなどの引数を指定できるものもある。statusは、当該デーモンの状態を表示、restartは停止後に再起動を行う。

rc.local

rc?.d/ディレクトリ内にS99localという名前のシンボリックリンクが作られており、ランレベルに応じてさまざまなデーモンが起動されていった、その最後に実行されるスクリプトだ。そのマシンに特有の設定やドライバが必要な場合は、このファイルに記述するといいだろう。

ハードウェアに関する設定ファイル

システム起動時に参照されるファイルのうち、特にハードウェアに関連しているものとしては、以下の2点があげられる。

fstab

fstabは、ファイルシステムを構成するパーティションや、CD-ROMなどのリムーバブルメディアなどのマウントポイントを記述するためのファイルである。

fstabの1項目は、スペースで区切られた6つのフィールドからなる（**リスト3**）。1番めはデバイス名を指定する。NFSやSMBファイルシステムなどを用いて、リモートマシンのパーティションをマウントする際には、ホスト名も指定する。2番めはマウントポイント、3番めはファイルシステムの種類、そして4番めには、ファイルシステム固有のオプションを記述する。5、6番めはそれぞれdump／fsckコマンドの対象にするかどうかを指定するものだ。dumpについては、ハードディスク上のパーティションを1、それ以外は0にしておけば、ハードディスクだけがdumpの対象になる。fsckについては、1以上の番号を付けると、その順にfsckによるチェックを受けるが、現在のfsckは並列実行が可能なため、ルートパーティションに1、そのほかのハードディスク上のパーティションに2をふっておけばいいだろう。

modules.conf

Linuxのデバイスやファイルシステムのドライバモジュールの多くは、必要に応じて、カーネルに組み込むことができるローダブルモジュールだ。この機能のおかげで、カーネルが不必要なドライバを抱え込んで巨大化することを防止できる。

modules.confは、ローダブルモジュールに別名（alias）を付けて、起動時に自動的に組み込みために用いられる（**リスト4**）。カーネル2.2ではTurbolinux以外のディストリビューションでconf.modulesというファイル名が用いられていたが、カーネル2.4でmodules.confに統一された。

▼リスト3　fstab

```
1: /dev/hda7     /             ext2      defaults         1 1
2: /dev/fd0      /mnt/floppy   auto      noauto,owner     0 0
3: none          /proc         proc      defaults         0 0
4: none          /dev/pts      devpts    gid=5,mode=620   0 0
5: /dev/hda5     swap          swap      defaults         0 0
6: /dev/cdrom    /mnt/cdrom    iso9660   noauto,owner,ro  0 0
```

▼リスト4　modules.conf

```
1: alias eth0 eepro100
2: alias scsi_hostadapter aic7xxx
3: alias usb-controller usb-uhci
```

シェル

システムコンソールやターミナルからログインするとプロンプトが表示され、bashやcsh、zshなどのシェルが起動される。そのシェルに関係する設定ファイルを見てみよう。

bashrc

Linuxの標準シェルであるbashの起動時に、全ユーザー共通の設定項目を記すためのファイル。bashが自動的に読み込むわけではなく、各ユーザーのホームディレクトリにある、.bashrcの先頭部分でこのファイルを実行している。

csh.cshrc
csh.login

csh系のシェルを使用する際に用いる設定ファイルのテンプレート。

cshは、起動されるごとに~/.cshrcを実行し、ログイン時にはさらに~/.loginを実行するようになっている。そこでcsh.cshrc、csh.loginをそれぞれ~/.cshrc、~/.loginにリネーム・コピーするか、~/.cshrc、~/.loginの先頭で実行するようにすれば、全ユーザー共通の環境を設定できる。

DIR_COLORS

ファイル一覧を表示するlsコマンドの出力をカラー化するための設定ファイルだ。ディストリビューションやターミナルによっては、最初からディレクトリが青、リンクが水色などで表示されているはずだ。

DIR_COLORSの内容は大きく分けて、(1) カラー化できるターミナルの指定、(2) ファイルの種類による色の設定、(3) ファイルの拡張子による色の設定の3セクションからなる。(2)、(3) の部分を書き換えることで、lsの表示を好みの配色に変更可能だ。

リスト1に示すように、ファイルの種類と表示色や属性（太字など）を並べて指定するようになっている。たとえば通常のファイルの表示を、黄色い文字と黒い背景にしたければ、

```
FILE 33:40
```

のように指定する。

“ls --color”とすれば、好みの表示色でリストが表示されるはずだ。Red Hatなどでは、シェル起動時に読み込まれるファイル（~/.bashrcなど）に、lsのaliasが定義されているのでカラーで表示される。

issue
issue.net
motd

ターミナル画面でログインするときに表示される、OSやカーネルのバージョンメッセージが記録されているファイル。

コンソールにはissueの内容が、telnetで接続したときはissue.netの内容が、「login:」プロンプトの前に表示される。通常は、Linuxの起動時に実行されるrc.localのなかで、ディストリビューション名、カーネルバージョンなどを元に作られる。

Turbolinuxでtelnetから接続したときに「login:」だけでissue.netが表示されないのは、xinetd.d/telnetのなかで、telnetdのオプション-hによって表示を抑制しているためである。

motdは、“Message Of The Day”の略語で、ログインした直後に表示されるメッセージファイルだ。デフォル

▼リスト1　DIR_COLORSディレクトリは青、リンクは水色がデフォルト

```
 44: # Attribute codes:
 45: # 00=none 01=bold 04=underscore 05=blink 07=reverse 08=concealed
 46: # Text color codes:
 47: # 30=black 31=red 32=green 33=yellow 34=blue 35=magenta 36=cyan
37=white
 48: # Background color codes:
 49: # 40=black 41=red 42=green 43=yellow 44=blue 45=magenta 46=cyan
47=white
 50: NORMAL 00       # global default, although everything should be
something.
 51: FILE 00         # normal file
 52: DIR 01;34       # directory
 53: LINK 01;36      # symbolic link
 54: FIFO 40;33      # pipe
 55: SOCK 01;35      # socket
 56: BLK 40;33;01    # block device driver
 57: CHR 40;33;01    # character device driver
```

トでは、Red Hat系は空のファイル、Turbolinuxは「Welcome to Turbolinux.」が記録されている。複数のユーザーが利用するシステムでは、メンテナンスの予定を書き込んでおいて、利用者に告知するために使用するのがよいだろう。

kon.cfg

日本語コンソールエミュレータkonの設定ファイルだ。ディスプレイの表示範囲やフォント、文字コードの設定などが書かれている。

profile
profile.d/

profileは、ログイン時に実行するファイルで、全ユーザー共通の設定ファイルである。profile.dディレクトリには、各種のパッケージ用の初期設定スクリプトがあり、bashの場合、profileのなかでprofile.d/＊.shファイルを実行している。csh系の場合には、profile.d/＊.cshが使われる。

リスト2の28行めでは、各種の環境変数を設定しており、30行めからはprofile.d/以下のスクリプトを実行している。

securetty

rootアカウントでログインできるターミナルを指定するファイル。RedHatでは、vc/1～vc/11とtty1～tty11が記録されている。telnetで接続する場合には、pts/0、pts/1……というターミナル名が利用されるため、ローカルのコンソール以外ではrootユーザーでログインできないように制限されている。

ネットワーク経由でrootユーザーになるには、一般ユーザーでログインしたあとで、suコマンドを使用する。その場合でも、セキュリティを考慮するとtelnetではなく、暗号化シェルsshを利用したほうがよいだろう。

shells

ログインシェルとして指定可能なシェルのフルパスが記されている。chshコマンドでログインシェルを変更するときに参照される。**画面1**のようにすると、ユーザーのログインシェルがbash2（bashバージョン2）に変更される。

ssh/

フリーのネットワーク接続ツールSecure Shellプロトコルサーバ OpenSSHの設定ファイルのためのディレクトリ。OpenSSHには、telnetやrlogin、ftpの代わりに使用するコマンドが含まれており、パスワードを含むすべての通信を暗号化する。

OpenSSHは、ローカル側のプログラムとリモートホスト側のsshdデーモンとで通信を行う。このディレクトリには、ローカルプログラム用の共通設定ファイルssh_configや、リモートホスト側のsshdの設定ファイルsshd_config、リモートホスト側の秘密鍵ssh_host_key、公開鍵ssh_host_key.pubなどが置かれている。

sudoers

sudoersは、別のユーザーとしてコマンドを実行することができるsudoプログラムが参照するファイルで、どのユーザーが誰の権限で何を実行できるかのリストを記録しておく。

ダイヤルアップ接続に使うpppdや、デバイスファイルを直接アクセスするプログラムなどで、rootや特定のユーザーアカウントで実行する必要がある場合には、sudoコマンドを使えば、一般ユーザーのアカウントでは実行できないプログラムを、利用可能にすることができる。

このファイルの修正はrootユーザーで、viエディタと同じ操作方法のvisudoコマンドで編集する必要がある。visudoを使うことで編集中にsudoersファイルがロックされ、変更後に構文チェックが行われる。

sudoを使えばすべてをrootユーザーになって行うよりも安全であるが、そのなかからシェルや別のプログラムを起動できるエディタなどに適用すると、root権限で何でも行えるようになってしまうため、sudoersの設定は気をつける必要がある。

▼リスト2　profile全ユーザーに共通の項目を設定する

```
28: export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
29:
30: for i in /etc/profile.d/*.sh ; do
31:         if [ -x $i ]; then
32:                 . $i
33:         fi
34: done
```

```
$ chsh
user1 のシェルを変更します。
Password:
新しいシェル [/bin/bash]:
/bin/bash2
シェルを変更しました。
```

画面1　ログインシェルをbash2に変更する

システム設定

/etcディレクトリ以下のファイルは、すべて「システム設定」用とも言ってもいいのだが、このセクションでは特にLinuxが起動したあとに必要になるファイルを中心に取り上げる。

画面表示関連

screenrc

VT100互換の端末エミュレータ機能を提供する画面管理プログラムscreenの設定ファイルのサンプルである。キャラクタ表示のみ可能な端末でも、同時に複数の端末エミュレータを利用可能にするので、複数のホストにログインする場合などに便利だ。しかし、日本語表示に対応していないため、国内ではあまり使われていない。X Window Systemを利用できるPCが手軽に入手できる現状では、screenコマンドの出番はほとんどないだろう。

X11/

X Window System（XFree86）に関連する設定ファイルが置かれている。

X Window System本体の起動に関する設定ファイルだけでなく、アプリケーションの設定ファイル、ウィンドウマネージャやディスプレイマネージャの設定ファイルなども含まれている。

▼リスト1　/etc/adjtimeファイルの内容

```
3.718193 1001254501 0.000000
1001254501
LOCAL
```

ハードウェアに関するファイル

adjtime

マザーボード上にある、ハードウェアクロックの取得と設定を行うhwclockコマンドが利用するファイルだ。

ハードウェアクロックは不正確なので、adjtimeファイルは少しでも誤差を減らすために用いられる。内容は、**リスト1**に示した3行である。1行めには、以下に示す3つのパラメータがスペース区切りで記されている。

- **1日あたりの時刻のズレ（浮動小数点実数）**
- **前回時計を合わせた時刻（1970年1月1日0時0分0秒 UTCからの経過秒数）**
- **0.000000（ゼロ）**

2行めは1行めの2個めのパラメータと同じ値が、3行目はハードウェアクロックがUTC（Universal Time Coordinated：世界協定時）かローカルタイムかを示す文字列が格納される。これらの値は、hwclockコマンドが設定するので、ユーザーが変更したりする必要はない。

余談だが、上記の方式で表した時間は、つい最近10億（10桁）を超えたばかりである。このデータを格納する領域を9桁分しか持たないプログラムには、いわゆる「2001年9月9日問題」が起きると言われていた。幸いLinuxは無事にこの問題を乗り切ることができたようだ。

せっかく用意されているadjtimeだが、インターネットが利用できる環境ならば、NTP（Network Time Protocol）を用いたタイムサーバから正確な時間を取得するほうがいいだろう。

conf.linuxconf

コマンドライン、GUI、Webブラウザなど多彩なインターフェイスを持った設定ツールであるlinuxconfが参照する設定ファイル。前回起動時に使われたロケール名などが記されているが、特に手動で設定する必要はない。

fb.modes

フレームバッファデバイスの設定を行う、fbsetコマンドが参照するデータベースファイルだ。各解像度における周波数やタイミングなどのデータがある。

フレームバッファデバイスは、X Window Systemが利用するものではないので、fbsetコマンドもあまり使われる機会はないだろう。

fdprm

フロッピーディスクのパラメータが定義されているファイル。ディスクドライブのパラメータを変更する、setfdprmコマンドが参照する。fdprm自体をユーザーが変更する必要はない。

filesystems

mountコマンドで「-t」オプションでファイルシステムの種別を何も指定しなかったり、「auto」を指定しておくと、mountはファイルシステムを自動認識しようと試みる。それがうまくいかなかった場合に、このfilesystemsに列挙されている形式でマウントを試みる。

filesystemsの内容は、ファイルシステム名が並べられているだけ（**リスト2**）で、先頭に「nodev」が付けられているファイルシステムは除外される。一般的に利用されるファイルシステムは、たいてい自動認識されるため、管理者がこのファイルに追加する必要はほとんどない。

gpm-root.conf

コンソール画面上で用いるマウスハンドラ、gpm-rootの設定ファイルである。gpm-rootコマンドを常駐させたのち、コンソール画面上でのマウスをクリックすると、gpm-root.confに記述したメニューが表示される。

isapnp.gone

プラグ&プレイに対応したISA拡張カードの設定ツールであるisapnptoolsで使用されるファイルだ、プラグ&プレイに対応していないデバイスが用いている、I/Oアドレス、IRQ、DMAなどのリソースを記述しておくと、isapnptoolsがリソースを割り当てる際に、これらの値を避けてくれる。

デフォルトでは、いくつかのサンプルがコメントアウトされた状態で記述されている。古いISA拡張カードを用いる際には、これを参考に書けばいいだろう。

pcmcia/

PCカード用の設定ファイルが置かれている。

configというファイルに、各PCカードのクラスと必要なローダブルモジュール（デバイスドライバ）が記されている。

そのほか、モデムやSCSIカードなどのカテゴリごとに設定ファイルと、パラメータファイル（拡張子opts）が用意されている。

hotplug/

Linuxはカーネル2.4以降、USBのサポートが進んでいる。USBの特徴のひとつである、ホットプラグ機能をLinuxで実現するための「hotplug」パッケージの設定ファイルが置かれているディレクトリだ。/sbinディレクトリに置かれるhotplugという名のシェルスクリプトによって読み込まれる。

なおカーネル2.4では、カードバスのホットプラグにも対応している。

makedev.d/

/dev以下にはデバイスファイルと呼ばれる、ハードウェアにアクセスするためのファイルが置かれている。デバイスファイルは非常に数が多く、Red Hat Linux 7.1の場合、5000個以上もある。

これらのデバイスファイルを効率的に作成／管理するために、MAKEDEVというコマンドが用意されており、makdev.dディレクトリには、MAKEDEVが利用する設定ファイルが置かれている。

devfsd.conf

Linuxカーネル2.4で新たに導入された機能のひとつに、Devfs（Device File System）がある。Devfsは、

- **デバイスファイルの動的生成**
- **階層化可能**

といった今までのデバイスファイルではできなかった機能を持ち、柔軟かつ効率的な管理が行える。

devfsd.confは、Devfsの機能を提供するデーモン、devfsdの設定ファイルだ。しかし、カーネル2.4を採用しているRed Hat系ディストリビューションも、デフォルトではdevfsdを使っていないので、devfsd.confを編集したところで、システムには何の影響もない。Devfsのもたらす変更は大規模なので、互換性の点からすぐに採用には踏み切れないからと思われる。

lpd.conf lpd.perms

以前から使われてきたLPR印刷システムに代わる、LPRngプリンタスプーラシステムの設定ファイルだ。

今までのlpr、lpq、lprmコマンドなどと同じインターフェイスを持つうえに、以下の特徴を備える。

▼リスト2　/etc/filesystemsファイルの内容

```
1: ext2
2: nodev proc
3: nodev devpts
4: iso9660
5: vfat
6: hfs
```

▼リスト3　/etc/default/useraddファイルの内容

```
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

- **動作が軽い**
- **複数のプリンタを1つのキューで利用**
- **より厳密なセキュリティチェック**
- **改良された認証システム**

lpd.conf／lpd.permsともに、初期状態では、多くの設定がコメントの形で無効化されている。これらを見ればわかるように、非常に柔軟な設定が行える。

ユーザー／グループ関連の設定

UNIX系OSでは、ユーザー／グループに関する設定ファイルもテキスト形式になっている。セキュリティ上の問題が起きないように、いろいろと工夫がされているが、操作する際には十分に注意しよう。

useraddコマンドで、新たにユーザーアカウントを作成する際のデフォルト設定を記したuseraddファイル（**リスト3**）が置かれている。

「GROUP」は、ユーザーが標準で属するグループ（デフォルトグループ）を指定する。ただし、Red Hat Linuxのuseraddコマンドは、ユーザーIDとグループIDが同じになるように変更されている。そのため、ここで指定されている値は、useraddコマンドに「-n」オプションを付けて、この機能を抑制した時だけ有効になる。

「HOME」は、ユーザーのホームディレクトリを作成するディレクトリを指定する。

4行めの「EXPIRE」は、パスワードの有効期限を、日単位で指定する。また3行めの「INACTIVE」は、パスワードの有効期限が過ぎたのち、当該アカウントを無効にするまでの期限を同じく日単位で指定する。「-1」を指定すると、アカウントは無効にされることはない。また「0」だと、有効期限が切れると同時にアカウントが無効になる。

パスワードの有効期限と、期限が切れたのちのアカウントの無効化（ロック）は、後述するshadowファイルに書き込まれる。

「SHELL」は、ユーザーが標準で使用するシェルをフルパスで指定する。管理者の好みであまりメジャーでないシェルをデフォルトにすると、ユーザーが困るので、気をつけよう。

「SKEL」には、ユーザーのホームディレクトリに置かれる個人用の設定ファイルのひな形を格納しているディレクトリを指定する。デフォルトでは、/etc/skelだ。

ユーザーアカウントを作成すると、/home以下にホームディレクトリが作成され、シェルやアプリケーションの設定ファイル（ドットファイルであることが多い）のひな型がコピーされる。skelディレクトリには、これらの設定ファイルが置かれている。

これらのファイルには、各ユーザーのアカウント情報が含まれている。ファイル名から想像すると、passwdファイルに、ユーザーのパスワードが保存されているように思えるが、そうではない。初期のUNIXでは、暗号化されたパスワードがpasswdファイルに含まれていたが、誰でも読めるpasswdファイルにパスワード（一応暗号化されてはいるが）を含むのは、セキュリティ上問題があるため、今ではパスワードはshadowファイルに含まれている（シャドウパスワード）。shadowファイルは、管理者以外からは読めないようにしておく。

passwdファイル内では、1つのアカウントに関する情報が、1行にまとめられている。各行は、「：」で区切られた7つの項目からなる。項目の内容は、先頭から順に、ユーザー名、パスワード、ユーザーID、グループID、フルネーム（またはコメント）、ホームディレクトリ、標準シェルだ（**リスト4**）。前述のように、パスワードはshadowファイルに移されているため、パスワードの項目には、「x」が入っている。

shadowファイルには、「：」で区切られた9つの項目を持った行が、アカウントの数だけ含まれている。各項目の内容は、**表1**のとおりだ。

passwdもshadowもテキスト形式ではあるが、エディタで直接書き換える

カラム	内容
1	ログイン名
2	暗号化されたパスワード
3	最後にパスワードを変更した日
4	パスワードが変更可能となるまでの日数
5	パスワード変更期限までの日数
6	パスワードの有効期限が迫っていることを警告する日数
7	パスワードの有効期限切れからアカウントがロックされるまでの日数
8	アカウントが使用不可になるまでの日数
9	予約

表1　/etc/shadowファイルの書式

▼リスト4　/etc/passwdファイルの見本

```
giko-f:x:512:512:Giko Fussar:/home/giko-f:/bin/bash
```

ことはなるべく避け、passwdコマンドを利用するようにしよう。どうしてもエディタを使う場合は、passwd専用のviであるvipwを用いる。vipwは、たとえ編集中に停電などの事故が起きても、passwd／shadowを失うことのないように、工夫されている。

passwd-、shadow-という似た名前のファイルは、passwdやshadowのバックアップファイルである。

group gshadow

groupファイルには、各グループにどのユーザーが所属しているのかが記述してある。各行が1つのグループを表しており、「：」で区切られた以下の4つのフィールドが含まれている。

グループ名:パスワード:GID:リスト

「リスト」は、グループに所属しているユーザーのリストである。複数のユーザー名は「,」（カンマ）で区切られている。「GID」は、グループIDで、Red Hat系ディストリビューションでは、デフォルトでユーザーIDと同じ値になる。

パスワードは、newgrpコマンドでグループIDを変更する際に必要になる。このフィールドが空の場合は、パスワードが設定されていないことを表す。

前述のpasswd／shadowと同じく、groupファイルにもシャドウパスワードが用いられており、パスワードフィールドには、「x」が入っている。そしてグループパスワードの実体は、gshadowファイルに含まれている。gshadowファイルは、「：」で区切られた以下の4つのフィールドが含まれている。

グループ名:パスワード:管理者:リスト

group／gshadowもテキスト形式ではあるが、直接エディタで編集せず、変更する際にはgroupmodコマンドなどを用いるべきだ。

予想はつくと思うが、group-、gshadow-は、group、gshadowのバックアップである。

pam.d/ security/

pam.d、securityディレクトリには、PAM（ Pluggable Authentication Modules：差し替え可能な認証モジュール）に関連した設定ファイルが置かれている。Linuxディストリビューションの多くは、ユーザー認証やサービスへのアクセス認証にPAMを利用している。PAMはLinux以外にも多くのUNIX系OSで用いられている。

PAMを利用することで、個々の認証方法の差異を意識することなく、安全な認証手続きを行える。また、既存のプログラムを再コンパイルすることなく、認証手続きを付加することが可能になる。

PAMは高機能ではあるが、不用意に用いるとセキュリティホールの原因になってしまったり、ログインができなくなってしまう可能性もあるので、使用時には注意が必要だ。

pwdb.conf

アプリケーションからユーザー情報を簡単に取得できるための、ユーザーパスワードデータベースライブラリ（libpwdb）の設定ファイルである。libpwdbは、PAMと協調して働くように作られている。基本的に、pwdb.confをユーザーの手で変更する必要はない。

そのほか

ld.so.conf

Linuxのプログラムは、その一部を実行時にライブラリからリンク（ダイナミックリンク）することができる。このようにしておくことで、プログラムのサイズを小さくできる。

ダイナミックリンクされるライブラリは、共有ライブラリと呼ばれる。ld.so.confは、共有ライブラリの存在場所を指定するための設定ファイルだ。

Red Hat系ディストリビューションにtarボールから新しいプログラムをインストールした際には、ld.so.confを手動で書き換えなければならないことがある。

ld.so.confを変更した場合には、コマンドラインからldconfigコマンドを実行する必要がある。ldconfigコマンドは、ld.so.confの中身と、歴史的に必ず共有ライブラリが置かれている/usr/lib、/libディレクトリを検索して、ダイナミックリンク用のキャッシュ(ld.so.cache)を再構成する。ld.so.cacheは、バイナリファイルであり、直接編集することはできない。

rpc

リモートプロシージャで使用される、プログラムとサーバの名前／番号の対応表が格納されている。

リモートプロシージャは、ネットワーク上の別のホストにアクセスするためのプロトコルで、分散コンピューティング環境で利用される。

デーモン

Linux（UNIX）では、多数のデーモン（サーバ）プロセスが動作している。これらは、クライアント／サーバ形式で動作し、さまざまな機能を実現している。ここではこれらのデーモンのうち、ネットワークサービス以外のデーモンに絞って解説することにする。

amd.conf amd.net

amd.confは、ファイルシステムにアクセスしようとしたときに、ファイルシステムが自動的にマウントされる機能を実現するamdデーモンの設定ファイルだ。

amd.netには、ファイルシステムをどのようにマウントするかを記述したマップファイルを定義しておく。

anacrontab

anacronは、定期的なコマンドを実行するためのデーモンだ。同様の機能に、後述するcronデーモンがあるが、anacronデーモンは連続稼働していないシステム向けのものだ。つまり、使用していないときに電源を切るようなマシン向けに用意されたcronデーモンといえるだろう。

anacrontabは、このanacronデーモンが実行するコマンドを指定するファイルだ。書式は、1行ごとに繰り返しの周期（日単位）、実行時の遅延時間（分単位）、名前、実行するコマンドの4つをスペースで区切って書く。anacrontabのデフォルト（**リスト1**）では、繰り返し周期が1日、1週間、1カ月のコマンドを繰り返し実行するように設定されている。各行で指定しているrun-partsという記述は、指定したディレクトリ以下のコマンドをすべて実行するための指定だ。

連続稼働が前提でないanacronコマンドは、前回に起動した日時を記録し、今回起動時の時間との差を取った結果とanacrontabに記されている繰り返しの周期と比較する。そして、繰り返しの周期のほうが短い場合は、その行に書かれた遅延時間が過ぎるとコマンドを実行する。たとえば、前回anacrontabを起動したのが3日前だとすると、**リスト1**の9行めの周期（1日）のほうが短いので、5分後にcron.dailyディレクトリにあるコマンドが実行される。

auto.master auto.misc

前述したamdデーモンと同様、ファイルシステムを自動的にマウントするautomountデーモンの設定ファイルだ。

auto.masterは、マウントポイントとなるディレクトリと、マップファイルの名前（auto.misc）を記述する。さらに、ファイルシステムを一定時間以上使用していない場合に、自動的にアンマウントするように指定することもできる。デフォルトでは**リスト2**のように指定されている。これは、automountでマウントするファイルシステムは、/misc以下にマウントされ、マウントされるファイルシステムのマップファイルは/etc/auto.miscに記載されているという意味だ。さらに、60秒間アクセスされなければ、自動的にアンマウントするという指定になっている。

マップファイルauto.miscでは、自動的にマウントするファイルシステムを定義する。書式は、ディレクトリ名、ファイルシステムタイプ（マウントタイプ）、デバイスファイルの順で指定する。

リスト3の7行目では、デバイスファイル/dev/cdromを、ISO9660形式の読み取り専用でcdディレクトリにマウ

▼リスト1　anacrontabファイル

```
  5:SHELL=/bin/sh
  6:PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
  7:
  8:# These entries are useful for a Red Hat Linux system.
  9:1       5       cron.daily      run-parts /etc/cron.daily
 10:7       10      cron.weekly     run-parts /etc/cron.weekly
 11:30      15      cron.monthly    run-parts /etc/cron.monthly
```

▼リスト2　auto.masterファイル

```
  6: /misc    /etc/auto.misc   --timeout=60
```

▼リスト3　auto.misc

```
  7: cd              -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom
```

ントするように指定している。

このように設定しておくことで、マップファイルに指定してあるディレクトリを参照しようとすると、automountデーモンが自動的にマウントし、デバイスにアクセスできるようになる。一般ユーザーでもCD-ROMをマウントできるので非常に便利だ。

ネットワークで接続されたほかのUNIXマシンのディレクトリや、Windowsのパーティションや NFSも自動的にマウントさせることができる。

at.allow
at.deny

指定時間になると指定したコマンドを自動的に実行するのがatデーモンだ。at.allowとat.denyファイルは、このatデーモンを実行できるユーザーのパーミッションを指定するためのファイルだ。

なお、2つのファイルには優先順位があり、atデーモンは次のような順序でユーザーのパーミッションを判断する。

①at.allowがあれば、at.allowに記載されたユーザーだけが利用可能
②at.denyがあれば、at.denyに書かれているユーザーだけが利用不可
③どちらもなければ、一般ユーザーは使用不可

ユーザー名は1行ずつ列挙する。

なお、デフォルトでは空のat.denyファイルだけが存在し、at.allowファイルは存在しない。このため、すべてのユーザーがatデーモンを利用することができるようになっている。セキュリティに配慮するのであれば、at.allowファイルを作成し、許可されたユーザーのみが利用できるようにしておいたほうがいいだろう。

cron.＊/
crontab

atデーモンと似た機能にcronデーモンがある。atデーモンは指定した時刻にコマンドを一度だけ実行するのに対し、cronデーモンは定期的に繰り返しコマンドを実行する。定期的なバックアップや、ログの整理などに利用すると便利な機能だ。

cronデーモンが実行するコマンドは、crontabファイルで設定する。

crontabファイルの書式は、コマンドが実行される日時、実行する際のユーザー権限、コマンド名をスペースで区切って記述する。日時の指定は、分、時、日、月、曜日の順で指定する。曜日の指定は数値で行い、0が日曜日、1が月曜日……6が土曜日（7も日曜日）という順になっている。さらに、日時の各項目は、その項目を無視する「＊」や、増分値を指定する「／」、複数の項目を指定できる「,」が指定できる。なお、曜日と月の指定は、英語の先頭3文字（sun、julなど）で指定することもできる。

リスト4は次のような予定でコマンドを実行する書式の例だ。

▼リスト4　crontabファイルの書式

```
2 5 1,15 * * root /usr/local/fullbackup
2 0 * * 0 root /usr/local/backup
```

▼リスト5　crontabファイル

```
 1: SHELL=/bin/bash
 2: PATH=/sbin:/bin:/usr/sbin:/usr/bin
 3: MAILTO=root
 4: HOME=/
 5:
 6: # run-parts
 7: 01 * * * * root run-parts /etc/cron.hourly
 8: 02 4 * * * root run-parts /etc/cron.daily
 9: 22 4 * * 0 root run-parts /etc/cron.weekly
10: 42 4 1 * * root run-parts /etc/cron.monthly
```

・毎月1日、15日の午前5時2分にroot権限で/usr/local/fullbackupコマンドを実行
・毎週日曜日の午前0時2分にroot権限で/usr/local/backupコマンドを実行

cronデーモン関連として、以下のディレクトリも用意されている。

```
cron.d
cron.hourly      cron.daily
cron.weekly      cron.monthly
```

このうち、cron.d以外のディレクトリはそれぞれ、毎時、毎日、毎週、毎月実行されるコマンドを入れておくディレクトリだ。このディレクトリにコマンドを入れておけば、それぞれのタイミングで実行されることになる。各ディレクトリ内のコマンドの実行間隔は、前述したcrontabファイルに記載されている（**リスト5**）。

crond.dディレクトリは、crontabファイルと同じ書式のファイルを入れておくことで、crontabと同様の設定を行うことができる。こちらはcrontabファイルとは異なり、コマンドごとにファイルを分割して整理することができる。

ネットワーク

Linuxは非常に優れたネットワーク機能を持つOSだ。ここでは、ネットワーク関連の機能とサービスについて解説する。

問い合わせ関連の設定ファイル

ここではDNSなど、問い合わせに関するファイルを解説する。

hosts

DNS（Domain Name System）を運用していないネットワークや、比較的小規模なLANにおいて名前の解決を行うために使われるのがhostsファイルだ。DNSを運用している場合でも、ローカルなホストの名前の解決などに利用することもできる。

hostsファイルには、ホスト名とIPアドレスの対応を列挙しておく。いわば、ホスト名とIPアドレスの対応表だ（**リスト1**）。

このhostsファイルによる名前の解決は、DNSが導入されるまでは標準的な方法だった。

resolv.conf

DNSを使ったシステムにおいて、名前の解決を実現するリゾルバの設定を行うための設定ファイルがresolv.confだ。このファイルでは、ドメイン名を省略した短いホスト名を指定した際に補完されるドメイン名の設定や、上位のネームサーバの指定などを行う。

リスト2の1行目にあるdomainで始まる行は、自分のホストが所属しているドメイン名だ。続くsearchで始まる行は、名前解決の際に追加するローカルドメインのリストを記述する。多くの場合は、sales.ascii-linux.comといったサブドメインを指定する。ただし、searchに指定できるドメイン名は6個、256文字以下という制限がある。

最後のnameserverは上位のネームサーバ（DNS）のアドレス指定だ。ここに指定するアドレスはIPアドレスだ。

host.conf nsswitch.con

名前の解決をどのように行うかを指定するのがhosts.confファイルだ。たとえば、まずhostsファイルを参照し、hostsファイルで名前の解決ができない場合はDNSを参照するといった設定を行うことができる。**リスト3**では、「order hosts,bind」と記述されているが、これは最初にhostsファイルを参照し、続いてDNSの実装であるBINDを参照するという設定だ。次の「multi on」は、複数のIPアドレスが割り当てられているホストの場合、名前の問い合わせに対して複数のIPアドレスを返すという意味だ。

nsswitch.confはシステムデータベースとNSS（Name Service Switch）の設定ファイルで、Cライブラリがどのような順番で名前の解決を行うかを指定するためのファイルだ。

ここでは、hostsファイルやDNSのほか、NIS（Network Information Service）や、NIS+の順序なども指定することができる。

named.boot named.conf

これら2つのファイルはDNSサーバBINDの設定ファイルだ。named.bootは古いバージョンのBIND 4用の設定ファイルで、named.confはBIND 8以降の設定ファイルだ。BIND 4とBIND 8では設定ファイルの書式が異なるので個別のファイルとなっている。BIND 4用の設定ファイルを収録しているのは今回解説するディストリビューションの中ではTurbolinux 7のみだ。

dhcpc/ dhcpcd/

これらのディレクトリは、IPアドレスを動的に割り当てるDHCP（Dynamic Host Configuration Protocol）の

▼リスト1　hostsファイル

```
127.0.0.1   localhost.localdomain   localhost
192.168.0.100     lmsvr00
192.168.0.101     lmpc01
192.168.0.102     lmpc02
```

▼リスト2　resolv.confファイル

```
domain ascii-linux.com
search sales.ascii-linux.com
nameserver 192.168.0.1
nameserver 192.168.1.1
```

クライアントデーモンのひとつである、dhcpcdが使用するディレクトリだ。DHCPサーバから取得したネットワーク情報がこのディレクトリに保存される。しかし、多くのRed Hat系ディストリビューションでは、dhcpcdはインストールはされるものの、pumpという別のDHCPクライアントデーモンが使われるため、これらのディレクトリは空のままとなる。

また、Turbolinux 7ではdhclientという別のDHCPクライアントデーモンが採用されており、dhcpcdはインストールされない。このため、Turbolinuxではこれらのディレクトリ自体が作成されない。

rndc.conf

BIND 8までは、BINDの動作をコントロールしたり、起動させるためのユーティリティとしてndcが使われていた。しかし、BIND 9からはネットワーク経由で利用できるrndcコマンドに変更された。rndcコマンドはndcコマンドと違い設定ファイルが必要で、この設定ファイルがrndc.confとなる。

rndc.confは、named.confと同じ構造の設定ファイルだ。

nscd.conf

ネームサービスキャッシングデーモンnscdの設定ファイルがnscd.confだ。nscdは、NIS＋やDNSなどのネームサービスで参照した情報をキャッシュするデーモンで、ネットワークを介してのデータ参照を減らすことができる。

nscd.confでは、キャッシュする内容やキャッシュを保持する時間を指定することができる。

ldap/openldap/ldap.conf

これらのディレクトリは、ディレクトリサービスであるLDAP（Light weight Directory Access Protocol）に関連するファイルが納められている。

ldapディレクトリはTurbolinux 7に、openldapディレクトリは、Red Hat Linux 7.1、LASER5 Linux 7.1、Kondara MNU/Linux 2.0に存在するディレクトリだが、どちらもディレクトリ名が違うだけでOpenLDAP関連の同様のファイルが収録されている。

LDAPの設定自体は、このディレクトリにあるldap.confで行うが、これとは別に同名のldap.confというファイルが/etcディレクトリに存在する。こちらは認証モジュールであるPAM（Pluggable Authentication Modules）のLADP用の設定と、LDAPをNSSで利用するためのクライアントモジュール、nss_ldapの設定ファイルだ。このファイルでは、LDAPサーバの名前やポート番号、セキュリティ関連の設定を行う。同じ名称だが、機能が違うので注意が必要だ。

identd.conf

identificationプロトコルによるユーザー情報問い合わせに応えるidentdの設定ファイル。

セキュリティ関連の設定ファイル

ここでは、アクセス制限などセキュリティに関するファイルを解説する。

xinetd.conf xinetd.d/

Linux（UNIX）では、さまざまなネットワークサーバデーモンが動作している。しかし、使用頻度が低いサーバを常に稼働させておくことは、メモリなどのリソースを無駄に消費しているといえる。かといって、サーバを停止した状態ではサービス自体を提供することができない。

そこで考えられたのがインターネットスーパーサーバだ。インターネットスーパーサーバは、接続要求を監視し、サーバへの接続要求を受け付けると対応するデーモンを起動する。この仕組みを使うことで、デーモンを常に起動しておく必要がなくなるのだ（ただし、DNSやsendmailなど頻繁にアクセス要求が発生するサーバなどには向かない）。

以前までは、inetdというスーパーサーバが利用されていたが、最新のディストリビューションではinetdに代わり、xinetdが採用されるようになった。xinetdはinetdの機能に加え、アクセス制限などの機能が追加されている。このため、以前のようにアクセス制御を行うtcp_wrappersと組み合わせて使う

▼リスト3　host.confファイル

```
order hosts,bind
multi on
```

▼リスト4　xinetdによるアクセス制限（wu-ftpd）

```
 1:# default: on
                ……省略……
13:        nice        = 10
14:        disable     = no
15:        only_from   = ascii-linux.com 192.168.0.1/255
16:}
```

ascii-linux.comと、192.168.0.1～192.168.0.255までを許可する指定

が必要なくなった。

xinetdでは、xinetd自体の設定を行うxinetd.confファイルと、デーモンごとの設定ファイルが用意されている。デーモンごとの設定ファイルはxinetd.dディレクトリに置かれている。

xinetdによるアクセス制限はこれらのデーモンごとの設定ファイル内で、アクセスを許可するホストはonly_from、アクセスを拒否するホストはno_accessとして指定する。たとえば、FTPできるホストを自分のドメインとLAN内に限定するのであれば、FTPサーバの設定ファイル（wu-ftpd）に**リスト4**のように追加する。

このほか、サーバへ接続できる時間帯を制限することも可能だ。こちらは、access_timesに続けて接続可能な時間帯を指定する。たとえば、「access_time=9:00-17:00」とすることで午前9時から午後5時までのアクセスに限定することが可能だ。

これら2つのファイルは、スーパーサーバから起動されるデーモンに対するアクセス制限を行うtcp_wrappers（tcpd）の設定ファイルだ。しかし、前述したように最新のディストリビューションではスーパーサーバがinetdからxinetdに変更され、tcp_wrapperが必要なくなった。しかし、デーモンによってはこれら2つのファイルを直接参照しているものもあるので設定自体は必要となる場合もある。

hosts.allowは接続を許可するホストを、hosts.denyには拒否するホストをデーモンごとに記述する。hosts.denyには、

```
ALL : ALL
```

と記述し、接続を許可するサービス、ホストだけをhosts.allowに追加すると安全だ。たとえば、hosts.allowに、

```
ALL : 127.0.0.1, 192.168.0.
```

と記述しておけば、すべてのサービスはlocalhost（127.0.0.1）と、IPアドレスが192.168.0.0～192.168.0.255のホストだけが利用できるようになる。

▼リスト5　ftphostsファイル

```
allow hogehoge ascii-linux.com
deny  hogehoge attacker.coj.p
```

公開サーバ系の設定ファイル

ここでは、インターネットで一般的な公開サーバに関する設定ファイルを解説する。

これらのファイルは、FTPデーモンであるwu-ftpdの設定ファイルだ。このうち、ftpaccessがwu-ftpdの動作の詳細を設定するファイルで、これ以外のファイルはftpconversionsを除けばセキュリティ関連の設定ファイルである。

ftpusersは、FTPを許可しないユーザーを記述する。初期状態では、セキュリティ上問題になる可能性があるrootやdaemonなどが設定されている。

ftpgroupsは、ftpusersと同様にFTPを許可しないグループを記述する。

ftphostsではホストのユーザーごとにアクセス許可／禁止を設定できる。ftpuserとは異なり、特定のホストの特定のユーザーのみといった細かい設定が可能だ。設定は、allow（許可）またはdeny（禁止）に続けてユーザー名とホスト名、またはIPアドレスを指定する（**リスト5**）。

ftpconversionは、tarなどを利用したアーカイブ転送や、gzipなどによる圧縮転送を行う場合にプログラムと拡張子の対応を定義するファイルだ。標準的なものはあらかじめ定義されているので、変更する必要はないだろう。

なお、Turbolinux 7やKondaraMNU/Linux 2.0では、wu-ftpdではなくProFTPDを採用している。このため、これらのファイル群が存在しない。ただし、ProFTPDでもftpusersを使用するため、このファイルだけは存在している。

このファイルは、FTPサーバであるProFTPDの設定ファイルだ。このため、ProFTPDを採用しているTurbolinux 7と、Kondara MNU/Linux 2.0、Vine Linux 2.1.5などにのみ存在している。なお、Turbolinux 7ではproftpdというディレクトリ内にproftpd.confが置かれている。

ProFTPDは単独で細かいアクセス制限ができるため、通常はスーパーサーバを経由せず単独で起動することが多い。この設定は、proftpd.conf内にあるServerType行で行う。この行にstandaloneを指定した場合は単独で、inetdを指定した場合はxinetd経由で起動される。

このディレクトリは、Webサーバ

Apacheが使用するディレクトリで、さらにconf、logs、modulesというディレクトリが存在する。このうちconfディレクトリ以外はシンボリックリンクだ。

confディレクトリにはhttpd.conf、srm.conf、access.conf、magicといった、Apacheの設定ファイルや関連ファイルが収められている。ただし、現在ではsrm.confとaccess.confは使われなくなり、設定ファイルはhttpd.confにまとめられた。

なおTurbolinux 7では、confディレクトリにSSL関連のディレクトリと設定ファイルhttpd.bootopt、ソースコードのバージョン管理システムであるCVS（Concurrent Versions System）リポジトリをWeb上で公開するためのCVSwebの設定ファイルcvsweb.confも置かれている。

logsディレクトリは、/var/log/httpdディレクトリへのシンボリックリンクで、Apacheのログが格納される。また、modulesディレクトリは、/usr/lib/apacheまたは、/usr/libexec/apacheへのシンボリックリンクで、Apacheの機能拡張モジュールが収められている。

Turbolinux 7とKondara MNU/Linux 2.0では、サーバサイドスクリプト言語であるPHP関連のファイルもhttpdディレクトリに置かれている。

mime.types

WebサーバApacheが利用するMIMEタイプの定義ファイルがmime.typesだ。MIMEタイプは、「メディアタイプ／メディアサブタイプ 拡張子」とった形式で指定するが、一般的なMIMEタイプは設定済みなので通常は書き換える必要はない。なお、Turbolinux 7ではhttpdディレクトリ内にも存在する。

squid/

このディレクトリは、Webのキャッシュサーバ（プロキシサーバ）Squidの設定ファイルを格納するディレクトリだ。Squidの設定は、このディレクトリにあるsquid.confで行う。Squidを使ってWebコンテンツをキャッシュすることで、ネットワークトラフィックの軽減が実現できる。

rc.news
news/

これらは、NetNewsサーバINNが使用する設定ファイルとディレクトリだ。

ntp/
ntp.conf

サーバなどの内部時計が狂うと、ファイルのタイムスタンプが前後してしまうなど、重大な問題となる可能性がある。特に、ログファイルに正確な時間が記録できなかったり、最新のファイルが過去のファイルに上書きされてしまったりする可能性がある。

こうした問題を回避するために、NTP（Network Time Protocol）が用意されている。NTPを使うと、ネットワークを介してマシンの時計を合わせることができる。LinuxでNTPを利用する場合、xntp3というパッケージが使われることが多い。xntp3には、xntpdというNTPサーバが含まれており、ほかのNTPサーバと協調動作することで時計の同期を取ることができる。

このxntpdが動作する際に必要なデータを格納するディレクトリがntpで、ntp.confはxntpdの設定ファイルだ。

mail/
sendmail.cf
sendmail.st
smrsh/

これらのディレクトリとファイルは、メール配送エージェントsendmail関連のファイル群だ。

sendmailの動作を決定するメインの設定ファイルがsendmail.cfになる。このファイルの記述方法は極めて複雑なので、比較的簡単な書式のマクロファイルsendmail.mcを編集し、GNUのマクロプロセッサm4で処理することでsendmail.cfを作成することも多い。日本では、sendmail.cfの作成ツールとして、WIDEプロジェクトの中村素典氏が作成したCFというパッケージがよく使われるが、こちらは最新のsendmailには対応していない。

sendmail.stファイルは、sendmailが動作状態を書き込むファイルなのでシステム管理者が変更する必要はない。

mailディレクトリはsendmail関連のファイルが格納されたディレクトリだ。

smrshディレクトリは、制限シェルであるsmrshから実行できるコマンドを保存しておくディレクトリだ。このディレクトリにあるコマンドは、sendmailから呼び出されたsmrshが実行できる唯一のコマンド群となる。これはセキュリティ上の配慮だ。実際にはコマンドそのものではなく、コマンドへのシンボリックリンクを作成するのが一般的だ。

aliases
aliases.db

aliasesは、メール配送エージェントであるsendmailが配信するメールアド

レスの別名を定義するファイルだ。たとえば、

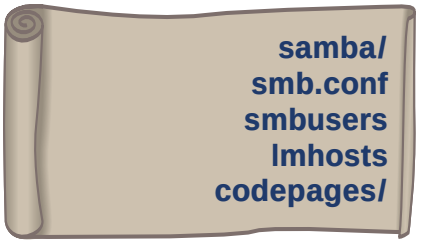
```
sales: taro, hanako
```

と定義しておくと、sales宛てに送信されたメールをtaroとhanakoに配送する。デフォルトでは、postmaster宛のメールなどがroot宛に配送されるように設定されている。この別名の機能を使えば、root宛のメールを管理者の一般ユーザーアカウントに配送することができる。なお、sendmailは実際にはaliasesではなく、aliases.dbというファイルを参照している。これは、aliasesを元に作成されるデータベースファイルだ。したがって、aliasesを書き換えたらaliases.dbファイルを更新する必要がある。aliases.dbファイルの更新はnewaliasesコマンドを使う。このコマンドを実行するにはrootユーザー権限が必要だ。

このディレクトリには、オーディオストリーミングサーバを実現するicecastの設定ファイルが格納される。icecastは、サーバ機能（icecast server）とオーディオストリームの送信とコントロールを行う（shout）の2つのプログラムで実現されている。オーディオストリームのリクエストや制御はshoutが行い、icecastはshoutから受け取ったストリームを送信する。

ファイル／プリンタサーバ関連の設定ファイル

ここでは、LANなどで使われるファイルやプリンタの共有に関する設定ファイルを解説する。



これらのファイルとディレクトリは、WindowsからLinuxに接続されたファイルシステムとプリンタを共有するためのSambaの設定ファイル群だ。

smb.confがSambaの基本的な設定を行うファイルで、共有するディレクトリやプリンタの設定を行う。

smbusersは、Linuxのユーザー名とSambaのユーザー名を対応づけるための設定ファイルだ。

lmhostsはSambaがWindowsネットワークにあるホストの名前（NetBIOS名）を解決するための定義ファイルで、ホスト名、IPアドレスを定義する対応表となる。

codepagesディレクトリには、各国語に対応するコードページのデータが格納されている。

なお、Turbolinux 7以外のディストリビューションは、/etc/sambaディレクトリにこれらのファイル群が格納されている。

ネットワークでファイルシステムを共有するNFS（Network File System）のセキュリティ情報を設定するファイルがexportsだ。NFSを使えば、UNIX（Linux）のディレクトリをネットワークを介してほかのマシンにマウントできるようになる。

exportsには、マウントを許可するマシン名と、ディレクトリ名、許可属性を記述する。たとえば、**リスト6**のように記述すると、/home/shareディレクトリをhoge01、hoge02というマシンから読み書き可能な属性でマウントでき、hoge03からは読み出しのみ可能な属性でマウントできるようになる。

MacintoshからLinuxのファイルシステムとプリンタを共有するためのNetatalkに関連する設定ファイルが収録されているディレクトリだ。

AppleVolumes.defaultとAppleVolumes.systemは、ファイルを共有するためのafpd（Apple Filing Protocol Daemon）を設定するためのファイルだ。それぞれ、ゲストユーザー用、ユーザーアカウントを持つユーザー用に分かれている。このファイルには、共有するディレクトリの指定や、コードページ、拡張子の対応などを記述する。

afpd.confは、afpdの設定ファイルで、afpdの動作と設定を行うためのファイルだ。コードページや認証関連の設定を行う。前述した2つのファイルのパスもここで指定する。このファイルで指定するコードページは、nlsディレクトリを指定する。

atalkd.confは、Macintoshのプロトコルであ るAppleTalkを処理するための、atalkd（AppleTalk Daemon）を設定するためのファイルだ。このファイルでは、インターフェイス名やIPアドレス、ゾーンなどを指定する。

netatalk.confは、afpdが使用するもっとも基本的なファイルで、AppleTalk名やゾーンの設定を行う。

papd.confは、papd（Printer Access

▼リスト6　exportsファイル

```
/home/share hoge01(rw) hoge02(rw) hoge03(ro)
```

Protocol Daemon）の設定ファイルで、プリンタ関連の設定ファイルだ。

nwserv.conf
nwserv.stations

これら2つのファイルはLinuxマシンをNetWareサーバとして稼働させるための mars_nwe（ MARtin Stover's NetWare Emulator）の設定ファイルだ。mars_nweを使うと、Netwareクライアントから Linuxマシンに接続してファイルやプリンタを共有することができるようになる。

nwserv.confはmars_nweの設定ファイルで、このファイルで共有するディレクトリなどを設定する。

nwserv.stationsには、どのLinuxマシンがプライマリファイルサーバとなるマシンかを登録する。

そのほかの設定ファイル

ここでは、これまでのジャンルに当てはまらないネットワーク系の設定ファイルを解説する。

cipe/

離れた場所にあるLANをインターネットで接続し、あたかも同一のLANであるかのように利用できるVPN（Virtual Private Network）の実装であるCIPEが使用するディレクトリだ。デーモンはcipedとなる。

cipeディレクトリには、通信を行う際に必要となるスクリプトと、LANの設定ファイルを格納する。

logcheck/

ログファイルにはアタックの痕跡など、重大な情報が記録されている可能性もある。logcheckは、こうしたログの中から、重要と思われる内容を抜き出してrootユーザー宛にメールを送信するログチェックツールだ。

logcheckがログファイルから抜き出す内容は、logcheckディレクトリ内にある定義ファイルの内容とパターンが一致したものだ。定義内容はワイルドカードで指定することができる。

logcheck.hackingはハッキングやアタックなどの痕跡と思われる内容、logcheck.violationsはエラーや失敗などのうち、通知するものが記述されている。ここに記述されたパターンと一致したログがあった場合は、それぞれ「Active System Attack Alerts」、「Security Violations」として通知される。

残り2つのignoreが付いたファイルは、通知対象外とする内容だ。

なお、logcheckは1時間ごとにcronデーモンによって起動される（/etc/cron.hourlyディレクトリに置かれる）。

pxe.conf
mtftpd.conf

pxe.confは、コンピュータの起動やOSのインストール、診断などをネットワークを経由したリモート環境で行えるようにするPXE（Preboot eXecution Environment）が使用するファイルだ。ただし、PXEを使用するには、PXEに準拠したROM BIOSを搭載したネットワークカードが必要だ。PXEに準拠したネットワークカードを利用すれば、ディスクレスシステムを構築することができる。また、Linuxのブートイメージを用意することで、クライアントマシンにネットワーク越しにLinuxをインストールすることも可能だ。

なお、ブートイメージを転送する際にはMTFTP（Multicast Trivial File Transfer Protocol）サーバが必要となる。このMTFTPサーバの設定を行うのがmtftpd.confファイルだ。

services

RFCで定義されているウェルノウンポート番号とサービス、プロトコルの種類を定義しているファイルがservicesだ。

このファイルに定義されている内容はRFCで定義されているものなので、書き換える必要はない。また、ネットワークデーモンをrpmコマンドを使ってインストールすれば、必要な情報が自動的に追加される。

snmp/

ネットワーク管理プロトコルである、SNMP（Simple Network Management Protocol）サーバの設定ファイル、snmpd.confが置かれているディレクトリだ。SNMPは、ネットワークに接続されている機器類が正常に動作しているかどうかをリモートで監視することができる。

wlan/

Turbolinux 7にのみ存在するこのディレクトリは、無線LAN用の設定スクリプトが保存されている。この無線LAN用の設定スクリプトは、linux-wlan Projectが開発しているIntersilのPRISMというチップセット用のものだが、互換チップセットを搭載した無線LANカードでも使用することができる。日本で入手できる無線LANカードでは、メルコの製品などが該当する。

アプリケーション

ディストリビューションにはいろいろなアプリケーションソフトウェアが含まれている。これらのソフトウェアの中には、/etcディレクトリに設定ファイルを置くものも多い。

このセクションでは、アプリケーションソフトウェアや、ツールの設定ファイルをいくつか紹介しよう。

a2ps.cfg
a2ps-site.cfg

テキストファイルをPostScriptに変換するツールa2psの設定ファイル。1ページごとにヘッダやフッタを付けて、出力日時、ファイル名、ページ番号を出力してくれる。a2psは、デフォルトではプリンタへ出力するので、

```
a2ps　テキストファイル名
```

とするだけで、テキストファイルをプリントアウトしてくれる。もちろん、printtoolなどでプリンタの設定をあらかじめ行っておく必要がある。

a2ps.cfgには、**リスト1**のように、用紙ごとの印刷範囲マージンの指定や、1枚に出力するページ数などが記されている。

a2ps-site.cfgは、a2ps.cfgから読み込まれるファイルで、デフォルトの用紙サイズやプリンタごとの設定といったローカルな設定を記述する。

たとえば、**リスト1**の83行目にある「Options: -2」を、-1に変更すれば、1枚に1ページ分が印刷される。

a2psでは、「-O　ファイル名」オプションを付けて、PSファイルを作成することもできる。Red Hat 7.1のGNOME上で、GV（Ghostview）によってそのPSファイルを表示したのが、**画面1**である。

amanda/
amandates

バックアップツールAmanda（Advanced Maryland Network Disk Archiver）の設定ファイル。Amandaは、サーバ／クライアント型で動作するソフトで、複数のコンピュータからネットワークを経由して、サーバに接続されたテープドライブへデータをバックアップすることができる。

バックアップセットごとにディレクトリ（たとえばDailySet1）を作成して設定ファイルを用意しておく。バックアップデバイスの指定やネットワークのトラフィック量、バックアップサイクルなどを指定することができる（**リスト2**）。

amandatesは、Amandaがバックアップした日時などを記録するファイルである。

詳細な設定は、AmandaのWebページ（http://www.amanda.org/）を参照してほしい。

▼リスト1　a2ps.cfgの設定例（一部）

```
 41: ########################################################################
 42: # 1)            Definition of some media                               #
 43: # (Must be defined before --medium)                                    #
 44: ########################################################################
 45: # Medium: name, width height [llx lly urx ury]
 46: Medium: A3          842    1190
 47: Medium: A4          595     842
 48: Medium: A5          420     595
 49: Medium: B4          729    1032
 50: Medium: B5          516     729
 51: Medium: Letter      612     792
       :
 82: # Two virtual pages per sheet
 83: Options: -2
```

1枚の用紙に入れるページ数の設定

▼リスト2　amanda/DailySet1/amanda.confの内容（一部）

```
  9: org "DailySet1"          # your organization name for reports
 10: mailto "amanda"          # space separated list of operators at your site
 11: dumpuser "amanda"        # the user to run dumps under
 12:
 13: inparallel 4             # maximum dumpers that will run in parallel
 14: netusage  600 Kbps       # maximum net bandwidth for Amanda, in KB per sec
 15:
 16: dumpcycle 4 weeks        # the number of days in the normal dump cycle
 17: runspercycle 4 weeks     # the number of amdump runs in dumpcycle days
 18: tapecycle 25 tapes       # the number of tapes in rotation
```

dumpdates

BSD由来のバックアップツールdumpは、ファイルシステム（1つのパーティション）中のファイルをテープなどのデバイスにバックアップする。バックアップを作成すると、dumpdatesにダンプレベルと日付、時間を記録する。たとえばdumpdatesに、

```
/dev/hda6 3 Sun Aug 27 20:36:01 2001
```

と記録されていれば、ファイルシステム/dev/hda6が最後にバックアップされたのは8月27日で、ダンプレベルは3だという意味になる。ダンプレベルは0～9の数字でdumpコマンド実行時に引数として指定する。dumpコマンドは、指定された引数より小さなダンプレベルのバックアップ記録がdumpdatesファイル中に見つかれば、その日付以降に更新されたファイルをバックアップし、見つからなければファイルシステム全体をバックアップする。このようにして差分バックアップを取ることが可能になっているわけだ。

jfbterm.conf

kon（漢字コンソール）と同様の機能を持つJFBTERMの設定ファイル。JFBTERMは、Linuxのコンソールをフレームバッファを利用して表示させた場合に、コンソール上で日本語を表示可能にするツールである。

フレームバッファコンソールは、カーネル2.2から追加された機能で、ブート時にペンギンのロゴが表示されたり、コンソールの表示文字数を多くしたりすることができる。ただし、konは使用できないため、代わりにjfbtermコマンドを利用する。

jfbterm.confには、環境変数TERMに設定する値やフォントの指定を記述するが、ほとんどの場合は変更する必要はないだろう。

htdig.conf

Webページの全文検索システム「ht://Dig」（http://www.htdig.org/）のツールである、htdig、htsearch、htmergeなどで使用される設定ファイル。

インターネット上のWebページや、wget、WWWOFFLE（World Wide Web Offline Explorer）などによってダウンロードしたHTMLファイルを指定することで、インデックスを作成し、キーワードで検索できる。

ht://Digは、KDEからも利用されており、KDEヘルプセンターの検索タブによって表示される画面から、マニュアルやman、infoなどを検索することができる（**画面2**、**3**）。

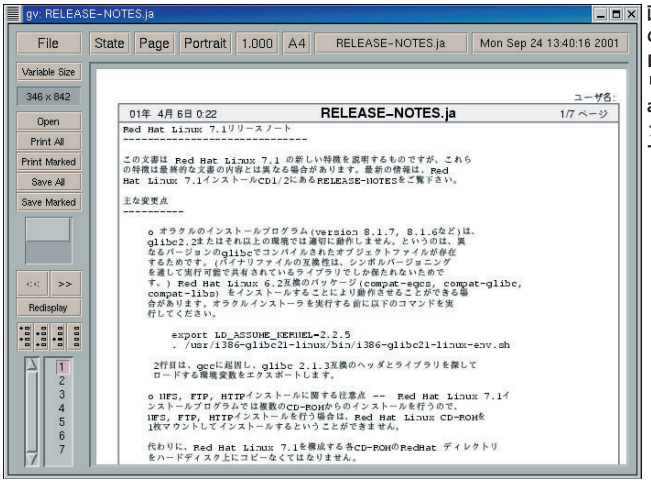
info-dir pinforc

GNUプロジェクトでは、多くのドキュメントをGNU texinfo形式で配布している。ハイパーテキストのファイル形式なので、リンクを辿っていくことで必要な情報を順次見ていくことができる。

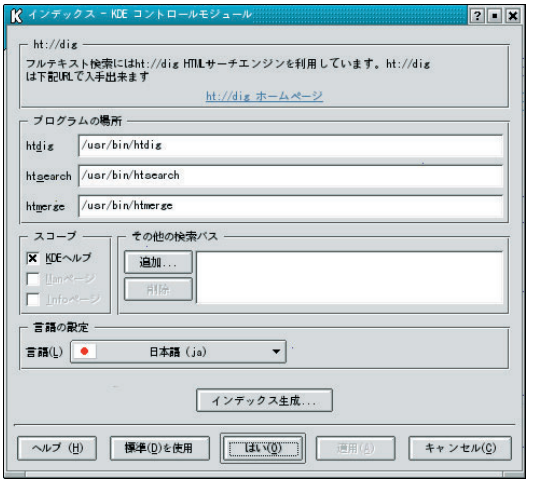
テキストベースのtexinfoリーダinfoや、テキストWebブラウザであるLynxのような操作方法を持つpinfo、そしてGNOMEのヘルプブラウザ（**画面4**）から見ることが可能だ。

infoまたは、pinfoを使用する場合は、コマンドラインから引数なしで起動すると、デフォルトファイルとして各種ドキュメントへのリンクを持つinfo-dirファイルが開かれる。引数として、調べたいコマンドやファイル名を付ければ、それに対応するドキュメントが表示される。

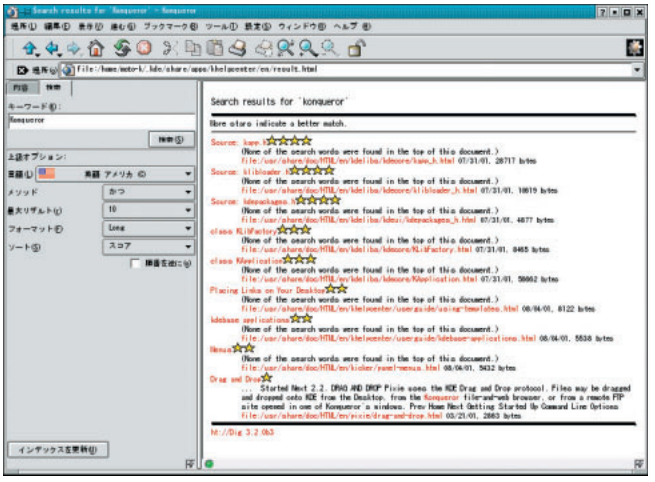
pinforcは、pinfoコマンドの設定ファイルで、画面上の本文やハイパーリンクの色の指定やキー操作などが書か



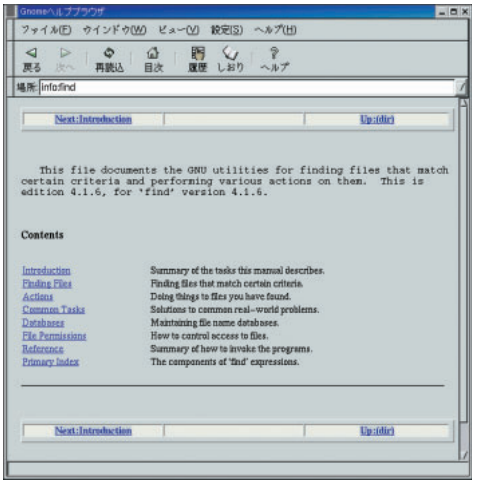
画面1　gvでa2psの出力を表示
Red Hat Linux 7.1のリリースノートを、a2psでPostscriptファイルに出力し、gvで表示したところ。



画面2　KDEヘルプセンターからインデックスを更新する「プログラムの場所」に、ht://Digのパスが指定されている。「その他の検索パス」で追加も行える。



画面3　KDEヘルプセンターの検索タブでキーワード検索を行う
キーワード検索は、ht://Digのツールが使用され、ドキュメントの全文検索の結果スコアが☆マークで表示される。



画面4　GNOMEヘルプブラウザ
ハイパーテキストのドキュメントで記述されたinfoの中の、findコマンドの説明をGNOMEヘルプブラウザで見たところ。

れている。

logrotate.conf
logrotate.d/

カーネルやデーモンの稼働状況を記録したログは、放っておくと際限なく増えてしまう。かといってログを取らないのは危険だ。この悩みを解決するのがlogrotateユーティリティだ。logrotateは、logrotate.confで設定する内容に従って一定期間ごとにログファイルを分割し、古いログを圧縮／削除する。

Red Hat Linux 7.1の/var/logディレクトリを見てみると、ログはファイル名の末尾に、.1、.2などの数字を付けて保存されているのがわかる（**画面5**）。

デーモンとして常駐しているWebサーバのApacheなどでは、アクセスがあるたびにログファイルに書き込みを行っている。ログファイルはオープンされたままなので、logrotateがコピーしたりファイル名を変更しても、すでにオープンされているファイルへログを書き込むことになる。

そのため、ログファイルをバックアップ用のファイル名に変更した時点で、デーモンに対してシグナルを送り、新しいログファイルに書き込むように指示するのが、logrotate.dディレクトリに置かれている各サービス用のファイルだ。

```
# ls -p /var/log
XFree86.0.log  cron.2          maillog.1   pacct     spooler     vbox/
boot.log       dmesg           maillog.2   sa/       spooler.1   wtmp
boot.log.1     fax/            messages    samba/    spooler.2   xferlog
boot.log.2     htmlaccess.log  messages.1  savacct   squid/      xferlog.1
canna/         httpd/          messages.2  secure    statistics  xferlog.2
cron           lastlog         netconf.log secure.1  usracct
cron.1         maillog         news/       secure.2  uucp/
```

画面5　Red Hat Linux 7.1のログディレクトリ

logrotate.confは、**リスト3**の例では、1週ごとに分割され、4週間分のバックアップを保存するように指定されている。ログが非常に大きくなる場合には、コメントアウトされている15行めのcompressの“＃”を外してやれば、ログファイルを圧縮して保存することができる。

logrotate.confやlogrotate.dディレクトリ以下にあるファイルの書き方については、man logrotateで見ることができるので、そちらを参考にするとよいだろう。

medusa/

ディスク上の全ファイルの情報を検索するツールMedusaの設定ファイルがあるディレクトリ。Medusaは、GNOME 1.4に含まれる高機能ファイルマネージャNautilusから利用されるので、GNOME 1.2を使用しているRed Hat Linux 7.1とLASER5 Linux 7.1には含まれていない。

Medusaは、findコマンドとは違って検索のたびに全ファイルを調べるのではなく、ディスク上のファイル情報をあらかじめインデックス化しておくことで高速に検索することができる。また、アプリケーションの実行によってファイルが書き換わることで、インデックスの情報が古くなってしまうため、medusaデーモン（ファイル名はmedusa-searchd）によってインデックスを一定の間隔で更新する。

MedusaがインストールされるKondaraやTurbolinuxでも、デフォルトでは動作しないようになっているようだ。設定ファイルmedusa/medusa.confは、**リスト4**のようにアクティブに

なっていない。そこで、Medusaを使えるように「enable=no」を「enable=yes」に変更する。

最初のインデックスファイルは、medusa-indexdプログラムで作っておく。/var/medusaディレクトリがない場合は、mkdir /var/medusaとして作っておこう。システムにインストールされているファイルの容量にもよるが、medusa-indexdは、ディスク上の全データをインデックス化するため非常に時間がかかるのと、インデックスファイル自体も大きなサイズ（数十Mバイト以上）になることに気をつけておこう。

インデックス化しないファイルは、medusa/file-index-stoplistに書いておく。初期値では、/procと/devディレクトリが記されている。また、medusa/mount-type-stoplistには、マウントしたファイルシステムをインデックス化しないように、nfs、afs、shm、autofs、smbfsが記されている。

インデックスファイルができたら、

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

として、medusaデーモンを再起動しておく。

これで、Nautilusの検索ボタン（**画面6**）によってハードディスク上のファイルの検索を行うことができるようになる。

mtools.conf

Linuxは、DOSやWindowsで使われるFATファイルシステムをサポートしている。しかし、FATファイルシステムのフロッピーディスクを読み書きするのに、いちいちマウントしたりアンマウントするのは意外と面倒なものだ。

mtoolsは、マウントすることなく、FATやVFATファイルシステムのディスクを扱えるツール群だ。mcopyやmdel、mdirなど、DOSでおなじみのコマンドに“ m ”が付いたツールがいろいろと用意されている。

たとえば、

```
# mcopy /etc/issue a:issue
```

とすれば、/etc/issueファイルをFATファイルシステムのフロッピーディスクにコピーできる。

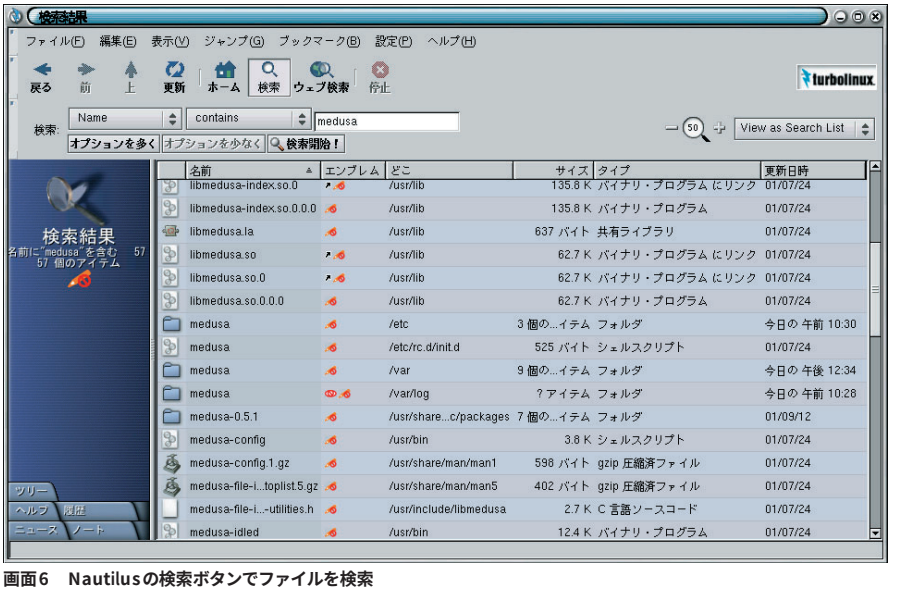
この例ではフロッピーディスクを指定するのに“ a: ”としている。これは、

▼リスト3　logrotate.confの内容

```
 2: # rotate log files weekly
 3: weekly          ← 1週間単位でログを分割
 4:
 5: # keep 4 weeks worth of backlogs
 6: rotate 4        ← 4週間分を保存する
 7:
 8: # send errors to root
 9: errors root
10:
11: # create new (empty) log files after rotating old ones
12: create
13:
14: # uncomment this if you want your log files compressed
15: #compress       ← コメントを外すとログを圧縮する
```

▼リスト4　medusa/medusa.confの内容

```
# Use this file to turn medusa off completely on your
# system.  Setting enabled to "yes" will turn on medusa services.
# The default is that enabled is set to "no" and services are not
# active.
enabled=no          ← noをyesに変更しないと動作しない
```



画面6　Nautilusの検索ボタンでファイルを検索
GNOMEのファイルマネージャNautilusでは、ディスクファイルの情報をインデックス化して検索できるMedusaを利用して検索を行う。

mtools.confの中でドライブa:を/dev/fd0として設定済みだからだ（**リスト5**）。このほかにも、ハードディスクのFATパーティションを任意のドライブに割り当てることもできる。

namazu/

日本語全文検索システムNamazu（http://www.namazu.org/）の設定ファイルが収められている。Namazuは、アスキー（http://www.ascii.co.jp/）をはじめとする多くの企業や大学、官公庁などのWebサイトで実際に利用されている日本語全文検索システムである。

CGIとして動作させる以外にも、Tck/TkやPerl、Ruby、Javaなどを利用した検索クライアントソフトも開発されている。

wgetやWWWOFFLEなどでダウンロードしたHTMLファイルを元に、自分用の検索システムを構築するのに利用するのもよいだろう。

odbc.ini
odbcinst.ini

PostgreSQL、MySQLなどのデータベースとアプリケーション間で通信するためのODBC（Open Database Connectivity）ドライバの設定ファイル。ODBCは、Microsoftによって提唱されたデータベースアクセスの標準仕様。

termcap

端末の特性を指定するデータベースファイル。ただし現在のLinuxではterminfoが使われており、termcapは互換性を維持するために残されている。

UNIXやLinuxでは、さまざまな種類の端末が使われてきた。現在のPCはグラフィックスの表示ができてあたりまえだが、以前はキャラクタ端末をUNIXマシンにつないで使うのが普通だったのだ。X Window Systemで動作するxtermやktermなどは、「端末エミュレータ」と呼ばれているのはご存じのとおりだ。

一般に、UNIXやLinuxのプログラムは特定の端末に依存したコーディングはなされていない。画面の制御を実際に行うのは画面操作ライブラリの役目だ。画面操作ライブラリは、環境変数TERMに設定された端末の種類をもとに、termcapあるいはterminfoデータベースを調べ、その端末固有のエスケープシーケンスをマッピングする。

Kondara MNU/Linux 2.0のtermcapを見てみると、ファイルの末尾にkonとjfbtermのエントリが追加されている（**リスト6**）。環境変数TERMに“kon”や“jfbterm”が設定されていれば、このエントリが使用される。

最初の行は、端末の名前とそのエイリアスだ。この例では名前が“kon”で、エイリアスが“kanji on console”となる。エイリアスは複数あってもよく、スペースを含まないエイリアスは名前同様に環境変数TERMに設定して使うことができる。

あとの部分は、：（コロン）で区切られた機能コードと値の定義だ。たとえば、14628行目のcoコードは端末画面の桁数を、liコードは行数を定義する。すなわち、この端末は80桁25行ということになる。そのほかの機能コードについては、man termcapで調べてほしい。

最初にも書いたとおり、最近のLinuxでは、termcapの代わりにterminfoが使われている。terminfoは、termcapに似た書式のソースファイルを、マシンが処理しやすいバイナリのデータベースファイルに変換して使う。データベースファイルは、エントリごとに独立したファイルになっていて、/usr/share/terminfoディレクトリ以下にある、名前の頭文字のサブディレクトリに収められている。ktermだったら、/usr/share/terminfo/k/ktermという

▼リスト5　mtools.confの例（一部）

```
 1: # Example mtools.conf files.  Uncomment the lines which correspond to
 2: # your architecture and comment out the "SAMPLE FILE" line below
 3:
 4: # Linux floppy drives
 5: drive a: file="/dev/fd0" exclusive 1.44m mformat_only
 6: drive b: file="/dev/fd1" exclusive 1.44m mformat_only
 7:
 8: # First SCSI hard disk partition
 9: #drive c: file="/dev/sda1" 10:
11: # First IDE hard disk partition
12: #drive c: file="/dev/hda1"
```

1台目のフロッピーディスクをa:ドライブにする

2台目のフロッピーディスクをb:ドライブにする

1台目のSCSIハードディスクにある最初のパーティションをc:ドライブにする例（コメントになっている）

1台目のIDEハードディスクにある最初のパーティションをc:ドライブにする例（コメントになっている）

ことになる。

terminfoのデータベースはバイナリファイルなので、テキストビューアで見ることはできない。コンパイル済みのファイルはinfocmpコマンドでソースに変換できるので、内容を変更したいときは変換したソースを書き換え、ticコマンドでコンパイルし直すことになる。

tripwire/

システムファイルやディレクトリに対する変更を検出するセキュリティソフトTripwireの設定ファイルが収められている。

Tripwireは、あらかじめ各ファイルの内容を元に作成しておいた基準データベースと、実際のファイルを比較し、改ざんされていないかを検出することができる。ファイルの変更日時やサイズといった各種属性と、ファイル自体のハッシュ値を比較するので、信頼性の高いチェックが行われる。

Linuxサーバをインターネットに接続していると、セキュリティホールを突いたアタックなどによって侵入され、Webページの内容などのファイルを勝手に書き換えられてしまうこともある。通常アタックしてきたクラッカーは、侵入の痕跡を見つからないように隠してしまうので、簡単には気がつかないことも多いが、Tripwireを適切に使用すれば、侵入されたことがメールで報告されるので、素早く対策をとることが可能になる。

Tripwireのポリシーファイルtripwire/twpol.txtには、非常に多くのファイルが記述されている。実際の運用に際しては、ポリシーファイルの設定をシステムにインストールされているファイルと一致するように修正する必要があるだろう。

wgetrc

wgetは、HTTPとFTPに対応した一括ファイルダウンロードツールだ。コマンドラインで動作し、ダウンロードしたいファイルのURLを引数に指定する。プロキシサーバにも対応し、FTPのpassiveモードを利用することもできるので、ファイアウォールの内側からでも使うことができる。

転送に失敗したときは、リトライさせることもできるし、サーバが対応していれば、途中までダウンロードしたファイルの続きをダウンロードすることも可能だ。プロキシサーバを利用するかどうか、プロキシサーバの指定、リトライ回数の指定などは、コマンドラインオプションでも指定できるが、wgetrcに設定しておくとよいだろう(**リスト7**)。

wgetをGUIで使うには、GTransferManager(http://gtm.sourceforge.net/)を利用すればよいだろう。

▼リスト6　termcapファイルにあるkonとjfbtermのエントリ(Kondara)

```
14627: kon|kanji on console:\
14628:         :am:eo:mi:ms:ut:xn:xo:\
14629:         :co#80:it#8:li#25:\
14630:         :&7=^Z:@7=\E[4~:AL=\E[%dL:DC=\E[%dP:DL=\E[%dM:\
14631:         :F1=\E[23~:F2=\E[24~:F3=\E[25~:F4=\E[26~:F5=\E[28~:\
14632:         :F6=\E[29~:F7=\E[31~:F8=\E[32~:F9=\E[33~:FA=\E[34~:\
14633:         :IC=\E[%d@:K2=\E[G:al=\E[L:bl=^G:cd=\E[J:ce=\E[K:\
14634:         :ch=\E[%i%dG:cl=\E[H\E[J:cm=\E[%i%d;%dH:cr=^M:\
14635:         :cs=\E[%i%d;%dr:ct=\E[3g:cv=\E[%i%dd:dc=\E[P:dl=\E[M:\
14636:         :do=^J:ei=\E[4l:ho=\E[H:ic=\E[@:im=\E[4h:k1=\E[[A:\
       :
14646: jfbterm|Japanized framebuffer terminal:\
14647:         :am:eo:mi:ms:ut:xn:xo:\
14648:         :co#80:it#8:li#25:\
14649:         :&7=^Z:@7=\E[4~:AL=\E[%dL:DC=\E[%dP:DL=\E[%dM:\
```

▼リスト7　wgetrcの例(一部)

```
29: # You can lower (or raise) the default number of retries when
30: # downloading a file (default is 20).
31: tries = 30
       :
73: # You can set the default proxy for Wget to use.  It will override the
74: # value in the environment.
75: http_proxy = proxy.ascii-linux.com:8080
76: ftp_proxy  = proxy.ascii-linux.com:8080
77:
78: # If you do not want to use proxy at all, set this to off.
79: use_proxy = on
```

リトライ回数の上限(コメントを外して書き換える)

HTTPプロキシとポートの指定(コメントを外して書き換える)

FTPプロキシとポートの指定(新たにこの行を加える)

プロキシサーバを利用する(コメントを外して書き換える)

メジャーバージョンアップで変わる設定方法

Red Hat Linux 7のデビューを皮切りに、RPM採用ディストリビューションの多くがメジャーバージョンを上げてきた。これらは、Linuxカーネル 2.4を採用し、さまざまな機能が強化されると同時に、システムの設定方法も一部が変更されている。このため、以前のバージョンを使っていたLinuxユーザーは、バージョンアップ後の設定で悩むことになりそうだ。そこで、Red Hat Linux 6.2をベースにしているVine Linux 2.1.5と、最新のディストリビューションを比較して、設定方法を中心に変更点をまとめてみた。Red Hat Linux 6.2ベースのディストリビューションからバージョンアップをするユーザーは参考にしてほしい。

なお、本稿では、Vine Linux 2.1.5をVine 2.1.5、Red Hat Linux 7.1 日本語版をRed Hat 7.1、LASER5 Linux 7.1をLASER5 7.1、Kondara MNU/Linux 2.0をKondara 2.0、Turbolinux 7 WorkstationをTurbo 7 WSとそれぞれ表記する。

X Window System

2001年版のディストリビューションは、XFree86 3.xに代わり、3Dグラフィックスの描画性能やフォントレンダリング機能が強化されたXFree86 4.xを採用している（**表1**）。XFree86 4.xは、古いグラフィックスカードをサポートしないため、一部のディストリビューションでは、XFree86 3.3.6のフレームバッファサーバもインストールされる。

XFree86のバージョンが上がったことで、設定ファイルの書式が変更された。また、設定ファイルの名前が“XFree86Config”から“XFree86Config-4”に変わっているディストリビューションが多い。XFree86の設定は、ツールによって簡単に行えるが、設定ファイルをエディタで書き換える場合は注意してほしい。ほとんどの場合、Red Hat 7.1、LASER5 7.1、Kondara 2.0では、Xconfigurator、Turbo 7 WSではturboxconfigという設定ツールを使えばよいだろう。

■設定ファイル名変更のわけ

設定ファイルの名前が変更されたのはなぜなのか、興味のある読者のために説明しよう。一般のユーザーがX Window Systemを起動する際、XFree86 3.xは、**リスト1**の順番で設定ファイルを検索し、最初に見つかったファイルの内容に従って動作する。XFree86 3.3.6を採用しているVine 2.1.5では、設定ファイルの本体は/etc/X11/XF86Configであり、/usr/X11R6/lib/X11/XF86Configからリンクが張られている。

いっぽう、XFree86 4.xでは、**リスト2**に示した順で設定ファイルを検索する。これを見ればわかるように、“XF86Config”よりも、同じディレクトリにある“XF86Config-4”のほうが優先順位が高い。したがって、“XF86Config-4”をXFree86 4.xの設定ファイルとしておけば、XFree86 3.x用の設定ファイルと区別することができるのだ。

実際に、Red Hat 7.1、LASER5 7.1、Kondara 2.0では、/etc/X11/XF86Config-4をXFree86 4.x用の設定ファイルとして、/etc/X11/XF86ConfigをXFree86 3.x用の設定ファイルとして扱っている。これらのディストリビューションに含まれるXconfiguratorは、XFree86の3.xと4.xの両方に対応しており、それぞれの設定ファイルを生成してくれる。

Turbo 7 WSでは、XFree86 3.xはインストールされないのでXFree86 4.1.0で/etc/X11/XF86Configを設定ファイルとして利用する。

また、Kondara 2.0では、デフォル

	XFree86	XF86Config
Vine 2.1.5	3.3.6	XF86Config
Red Hat 7.1	4.0.3／3.3.6（FB）	XF86Config-4／XF86Config
LASER5 7.1	4.0.3／3.3.6（FB）	XF86Config-4／XF86Config
Kondara 2.0	4.0.3	XF86Config-4／XF86Config
Turbo 7 WS	4.1.0	XF86Config

表1　XFree86のバージョンと設定ファイル

▼リスト1　XFree86 3.xの設定ファイル検索順

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

<hostname>はそのマシンのホスト名を表す。

トではXFree86 3.xをインストールしないが、拡張パッケージ集のKondara ZooにXFree86 3.3.6が収録されているために、“XF86Config-4”と“XF86Config”の両方を用意しているものと思われる。

パケットフィルタリング

Linuxカーネル2.4には、netfilterという新しいパケットフィルタ機構が組み込まれている。この機能をフルに活用するためには、iptablesというパッケージを利用してフィルタリングの設定を行う。

しかし、iptablesは、カーネル2.2で使われてきたipchainsとは設定の書式が異なるため、従来のユーザーのためにipchainsと互換性を持たせるモジュールが用意されている。このモジュールを利用することで、カーネル2.2と同様にipchainsコマンドによるフィルタリング設定が可能になるが、この場合、netfilterで拡張された機能は利用できない。また、iptablesとの同時利用もできない。

各ディストリビューションがiptablesとipchains互換機能のどちらをデフォルトで使っているのかを**表2**にまとめたので、参考にしてほしい。

▼リスト2　XFree86 4.xの設定ファイル検索順

```
/etc/X11/<cmdline>
/usr/X11R6/etc/X11/<cmdline>
/etc/X11/$XF86CONFIG
/usr/X11R6/etc/X11/$XF86CONFIG
/etc/X11/XF86Config-4
/etc/X11/XF86Config
/etc/XF86Config
/usr/X11R6/etc/X11/XF86Config.<hostname>
/usr/X11R6/etc/X11/XF86Config-4
/usr/X11R6/etc/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.<hostname>
/usr/X11R6/lib/X11/XF86Config-4
/usr/X11R6/lib/X11/XF86Config
```

<cmdline>はコマンドラインで-xf86configオプションを使って指定したファイル名、$XF86CONFIGは環境変数XF86CONFIGに設定したファイル名を表す。

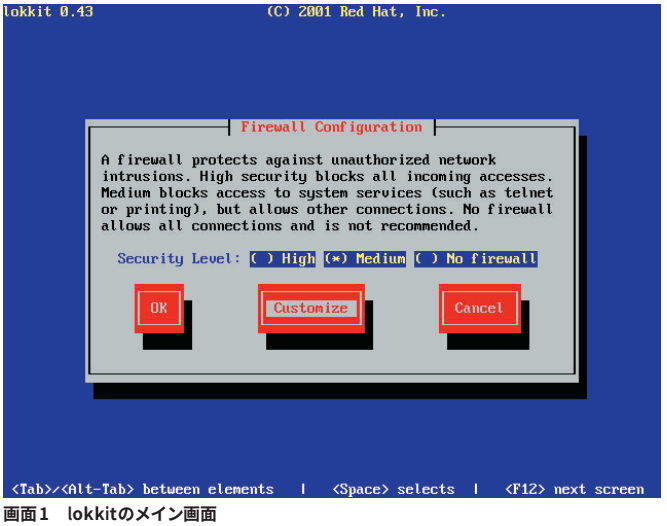
■パケットフィルタの簡単設定

Red Hat 7.1、LASER5 7.1は、インストーラの「ファイアウォールの設定」画面でパケットフィルタのルールを設定できる。セキュリティレベルを「高」、「中」、「ファイアウォールなし」の中から選び、FTPなどのサーバを運用する場合は、動作させるサーバを追加設定するだけで、適切なフィルタルールが設定されるというすぐれものだ。インストール後もlokkitコマンドで設定を変更することが可能になっている（**画面1**、**2**）。

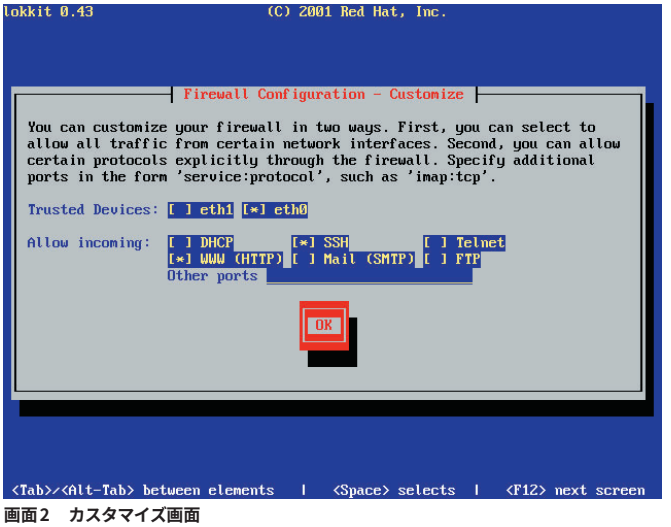
Kondara 2.0でも、インストール中に同様の画面が表示されるし、lokkitコマンドも用意されているのだが、これらで設定した内容は効力を持たないので注意したい。これは、Kondara 2.0がデフォルトでiptablesによるパケットフィルタ設定を採用しているためだ。インストーラのファイアウォール設定やlokkitは、ipchainsコマンドに対

	コマンド
Vine 2.1.5	ipchains
Red Hat 7.1	ipchains互換
LASER5 7.1	ipchains互換
Kondara 2.0	iptables
Turbo 7 WS	iptables

表2　パケットフィルタの設定



画面1　lokkitのメイン画面
3段階のセキュリティレベルを選び、パケットフィルタの設定を簡単に行える。



画面2　カスタマイズ画面
接続を許可するサーバを追加設定する。

応したツールであり、iptables環境では利用できない。Kondara 2.0は、ipchains互換モジュールをインストールしないので、これらのツールを使うのは難しい。

また、Turbo 7 WSでは、インストールの際に「セキュリティレベル設定」を「高レベル」、「中レベル」、「低レベル」の中から選択するが、これはパケットフィルタの設定ではなく、起動するネットワークサーバの種類を簡単設定するというものだ。これについては後述する。

■iptables環境では

iptablesを採用しているKondara 2.0とTurbo 7 WSでは、フィルタルールを簡単に作成してくれるツールがないので、どうにかしてルールを設定しなければならない。Linuxソフトウェアの検索サイトfreshmeat（http://www.freshmeat.net/）でiptablesをキーワードに検索すると、いくつかのフィルタリングルール設定ツールが見つかるので、これらを利用するのもよいだろう。もちろん、iptablesコマンドを使って、ゼロからルールを設定しても構わない。

設定が済んだら、その状態のルールセットを保存して、Linuxの起動時から有効になるようにしておくと便利だろう。実は、その仕組みはすでに備えられているのだ。rootユーザーになって、**画面3**のようにすれば保存操作は完了だ。次に、起動時に保存したルールが有効になるようにしよう。

```
# chkconfig --level 345 iptables on
```

これは、ランレベルが3〜5で起動する際に、“iptables”というrcファイルを実行するようにする設定だ。このrcファイルは、/etc/init.d/iptablesというシェルスクリプトで、先ほど保存した内容に従ってiptablesコマンドを実行する。

Vine 2.1.5を含め、ipchainsを採用しているディストリビューションも同じような仕組みを利用している。すなわち、ipchains-saveコマンドで/etc/sysconfig/ipchainsに保存したルールセットを“ipchains”というrcファイルで有効にしているのである

iptablesに対応するlokkitのような、簡易ルールセット設定ツールの登場に期待したい。

```
# iptables-save > /etc/sysconfig/iptables
# chmod 600 /etc/sysconfig/iptables
```

画面3　iptablesのルールを保存する

▼リスト3　Vine 2.1.5のinetd.conf（一部）

```
#ftp    stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
#ftp    stream  tcp     nowait  root    /usr/sbin/tcpd  in.proftpd
#telnet stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#gopher stream  tcp     nowait  root    /usr/sbin/tcpd  gn
```

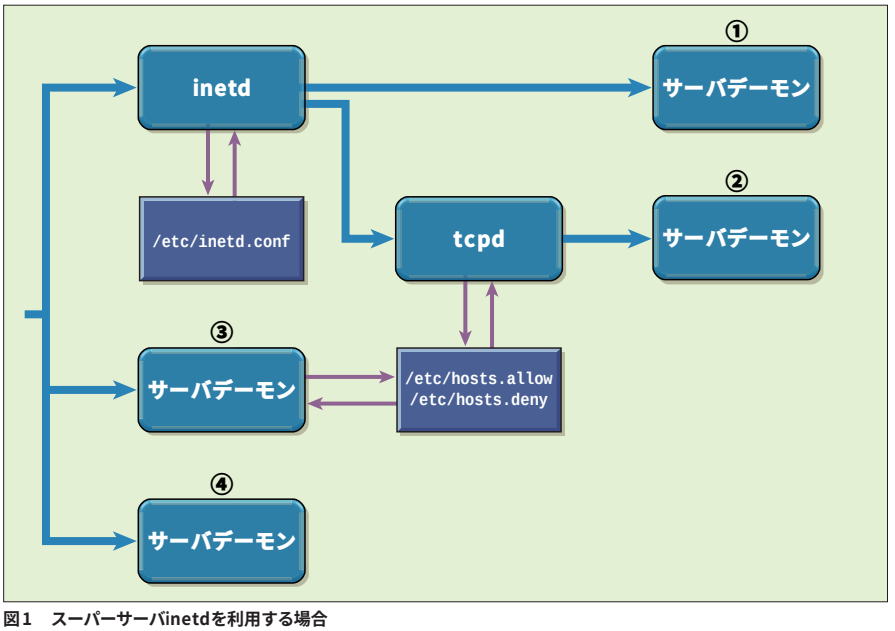


図1　スーパーサーバinetdを利用する場合

新しいスーパーサーバ — xinetd

Red Hat Linux 5.xや6.x系のディストリビューションを使ってきたユーザーにとって、/etc/inetd.confという設定ファイルはなじみ深いものだといえよう（**リスト3**）。インストールが終わると、まず、エディタでinetd.confを開いて利用したいサービスの書かれた行のコメントマークを外し、inetdを再起動したものだ。しかし、2001年版ディストリビューションにはこのファイルが存在しない。これは、従来使われてきたスーパーサーバinetdが、より高機能なxinetdに置き換わったためだ。

■スーパーサーバとは

ここで、UNIXやLinuxのネットワークサーバの起動方法をおさらいしてお

こう。ネットワークサーバのプログラムの基本動作は次のようになる。プログラムが起動されると、接続要求が来るまで待機しており、接続されたらデータをやりとりする。データのやりとりが終了すると再び待機状態になる。

しかし、ネットワークサーバの種類が増えてくると、あまり頻繁に利用されないサーバをすべて待機させておくのはメモリをはじめとする資源の無駄となる。そこで、接続の待ち受けを専門とするサーバを立てて、接続されると要求されたサーバを起動するようになった。この待ち受け専門サーバが、「スーパーサーバ」と呼ばれるものだ。

図1にある①のパターンを見てほしい。スーパーサーバのinetdは、外部からの接続要求があると、設定ファイルの/etc/inetd.confを参照し、その内容にしたがって各種サーバを起動する。このとき、inetd.confに書かれていない（コメントアウトされた）サーバは起動されない。

世の中が物騒になってくるとセキュリティにも気を付けなくてはいけない。そこで、接続を許すホストやネットワークを限定するための仕組み、TCP wappers（tcpd）が考案され、Linuxでも標準的に利用されてきた。inetdは、接続要求を受けるとサーバそのものの代わりにtcpdを起動する。tcpdは、アクセス規制の設定ファイル/etc/hosts.allowと/etc/hosts.denyを参照し、接続が許可される場合は引数として渡されているサーバ本体を起動する。**図1**の②がこのパターンだ。**リスト3**を見ると、inetdがtcpdを起動してin.telnetdなどのサーバ名を渡しているのがわかるだろう。

もちろん、ネットワークサーバがすべてスーパーサーバから起動されるわけではない。通常、SMTPサーバやSSHサーバなどはシステムとともに起動され、接続を待ち受けている。このような形態のサーバをスタンドアロンサーバという。スタンドアロンサーバは、アクセス規制のための接続制御を行わないのだろうか？　その答えは、するものもあるし、しないものもあるということになる。たとえば、OpenSSHデーモンは、TCP wrappersに含まれるライブラリを利用して、/etc/hosts.allowと/hosts.denyによる接続制御を行うことができる（**図1**の③）。今回取りあげたディストリビューションに含まれるOpenSSHデーモンはすべてTCP wrappersのライブラリが組み込まれている。

また、FTPサーバのProFTPDなどは、自身の設定ファイルで接続制御を設定可能なので、**図1**④のパターンで運用されることが多い。

■xinetdの動作

xinetdは、inetdとtcpdの持つ機能を合わせ持ち、さらに機能を強化したスーパーサーバだ。xinetdを使えば、ネットワークサーバごとに接続元による接続制御だけでなく、時刻や同時接続数による制御を行うこともできる。

xinetdの設定ファイルは、/etc/xinetd.confと/etc/xinetd.dディレクトリ以下の各ファイルだ。xinetd.confでは、すべてのサーバに共通の設定を行い、サーバごとの設定は、/etc/xinetd.dディレクトリ以下のファイルで行う。たとえば、telnetサーバの設定は、/etc/xinetd.d/telnetというファイルに書くことになる。xinetdを使った場合の接続概念を**図2**に示す。この図の②と③は、スタンドアロンサーバを示しており、**図1**の③、④と同じパターンである。

Red Hat 7.1、LASER5 7.1、Kondara 2.0のxinetdには、TCP wrappersのライブラリが組み込まれているので、/etc/hosts.denyに「ALL: ALL」と書くと接続を受け付けなくなるので気を付けよう。また、今回紹介しているディストリビューションに含まれるxinetdには、セキュリティホールが見つかっているので、各ディストリビューターのサイトから最新版を入手してアップデートする必要がある。

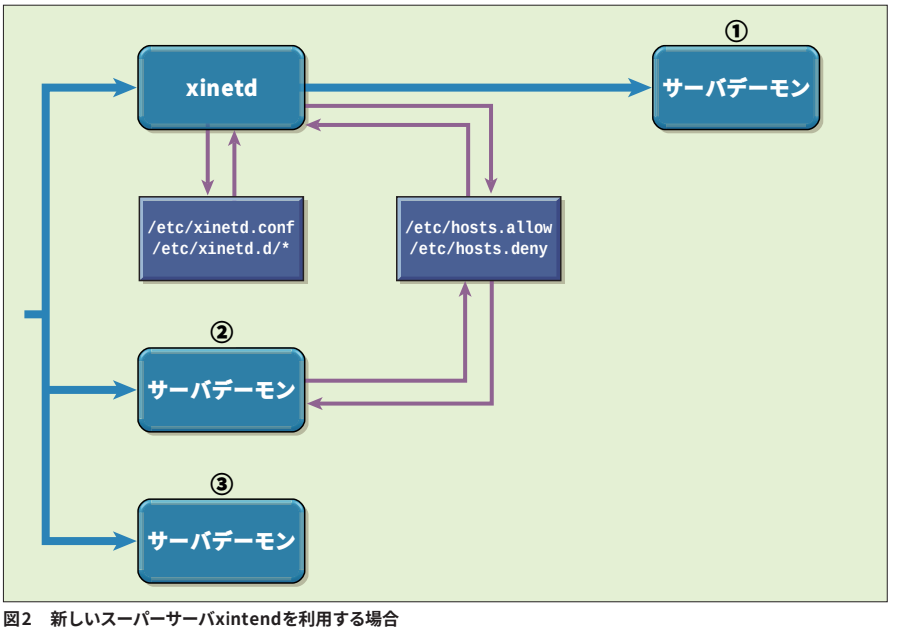


図2　新しいスーパーサーバxintendを利用する場合

■サーバの起動設定

前述の通り、RPM系のディストリビューションでは、/etc/xinetd.dディレクトリにネットワークサーバごとの設定ファイルを置いている。そのため、各サーバを有効にするかどうかはそれぞれのファイルに記述することになる。inetdでは、すべてのサーバを/etc/inetd.confで設定していたが、それとは大きく異なっているのだ。

では、xinetdでサーバを有効にする方法を見てみよう。**リスト4**は、xinetdのtelnetサーバ設定ファイルだ。「disable = no」という行は、telnetデーモンを有効にすることを意味している。「disable = yes」にすればそのサーバデーモンは起動されない。アクセス制限など、ほかのオプションについては、man xinetd.confを実行してmanページを参照してほしい。なお、設定ファイルを書き換えたら、次のようにしてxinetdの動作に反映させる必要がある。

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

さて、これでxinetdを介して使われるネットワークサーバの有効／無効を切り替える方法がわかったわけだが、各ディストリビューションには、Linuxの起動時に各サーバプログラムを有効にするか無効にするかを設定する複数のツールが含まれている。従来のバージョンでは、これらのツールはスタンドアロンサーバの設定しかできなかったが、ntsysvとchkconfigコマンドは、xinetdから起動されるサーバも扱えるように拡張されている。具体的には、先ほどの「disable」の行を見て、現在の状態を表示したり、書き換える機能が付加されているのだ。

現在の設定内容を表示する場合は、chkconfigコマンドを使うとわかりやすいだろう（**画面4**）。はじめにスタンドアロンサーバの設定内容がランレベルごとに表示され、続いてxinetdを経由するサーバの設定が表示される。次のようにすれば設定の変更も可能だ。

```
# chkconfig telnet on
```

これは、xinetdからtelnetサーバを起動できるようにする例だ。起動しないようにするには、「on」を「off」に変えて実行すればよい。スタンドアロンサーバの設定を変更するときには、ランレベルの指定もできる。たとえば、ランレベル3、4、5でWebサーバ（httpd）を起動しないようにするには次のようにする。

```
# chkconfig --level 345 httpd off
```

同様のことをntsysvコマンドで行うには、

```
# ntsysv --level 345
```

として起動し、カーソルキーで変更したいサーバを選んでスペースバーで有

▼リスト4　/etc/xinetd.d/telnetの例

```
service telnet
{
        disable = no
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
}
```

```
# chkconfig --list
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
alsasound       0:off   1:off   2:off   3:off   4:off   5:off   6:off
                        :
                        :
                    <中略>
                        :
                        :
ripd            0:off   1:off   2:off   3:off   4:off   5:off   6:off
ripngd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
zebra           0:off   1:off   2:off   3:off   4:off   5:off   6:off
xinetd based services:
        systat: off
        finger: off
        chargen:        off
        chargen-udp:    off
        daytime:        off
        daytime-udp:    off
              :
              :
          <以下略>
```

画面4　xinetdに対応したchkconfigの出力（抜粋）

効／無効を切り替える。ntsysvでは、スタンドアロンサーバとxinetdによって起動されるサーバを分けて表示しないし、ランレベルごとの設定内容も表示されないので気を付けよう。

なお、Turbolinuxで標準的に使われてきたツールのturboserviceは、スタンドアロンサーバの設定しかできないので、xinetdを介して起動するサーバはntsysvかchkconfigコマンドで設定しよう。

■Turbo 7 WSのセキュリティレベル

Turbo 7 WSは、インストール時に「セキュリティレベル」の設定を行う。「高レベル」を選ぶと、必要最低限のサーバのみを有効にし、「中レベル」ではクライアント用途に必要なものを有効にするという。「低レベル」では、ほとんどすべてのサーバを起動する（メモリもたくさん消費する）。インストールしてからこの設定を変更するには、/etc/sysconfig/securityをエディタで開き、「THRESHOLD＝」に続く数値を書き換え、setconfigコマンドを実行する。書き換える数値は、「高レベル」が80、「中レベル」が50、「低レベル」が20に対応する。これで、大まかな設定を行い、chkconfigコマンドなどで自分の環境に合った設定に調整するとよいだろう。

ネットワークサーバ関係の変更点

Red Hat Linux 6.2までは、標準的にインストールされるFTPサーバはwu-ftpd、メールサーバはsendmailと相場が決まっていた。しかし、これらのサーバソフトウェアに代わり、ProFTPDやPostfixといった、より新しく、セキュリティを強化したサーバソフトウェアを採用するディストリビューションが増えてきた（**表3**）。

wu-ftpdの設定ファイルは、/etc/ftpaccessだが、ProFTPDは/etc/proftpd.confで設定する。また、sendmailの設定を/etc/sendmail.cfで行うのに対し、Postfixは/etc/postfix/main.cfで行う。これらは、書式も異なっているので、簡単には乗り換えられないかもしれない。

■FHS準拠のディレクトリ構成

本特集の最初でも説明しているとおり、FHSに対応することで、WebサーバのApacheがHTMLファイルを置くディレクトリ（DocumentRoot）と、ftpユーザーのホームディレクトリが変更されている（**表4**）。

ApacheのDocumentRootは、設定ファイル/etc/httpd/conf/httpd.confで指定するのだが、Red Hat Linux 6.2やVine 2.1.5など、従来のRPM系ディストリビューションでは、/home/httpd/htmlとなっていた。Kondara 2.0以外の新バージョンでは、これが/varディレクトリ以下に移されている。

また、ftpユーザー、またはanonymousユーザーがFTPサーバにログインしたときに使うディレクトリも/varディレクトリ以下の階層に置かれるようになった。これは、/etc/passwdで指定されている。

これらの変更は、FHS 2.xが、頻繁に書き換えられる可能性が高いデータは/varディレクトリ以下に置くと規定していることを反映したものだ。ディレクトリ構成がこのように変更されたことで、Webサーバやanonymous FTPサーバを運用する場合には、/varディレクトリに置かれるデータの容量が増えることになる。パーティションを細かく分けるユーザーは、/varに割り当てるパーティションを大きめに取るようにしたい。

最後に

今日では、/etcディレクトリ内に膨大な数にのぼる設定ファイルが存在するが、決して恐れることはない。Linuxを使ううえで実際に書き換える必要があるファイルは限られているし、RPM系のディストリビューションにはさまざまな設定ツールが用意されているからだ。しかし、いくら便利なツールがあっても、/etcディレクトリをブラックボックスとして扱わずに、日頃からいろいろな設定ファイルに慣れ親しんでおくことをお勧めしたい。イザというときにエディタひとつあればシステムの設定をできるのがLinuxの良いところなのだ。

Linux Wizardへの道は長く、厳しいけれど、勇気を出してそこを目指せば未来はきっと開けるに違いない。

皆さんの健闘を祈る。

	FTPサーバ	メールサーバ
Vine 2.1.5	ProFTPD（S）	Postfix
Red Hat 7.1	wu-ftpd（X）	sendmail
LASER5 7.1	wu-ftpd（X）	sendmail
Kondara 2.0	ProFTPD（X）	Postfix
Turbo 7 WS	ProFTPD（S）	sendmail

表3　FTPサーバとメールサーバ
※FTPサーバの（S）はスタンドアロン、（X）はxinetdからの起動を表す。

	DocumentRoot	ftpユーザーディレクトリ
Vine 2.1.5	/home/httpd/html	/home/ftp/
Red Hat 7.1	/var/www/html	/var/ftp/
LASER5 7.1	/var/www/html	/var/ftp/
Kondara 2.0	/home/httpd/html	/var/ftp/
Turbo 7 WS	/var/www/html	/var/ftpd/

表4　ApacheのDocumentRootとftpユーザーディレクトリ

Linuxのスタンダード GUI環境を使いこなすためのHints & Tips

GNOME デスクトップ 自由自在

文：編集部
Text : Linux magazine

先月号のKDEに続くデスクトップ特集第2弾として、今回はGNOMEを取り上げる。Linuxのスタンダードな統合デスクトップ環境として、多くのディストリビューションで標準採用されているGNOMEを使いこなせるようになれば、あなたも立派な「Linux GUIマスター」になれる。

初級Linuxerはもちろん、すでにLinuxに習熟した方も「Linuxはコマンド命でしょ」とか、「デスクトップはやっぱWindows」なんて言ってないで、スキルの幅を広げるためにも、ひとつお付き合いくださいませ。

頭の整理〜予備知識を少々

実際の設定方法や使い方の説明に入る前に、知っておきたい関連情報を整理しておこう。

■統合デスクトップ環境って何？

GNOMEは、Linuxを始めとするUNIX系OSで利用できる統合デスクトップ環境として、KDEと並び高い評価を得ている。では、そもそも「統合デスクトップ環境」とは何なのだろうか。

ごくごく簡単に定義すると、統一された外観と操作性を備えたグラフィカルなユーザーインターフェイスを持つ、アプリケーションやツール、そしてデスクトップ（ワークスペース）を提供する操作系もしくは、そのための枠組みだといえる。

具体的には、アプリケーション間で統一されたメニュー／ダイアログ／ボタン、ファイル（データファイル、実行ファイル）やそれへのリンクを自由に配置できるデスクトップ、カスタマイズ性の高いシステムメニューやアプリーションランチャなどを備え、マウスあるいはキーボードショートカットによるコピー／カット＆ペースト、ドラック＆ドロップによるファイルコピーやアプリケーション起動などの操作を実現する。

こうした環境はMac OSやWindowsにおいてかなりのレベルで実現されており、これらのOSが幅広い層の多くのユーザーを獲得していることからわかるように、そのOSに関する知識や習熟度にかかわらず直感的な操作が可能で、ユーザーの利便性を大きく向上させることに成功している。Linuxでも、GNOMEとKDEの進化によって、入門者に対する間口は確実に広がってきている。

■ウィジェットとツールキット

前述したような、統一された外観を持つダイアログ、ボタンなどのGUI部品は「ウィジェット」と呼ばれている。そして、さまざまな種類のウィジェットとそれらをプログラムから利用するためのインターフェイスを提供するのが「ツールキット」と呼ばれるライブラリ集である。

GNOMEは、GTK＋というツールキットをベースにプログラムが開発されている。ちなみに、GTK+の「G」はGNOMEのGではなく、GIMPのGである。GTKはもともとGIMP用に開発されたツールキットなのだ。

■アプリケーションフレームワーク

「GNOMEに準拠した」アプリケーションは、GTK＋をベースに開発されるというだけでなく、あらかじめ決められた枠組み（フレームワーク）に従っていなければならない。

たとえば、メニューの構成やショートカットキーの定義といったデザインと操作性にかかわるものから、ほかのアプリケーションとデータ（オブジェクト）をやり取りするための仕組みといった、よりテクニカルなものまで、各レベルにおいてフレームワークが規定されている。

●

さて基礎知識が整理できたところで、次ページからは、いよいよ設定とカスタマイズの方法について詳しく解説していくことにしよう。

Column

GNOME豆知識

あんまり役には立たないけれど、薀蓄を傾けるためには欠かせない豆知識をちょこっと紹介しておこう。

●

まず知っておきたいのが、GNOMEとは「GNU Network Object Model Environment」の頭文字をとったものであるということ。薀蓄の第一歩である。

次にGNOMEをどう読むかという問題。日本では「ぐのーむ」という読み方が一般化していて、編集部でもこの読み方が定着している。GNOME ProjectのWebサイト（http://www.gnome.org/）で確認したところ、これで正解であるようだ。通常、英語ではこの綴りで「G」を発音しないと思うのだが、英語圏の一般ピープルはどう発音しているのだろうか？　知りたいところである。

本来のgnomeという単語には「大地の精」とか「地の精霊」という意味がある。ロールプレイングゲームやファンタジー小説に興味のある方なら、日本語表記で「ノーム」として登場するキャラクターとしておなじみだろう。老人の容貌に小柄な体という愛らしい外見を持ち、鉱山などの地中で宝探しをしているとされる。メニューパネルに表示される、おなじみの足跡マークもこの精霊の足跡なのだそうだ。

gnomeの語源はギリシャ語の「gnosis」で、これは知恵を意味する言葉だ。また、gnomeには「金言」という意味もあるそうだ。なかなか、深みのある単語なのである。

GNOMEカスタマイズTips!

ひと口に「GNOMEのカスタマイズ」といっても、デスクトップの外観から、個々のアプリケーションのオプション機能に至るまで、あらゆるレベルにおいてそれが可能であるため、すべてを網羅することはできない。ここでは、ユーザーインターフェイスの基本となるデスクトップとパネルを中心に、なるべく多くの項目について設定方法を紹介しつつ、GNOMEデスクトップのインターフェイスを使いこなす「コツ」のようなものを伝えられるようにしたい。また、新しいGNOMEの目玉であるNautilusについても解説する。

なお、以降の記述はGNOME 1.4をベースにしている。また、動作チェックは製品版のTurbolinux 7 Workstationで行った。

GNOMEデスクトップ環境のコンポーネント

ページ数もそう多くはないので、さっそくにでも各設定の説明に移りたいところだが、まずはGNOMEデスクトップ環境を構成している各コンポーネントについて簡単にまとめておこう。

■パネル

GNOMEのデスクトップ環境で作業する際のメインとなるユーザーインターフェイス。パネル上のアイコンやボタンは、メニュー、アプリケーションランチャー、アプレットのユーザーインターフェイスになっていて、それぞれ固有の機能を持っている。

■デスクトップ

言葉で定義するのは難しいのだが、要は画面上でウィンドウやパネルの背景になっている部分だ。もちろん単なる背景ではなく、その名のとおり、ファイル、フォルダ、メニューアイテムなど、任意のオブジェクトを置くことができるという作業領域としての役割を持っている。

複数の画面にまたがる大きなデスクトップを設定することも可能で、たとえばTurbolinux 7では、4画面分の大きさのデスクトップが標準で設定されている。1画面に表示できるデスクトップ領域のことを「ワークスペース」と表現することもある。

画面1　標準的な構成のGNOMEデスクトップ

■メインメニュー

パネルにある足跡のアイコンをクリックしたときに表示されるメニュー。このメニューはメニューパネルにも表示される。

■アプレット

パネルで動作する小さなプログラム。時計、タスクスイッチ、ボリュームメーター、システムモニタなど、さまざまな機能を持ったものが数多くある。

コントロールセンターのおさらい

GNOMEコントロールセンター（以下、コントロールセンター）は、GNOMEをカスタマイズする際に欠かすことのできないツールだ。GNOMEを使っていてまったく触ったことがないという方も少ないだろうが、念のためここで操作方法などをおさらいしておこう。

■起動方法

コントロールセンターは、メインパネルにある「ツールボックス」のアイコン（**画面1**のメインパネルの左から3つ目のアイコン）をクリックして起動する。また、メインメニューの［プログラム］－［デスクトップ設定］－

［GNOMEコントロールセンター］からも起動できる。root権限は必要なく、一般ユーザーでも実行可能だ。

■インターフェイス

コントロールセンターのウィンドウは左右に分割されており、左側のペインに設定項目のグループがツリー表示され、選択されたグループの設定ダイアログが右側に表示される（**画面2**）。ダイアログには、［試行］、［戻す］、［OK］、［キャンセル］の4つのボタンがある。

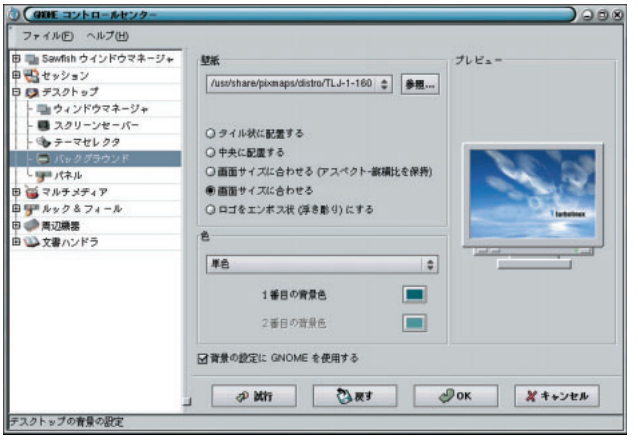
■基本操作

デスクトップカスタマイズの第一歩として背景の変更を行いながら、コントローラの基本的な操作を確認しよう。ウィンドウの左側にあるツリーの［デスクトップ］を展開し、［バックグラウンド］を選択する。

デスクトップに画像ファイルを貼り付けたい場合、右側のダイアログの［壁紙］セクションにある［参照］ボタンを押して画像ファイルを指定する。ディストリビューションによって異なるが、壁紙用のファイルは/usr/share/wallpapers/にいくつか用意されているはずだ。

［プレビュー］セクションに表示される情報では、実際に貼り付けたときの感じが掴みづらければ、［試行］ボタンを押して現在の設定を一時的に反映させるとよい。いくつかの設定を試していき、気に入ったものがあれば［OK］ボタンで変更を確定し、なければ［キャンセル］でダイアログを抜けられる。

あれこれ変更しすぎて元の設定がわからなくなった場合には、［戻す］ボタンを使おう。そのダイアログを変更前の状態に戻すことができる。［キャンセル］ボタンと違いダイアログが閉じられないので、ワンクリックだけ手間を省くことができるわけだ。



画面2　GNOMEコントロールセンター
このツールの操作法をマスターすることが、カスタマイズの基本中の基本である。あれこれ設定を試して、その効果を確かめてみよう。

すばやくプログラムを起動する

GNOMEに初めて触れる人は、メニューに登録されているアイテムの多さに面喰ってしまうのではないだろうか。

WindowsやMacなどの商用OSと異なり、Linuxのディストリビューションには、数多くのフリーソフトがそれこそ山のように収録されている。そのため、初期状態でもメニューアイテムの数が多くなってしまうのはいたしかたないのだが、これでは目的のプログラムにたどり着くだけでひと苦労する。

ユーザーが常用するアプリケーションはある程度限られているので、使いやすい状態にメインメニューを整理したいところだ。使用頻度の高いアプリケーションやユーティリティのメニューアイテムを、メニュー階層の浅いところにまとめておくと使い勝手が良くなるはずだ。

より手軽な方法として、パネルの持つアプリケーションランチャとしての機能を利用する手もある。また、デスクトップにアプリケーションへのリンクを作成するという方法もある。

■メニューを整理する

あとで必要になることもあるだろうから、すでに登録されているメニューアイテムを削除しない方法を紹介しよう。メインメニューに標準で登録されている「お気に入り」を使うのがポイントである。このサブメニューは、一般ユーザー権限で設定変更できるという

Column

マウスを使わずにプログラムを起動する

本文中で紹介したコントロールセンターの起動方法は、一般的なプログラムの起動方法でもある。いずれもマウスを使うものだが、マウスを使わずにプログラムを起動することもできる。

Alt+F2キーを押すと［プログラムの起動］ダイアログがオープンする（デフォルト設定時）。これはメインメニューの［実行...］で開かれるのと同じものだ。リストボックスに実行ファイル名を入力しEnterキーを押せば、プログラムが起動する。ちなみに、コントロールセンターの実行コマンドは「gnomecc」だ。

［プログラムの起動］ダイアログ

う利点があるのだ。

登録方法はいたって簡単で、メニューアイテムを右クリックし、表示されるポップアップメニューで［これを‘お気に入りメニュー’へ追加］を選択すればいいだけだ（**画面3**）。この手順を繰り返して、よく使うアプリケーションを「お気に入り」にすべて登録しよう。

念のためGNOMEコントロールセンターの［デスクトップ］セクションにある［パネル］の［メニュー］タブで、「お気に入り」メニューの［サブメニューに］がチェックされているか確認する。これで、メインメニューの「お気に入り」に、文字通りの役割が与えられたわけだ。

■パネルをランチャとして使う

パネルは、登録されたプログラムを起動するアプリケーションランチャの機能を持っている。先ほどコントロールセンターの起動に使ったアイコンがランチャの一例だ。

パネルにランチャを追加する方法はいくつかあるが、メニューアイテムをドラッグしてパネルにドロップするのが直感的でお手軽だ。ドロップしたメニューアイテムはアイコンとしてパネル上に表示される。

ランチャは、シングルクリックでプログラムを起動でき、メニューをたどっていく手間も省けるため、いくつかあるプログラム起動方法の中でも最高にシンプルだ。使用頻度の高いアプリケーションやユーティリティを選んで登録しておくと、とても便利である。

パネルからランチャを削除するには、アイコンを右クリックしポップアップメニューから［パネルから取り除く］を選択すればよい。この削除手順は、ランチャだけでなくメニューやアプレットの場合も同じだ。

■デスクトップにリンクを置く

プログラムの実行ファイルのリンクやメニューアイテムをデスクトップに配置しておけば、そのアイコンをダブルクリックすることでプログラムを起動できるようになる（**画面4**）。どちらもマウスを使って簡単に配置することが可能だ。

実行ファイルのリンクを配置するには、Nautilusなどのファイルマネージャから右ボタンでそのファイルのアイコンをドラッグ＆ドロップする方法が安全・確実である。この方法では、アイコンをデスクトップにドロップする際にポップアップメニューが表示され、［Link here］を選択することでリンクが作成できる。このため、ファイルそのものをデスクトップに置いてしまうといった誤操作の可能性がグッと減るのだ。

メニューアイテムは、ランチャアプレットのパネルへの追加と同じ手順でデスクトップに配置できる。メニューから、アイテムを普通にドラッグ＆ドロップするだけでいい。

●

プログラムの起動には、もうひとつデータファイルのアイコンをダブルクリックする方法がある。それには、特定のファイル形式とアプリケーションを関連づけるための設定が必要となる。詳しくは次項で解説しよう。

MIME型の設定

MIME型とは、先ほどもちょっと触れたように「ファイルの種類」と「処理を行うアプリケーション」とを関連づけたもの。設定は、コントロールセンターの［文書ハンドラ］セクションにある［ファイルの種類とプログラム］で行う。ここでの設定は、ファイルマネージャ（Nautilus）やデスクトップに反映され、特定種類のファイルアイコンをダブルクリックすると、指定されたアプリケーションが自動的に起動し、そのファイルがオープンされるようになる。

画面3　「お気に入り」へのアイテムの追加

画面4　デスクトップからプログラムを起動
Galeonのメニューアイテムと GIMP の実行ファイルへのリンクをデスクトップに配置した。どちらもダブルクリックすることでプログラムを起動できる。

これまでの方法に比べると、プログラムの起動直後にデータが読み込まれているという利点がある反面、少し設定が面倒になる。

画面5はプレーンテキストのMIME型編集ウィンドウを示している。デフォルトの設定では、拡張子が“.asc”、“txt”、“TXT”であるファイルのアイコンをダブルクリックするとNautilusのテキストビューにファイルの内容が表示されるようになっている。**画面5**のように設定変更すれば、ダブルクリック時にgEditが起動する。

それぞれのMIME型ごとにひとつずつ設定を行う必要があるので手間はかかるが、うまく設定すればデスクトップの使い勝手を大幅に向上させられる。上記のようにデフォルトの設定を見直して、使い慣れたアプリケーションが起動するように変更するだけで、ずいぶん違うはずだ。

パネルのカスタマイズ

パネルは、大きさや位置、色を変えるといったカスタマイズが可能だ。また、アプレットを追加することで、メニューとランチャ以外の目的でも使うことができる。

■パネルの種類

パネルには以下の4種類があり、長さと配置できる位置が微妙に異なっている。

・エッジパネル
デスクトップの幅いっぱいに表示される最も一般的なパネル。長さの変更はできない。上下左右の縁に沿って配置できる。

・アラインパネル
登録されているランチャやアプレットが表示できるよう自動的に長さが切り替わるパネル。上下左右の縁に沿って配置できるが、縁の両端または中央のみに位置が限られる。

・スライドパネル
アラインパネルとほぼ同じだが、縁に沿って自由に位置を変えられる（つまりスライドできる）点が異なる。

・フロートパネル
スライドパネルよりさらに自由度が高く、デスクトップの任意の場所に置くことができる。

・メニューパネル
デスクトップの上辺に沿って、幅いっぱいに表示されるパネル。メインメニューの内容がパネル上に表示される。長さ、位置とも変更不可。このパネルは、デスクトップ上に1つだけ作成できる。

■パネルのプロパティを変える

メニューパネル以外のパネルは、種類、大きさ、色などの属性（プロパティ）を自由に変更できる。メインメニューから、［パネル］－［プロパティ］とたどっていくとプロパティを変更するためのメニューが表示される。たとえば幅を変えるには、このメニューで［大きさ］を選択して表示されるサイズから好きなものを選べばよい。

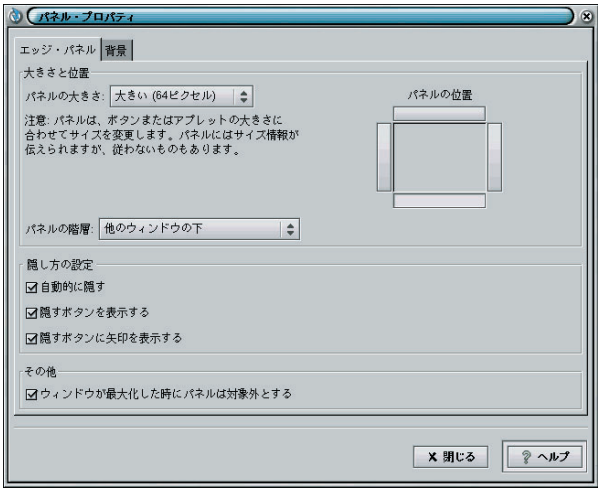
［種類］プロパティの細かな設定は、このメニューの［全てのプロパティ］で表示されるダイアログで行う（**画面6**）。

個々のパネルではなくパネル全般のプロパティを変更するには、コントロールセンターを使う。［デスクトップ］セッションの［パネル］を選択すると、パネルに関する数多くの設定項目が並んだダイアログが表示される。各項目は、具体的に文章で説明されているので特に難しいものはないはずだ。

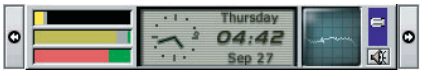
■パネルの追加と削除

メインメニューから［パネル］－［新規パネルの作成］とたどっていき、種類を選択するとデスクトップにパネルが追加される。パネルの削除は、メ

画面5　MIMEの設定ダイアログ



画面6　パネルのプロパティ設定ダイアログ



画面7　パネルに登録したアプレット
左から順に「CPU／メモリ使用状況メーター」、「AfterStepクロック」、「サウンドモニタ」、「ミキサー」の各アプレット。

インメニューの［パネル］にある［このパネルを削除］で行う。デスクトップ環境には最低でも1つはパネルが必要なため、パネルが1つしかないときは、このメニューアイテムは使用できない。

■パネルにアプレットを追加する

アプレットには、時計やタスクリストのように結構使えるものがある（**画面7**）。自分にとって有益なものを選んで追加していこう。

アプレットの追加は、メインメニューの［パネル］－［パネルに追加］－［アプレット］で行う。カテゴリー別にアプレットが表示されるので、追加したいアプレットのアイコンをクリックすればOKだ。

デスクトップのテーマ

GNOMEでは「テーマ機能」を使って、全体の統一感を損なうことなくユーザーインターフェイスの外観を変更することができる（**画面8**）。操作性や機能性には直接結びつくものではないが、用途によっては長時間にわたって作業を行うこともあるだろうから、ある程度は見た目にも気を配っておきたいところだ。

テーマ機能で変更できるのは、GTK＋のウィジェット（ダイアログ、ボタン、リストボックスなど）とウィンドウフレームの外観、それらで使われるフォントなどだ。ウィンドウフレームについては、使用しているウィンドウマネージャがテーマ機能を持っていることが条件となる。ここでは、最もポピュラーなSawfishを例に設定方法を説明する。

■テーマを変更する

まずGTK＋のテーマを変更してみよう。コントロールセンターの［デスクトップ］セクションに［テーマセレクタ］という項目がある。この項目を選択し、右側のダイアログにある［利用可能なテーマ］という一覧からテーマ名を選ぶと、下段にある［プレビュー］に表示されているウィジェットのサンプルが選択したテーマのものに切り替わる。

フォントを変更するには、［カスタムフォントを使用する］のチェックボタンをオンにしてフォント名が表示されたボタンを押す。フォントセレクタでフォントを選択して［了解］ボタンを押せば完了だ。

このダイアログでは、Themes.org（http://www.themes.org/）などで入手したGTK＋のテーマをインストールすることもできる。手順は簡単で、［新規テーマのインストール］をクリックしてテーマのアーカイブファイルを指定するだけで新しいテーマが一覧に追加される。

次にSawfishのテーマだが、これもコントロールセンターで設定できる。［Sawfishウィンドウマネージャ］セクションから［外観］を選択して、右側のダイアログの［標準ウィンドウフレーム］と［標準フォント］を指定する。こちらは、プレビュー機能がないので［試行］ボタンを使って効果を確認しよう。

Sawfishテーマの追加は、コントロールセンターからはできない。ただし、手順自体はやはり簡単で、ホームディレクトリの.sawfishディレクトリにthemesディレクトリを作成して、ここにテーマのアーカイブを展開すればよい。

Nautilusが来た！

Nautilusは、GMCに代わる次世代のファイルマネージャとして期待される

画面8　テーマの変更
GIMPをデフォルトのテーマ（左）から、MacOS風のテーマに変更。注意して見るとメニューのフォントも変わっていることに気づくはずだ。

画面9　Nautilusのメインウィンドウ
シンプルな2ペイン構成に見えるが、左側のペインの下部にあるタブをクリックすることで、ディレクトリツリーやニュースチャンネル、履歴などを表示するように切り替えることができる。右ペインの画像ファイルのサムネイルにも注目。

GNOMEのニューカマーだ。開発元のEazelが業務停止状態に陥ったため、先行きが心配されていたが、その後も開発プロジェクトの有志によるメンテナンスが続けられており、バグフィックスとチューニングが施されている。執筆時点の最新バージョンは1.0.4だ。

Nautilusの最大の特徴は、視覚的、聴覚的にファイルを表現する点にある。アイコンを使ってファイルの種類を判別できるようにすることはGMCでも可能だったが、Nautilusは画像ファイルのサムネイル表示、テキストファイルのプレビュー（テキストファイルのアイコンに内容が表示される）、MP3ファイルの自動再生（マウスポインタをアイコン上にポイントすると自動的にそのファイルが再生される）といった機能を備えている。

もうひとつの特徴は、デスクトップやNautilusで使われるフォントのアンチエイリアシング、洗練されたユーザーインターフェイスのデザイン（テーマ機能も備えている）など、見た目の美しさにこだわった造りになっている点だ。「ビジュアル系」ファイルマネージャと呼びたいような趣なのである。

そのほかにも、インデックス化による高速検索、Mozillaのコアを利用したWebブラウザ機能、サイドパネルへのネットニュースチャンネルの表示、ハードウェア情報の表示といったGMCにはない機能が盛り込まれている。

■Nautilusを高速化する

上記のように非常に多機能なNautilusだが、スムーズに動作させるには相当のマシンパワーが必要だ。今回のテストでは、動作チェックにDuron 600MHz、メモリ128Mバイトというマシンを使用したが、ときには起動に30秒近くかかることがあった。画像ファイルの多いディレクトリを表示する場合など、サムネイルを作成するためだろうか、いつまでもハードディスクアクセスが終わらないといったこともあった。

人によっては、サムネイル表示やアンチエイリアシングは「余計」な機能で、それよりもスムーズに動作してほしいという意見もあるだろう。そこで、Nautilusの設定を変更することで、なるべく軽快に動作するようカスタマイズする方法を探ってみることにしよう。

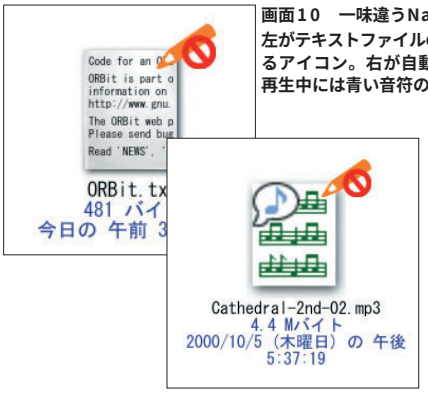
Nautilusの設定はユーザーのスキルに合わせて「初級」、「中級」、「上級」の3つのレベルに切り替えられる。より細かく設定するには、メニューバーの［設定］にある［上級者］をチェックするのを忘れないようにしたい。

設定はメニューバーの［設定］－［設定の編集］で行う。**画面11**のようなダイアログが表示されるはずだ。高速化の手始めとして、まずはアンチエイリアシングをオフにしてみよう。左の一覧から［外観］を選択し、右側のダイアログで［画像を滑らかにする（遅い）］チェックボックスをオフにする。これで、アイコンとフォントのアンチエイリアスが解除される。わざわざ設定項目に“（遅い）”と書いてあるくらいなので、かなりの効果が見込めるはずだ。

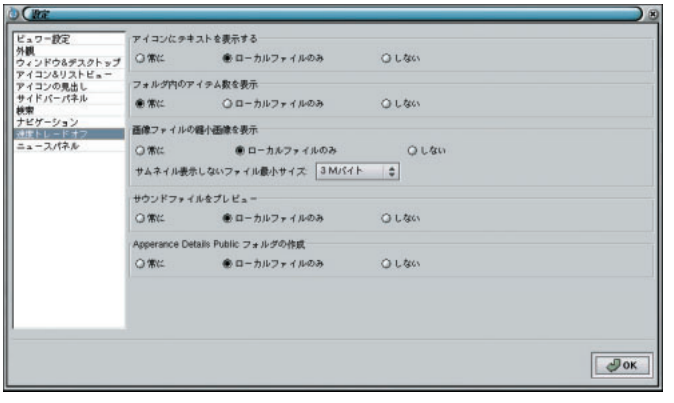
次にそのものズバリの［速度トレードオフ］を左の一覧から選択する。ダイアログに動作速度に関連する設定が表示される。これらの設定項目は、すべて［しない］に設定する。これで画像ファイルのサムネイル表示やMP3ファイルの自動再生などの「いかにも重そう」な機能がオフになる。

最後に、［サイドバーパネル］にある［ツリーにフォルダだけ（ファイルなし）を表示する］をチェックする。これでツリー表示がいくぶん速くなるはずだ。ついでに［アイコン&アンドリストビュー］にある［シングルクリックでアイテムを起動する］を選択しておけば、クリック回数の分だけ高速化される（のか？）。

実際にこの設定を試したところ、体感的になスピードは若干速くなった気がした。非力なマシンなら、より効果が感じられるかもしれないので、Nautilusが重くて動かないという方は試してみてほしい。



画面10　一味違うNautilusのアイコン
左がテキストファイルの内容をプレビューしているアイコン。右が自動再生中のMP3ファイル。再生中には青い音符のマークが表示される。



画面11　Nautilusの設定ダイアログ
この［速度トレードオフ］ダイアログでは、機能性と処理速度の優先度を設定できる。［常に］を選ぶと機能性が、［しない］を選ぶと処理効率が高くなる。

GNOMEアプリケーションガイド

先月のKDE特集でお伝えしたように、KDEのオフィススイートであるKOfficeは、アプリケーション単体としての機能はもちろん、アプリケーション間でのデータの連携もかなり進んできている。GNOMEにも、KOfficeに相当する「GNOME Office」というオフィスアプリケーションのスイートが用意されているが、その実力はいかなるものなのか？　ここでは、GNOME Officeに含まれるものを中心にGNOMEで利用できるアプリケーションを紹介していくことにしよう。

GNOME Office構想

GNOME ProjectのWebサイトにあるGNOME Officeの情報ページ（http://www.gnome.org/gnome-office/）に掲載されているリストをまとめたものが**表1**である。この表からもわかるように、1つのカテゴリに複数のアプリケーションがあるなど、通常のオフィススイートとはちょっと違っている点がある。これは、GNOMEのアプリケーションフレームワークに従っていて、水準以上の機能性を有しているソフトウェアをGNOME Officeとして随時追加する方針をとっているためである。もちろん、GPLまたはそれに準じるオープンソースソフトウェアライセンスの下で配布されるフリーソフトウェアであることが求められる。

どのアプリケーションをGNOME Officeに加えるのか、という議論はgnome-office-listというメーリングリストで行われている（登録用WebサイトのURLは、http://mail.gnome.org/mailman/listinfo/gnome-office-list/）。

GNOME Officeの各アプリケーションの開発は単独のプロジェクトとして進められている。また、スイートに含められる最終的なアプリケーションの数も明確には定められていない。

GNOME Officeの構想全体に大きな影響を与えそうなのが、OpenOfficeの存在だ。OpenOfficeは、米国のSun MicrosystemsがStarOfficeのオープンソースバージョンである。現在は、Sun Microsystemsが公開したソースコードを元に、OpenOffice.org（http://www.openoffice.org/）で開発が進められている。ライセンスについては、LGPL（GNU Lesser General Public License）とSISSL（Sun Industry Standards Source License）のデュアルライセンス方式がとられている。

OpenOfficeの各アプリケーションは将来的には、GNOME Officeに含められる予定だ。すでに、アプリケーション間でのデータ連携とMicrosoft Officeのデータの読み込みを、かなりのレベルで実現しており、予定通りにいけばGNOME Officeの中核となる可能性もある。OpenOfficeは、以下のアプリケーションで構成されている。

- **Writer**　ワープロ
- **Calc**　表計算
- **Impress**　プレゼンテーション
- **Draw**　図形描画
- **Chart**　グラフ作成
- **Formula**　数式エディタ

カテゴリー	アプリケーション	開発プロジェクトのWebサイト
ワープロ	Abiword	http://www.abisource.com/
表計算	Gnumeric	http://www.gnome.org/projects/gnumeric/
プレゼンテーション	Achtung	—
ラスター描画ツール	Gimp	http://www.gimp.org/
ベクター描画ツール	Sodipodi	http://sodipodi.sourceforge.net/
	Sketch	http://sketch.sourceforge.net/
図形作成	Dia	http://www.lysator.liu.se/~alla/dia/
グラフ作成	GUPPI	http://www.gnome.org/guppi/
イメージビューア	Eye of GNOME	—
メーラ	Balsa	http://www.balsa.net/
	Evolution	http://www.ximian.com/products/ximian_evolution/
Webブラウザ	Galeon	http://galeon.sourceforge.net/
プロジェクト管理	MrProject	http://mrproject.codefactory.se/
	Toutdoux	http://www.toutdoux.org/
データベースアクセス	GNOME-DB	http://www.gnome-db.org/
ファイナンス	GnuCash	http://www.gnucash.org/

表1　GNOME Officeに含まれるアプリケーション

ワープロ・表計算・プレゼン

オフィススイートの3大アプリケーションといえば、ワープロ、表計算、プレゼンテーションソフトである（独断）。GNOME Officeでは、AbiWord、Gnumeric、Achtungが用意されている。

それぞれについて見ていこう。

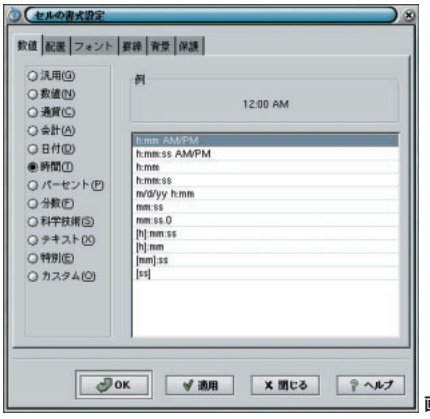
■AbiWord

名前のごとくワープロソフトである。フォントやテキスト形式の指定、インデント、段組み、画像ファイルの貼り付けといったワープロソフトとしての標準的な文書作成機能をはじめ、検索・置換、スペルチェック、無制限アンドゥなどの補助機能を備えている。決して多機能ではないが、Microsoft Word（97／2000）、RTF、XHTMLといったファイル形式もサポートしており、必要な機能はひと通り揃っていて、シンプルなぶん動作はとても軽快である。

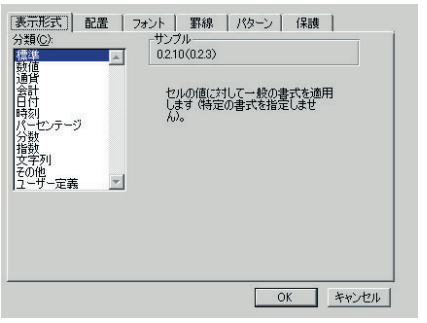
最新バージョンの0.9.2でも日本語には未対応で、Turbolinux 7で採用されているバージョン0.7.10を使って確認した範囲内でも日本語の表示・入力はできなかった。残念ながら、日本語ワープロとしてはまったく使えないというのが現状だ。

■Gnumeric

GNOMEネイティブの高機能な表計算ソフト。開発プロジェクトのWebサイトの記載によれば、Excelの95パーセントの機能を備えているそうだ（どのバージョンのExcelかは不明）。実際にメニューの構成やツールバーから利用できる機能などは、ほぼExcelと同等なものが用意されている。たとえば、**画面3**はGnumericのセルの書式設定ダイアログだが、**画面4**のExcel 97に近い設定項目が並んでいることがわかる。



画面3　Gnumericのセル設定ダイアログ



画面4　Excelのセル設定ダイアログ

動作チェックした範囲内では（バージョン0.64）、AbiWordと異なり、こちらは日本語の表示・入力に問題はなかった。

ただし、Excel 97のファイルを読み込んだ際に文字化けが発生した。英語のみのExcelファイルはきちんと読み込めたことからも、これは単に文字コードの問題であると思われる。また、テキストファイルのインポートでは、文字コードに関係なく日本語を含むデータの場合には、どれもうまく動作しなかった。

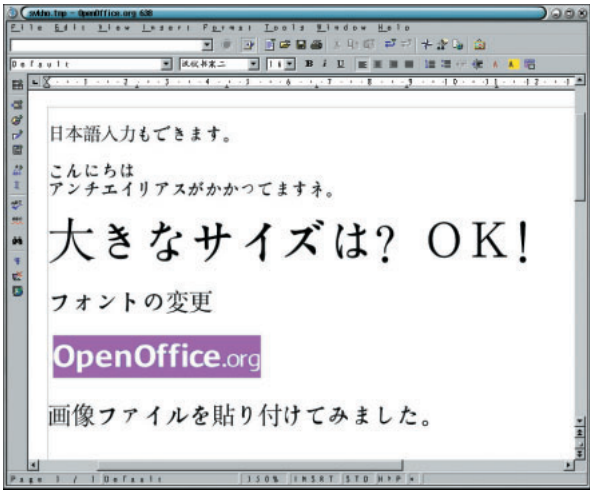
■Achtung

見慣れない単語だが、英語の「Attention」にあたるドイツ語であるようだ。プレゼンテーションの際に注意を引き付けるという意味で命名されたのだろう。

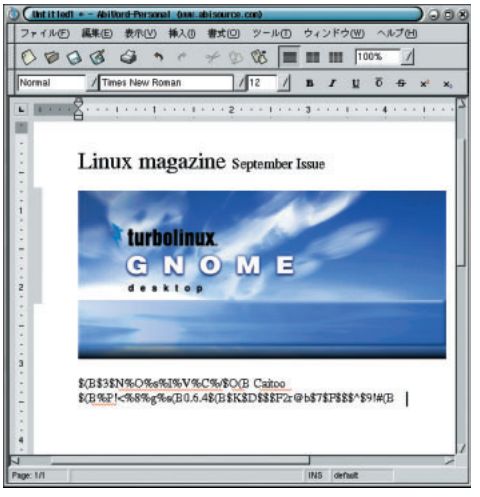
現在のところ入手先もGNOMEのCVSしかなく、プロジェクトのWebサイトも開設されておらず、プレゼンテーションソフトであること以外、詳細は不明である。なお、開発者向けのメーリングリストは設けられており、XimianのWebサイトに登録ページ（http://lists.ximian.com/mailman/listinfo/achtung/）が用意されている。

グラフィックス処理

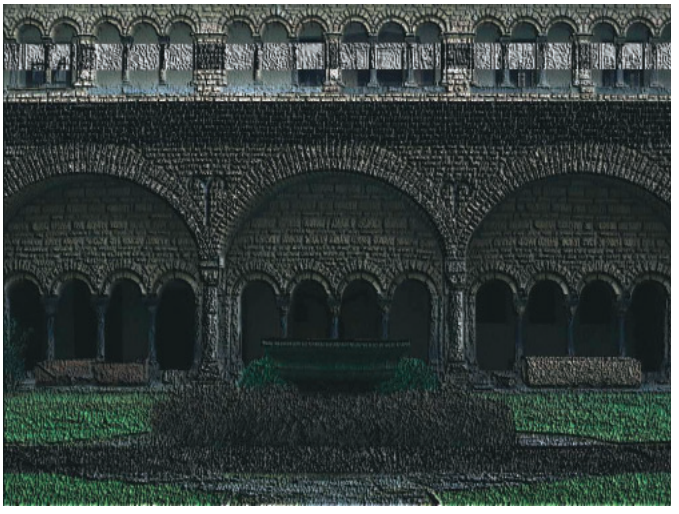
GNOMEで動作する最も有名なアプ



画面1　OpenOffice Writer
OpenOfficeのワープロソフトWriterのメインウィンドウ。フォントの変更ができないなど、完全ではないが日本語処理をサポートしている。



画面2　AbiWord
シンプルで軽快なワープロソフト。画面下部にあるように日本語は文字化けしてしまう。



画面5　GIMPで加工したフォトデータ
写真画像をGIMPの標準フィルタを使って絵画風に加工したもの。このほかにも、色調補正や特殊効果のための各種フィルタが標準で用意されている。

リケーションソフトであるGIMPを始め、GNOME Officeには多くのグラフィックス処理ソフトが含まれている。主なものをいくつか紹介しよう。

■GIMP

フォトレタッチに、ホームページの素材作りにと大活躍の画像処理ツール。非常に多くの機能を備えており、Macintoshの定番ソフトに匹敵する能力を持つことから「PhotoShopキラー」と呼ばれることもある。

■Sodipodi、Sketch

どちらも標準的な機能を備えたドローソフトだ。基本となる描画機能をはじめ、W3Cが標準化を進めているSVG（Scalable Vector Graphics）のサポート、PNG形式でのエクスポートなど、ほぼ同等の機能を持っている。ユーザーインターフェイスに違いがあるので、実際に試してみて自分に合ったほうを選べばOKだ。

現行バージョンでは、両者とも日本語テキストの表示・入力には対応していない。

■Dia

こちらもドローソフトだが、回路図やネットワーク構成図、フローチャートといった図表の作成に特化した機能が強化されている。図表の種類ごとに部品が用意されており、直線やテキストなどと組み合わせることで手軽に図表を作成できるようになっているのだ。作成した図を部品として登録することも可能だ。

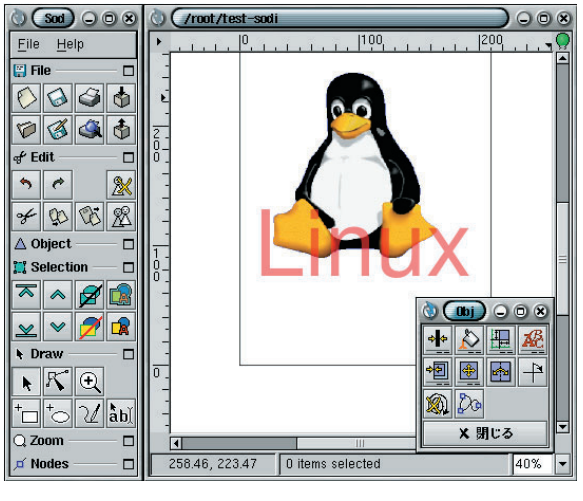
ファイルの保存はXMLベースの独自形式で行われるが、EPSおよびSVG形式へのエクスポートもサポートしている。なお動作チェックした範囲内では、一応日本語の表示・入力が可能であるものの、フォントサイズの変更を行うと文字化けするなどの問題もあった。

●

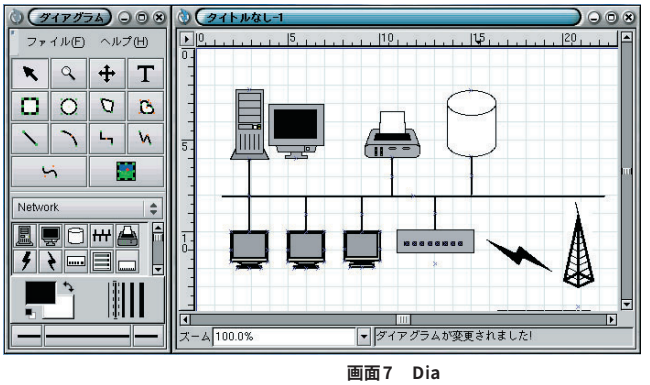
ここで、グラフィックス系では使用頻度が高いと思われるイメージビューアについて少し触れておきたい。

GNOME OfficeにはEye of GNOMEというイメージビューアが含まれているが、今回動作チェックを兼ねて少し操作してみた限りでは、機能も少なくいまひとつ使い勝手も良くなかった。全体的な印象として、アプリケーションの完成度はそれほど高くないと感じた。今後開発が進んでいくうちに機能アップすることを期待したい。

ということで、現時点ではほかのイメージビューアのほうがお勧めだ。エレクトリック・アイズ2やGQviewなどがGNOMEで利用できる「定番モノ」である。また、Turbolinux 7に収録されているgThumbも、なかなか使い勝手の良いビューアソフトだ。



画面6　Sodipodi
画面のTux君のように、GIFやビットマップなどのラスタ画像をインポートして加工することもできる。



画面7　Dia
標準で用意されている部品を使って簡単なネットワーク構成図を作ってみた。もちろん、より複雑な図表を作成することが可能だ。

インターネット関連ソフト

GNOME Officeには、Webブラウザとメールクライアントも用意されている。いずれも、多彩な機能を誇る実力派ぞろいだ。

■Galeon

本誌9月号のWebブラウザ特集においてLinuxのベストブラウザ（編集部独断選定）に輝いたスグレモノである。Webページをブラウズするコア部分はMozillaのコンポーネントを利用して動作する（WindowsのIEコンポーネントブラウザと同じ仕組みだ）。そのため、ブラウジングに関する基本的な機能はMozillaとまったく同等である。日本語の処理にも、まったく問題はない。最新バージョンのGaleon 0.12.1は、Mozillaのマイルストーンビルド0.9.4をベースとしている。

もちろん、オリジナルな機能も多く備えている。そのなかでも最大の目玉となるのが「タブ機能」である。これは、1つのウィンドウで複数のWebページを同時にオープンして、各ページに付いた「タブ」を切り替えながらブラウズできる機能だ。そのほかでは、自動ブックマーク機能や、「スマートブックマーク」と呼ばれるWeb検索用のアプレット、ブックマークに登録された情報を元に動的にポータルサイトのリンク集のようなページを作成する「マイポータル」機能など、ブックマーク関連の機能が充実している。

■Evolution

WindowsやMacでは一般的となった3ペイン構成のメインウィンドウを持つメールクライアントソフト。メーラとしての機能だけでなく、スケジュール管理とTODOリストの機能も備え、パーソナル情報管理ツールとしても使うことができる。まさに、GNOME版Outlookといった趣きのアプリケーションなのである。

起動時に「まだアプリケーションとしては未完成です」という開発チームからのメッセージが表示されるように、現在のバージョンはベータ版の段階である。メーラとしては、POP3とIMAP4をサポートし、マルチアカウント管理やアドレス帳などの標準的な機能を装備している。スケジュール管理、TODOリストについても、ひと通りの機能は実装されており、一応は実用レベルに達しているといえる。

日本語処理もサポートしているが、入力時のカーソルの動きがおかしくなるなど、やや不完全な部分も残っている。メインとなるメールの送受信については、日本語で行うことができた。Windowsのメーラとのメールのやり取りについても特に問題ないようだ。

■Balsa

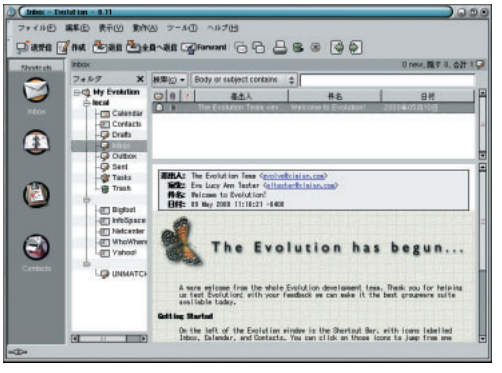
やはり3ペイン構成のオーソドックスなインターフェイスのメールクライアント。Evolutionと異なり、メーラとしての機能に限定した作りになっている。POP3、APOP、IMAP4をサポートし、アドレス帳、メールボックスのスレッド表示、スペルチェッカー、送信時の自動改行、といったメール処理を補助する機能も充実している。

日本語の表示・入力は行えるが、メールの送受信時のエンコード処理に問題があり、ほかのメーラとの日本語を含むメールのやり取りに不具合がある。現時点では、日本語サポートは不十分なようだ。

そのほかのアプリケーション

GNOME Office以外のものでも、GNOMEで利用できる高機能で人気の高いアプリケーションは数多くある。たとえば、グラフィックス処理ソフトのところで触れたエレクトリック・アイズ2やGQviewがそうだし、そのほかにも、gEdit（テキストエディタ）、Sylpheed（国産のメールクライアントソフト）、XMMS（マルチメディアプレーヤ）などがあげられる。

画面8　Galeon
表示されているWebページの上部に付いている「タブ」をクリックしてページを切り替える。この画面では［アスキーホームページ］と表示されたタブが選択されている。



画面9　Evolution
メニューやボタンなどのユーザーインターフェイスは、きちんと日本語化されている。入力まわりなどの細かな部分が修正されれば、日本語サポートも万全だ。

GNOMEアップデートのツボ

GNOMEをソースからコンパイルしてインストールするのは、非常に手間のかかる作業だ。何とかして、お手軽に最新のGNOMEデスクトップ環境を手に入れられないものだろうか？　あれこれ試してみた結果……。

この際なので

まず誰でも思いつくのが、GNOME 1.4を採用したディストリビューションをまるごとインストールするという手である。日本語ディストリビューションでは、Kondara MNU/Linux 2.0、HOLON Linux 3.0、Turbolinux 7 Workstationがバージョン1.4を採用している。つまり、今月号の付録CD-ROMに収録されているTurbolinuxをインストールすれば、GNOME 1.4をゲットできるわけだ。これがGNOME 1.4の環境を構築するための最も確実な方法であることは間違いない。

既存の環境でGNOMEをバージョンアップしたい場合でも、上記のディストリビューションを使っていて、アップグレードインストールが可能なら話は簡単だ。もしそうでなければ、データや設定ファイルなどのバックアップをとってから、ディストリビューションをインストールする方法が考えられる。システム全体の環境を再構築（移行）する必要があるため、やや手間はかかるが、少なくともGNOMEは確実に動作するはず。

問題は、使い慣れたディストリビューションを手放さなければならないという点だろう。細かい部分で設定方法が違っていたり、お気に入りのアプリケーションが使えなかったりといった不具合が出てくる可能性があるからだ。

コンパイルの手間を省く

ディストリビューションを乗り換えるのは、やはり抵抗があるというのであれば、バイナリなりソースコードなりのパッケージを入手してコツコツ地道にひとつずつアップデートしていくしかない。ソースからのコンパイルは時間がかかるので、できればバイナリパッケージを利用したいところだ。

Debianな人なら「aptとそれなりのネットワーク環境があれば依存関係を気にすることなくスイスイとアップデートできる」と言うだろう。では、パッケージ管理にRPM方式を採用しているディストリビューションではどうかというと、これが結構大変なのである。

RPM方式の場合、インストール時に依存関係をチェックしてくれはするものの、何らかの問題があっても、それを自動的に解決することはしてくれない。たとえば、rpmコマンドのメッセージで「パッケージAはパッケージBのインストールに必要とされています」と表示されたら、パッケージAを用意してインストールしてから、改めてパッケージBをインストールする必要がある。つまり、ユーザーが「人間apt」と化して依存関係のもつれた糸を解きほぐしていく必要があるのだ。

GNOMEのアップデートに必要なRPMパッケージの数は、確実に50を超えるはずだ。これらの依存関係を把握して、それを解決しながらインストールしていくのは、想像以上に大変な作業である。コンパイルの手間を省こうとして、かえって手順が複雑化してしまう可能性さえある。

ただし、大幅に手間を省ける裏技的な方法がないわけでもない。rpmコマンドには、依存関係を無視して強制的にインストールを行う“--nodeps”オプションがある。GNOMEのアップデートに必要なパッケージが手元にあれば、以下のコマンドを使って次々にインストールしていけばよいのだ（“gnome-hoge.rpm”の部分はインストールするパッケージのファイル名を適宜指定のこと）。

```
#rpm -U --nodeps gnome-hoge.rpm
```

一見、乱暴で粗雑なように思えるが、これはこれでなかなか有効なのである。その論拠は以下のとおりだ。

必要なパッケージは（ほぼ）揃っている（と思う）し、最終的には全部インストールするのだから、途中で細かい依存関係なんて気にしなくても（たぶん）大丈夫でしょう？

「必要なパッケージが揃っている」という前提がクリアされていれば、理論的には間違ってはいない。また、1つや2つパッケージが欠けていても、GNOMEがまったく起動しなくなるという可能性は思ったより低いはずだ。とはいえ、いくらおおらかな芸風のLinux magazineでも、この方法をお

勧めするものでないことだけは、ここでハッキリとお断りしておく。

さらにRPMパッケージについて

RPM方式には、もう1つの問題がある。複数のディストリビューターがこの方式を採用していて、各ディストリビューション用の複数のrpmパッケージがあるという点だ。ユーザーは、自分のシステム環境に合ったパッケージを見つけ出して入手しなければならない。そのためには、次のような手順が必要となるだろう。

①まず、現在使っているディストリビューションのFTPサイトにアップデートバージョンがあるか確認する（あれば④を進む）。
②見つからない場合は、システム的にソフトウェアコンポーネントの構成が近いディストリビューション（LASER5を使っているならRed Hatがそれにあたるだろう）のFTPサイトを探す（あれば④に進む）。
③やっぱり見つからない場合は、GNOME 1.4を標準採用しているディストリビューションのFTPサイトを探す。あるいは、そのディストリビューションが収録されている雑誌などの付録CD-ROMを入手する（この場合は④は省略）。
④GNOMEのアップデートに必要なパッケージを選択して、すべてダウンロードする。

実は④がかなり骨の折れる作業なのである。“ftp.hogehoge.com/pub/update/GNOME/”といったディレクトリにGNOMEのアップデートに必要なパッケージがまとめられていればラッキーだが、1つのディレクトリにすべてのアップデートパッケージがまとめて置かれていることのほうが多い。その場合、GNOMEのアップデートに必要なファイルを探し出し、すべてが揃っているか確認しなければならない。

先ほども触れたように、GNOMEに関連するパッケージはかなりの数にのぼる。そのすべてをチェックするのは、とてもイヤな作業である。また、依存関係を確認しながらひとつずつインストールする場合にはダウンロードし忘れたものがあっても、あとから別途入手すればよいが、例の「--nodeps作戦」を敢行している場合には致命的になりかねない。不足しているパッケージが増えるほど、それだけ不完全なアップデート状態になってしまうからだ。

奥の手は「Ximian GNOME」

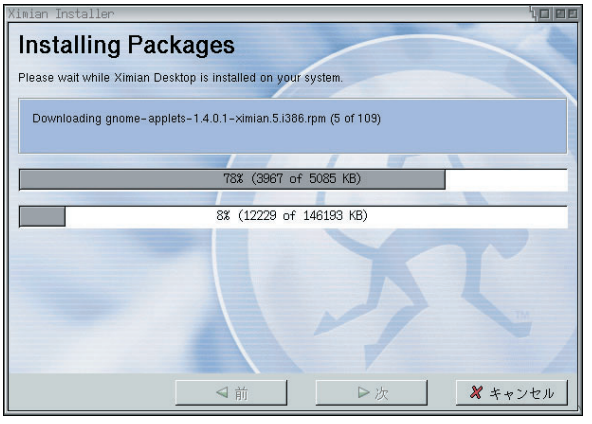
結局のところ、確実なアップデートを行うには、地道な依存性の確認を行っていくしかないのだろうか？　「なるべく手間をかけずにGNOME 1.4にアップデートする」ということだけに目標を絞れば、とても有効な解決策がある。「Ximian GNOME 1.4」へのアップデートを行う方法がそれだ。

Ximian GNOMEは、米国のXimian社によって独自の拡張が加えられたGNOMEパッケージ集だ。その拡張機能のひとつに「Red Carpet」と呼ばれる、ソフトウェアのインストールとアップデートを支援するツールがあり、これを使えばXimian GNOME 1.4へのアップデートをネットワーク経由で非常に簡単に行えるのである。

まず、ローカルで動作するインストーラをダウンロードする必要がある。XimianのWebサイトのダウンロードページ（http://www.ximian.com/download/）で、中央のドロップダウンリストから「Get Ximian Desktop Now!」を選択して「GO」ボタンを押す。表示されるページの下段のほうに「manual installation instructions」というリンクテキストがあるので、それをクリックする。リンク先のページに各ディストリビューションのインストーラが用意されている。該当するものがなければ、Red Hatの適当なバージョンのものをダウンロードしよう。

インストーラを入手したら、以下のコマンドをXのコンソールエミュレータで実行するだけでいい（実行例のファイル名はRed Hat用の場合）。インストーラの指示に従っていけば、ネットワークインストールが開始される。

```
$ gunzip installer-redhat-70-i386.gz
$ chmod +x installer-redhat-70-i386
$ ./installer-redhat-70-i386
```



画面1　専用インストーラを使ったXimian GNOMEのインストール
インストールが始まってしまえば、あとは何もやることがないのだが、とにかく時間がかかる。途中で失敗した場合には、その時点からダウンロードを再開できる（体験談）ので、もう一度全ファイルをダウンロードするという最悪の事態は避けられる。

インストールが開始されたら、あとはひたすら待っていれば、自動的に必要なパッケージがダウンロード・インストールされ、アップデート作業はめでたく完了となる。

以上のように手順は簡単だが、いいことずくめなわけではない。まず、Ximian GNOMEは、独自拡張されているせいもあってパッケージ数が非常に多い。今回行った動作確認では、Red Hat 7をベースにデフォルトの「Normal Install」モードを選択したのだが、この場合の総インストールパッケージ数は109、ファイルサイズの総計は約140Mバイトだった。ネットワーク環境にもよるが、完了までにかなりの時間と接続コストが必要となることは間違いない。

もう1つ、日本語対応の問題がある。GNOMEは、標準で多言語に対応する仕様になっており、メニューやメッセージなどはインストール先のロケール設定によって切り替わるようになっている。日本語のカタログデータも用意されているが、現状では完全ではない。また、アプリケーションによっては日本語サポート用のパッチが必要なものもある（特に日本語入力への対応はまちまちである）。日本語ディストリビューションでは、必要なパッチを適用した状態のGNOMEデスクトップが提供されているが、Ximian GNOMEではこの点が十分にサポートされていないのだ。

これが意外と……

最後にやっぱり紹介しておきたいのがソースからのコンパイルだ。これが意外と、いさぎよくてシンプルなのである。ソースからコンパイルしてインストールする場合、RPMパッケージのインストール状態を管理しているデータベースにはその情報は反映されない。コンパイル～インストールという作業がRPMによる管理の枠組みの外で行われているので、これは当然のことだ。結果として、RPMが持っているGNOMEパッケージに関する情報は意味のないものになってしまう。

■インストールの順序

ソースからのコンパイルでも、依存関係に注意が必要だ。ただし、これに関しては、ある程度情報が整理されている。GNOME ProjectのWebサイトに記載されているコンポーネントのリストは、インストール順になっている。また、FAQのページにもそれに関する解説がある。問題なのは、両者の順序に違いがある点だ。

そこで、各コンポーネントのREADMEとINSTALLファイルに目を通して依存関係をチェックし、上記のドキュメントも参考にしつつ、インストールに成功したのが以下に示した順番だ。

1. audiofile
2. esound
3. glib
4. gtk+
5. imlib
6. gtk-engines
7. ORBit
8. gnome-libs
9. libxml
10. libghttp
11. libgtop
12. gdk-pixbuf
13. oaf
14. GConf
15. gnome-vfs
16. libglade
17. scrollkeeper
18. control-center
19. gnome-core
20. gnome-print
21. bonobo
22. mc

このあとNautiulをインストールするには、その前に少なくともlibrsvgとeelをインストールする必要ある。そのほかのコンポーネントは、任意の順でインストールできるはずだ。

■作業手順

当然のことながら、まずソースコー

画面2　Ximian GNOMEの標準的なデスクトップ
初回起動時にウィザードを使ってデスクトップの設定を行えるようになっている。画面もそのウィザードで設定した標準的なXimianのデスクトップだ。

ドを入手しなければならない。容量の関係で付録CD-ROMにはソースアーカイブを収録できなかったため（申し訳ありません）、GNOME ProjectのFTPサイトから入手してほしい。最新の安定版は、ftp://ftp.gnome.org/pub/GNOME/unstable/latest/sources/に収められている。

コンパイル前の作業があと2つ残っている。以下の作業を片づけておこう。

- **RPMコマンドを使って、既存のGNOMEのパッケージを削除しておく（“rpm -e パッケージ名”）**
- **ホームディレクトリにあるGNOMEの設定に関するディレクトリ（.gnome、.gnome-desktopなど）を削除する**

準備が整ったところで、いよいよコンパイルに移ろう。tarボールを展開して展開先のディレクトリに移り、おなじみの「./configure」、「make」、「make install」を実行する。このとき、以下のオプションでGNOMEのインストール先を指定できる。

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

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

また、--sysconfdirオプションでシステム設定ファイルが置かれる場所を指定することもできる。これも、ほとんどのディストリビューションで“/etc”と指定して大丈夫なはずだ。

これらの値は、そのディストリビューションが従っているディレクトリ配置のポリシーに従うのが賢明である。GNOMEがインストールされていれば、次のコマンドで確認できる。

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

整理すると、以下のコマンドをtarボールの展開先のディレクトリで実行すればよいわけだ。

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

これを先ほどの順番でコンポーネントごとに繰り返していけば、GNOMEのインストールは完了だ。

ただし環境によっては、gnome-libsのインストールでconfigureのオプションとして“--enable-prefer-db1”を指定する必要があるかもしれない。コンパイルがうまくいかない場合に試してみてほしい。

Column

バージョンの確認

本文中で解説しているとおり、バイナリパッケージを利用する場合でも、ソースからコンパイルする場合でも、GNOMEの各コンポーネントやその他のシステムコンポーネントは、互いに依存関係にある。特にライブラリは、ほかのコンポーネントと依存関係にあることが多い。

こうした依存関係には、コンポーネントの有無だけでなく、そのバージョンもかかわってくる。GNOMEのコンポーネント同士は、同じFTPサイトのディレクトリからダウンロードしたものであればバージョンの問題はないが、「GNOME対その他のシステムコンポーネント」については、それぞれのバージョンをチェックして依存関係に問題がないか確かめる必要がある。このコラムでは、コンポーネントのバージョン確認方法を紹介しておくことにしよう。

RPM系のディストリビューションの場合、rpmコマンドを使ってバージョンを確認することができる。

```
$ rpm -q パッケージ名
```

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

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

ほとんどのGNOME関連ライブラリには、その構成を表示するためのコマンドが用意されている。コマンドファイル名は「ライブラリ名－config」という形式になっている。このコマンドに「--version」オプションを付けて実行するとバージョンが表示される。

たとえばgnome-libsの場合、以下のようになる。

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

共有ライブラリについては、/usr/libなどに置かれているライブラリファイルを参照する方法もある。ライブラリファイルの名前は、以下のような形式になっている。

```
ライブラリ名.so.X.X.X
```

例外もあるがほとんどのケースで“X.X.X”の部分がバージョンを表しているので、ファイル名からバージョンを判別できるはずだ。ほかの方法で確認できないないときは、これで確認しよう。

レーザーファイブ L-Card+

カード型Linuxマシンで遊ぼう

レーザーファイブ　L-Card+
http://www.laser5.co.jp/embedded/lcard/

第3回　L-Card+を温度計にしてみよう

文：吉田 功
Text : Isao Yoshida

L-Card+は組み込み用のワンボードマイコンです。何かを制御してこそ本領が発揮できます。それには、L-Card+から信号を引き出して、制御する装置に接続しなくてはなりません。そこで、今回はL-Card+での制御入門編として、温度センサを取り付け、ネットワークを介して離れた場所の温度を測定したり、ログを取ったりできる装置を作ってみることにしましょう。

L-Card+に温度センサを付けるには？

一般的な温度センサは、温度で出力電圧を変化させ、その電圧を測定して温度測定をします。電圧をコンピュータで扱えるデジタルデータ（数値）に変換するにはA/Dコンバータ（アナログ／デジタル変換器）という装置を使います。L-Card+に搭載されているVR4181というマイクロプロセッサには、A/Dコンバータが内蔵されていますので、この機能を使って温度を測定することにしました。

VR4181のA/Dコンバータの入力（アナログ入力）信号は、L-Card+の拡張バスコネクタ（**写真1**）から取り出せるようになっています。2つあるコネクタのうち、CN6のほうはISA互換バス関連の信号が多く、コネクタCN5のほうに、先月号で使用したパラレル入出力のGPIOや、今回使用するアナログ入力があります。

この拡張バスコネクタから信号を取り出せれば、入出力ポートを使用できますが、このコネクタは端子間のピッチが狭いので、これらを基板に取り付けてハンダ付けするには技術が必要です。しかし、レーザーファイブからオプションとして販売されている「拡張用ユニバーサルボード」を使えば、簡

写真1　拡張バスコネクタ
拡張バスには64ピンの高密度なコネクタを使用している。

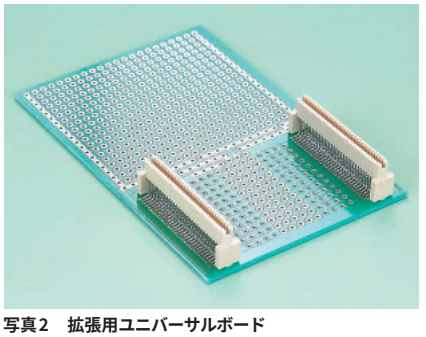


写真2　拡張用ユニバーサルボード
拡張バスコネクタからの信号を扱いやすいピッチに変換してくれる。

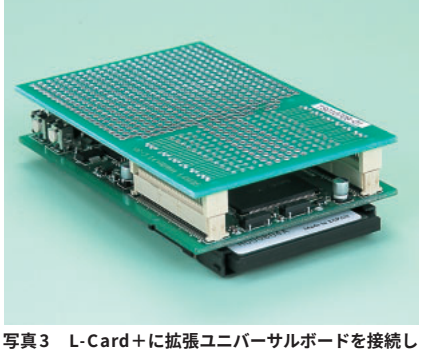


写真3　L-Card+に拡張ユニバーサルボードを接続したところ

単に信号を取り出すことができ便利です。7900円と少々値が張りますが、L-Card＋を本気で使おうとしたら、ぜひともほしいオプションです。

■拡張用ユニバーサルボードについて

拡張用ユニバーサルボードは、拡張バスコネクタ（68ピン×2）からの信号を、12個×8列で10分の1インチピッチ（約2.5mm間隔）に変換してくれる基板です（**写真2**）。これをL-Card＋につなぐと簡単に信号を引き出せるようになります（**写真3**）。

この基板には、L-Card＋と繋げるためのコネクタは実装済みですが、それ以外の部品は一切付いていません。その代わりに、基板にたくさんの穴が開いていて、そこに部品の足（端子）を挿してハンダ付けできるように丸い金属箔（ランド）が付いています。自由に使える部分の穴の数は20×19個あります。この部分に直接に部品をハンダ付けして、変換された拡張バス信号からリード線などで部品まで配線して使用します。

7900円もする基板ですし、いろいろな回路で実験に使いたいので、配線が簡単に脱着できるように、変換された拡張バス信号の部分に、**写真4**のソケットをハンダ付けして、そこにジャンパ用ワイヤ（**写真5**）を接続して使うことにしました。これなら、配線を簡単に付け替えられます。ただし、ソケットとジャンパワイヤの組み合わせは、長期間使用すると接触不良を起こす可能性があるので、あくまでも実験用と覚えておいてください。

今回使用したソケットは40ピンぶんが1本になっていて、それをカッターで好きな長さに切って使えるタイプです。おそらくIC用のソケットだと思われます。初めから12ピンぶんに切られたソケットなら、自分で切らずに使えて便利かもしれません。

拡張用ユニバーサルボードにソケットをうまく取り付けるコツをお教えしましょう。ソケットを基板にセットし、きちんと並べたらセロテープで借り止めします。こうしておけば、ひっくり返しても、ソケットがずれることがありませんので、安心してハンダ付けできます。テープが貼り付いている部分のピンは、テープを剥がしてからハンダ付けしないと熱で溶けてしまうかも

写真4　信号取り出し用に使用したソケット
簡単に配線の変更ができて便利。

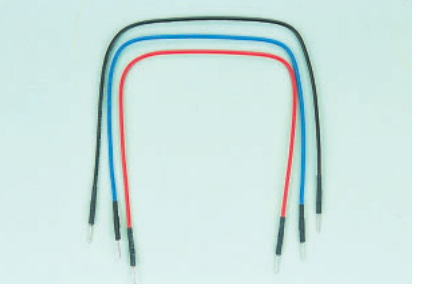


写真5　ジャンパワイヤ
ソケット同士の配線をするときに使用する。

Column

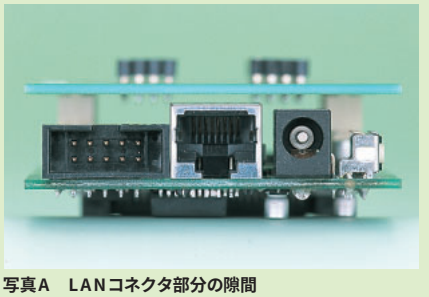
拡張用ユニバーサルボードの信号

L-Card＋の説明書には、拡張バスコネクタの信号の内容が書かれていますが、拡張ユニバーサルボードの説明書には、変換された12×8列の端子との対応表は書かれていません。そのため、自分で基板のパターンを追いかけたり、テスタであたるなどして調べる必要があります。筆者が調べた信号の一覧を105ページの**表1**にまとめておきました。

L-Card＋のコネクタ（64ピン×2）と変換された（12ピン×8）を比べると端子数が32個足りません。調べてみたところ、この足りないぶんは、グランド（GND）を配線していないためで、＋5VとGNDが1個ずつしか出ていません。もっとも、＋5VやGND端子は別配線されていて、ユニバーサル部分の外周に＋5VやGND用のラインが用意されていますので、困ることはありません。

ちなみに、L-Card＋は3.3Vで動作しています。入出力信号も0〜3.3Vの範囲で与える必要があります。しかし、拡張バスコネクタには3.3Vの電源出力はありません。電源ラインとして、5Vが用意されていますが、これはL-Card＋の電源コネクタの＋V側と直結しているだけです。たとえば、L-Card＋を9V電池で動作させたりすると9Vがかかってしまうことになります。接続する回路を設計するときは、L-Card＋に出力する電圧が3.3Vを超えないような工夫が必要です。

L-Card＋に拡張ユニバーサルボードを取り付けると、両者の基板の隙間は13mmあります。しかし、LAN用コネクタが大きく、この部分の隙間はわずか2mmしかありません（**写真A**）。まずいことに、LAN用コネクタはシールドケースが付いており、GNDに接地されています。そのため、この部分に部品を取り付ける場合、ハンダ付けした部品の足と接触しないよう配慮するか、この部分は利用しないようにしなければなりません。



写真A　LANコネクタ部分の隙間
L-Card＋に拡張用ユニバーサルボードを取り付けると、LANコネクタ部分の隙間が2mmしかなくなる

しれません。最初に、テープを貼っていない両端のピンをハンダ付けしてソケットを固定してからテープを剥がして、残りのピンをハンダ付けするようにしましょう。

■温度センサ基板を作ろう

拡張用ユニバーサルボードにソケットをハンダ付けしたら、今度は、温度センサ基板を製作しましょう。配線図は**図1**の通りです。LM35（LM35DZ）というのが今回の主役である温度センサです。LM35には3本の足があり、それぞれ電源（＋5V）、グランド（0V）、出力に割り当てられています。接続を間違えると、LM35だけでなく、L-Card＋自体を壊したり、火傷をしたりする可能性もあるので気を付けましょう。

LM35などの部品は、ユニバーサル基板という穴がいっぱい開いた基板の上にハンダ付けします。オプションの拡張用ユニバーサルボードとは違い、ハンダ付けできるのは片面だけですが、値段が安くお手軽です。

部品の足を基板の穴に通し、部品と部品の間を**図2**のように繋いでいきます。**写真6**は、完成した温度センサ基板です。

温度センサ基板と、拡張用ユニバーサルボードの間は、ジャンパワイヤで配線しました（**写真7**）。温度センサ側にも、拡張ユニバーサルボードで使ったソケットの余り使います。配線する端子は、5V、GND、AD0、の3つで、拡張用ユニバーサルボードの信号表（**表1**）と見比べて配線します（**写真8**）。

使用した部品リストは**表2**の通りでした。部品は秋葉原の「千石電商」で揃います。通販もできますし、Webペ

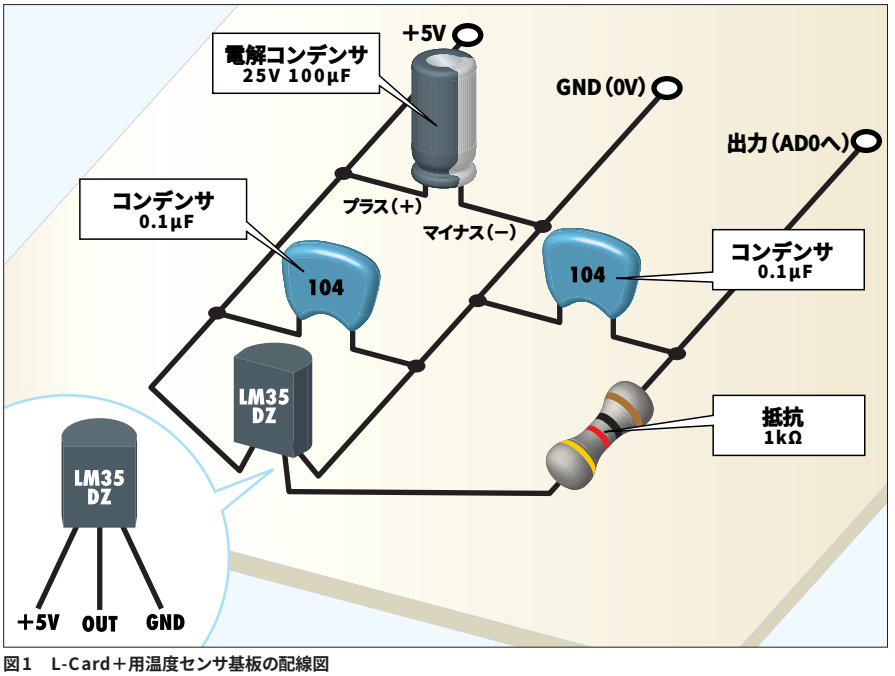


図1　L-Card＋用温度センサ基板の配線図

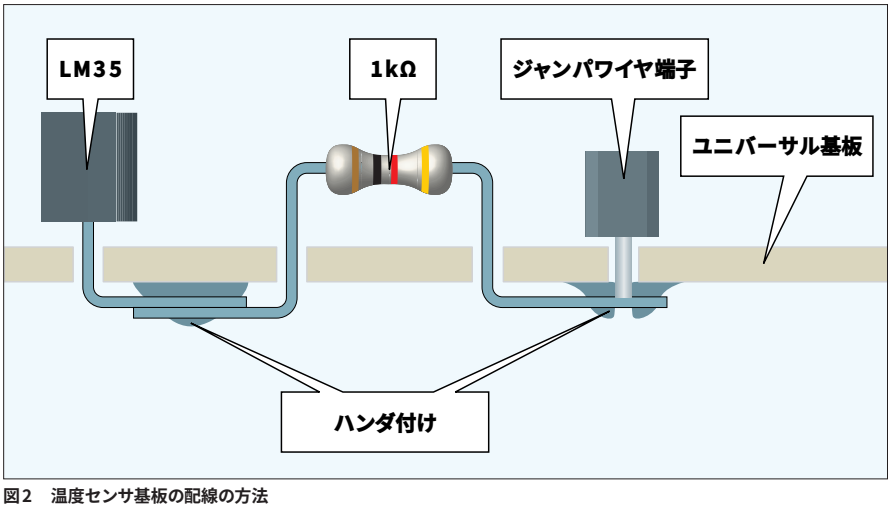


図2　温度センサ基板の配線の方法

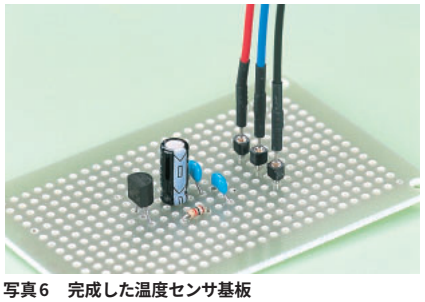


写真6　完成した温度センサ基板

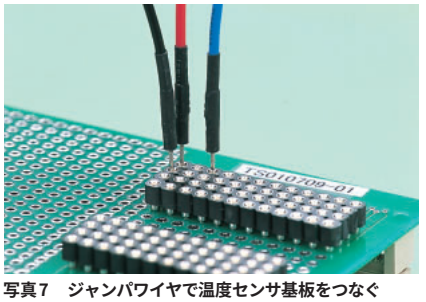


写真7　ジャンパワイヤで温度センサ基板をつなぐ

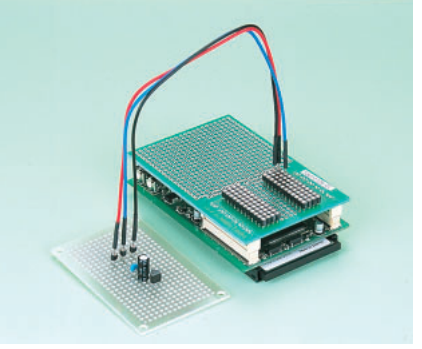


写真8　拡張ユニバーサルボードから配線を引き出す様子
ジャンパワイヤは、赤を5V、黒をGND、青をAD0にそれぞれ挿している。

ージ（http://www.sengoku.co.jp/index.htm）もあるので利用しやすい店だと思います。

■温度センサについて

温度センサで有名なものに、温度で抵抗値が変わるサーミスタと呼ばれるものがあります。価格も安いのでいろいろなところに使われています。**図3**のように抵抗を通してサーミスタに電流を流すと、温度によってA点の電圧が変化するので、電圧計を使って温度が測定できます。

しかしサーミスタには欠点もあります。個体によるばらつきが多いので、必ず校正が必要になることと、温度と抵抗値の変化の関係が直線的でなく**図4**のようなグラフになるので、電圧から温度を直読しにくいことです。

今回使用した、ナショナルセミコンダクタ社のLM35（**写真9**）というデバイスも、サーミスタ＋抵抗のように温度によって出力の電圧が変化します。しかし、測定可能温度の－55～＋150℃の間、温度による電圧の変化が一定なうえ、0℃なら0V、100℃なら1Vと、温度によって出力される電圧が決められているので、校正の必要がありません。さらに、＋25℃時に±0.5℃と精度もまずまずです。価格も250円程度ですので、無調整で簡単に使えるLM35は便利です。

ただし、L-Card＋で使うには厄介な面もあり、0℃以下の温度を測定するには、マイナスの電源（負電圧）が必要になります。L-Card＋は、負電圧電

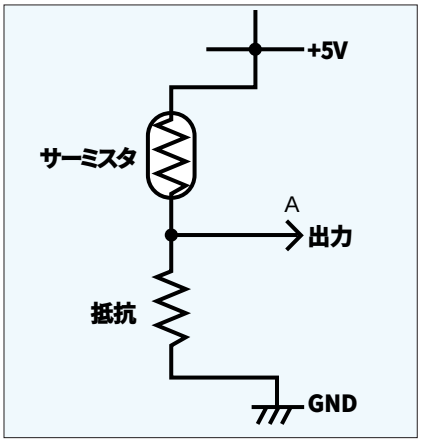


図3　サーミスタと抵抗を使った温度センサの回路

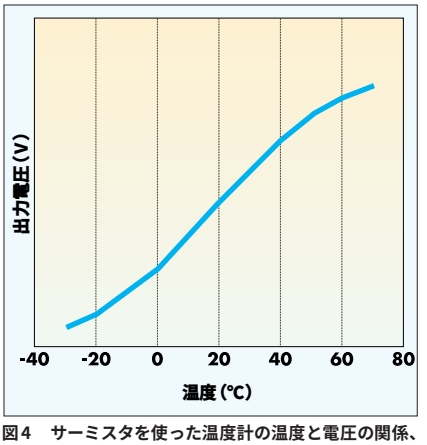


図4　サーミスタを使った温度計の温度と電圧の関係、直線的に変化していない

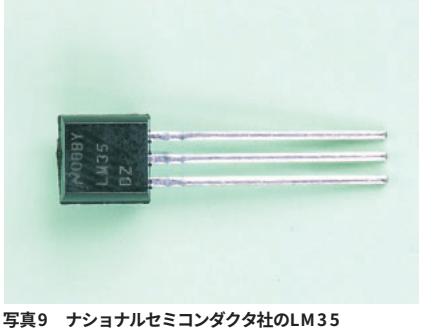


写真9　ナショナルセミコンダクタ社のLM35
リニアリティが良く校正の必要がないスグレものの温度センサ

CN5			
IOSC16/GPIO19	RESET	GND	5V
AD2	AIN	AOUT	N.C.
TPY1	TPY0	AD1	AD0
FPD1	FPD0	TPX1	TPX0
FPD5/GPIO13	FPD4/GPIO12	FPD3	FPD2
UBE	IORDY	FPD7/GPIO15	FPD6/GPIO14
FLM	POWER	IORD	IOWR
VPLCD/VPGP1	VPBIAS/VPGP0	CHCLK/LCDCS	LOCLK/MEMCS16
PCS0	GPIO02(LED6)	GPIO01	GPIO00
GPIO07	GPIO06	GPIO05	GPIO04
CLK48/PCS1/GPIO11	CD1	CD2	GPIO08(LED5)
MPOWER	CF_DEN	IRDIN	IRDOUT

CN6			
GPIO24/ROMCS2	ROMCS3	RIH	RCS
SYSDIR	SYSEN	GPIO22/ROMCS0	GPIO23/ROMCS1
D01	D00	MEMRD	MEMWR
D05	D04	D03	D02
D09	D08	D07	D06
D13	D12	D11	D10
A1	A0	D15	D14
A05	A04	A03	A02
A09	A08	A07	A06
A13	A12	A11	A10
A17	A16	A15	A14
A21	A20	A19	A18

表1　拡張ユニバーサルボードの信号

基板	ユニバーサル基板（小）
温度センサ	LM35DZ
抵抗	1kΩ（1/6W）
コンデンサ	積層セラミック　0.1μF×2個 電解　25V 100μF
ICソケット	丸ピン40P　1列
ジャンプワイヤ	サンハヤト JUMP WIRE PIN＋PIN 200mm（SPP-200）

表2　L-Card＋用温度センサ基板で使用した部品

Column

ハンダ付けのための用具の選び方と扱い方

拡張ユニバーサルボードにソケットを付けたり、温度センサを組み立てるにはハンダ付けが必要です。ハンダ付け初心者のためにコツを紹介しましょう。

ハンダ付けを成功させる秘訣は、用途に合った良い道具を選ぶことです。小型の電子部品を取り付けるには、小型のハンダゴテと細めで良質な糸ハンダを使わないとうまくいきません。と言っても、決して高級で高価な製品である必要はなく、名の通ったメーカーのベーシックな製品で十分です。

ハンダゴテは、15Wくらいのセラミックヒータを使ったタイプがよいでしょう（**写真B**）。重要なのはコテ先の形状です。細いほうが細かい作業に向いているのですが、熱の伝わりが悪く、逆に作業性が悪くなって失敗も多くなります。ある程度太いほうが熱が伝わりやすく作業がはかどります。私が使用しているのは写真Cの2本なのですが、先の細いほうは表面実装部品（フラットパッケージ）など微細なもののハンダ付け向きで、ふだんは太いほうを使用しています。どちらも、1000円程度で買ったものです。

次に、糸ハンダも細くて良質なものを選びましょう。太さは0.6〜0.8mm程度、JISマーク付きのものを選べば安心です。

ハンダ付けのコツは、あせらずに、部品や基板を十分に暖めることです。もちろん、あまり長時間コテを当て続けると、部品や基板を壊してしまいます。時間は長くて5秒程度、これ以上かかってしまいそうなら、いったん休憩して、部品や基板が冷えてから再度トライしましょう。

まず、ハンダ付けしたい部分にコテ先を当て0.5〜1秒ほど暖めつつ、糸ハンダをコテ先と部品の隙間に当てハンダを溶かします。このとき、糸ハンダをコテ先に軽く押しつける感じにするとスムースにハンダが溶けてくれます。糸ハンダやコテ先の角度を変えながら、ハンダが部品と基板の間に浸みてくるようにします。

ハンダの量はダンゴになったり、隣とくっついたりしないように少な目が理想的です。ただし、ユニバーサル基板にハンダ付けをするには、ある程度の量のハンダがないとうまく付かない場合もあります。とにかく、隣とくっつかないように気を付けましょう。

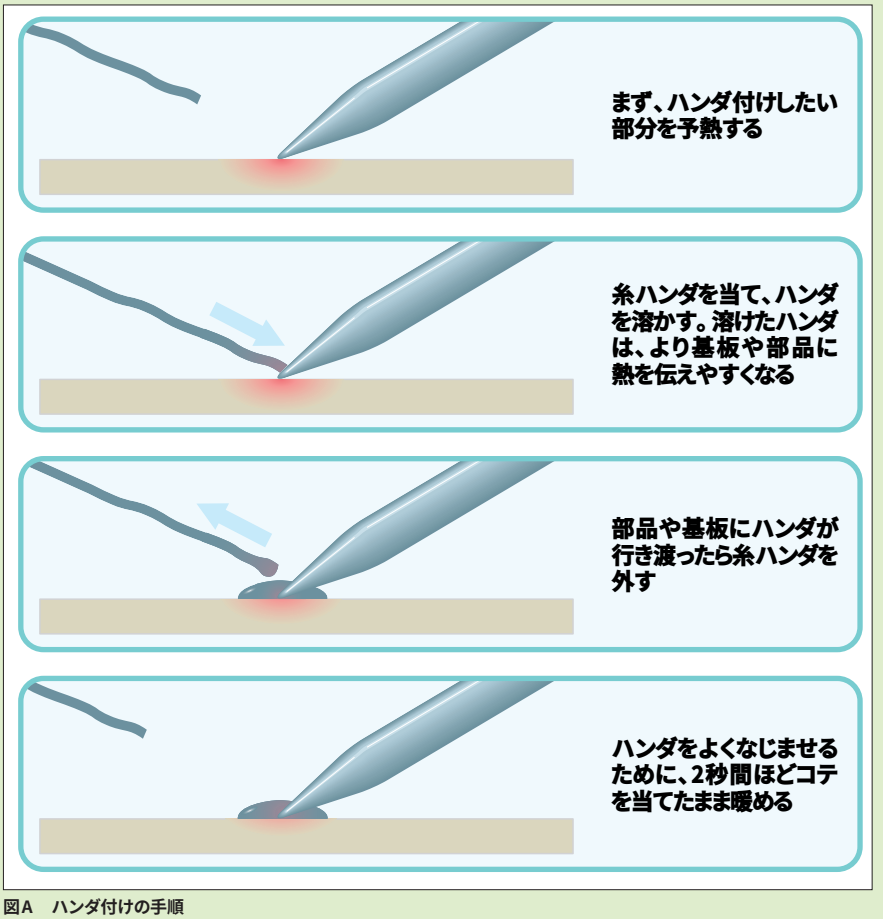
部品と基板に上手くハンダが付いたら、糸ハンダを外し、さらに1〜2秒間熱してハンダを馴染ませ終了です。この一連の作業（**図A**）を5秒以内でできると理想的でしょう。

あとから、ルーペで確認してみましょう。ちょっと不安があるようなら、再びハンダゴテを当て、部品に付いたハンダを溶かし、2秒間ほど待ってからコテを外します。

ハンダ付けする部品や基板が汚れていると、うまく付きません。汚れてしまったらアルコールで拭いたり、目の細かいサンドペーパーで磨いたりします。また、コテ先も常に綺麗にしておきましょう。専用のハンダゴテクリーナーも売られていますが、少し湿らせたティッシュペーパーで拭くのも効果があります。

写真B　セラミックヒータを使用した半田ゴテ

写真C　コテ先の太さで、扱いやすさが違うので、用途に合わせた選択をすべきだ



図A　ハンダ付けの手順

源を持っていないので、別に用意するか小細工しないと測定できません。

もうひとつの問題は、規格上150℃で1.5Vまでの電圧しか出ないことになっていても、電源電圧に5Vを使用していると、デバイスの破損や何らかのトラブルで、A/Dコンバータの最大入力電圧（3.3V）を超える電圧が出力されてしまう可能性があることです。最悪の場合、L-Card＋が壊れてしまうかもしれません。

入手できたVR4181の資料には、A/Dコンバータの規格が詳しく載っていませんでしたので、正確なところは不明ですが、保護用に1kΩの抵抗を通して、あとはVR4181内部の保護回路に期待することにしました。A/Dコンバータの入力インピーダンスは比較的高いようなので、この程度の抵抗であれば測定誤差も少ないと思います。

■VR4181のA/Dコンバータについて

VR4181のA/Dコンバータは、10ビットの分解能を持っています。これは、2の10乗ですから1024の分解能があるということになります。また、0VからA/Dコンバータ用電源の電圧までを測定することができますが、L-Card＋では、CPUなどの電原電圧と共用しているため、0Vから3.3Vまでとなり、3.3V÷1024＝約3.2mV単位で電圧が測定できることになります。

VR4181に内蔵されているA/Dコンバータは、**図5**のような構成になっています。入力は全部で8チャンネルあり、そのうち4チャンネルはタッチパネル用、1チャンネルは音声（オーディオ）用に割り当てられていて、それぞれ専用のハードウェアを持っています。残りの3個（AD0～AD2）は、汎用アナログ入力用に割り当てられています。これは、バッテリの残量測定などに使用することを想定しているもののようです。

全部で8チャンネルと言っても、実はA/Dコンバータは1つしか持っていません。8チャンネルのセレクタを使い、チャンネルを切り替えて使用します。

VR4181は、タッチパネルをつなぐために、PIU（タッチパネルインターフェイスユニット）という回路を持っていて、タッチパネルぶんの4チャンネルを自動的に切り替え変換して4つのレジスタに書き込んでくれます。これを

Column

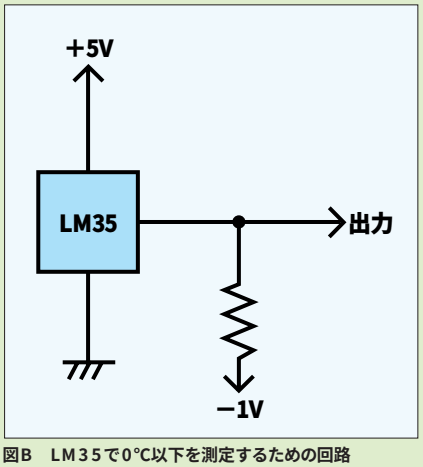
LM35で0℃以下の温度を測定する方法

LM35で0℃以下を測定する場合は、図Bのようにマイナスの電源が必要になります。出力も－50℃なら－0.5Vとマイナスになりますが一定です。デジタル電圧計で測定すれば、表示もマイナスになるので直読できて便利なのですが、L-Card＋のようなプラスの電源しか持たない装置に接続するのは少々厄介です

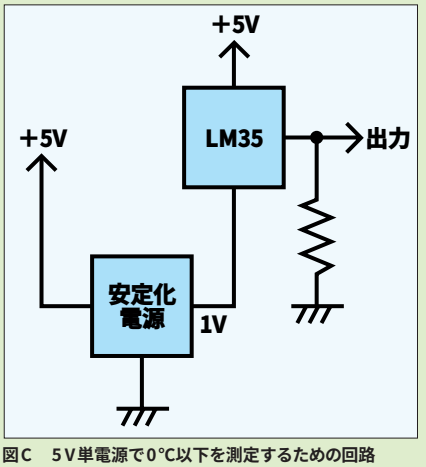
こんな装置に取り付ける場合は、図Cのような回路にします。LM35は最低でも4Vの電源を必要としますので、5V電源で動作させる場合は、－1Vのオフセットがかかるように1Vの安定化電源を用意します。この場合、出力される電圧は－50℃なら0.5V、50℃なら1.5Vと1V高くなります。ただし、1Vちょうどで精度の良い電圧を作るのが少々難しいかもしれません。

別のアプローチとして、図Dのような方法があります。オフセットする電圧の精度が悪くても、A/Dコンバータを使って、その電圧を測ればよいわけです。この回路では、ダイオードの順方向電圧降下の特性を利用して1～1.4V程の電圧を作り出していますが、この特性は温度や個体のバラツキによって変化します。そこで、この電圧を毎回測定してプログラムで減算すると、LM35が出力した電圧がわかる仕組みです。

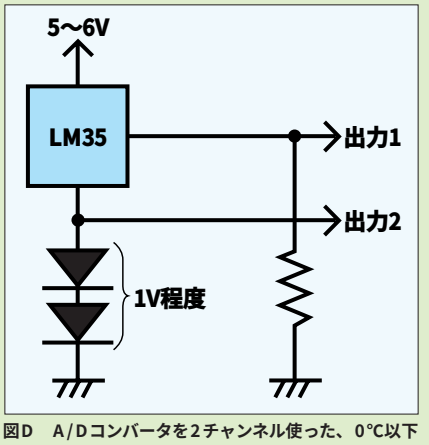
ただし、2カ所の電圧を測定するので、A/Dコンバータの測定誤差も2倍になります。分解能による誤差が±3.2mVですから、最悪±6.4mV（0.64℃に相当）ずれることになります。



図B　LM35で0℃以下を測定するための回路



図C　5V単電源で0℃以下を測定するための回路



図D　A/Dコンバータを2チャンネル使った、0℃以下を測定するための回路

一定時間に自動的にサンプリングし、変換が終了したら割り込みで知らせてくれるので、プログラムはサンプリングされたデータを元に処理するだけで済みます。

音声用のチャンネルはAIU（オーディオインターフェイスユニット）という回路を使い、自動的に高速サンプリングをして、さらにはDMAを使ってメモリに変換したデータ並べてくれます。保存するのも、データ加工するのも、メモリに並んだものを処理するだけなので、プログラムによるオーバーヘッドを減らすことができます。

今回のような温度測定は、せいぜい1分に1回程度変換すればよいので、これらの回路は使わず、プログラムで1データずつサンプリングする方法を取りました。

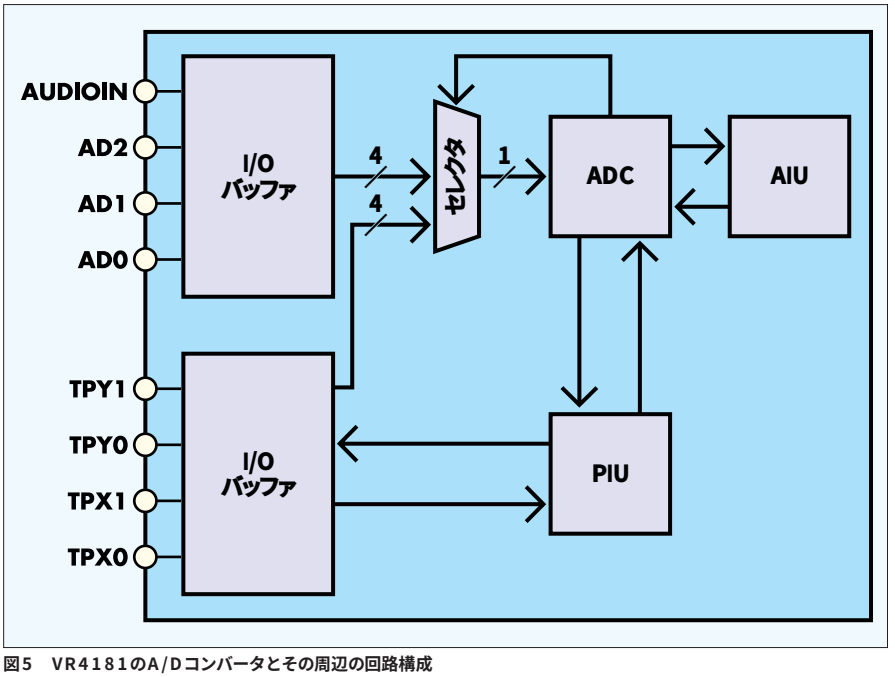


図5　VR4181のA/Dコンバータとその周辺の回路構成

■プログラムについて

プログラムは、先月紹介したLED点灯プログラムをベースに作っています。I/Oプログラミングの方法も基本的には同じなので、共通部分も多くなっています。

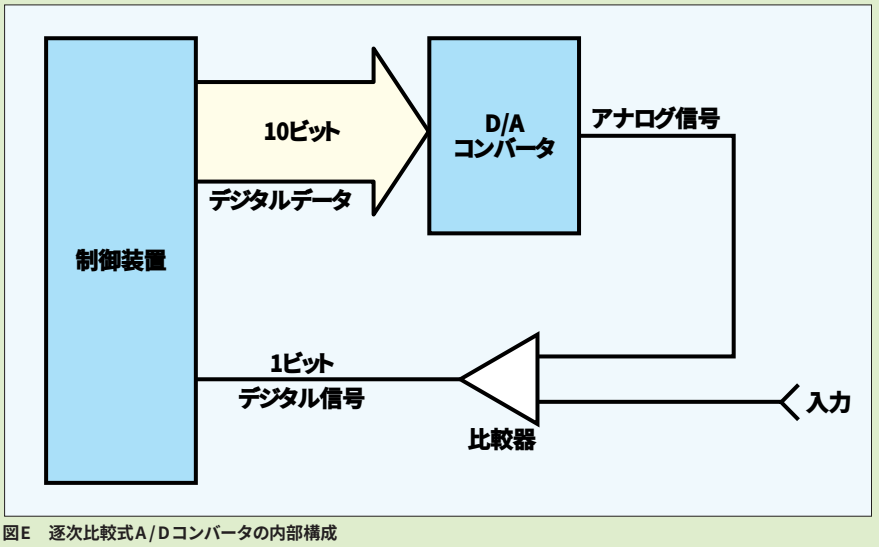
VR4181のA/Dコンバータを使うには、あちこちのレジスタを操作しなければならず、なかなか面倒です。忘れ

Column

A/Dコンバータのしくみ

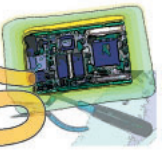
A/Dコンバータにはいろいろな種類があります。NECが出しているVR4181のユーザーズマニュアルには、搭載されたA/Dコンバータがどんな種類のものか書かれていないようです。ここでは、一般的な逐次比較式というA/Dコンバータのしくみを紹介しましょう。

逐次比較式は、図Eのように、デジタルをアナログにするD/Aコンバータと、電圧の比較器から構成されています。D/Aコンバータは、セットしたデジタル値に比例した電圧が出せる装置です。比較器の出力はデジタルのON／OFF信号なので、電圧が高いか低いかだけしかわからず、何V高いのかはわかりません。



図E　逐次比較式A/Dコンバータの内部構成

D/Aコンバータの分解能が10ビットだとして説明します。まず、1番上位のビットを「1」にして「1000000000」とします。このときD/Aコンバータからは最大電圧の半分が出力されます。この電圧と入力電圧を比較し、これより大きければそのままで、小さければ「0000000000」と変更します。たとえば、入力電圧が最大電圧の半分より高かったとしましょう。今度は2番目のビットも「1」にして「1100000000」とします。このときのD/Aコンバータの電圧は最大値の4分の3です。この電圧と入力電圧を比較し、2番目のビットを決定します。これを全部のビットに対して繰り返すと変換は終了です。



がちなのは、バスコントロールユニットにあるCMUクロックマスクレジスタで、これでA/Dコンバータ関連のクロックを有効にします。次にAIUにある、マイク制御入力レジスタを操作して、A/Dコンバータ用電源電圧をONにします。ただし、AIUの回路は使わないので、クロックを共有する必要はありません。これで、やっとPIUを使ってA/Dコンバータを操作できるようになります。

今回紹介したハードウェアとプログラム（**リスト1**）では、温度は0～150℃まで測ることができます。A/Dコンバータの分解能は約0.3℃、温度センサの誤差は±0.5℃程度ですので、表示は0.1℃単位としました。分解能や誤差以上の表示をしますので、最後の桁は若干ふらつきます。気持ちが悪い場合は、表示の単位を1℃ごとにするとよいでしょう。

VR4181は浮動小数点演算ユニットを持っていませんが、32ビットの整数演算が可能です。その表現能力は－2147483648～2147483647と大きいので、温度を10倍の数値で演算する固定小数点演算とし、表示は小数点以上は10分の1にしたものを、小数点以下はその余りとすることで、小数第1位の桁まで表示させています。

プログラムは起動時の引数の数で動作が変わるようになっています。引数がない場合は、一度だけサンプリングして終了します。本来は、この機能だけを実装して、連続的に温度を測定する場合は、cronを使って自動実行すればよいと思っていたのですが、L-Card＋に付属のrootイメージには、crontabがありませんでした。新たにインストールするよりは、アプリケーション側で対応したほうが簡単なので、引数が2つある場合は、サンプリングの間隔（1秒単位）と回数と解釈して動作します。たとえば、

```
% lm35 60 120
```

を実行すると、60秒間隔で120回（2時間）サンプリングします。温度表示は、標準出力に書き出しますので、ファイルに落としたいときは、リダイレクトを使います。

```
% lm35 60 120 > file
```

■最後に

最も有用な利用方法は、L-Card＋にtelnetで接続し、遠隔地の温度測定をすることでしょう。インターネットに接続できれば、地球の裏側からでも利用できます。また、一定間隔でログをとり続けられるので、1日の温度変化を見るのも面白いかもしれません。電池で動作させるときは、5VのACアダプタの代わりに、乾電池4本（6V）を使うとよいでしょう。

実験中に、サンプリングした温度が不安定になるというトラブルに見舞われました。原因を調べてみると、拡張ユニバーサルボードから出ている5V電源に、レベルの高いノイズが混入していることがわかりました。どうやら、L-Card＋は相当なノイズ源になっているようです。このノイジーな電源を使って温度センサを使っていたのでは、あまり正確なデータが得られません。試しに、センサのみを乾電池で動作させてみたところ、不安定さが軽減されました。そこで、センサの電源にノイズ対策用コンデンサを入れ、さらにアナログ入力端子にもコンデンサを入れて対策したところ、かなりの改善が見られました。L-Card＋にアナログ回路を付ける場合は、ノイズに注意することをお勧めします。

さて、次号ではL-Card＋でどんなことをして遊びましょうか。

▼リスト1　LM35温度計プログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <asm/page.h>
#include <asm/types.h>
#include <time.h>

/* VR4181レジスタ定義 */
#define CMUCLKMSK    0x004  /* CMU MASK */
#define ADCCLKEN     0x1A   /* CMU A/D & PIU */
```

▼リスト1　LM35温度計プログラム　続き1

```
#define MCNTREG      0x172  /* AIU CONT */
#define ADENAIU      0x8000 /* AIU A/Denable */
#define PIUCNTREG    0x122  /* PIU CONT */
#define ADCMDSCAN    0x0A   /* PIU A/D & PWR */
#define PIUSEQEN     0x04   /* PIU Sequence enable */
#define PIUCMDREG    0x12A  /* PIU command mode */
#define AD0SEL       0x04   /* PIU AD0 select */
#define PIUINTREG    0x124  /* PIU interrupt */
#define PADCMDINTR   0x40   /* PIU INTR CMD SCAN */
#define PIUAB0REG    0x2B0  /* PIU AD0 data */

#define ADVOLT       3300   /* AD refarence voltage(mV) */

unsigned char  *mapadrs ;

int  main(argc,argv)
int argc ;
char *argv[] ;
{
    int fd;
    unsigned int st, len,i ;
    char *dev="/dev/mem";
    int temp,delay,loop,count ;
    time_t tick,tickr ;

    if(argc<3)
    { /* オプションなし */
        loop=1; delay=0;
    }
    else
    { /* オプション解析 */
        loop=atoi(argv[2]);
        delay=atoi(argv[1]);
    }

    /* メモリデバイスをオープン */
    fd=open(dev, O_RDWR);
    if(fd<0)
    {
        fprintf(stderr,"cannot open %s\n", dev);
        exit ;
    }
    /* メモリのマッピング */
    st=0x0A000000;
    len=0x1000;
    mapadrs=mmap(0,len, PROT_WRITE,MAP_SHARED,fd,st);
    if(mapadrs == MAP_FAILED)
    {
         fprintf(stderr,"cannot mmap\n");
         exit ;
    }
    /* クロック供給 ON */
    *(short *)(mapadrs + CMUCLKMSK) |= ADCCLKEN ;
    /* メモリデバイスをクローズ */
    munmap(mapadrs, len);
```

▼リスト1 LM35温度計プログラム 続き2

```
    close(fd);

    /* メモリデバイスをオープン */
    fd=open(dev, O_RDWR);
    if(fd<0)
    {
        fprintf(stderr,"cannot open %s\n", dev);
        exit ;
    }
    /* メモリのマッピング */
    st=0x0B000000;
    len=0x1000;
    mapadrs=mmap(0,len, PROT_WRITE,MAP_SHARED,fd,st);
    if(mapadrs == MAP_FAILED)
    {
        fprintf(stderr,"cannot mmap\n");
        exit ;
    }
    /* A/D電源 ON */
    *(short *)(mapadrs + MCNTREG) = ADENAIU ;
    /* PIU コマンドモードに初期設定 */
    *(short *)(mapadrs + PIUCNTREG) = ADCMDSCAN ;
    *(short *)(mapadrs + PIUCMDREG) = AD0SEL ;
    if(delay>0) tickr = time(NULL) + delay;
    /* 自動サンプリング用ループ */
    for(count=1; count<=loop; count++)
    {
        /* 変換開始 */
        *(short *)(mapadrs + PIUCNTREG) |= PIUSEQEN ;
        /* 変換終了待ち */
        while(!(*(short *)(mapadrs + PIUINTREG) & PADCMDINTR)) ;

        /* データを取得し温度に変換する */
        temp = (i=*(short *)(mapadrs + PIUAB0REG) & 0x3ff)*ADVOLT / 1024;
        /* 割り込みフラグをクリア */
        *(short *)(mapadrs + PIUINTREG) |= PADCMDINTR ;
        if(delay>0) printf("%d ",count);
        printf("%d.%d\n",temp / 10,temp % 10);
        /* サンプリング間隔ウエイト */
        if(delay>0 && count<loop)
        {
            while((tick = time(NULL)) < tickr) ;
            tickr = tick + delay;
        }
    }

    /* メモリデバイスをクローズ */
    munmap(mapadrs, len);
    close(fd);
    exit ;
}
```

携帯電話 de Javaプログラミング

今回は、前回インストールしたBorland JBuilder 5 Personalとi-JADEを使って、簡単なiアプリの作成手順を説明します。iアプリを作成し、i-JADEを利用して実行結果を確認後、携帯電話へ配布するまでが目標です。ツールをうまく利用することでiアプリの作成はそれほど難しくないことがわかるでしょう。

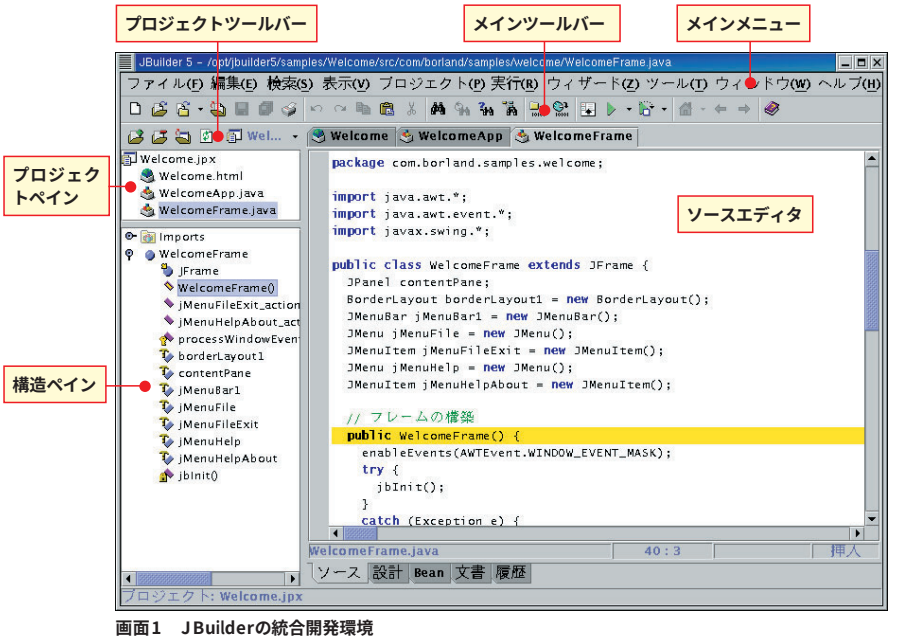
第2回　iアプリの作成その1

文：加藤大受
Text : Daiju Kato　　dkato@jcom.home.ne.jp

この連載で、iアプリを作成するために使用するボーランドのBorland JBuilder 5 Personal（以下、JBuilder）について簡単に説明しましょう。JBuilderは、Javaアプリケーションを GUI環境で開発できる統合開発環境を持った開発ツールです。統合開発環境では、アプリケーションの開発、コンパイル、デバッグ環境を提供していますので、この環境内でアプリケーションの開発・テストが実現できます。

JBuilderを起動すると**画面1**のような画面が表示されます。各部の名称については**画面1**を参照してください。また、iアプリの構築ではほとんど使用しませんが、**画面2**のようなビジュアルにユーザーインターフェイス環境を構築できるビジュアル設計ツールが装備されています。

なお、JBuilderではプロジェクトという単位でアプリケーションを管理します。プロジェクトとは、構築するアプリケーションのファイル、コンパイルオプション、デバッグオプションなどの設定を管理します。JBuilderでJavaアプリケーションを開発するには、最初に必ずプロジェクトを作成してから行うことになります。



画面1　JBuilderの統合開発環境

簡単なiアプリの作成

それでは、画面に文字列を表示する簡単なiアプリを作成してみましょう。

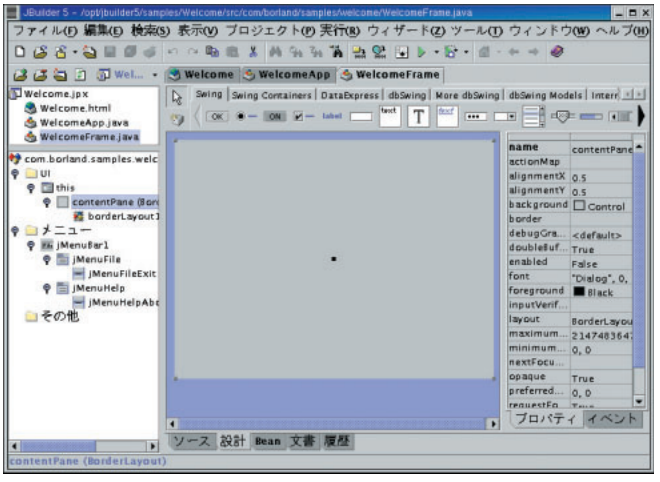
まず、JBuilderを起動します。JBuilderを起動するとWelcomeプロジェクトが開くので、メインメニューから［ファイル］－［プロジェクトを閉じる］を選択し、［プロジェクトを閉じる］ダイアログボックスで「Welcome.jpx」をチェックして、Welcomeプロ

ジェクトを閉じます。

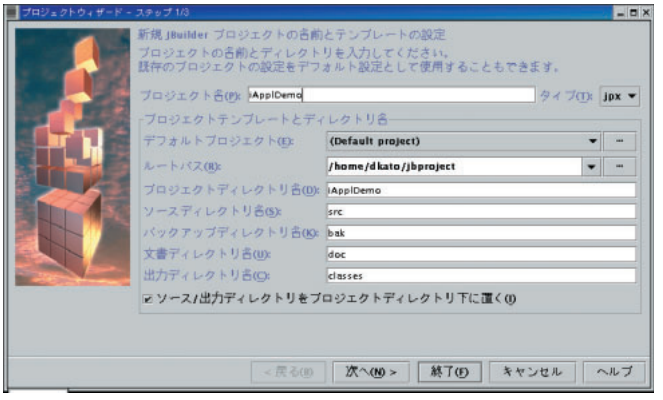
■プロジェクトの作成

前述したように、JBuilderで新規のアプリケーションを作成するにはプロジェクトを作成する必要がありますので、メインメニューから［ファイル］－［新規プロジェクト］を選択して、プロジェクトウィザード（**画面3**）を起動します。プロジェクトウィザードでは、［プロジェクト名］に「iApplDemo」と入力します。これで、プロジェクト名と同じ名称のディレクトリが作成され、その中に作成されたソースファイル、コンパイルされたバイトコードが納められます。

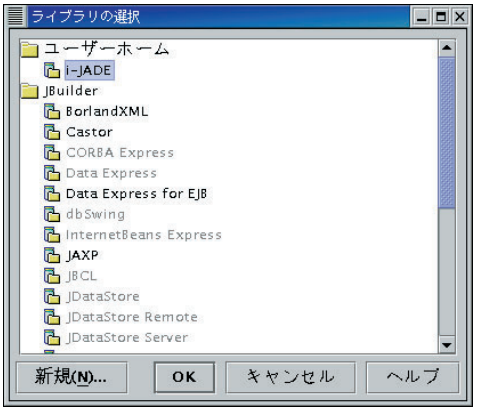
次に、［次へ］ボタンをクリックしてステップ2/3に移ります。ここでは、ソースなどの保存先はデフォルトのままで問題ありません。ただし、iアプリの構築にi-JADEを使用しますので、必須ライブラリにi-JADEを追加する必要があります。このため、［追加］ボタンをクリックして、前回ライブラリに設



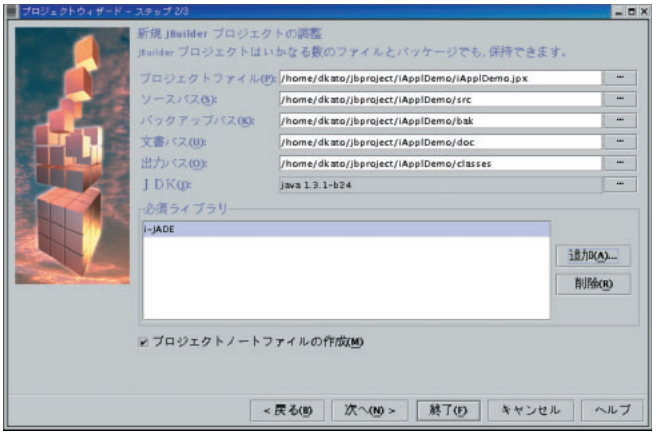
画面2　ビジュアル設計ツール



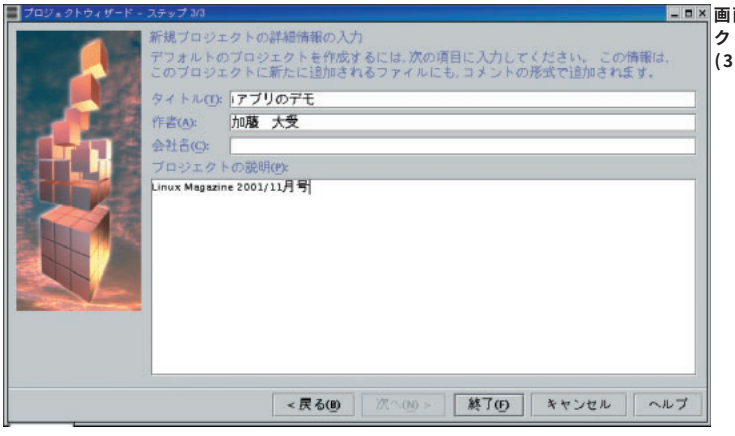
画面3　プロジェクトウィザード(1/3)



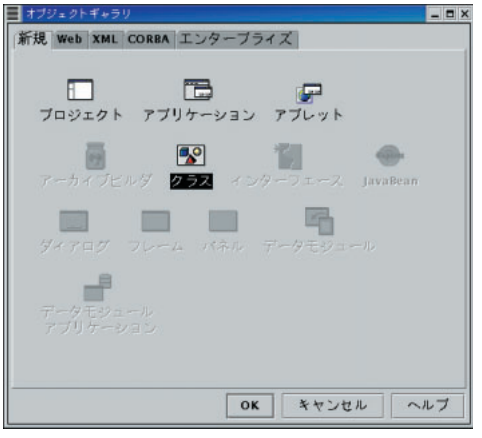
画面4　i-JADEの選択



画面5　プロジェクトウィザード(2/3)



画面6　プロジェクトウィザード(3/3)



画面7　オブジェクトギャラリ

定した「i-JADE」を選択します（**画面4**）。i-JADEが追加されると、**画面5**のように表示されます。

続いて、［次へ］ボタンをクリックして、ステップ3/3に移動します。このステップ3/3ではプロジェクトの情報を入力します。ここで入力された情報はソースのヘッダ部に設定されます（**画面6**）。

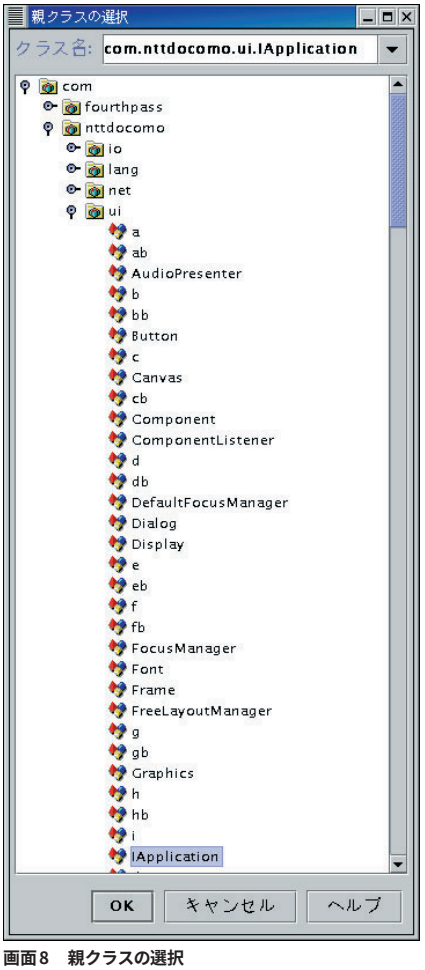
すべての設定が終了したら［終了］ボタンをクリックして、プロジェクトウィザードを終了します。プロジェクトは単なる器だけなので、続いてiアプリのプログラム作成となります。

■クラスの作成

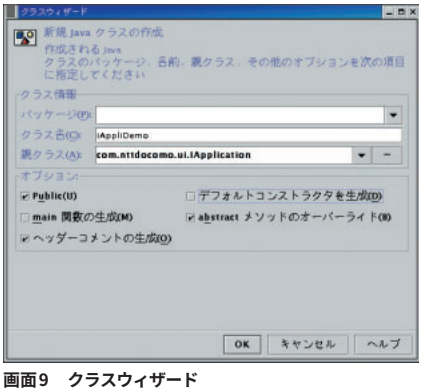
メインメニューから［ファイル］－［新規］を選択するとオブジェクトギャラリが表示されます（**画面7**）。オブジェクトギャラリでは、作成するJavaアプリケーションのひな形を生成するためのウィザードを呼び出すことができます。ここではクラスのアイコンをダブルクリックして、クラスウィザードを起動します。クラスウィザードでは、新規に作成するクラスの雛形を生成することができます。

クラスウィザードが起動したら、［クラス名］を「iAppliDemo」とします。なお、Javaにはパッケージと呼ばれる、作成するアプリケーションやクラスを1つにまとめる機能がありますが、iアプリではパッケージをサポートしていませんので［パッケージ］は空白のままとします。

続いて、親クラスを設定します。iアプリのメインアプリケーションは必ず、com.nttdocomo.ui.IAplicationクラスを継承しなければならない決まりとなっています。この決まりに従うため、親クラスをこのクラスに設定します。［親クラス］横のボタンをクリックして、「親クラスの選択」ダイアログボックス（**画面8**）を開き、com.nttdocomo.ui.IApplicationを選択します。［オプション］では作成するクラスファイルの設定を行うことができます。ここでは、「デフォルトコンストラクタを生成」のチェックを外して、［OK］ボタンをクリックします（**画面9**）。



画面8　親クラスの選択



画面9　クラスウィザード

これで、iAppliDemoクラスが生成されました。このクラスはiアプリを動作させるためのメインクラスとなります。今回は画面に文字列を表示させますので、メインクラスから呼び出されるクラスを作成しなければなりません。

iアプリの画面の描画にはキャンバスを使用します。キャンバスを使用するには、com.nttdocomo.ui.Canvasクラスを継承したクラスを定義しなければなりません。先ほどと同様に、［ファイル］－［新規］でオブジェクトギャラリを表示させ、クラスウィザードを起動します。続いて、［パッケージ］を空白とし、作成するクラス名を「myCanvas」とします。［親クラス］は前述したように、com.nttdocomo.ui.Canvasクラスですので、「親クラスの選択」ダイアログボックスからこのクラスを選択します。iAppliDemoクラスのときと異なり、今回は「デフォルトコンストラクタを生成」のチェックが必要となります。設定が完了したら［OK］ボタンをクリックして、myCanvasクラスを生成します。これで、文字を画面に表示させるために必要な2つのファイルのひな形が生成されました。それでは、コードを追加してiアプリを仕上げてみましょう。

■プログラムの作成

まず、画面への文字列の描画を行うmyCanvasクラスを仕上げます（**リスト1**）。myCanvasクラスは画面への描画・文字の出力のためにCanvasクラスを継承しています。これにより、作成されたキャンバスの領域に図や文字を描画することが可能です。

キャンバスを使用するためには、まずキャンバスのサイズを取得します。キャンバスのサイズは携帯電話の表示領域のサイズとなりますので、ディス

プレイのサイズを取得します（**リスト1**の①）。

キャンバスへの書き込みはpaint()メソッドで行います（**リスト1**の②）。文字列を書き込む前にキャンバス全体を初期化するために、clearRect()メソッドを呼び出しています。先ほど取得したキャンバスの幅、高さを指定することで全体の初期化が可能になります。このclearRect()メソッドは指定された領域をクリアする機能ですので、部分的な領域のクリアも可能です。

画面を初期化したら、drawString()メソッドを使用して文字列を書き込みます。

画面のサイズは機種によって異なりますので（**表1**）、文字表示位置をディスプレイの中央などにしたいときは、キャンバスのサイズから計算で中心を求め、書き込み位置を指定する必要があります。

機種	画面サイズ
P503i、F503i、N503i、P503iS	120×130
SO503i	120×120
D503i	132×126

表1　機種別画面サイズ

続いてメインアプリケーションであるiAppliDemoクラスを仕上げます。iAppliDemoクラスはiアプリの開始・終了などを管理するクラスです。iAppliDemoクラスは、IApplicatinクラスを継承しています。IApplicationクラスでは**表2**のようなメソッドが用意されています。iAppliDemoクラスは、これらのメソッドを使用してiアプリの開始・終了・再開を実現します。

メソッド名	機能
start()	iアプリの開始
terminate()	iアプリの終了
resume()	iアプリの再開

表2　IApplicationクラスのメソッド

iAppliDemoクラス単独では画面上に文字列を表示することはできませんので、先ほど作成したmyCanvasクラスを呼び出すことによって文字列の表示を行います。**リスト2**ではまず、表示するキャンバスを定義しています（**リスト2**の①）。続いて、iアプリを開始したときに定義されたキャンバスを生成するために、start()メソッドでキャンバスを作成し（**リスト2**の②）、作成したキャンバスをディスプレイに表示します（**リスト2**の③）。これで、文字列を表示するiアプリの作成が終了です。

■iアプリの実行

それではJBuilder内からi-JADEを起動して、作成したiアプリを起動してみましょう。まず、作成したプロジェクトからi-JADEが起動されるように設定する必要があります。メインメニュー

▼リスト1　myCanvasクラス（myCanvas.java）

```
import com.nttdocomo.ui.*;
/**
 * タイトル:  iアプリのデモ
 * 説明:     Linux Magazine 2001/11月号
 * 著作権:    Copyright (c) 2001
 * 会社名:
 * @author 加藤　大受
 * @version 1.0
 */

public class myCanvas extends Canvas {

 int width;
 int hight;

  public myCanvas() {
    //画面サイズの取得 ①
    //携帯電話のディスプレイの領域を取得
    width= Display.getWidth();
    hight= Display.getHeight();
  }

  //キャンバスへ書き込み ②
  public void paint(Graphics g) {
    //画面の初期化
    g.clearRect(0,0,width,hight);
    //文字列の描画
    g.drawString("iアプリデモ",5,20);
  }
}
```

▼リスト2　iAppliDemoクラス

```
import com.nttdocomo.ui.*;
/**
 * タイトル:  iアプリのデモ
 * 説明:     Linux Magazine 2001/11月号
 * 著作権:    Copyright (c) 2001
 * 会社名:
 * @author 加藤　大受
 * @version 1.0
 */

public class iAppliDemo extends IApplication {

  //キャンバスの定義
  myCanvas canvas; ①

  public void start() {

  //キャンバスの作成
  canvas = new myCanvas(); ②
  //キャンバスの表示
  Display.setCurrent(canvas); ③
  }
}
```

▼リスト3　iAppliDemo.jam

```
PackageURL = iAppliDemo.jar
AppSize = 865
AppName =
AppVer =
AppParam =
AppClass = iAppliDemo
SPsize =
LastModified = Sat, 15 Sep 2001 16:15:49
KvmVer =
```

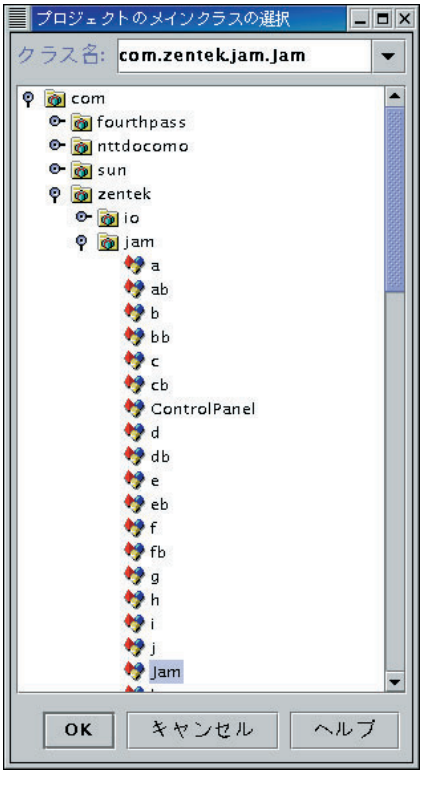
▼リスト4　iAppliDemo.html

```
<html>
<OBJECT declare id="1" data="iAppliDemo.jam"
    type="application/x-jam">
</OBJECT>
</html>
```

から［プロジェクト］－［プロジェクトプロパティ］を選択し、「プロジェクトプロパティ」ダイアログボックスを表示します。次に、［実行］タブをクリックし、［メインクラス］の右にあるボタンをクリックして、プロジェクトのメインクラスの指定を行います。表示された「プロジェクトのメインクラスの選択」ダイアログボックスで、com.zentek.jam.Jamを選択します（**画面10**）。

i-JADEがメインクラスとして設定されると、**画面11**のように表示されます。これで、現在のプロジェクトからi-JADEを呼び出すことができます。

続いて、メインメニューから［実行］－［プロジェクトの実行］を選択して、作成したiアプリを実行してみましょう。ここで、作成したiアプリがコンパイルされます。エラーなくコンパイルが終了すると、i-JADEが起動し、「i-JADE携帯電話エミュレータコントロールパネル」ダイアログボックスが表示されますので、［ファイル］－［iアプリのオープン］を選択し、classesディレクトリ内にあるiAppliDemo.classを選択します。さらに［実行］ボタンをクリックすると、**画面12**のように作成したiアプリがエミュレータに表示されます。



画面10　i-JADEの選択

画面12　iアプリの実行

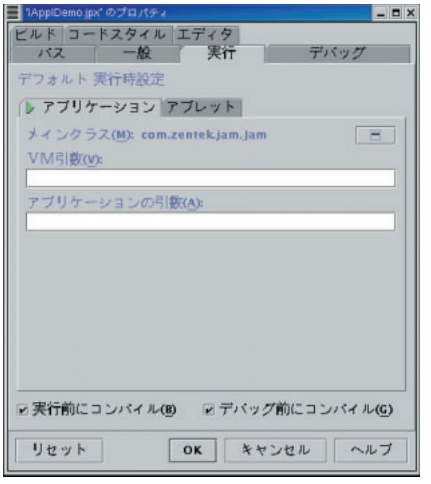
```
$ /usr/java/jdk1.3.1/bin/jar cvfM iAppliDemo.jar iAppliDemo.class myCanvas.class
```

画面13　jarファイルの作成

iアプリの配布準備

まず、作成したファイルを格納するjarファイルを作成します。jarファイルはjarコマンドで作成することができます。jarコマンドのオプションは「cvfM」を指定します（**画面13**）。

jarファイルを作成したら、続いてjamファイルをテキストエディタで作成します（**リスト3**）。AppSizeには作成したjarファイルのサイズを指定し、AppClassにはIApplicationを継承した



画面11　「プロジェクトプロパティ」ダイアログボックス

クラスを、LastModifiedには最終更新日を指定します。

iアプリの配布

最後に公開用のページを作成してWebサーバに転送すればiアプリの配布は完了です。**リスト4**は作成したiアプリの公開を行うiAppliDemo.htmlのページです。<OBJECT>タグのdataパラメータに作成したjamファイルを指定することでiアプリの配布が可能になります。

iアプリの携帯電話へのインストール

実際に作成したiアプリを携帯電話で実行するには、先ほど作成したjarファイル、jamファイル、ダウンロード用のHTMLファイルをアップロードしたWebサーバにアクセスします。

iモード対応の携帯電話がWebサイトにアクセスしたときに<OBJECT>タグを見つけると、そのタグに結びつけられているjamファイルがダウンロードされます。このとき、jamファイルの内容が画面に表示され、ユーザーによるダウンロード確認が行われます。その後、iアプリがダウンロードされ、携帯電話にインストールされます（端末の設定によってダウンロード確認を省略することも可能です）。

●

今回は、iアプリの作成から配布まで細かく説明しました。手順さえ間違えなければそれほど難しくないことが理解できたと思います。次回は図形の描画などについて説明し、少しずつ凝ったiアプリを作成してみましょう。

Column

Windows環境でのiアプリの配布について

Windows環境では、J2ME Wireless SDK for the DoJa（以下、DoJa）を使って配布ファイルを作成することができます。

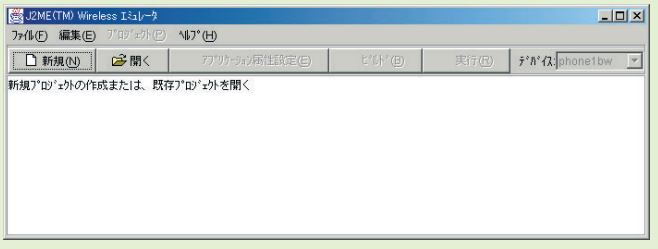
DoJaを使う場合、ソースファイルをWindows上でコンパイルし直す必要があります。このとき、文字コードをシフトJISに変更する必要もあります。

手順は、作成したiAppliDemo.javaとmyCanvas.javaをWindows環境にコピーし、文字コードをシフトJISに変更します。次に、DoJa（**画面14**）を起動し、新しいプロジェクトを作成するために［新規］ボタンをクリックし、「新規作成」ダイアログボックスにプロジェクト名として「iAppliDemo」を指定し、［プロジェクトの作成］ボタンをクリックします。続いて、iAppliDemo.javaとmyCanvas.javaをプロジェクトのソースディレクトリ（C:\J2MEWSDK4DOJA\apps\iAppliDemo\src）にコピーし、［開く］ボタンをクリックして先ほど作成した「iAppliDemo」プロジェクトを読み込みます。

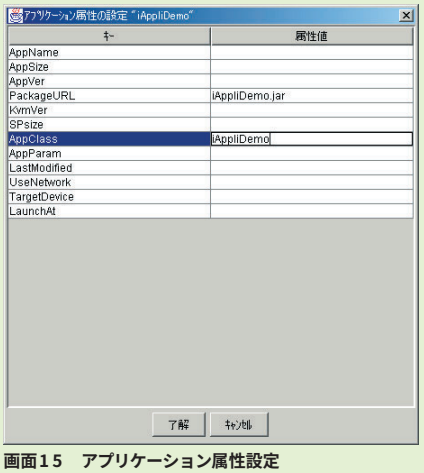
プロジェクトを読み込んだら、［アプリケーション属性設定］ボタンをクリックして、［アプリケーション属性設定］ダイアログボックス（**画面15**）を表示します。

ここで表示されるAppClassキーにはIApllicationクラスを継承したクラスを指定しますので、「iAppliDemo」を指定します。

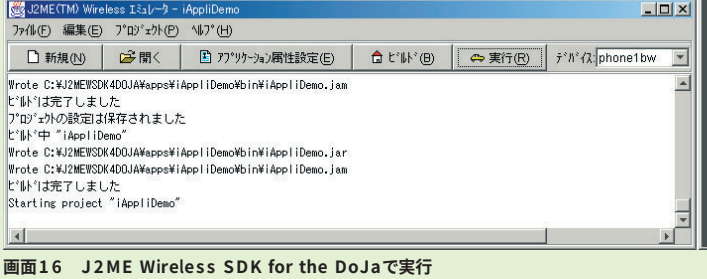
指定が終了したら、［ビルド］ボタンをクリックしてビルドを実行します。ビルドに成功すると、\binディレクトリ以下にiAppliDemo.jarとiAppliDemo.jamが生成されますので、［実行］ボタンをクリックして作成したjarファイルとjamファイルが正しいかどうかを確認してみましょう。i-JADEで起動したときと同じように表示されれば問題ありません（**画面16**）。



画面14　J2ME Wireless SDK for the DoJa



画面15　アプリケーション属性設定



画面16　J2ME Wireless SDK for the DoJaで実行

Software Catch Up

Microsoft Officeを意識した日本語オフィススイート

Hancom Office 1.5J

韓国生まれのHancom Office 1.5Jは、充実した機能を誇るオフィススイートである。また、Microsoft Officeとの親和性が高く、Linuxをデスクトップ環境として使う際の強力なビジネスアプリケーションといえるだろう。

文：塩田紳二
Text : Shinji Shioda

価格	1万5800円
問い合わせ先	ハンコムリナックス 03-3258-3401 http://hancom.co.jp/

Hancom Office 1.5Jは、韓国Hancom Linux, Inc.が開発したLinux用オフィススイートである。以前紹介したときには、Hancom Office for Linuxという名称であったが、今回のパッケージでは若干名称が変更されている。なお、発売元のハンコムリナックスは、韓国Hancom Linux, Inc.の日本法人だ。

発売されたばかりのHancom Office 1.5Jだが、後継であるHancom Office 2.0（英語版）もすでに発表されている。こちらは11月中に英語版が、年内には日本語版がリリースされる予定とのことで、同社のHancom Officeへの力の入れようがうかがえる。

Hancom Office 1.5Jは、以下のようなソフトウェアから構成されている。

①Hancom Word 5.2（画面1）
②Hancom Sheet 1.5.1（画面2）
③Hancom Presenter 1.2.5（画面3）
④Hancom Painter 1.2.5（画面4）
⑤VJE-Delta Ver.3.0 for Linux
⑥研究社 新英和中辞典
⑦15書体のダイナフォント（TrueType）
⑧Hancom Shell（画面5）

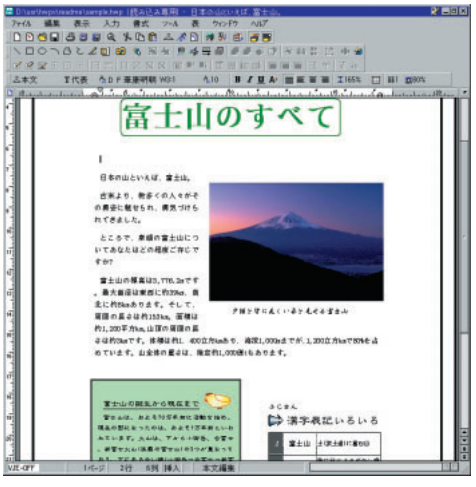
さらに、Hancom Linux, Inc.がカスタマイズしたK Desktop環境「Hancom KDE」も付属する。

このHancom Office 1.5Jは、主要なディストリビューションに対応しているが、Hancom KDEおよび、VJE-Deltaは、Red Hat Linux 7.1、LASER5 Linux 7.1など、特定のディストリビューションのみに対応している。

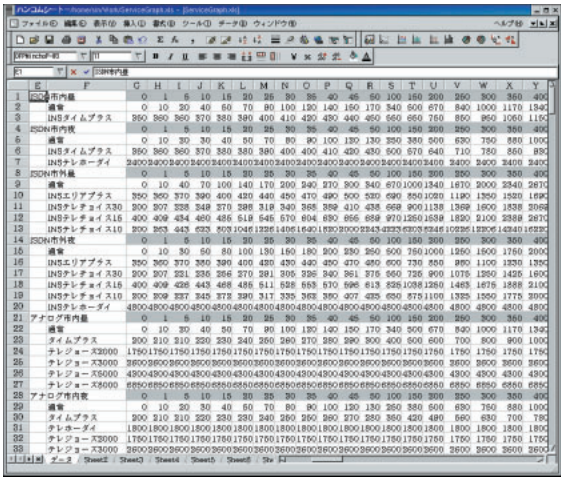
なお、Turbolinuxの場合には付属のインストーラではなく、rpmコマンドを使ってユーザーが手動でインストールする必要がある。

パッケージにはインストール用CD-ROM2枚、マニュアル2冊が含まれている。インストールCD-ROMは、1枚はHancom Office 1.5J用で、もう1枚はHancom KDE用である。

マニュアルも1冊はHancom Office



画面1　Hancom Word（ワープロ）
Hancom Wordはワープロで、WindowsアプリケーションをWINEを使って移植したものだ。



画面2　Hancom Sheet（表計算）
Hancom Sheetは表計算プログラムだ。こちらはLinux用に開発されたプログラムである。

1.5J用で、もう1冊がHancom KDE用となっている。

Hancom Office 1.5Jのマニュアルは15mmほどもある本格的なもので、インストールや4つのオフィスツールについての解説が記載されている。これはリファレンスではなく、使い方を解説したものである。

インストール

前述のようにTurbolinux以外であれば、付属のインストーラを使ってインストールが可能である。インストーラの起動は、インストールCD-ROMにあるスクリプトファイル、「install」を起動して行う。なお、インストールを行うためにはrootでログインしている必要がある。

インストーラはGUIを使ったもので、使用許諾やシリアルナンバーの入力など4つのステップからなる。

このインストーラでは、4つのオフィスツールとHancom Shell、フォント、辞書のインストールのみを行い、Hancom KDEおよびVJE-Deltaのインストールは行われない。また、このインストーラは4つのオフィスツールとHancom Shellをスタートメニューに登録する。

実際にインストールされるものはrpmファイルであるので、付属ドキュメントにあるようにrpmコマンドを直接使ってのインストールも可能である。

VJE-Deltaのインストールは別途、VJE-INSTALLを使って行う。ただし、このインストーラで正しくインストールできるのは、Red Hat Linux 7.1、Vine Linux 2.1、LASER5 Linux 7.1だけである。ただし、VJE-Deltaのバイナリはrpmとなっているため、ほかのディストリビューションについては手動でのインストールも可能だと思われる。

なお、VJE-Deltaをインストールしてしまうと起動スクリプトが/etc/bashrcに記述され、従来の日本語入力システムの設定と共存ができないようになっている。この点は残念である。

Hancom Word 5.2

Hancom Word 5.2（以下、Word）は、韓国のHaansoft（http://www.haansoft.com/japan/。Hancom Linux, Inc.の親会社）が開発したWindows用韓国語ワープロ「アレアハングル」がベースになっている。Windows版のほうは、Hannsoftが「アレアハングルミレニアム」という名称で日本語版も別途販売している。

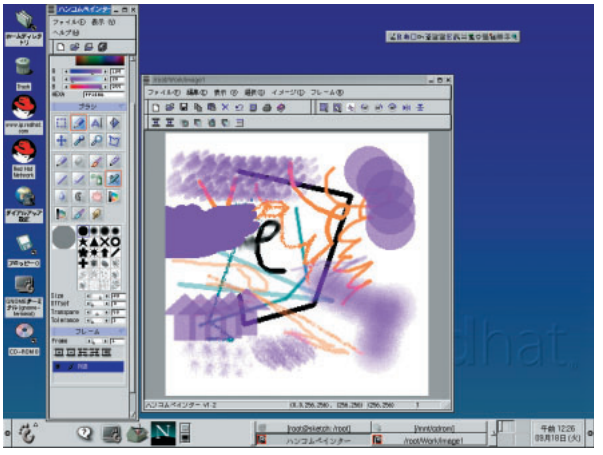
Wordは、Windows版のアレアハングルをベースに、WINEを使ってコンパイルされたLinuxアプリケーションになっている。

Wordのウィンドウは、メニューバーの下にツールバーが並び、文章作成領域の上と左にルーラーが付くもので、ワープロとしては標準的なデザインを持つ（**画面6**）。文書ごとにウィンドウを開くSDI形式になっており、ウィンドウメニューから、Wordウィンドウの整列や、Wordウィンドウ間のフォーカス切り替えなどが可能となっている。ただし、フォーカス切り替えのあと、ウィンドウを手前に持ってくるかどうかはウィンドウマネージャの設定に依存する。

ツールバー（Wordではツールボックスと呼ぶ）はメニュー上に4つあり、それぞれのオン／オフが可能だ。さらに、それぞれのツールバー上のボタンをユーザーが設定することも可能だ（**画面7**）。また、メニューからは直接オン／オフできず、カスタマイズもできないが、インターネット用の「インターネ

画面5　Hancom Shell
Hancom Officeに含まれるオフィスツール4つを簡単に起動させるためのランチャとして機能する。

画面3　Hancom Presenter（プレゼンテーション作成）
Hancom Presenterはプレゼンテーション資料作成プログラムだ。



画面4　Hancom Painter（イメージ編集）
Hancom Painterはビットマップイメージの編集プログラムだ。

ットツールボックス」というツールバーがあり、これは対応するツールバーボタンからのみオン／オフが可能だ。つまり、このツールバーを使うには、4つのメニューバーのどれかにオン／オフボタンを追加し、それを使って表示を行わせる必要がある。

垂直スクロールバーはウィンドウ右側にあるが、水平スクロールバーはウィンドウ下部にあるステータスバーの右側にある。多くのソフトウェアでは、ステータスバーの上に水平スクロールバーがあるが、これはちょっと珍しい。ワープロの場合、文書の横幅をウィンドウと一致させ、1行がすべて見えるような形で編集することが多く、実際に水平スクロールバーのお世話になることがあまりないことを考えると合理的なやり方なのかもしれない。

文書作成領域では、マウスの右クリックによる「コンテキストメニュー」、つまり選択したオブジェクトに応じたメニュー表示と、テキストなどのドラッグによる移動が行える。このあたりは、Windowsの標準的なユーザーインターフェイスを取り入れているようである。特にテキストなどのドラッグによる移動は、慣れてしまうと頻繁に使うため、この機能がないとイライラすることもあるぐらい「中毒性」のある機能である。こうした細かい点をWindowsのスタイルに合わせてくれると、移行の際にストレスを感じることもない。

Wordの文書内にはテキストのほかに、表、図版、テキスト枠などのオブジェクトと、コードと呼ばれる特殊データを入れることができる。コードは、たとえば数式や今日の日付などとして表示されるオブジェクトで、Microsoft Wordなどではフィールドと呼ばれるデータである。これらは、通常では文字列などのように表示されているが、「組版符号」を表示させるとコードとして表示される。

図版は写真データなどのビットマップと、ベクトル図形などが利用できる。また、イラスト集としてベクトル図形データが付属している。どちらも、設定で本文と重ね合わせたり、本文が図版をよけるように並べるといった表示が可能だ。また、ベクトル描画はWord内で図形描画ツールボックスを使うことで描画することもできる。

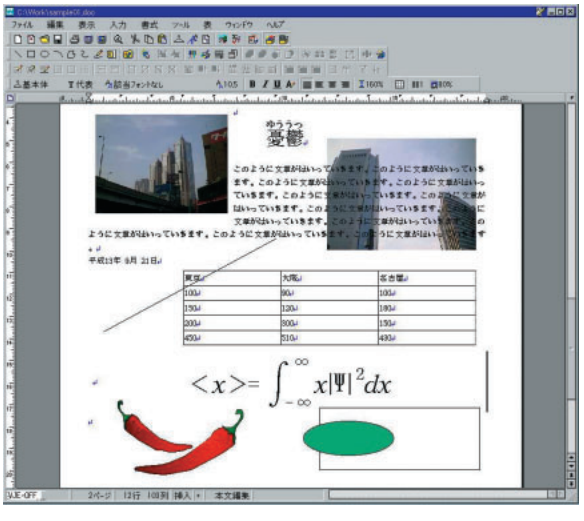
ビットマップデータは、図版の拡大縮小、トリミング、周囲の余白、キャプション（説明文字列）、明るさやコントラストの指定、モノクロ化などが行える。ビットマップそのものの編集や加工はできないが、さまざまな効果を施して印刷を行う場合などに適切な状態にさせることができるだろう（**画面8**）。

長文作成関連としては、目次、索引の作成機能がある。目次や索引の作成は、あらかじめ本文中の目次項目や索引に入れる単語にマークを付けておき、あとから目次（見出しタイトルとページ数）や索引の入った文書ファイルを作成させるものだ。目次はこの方法でもかまわないが、索引は単語ファイルを作って、これを元に本文を検索して自動的に索引を作るような機能が欲しいところだ。

表もオブジェクトとして扱われ、表内の数値を使った計算などが行える。元となったWindows版のワープロは高機能で、オフィススイート化する前は、ワープロ単体でいろいろなことができるようになっていたためこうした機能があるわけだ。こうした機能のほかにも、プレゼンテーション機能が用意されている。これはWordで作成したページに背景を付け、全画面表示にするものだ。これで簡単なプレゼンテーションが行える。

付加機能としては、研究社の新英和辞典を取り込んだ辞書機能がある（**画面9**）。こちらは、英単語の辞書引きができるものだ。検索単語のインクリメンタル検索ができるようになっているのだが、ちょっと動作にもたつきを感じる。特にWord本文との連携機能はないようだが、右クリックメニューで本文テキストのコピー、ペーストが行えるため、文章中の単語1つを調べるのもそれほど難しくない。また、ツールバーボタンで表示中の辞書本文のコピーも可能だ。

このほか、作成中にエラーなどで



画面6　Hancom Wordのウィンドウ
Microsoft Wordとほぼ同じインターフェイスを持つ。なお、イメージや表、数式、ベクトル図形などを文章中に配置することも可能だ。また、日本向けの機能としてルビを振る機能もある。

Wordが異常終了した場合、残っているファイルを元に作成中の文書を復元する機能もあるようだ。評価中に、Wordがフリーズしてほかのウィンドウを含め操作ができなくなったが、このとき、telnetでWordや関連プログラムをkillしたのち再度Wordを起動すると、メッセージを表示し、文書が復元された。特にセーブした時点ということではなく、テンポラリファイルなどに残っていたデータを復元すると思われる。

さて、Windows用アプリケーションとの互換性だが、RTF（MicrosoftのRich Text Format）、HTML、Lotus 1-2-3のワークシートファイル、DOC形式（Microsoft Word形式）、一太郎文書ファイル（J?X形式）、WordPerfect文書ファイル（wps）、テキストファイルの読み込みをサポートしている。

実際にMicrosoft Wordの文章を読み込ませてみたが、段組された文章は段組が解除されてしまった（Word自体には段組みの機能はある）。この現象は、かつてここで評価したHancom Word（Red Hat Linuxバンドル版。本誌2000年12月号掲載）と同じである。おそらく文書変換フィルタの仕様だとは思うが、なるべく忠実に文書を読み込んでもらいたいものである。

ファイルを開くためのダイアログボックスは、Microsoft Wordなどが使っているダイアログボックスに似ており、表示形式の変更やディレクトリ移動、プレビューの表示、検索機能などが実行できる（**画面10**）。

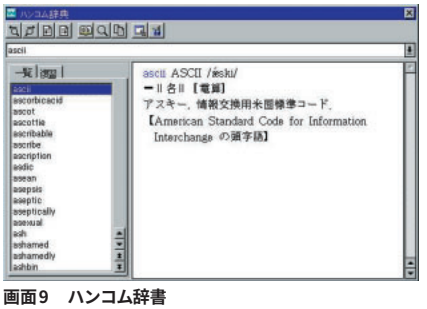
なお、デフォルトの文書用ディレクトリとして、ユーザーのホームディレクトリにWorkというディレクトリを作り、その下のサブディレクトリを「グループ」として管理することができる。これは、ファイルオープンダイアログにタブとして表示され、ディレクトリツリーを使うことなく簡単に開くことが可能になっている。また、このタブには各ユーザー共通のサンプルフォルダやファイルの利用履歴などの表示機能もある。

日本語入力は、付属のVJE-Deltaだけでなく、Cannaにも対応しており、kinput2を使う標準設定でも日本語入力が可能だ。

ただ、残念なことに本文文章に挿入する形で日本語入力を行うと、入力文字が元の本文テキストの上に重なる形で表示されるため、使い勝手がいまいちである（もっとも完全に覆い隠すわけではないので、下の文章を見ることはできるが）。

このあたり、かつてのDOSエディタなどならともかく、ワープロなら変換途中の文書を文中に直接挿入するような形にして、うしろの文章を隠さないようにしてほしいところである。

画面7　ツールバーカスタマイズ
Hancom Wordではツールバーは4つしかないが、各ツールバーに配置するボタンをユーザーが定義できる。



画面9　ハンコム辞書
研究社の新英和中辞典を入れたハンコム辞書。単語のインクリメンタル検索が可能。

Hancom Sheet 1.5.1

Hancom Sheet（**画面11**。以下、Sheet）は、表計算プログラムである。ウィンドウ全体の印象は、Microsoft Excelなど、一般的な表計算ソフトのものと変わらない。ウィンドウはメニューバーの下にツールバーがあり、その下に数式バー（セルへの数値や式の入力を行う部分）があり、ウィンドウ下部にはステータスバーがある。

セル部分は、A1形式（行を数字で、列をアルファベットで表す形式。Lotus 1-2-3が採用していた形式で、現在では表計算の一般的なセル指定方法）で見出しが付く。選択中のセルは、太い枠で表される。また、1つのデータファイルは、複数のワークシートから構成されるブック形式（Microsft Excelが採用しているワークシートファイルの論理的な構成）である。

画面8　イメージ修正
文書中に配置したビットマップ画像に対して、色やコントラストなどの指定が可能。

画面10　Hancom Wordファイルオープンダイアログ
文書オープンダイアログでは文書のプレビューが表示され、ディレクトリ作成などの作業も行える。

このSheetは、Wordと違って、MDI形式（メインウィンドウの中にデータファイルに対応する複数の子ウィンドウが配置され、子ウィンドウがメインウィンドウからはみ出さない形式）になっている。このため、メニューバー右側には、ワークシートウィンドウを制御するためのボタン（次のウィンドウ、ウィンドウ最大化、ウィンドウのクローズ）が並ぶ。また、このワークシートウィンドウは、アイコン化してメインウィンドウの下部に並べることも可能だ。このときのアイコンは、Windowsなどでおなじみの短冊状のものになる（**画面12**）。

Sheetには、3つのツールバー（SheetではWordと違ってツールボックスとは呼ばないようである）があり、その位置を動かしたり、メインウィンドウの4つの辺に沿って配置するなどの操作が可能だ。ただし、フローティング状態（ツールバーを自由な位置に置けるようにすること）にすることはできない。

ワークシートウィンドウ上では、マウスの右クリックでポップアップメニューが表示される。これは、選択されているものに応じてメニュー内容が変化するコンテキストメニューである。

選択されているセルは太枠で表示されるが、枠右下にあるハンドル（黒い矩形）をドラッグすることで、選択範囲を使ったセル内容のコピーが行われる。このとき、数値が含まれていれば、その数値を増加させる形でコピーが行われる。たとえば、1が入っているセルだった場合、2、3、4……となるわけだが、最初に10、20という値を持つ2つのセルが選択されてれば、30、40、50……と最初の選択中のセルから等差数列を作り出してコピーが行われる。このあたりは、Microsoft Excelの動作とほとんど同じである。ほとんどというのは、Microsoft Execl 2000から、値が数値で、単独のセルをハンドルを使ってコピーした場合には、数値を増加せずにそのままコピーするように変更されたからである。

また、枠のハンドル以外の部分をドラッグするとセルの移動となる。このとき、Microsoft Excelでは、Shiftキーを押しながら操作するとドラッグ先への挿入移動となり、Ctrlキーを押しながら行うとコピー（元のセルはそのまま）となるが、こちらの2つの動作は実装されていないようである。

できれば、このあたりまで合わせてくれると移行の際のストレスが小さいのだが……。また、Sheetは、マウスのホイールによるスクロールには対応していないようである。WordやHancom Presenterなど、ほかのツールは対応しているのにSheetはマウスのホイールを使ってスクロールすることができないのは残念だ。

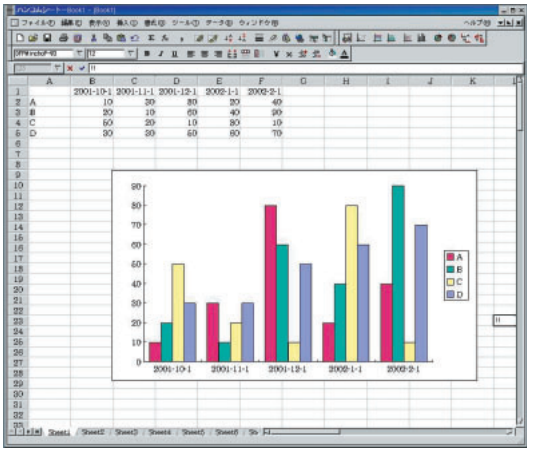
ワークシート上には、オブジェクトとしてはグラフを置くことができるが、ベクトル図形やビットマップ図形を置く機能はないようだ。グラフは、ワークシート上で範囲を選択してボタンを押せばウィザードが起動し、質問に答える形で作成を行う（**画面13**）。

セル内には、数値、文字列、数式のほかにハイパーリンク（URL）を入力することができる。また、セルごとにメモを付けることもできる。メモの付いたセルは、右肩の部分に赤い三角が表示され、カーソルをセル上に重ねると矩形のメモウィンドウがセルを指す矢印とともに表示される。

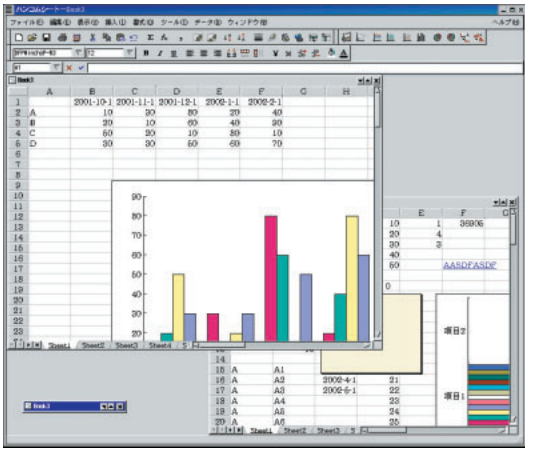
セルに対して最初に「=」で入力を始めると、数式の入力となる。セルの座標はA1形式で行い、Microsoft Excelでは指定できるR1C1形式（もともとはマルチプランのセル座標指定方法）では指定ができない。

セルに対しては、背景色や文字色、表示書式などが設定できる。また、条件により書式を変更する「条件付き書式」もサポートされている。

Windows上でMicrosoft Excelで作成した8Mバイトのワークシートファイルを読み込ませたところ、動作はかな



画面11　Hancom Sheet
Hancom Sheetはグラフの作成が可能な表計算プログラムで、Microsoft Excelのファイル読み込みなども可能。



画面12　MDI形式を採用したHancom Sheet
Hancom Sheetは複数のワークシートファイルを開いたとき、それぞれのウィンドウがメインウィンドウの子ウィンドウとなるMDI形式のウィンドウを持つ。

り遅くなるものの、データ自体はきちんと読み込まれた（**画面14**）。ただし、元のファイルにあった「条件付き書式」が欠落してしまった。これは、再度Sheet上で指定を行えばきちんと動作した。これは、前回の評価版のときには正しく読み込めなかったものである（β版のためこのテストについては触れなかった）。この結果を見る限り、Sheetのプログラムとしてのクオリティはかなり上がったようである。実際、大きなファイルを読み込むとウィンドウ更新の時間はかなりかかるようになるのだが、データを読み込んでも書式以外は欠落しなかったし、計算もちゃんと行える。あとは、速度的なチューニングを残すのみという感じである。

なお、Sheetは、Microsoft Excelファイルのほかにテキスト形式やCSV形式（カンマ区切りファイルなど）にも対応している。

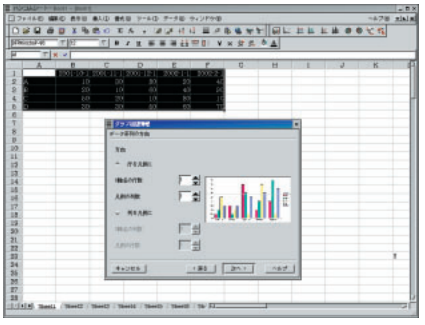
Sheetのオンラインヘルプ（**画面15**）はプログラムがWordとは別で、左側に目次や索引、検索などの機能を実現する部分があり、右側にヘルプ本文を表示する形式になっている。

Hancom Presenter 1.2.5

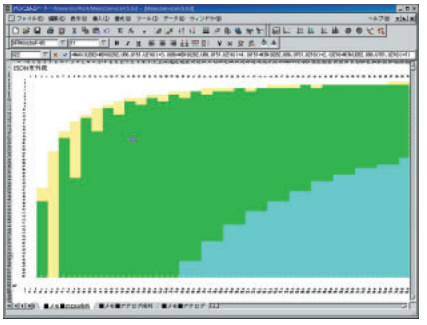
Hancom Presenter（**画面16**。以下、Presenter）は、プレゼンテーション資料作成ツールである。このツールでは、作成したスライド（実際には、OHPや画面イメージなのだが、かつて35ミリスライドを使ってプレゼンテーションしていた頃からの名残で、多くのソフトは作成する1枚のイメージをスライドと呼ぶ）を、オンラインでプレゼンテーションすることも、プリンタで印刷することもできる。また、スライドデータを元に配布資料や発表者ノートなどを作成することができる。

スライドは、背景、ページレイアウトと、そこに配置されるテキストや表、グラフ、図などから構成されている。図には、クリップアート、ビットマップイメージ、そしてオートシェイプ（あらかじめ定義されたベクトル図形）、線や多角形などのベクトル図形などが利用できる（**画面17**）。

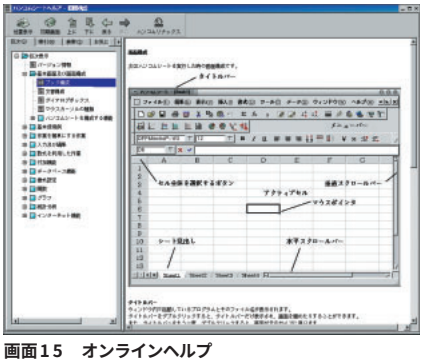
新規にプレゼンテーションを作る場合、レイアウトを指定して1枚のスライドを作ることも、背景を指定してレイアウトのないスライドを作ることもできる（それぞれ、あとから背景やレイアウトが指定できる）。さらに、あらかじめ用意されたスライドの組み合せ



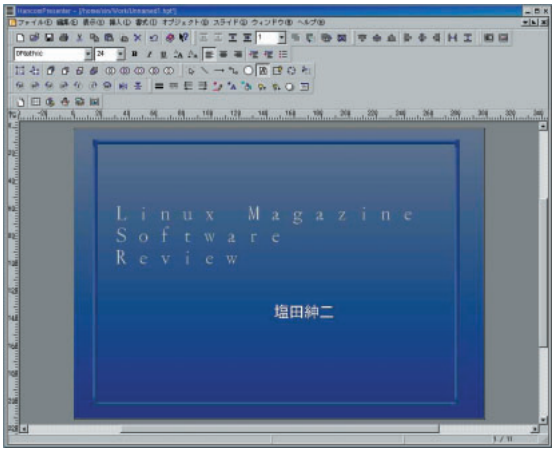
画面13　グラフウィザード
ワークシート上でグラフ化したい範囲を選択してツールバー上のボタンを押せばウィザードが起動し、グラフの作成が簡単に行える。



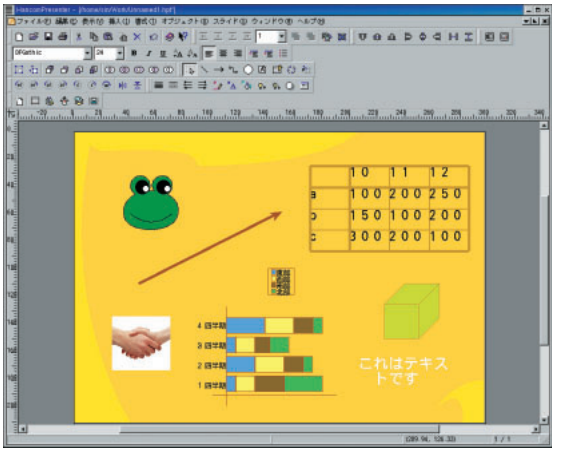
画面14　Microsoft Excelファイルの読み込み
Microsoft Excelで作成した8Mバイトほどあるワークシートもほとんど正しく読み込めた。画面の色が付いている部分は条件付き書式で、この部分は読み込み時に欠落したが、再度設定することで正しく動作した。



画面15　オンラインヘルプ
Hancom Sheet、Hancom Presenter、Hancom Painterのヘルプは、Hancom Wordとは別のヘルプ表示プログラムを持っている。



画面16　Hancom Presenter
Hancom Presenterは オンライン、またはプリンタを使ったスライドの作成が可能なプレゼンテーション資料作成プログラムだ。



画面17　Hancom Presenterで利用できるオブジェクト
スライド上にはクリップアート、表、グラフ、ビットマップ、テキストなどを配置することができる。

（たとえば、販売戦略用、事業計画用など。これをPresenterではテーマ別書式と呼んでいる）をひな形としてプレゼンテーションを作ることもできる（**画面18**）。

アプリケーション全体のイメージとしては、描画ソフトに近い部分がある。というのは、スライドの並べ替えを行うようなスライド一覧表示や、タイトル、本文テキストのみを表示するアウトライン表示といった表示を行うことができず、スライドを1枚1枚表示させつつ、編集を行う必要があるからだ。

ウィンドウは、Sheetと同じくMDI形式で、ツールバーなどの機能も同じである（ただし、Presenterではツールボックスという表記になっている）。おそらくは、基本的な部分でコードを共有しているのだと思われる。

中心となる機能は、ベクトル図形の編集である。基本的に配置されるオブジェクトのほとんどは、このベクトル図形の一種で、タイトルや本文テキストもベクトル図形によるテキスト枠として扱われているからである。ベクトル図形には、テキストを付けることができ、その枠や背景を透明にすれば、スライド上は本文テキストのように見えるわけだ。スライドのレイアウトとは、こうしたベクトル図形を最初から配置したものなのである。

画面18　プレゼンテーションの作成
スライドはレイアウト、背景書式を指定して作成できるほか、あらかじめ作成されたプレゼンテーション（複数スライド）を使って作成することもできる。

これとは別に、各スライドに指定されるのが背景で、こちらもあらかじめ用意されたものから選択する。

残念なことにSheetとの連携機能がなく、Presenterの機能を使って表やグラフを作らなければならない。プレゼンテーション用資料は多くの場合、さまざまな基礎データを加工したものを使って作られるため、ほかのソフトで作成したデータが利用できないのが残念である。

Hancom Painter 1.2.5

Hancom Painter（**画面19**。以下、Painter）は、ビットマップイメージの作成ソフトである。簡単にいえば、GIMPのようなソフトである。

ビットマップイメージをゼロから描画して作成することはもちろん、すでにあるJPEGなどのビットマップデータを読み込んで加工するフォトレタッチ的な利用も可能である。さらに、GIFアニメーションの作成機能もあるため、HTMLなどに埋め込むGIFアニメーションなどを作る場合にも利用できる。ただし、イメージに対してフィルタをかけるような機能はないために、フォトレタッチといってもイメージの加工にはちょっと限界がある（明るさ、コントラスト、γ値を変えることなどはできる）。どちらかというと描画のほうに力点があるといった感じだろう。

ウィンドウは、以下のような3つの独立したウィンドウから構成されている。

- **ペイントツールボックス**
  メニューやツールボックス
- **イメージ作業ウィンドウ**
  編集中の画像が表示されるメインのウィンドウ
- **ファイルリストウィンドウ**
  画像ファイルのサムネールを表示

このあたりの感じもGIMPと似たところがある。Windowsなどでは、Photoshopのようにメインウィンドウの中に画像ウィンドウやツールパレットが並ぶものもあるが、複数のウィンドウに分かれているこちらの形式のほうがツールや画像の配置が自由で、作業がやりやすい。特に、複数画像を同時に処理するような場合には画像表示ウィンドウの配置に自由度が高い。

ペイントツールボックスは、メニューバーとツールボックス（SheetやPresenterと同じく4つの辺に沿って配置が可能）および、以下の4つのツール群からなる。

- **インフォメーション**
  イメージ上でのポインタの位置などを表示
- **カラー**
  色指定を行う
- **ブラシ**
  描画ツールやブラシ形状を指定
- **フレーム**
  GIFアニメーションのフレームを指定

このツール群はペイントツールボックスウィンドウにあり、個々に表示、折り畳みが可能だ。ただし、この順で上下に並び、順番の入れ替えはできないようである。

イメージ作業ウィンドウにもメニューバーやツールボックス（3種。こちらも配置が変更できる）があるが、ここで実行できるのはイメージの回転や明るさ、コントラスト変更など、イメージ全体に渡って行われる処理である（**画面20**）。

ファイルリストウィンドウは、イメージ画像を管理するためのもので、

JPEGなどの画像ファイルをサムネイル形式で表示できる（ただし、画像ファイル以外はフォルダしか表示されない）。ウィンドウの左側はディレクトリツリーになっており、ディレクトリ間の移動が簡単に行える。もちろん、ここから画像を指定してPainterで開くこともできるし、画像の削除やコピーといった作業も行える。

Painterは、ネイティブ形式であるhifファイル以外には、JPEG、GIF、BMP（Windowsのビットマップ形式）、XBM/XPM、PNG、PBM/PPMといったファイル形式に対応する。読み込みはもちろん、保存時にもファイル形式の指定が可能だ。

最近のフォトレタッチツールでは、同系色の部分を選択する「マジックワンド」と呼ばれる範囲選択機能が付いていることが多いが、このPainterにもその機能がある。これを使うと、複雑な形状の切り抜きが可能で、たとえば写真に写っている人物だけを切り抜くといった使い方ができる。

描画機能はどちらかというとペンやエアブラシ、スプレー、ボカシ、にじみといった「水彩的」な機能が多く、単純に丸や四角を書くようなツールはない。挿絵的なイラストを書くなどの感じで、説明的な絵を描くのには向いていないだろう（そういう用途には、Presenterにあるベクトル描画機能がある）。

GIFアニメーションは、1つのファイルの中に複数のGIFイメージが入ったものである。PainterのGIFアニメーション作成機能は、フレームごとに画像を作成し、それらをまとめてアニメーションを作成するもので、関連機能としてはフレームの作成、削除、順番の入れ替えなどがある。

オフィスツール全般

現時点では、Hancom Officeの各ツール間の連携機能はテキストのコピー程度しかない。このため、各ソフトが表やグラフの編集や作成機能を持つという、機能の重複がある。たとえば、Sheetでグラフを作ったとしても、それをPresenterに入れることはできず、Presenterで別途グラフを作成しなければならない。これはかなり大変な作業でもある。

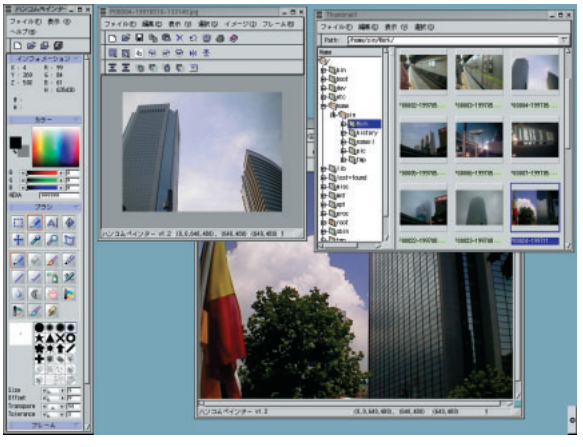
このあたりをWindowsのアプリケーションのように、データのリンクやデータの埋め込みを使って行わせるためには、OS側にそのための機能が必要である。Windowsでは、COMと呼ばれるオブジェクトインターフェイスを使うことでこれを可能にしている。KDEやGNOMEではこのあたりのサポートを行っている最中で、Hancom Officeも次期バージョンはKDEの機能を使ってこれを実現する予定だという。

なお、Wordでは、ヘルプとメニュー項目や機能などの間に相違があり、メニューにある機能の解説がなかったり（辞書機能）、ヘルプには解説があるものの、メニュー項目がないといった部分があった（マクロ機能など）。どうも、Windows版のヘルプをそのまま持ってきているような感じである。また、Wordのみヘルプ表示プログラムが別になっており、利用方法も違っている。

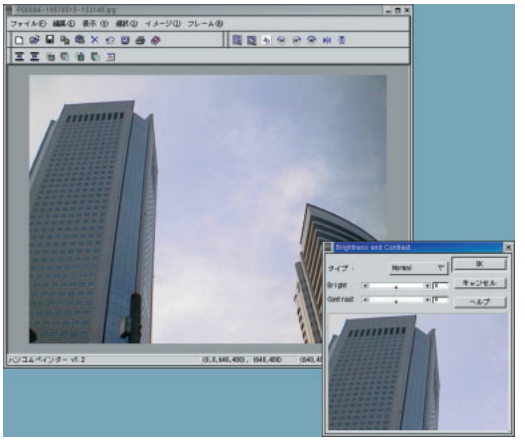
付属のマニュアルは丁寧なものだが、リファレンスがなく、このあたりはオンラインヘルプに頼らねばならない。

今回の評価は、Red Hat Linux 7.1で行ったが、評価中にプログラムが突然落ちてしまうようなことはほとんどなく、全体としてはクオリティが上がっているように感じる。しかし、前述のようにWordにはヘルプとプログラムの間に相違が見られるなど、製品としては少しツメの甘い部分もある。

なお、前述した次期バージョンのHancom Office 2.0では、メーラーやスケジューラ、データベースなどが追加され、8つのソフトウェアで構成されるオフィススイートとなる予定だ。また、すべてLinuxネイティブなコードとなるとのことで、安定度や動作速度の向上も期待される。



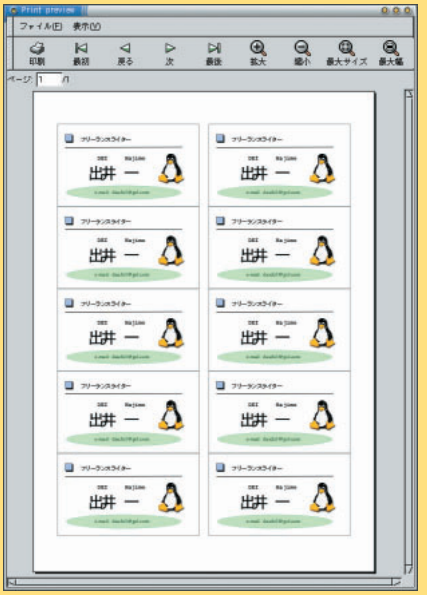
画面19　Hancom Painter
Hancom Painterはビットマップイメージ編集プログラムで、写真のレタッチなどにも利用できる。



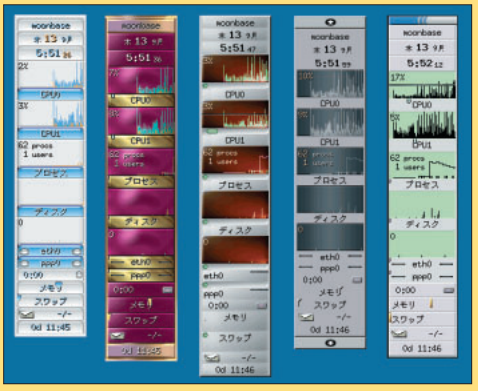
画面20　イメージ作業
イメージ作業ウィンドウでは、明るさ、コントラスト調整など、画像全体に関する作業のためのメニューがある。

Free Application Showcase

文：出井 一
Text : Hajime Dei



gLabels　P.130



GKrellM　P.127

Geki5　P.138

※紹介したソフトは、すべて付録CD-ROMに収録されています。

CPU使用率などの各種情報をグラフ表示

GKrellM

バージョン：1.2.2　　ライセンス：GPL

http://web.wt.net/~billw/gkrellm/gkrellm.html
http://www.muhri.net/（テーマ）

GKrellMは、CPUの使用率やイーサネットのパケット流量、メモリやスワップの使用量などを監視して、リアルタイムにチャート（グラフ）として表示するモニタリングソフトだ。メールチェックやCD-ROMのマウント機能を内蔵しているのに加え、プラグインを使って機能を拡張したり、テーマ（スキン）によって外見を変えることも可能だ。表示内容はダイアログで柔軟にカスタマイズできる。実行にはGTKが必要だ。

●ビルドとインストール

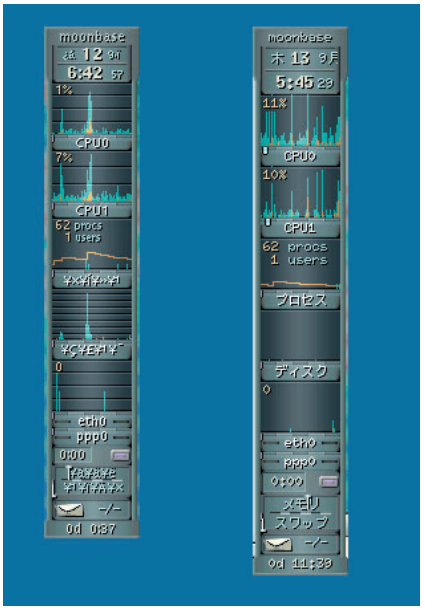
GKrellMは、ファイル一式をtar＋gzipしたtarボールのほか、RPMパッケージも配布されている。ただし、このRPMバイナリパッケージは、日本語カタログなどを含まない英語版だ。以下の手順で国際化対応版をビルドしたほうがよいだろう。

tarボールを利用する場合は、ビルドの前に「./enable_nls」を実行すれば、国際化対応の準備が整う。あとは、「make」でビルドし、「su」でrootになってから「make install」としてインストールすればいい。

一方、RPMを利用する場合は、rootになった状態で「rpm -i gkrellm-1.2.2-1.src.rpm」としてソースパッケージをインストールし、/usr/src/redhat/SPECSに展開されるgkrellm.specをエディタで修正する。

修正点は以下の3カ所。まず、35行目付近の「%build」という行の次に、「./enable_nls」という行を挿入する。続いて、44行目付近の「make install」で始まる行の末尾に「 LOCALEDIR=$RPM_BUILD_ROOT%{_datadir}/locale」を追加する。最後に、54行目付近の「%{_includedir}…」という行の次に、「%{_datadir}/locale/*/LC_MESSAGES/gkrellm.mo」という行を挿入すれば準備完了だ。

あとは、「rpm -bb gkrellm.spec」とすれば、/usr/src/redhat/RPMS/i386に国際化対応のバイナリパッケージが作成される。



画面1　縦長のウィンドウにCPU使用率などのチャートが表示される。

●1秒ごとに情報が更新される

以下の説明は、国際化対応のGKrellMに基づいているため、英語版を利用する場合はメニューの名前などを適宜読み替えてほしい。

画面2　各チャートの表示形式などはこのダイアログで変更できる。

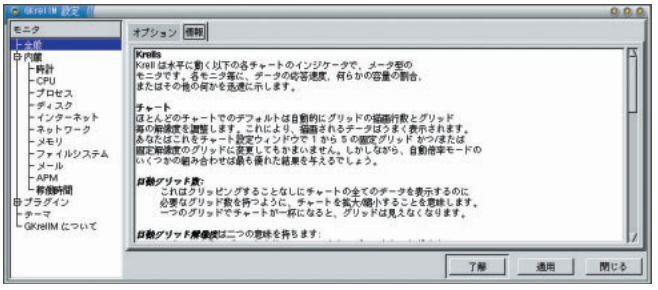
ktermなどのコマンドラインから「gkrellm&」として起動すると、CPU／プロセス／ディスク／イーサネットのチャート（グラフ）や、メモリ／スワップのメーターなどが、縦長のウィンドウに並んで表示される（**画面1左**）。日本語のパネル名（「プロセス」など）が文字化けしているが、後述する設定により修正できる（**画面1右**）。

各チャートは1秒単位で更新され、縦軸の縮尺は測定値によって自動的に切り替わる。また、どのチャートもシアンとオレンジの2色で異なるデータを表示する。たとえば、CPUチャートではユーザータイムとシステムタイム、ディスクチャートではリードとライトといった具合だ。

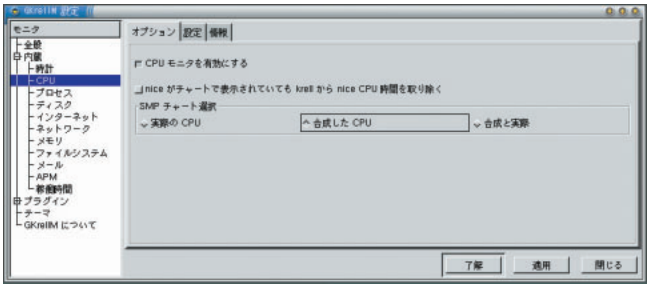
なお、各チャートの左上に表示される数値情報（チャートラベル）は、チャートをクリックすれば消すことができる。また、チャートを右クリックして設定ダイアログを開けば、チャートの表示形式や解像度、高さなどを柔軟に変更可能だ（**画面2**）。

●設定内容のカスタマイズ

ウィンドウ上部のホスト名が表示されている部分を右クリックして、メニューから[設定]を選択すると、GKrellMの設定ダイアログが開く。左側にジャンルツリー、右側に設定用のタブ付きページという構成だ。国際化対応のGKrellMでは、項目名や情報の内容が



画面3　ジャンルツリーとタブ付きページで整理された設定ダイアログ。



画面4　内蔵機能は、表示の有無やパラメータなどを個別に変更可能だ。

日本語で表示される（**画面3**）。

たとえば、ウィンドウの横幅を変えるには、左側のツリーの一番上にある[全般]を選択し、右側のタブ付きページを[オプション]ページに切り替えて、[GKrellM幅]の値を変更する。初期設定は小さめなので、もうちょっと大きな値にするといいだろう。

また、ツリーの[内蔵]以下の項目には、時計やCPU、プロセスなどの内蔵機能が並んでいる。ここでは、チャート・メーター表示の有無やパラメータの変更、パネルクリック時に実行するコマンドなどを個別に設定できる。必要ないと思われる機能を非表示にして、ウィンドウの表示をすっきりさせると使いやすくなるだろう。

たとえば、初期設定ではCPUの数だけ表示されるCPUチャートをひとつにまとめるには、ツリーの[内蔵]－[CPU]を選択し、[オプション]ページの[合成したCPU]をチェックする（**画面4**）。また、[CPUモニタを有効にする]のチェックを外すと、CPUチャート自体がウィンドウから除外される。

なお、こうした設定は、各ユーザーのホームディレクトリにある.gkrellmディレクトリ以下に保存され、次回起動時に自動的に復元される。設定を初期状態に戻すには、「rm -rf ~/.gkrellm」として削除すればいい。

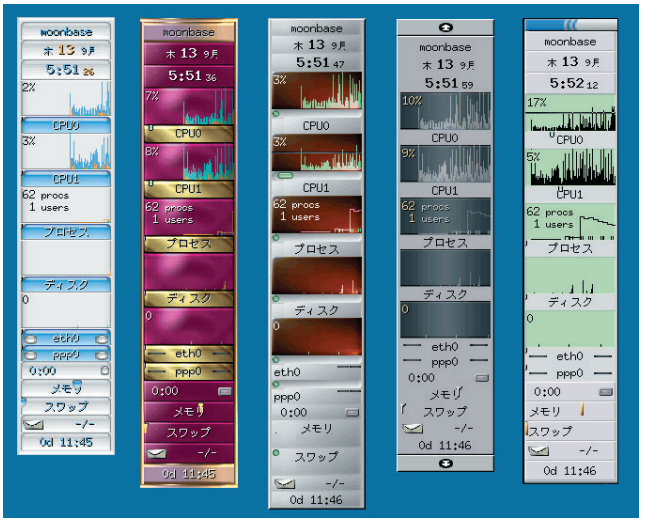
●テーマを導入して見栄えを変える

GKrellMは、テーマ（スキン）と呼ばれる一連の画像ファイルを導入することで、簡単に外見を変更できる。テーマサイトの「muhri.net」には160種類以上のテーマが登録されているので、気に入ったものをダウンロードしよう。本誌付録CD-ROMにも、数個のテーマを収録している（**画面5**）。

これらのテーマをGKrellMに組み込むには、ホームディレクトリの.gkrellm/themesディレクトリで、各テーマのtarボールを展開すればいい。

設定ダイアログのツリー中の[テーマ]を選択すると、利用できるテーマの一覧が[テーマ]ページに表示され、テーマ名をクリックするだけで外見を切り替えられる。また、GKrellMのウィンドウをクリックしてからPキーやNキーを押すと、前後のテーマにすばやく切り替え可能だ。

なお、テーマを作成するためのリファレンス（英文）は、GKrellMのサイトやtarボール内にあるので、オリジナルのテーマを作成する際には、参考にしよう。

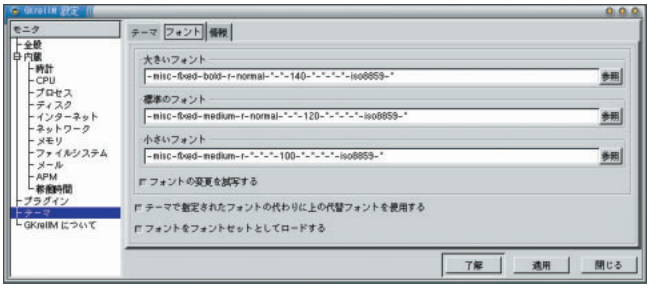


画面5　テーマ（スキン）を使うと、簡単に外見を切り替えられる。

●パネル名の文字化けを解消する

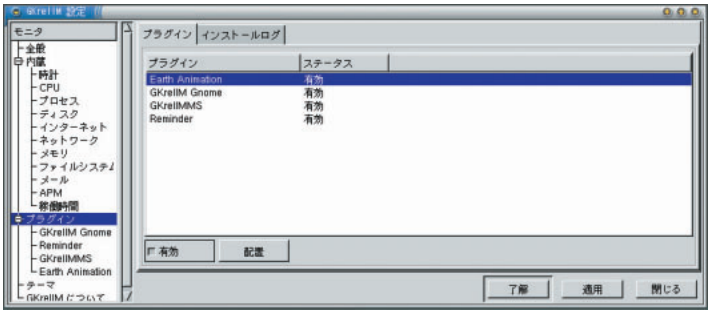
GKrellMの各テーマには、パネル名の表示に使うフォント情報が含まれる。たいていのテーマは海外で作られているので、設定されているフォントは英語フォントだ。

一方、GKrellMの日本語カタログには、「プロセス」や「ディスク」といっ



画面6　日本語のパネル名の文字化けを解消するには、この設定が不可欠。

画面7　地球の表示や覚え書きの記録、XMMSの制御をプラグインで行う。



画面8　リスト中のプラグインは[有効]をチェックすると組み込まれる。

た日本語のパネル名が含まれており、英語フォントでこれらを表示すると文字化けしてしまう。

文字化けを解消するには、GKrellMに用意された強制的なフォントの差し替え機能を利用する。設定ダイアログのツリー中の[テーマ]を選択し、右側の表示を[フォント]ページに切り替えよう。下部の2つのチェックボックスをどちらもチェックし、3種類のフォント名の先頭部分を「-misc-fixed-」に変更すればいい（**画面6**）。

これで、「プロセス」や「ディスク」といったパネル名、上部のカレンダー部分などの文字化けが解消し、日本語で表示されるようになる。

●プラグインのインストール

テーマ（スキン）が外見を変更するのに対し、GKrellMの機能を拡張するのがプラグインの役目だ。

GKrellMのWebサイト内にあるPluginsページには、現時点で40個程度のプラグインが紹介されており、SETI@Homeや天気予報のモニタリングなど、さまざまな機能をGKrellMに追加できる。

本誌付録CD-ROMには、GNOMEとの親和性を高める「GKrellM-Gnome」、マルチメディアプレーヤのXMMSを制御する「GKrellMMS」、小さな地球がアニメーションする「Earth Animation」、アラーム機能付きの覚え書き「gkrellm-reminder」の4つを収録した（**画面7**）。いずれも、rootになった状態でtarボールをビルドすればいい。なお、GKrellMMSのビルドにはxmms-develパッケージが必要だ。

プラグインのインストール先は2種類ある。全ユーザーが使えるようにするには、rootになった状態で/usr/lib/gkrellm/pluginsディレクトリにコピーする。一方、特定のユーザーだけ使えればいいなら、各ユーザーのホームディレクトリに作成された.gkrellm/pluginsディレクトリにプラグインをコピーすればいい。

なお、プラグインの読み込みはGKrellMの起動時に行われるため、インストール後はGKrellMをいったん終了し、再度起動する必要がある。

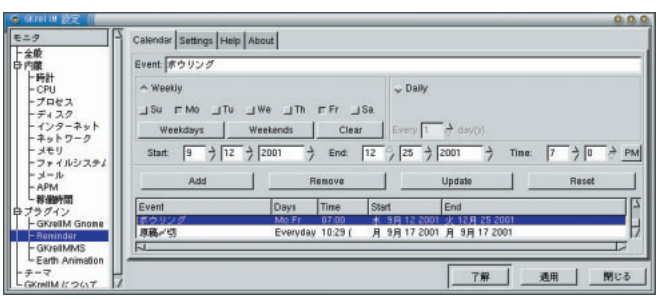
●プラグインの組み込みと設定

設定ダイアログを開いて、ツリー中の[プラグイン]をクリックすると、インストール済みのプラグイン一覧が表示される（**画面8**）。組み込みたいプラグインをクリックで選択し、下部にある[有効]をチェックしよう。[配置]ボタンを押せば、プラグインの表示位置も変更できる。

[プラグイン]以下のツリーには、組み込んだプラグインの名前が表示され、それぞれの設定を行える。設定項目を1つも持たないプラグインから、複数の設定ページを持つプラグインまで内容はさまざまだ。

たとえば、覚え書きの機能をGKrellMに追加するgkrellm-remiderの設定ページでは、イベントごとに日時と表示する文字列の組み合わせを登録できる（**画面9**）。一度だけの予定はもちろんのこと、何日にもわたる予定や、特定の曜日に繰り返し起こる予定なども柔軟に設定可能だ。

指定した日時の15分前（変更可）になると、パネル上のiマークが赤く点滅して知らせてくれ、パネルをクリックするとイベントの内容が別ウィンドウに表示される（**画面10**）。



画面9　gkrellm-reminderに日時とイベントの組み合わせを設定する。

画面10　指定した日時の15分前にこうしたウィンドウで知らせてくれる。

住所ラベルや名刺などを作成・印刷する

gLabels

バージョン：0.1.2　　ライセンス：GPL

http://snaught.com/glabels/

gLabelsは、封筒やフロッピーなどに貼るラベル類や名刺などをグラフィカルな環境で作成し、各種シートにプリンタで印刷するソフトだ。ラベル上には文字列や図形、画像、バーコードなどを自由に配置でき、日本語の文字列も問題なく扱える。初期設定で用意されたシートは外国製品ばかりだが、XMLを利用して国産のシートの設定を追加することも可能だ。動作にはGNOME／GTK+が必要となる。

●ビルドとインストール

gLabelsのビルドには、GNOME／GTK+のほか、gnome-print、db1、VFlib2、freetypeの各ライブラリが必要になる。

これらのライブラリは、たいていのRed Hat系ディストリビューションでは最初からインストールされている（インストール時の設定によって異なる）。ただし、本体のパッケージだけでなく、インクルードファイルなどを含むdevelパッケージ（gnome-print-develなど）も必要になるので、忘れずにインストールしておこう。

gLabels自体は、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」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

また、tarボールに含まれるSPECファイルを利用して、RPMパッケージを作成することもできる。「rpm -tb glabels-0.1.2.tar.gz」とするとバイナリパッケージが作成されるので、「rpm -Uvh /usr/src/redhat/RPMS/i386/glables-0.1.2-1.i386.rpm」としてインストールすればいい。

●起動とシートの選択

ktermなどのコマンドラインで「glabels&」とするか、GNOMEメニューの[プログラム]－[アプリケーション]－[gLables]を選択すると、スプラッシュ画面とともにgLabelsのウィンドウが開く（**画面1**）。

まずは、ツールバー左端の[New]ボタンを押して、使用するシートの種類を選択しよう。住所ラベルやフロッピー用ラベル、名刺カードなど18種類の市販シートが用意されている。

これらはいずれも外国製品だが、Avery社（http://www.avery.com/）の、プリンタ印刷用ラベル用紙は、デファクトスタンダードと言ってもよく、国内でも比較的入手しやすい。また、ラベル設定ファイル（labels-config.xml）を修正すれば、国産製品も利用可能だ。ファイルの内容はXML形式で、シートの名称やラベルのサイズ、マージン、縦横に並べる数などの情報をテキストで記述すればいい。

ラベル設定ファイルは、インストール前ならtarボール展開先のdataディレクトリ、インストール後は/usr/local/share/glabelsディレクトリ（RPMを利用した場合は/usr/share/glabels）にある。

●ラベル上に文字列を配置

左側のツールバー上から2番目の[T]ボタンを押してから、ラベル上でクリックすると、その位置に「Text」という文字列が配置される。位置はドラッグで自由に変更可能だ。

文字列をダブルクリックすると、文字列の内容やフォントを変更するためのダイアログが開く。

表示する内容は、[Text]ページで入力する。複数行にわたる文字列や、日本語を含む文字列も問題なく入力や編集を行える（**画面2**）。

一方、[Appearance]ページでは、フォントの種類や大きさ、文字色、内容が複数行にわたる場合の配置（中央寄せなど）を設定できる。なお、文字列に日本語が含まれる場合、日本語フォント（GothicやMinchoなど）を選択しないと、印刷時に文字の位置が大幅にずれてしまうので注意しよう。

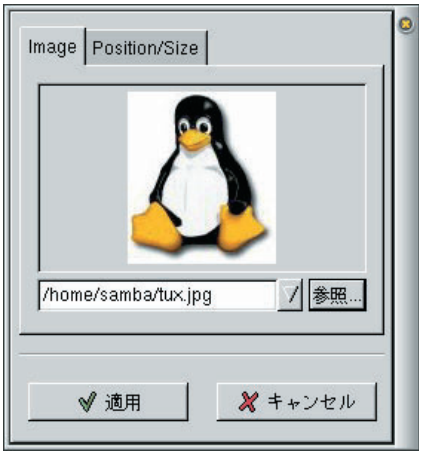
●矩形などの図形も配置可能

左のツールバー中央の3つのボタンで、3種類の図形（矩形／直線／楕円）をラベル上に配置できる（**画面3**）。配置する際に、クリックではなくドラッグを利用すると、最初から好みのサイ

画面1　起動時するとスプラッシュ画面とともに小型のウィンドウが開く。

画面2　複数行の文字列や日本語も問題なく入力・編集できる。

画面3　文字列のほか、矩形や楕円、直線などの図形も配置できる。



画面4　ラベル上に配置する画像はプレビュー画面で確認しよう。

ズで図形を配置可能だ。

一方、配置後にサイズを変更したり、線の太さや色、内部の色を変更したい場合は、図形上でのダブルクリックで開くダイアログを利用する。ラベル上で直接サイズを変更できないため操作が少し面倒だ。

なお、複数の図形を重ねてロゴなどを作成する場合、図形上で右クリックして、ポップアップメニューの[Bring to front]や[Send to back]を選択すれば、重なる順番を変更できる。

右クリックメニューの操作は、文字列や画像など、どのオブジェクトでも共通だ。たとえば、削除したいオブジェクトがあれば、その上で右クリックして[Delete]を選択すればいい。

●画像やバーコードを配置する

画像については、左のツールバーの下から2番目のボタンを押し、ラベル上でドラッグすることで画像の位置と大きさを指定する。

表示する画像ファイル名は、配置後にダイアログを開いて指定する。JPEG／PNG／TIFFなど数多くの画像形式に対応しており、選択した画像のプレビューも可能だ（**画面4**）。

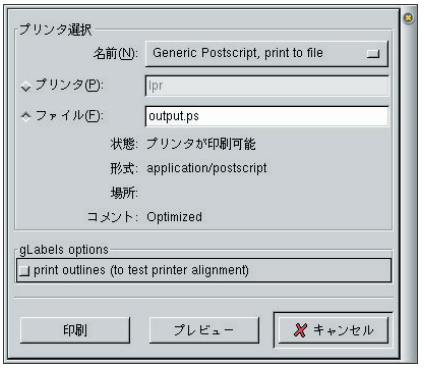
このほか、ラベル上にバーコードを配置することもできる。左のツールバー下端のボタンを押し、ラベル上でクリックすればいい。

バーコードの形式は、アメリカ郵便用の「PostNet」と、数字／記号／アルファベットを埋め込める「CODE39」の2種類に対応している。バーコードの内容などは、ダブルクリックで開くダイアログで設定する。

●プレビューしてから印刷

作成したラベルは、lpr経由でプリンタに印刷できるほか、PostScript形式やPDF形式でファイルに保存することもできる。なお、PDF形式でファイルに保存した場合には、日本語が正しく扱われないようだ。

ツールバーの[Print]ボタンを押すと印刷ダイアログが開くので、[名前]のリストボックスで印刷方法を選択する（**画面5**）。



画面5　lpr経由で印刷するほか、PSファイルなどへの保存も可能だ。

たとえば、lpr経由で印刷する場合は最初の「Generic Postscript」、PostScript形式でファイルに保存する場合は2番目の「Generic Postscript, print to file」を選択して、[印刷]ボタンを押せばいい。その前に[プレビュー]ボタンで印刷イメージを確認することもできる（**画面6**）。

なお、新たなシートの設定を追加して、マージン設定などがまだ調整できていない場合は、[print outlines]をチェックしておけば、ラベルが枠線付きで印刷される。

画面6　印刷イメージをプレビュー画面で確認することもできる。

大量のファイルの名前をまとめて変更する

Krename

バージョン：0.6　　ライセンス：GPL

http://freshmeat.net/projects/krename/

Krenameは、複数のファイルの名前をウィザード形式で変換するソフトだ。元のファイルはそのまま残して、変換後の名前でファイルを別ディレクトリにコピー／移動することもできる。また、ファイル名に連続する番号を付けたり、タイムスタンプや属性情報を変更するなど、さまざまな機能が用意されている。変換結果を前もって確認することも可能だ。動作にはKDE／Qtが必要となる。

●ビルドから起動まで

Krenameは、tarボールとRPMバイナリパッケージが配布されている。バイナリパッケージはRed Hat 7用なので、他のディストリビューションではtarボールからビルドしたほうがいい。「./configure」→「make」でビルドし、「su」でrootになってから「make install」としてインストールする一般的な手順だ。

ktermなどのコマンドラインで「krename&」とするか、KDEメニューの[ユーティリティ]－[Krename]を選択すると、開発版についての警告が表示されてからウィンドウが開く（**画面1**）。

●ファイルを登録する

Krenameは、ウィザード形式を採用しているので、それぞれの画面で必要な処理を行い、[Next>]ボタンで次の画面に切り替えていく。もちろん、[<Back]ボタンでいつでも前の画面に戻ることができる。

最初に行うのは、変換したいファイルの登録だ。右上の[Add Files]ボタンを押してオープンダイアログを開き、対象となるファイルを選択する。Ctrl－クリックによる複数選択や、Shift－クリックによる範囲選択にも対応している（**画面2**）。

ウィンドウ内には登録したファイルの名前が一覧表示され、右上の3つのボタンを使えば、さらにファイルを追加登録したり、指定したファイルをリストから取り除ける。

●ファイルの処理を選択する

[Next>]ボタンを押すと、次の画面に切り替わる。ここでは、選択したファイルの処理に関するオプションを以下の中から選択する（**画面3**）。

まず、元のファイルはそのまま残し、指定した名前で別のディレクトリにコピーする場合は、[Copy files to destination directory]をチェックする。また、ファイルの名前を変更した状態で別のディレクトリに移動するには、[Move files to destination directory]をチェックする。

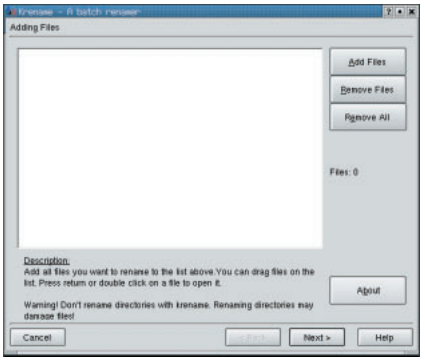
これら2つの選択肢をチェックした場合は、コピー／移動先のディレクトリを[Destination Directory]フィールドに指定する必要がある。なお、異なるパーティションへのファイルの移動はできないので注意されたい。

これに対し、既存のファイルの名前を直接変更したいなら、[Rename input files]をチェックする。もちろん、この場合はディレクトリを指定する必要はない。

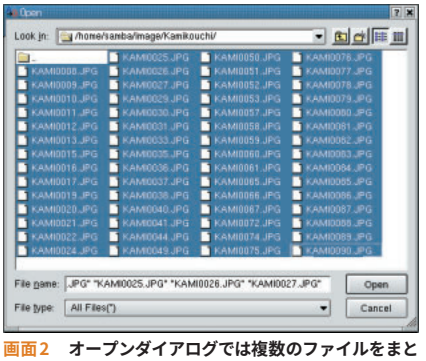
●ファイルの日時や属性を設定

続く2つの画面では、ファイルの作成（変更）日時と属性／所有者情報をそれぞれ変更できる。これらを変更する必要がなければ、そのまま[Next>]ボタンを押して次に進もう。

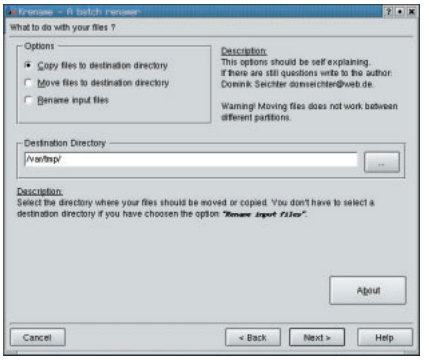
もし、複数のファイルの作成日時を統一したいなら、[Change date & time]にチェックを付け、その下のカレンダーなどを利用して日付や時間を指



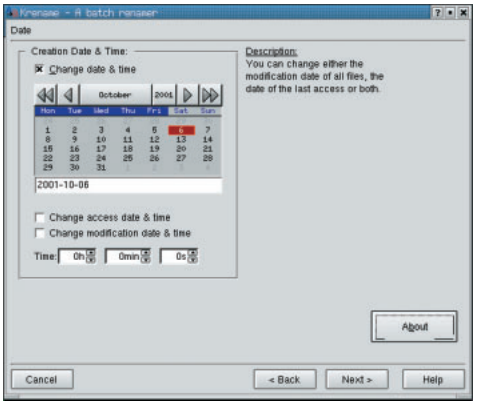
画面1　ウィザード形式で複数ファイル名の変換を効率よく行える。



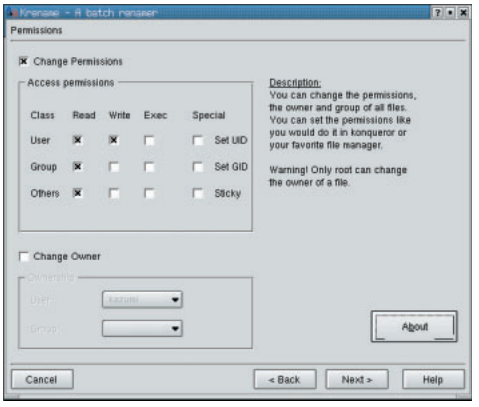
画面2　オープンダイアログでは複数のファイルをまとめて選択可能だ。



画面3　コピー／移動／名前変更の3つの選択肢からオプションを選択。



画面4　複数のファイルの作成日時などを統一することも可能だ。



画面5　ファイルの属性や所有者情報を変更する画面も用意されている。

定する（**画面4**）。ファイルの作成日時（time）だけでなく、アクセス時間（atime）やステータス変更時間（ctime）もあわせて変更可能だ。

その次の画面では、ファイルの属性（パーミッション）と所有者（オーナー）情報を変更できる（**画面5**）。ただし、所有者情報を変更できるのは、rootになった状態でKrenameを実行している場合に限られる。

●ファイルの名前を変更する

最後の画面では、変換後のファイル名を指定する（**画面6**）。大文字のファイル名を小文字に変換したり、連続する番号を付けることも可能だ。

ファイル名を指定する[Filename]には、特殊な意味を持つ記号（トークン）を指定できる。具体的には、

$：元のファイル名（拡張子を除く）そのものに置換
%：元のファイル名（拡張子を除く）を小文字に変換した文字列に置換
#：連続する番号に置換

の3つだ。これ以外の文字は、そのままファイル名として使われる。

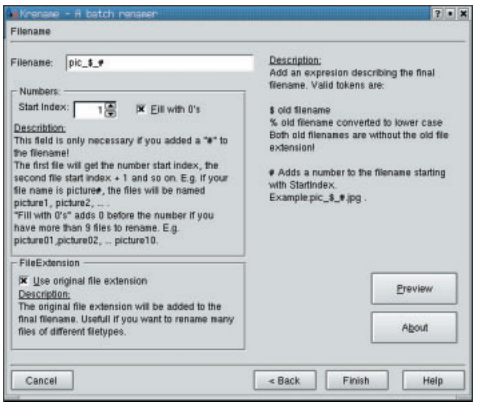
たとえば、元のファイル名が「foo」「bar」であるとする。[Filename]に「hoge_$」と指定すると、変換後のファイル名はそれぞれ「hoge_foo」「hoge_bar」になる。

ファイルに連番を付ける場合は、[Filename]の設定に「#」を含め、連番を開始する値を[Start Index]で指定する（初期値は「0」）。たとえば、[Filename]に「pic#」と指定すると、元のファイル名に関係なく、変換後のファイル名は「pic1」「pic2」になる。このとき、[Start Index]の右にある[Fill with 0's]オプションをチェックしておくと、ファイルの個数が10～99個の場合は「pic01」、100～999個の場合は「pic001」など、自動的に桁数を揃える処理も行われる。

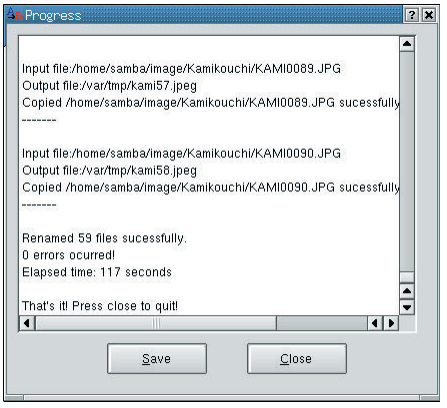
なお、初期設定では元のファイル名の拡張子がそのまま使われる。拡張子を変更したい場合は、左下の[Use original file extension]のチェックを外し、拡張子を含めたファイル名を[Filename]に設定すればいい。

ファイル名を設定したら、右側の[Preview]ボタンを押そう。実際の処理を行うことなく、変換結果をプレビューできる。複雑な設定を行った場合は、思い通りの結果になっているか慎重に確認したい。

変換結果に確信が持てたら、画面下の[Finish]ボタンを押す。これで、ファイル名の変換などの処理が実際に行われ、処理した結果が別ウィンドウに表示される（**画面7**）。



画面6　ファイル名の設定では3種類のトークン（$、%、#）を活用しよう。



画面7　変換処理の結果は別ウィンドウにまとめて表示される。

RPMパッケージを安全な環境で作成、テストする

RUST

バージョン：0.1　　ライセンス：GPL

http://www.rusthq.com/

RUSTは、グラフィカルな環境で直感的にRPMパッケージを作成するソフトだ。インストールするファイルをドラッグ＆ドロップで指定するだけの簡単操作だ。実際の処理は、コンソール系ツールcRUSTが行う。上級者であれば、cRUSTを直接実行して、既存のシステムファイルに影響しない安全な砂箱（sandbox）環境で、tarボールからRPMパッケージを作成できる。実行にはGNOME／GTK+が必要だ。

●ビルドとインストール

RUSTは、tarボールとRPMバイナリパッケージの両方が配布されている。このバージョンでは、ディレクトリ名に関する不具合があるため、tarボールを利用したほうがいい。

tarボールを展開したら、src/definitions.hをエディタで編集し、36行目に含まれる「.trash」のドット（.）を削除して「trash」に修正する。

あとは、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

●基本的な使い方

ktermなどのコマンドラインで「rust&」とするか、GNOMEメニューの[プログラム]－[開発ツール]－[Rust]を選択すると、左右にパネルを持つウィンドウが開く（**画面1**）。左パネルには実際のシステムのファイルリスト、右パネルには新ルート（new root）のファイルリストが表示される。

ファイルリストの操作方法は共通で、サブディレクトリをダブルクリックするか、ツールバーの[Up]ボタンでカレントディレクトリを移動できる。ファイルの選択はクリックにのみ対応しており、複数選択はできない。

●GUIによるパッケージの作成

（1）新ルートを設定

RPMパッケージ作成の際、仮想的なルートディレクトリ（/）として扱われる「新ルート」を、[New Root]－[Select]で指定する（**画面2**）。

初期値は/tmp/rustだが、ホームディレクトリ以下のサブディレクトリを指定してもいい。ホームディレクトリの指定に「~」は使えないので、フルパスで記述すること。

続いて、[New Root]－[Initialize]を選択して初期化を行う。しばらく待っていると、「/usr」など基本的な構成のディレクトリツリーが新ルート以下に作成され、右側のファイルリストに表示されるはずだ。

（2）パッケージの詳細を設定

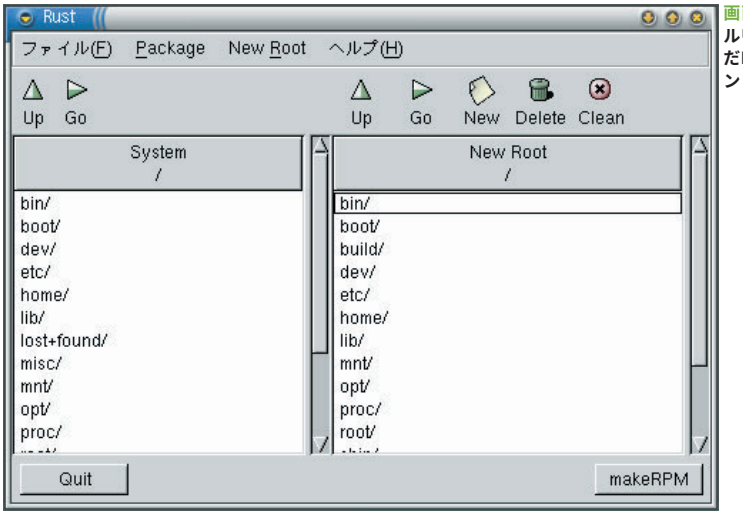
[Package]－[プロパティ]を選択すると、RPMパッケージの作成に必要な各種の情報をまとめて入力するダイアログが表示される（**画面3**）。

作成したパッケージを個人的に使うだけなら、パッケージ名とバージョン、リリース番号だけ設定すればいい。パッケージを他人に配布する場合は、簡単な説明や詳細な説明、分類、ライセンスといった項目も正確に記述する必要がある。

（3）ファイルを新ルート以下にコピー

RPMパッケージに含めたいファイルを左側のシステムパネルから選択し、ドラッグ＆ドロップを使って右側の新ルートパネルにコピーする。

コピーするファイルやコピー先ディレクトリを間違えた場合は、そのファイルを選択して[Delete]ボタンを押せば、新ルートパネルのファイルリストから削除される。



画面1　2つのファイルリストが左右に並んだRUSTのメインウィンドウ。

画面2　最初に、仮想的なルートディレクトリとなる「新ルート」を設定。

(4) RPMパッケージの作成

ウィンドウ右下の[makeRPM]ボタンを押し、メッセージボックスの[OK]ボタンを押すと、RPMパッケージの作成が始まる。もう一度メッセージボックスが開けば処理は完了だ（**画面4**）。

なお、RUSTにより作成されたRPMバイナリパッケージは、ホームディレクトリ以下のrust/RPMS/i386ディレクトリに格納される。これを通常の手順でインストールすればいい。

●tarボールからパッケージを作成

tarボールのみ配布されているソフトのRPMパッケージを作成するには、RUSTだけでなく、コンソール系ツールのcRUSTをユーザーが直接実行する必要がある。ktermなどの端末ソフトを起動しておこう。

なお、以下の説明に登場する「/tmp/rust」は新ルートの初期値なので、別のディレクトリを指定している場合は適宜読み替えて欲しい。

(1) 新ルートの初期化とコピー

ktermなどのコマンドラインで、

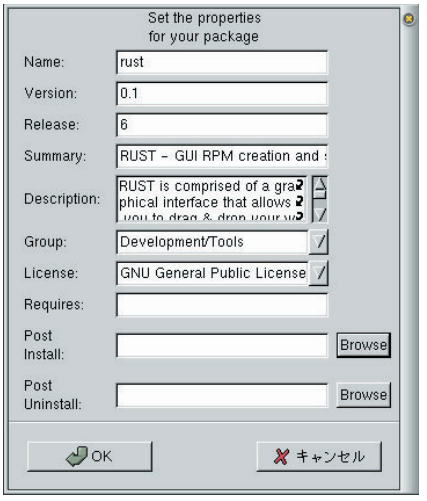
```
$ crust --init --copy /tmp/rust
```

とする。初期化に加え、システムファイルのコピーとMD5チェックサムの計算を行うため、処理にはしばらく時間がかかる。また、新ルートを含むパーティションには、ある程度の空き容量が必要だ。

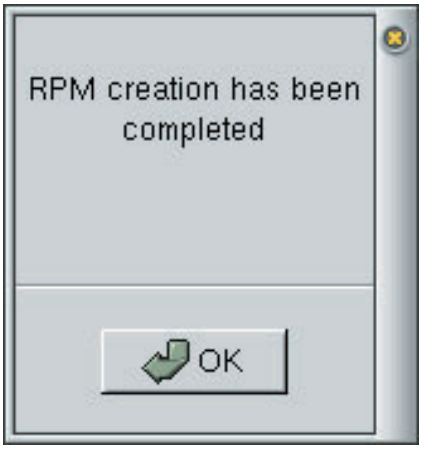
(2) tarボールのビルド

インストールしたいソフトのtarボールを、/tmp/rust/buildディレクトリに展開し、「./configure」→「make」といった手順でビルドする。

RPMパッケージ作成時に、更新／追加されたファイルを検出する際、buildディレクトリ以下のファイルだけは対象外になる。そのため、tarボールの展開やビルドはこのディレクトリ以下で行う必要がある。



画面3　RPMパッケージの作成に必要な情報をこのダイアログに記述する。



画面4　このメッセージが表示されればRPMパッケージの作成は完了だ。

(3) 砂箱環境でインストール

あらかじめ「su」としてrootになった状態で、

```
# crust --chroot /tmp/rust
```

とすると、プロンプトが「bin/sh-2.04#」などに変化し、以後は/tmp/rustがルートディレクトリ（/）として扱われるようになる。この状態をRUSTでは「砂箱」（sandbox）と呼んでいる。

あとは、buildディレクトリ以下のtarボール展開先に移動し、「make install」などとしてインストールすればいい。砂箱の中では、RPMパッケージからファイルがインストールされるのは/tmp/rust以下のディレクトリなので、既存ファイルの上書きや削除の心配をせずに、RPMパッケージのテストが行える。

「exit」として砂箱状態から脱出したら、念のために、

```
# chown -R ユーザー名 /tmp/rust
```

として、/tmp/rust以下の全ファイルの所有者をあなたに変更する。もちろん、「ユーザー名」の位置には、あなたのユーザー名を記述すること。

もう一度「exit」とすると、crustの砂箱環境を抜け、元の一般ユーザーの状態に戻る。

(4) RPMパッケージの作成

RUSTを起動して、RPMパッケージの作成を行う。まずは、[New Root]－[Select]で新ルート（/tmp/rust）を選択する。また、MD5チェックサムによるチェックを有効にするため、[New Root]－[Use MD5 sum]を選択しよう。

あとは、[Package]－[プロパティ]を選択して、パッケージ名やバージョン、リリース番号などを設定し、左下の[makeRPM]ボタンを押せばいい。tarボールからのインストールによって更新・追加されたファイルが検出され、それらのファイルで構成されたRPMバイナリパッケージが作成される。

マルチプレイヤーの戦略級シミュレーションゲーム

TEG

バージョン：0.8.0　　ライセンス：GPL

http://teg.sourceforge.net/

TEG（Tenes Empanadas Graciela）は、ターン制のマルチプレイヤー戦略級シミュレーションゲームだ。2～6人のプレイヤー（ロボット含む）が、世界を舞台に覇権を競う。サーバとクライアント、ロボットの各ソフトが含まれているので、人間相手のネットワーク対戦はもちろん、ローカルマシンでソロプレイすることも可能だ。汎用ゲームサーバGGZ対応版も用意されている。動作にはGNOME／GTK+が必要だ。

●ビルドとインストール

TEGは、tarボールとRPMバイナリパッケージが配布されている。バイナリパッケージはRed Hat 6用と7用が用意されているので、ディストリビューションに合わせて選択しよう。

tarボールの場合は、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

サーバソフトのtegserver、クライアントソフトのtegclient、コンピュータプレイヤー（ロボット）のtegrobotがインストールされる。

●クライアントとサーバを起動

ktermなどのコマンドラインで「tegclient&」とするか、GNOMEメニューの[プログラム]－[ゲーム]－[T.E.G. client]を選択すると、美しいマップを持つクライアントソフトが起動して、サーバ接続用のダイアログが自動的に開く（**画面1**）。

ローカルマシンでソロプレイする場合、オプション設定は初期値のまま、[Start server locally]にチェックを付けて[OK]ボタンを押そう。自動的にxtermのウィンドウが開いて、TEGサーバが起動する（**画面2**）。

一方、ネットワークで接続された他のマシンでTEGサーバを起動済みなら、[Server name]にそのホスト名を指定すればいい。

サーバとの接続に成功すると、色の選択ダイアログが開く。ここで選択した色が、プレイ中のあなたの部隊の色になる。なお、選択した色がすでに使われている場合は、サーバが自動的に別の色に変更する。

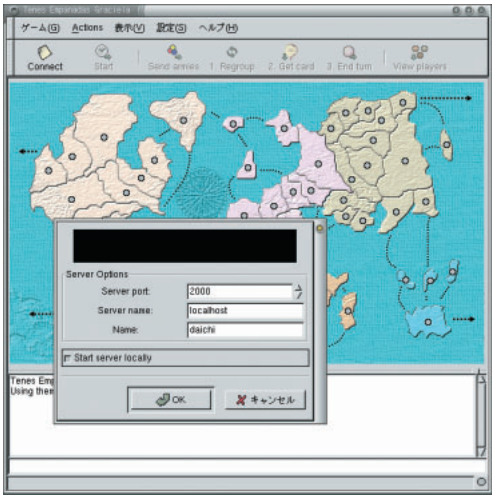
●ロボットの起動とプレイ開始

クライアントウィンドウの下部エリアは、単にゲーム中のステータス表示に使われるだけでなく、プレイヤー間のチャットにも利用できる。発言の先頭には、先ほど選択した色と名前が表示される。

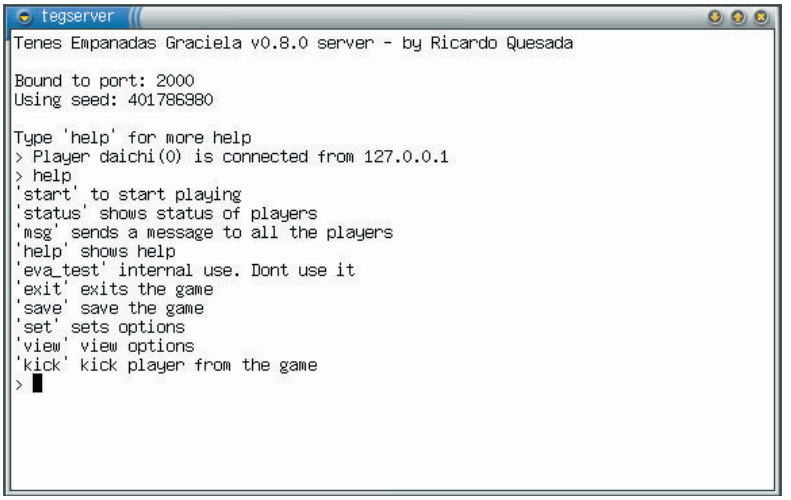
TEGのプレイには、少なくとも2名（最大6人）のプレイヤーが必要だ。現在サーバに接続しているプレイヤーは、ツールバーの[View players]ボタンで確認できる（**画面3**）。

プレイヤーの数が足りない場合は、[ゲーム]－[Launch robot]を選択してロボットを起動しよう。TEGのロボットは、チャットで発言するなどなかなか活動的だ。複数のロボットを起動して対戦することも可能だ。

プレイヤーが集まったら、ツールバーの[Start]ボタンを押し、ゲームの種類（勝利条件）を選択する。ひとつは、世界全域の征服で勝利となる「世界征服」、もうひとつは、プレイヤーごとに指定された条件を満たせば勝利となる



画面1　TEGのクライアントを起動したら、まずはサーバに接続する。



画面2　xtermに表示されたサーバ。コマンド入力による制御も可能だ。

「秘密任務」だ。

●部隊の初期配置を行う

TEGはターン制なので、各プレイヤーが順番に部隊の配置や移動（戦闘）などの操作を行う。最初の2ターンは、部隊の初期配置に使われる。

マップ内の各国の円の色がその国を支配するプレイヤー、円内の数字が部隊数を示している。開始直後は、いずれかのプレイヤーの軍隊が1部隊ずつランダムに配置されている。

最初のターンでは、5部隊を自分の国に追加配置する。自分の色の円をクリックして部隊数を増やそう（上限はない）。確定前なら右クリックで減らすことも可能だ。ツールバーの[Send armies]ボタンを押すと、配置を確定してサーバに送信する。

次のターンでは、他のプレイヤーの部隊の配置状況を検討しつつ、さらに3部隊を追加配置する。[Send armies]ボタンを押すと、いよいよ本格的なプレイの開始だ。

●敵国と戦闘を行う

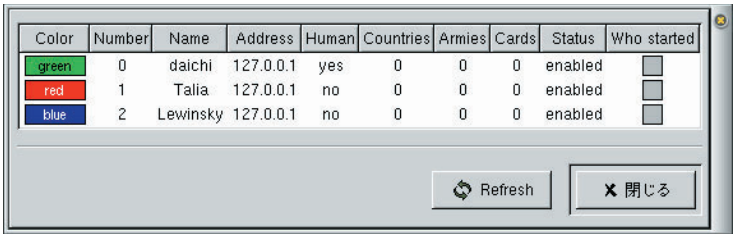
戦闘は、攻撃側の国→防御側の国という順番で、2つの国の円をクリックすることで行う。もちろん、攻撃側は自分の国、防御側はそれ以外の国でなければならない。また、両国は国境を接しているか、点線で結ばれている必要がある。

さらに、攻撃側の国には最低でも2部隊が駐留していなくてはならない。1部隊は自国に残るため、攻撃に参加する部隊の数は、（部隊数－1）個となるからだ。一方、防御側では全部隊が防御に参加する。

判定はサイコロで行われる（**画面4**）。振られるサイコロの数は、攻撃／防御に参加する部隊数と同じだ（3部隊以上の場合は3個）。それぞれのサイコロの値が大きい順に並べられ、ひとつずつ比較される。

もし、攻撃側の値のほうが大きければ、防御側の部隊が1つ消滅し、等しいか小さければ攻撃側の部隊が1つ消滅する。もし、防御側の部隊が全滅した場合は、攻撃側がその国を支配し、最大3部隊を移動できる。ターン内の戦闘数は制限されていないので、大量の部隊を配置していれば繰り返し戦闘が可能だ。

画面3　サーバに接続中のプレイヤーを確認。プレイには2名以上必要だ。



●戦闘後の行動とターンの終了

戦闘をすべて終えたら、そのターン内で行える行動は、部隊の配置替えと国カードの取得しかない。

配置替えを行う場合は、ツールバーの[Regroup]ボタンを押し、移動元の国→移動先の国という順番で、2つの国の円をクリックする。戦闘の場合とは違って、どちらも自分の国でなければならない。

一方、国カードの取得は[Get card]ボタンで行う（**画面5**）。自分の国のカードを引いた場合は、自動的に2部隊が追加される。カードの絵柄は4種類あり、同種・異種のカードを揃えることで多数の部隊と交換可能だ（詳細はマニュアルを参照）。

最後に[End turn]ボタンを押すと自分のターンが終了し、他のプレイヤーのターンに移る。

●これ以後のターンでは

すべてのプレイヤーが行動を終えると、次のターンが開始され、新たな部隊の配置に移る。以後の行動は前のターンとまったく同じだ。

基本的には、各ターンのはじめにプレイヤーが所持する全部隊の半数の部隊が新たに追加される。さらに、色分けされた各大陸のすべての国を支配すると、ボーナスとして数個の部隊が追加される。

このため、「ひとつの大陸内のすべての国の征服を目指して戦闘し、それが成功してから別の大陸へ侵攻する」という戦略が有効となる。

画面4　戦闘結果はサイコロを使って判定される。大戦力を集中せよ。

画面5　引いたカードの国を支配していれば即座に部隊が追加配置される。

ギャラクシアン風のシューティングゲーム

Geki5

バージョン：0.3.0　　ライセンス：GPL

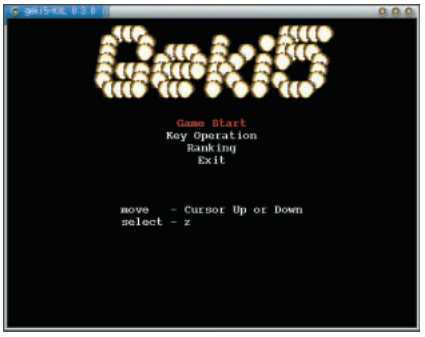
http://www2.mwnet.or.jp/~fc3srx7/

Geki5は、ギャラクシアンに似た画面固定タイプの国産シューティングゲームだ。以前紹介した「Geki2」や「Geki3」と同じシリーズで、ボスキャラは登場しないものの、アイテムによる武器のパワーアップが可能だ。現時点では、全部で8つのステージが用意されている。実行には同じ作者のKXLライブラリ（1.1.1以降）が必要だ。

●ビルドとインストール

KXLライブラリは、tarボールとRPMのバイナリ／ソースパッケージが配布されている。tarボールを利用する場合は、「./configure」→「make」でビルドし、「su」でrootになってから「make install」でインストールする一般的な手順だ。

Geki5のほうは、tarボールのみ配布されている。インストール手順はKXLとまったく同じだ。説明は省略するが、configureスクリプトの実行により生成されるSPECファイルを利用して、RPMパッケージを作成することもできる。



画面1　Gekiシリーズおなじみのタイトル画面。zキーでプレイ開始だ。

●基本的な遊び方

ktermなどのコマンドラインで「geki5&」とすると、ウィンドウが開いてタイトル画面が表示される（**画面1**）。zキーを押すとプレイ開始だ。

画面下に位置する自機を←／→キーで左右に移動させつつ、zキーで弾を発射して、画面上部の敵を倒そう。zキーを何度も押せば、ある程度の連射が可能だ。

しばらく画面上部を左右に移動しているだけの敵は、時間の経過とともに赤い弾を発射しながら降下してくる。敵の弾や敵自体に触れないように攻撃しよう（**画面2**）。

ステージ内のすべての敵を倒せばクリアで、次のステージに移る。一方、敵の弾や敵自体に触れると自機が爆発してライフが減り、ライフが0になるとゲームオーバーだ。

●アイテムでパワーアップ

先のステージでは、何回も攻撃を加えないと倒せないような「硬い」敵が登場する（**画面3**）。こちらもアイテムでパワーアップしよう。

このゲームでは、敵を倒すと水薬の形をしたアイテムが落ちてくる。落下スピードはけっこう速いので、敵の弾と間違えないように回収しよう。自機がアイテムに触れると、画面左上のパワー値が増加する。

パワー値が50増えるたびに、弾を連射できる限界が増えていく（初期値は3）。パワー値は自機が破壊されてもリセットされないので、先のステージに備えて、できるだけアイテムを回収するようにしよう。

画面2　画面上部から迫りくる敵や敵の弾をかいくぐって攻撃せよ。

画面3　ステージが進むにつれ、一撃では破壊できない敵も登場する。

拡大／縮小や形式変換に優れた画像ビューア

Image Viewer

バージョン：0.1.6　　ライセンス：GPL

http://wolfpack.twu.net/utilities.html

Image Viewerは、イメージライブラリのImlib（またはImlib2）を利用した画像ビューアだ。さまざまな画像形式に対応しており、表示した画像を別形式で保存することもできる。また、マウスのドラッグやツールバーのボタンなど、さまざまな方法で表示画像を拡大・縮小できるのが特徴だ。このほか、画面キャプチャや印刷、ルートウィンドウ（壁紙）への描画機能なども備えている。実行にはGTK+が必要だ。

●ビルドとインストール

Image Viewer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ので、tarボール展開先で「cd iv」→「make all」としてビルドし、「su」でrootになってから「make install」でインストールする。

なお、標準設定ではImlibが使われるため、Imlib2を利用したバージョンを作成する場合は、専用のmakeファイルを利用する必要がある。具体的には、「make -f Makefile.Imlib2 all」とビルド時に指定すればいい。

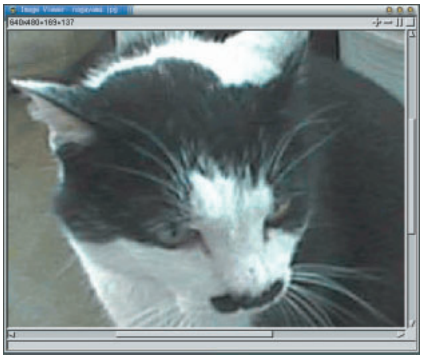
●ドラッグやボタンで拡大／縮小

ktermなどのコマンドラインで「iv&」として起動すると、Image Viewerのウィンドウが開いてタイトル画像が表示される（**画面1**）。

操作は右クリックメニューかショートカットキーで行う。たとえば、画像を表示するには、右クリックメニューの[Open]を選択して（あるいはCtrl－Oキーを押して）ダイアログを開き、目的のファイルを選択すればいい。JPEG／PNG／TIFF／BMPなど、ほとんどの画像形式に対応している。

表示中の画像は、左ボタンのドラッグで上下左右にスクロールし、中ボタンのドラッグで拡大縮小する（**画面2**）。また、ツールバーには、拡大／縮小を行う[＋]／[－]ボタンのほか、元の縮尺に戻す[‖]ボタンや、現在のウィンドウサイズにフィットさせる[□]ボタンも用意されている。

画面1　Image Viewerの起動時にはタイトル画像が表示される。



画面2　表示した画像は、マウスのドラッグでスクロールや拡大が可能だ。

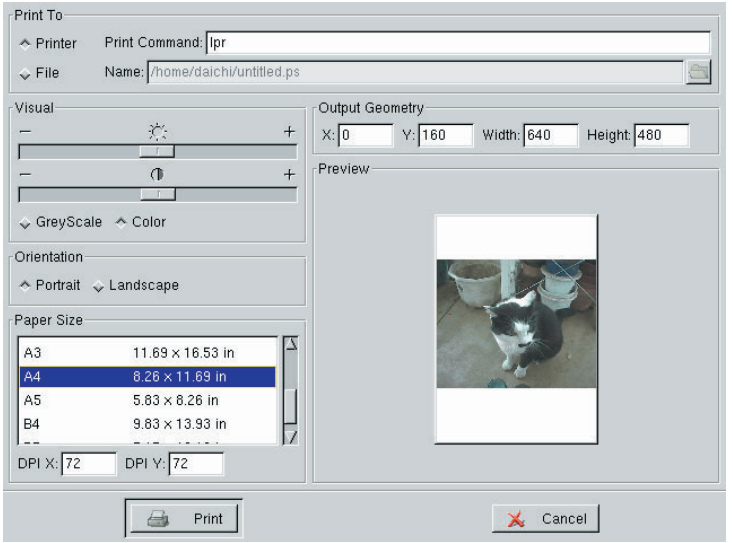
●画像の保存や印刷を行う

表示中の画像を別形式で保存するには、右クリックメニューの[Save As]（またはCtrl－Sキー）でダイアログを開き、[Type]のリストから画像形式（計35種類）を選択する。

また、[Print]（Ctrl－Pキー）で開くダイアログでは、表示中の画像をlpr経由で印刷できる。用紙サイズの選択やコントラストの調整、印刷位置の指定などが可能だ（**画面3**）。

このほか、右クリックメニューの[Grab Window]（またはCtrl－Gキー）では、画面上のウィンドウや任意の矩形領域を画像としてキャプチャーできる。左ボタンはウィンドウの選択、中ボタンは矩形領域の指定だ。

画面3　コントラスト調整や印刷位置の指定が可能な印刷ダイアログ。



再生中のサウンドをさまざまに視覚化する

eXtace

バージョン：1.6.1　　ライセンス：GPL

http://extace.sourceforge.net/
http://www.fftw.org/（FFTW）

eXtaceは、再生中のサウンドを高速フーリエ変換（FFT）し、得られたスペクトル情報をリアルタイムに視覚化するソフトだ。3Dサーフェスやスパイク、オシロスコープ、グラフィックイコライザ、スペクトログラムなど豊富な表示モードを持つ。再生中もボタン操作で表示方法を自由に切り替え可能だ。動作にはサウンドドライバ（esdまたはALSA）と、FFTライブラリのFFTWが必要となる。

●ビルドとインストール

FFTWはtarボールとRPMソースパッケージが配布されているので、ディストリビューションに合わせて選択しよう。tarボールからのビルドとインストールはconfigureスクリプトを利用する一般的な手順だ。

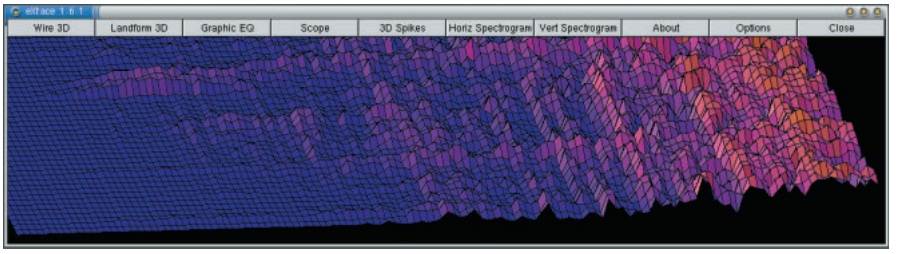
RPMソースパッケージを利用する場合は、まず「rpm --rebuild fftw-2.1.3-1.src.rpm」としてリビルドする。続いて、作成されたバイナリパッケージを、「rpm -Uvh /usr/src/redhat/RPMS/i386/fftw-＊2.1.3-1.i386.rpm」として、develパッケージとともにインストールすればいい。

eXtaceのほうも、tarボールとRPMパッケージが配布されている。ただし、RPMパッケージはMandrake 8用なので、国産ディストリビューションではtarボールを利用したほうがいい。ビルドとインストールはconfigureスクリプトを利用する一般的な手順だ。

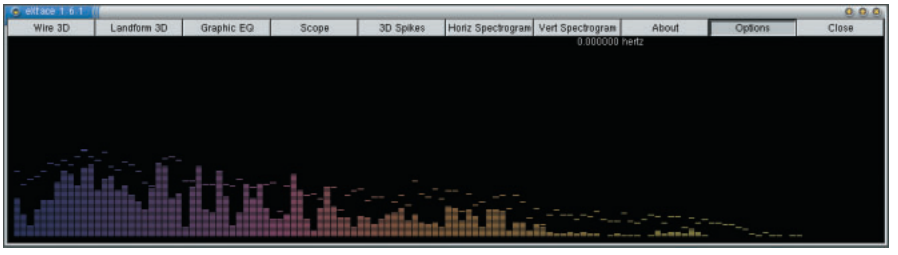
●ボタンひとつで表示を切り替え

ktermなどのコマンドラインで「extace&」とすると、上部にボタンが並んだウィンドウが開く。その後、esdやALSAを利用してサウンドを再生すると、FFTにより得られたスペクトル情報がリアルタイムに3Dサーフェスとして表示される（**画面1**）。

スペクトル情報の表示方法は、ウィンドウ上部のボタンで変更できる。左端から、3Dワイヤーフレーム、3Dサーフェス、グラフィックイコライザ（**画面2**）、3Dスパイク、オシロスコープ、水平・垂直スペクトログラムの7種類で、サウンド再生中に切り替えることも可能だ。3D表示に関しては、傾きや更新速度を小型のDirectionウィンドウで変更できる。



画面1　再生されるサウンドのスペクトル情報を3Dサーフェス表示する。

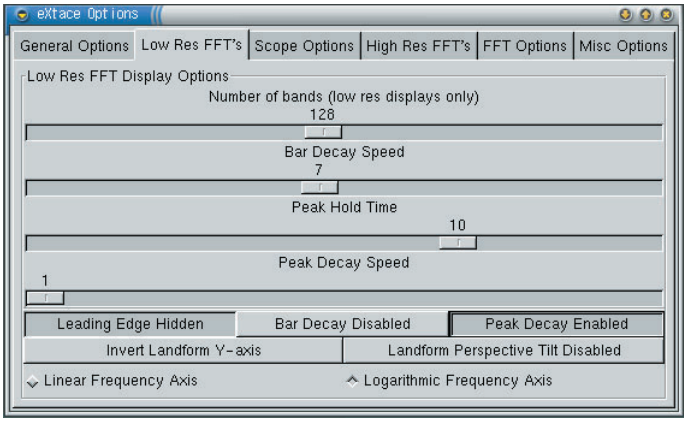


画面2　2Dのイコライザ表示。ピーク値にディケイをかけると格好いい。

右から2番目の[Option]ボタンで開く設定ダイアログには、さまざまな設定項目がジャンル別に分類されている。たとえば、[Low Res FFT's]ページでは、グラフィックイコライザのバンド数やピーク値のディケイ速度などをカスタマイズできる（**画面3**）。

なお、tarボール展開先のextaceディレクトリに、サイン波発生ツール（sine）が用意されている。ALSAでは「./sine > /dev/dsp」、esdでは「./sine | esdcat」とすればいい。

画面3　イコライザのバンド数やディケイ速度はこのページで設定する。



対話的な操作で正規表現の文字列を生成する

^txt2regex$

バージョン：0.6　　ライセンス：GPL

http://txt2regex.sourceforge.net/

^txt2regex$は、正規表現の文字列を対話的な操作で生成するシェルスクリプトだ。正規表現は、egrepやPerl、vi、Postgresなど数多くのツールで検索や置換に使われるが、初心者にとってはさまざまな記号（メタ文字）の使い方を習得するのが難しい。このソフトを使えば、一連の質問に答えていくだけで、各種ツール用の正規表現が自動的に生成される。実行にはbashのバージョン2が必要だ。

●スクリプトの修正とインストール

^txt2regex$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。シェルスクリプトなので、ビルドの必要はない。

オリジナルのスクリプトでは、履歴の区切りにグラフ文字（0xa4）が使われているため、日本語端末では履歴部分が文字化けしてしまう。これを解決するには、tarボールの展開先で、「perl -pi -e 's/\xa4/:+:/g' txt2regex.sh」として、区切りを別の文字列「:+:」に変更する必要がある。

また、筆者が作成した日本語カタログ（ja.po）が本誌付録CD-ROMに収録されている。tarボール展開先のpoディレクトリにコピーしておこう。

以上の処理を完了したら、「su」としてrootになった状態で「make install」でインストールする。

●対話的な操作で正規表現を生成

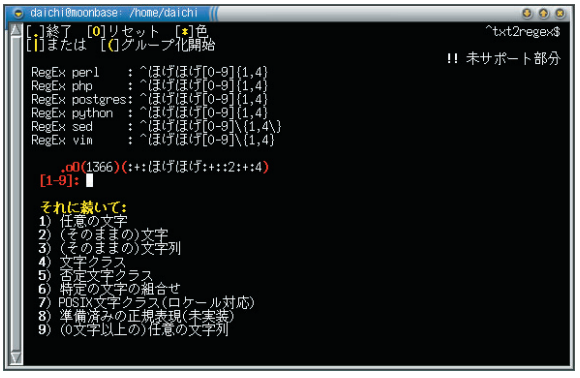
ktermなどのコマンドラインで「txt2regex」として起動すると、カラフルな画面が表示される。端末の背景が白に近い場合は、「txt2regex --whitebg」とするといい。

画面中央には、正規表現を生成するための質問が黄色で表示される。マッチを始める位置や文字の内容、繰り返す回数などだ。下に並んだ選択肢から当てはまるものを選んで番号を入力すればいい（**画面1**）。

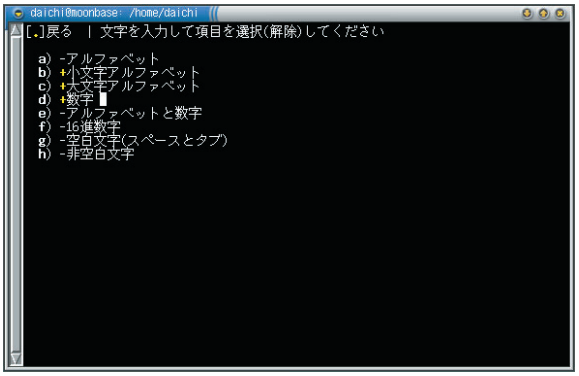
選択肢の中には、文字や文字列をキー入力したり、POSIX文字クラスのように別の画面で項目を選択させるものもある（**画面2**）。別画面の項目は複数選択でき、「.」を入力すると元の画面に戻る。

このようにして質問に答えていくと、画面上部の「RegEx プログラム名:」のうしろに正規表現の文字列が表示される（未サポート部分は「!!」となる）。プログラムとしては、PerlやPostgres、vimなどが選択されているが、「/」で画面を切り替えると、egrepやEmacs、JavaScriptなど、計23個のプログラムから選択できる。

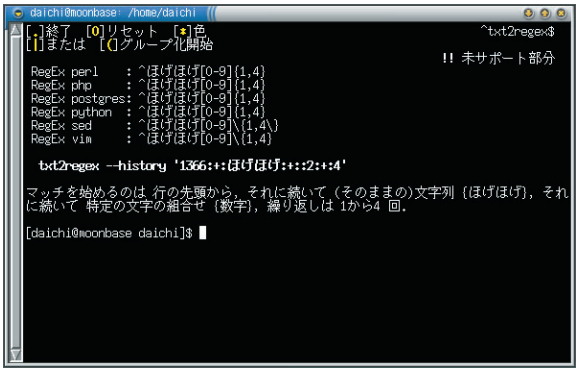
求める正規表現が得られたら、「.」を入力すると、それぞれのプログラムで利用できる正規表現の文字列と、その意味を言葉で説明した文章が表示される（**画面3**）。



画面1　画面に表示される質問に答えていくだけで正規表現が生成される。



画面2　POSIX文字クラスの選択は、別の画面で行われる（複数選択可）。



画面3　各プログラム用の正規表現文字列とともに文章による説明も表示。

隠喩としてのコンピュータ

これが俺達の世界、隠しきれない世界

文：豊福 剛
Text : Tsuyoshi Toyofuku
illustration : hmm

まるで映画のような、と誰もが形容した。そして、これは映画ではない、現実なのだ、と我に返る。アメリカの航空機の運行がすべてストップした。株式市場は6日間もストップした。これは映画ではない、現実なのだ。これでもか、これでもかと反復されるWTCビル崩壊の映像。この現実をどのように把握すればいいのか。

この映画を信じろ、そして戦え

これは事故ではない、テロである。そして、これはテロではない、戦争である、と宣言される。その瞬間、この現実は再び映画に反転する。これはパールハーバーである。これは民主主義に対する挑戦である。これは善と悪の闘いである。いったい敵は誰なのか。イスラム原理主義、いや、そうではなく、イスラム過激派。黒幕はビンラディン。アフガニスタンに潜伏中。このようにして現実が映画に反転する。映画に証明は求められない。大統領が、CIAが、FBIが、事件の真相を把握している。この映画を信じろ、そして戦え。このようにして映画が進行する。

見ることは信じることである。4機の民間旅客機がハイジャックされ、そのうちの2機がWTCビルに激突し、1機がペンタゴンに激突し、1機がピッツバーグに墜落した。最後の1機は軍によって撃墜されたのではないか、という疑いもある。はたして、それがどのように墜落したのか、真相は明らかにされていない。あのような状況においては、撃墜されることもあるという判断は明らかになった。しかし、いまはそれどころではない。これは戦争なのだから。そして映画は、見る者の疑問などお構いなしに、どんどん先に進んでいく。

確実なのは、この4機の民間旅客機がハイジャックされたことである。誰が、どのようなやり方で、ハイジャックを行ったのか。19人の実行犯容疑者の名前は公表された。それぞれの旅客機の乗客名簿があって、そこから割り出したにしても、しかしこれらの容疑者が偽名を使っているのは当然だろう。なぜ、この19人が容疑者として断定できたのか、その根拠は明らかにされていない。

この19人は、FBIやCIAのブラックリストに載っていたのかもしれない。彼らの行動の一部は密かに監視されていたのかもしれない。ピッツバーグに墜落した旅客機では、犯人と乗客の一部が機内でもみあったらしいが、その中にはFBIやCIAの捜査官がいたのかもしれない。確実なこと

は何もわからない。しかし、容疑者の名前が公表されると、人々の不安や恐怖は幾分か軽減される。しかし、それは、ほんとうのところは、気休めにすぎないのだ。

こんな事態は想定してなかった

ハイジャックの犯行には、小型ナイフが使われたらしい。アメリカの国内線の場合、空港でのセキュリティ・チェックに不備があったことが指摘されている。しかし、今回のようなハイジャックに対しては、どれだけセキュリティ・チェックを強化したとしても、万全の対策にはならないだろう。なぜなら、ナイフすらなくても、格闘技に長けていればハイジャックできてしまうだろうから。

旅客機の操縦は、離陸と着陸は難しいとしても、それ以外の飛行そのものの操縦については、実はそれほど難しいものではないらしい、ということも、今回のテロが可能になった技術上の大きな要因である。容疑者の何人かが、あの有名なフライトシミュレータのゲームソフトで操縦の練習していたという目撃証言もあった。実際それが、どれほどの効果があるものなのかはわからないのだけれども、現代の旅客機がコンピュータによる操縦支援や自動操縦を不可欠なものとしている以上、今回のテロの技術的なハードルは、実はそれほど高くないというのは、事実なのかもしれない。

そうだとすると、今後も同様のテロが発生する可能性は否定できない。そして、ハイジャックされた旅客機のターゲットが、たとえば原子力発電所だったら、といった想像が、さらなる不安と恐怖をかきたてる。

2機の旅客機によるミサイル攻撃。WTCビルは、このミサイルの激突の衝撃と爆発によって、その建物の上層階部分が破壊されただけでなく、自らの重みに耐え切れなくなったかのように、建物全体が崩れ落ちていった。そしてビルの自壊による陥没は、さらにその周辺のビルの倒壊をも巻き込んでいった。そこまで完全に計算したうえでの激突だったのか、それはわからない。おそらくエレベータは停止し、非常階段には逃げ惑うあまりの多くの人々がいて、安全な場所に避難できるまでには、絶望的な時間を要したのだろう。いったん崩れ始めたビルは、あのように一瞬にして崩壊してしまうものなのか。超高層ビルのセキュリティは、あのようなテロに対しては、あまりにも無力であるだけでなく、それ自体が圧倒的な数の人命を一瞬にして奪

う恐るべき兵器に転化してしまうのだ。

このようなテロによる被害から身を守るためには、極論すれば、高層ビルには近づかない、そして飛行機も利用しない、という対応しかないのかもしれない。絶対安全な高層ビル、絶対安全な飛行機、そのようなものはありえないのだし、セキュリティは常に経済効率とのトレードオフにおいて成立していることを、改めて認識するしかないのだ。

テロ対策の代償

しかし、そもそも、テロを行う者がいなければテロは起りえないのだから、テロ問題の根本は、誰が、いかなる意図をもって、テロを行うのかにある。ただし、戦術の問題は無視できないのであり、このような戦術をとれば、たった数人の実行によって、これだけの大規模で圧倒的な暴力になりえることが実証されてしまったのは、テロリストにとっては、あまりに大きな成功であり、国家安全保障の立場にとっては、あまりに大きな失敗であるのだ。

今回のテロがイスラム過激派によるものだとしても、しかし、この戦術を行使しえる主体は、かならずしもイスラム過激派だけに限定されるものではないだろう。この戦術はひとり歩きしてしまうのだ。第二、第三のユナボマーが出てきて、このような戦術を行使する可能性は、誰も否定できないのだから。

いったいテロと戦争の違いはどこにあるのか。それは、敵が内部にいるか外部にいるかの違いであるように思う。そう考えると、これはテロではない、これは戦争であるという宣言は、敵を外部に設定すると同時に、内部における結束を命じる、政治的な、あまりに政治的な権力の発動であるのだ。

いかなる理由があれ、テロを許してはならない、断固たる対応をとらなければならない。それはあまりに正しい判断であり、それに異議を唱えることは難しい。しかし、だからこそ、このような有無をいわせぬ正しさに対しては、あえて疑いの目を向けなければならない。

敵は外部にいるだけでなく、内部にもいる以上、内部にいる敵を狩り出すために、あらゆる手段を駆使した監視のシステムが発動することになるだろう。このような監視のシステムがいったん発動してしまえば、それは必ずひとり歩きをしてしまう。テロリスト狩りのためのシステムは、誰がテロリストであるかを洗い出すために、ありとあらゆ

る人々のプライバシーを監視するシステムに容易に転化してしまうだろう。

「USA」のシュプレヒコール、「God bless America」の大合唱。テロに対する軍事的な報復、それは、やられたらやり返すという、あまりにわかりやすい心理だ。アメリカはテロに対する報復を戦争として展開しようとしている。しかし、戦争は国家対国家の戦争である以上、アフガニスタン、そしてイラクを、アメリカの敵であるとする戦争の枠組みを強引に形成しようとしている。そして、日本に対しても、湾岸戦争のとき以上の具体的な貢献が要請されている。

報復の連鎖はすでに始まっていた

すでにアメリカは、クリントン政権のときに、ケニアとタンザニアでのアメリカ大使館爆破テロに対する軍事的報復として、アフガニスタンとスーダンに対するミサイル攻撃を実行した。この攻撃に、はたして正当性があったのかどうかを問うには、もはや遅すぎる。この戦争が戦争だとすれば、それはすでに1998年に始まっていたのだ。すでに報復の連鎖は始まっていたのだ。しかし、アメリカのメディアでは、このミサイル攻撃のことは、けして問題提起されることがなく、ケニアとタンザニアでの大使館爆破テロは、あくまでビンラディン黒幕説を裏付ける状況証拠としてだけ報じられている。

アメリカの正義がダブル・スタンダードであることは、イスラエルのパレスチナに対する攻撃が、国連決議に違反したものであるにも関わらず、それを黙認するだけでなく、経済的・軍事的にもイスラエルを支援していることからも明らかである。パレスチナ問題と今回のテロが、どのような関係にあるのか、慎重に検証されているわけでもないのに、メディアでは反パレスチナ感情を煽るかのように、アメリカでのテロのニュースを喜び、パレードをするパレスチナの姿が報道された。しかし、その映像は1991年に撮影されたものではないかという指摘もある。

『リナックスの革命』の序文で、リーナスは「戦争もまちがいなく生き残りのために始まって、社会秩序の維持手段へと発達して、そして今やどうしようもなくエンターテイメントへの道を進んでいる」と書いているけれども、そのようなシニカルな認識は、WTCビルとともに木っ端微塵に吹き飛んでしまったことだろう。

Profile

とよふく つよし

1962年東京生まれ。メディアデザイン研究所技術顧問。訳書に『Javaプログラムクイックリファレンス』『Java分散コンピューティング』（オライリージャパン）『GIMPパーフェクトガイド』（エムディーエヌコーポレーション）などがある。

2つのグローバリゼーション

文：安田幸弘
Text : Yukihiro Yasuda

憂鬱な日

9月11日、ぼくは成田に向けて飛び立つ飛行機が離陸するまでの時間つぶしに、一緒に帰国する仲間と一緒にウルグアイの古い港町、モンテビデオのマーケットを歩いていた。

マーケットの土産物屋のおやじが、「ほんの15分ぐらい前、ニューヨークでハイジャックされた飛行機がビルに突っ込んだらしいよ」なんて言っていたのだけれど、そのときは「ひどいねえ」なんて答えて、30秒で忘れてしまった。その「ひどさ」を知ったのは、ゆっくりモンテビデオ最後の昼食を楽しみ、あたりを歩きまわってホテルに帰ってきたときだった。

ロビーのテレビが報じるCNNのニュース画面には、飛行機が貿易センタービルに突入するあの場面、そして「アメリカのすべての空港は閉鎖された」というアナウンス。一行は絶句した。ぼくはアトランタ経由で、ほかの仲間はシカゴ経由のチケットを持っていたのだ。

結局、あちこち奔走して、ニューヨークに帰る予定のメンバーを除き、全員13日にヨーロッパ経由で無事帰国できたのだが、それにしてもとんでもない事件だった。

この類い稀な大規模テロに対して、米国は完全に逆上し、理性を失ってしまったようだ。大統領は「戦争だ」と叫び、報復攻撃だの、インターネットの監視を強化するだの、暗号ソフトに裏口を義務づけるだの……。困ったものである。暴力で暴力を封じ込めることなんて、できっこないのだ。仮に、今回のテロを起こしたテロリストのネットワークを殲滅したとしても、また別のネットワークが生まれるだけではないのだろうか。いつまでたっても、地上からテロと悲劇はなくならない。

もちろん無辜の市民の命を奪うテロに弁護の余地はない。世界中が理性的にこの問題を解決する努力を傾けるべきだろう。しかしそれとは別に、世界は何でもかんでも米国一辺倒というところから生まれているいろんな問題について、もう一度、考え直さなきゃいけない。

市民社会のグローバリゼーション

この数年、地球上いたるところでは、グローバル化の嵐の中で、さまざまな試行錯誤が繰り返されてきた。国と国とを隔てるあらゆる障壁を取り払って、豊かで平和な社会を作ろうということだ。決して悪い話じゃないのだが、多様な価値観や歴史、社会構造を持つ世界中の人々を簡単にひとつにすることなんて、そう簡単にできるものではない。特に、昨今のグローバリゼーションが、多国籍企業や投資家の論理だけで推進されてきたおかげで、本来歓迎すべきグローバリゼーションは、ずいぶんゆがんだものになってしまった。

ジェット機でツインタワーに突っ込んだことに比べれば規模は小さいけど、'99年のシアトルでの抗議行動から最近のジェノバ・サミットに至るまで、国際会議でマクドナルドやスターバックスに石が投げられたのだって、テロの一種と言えるかもしれない。両者は背景は違っても、その根っこには米国主導のグローバル化に対する不満があるのだろう。

しかし、世界のあちこちで、そんな企業主導のグローバル化とは違う、市民社会のグローバル化を模索する動きもある。

市民社会のグローバル化をリードしているのが世界各地の非政府組織だが、ぼくがあのときモンテビデオにいたのは、ICANNのモンテビデオ会議に出席するためだった。ご存じの通り、ICANNは数年前まで米国のIANAが管理していたインターネットのドメイン名やIPアドレス、プロトコルなどを管理する世界的なNGOである。米国の商務省の管轄下にある非営利法人であり、米

国の影響を完全に排除はできないまでも、グローバル化の時代にふさわしく、世界中のインターネット利用者が一般会員(At Large会員)として運営に参加することができる。

米国という国の面白いところは、一方では連邦政府が国益を理由に強引な行動を取る反面、もう一方では普遍的な民主主義を掲げる非営利団体やNGOが積極的な活動をしていることだ。ぼくがモンテビデオの会議に出席することができたのも、この一般会員制度を支援する「ザルツブルグ・プロジェクト」という旅費支援プログラムのおかげだった。このプログラムは、ICANNの一般会員制度を支援する米国のマークル財団のプロジェクトで、世界中のインターネットを利用する非営利グループから毎回数十人が会議に参加し、インターネットの管理について積極的な議論が行われている。

ICANNそのものが米国商務省の管轄下にあり、米国連邦政府の影響から無縁ではない組織ではありながら、ICANN自身はそれと一線を画そうとしている。これからのグローバル化にとって「脱米国」が重要なキーワードなのだ。

オープンソースが示すもの

そしてもうひとつの市民社会によるグローバル化の成功例は、いうまでもなくオープンソース・ソフトウェアだろう。

オープンソースの代名詞であるLinuxが米国ではない欧州で始まり、世界中のハッカーを巻き込んで成功したことは、誰もが参加できる開かれたグローバル化が、万人の支持を得られるものを作り上げることができることを立証した。オープンソースは、現在の米国主導のグローバル経済のように、国境を超えた企業の利益を保証するものではなく、国境を超えて人々の利益を保障する知的生産活動なのである。

国境を超えたコミュニケーション

10年ほど前の湾岸戦争の時、やはり世界中のメディアは戦争を叫び、それに反対する小さな声は、圧倒的なマスコミの情報操作によって掻き消されてしまった。そして猛烈な空爆の中で多くのイラクの市民が死んでいった。しかしそれから10年、イラクの体制は変わらず、フセインも健在である。

今回の憎むべきテロに対して、米国がメンツをかけた戦争を始めようとしている。しかし、米国は今回のテロの首謀者と一方的に指名したビンラディン氏やテロのネットワークを壊滅することができるのだろうか。仮にそれができたとしても、米国をターゲットとする世界中のテロリストを壊滅させるわけにはいかず、テロの恐怖を除くことはできないのではないだろうか。米国の大統領は、「米国人を見たら殺せというテロ組織と話し合いの余地はない」という。しかし、どんなテロ組織だって、普通の人々や市民の支援がなければ存続することはできず、どんなに対立している国の間でも、市民レベルでの話し合いは不可能じゃない。

国家や企業だけが独走するグローバル化が置き忘れた市民社会のグローバル化ということの意味を、世界の国々の指導者はもっと真剣に考えて欲しい。そして、国境を超えてコミュニケーションのインフラ作りに取り組んでいるわれわれオープンソースのコミュニティも、たまにはソースコードから離れて、そんな市民レベルでのコミュニケーションの可能性ということにも想像を巡らせてみてもいいのではないだろうか。

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

8・16　Palm、Beの技術を買収
8・27　Microsoft、Xboxの国内販売を2月に延期
8・28　Intel 2GHz Pentium4発表
8・29　Gatewayが日本市場撤退
9・3　HPがCompaqを買収

このところ不運続きです。8月に2週間のアメリカ取材が決まったら、家族から文句を言われ、アメリカでは事故るし（そのせいでLinuxWorldに行きそびれた）、帰ってきたら、スーツケースで足をケガするし、おまけに、ちょっとした手術で入院するハメに陥ってしまいました。なんかへんだと思って、佐野厄除け大師のホームページで調べてみたら、「男の大厄42才」で前厄になっているではないですか。なんと、前厄でこの程度なら、来年がもっと怖い……。やはり、初詣に行かなかったバチが当たったのか？　そういえば、子供の七五三もなんとか理由をつけて行かなかったしなぁ。退院したらお参りしないと。

HPがCompaqを買収

IT業界、今年最大のニュースに確定間違いなしなのが、HPによるCompaqの買収でしょう。ウワサによると、Compaqは、HPが買収を決める前に日本企業に買収を打診していたのだとか。また、アナリスト向けには、「サービスビジネスを強化することで体質改善」といった説明をしており、従来のハードウェアビジネスでは、今後苦しいということは以前よりわかっていたらしい。

ではなぜ、Compaqはこんなに苦しくなってしまったのだろうか？　いろいろと考えるに、DECなどの買収で規模的にも巨大になったCompaqは、かなり非効率な体質になっていたのではないだろうか？　DECといえば、かつては世界第2位のコンピュータメーカー。メインフレームで独走するIBMを追う立場にあったのである。それが、いわゆるダウンサイジングのトレンドに乗れず、PC関連のビジネスがうまくいかなかったために凋落。Compaqが買収ということになったわけである。おそらくこのときCompaqが欲しかったのは、世界各国にまたがる営業、サービス、サポート体制や教育システム、研究開発組織といったインフラ的な部分ではなかったかと思われる。

そして今回、HPがCompaqを買った理由も、旧DECの資産が欲しかったのではないだろうか？　いままで、PCメーカーは、単なるハードウェアメーカーに徹していればよかった。しかし、Dellのような直販メーカーが登場すると、コスト競争では従来型の代理店、小売店を経由したハードウェアビジネスは、ほとんど成り立たなくなりつつある。

そうなると、必然的にユーザーに対して、ハードウェアだけでなく、ソリューションを販売するサービスビジネスなどへシフトしていかねばならない。しかし、これには、それなりの人員やインフラが必要である。

かつてのメインフレームメーカーのうち、IBMが生き残っているのは、メインフレームがもともとそういう商売だったからである。それ以外のメインフレームメーカーは、多くが撤退か、サービスビジネス中心に切り替わっていかざるを得ない状態である。

苦しい状態は、HPも同様である。昨年、本業ともいえる測定器部門を分社し、HP自身はコンピュータ専業メーカーとなったが、ハードウェアの単体販売が苦しいのは同じ。やはりサービスビジネスに移行する必要がある。だが、HP自身は分社化で組織的にはまだ脆弱な部分があった。軽い組織というのは、確かに動きやすいが、サービスなど人海戦術が必要な分野ではやはり不利。そこで目をつけたのが、Comapqが持っていた旧DECの資産というわけだ。

巨大IT企業（たしかに一時Compaqは世界最大のコンピュータメーカーとなった）になろうとDECを飲み込んだ

のはいいが、結局自らもHPに飲み込まれてしまうという皮肉な運命。

このニュースの直前にゲートウェイが日本市場から撤退というニュースがあって、ちょっとびっくり。ニュースが出たときには、すでに営業を停止していたとか。いやー、撤退するのも素早いね。米国の企業って、こうした吸収や合併、判断がダイナミックで、それが世界をリードする実力になっている感じ。日本のコンピュータ関連企業もみな苦しそうだが、日本の場合、「総合電器メーカー」がコンピュータメーカーを兼ねていたり、財閥、系列なんてのがあって、米国に比べるとこうした派手な動きがあまりない。昨今の不況を考えると、日本でもコンピュータビジネスを再考するところが出てくるのかも。

インテルついに2GHz達成

8月の2週間のアメリカ取材の目的のひとつは、インテルが開催するIDF（Intel Developer Forum）だったのだが、そこでインテルは、ついに2GHz版のPentium 4を発表した。派手な発表だったけど、AMDが1GHzを達成したときのほうが「大台に乗ったぁ！」って感じで、騒ぎが大きかったような気もする。インテル自身もそれをわかっていたのか「GHzの先にあるもの」なんてイメージで、いろいろな技術発表を用意していた。

Compaqは今年6月に、Alphaの開発資産をインテルに売却し、Alpha製品ラインの段階的な縮小を決めていた。そのとき開発中だったのが、Thread Level Parallelismと呼ばれる技術で、CPU内部の並列実行をスレッドレベルにまで引き上げるもの。簡単にいえば、1つのCPUの中に論理的に2つのCPUを入れるものなのだが、今回インテルは、現在のXeon（Pentium 4ベース）に取り入れる技術としてHyper Threadingを発表している。

Hyper Threadingは、バスや実行ユニットを共有する2つのCPUを1つのパッケージに入れたもの。現在のCPUでも複数命令を同時実行しているが、2つのスレッドを実行すると、互いに依存しない同時実行可能な命令が増えるため、1つの実行ユニットであっても実行効率が上がる。インテルによると30%以上の性能向上があるのだとか。また、ハードウェア的には、1つのCPUとなるので、システム全体としてもマルチプロセッサよりも安価になる。どうせCPUを2つ入れても2倍速くはならないんだから、見かけ上デュアルプロセッサにして3割以上速くなればいいんじゃない、という考え方だ。

もうひとつの目玉は、新しいI/Oアーキテクチャである3GIO（仮称）である。今回のIDFでは、その概要が公開された。これによると、1方向のシリアルライン2つを組み合わせて1つのリンクを作るものらしい。高速な転送が必要なときには、このリンクを複数組み合わせて、転送速度を向上させる。リンク1つを1xと呼び、4つ組み合わせれば4xだ。1つのシリアルラインは、差動型で2つの信号線からなる。つまりたった4本の信号で従来のPCIバスを置き換えるものにするらしい。

PCIを管理するPCI-SIGは、この3GIOをPCI 3.0仕様のひとつとして取り入れる予定。PCIコネクタの横に3GIOコネクタを置いた共有型スロットみたいなものを考えているようだ。こちらは2004年ぐらいから登場ということで、ちょっと先の話。

そのほかの目玉として、Itaniumの次期コアであるMckinleyやその次のCPUであるMadisonの話がもう出ている。ようやく製品が出荷されたばかりというのに、結構気が早い感じだが、IDFは開発者向けのイベント。これぐらい先の話がないと参加する意味もない。

Xboxってどうなんでしょ

9月の14日に発売された任天堂のゲームキューブだが、なんでも外国人のゲームマニアが秋葉原に大挙してやってきては朝から並んでいたのだとか。

PlayStation2、ゲームキューブときて、次に出る予定なのがMicrosoftのXboxなのだが、日本国内での発売を2月22日に延期した。もっとも、北米での発売は、11月8日である。今年のクリスマス商戦は、すでに先行しているPS2と新規登場のゲームキューブ（米国では11月18日発売とか）、そしてXboxの三つ巴になるわけだ。

それで日本なのだが、クリスマスには間に合わないし、2月では正月のお年玉ねらいにも遅すぎる。なにせ、オモチャ屋は12月、1月が稼ぎどきだからね。日本の子供は2月までお年玉なんか持っちゃいないよ。となると、日本でのXboxは、かなり厳しいビジネスになるのではないかとの観測がもっぱらだ。

IDFの情報はWebで入手できる
http://www.intel94.com/

正しきタコへの道

初級Linuxer養成講座

Linuxのコミュニケーションツール

「パソコンは重要なコミュニケーションツール」という文句は、いやになるほど聞かされていて当たり前のことになったが、Linuxでもそれは同じことだ。Webや電子メールなどのネットワークアプリケーションが使えるのはもちろんのことだが、もっと気軽な文字によるコミュニケーションツールも利用できるのだ。今回はそのようなツールをいくつか紹介しよう。

文：竹田善太郎
Text : Zentaro Takeda

単なる**物見遊山**が主な目的だが、若干の「情報収集」も兼ねて、1年に2～3回、定期的に米国を訪問するようになってから数年経つが、最近まで彼の地で**携帯電話**を必要とするような事態はほとんどなかった。**スポンサーなし**で、気ままにふらふらする旅行がほとんどだったので、定期的に連絡をとったり、移動中につかまらないと困るような用事がないからだ。

それでも、レンタカーで移動したり、乗る予定の飛行機の便がキャンセルになったり、ホテルを急に移らなければならないような事態が起こったりすると、やはり電話が手元にあると安心である。

そんなわけで、去年あたりから米国で利用できる携帯電話についていろいろと調べてみた。まず、もっとも一般的と思われる**レンタル携帯電話の料金があまりにも高価**なのに驚いた。基本料金として、相当の金額を取られるうえ、通話料金が1分300円もするなど、個人がポケットマネーで**気軽に**使えるような代物ではない。

昨年あたりから使えるようになった次世代携帯電話（CDMA方式）では、「国際ローミング」のサービスがあると聞いてこれも調べてみたが、通話料はレンタル携帯電話と変わらず高い。しかも、米国内にいて米国内の電話にかけるときでも、**国際通話と同じ料金**がかかるという。さらに、通話可能なエリアがかなり限られていて、自分の立ち回り先ではほとんど利用できない。

それでは、日本国内でも旅行者に人気のある**プリペイド携帯電話**はどうかと調べてみると、米国内には携帯電話サービスを行っている会社が多数あるのだが、調べた範囲内ではAT&Tワイヤレス社のみがプリペイド携帯電話を販売していることがわかった。アナログとデジタルの両形式に対応するので、米国内ほぼすべての地域で利用可能なうえ、日本との間の国際電話も問題なくできる。

通常の携帯電話よりは若干通話料金が高いが、**レンタルの携帯電話やホテルの電話に比べればはるかに安い**。通話料金の補充はインターネットのホームページからも可能なので、日本に居ながらにして、残額の管理ができるのもよい。

そんなわけで、今年の1月にこのプリペイド携帯（**写真1**）を入手して、使っているのだが、**ちょっとした問題**があることがわかった。この8月に出かけたサンフランシスコの周辺地域（いわゆる「ベイエリア」）に限って、このプリペイド携帯を扱っている店舗が**見当たらない**のである。よくよく事情を聞いてみると、同じ「AT&Tワイヤレス」の看板を出していても、その実態は別会社で、全米でサービスをしている「AT&Tワイヤレス」とは、

写真1　AT&Tワイヤレスのプリペイド携帯
買った当時は「Prepaid Advantage」という名前のパッケージで販売されていたが、現在では「Free 2 Go Wireless」というような商品名に変わっているらしい。ちなみに電話機の機種は、Nokiaの5160iで、日本でいうところの「ショートメッセージ」の送受信にも対応している。オプションのフェイスプレートを自由に交換できるところもよい。

扱っている製品やサービスが違うのだそうだ。プリペイド携帯電話も売っていないわけではないのだが、ここで扱っていたのは、ベイエリア内だけで使える**地域限定のプリペイド携帯電話**だけで、全国版プリペイド携帯を扱っていないばかりか、通話時間を補充するためのカード（スクラッチカードのようなもの）すら置いていない。

もちろん、全国版プリペイド携帯の通話には支障がなかったのだが（なぜか電話機の電話帳機能を使って電話をかけることだけができなかった）、この時期だけ、たまたまなのか、インターネットからの通話時間の補充ができない状態だったため、**残りの通話可能時間を気にしながら**通話することになって、ちょっと不便だった。

思えば、日本の国鉄が分割民営化で「JR」になったとき、区域をまたいだ区間の切符を買うのが面倒だったり、精算に時間がかかるなどの不便な思いをさせられたものだか、今回の携帯電話の件も、根っこは同じようなものだと感じた。公平な競争も大事だが、業者間の**なわばり**のために、**ユーザーが不便な思いをさせられる**のは真っ平ごめんだ。

もっとも原始的な「チャット」機能

さて、今回からは「シェル本体」から少し離れて、Linuxのコマンドラインで使える**ちょっと便利なコマンド**を紹介していくことにしよう。

以前、本連載でもちょっと触れたのだが、1台のUNIXマシンを何人ものユーザーが共有していた時代には、

```
$ echo "Hello. How are you?" > /dev/ttyp2
```

といったように、別のユーザーの「仮想端末」に対して文字列メッセージを送りつける**いたずら**がよく行われていた。いわゆる、**簡易チャット**として使われることもある方法なのだが、相手先の画面にいきなり文字列が表示されるので（**画面1**）、よくよく知った相手以外にこのようなことをすると、先方を**怒らせて**しまうこともある。しかし、他の人に知られないように食事に誘ったりするのには便利な方法である。

ただし、この方法は、/dev/tty??のデバイスファイルのアクセス権設定によっては使えないことも多く、また、相手がどの端末を使っているかを、wコマンドなどを使ってあらかじめ調べておかなければならないので面倒だ。

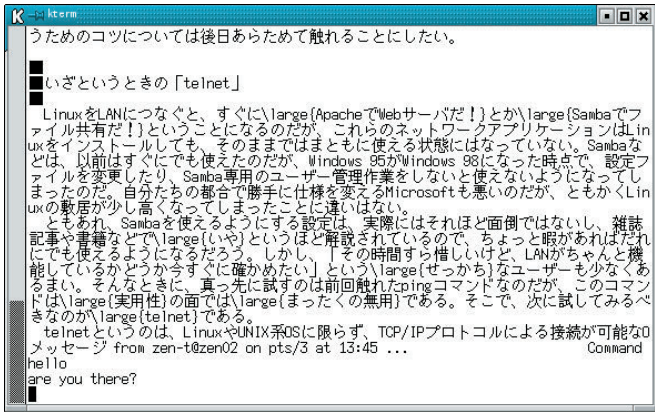
実は、先ほどの例のような「echoコマンド」を使わなくても、ほかの端末の画面にメッセージを表示する専用のコマンドが用意されている。今では、使っている人は**ほとんどいない**のかもしれないが、他人にメッセージを送る以外の使い道もないわけではなく、また、Linuxの仕組みを知るのに役に立つかもしれないので、このようなコマンドがあることだけでも知っておいて損はないだろう。

writeコマンド

相手の端末に文字列を送りつける機能だけを持つ、もっとも簡単なコマンドが**writeコマンド**である。これは、echoコマンドを使った例である

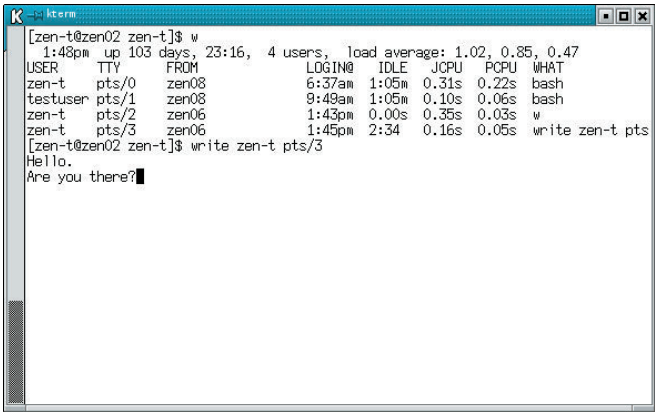
```
$ echo "hello" > /dev/pts/0
```

と、まったく同じことを行うコマンドだ。同じマシンにログインしている別のユーザー（たとえばuser1）にメッセージを送りたいときには、



画面1　端末に突然届く文字メッセージ

テキストエディタなどでの作業中は、画面が崩れてしまうのでびっくりさせられる。もちろん、Emacsやviなら「Ctrl-L」を入力すれば、画面を再描画できるので問題はないのだが……。



画面2　writeコマンドの使い方

相手が1人で複数ログインしているときは、ユーザー名のあとに端末名も指定する。使い方は、送信したい文字列をどんどん入力するだけでよい。入力がすんだら、Ctrl＋Dキーを押せば、入力済みの文字列が相手先に送信される。

```
$ write user1
```

を実行する。続いて、送信したい文字列をキーボードから入力すれば、その文字列がそのまま、相手先の画面に表示される。入力を終えるときには、**Ctrl＋Dキー**を押す（**画面2**）。この操作は、データの終わりを表すときの決まり文句だ。

writeコマンドは、このように文字列を相手先の端末に送りつける機能しか持たない。メッセージを受け取った相手が返事を返したり、相手がメッセージを読んでいるかどうかを確認することはできない。まさに、**一方的なコミュニケーション**なので、あまり便利には感じられないかもしれないが、自分自身にメッセージを送ることもできるので、たとえば次のような使い方もできる。

```
$ (sleep 100m; write user1 << EOF)&
> meeting at room #3
> EOF
```

ここで、「<<」という見慣れないリダイレクト記号を使っているが、これは「ヒアドキュメント」という機能で、「これから入力する文字列を、EOFという文字列が現れるまで、直前のコマンドの標準入力として送る」という意味になる。ヒアドキュメントについては、いずれ説明することにして、ここではこのような決まり文句として見ていてほしい。

上のコマンドラインの意味は、**100分待ったあとで（sleepコマンド）、user1宛に“meeting at room#3”という文字列を送りつける**ということになる。最初のコマンドライン行に「&」をつけているので、このコマンドラインはバックグラウンドジョブとして実行される。つまり、簡単なアラーム機能として使ってみたわけだ。

user1の部分を自分自身のログインIDにすれば、一定時間後に自分自身にメッセージを送ることもできるわけだ。

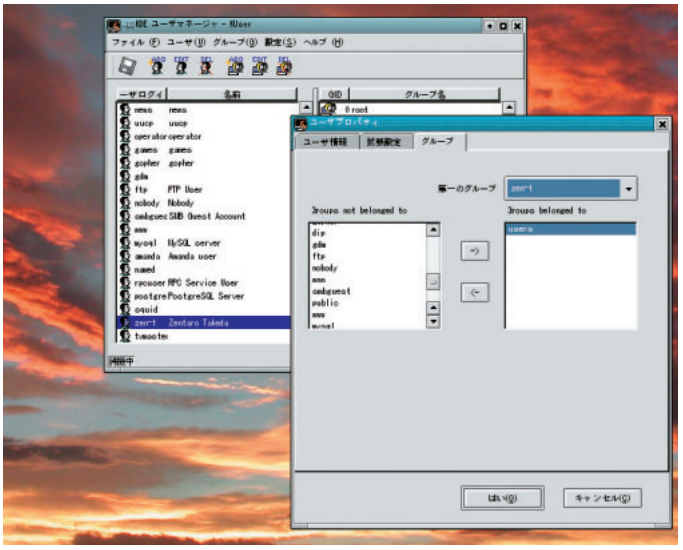
writeコマンドがうまく使えないこともある。とくに、最近のLinuxディストリビューションでwriteコマンドを使って別のユーザー宛にメッセージを送ろうとすると、

```
$ write user2
```

```
write: /dev/pts/1: Permission denied
```

というエラーメッセージが返されることが多い。これは、別のユーザーから自分の仮想端末への書き込みを行えないような設定になっているからだ。

ユーザーIDが異なっていても、グループIDが同一ならwriteコマンドが使えることもある。ただし、デフォルトの設定のままユーザーアカウントを作成していると、それぞれのユーザーIDに別々のグループIDが割り当てられていることが多いので、ユーザー管理ツールを使って、writeコマンドでメッセージ交換をするユーザーには同じグループIDを割り当てるようにするしかないだろう（**画面3**）。



画面3　ユーザー管理ツール（KDE環境の場合）
writeコマンドでメッセージを送信できるようにするには、同一のグループに属するようにユーザーを設定しなければならないことが多い。

wallコマンド

writeコマンドをさらに**強力**にしたコマンドが**wallコマンド**である。このコマンドは、マシンにログインしている**すべてのユーザーにメッセージを送る**という、なかなか**強権的**なコマンドである（**図1**）。

wallコマンドは次のように使う。

```
$ wall
Hello, I'm your master.
(Ctrl+D)
```

wallコマンドを実行してから、適当な文字列を入力し、最後にCtrl＋Dキーを押すと、ログインしているすべてのユーザーの端末上に、**画面4**のようなメッセージが表示される。

このメッセージを見て、**マシンをシャットダウンするときに似たようなメッセージを見たことがある**のに気がついた人がいるだ

ろう（**画面5**）。まさにそのとおりで、Linuxを停止させるときに使うshutdownコマンドは、内部でwallコマンドを呼び出して、ログインしているユーザーに対して「マシンをシャットダウンするから作業を中止せよ」という意味のメッセージを送信しているのである。

wallコマンドを一般のユーザーが使うようなことは、あまりないかもしれない。時節柄、ちょっと不謹慎なたとえかもしれないが、たとえばオフィスにテロリストが乱入してきたので、社員全員に**逃げろ**という緊急メッセージを送るような場合でもない限り、使う必要はないだろう。あとは、システム管理者が、ユーザーにメンテナンス上の情報を送ることくらいしか思いつかない。

ちょっと話はそれるが、writeコマンドも同様に、システム管理者が特定のユーザーに**緊急メッセージ**を送るときに使う場合が多かった。昔は、みんなで共有しているマシン上で、負荷の大きなコマンドを実行すると、システム管理者がすかさずwriteコマンドを使って、

```
Do not use Emacs on this machine!
Or I will remove your account!!
```

などという**脅迫めいたメッセージ**を送ってきたものだった。今なら、**Emacsを使ったくらいで怒られる**ようなことはないだろうが、会社のコンピュータを使って、ちょっといたずらをしているときに、上のようなメッセージが表示されたら、**素直に謝って**、事を荒立てないようにしたほうがよいだろう。

talkコマンド

いままでのwriteコマンドとwallコマンドは、文字のメッセージを一方的に送りつけるだけのコマンドだったが、双方向に文字メッセージを送り合う、いわゆる**チャット**を行うための**talk**というコマンドもある。文字通り「会話」をするためのコマンドだ。

talkコマンドは、相手先の端末に直接文字列を送りつけるのではなく、**talkd**という専用のプログラムを仲介してメッセージのやりとりをするので（**図2**）、writeやwallよりもずっと**進化した**メッセージ交換の手段である。

talkコマンドは、次のように使う。

```
$ talk 相手のユーザーID
```

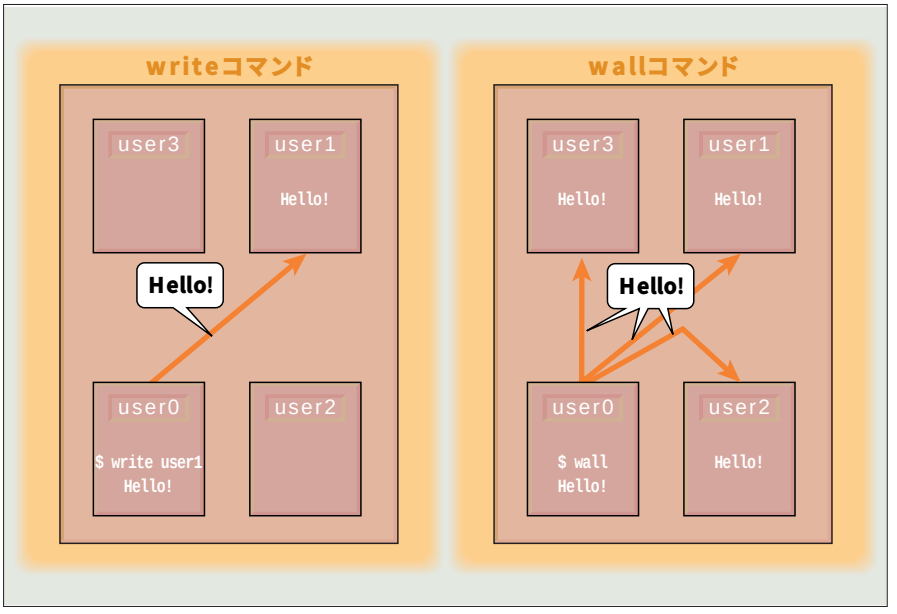
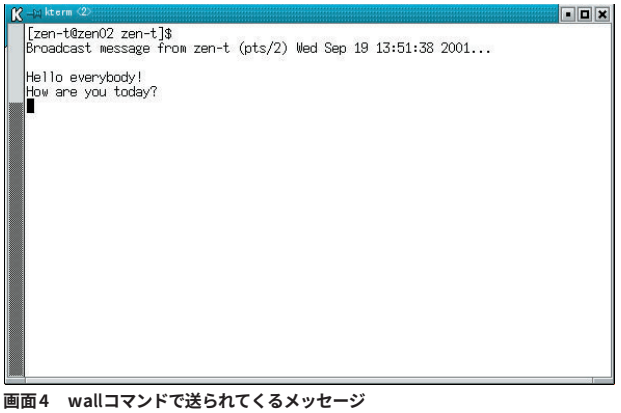
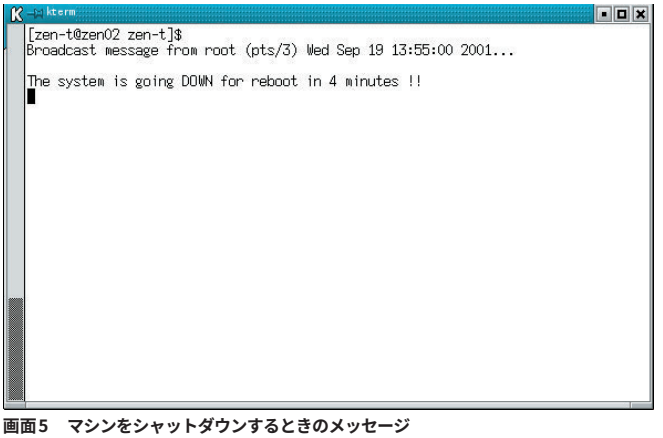


図1　writeコマンドとwallコマンドの違い



画面4　wallコマンドで送られてくるメッセージ
writeコマンドのメッセージに似ているが、先頭部分が「Broadcast message from....」となっているところが違う。



画面5　マシンをシャットダウンするときのメッセージ
メッセージの内容をよく見れば、wallコマンドと同じであることに気づくだろう。

たとえば、user2というユーザーIDの相手とチャットをしたい場合には、コマンドラインで次のように入力する。

```
$ talk user2
```

すると、user2の端末には、**画面6**のようなメッセージが表示されて、user1がtalkコマンドを使って話したがっていることを通知する。user2は、チャットに応じる場合には、次のコマンドラインを実行する。

```
$ talk user1
```

するとuser1とuser2の両者とも、**画面7**のような状態になる。画面は上下に2分割されていて、上側には自分が入力した文字列が表示され、下側には相手が送ってきた文字列が表示される。チャットを終了したいときには、Ctrl＋Cキーを押せばよい。

パソコン通信やインターネットプロバイダが提供しているチャットシステムでは、ユーザー名（ハンドル名）とメッセージがずらずらと表示されることが多いが、talkコマンドの場合は、送信した文字列と受信した文字列が分かれて表示される。

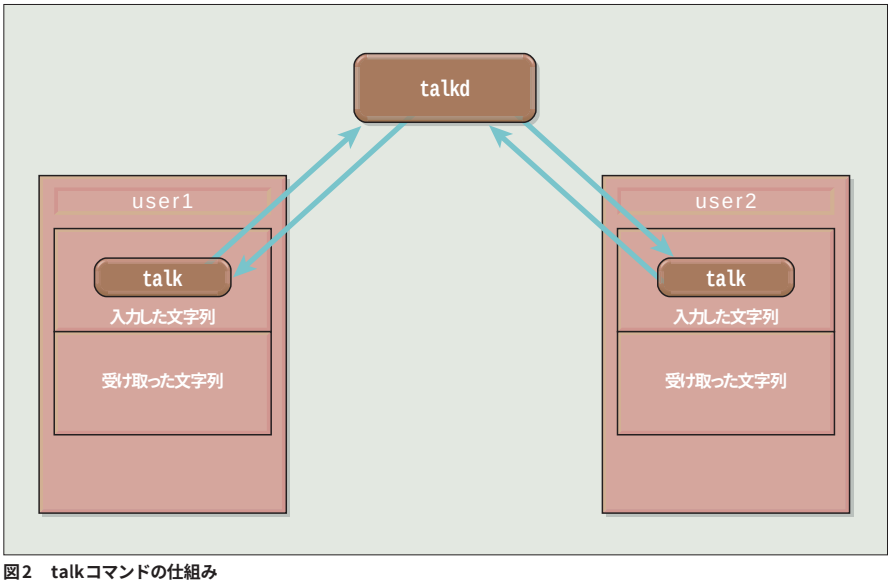


図2　talkコマンドの仕組み
メッセージを直接仮想端末に送るのではなく、「talkd」というプログラムがメッセージのやりとりを仲介している。したがって、talkdが起動していない状態ではtalkコマンドは使えない。

■日本語は使える？

送信元と受信先の日本語環境の設定が一致していれば、**日本語のメッセージも送受信できる**。ただし、使っているターミナルエミュレータの種類や、日本語入力システムの種類によっては、うまくいかないこともある。

■ネットワークを介した接続

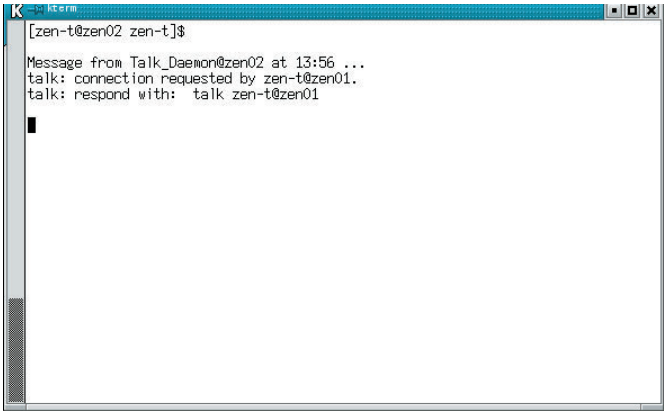
talkコマンドは、同じマシン上だけでなく、ネットワークで接続されている別のマシンにログインしているユーザーとも対話できる（**図3**）。たとえば、linux01というマシンにログインしているuser1が、linux02というマシンにログインしているuser2と対話したいときには、次のようにtalkコマンドを起動する。

```
$ talk user2@linux02
```

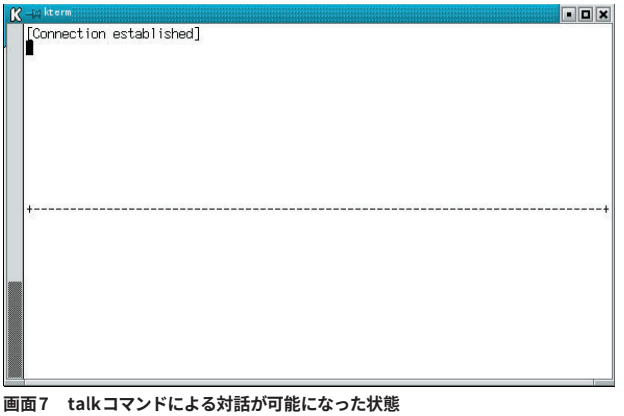
すると、linux02上のユーザーuser2の画面上には、**画面6**のように、マシンlinux01のユーザーuser1が、talkコマンドで話したがっていることが通知される。そこで、

```
$ talk user1@linux01
```

と入力すると、**画面7**と同様に、2つの



画面6　talkコマンドで「呼びかけられた」ときの画面
どのマシンの、どんなユーザーIDから呼びかけがあったかが表示される。



画面7　talkコマンドによる対話が可能になった状態
画面上半分は自分が入力した文字列、下半分には相手が入力した文字列がリアルタイムに表示される。

マシンにまたがった2人のユーザーの間で、文字によるチャットができる状態になる。

ネットワークを介したtalkコマンドは、LANだけでなく、インターネットを経由しても使えるようになっている。残念ながら、セキュリティ上の心配事が多いため、インターネットを介したtalkコマンドは使えないように設定されている場合がほとんどのようだ。

現在は、インターネットを使ったチャットをしたいのであれば、それ用のツールが別に用意されているので、わざわざtalkコマンドを使わなくてもよいのだが、インストール作業やユーザー登録などの手間が必要になることも多い。同じオフィスの同僚とちょっとチャットしたいというときには、talkコマンドを覚えておけば便利だろう。ただし、生産性が下がるので、**業務時間内のチャットは禁止**という会社も多いようなので、talkコマンドを使っているのを見つかって**クビになってしまった**ということのないよう、注意してほしい。

メッセージを拒否したいときは？

仕事が忙しかったり、テキストエディタなどを使って微妙な作業をしている最中に、writeやtalkで呼びかけられると、迷惑に思うこともあるだろう。

このような場合には、write、wall、talkなど、一切のメッセージを拒否することもできる。メッセージの受け取りを拒否するか、許可するかを設定する**mesgコマンド**を使うのだ。

mesgコマンドの使い方は簡単である。メッセージの受け取りを拒否したい場合には、次のコマンドラインを実行する。

```
$ mesg n
```

反対に、受け取りを許可したい場合には、次のコマンドラインを実行する。

```
$ mesg y
```

「mesg n」を実行しているユーザーに対して、たとえばtalkコマンドを実行してみると、**画面8**のようなメッセージ表示されて、相手先がメッセージの受け取りを拒否していることがわかる。

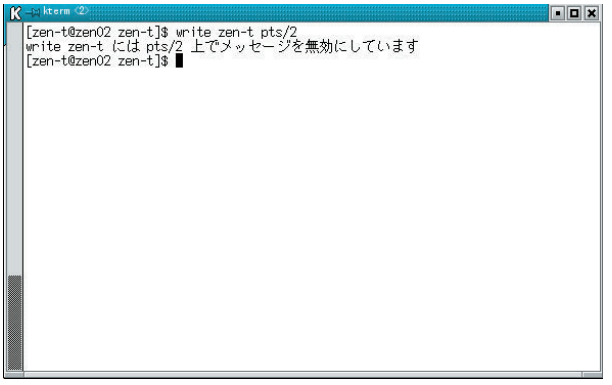
現在自分がメッセージ受け取りを許可しているかどうかは、オプションをつけずにmesgコマンドを実行すればよい。

```
$ mesg
is y
```

上のように「is y」と表示された場合は、「mesg y」、すなわち、メッセージの受け取りを許可していることを示す。「is n」と表示された場合には、「mesg n」の状態で、メッセージ受け取りを拒否していることを表す。

なお、「mesg n」の設定をしていても、**rootユーザーからのメッセージだけは拒否されない**。したがって、マシンをシャットダウンするときの通知などは、通常どおり表示される。

画面8　メッセージ受け取りが拒否されたときのメッセージ
これはwriteコマンドの例。talkコマンドのときは「your party is refusing messages」などと表示される。



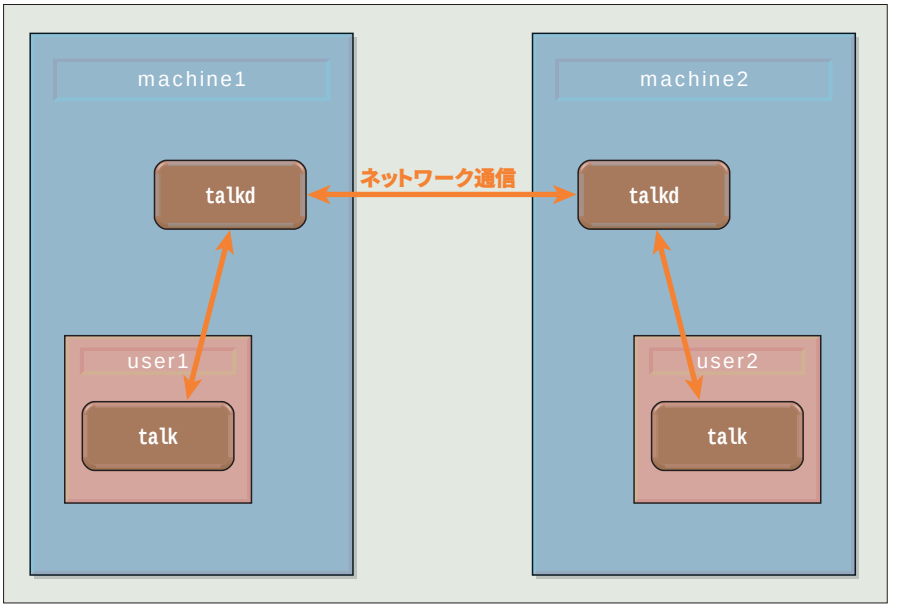


図3　ネットワークを介したtalkコマンドの動作

新連載

初等 Ruby 講座

Ruby知ってますか。Rubyは便利なんです。それになにより気持ちがいいんです。今回から始まるこの連載では、Ruby作者自らが、Rubyを知らない人でもRubyを使えるようになるような解説を試みます。乞うご期待。

第1回　はじめの一歩

文：まつもと ゆきひろ
Text : Yukihito "Matz" Matsumoto

はじめまして。まつもと ゆきひろと申します。一部では“Matz”としても知られていますが、Rubyを作っている人です。

私は頼まれるとなかなかイヤと言えない性格なので、以前からおつきあいのあったLinux magazineの編集の人からの「Ruby入門の連載をお願いします」という依頼に、つい「はい」と答えてしまいました。

開発者としてRubyとは初めからの付き合いですし、それも今年でもう8年にもなります。正直なところ、初心はすっかり忘れてしまいました。

問題は初心を忘れているという点だけではなくて、そもそもちゃんとした初心者向けの文章を書いたことがないっていうことのほうが重要な気がします。そういえば、出たばっかりの『Rubyを256倍使う本 太陽編』じゃなかった『黄道編』でも、私の書いたREADME.EXT.jpという文書（拡張ライブラリの書き方を解説している）に対して、

Rubyの内部構造を知っている人が書いている

と断言されていました。一瞬「そりゃ内部構造を知らなきゃREADME.EXT.jpは書けないでしょ」とツっこんだのですが、要するに「内部構造を知っている人にしか読めない文書だ」という意味なんでしょう。やはり初心者向けの連載を引き受けたのは無謀であったか……。

というわけで（なにが「というわけ」なんだか）、入門向けのわかりやすい文章を書く自信はまったくないのですが、それなりに努力しますのでよろしくお願いします。

皆さんの評判が良ければ、連載が続くかもしれません。

連載の構成

本連載は毎月以下の3部構成にしようと考えています。

● Ruby入門

この連載の本文です。できるだけRubyのことをまったく知らない人も学べるような内容にしようと思っています。みなさんにとっては一番やさしく、私にとっては一番むずかしい部分です。

●知られざるRuby

見かけは単純でも奥が深いことが多いのがRubyです。このパートではRubyの一部を取り上げて、かなり突っ込んだ内容まで解説します。初心者を卒業した人にも役に立つ内容にしたいと考えています。

● Ruby開発日記

せっかく開発者自らによる連載なのですから、開発者でないと書けないことも載せたいと思って作ったのが、この「Ruby開発日記」です。Ruby開発の最新トピックや、私自身の近況を毎月少しずつ紹介しようと思います。

Rubyとは

前振りはここまでにして、ここからが本文です。今回は初回ですから、そもそもRubyとはいったい何かという点から始めたいと思います。ひと言で言ってしまうとRubyは、

オブジェクト指向プログラミング言語

なのですが、これではわかる人にしかわかりませんから、もうちょっとていねいに説明します。まず「オブジェクト指向」はとりあえず置いておきましょう。今月は「オブジェクト指向」の話はしません。で、残りの「プログラミング言語」のほうですが、こちらは「プログラムを作るときに使う言語」です。

コンピュータは人間に代わっていろいろな仕事を速く正確に行ってくれますが、正直なところあまり頭は良くありません。SF映画に出てくるような柔軟な発想で人間を助けてくれるパートナーなんてまだまだ夢のまた夢です。現在のコンピュータが仕事をしてくれるのは、だれか人間がその仕事の手順を正確にコンピュータに教えてあげてくれたからです。その手順書を「プログラム」と呼びます。

繰り返しになりますが、コンピュータはあまり頭が良くありませんので、この手順書を人間の言葉で書いても読めません。ですから、コンピュータにもわかる言語で書いてやる必要があります。これが「プログラミング言語」です。

プログラミング言語にはいろいろな種類があります。たとえば、BASIC、FORTRAN、C、Pascal、Lisp、Perl、Smalltalkなどが挙げられます。いくつかは聞いたことがあるのではないでしょうか？

Rubyはそれらと同じプログラミング言語です。すなわち、Rubyを使うということは、プログラムを作る、つまりコンピュータに仕事の手順を教えてやるということなのです。

なぜRuby？

世の中にプログラミング言語がいくつあるのか正確に知っている人は誰もいません。以前、数千はあるのではという文章を読んだことがあります。まあ誰も知らないマイナーなものはこの際置いておくとしても、相当メジャーなものに限っても両手では足りないくらいあるでしょう。

その中で、なぜRubyを使うのか、あるいはなぜRubyを学ぶのか、という疑問が出てくるのは当然だと思います。ここで「学ぶ必要はない」と答えてしまうと、私のRubyの作者としての面目は丸つぶれですし、この連載も初回にして最終回になってしまいます。それは私にとって個人的に非常にまずいので、とりあえず「学ぶ必要はある」として考えましょう。

他の言語に比べて、Rubyが優れている理由はいくつかあるのですが、代表的なものを列挙しておきます。

●手軽に実行できる

Rubyのプログラムは書いてすぐに実行できます。これはRuby言語を解釈するプログラム（これをインタプリタと呼びます）が、プログラムを読み込んですぐに実行するからです。一方、C言語などの場合では、プログラムはまずコンピュータが直接実行できる形式に変換するプログラム（これをコンパイラと呼びます）が解釈しますから、実際に実行する前に1ステップ必要になります。このコンパイルと呼ばれるステップは、場合によっては数時間かかりますから、すぐ実行というわけにはいきません。

●機能が強力

あなたが自分でプログラムを書いてコンピュータにやらせたい仕事はなんでしょうか？　人によってそれぞれだと思いますが、Linux magazineの読者の皆さんならば、テキスト処理やファイル操作、システム管理などの仕事が多いのではないでしょうか。Rubyはそういう仕事が大の得意です。Rubyはこの辺りを先輩にあたる言語であるPerlから学びました。

●わかりやすい

わかりやすいと感じるかどうかは人によって違うと思うので、この「わかりやすい」という長所には主観がたっぷり入っています。すくなくとも私はわかりやすいプログラムが書きやすいようにRubyを設計しました。もっともプログラムがわざわざわかりにくくなるように言語を設計する人はあんまりいないと思います。いや、世の中にはBrainf＊ckという言語のような「わかりにくいことそのものが存在意義」というプログラミング言語もあるにはありますけど。

Rubyは文法がシンプルで同じ仕事を比較的簡潔に書けるという点、充実したライブラリのおかげで自分で書かなくてはいけないことが少ない点、それから今後説明するオ

ブジェクト指向という考え方に基づいて設計されているので言語に一貫性がある点などが、わかりやすさの理由になっていると考えています。

●フリーソフトウエア

私にとってあまりに当たり前なのでつい忘れてしまうのですが、人によっては重要な特長であると思います。Rubyはフリーソフトウエア、あるいはオープンソースソフトウェアであるため、利用に際して費用がかかりません。開発者である私に対しても謝礼や代金を払う必要は一切ありません。

サンプルRubyプログラム

くどくどと説明ばかりを読んでいてもイメージがわかないと思いますので、ここで実際のRubyプログラムをいくつか見てみましょう。まずは超簡単なものから。

```
puts 1+2
```

これは1+2を計算して出力するものです。putsは1行出力のための手続きです。別にRubyだから特別というふうには見えません。実際にはこのようなプログラムでも背後で「オブジェクト」がいろいろ動いているのですが、その話は来月以降ゆっくり説明したいと思います。

次のものはもうちょっと複雑です。以下のプログラムはコマンドライン引数として指定したファイルのそれぞれの行数を調べます。

```
 1  total = 0
 2  nfile = 0
 3  for file in ARGV
 4    n = 0
 5    File::foreach(file) do
 6      n += 1
 7    end
 8    printf "file %s has %d lines\n", file, n
 9    total += n
10    nfile += 1
11  end
12  printf "total %d files %d lines\n", nfile, total
```

1行目と2行目は変数の初期化です。総行数を保存するtotalという変数とファイル数を保存するnfileという変数を両方とも0にしています。

ARGVはコマンドライン引数です。for文で各ファイルに対して繰り返しています（3行目）。引数に指定したファイル名がそれぞれ変数fileに入って繰り返されます。

File::foreach（file）はfileで指定したファイルの各行に対して繰り返す手続きです（5行目）。今回は行数を数えるだけですから、ファイルごとの行数を変数nに足していくだけです。

各行に対する繰り返しが終了したら、printf手続きを使って、そのファイルの行数を出力し（8行目）、トータルの行数とファイル数を増やしておきます（9、10行目）。最後にトータルのファイル数と行数を出力して終わりです（12行目）。

他の言語でプログラムを書いたことがある（あるいは読んだことがある）人ならば、あまり説明しなくてもかなり推測できるのではないかと思います。この「常識が通用する」部分もRubyの長所のひとつです。

せっかくのプログラムですから、実行してみましょう。とりあえず上記のプログラム（行番号を除く）をエディタを使ってファイルにします。ここではcount.rbという名前だとします。これをrubyインタプリタの引数にします。それから行数を数えたいファイル名を続けて並べます。ですから、たとえばカレントディレクトリのヘッダファイル（拡張子が“.h”のファイル）の行数が数えたければ、

```
% ruby count.rb *.h
```

```
% ruby count.rb *.h
file config.h has 118 lines
file defines.h has 94 lines
file dln.h has 31 lines
file env.h has 60 lines
file intern.h has 414 lines
file node.h has 358 lines
file re.h has 41 lines
file regex.h has 228 lines
file rename2.h has 307 lines
file ruby.h has 623 lines
file rubyio.h has 66 lines
file rubysig.h has 90 lines
file st.h has 46 lines
file util.h has 53 lines
file version.h has 4 lines
total 15 files 2533 lines
```

画面1　Rubyプログラムの実行例

と実行します。出力例を**画面1**に示します。どうです？そんなにむずかしくないでしょう？

実はRubyを使わなくてもLinuxなら、wc -l *.hとすれば行数は簡単に求まるのですが、ここでは「自分でプログラムしたのだ」という満足感を得たのだということにしておきましょう（笑）。これは最初の一歩なのですから。そのうち、既存のコマンドでは簡単にはすまないこともできるようになります。

初めに下準備を

さて、Rubyを使ってみたくなりましたか？　え、ならない？　では、念力を送らなければ……。さあ、使いたくなりましたか？

では、Rubyを使ってみたくなったところで、お手元のマシンでRubyが使えるかどうか確かめてみましょう。

最近の多くのLinuxでは最初からRubyがインストールしてあります。試してみましょう。xtermなどの画面から、シェルプロンプトに対して以下のように入力します（"%"はシェルプロンプトです）。

```
% ruby -v
ruby 1.6.4 (2001-08-06) [i386-linux]
```

などと表示されたらRubyが使えます。バージョンや日付は異なっているかもしれません。使える人はラッキーでした。使えない人はなんとかしましょう。

インストール

というわけで、使えなかった人はRubyをインストールする必要があります。インストール手順はRuby公式サイトのhttp://www.ruby-lang.org/ja/install.htmlなどを参考にするとよいとありますが、若干情報が古いようです（最新は1.4.5とか書いてあるし）。

どうせですから、最新をインストールしましょう。Windows使いならともかくLinuxなら楽勝です。実際、私もLinuxでRubyを開発してますから。

まず、ソースコードを入手します。「超」最新安定版であるstable-snapshotが良いでしょう（「超」は「最新」にかかります）。

```
ftp://ftp.ruby-lang.org/pub/ruby/stable-snapshot.tar.gz
```

を手に入れます（付録CD-ROMにstabe-snapshot.tar.gzを収録）。同じディレクトリにあるruby-1.6.4.tar.gzがリリース安定版です。たぶん、今月号が出る頃にはruby-1.6.5.tar.gzに置き換わっていると思いますが。

ソースを入手したら、それを展開します。適当なディレクトリで以下のコマンドを実行します。

```
% tar zxvf stable-snapshot.tar.gz
```

つらつらとメッセージが出てrubyディレクトリができているはずです。あとはコンパイルです。以下の手順でコマンドを実行してください。

```
% cd ruby
% ./configure
% make
```

これでコンパイルは完了です。実際のインストールの前にテストしておきましょう。

```
% make test
test succeeded
```

と出れば、成功です。Linuxではコンパイルもテストも失敗しないはずです。あとはインストールです。rootになって、

```
% su
# make install
```

とすれば、インストールは完了です。もしあなたがroot権限をお持ちであれば、このようにLinuxでのインストールは簡単です。もし、rootになれないのであれば、つまりそのマシンにはほかに管理者がいるということですから、その管理者の方にお願いするのが一番手っ取り早いでしょう。

「ねえ、Rubyをインストールして」

もちろん、自分のホームディレクトリにインストールするという方法もあるにはあるのですが、やはりここはそのマシンを使うすべての人の幸せのため管理者の方にお願いするのが正解でしょう。

また、もしあなたがなんらかの事情でWindowsでRubyしたいと強く望んでいるのでしたら、

```
http://www.pragmaticprogrammer.com/ruby/downloads/ruby-install.html
```

からバイナリをダウンロードしたほうがよいかもしれません。ただし、私はWindowsのことはなんにもわかりませんから、くれぐれも私には質問しないでくださいね。お願いします。

Rubyの使い方

とりあえずRubyインタプリタの使い方を説明しておきましょう。といっても全然むずかしくなんかありません。

```
% ruby プログラム
```

「プログラム」の部分にはエディタなどで入力したRubyプログラムのファイル名が入ります。これでRubyプログラムが実行できます。

Rubyプログラムファイルには、たとえばsample.rbというように“.rb”という拡張子を付けるのが慣習になっていますが、別に付けなければならないというものではありません。

実際には、Rubyインタプリタはたくさんのコマンドラインオプションがあるのですが、とりあえず今回は初回ですから、必要はないでしょう。興味のある人は、

```
% ruby -h
```

と実行すると一覧を見ることができます。

対話型Rubyを使ってみよう

もうひとつ便利なツールirbも紹介しておきましょう。Rubyインタプリタがインストールされていれば、たぶんirbもインストールされていると思います。irbはinteractive ruby（対話型Ruby）の略です。irbを使えばプログラムを対話的に入力できるので、手軽に実験することができます。irbの起動にはただirbコマンドを実行するだけで構いません。

```
% irb
irb(main):001:0>
```

起動するとプロンプトが出てきますから、それに向かって実行したいRubyのプログラムを入力します。

```
irb(main):001:0> 1+1
2
irb(main):002:0>
```

複数の行に渡るプログラムもそのまま入力できます。

```
irb(main):002:0> def fact(n)
irb(main):003:1>  if n == 0
irb(main):004:2>    1
irb(main):005:2>  else
irb(main):006:2*   n*fact(n-1)
irb(main):007:2>  end
irb(main):008:1> end
nil
irb(main):009:0> fact(10)
3628800
irb(main):010:0> exit
```

プロンプトの最初の数字は入力した行番号、次の数字はネスト（入れ子）のレベルです。

irbを使えば、Rubyの挙動を簡単に確認できるので非常に便利です。やはり、実際に使ってみるのが一番勉強になります。

知られざるRuby
（中上級者向け）

open

今回のテーマはopenです。入出力をするためにファイルをオープンするのがopenです。openはKernelモジュールで定義されているメソッドで、Kernelモジュールはすべてのクラスの親であるObjectにインクルードされているので、openはどのクラスでも用いることができます。たとえば、このように使います。

```
f = open("/usr/share/dict/words")
```

```
f.grep(/Ruby/){|line| print line}
```

これは、/usr/share/dict/wordsにある辞書ファイルからRubyという単語のある行を探して出力する小プログラムです。

openメソッドの引数は以下のようになっています。

```
open(path[, mode[, perm]])
```

pathはオープンするファイルのパス名、modeはファイルの入出力モードを決定する文字列または整数フラグ、permはopenにより新しいファイルが作られたときのパーミッション（アクセス権）を整数で指定します。カギかっこ“[]”で括られた部分は省略が可能です。

modeを文字列で指定するには、**表1**の入出力モードを指定します。文字列によるモード指定では2文字目に“b”を追加することができます。これはバイナリモードでの読み込みを意味しますが、UNIXのような、もともとバイナリモードという概念がないOSでは“b”を指定しても単に無視されます。

整数フラグの場合、基本的な動作モードを**表2**の定数から指定します。それを補助する動作として**表3**の定数を、“|”（ビットOR）で複数指定することができます。これらの定数は、Fileクラスで定義されていますので、たとえば、

```
open(path, File::RDWR|File::CREAT, 0666)
```

のように記述します。

デフォルトの入出力モードは読み込みモード（“r”またはRDONLY）です。

新規に作るファイルのパーミッションを指定する第3引数permを省略した場合には0666を指定したことになります。0666というのは「獣の数」ではなくて、パーミッションを指定する8進数です。最初の0は、この数字が8進数であることを示します。続く3つの数字は、左からそれぞれ「ファイルの所有者」、「ファイルの所有グループに属するユーザー」、「それ以外のユーザー」に対するアクセス許可権を指定しています。8進数の1桁は3ビットなのですが、それの各ビットで「読み込み権」、「書き込み権」、「実行権」を表現します。6は2進数で表現すると110、つまり「読み込み権と書き込み権は許可するが実行権は許可しない」という意味になります。

これをまとめると、0666はファイルの所有者（自分が新規に作成するのだからプログラムの実行者と同じ）、ファイルの所有グループに所属するユーザー、その他のユーザーの誰もに対して、読み込みと書き込みを許可し、実行を許可しないパーミッションという意味になります。

ただし、この値がそのまま使われるわけではなく、プロセス単位のパーミッションマスク（umask）で修正され、（perm & ~umask）が実際の値になります。この値は新規にファイルを作るときにだけ有効で、すでに存在するファイルにpermを指定してオープンしたとしても、ファイルのパーミッションは変化しません。

以上のパーミッションの挙動はUNIX系OSでは共通のものです。では、そのようなファイルのアクセス権を持たないOSではどうなるのでしょうか？　それはそのOSでのPOSIXエミュレーション層の実装に依存するのですが、基本的には、ない袖は振れないので、存在しないアクセス権を指定しても無視するというのが基本です。

ファイルをオープンしたopenはFileクラスのオブジェクトを返します。openにはファイルだけでなく、コマンドを指定することができます。openはpathが“|”で始まるときには、それに続く文字列をコマンドとして起動し、そのコマンドの標準入出力に対してパイプラインをつなぎ、それに対応するIOクラスのオブジェクトを返します。pathが“|”で始まる場合には第3引数を指定するとエラーになります。

Perlを使っている人は注意する必要があるのですが、読み込みでも書き込みでも“|”は先頭に来ます。

さらにコマンド名が“-”であるとき、つまりpathに“|-”を指定した場合には、rubyインタプリタをforkし

指定する文字列	入出力モード
"r"	読み込みモード。ファイルが存在しなければエラーになる
"r+"	読み込み・書き込みモード。ファイルが存在しなければエラーになる
"w"	書き込みモード。ファイルがなければ作る。あればサイズを0にする
"w+"	読み込み・書き込みモード。ファイルがなければ作る。あればそのファイルのサイズを0にする
"a"	追加書き込みモード。ファイルがなければ作る。ファイル末尾にseekする
"a+"	追加読み込み・書き込みモード。ファイルがなければ作る。ファイル末尾にseekする

表1　openで指定する入出力モード（文字列の場合）

て子プロセスを作り、その子プロセスの標準入出力との間にパイプラインをつなぎます。

openがブロックとともに呼び出された時、openはファイルをオープンして、そのオープンしたファイルをパラメータとしてブロックを実行し、ブロックの実行が終了するとファイルをクローズします。

つまり、以下のようになります。

```
open(path, mode) do |f|
        ⋮
end
```

これは以下のコードとほぼ同じ動作をします。

```
f = open(path, mode)
begin
        ⋮
ensure
  f.close
end
```

これはつまりブロック付きでopenを呼び出した場合、途中でエラーが起きたり、例外などで実行を中断したとしても必ずファイルがクローズされるということです。この形式の利用は積極的に推奨されています。

この呼び出し方をした場合にはopenの戻り値はファイルのオブジェクトではなく、ブロックで一番最後に評価した式の値になります。これは、もうクローズしてしまったファイルオブジェクトよりもブロックの評価値のほうが使い道があるだろうという配慮からそうなっています。

openはファイルとコマンドを両方開くことができる高機能なメソッドですが、コマンドではなくファイルしか開かないことが明らかな場合、Fileクラスのメソッドを使うことができます。

```
File::new(path[, mode[, perm]])
File::open(path[, mode[, perm]])
```

これらのメソッドの引数の意味はopenメソッドと同様ですが、pathの先頭の“｜”をチェックしません。ですから（あまりないことですが）、ファイル名の先頭に“｜”が付いていてもオープンすることができます。また、これらのメソッドはFileクラスのオブジェクトを返します。

また、File::newはブロックを取りませんが（与えられても無視します）、File::openはopen同様にブロック付きで呼び出すことができます。動作もopenと同じです。これは、newはオブジェクトの生成に専念するというルールからで、ほかのクラスのメソッド、たとえばDir::newとDir::openでも同じような関係が成立しています。

ファイルを開くことだけが目的のFile::openとは逆に、コマンドを開くことが明らかな場合にはIO::popenが使えます。

```
IO::popen(cmd[, mode])
```

こちらにはcmdの先頭に“｜”は付きません（付けてはいけません）。IO::popenはopenメソッド同様にブロックを取ります。IO::popenの戻り値はIOクラスのオブジェクトです。

指定する定数	入出力モード
RDONLY	読み込みモード
WRONLY	書き込みモード
RDWR	読み込み・書き込みモード

表2　openで指定する入出力モード（整数フラグの場合）

orで指定する定数	入出力モード
CREAT	指定したpathにファイルが存在しなかった場合は作成する。この定数が指定されない場合、ファイルが存在していなければエラーになる
EXCL	CREATと一緒に使用された場合、ファイルがすでに存在した場合にエラーとなる
NOCTTY	指定したpathがterminalデバイスを参照していて、かつプロセスが制御端末を持たない場合でも、そのterminalが制御端末にならない
TRUNC	ファイルがすでに存在し、通常ファイルであり、書き込み可モード（RDWRまたはWRONLY）が指定されている場合、そのファイルの長さは0に切り詰められる
APPEND	ファイルを追加モードでオープンし、ファイル末尾にseekする
NONBLOCK NDELAY	ファイルをnonblockingモードでオープンする。openをはじめとする、ファイル・ディスクリプタに対するすべてのシステムコールは、ブロックしない。ブロックが発生しそうな場合にはシステムコールが失敗する
SYNC	ファイルを同期I/Oモードでオープンする。ファイル・ディスクリプタに対するすべてのwriteは、実際に書き込みが完了するまで終了しない

表3　openで指定する入出力モードの補助指定

Ruby開発日記

デンマーク訪問記

今回のトピックは、9月10日から14日までデンマークのオーフス市で開催されたJAOOのリポートです。

JAOOは毎年デンマークで開催されている国際的なJavaとオブジェクト指向のカンファレンスで、今年で5回目になるのだそうです（ちなみに国際カンファレンスということで、すべてのプレゼンテーションはデンマーク語ではなく、英語で行われました）。JAOOはなにの略かとスタッフに尋ねたところ、やっぱりJava and OOの略だそうです。なんか安易ですね。

ただ、カンファレンスは安易どころではなく、スピーカーにはアメリカから来た有名人がずらりとそろっています。たとえば、「リファクタリング」のMartine Fowlerや、Wikiの提唱者Ward Cunningham、Java方面でも知られているDoug Leaなど、どこかで名前は聞いたという人がたくさんです（敬称略）。

私はUML方面の人はあまり知らないのですが、そちらでも知られた人が来ていたようです。その中でもAlistair Cockburnという人がパワフルで印象的した。なんか有名人と直接会えて感動してしましました。なんだかミーハーですね。

私たちRuby関係者にとっての有名人といえば、「Programming Ruby」の著者であるPragmatic ProgrammersことDave ThomasとAndy Huntの2人もスピーカーとして参加していました。昨年のJAOOでは、彼らはRubyを紹介するプレゼンテーションを行い好評だったのだそうです。今年もプログラミング一般に関してのプレゼンテーションと、Rubyのチュートリアルとを担当していました。

実は、彼らの推薦もあって今年はRubyの作者を招待してのプレゼンテーションを、という話になったのだそうです。そうなんです。今回のデンマーク訪問は単なる参加者ではなく、講師としてプレゼンテーションするためだったのです。英語も怪しいのに大丈夫だろうかという不安もよそに、とにかく引き受けてしまったのだからしょうがない、と覚悟を決めて出かけてきました。悪天候のため飛行機のスケジュールが狂いまくりで、不安なものもありましたが、10日深夜に無事オーフスにたどりつきました。

私の出番は11日の午前のパネルセッション「The Role of Scripting Language」と午後のプレゼンテーション「Object-Oriented Scripting in Ruby」でした。パネルセッションでは、わたし、「Practical Programming in Tcl and Tk」の著者であるBrent Welch、それとPerlの偉い人Michel G. Schwernの3人で「スクリプティングとはなにか」というテーマで討論しました。といってもわずか45分しかありませんでしたから、それぞれの（言語の）立場を紹介しただけで終わってしまいました。

私としては「スクリプティングとはたんなる短いプログラムのことではない、人間指向のプログラミングだ」とかいうようなことを言いたかったのですが、伝わったかどうかはかなり疑問です。ああ、もっと英語が話せたらなあ。でも、まあ「度胸で話せばなんとかなる」という妙な自信を強めたことも確かです。

午後のプレゼンテーション「Object-Oriented Scripting in Ruby」は、Rubyの思想や背景を踏まえて、スクリプティングとはなにかという午前中のパネルの続きのような内容を紹介しました。つたない説明でしたが、ある程度興味をもってもらえたようです。このプレゼンテーションの資料は、

```
http://www.ruby-lang.org/en/jaoo2001/
```

で公開しています。実はデンマーク人は（日本人のように？）とても礼儀正しくおとなしいので、どのセッションでもほとんど質問が出なかったのですが（質問するのはアメリカ人ばかり……）、なぜか私のプレゼンテーションに対しては、ある程度活発な質問が出ました。内容が良かったせいか、私の英語があまりに下手だったせいかはわかりませんが、後者でなかったらよいなあと思っています。

プレゼンテーションが終わると、みんながざわざわしていました。ちょうどその頃例の同時多発テロのニュースが報道されたからです。JAOOの講師はほとんどがアメリカ人なのですが、彼らは自分の国が攻撃されたショックと、飛行機の飛行が禁止されて帰れるかどうかわからない不安からかなり動揺していました。その後の連絡でPragmatic Programmersはうちに帰り着くのが3日くらい予定より遅れたのだそうです。大変でした。

私にとっての今回のJAOOでの収穫といえば、たくさんの人に直接会うことができた点でしょう。たとえば、Dave ThomasとAndy Huntに初めて会いました。彼らとはすでに数えきれないくらいメールやIRCでやりとりをしていたのですが、直接会ったのは初めてでした。

Martin Fowler、Ward Cunningham、Doug Leaなどと直接言葉を交わせたのはかなり感激ものでした。Kent Beckは今年は不参加だそうで会えませんでした、残念。

ほかにも一緒にパネルに参加した人たちとの会話も刺激的でした。とくにBrent Welchは波長が合って、RubyとTclお互いに参考にできるところはないかとかいろいろと話をしました。正直Tclを見直しました。それから、オブジェクト指向言語の父Simulaを開発したというOslo大のKristen Nygaard教授にも会ったのも貴重な経験でした。彼にとってはすべてのオブジェクト指向言語は、いわば孫か曽孫のようなものだそうです。クラスとか継承とかの概念はみんなSimulaが発明したのですから、確かにそうかもしれません。

私は、飛行機のチケットの関係で木曜日に出発しなければならなかったのですが、向こうにいたのが54時間だけで、移動には58時間かかったというなんだか強行軍でした。なかなか珍しい経験をしたと思います。

次回はLinux Conference Japanをレポートする予定です。あるいはフロリダ州タンパで開かれる第1回Ruby Conferenceについてレポートできるかもしれません。

サーバ百科
基礎から学ぶネットワークの仕組み

これまでの回で基本的なサーバの動作原理を説明してきたが、具体的なサーバの運用方法の解説の前に、UNIX系のサーバの「常識」を知っておこう。

第3回　サーバの起動と設定ファイル

文：安田幸弘
Text : Yukihiro Yasuda

UNIX上で稼動するネットワークサーバには、起動の方法によっていくつかの基本的な種類がある。今回は、それらの説明から始めよう。

■UNIXのサーバの種類

まず、もっとも基本的なサーバは、スタンドアロンと呼ばれる種類のもの。これはサーバが起動されるとメモリに常駐してポートを監視し、外部からの接続を待ち受ける。このタイプのサーバは、外部からの接続に対して迅速に反応することができるので、頻繁に起動されるサーバや高速な反応が必要なサーバは、OSの起動時に実行されるシェルスクリプトであるrcファイルなどからOSの起動とともに実行され、シャットダウンされるまで稼動し続ける。

rcファイルから起動させる方法は、OSやディストリビューションによって若干異なるが、伝統的なBSDスタイルのOSならrc.localと呼ばれる起動ファイルにコマンドラインを記述してやればよい。それに対して、System V系のOSやRed Hat LinuxなどのLinuxディストリビューションでは、/etc/rc.d/の下にランレベルに応じた起動スクリプトが用意され、起動時のランレベルによって必要なサーバが起動される。Linuxの場合、SysVInitを使うディストリビューションでは、control-panelやntsysvなどの設定ツールで、ランレベルごとに各サーバの起動あるいは停止を設定することができるので、ほとんどの場合はこうした管理ツールを使えばいい。なお、具体的なランレベルによる起動／停止についてはあとで説明することにしよう。

OSの起動時からシャットダウンまで動き続けるスタンドアロンのサーバに対して、inetdと呼ばれるスーパーサーバを使って起動するタイプのサーバがある。inetd（最近はセキュリティを強化したxinetdが使われることも多い）は、サーバを起動するためのサーバで、inetd自身はrcファイルから起動されて、ポートへの接続を監視する。そしてどれかのポートに接続があれば、そのポートに対応するサーバを起動するというものだ。もちろん、inetdから起動されたサーバも、起動後はスタンドアロンと同様に外部からの接続に対して適切な反応を返す。

あまり頻繁に使用されないサーバがいくつも起動していると、それだけで大量のリソースを消費してしまうが、inetdのようなスーパーサーバを使えば、常時起動しているサーバはスーパーサーバだけなので、余計なリソースを消費せずに済むのがメリットだ。たとえば、telnetやFTPなどは、最初の接続までは多少の時間がかかっても、その後の通信は起動したサーバによって管理されるので反応の遅れはほとんど問題にならない。そのため、telnetdやftpd、pop3dのように、それほど高速な反応が要求されないサーバは、inetdから起動されることが多い。

もちろん、telnetdやftpdをスタンドアロンで起動することもできるが、その場合、telnetdなら-debugオプションと接続を受け付けるポートを指定し、ftpdの場合は、-Dオプションで端末を切り離して起動するなど、起動の方法

が異なる。

ただし、スタンドアロンから起動するか、スーパーサーバから起動するかは、サーバのプログラムによって決まるため、必ずしもすべてのサーバがスタンドアロンでもスーパーサーバでも起動できるとは限らないので、起動方法を変更する場合はサーバのマニュアルを参照して適切なオプションを指定しなければならない。

■スーパーサーバ　inetdとxinetd

rc.dから起動されるスタンドアロンのサーバについては、単に起動のためのシェルスクリプトをインストールするだけなので、特に説明の必要はないと思う。ここでは、スーパーサーバのinetdからの起動の書式について簡単に説明しておこう。

inetdからの起動の設定は、/etc/inetd.confにテキストファイルの形式で次のような書式で記述されている。

```
service type prot wait user prog arg
```

それぞれの項目内容は次の通り。

service: サービス名
type:　ソケットタイプ（stream、dgramなど）
prot:　プロトコル（tcp、udpなど）
wait:　waitの指定（waitまたはnowait）
user:　起動ユーザー
prog:　サーバプログラム名
arg:　サーバ引数

たとえば、FTPサーバなど、ほとんどのTCPサーバは次のような書式で/etc/inetd.confに指定されている。

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
```

この行は、まず行頭に「ftp」というサービス名が指定されているが、このサービス名は/etc/servicesの中で定義されているものを使う。/etc/servicesを見ると、「ftp 21/tcp」というような行があるので、確認してほしい。

次に、ソケットのタイプが「stream」、プロトコルはTCPで、サーバの終了を待たずに（nowait）、rootの権限で、/usr/sbin/tcpdを起動、in.ftpdを-lと-aオプション付きで起動している。

ちなみに、サーバプログラム名の/usr/sbin/tcpdは、TCP wrappersと呼ばれるセキュリティプログラムの名前で、TCP wrappersを使わなければサーバプログラム名を/usr/sbin/in.ftpdとしてもかまわない。TCP wrappersを介してサーバを起動する場合は、/etc/hosts.allow、/etc/hosts.denyでサーバへのアクセスの許可や拒否の条件を指定することができる。

inetd.confでは、起動するサーバの種類によって細かくソケットタイプやプロトコルの指定ができるが、ほとんどの場合、インストール時に書き込まれるinetd.confの内容を参考に、適当に書き換えればいい。注意すべき部分としては、ソケットのタイプとプロトコル、wait／nowaitの指定と起動ユーザーだが、前の回で説明したように、ストリーム型のサーバならソケットのタイプはstream、データグラム型のサーバならdgramを指定する。ほとんどの場合、ストリーム型のサーバならプロトコルはtcpで、データグラム型はudpになる。

wait／nowaitの指定は、起動したサーバが重ねて接続を受け付けることができるかどうかによって異なる。ほとんどの場合はnowaitを指定するが、簡単なサーバではwaitを指定しなければならないことがある。これはサーバのプログラミングによるのだが、サーバが起動したあとに、新しくポートを開き直すことができるかどうかによる。つまり、新しくポートを開き直さないようなサーバではwaitを指定し、開き直すサーバではnowaitを指定する。どちらを指定するかはサーバ次第だが、インターネットで公開するようなサーバの場合は、ほとんどの場合、nowaitを指定できるはずだ。

次に、起動ユーザーだが、ここでは起動したサーバの実行パーミッションを指定する。ここにはrootを指定するようになっていることが多いのだが、rootでの起動はセキュリティ的な問題が発生する可能性がある。そのため、サーバの実行に支障のない範囲で、できるだけ権限の小さいパーミッションを指定することが望ましい。

ところで最近では、inetdの代わりにxinetdと呼ばれるスーパーサーバも使われるようになっている。

xinetdは、やはりinetdと同様にrc.dから呼び出されて常駐するスタンドアロンのサーバで、inetdと同じ目的で使われる。inetdと異なるのは、セキュリティ面を中心に機能が大幅に強化されていることだ。xinetdにはTCP wrappersの機能が含まれ、また起動するサーバ数の上限や、各種のリソースの上限を指定できるなど、DoS攻撃やアクセスの集中による問題を回避する設定ができるようになっている。

さて、xinetdの動作は、/etc/xinetd.confで設定するが、Red Hat Linuxなどでインストールされるxinetdでは、xinetdそのものの動作を/etc/xinetd.confで設定し、それぞれのサーバの起動方法については、/etc/xinetd.d/以下のファイル群で指定するようになっている。

/etc/xinetd.confは、全体の設定のためのファイルで、インストール直後は「defaults」というラベルで**リスト1**のような内容が設定されている。

xinetd.confのデフォルトの設定は、特に変更する必要はないと思うが、もちろん必要に応じて書き換えてもかまわない。

さて、xinetdでサーバを起動するために必要な設定は、xinetd.confのincludedirで指定されたディレクトリから読み込まれるようになっている。もちろん、起動するサーバが少ない場合は、xinetd.confに直接書き込んでしまってもかまわない。

さて、xinetd.dに個々のサーバの起動設定を書き込む場合は、ちょうどスタンドアロンのサーバをrc.d以下のファイルで起動するように、/etc/xinetd.d/以下にそれぞれのサーバの名前（サービス名）に対応するファイルを作って、その中で設定することになる。基本的には、次のような書式で起動の条件を指定すればよい。

```
service サービス名
{
        disable         = yesまたはno
        socket_type     = ソケットタイプ
        wait            = yesまたはno
        user            = 起動ユーザー
        server          = サーバ名
        server_args     = 引数
}
```

ここで設定すべき各項目は、inetd.confでの設定とほとんど同じだが、xinetdは、TCP wrappersの機能を含むセキュリティ上の機能が追加されていて、上記以外にもさまざまな機能を設定できる。

たとえば、アクセスを拒否したいホストやネットワークは「no_access =」の行にIPアドレスの範囲やホスト名を記述することができる。また、特定のホストからのアクセスだけを許可したい場合は、「only_from =」で許可するホストやネットワークを指定できる。

このほかにも、xinetdではいろいろな指定が可能だが、とりあえず、これだけの知識があれば、サーバを起動するための設定は可能だろう。

なお、inetd.conf、またはxinetd.confを書き換えた場合は、その内容をスーパーサーバに知らせるために、次のコマンドを実行して設定を読み直させること。

・inetdの場合

```
kill -HUP `cat /var/run/inetd.pid`
```

・xinetdの場合

```
kill -USR1 `cat /var/run/xinetd.pid`
```

■TCP wrappers

xinetdにはクライアントからのアクセスを制御する仕組みがあるが、従来のinetdではTCP wrappers（tcpd）と呼ばれるプログラムを組み合わせてアクセス制御を行うことになる。

TCP wrappersは、それぞれのサーバへの接続ごとに、接続元のIPアドレスやドメイン名を調べて、アクセスの可否や必要なプログラムの起動を行うソフトウェアである。外部からサービスの要求があったとき、TCP wrappersはソースルーティングを検出してクライアントのIPアドレス詐称をチェックし、DNSを逆引きしてドメイン名が正しいことをチェック、そして/etc/hosts.allowと/etc/hosts.denyの設定内容に従ってアクセスの許可、もしくは拒否を行う。また、単にアクセスの可否を決めるだけでなく、クライアントのアドレスによって、適切なプログラムを起動することもできる。

TCP wrappersは、主にinetdから起動されるサーバのアクセス制御に使われるが、パッケージに含まれるアクセス制御ライブラリのlibwrapをリンクすることでスタンドアロンのサーバでもTCP wrappersの機能を使うことができる。

▼リスト1　/etc/xinetd.confの内容

```
defaults
{
  instances = 60                ← 同時に起動できるサーバの数
  log_type  = SYSLOG authpriv   ← ログファシリティ
  log_on_success = HOST PID     ← 起動成功時のログ記録内容
  log_on_failure = HOST         ← 起動失敗時のログ記録内容
}
includedir /etc/xinetd.d        ← サーバごとの設定ファイルのディレクトリ
```

アクセスの制御は、/etc/hosts.allowに許可の条件、/etc/hosts.denyに拒否の条件を次のような書式で列挙することによって設定する。

サーバ名： クライアントのアドレス

または、

サーバ名： クライアントのアドレス： 起動するコマンド

たとえば、特定のホスト以外はFTPサーバを利用できないようにする場合は、次のように設定する。

・hosts.denyの設定

```
in.ftpd: ALL
```

・hosts.allowの設定

```
in.ftpd: 192.168.0., 127.0.0.1
```

この設定は、hosts.denyでin.ftpd（FTPサーバのプログラム名）にALLを指定することで、まずすべてのアドレスからFTPの利用を禁止し、hosts.allowでin.ftpdに対して192.168.0.と127.0.0.1を列挙することで192.168.0.0から192.168.0.255までのアドレスの範囲と127.0.0.1（ローカルホスト）からFTPへのアクセスを許可している。

実際には、hosts.denyには「ALL： ALL」という1行だけを書き込んで、デフォルトですべてのホストからすべてのアクセスを禁止し、接続を許可するホストだけをhosts.allowで指定するほうが便利だ。具体的には、たとえば次のように設定すると、ローカルネットワーク（192.168.0.0～192.168.0.255）とローカルホスト（127.0.0.1）からのアクセスが許可され、in.ftpdには.example.co.jpのすべてのホスト、in.telnetdにはhome.example.co.jpからのアクセスが許可される。

・hosts.denyの設定

```
ALL: ALL
```

・hosts.allowの設定

```
in.ftpd: .example.co.jp, 192.168.0., 127.0.0.1
in.telnetd: home.example.co.jp, 192.168.0., 127.0.0.1
ALL: 192.168.0., 127.0.0.1
```

なお、TCP wrappersの設定内容をチェックするために、tcpdchkというコマンドが用意されている。設定後にtcpdchkを実行して、設定に問題がないことを確認しておくとよい。また、tcpdmatchというコマンドで正しい設定ができているかどうかをチェックすることができる。たとえば、「tcpdmatch in.ftpd example.com」というコマンドは、example.comからFTPへの接続ができるかどうかをチェックし、接続が許可される場合は「access： granted」、接続が許可されない場合は「access： denied」が表示される。

TCP wrappersでは、接続が拒否された場合に任意のスクリプトやプログラムを実行することができる。

たとえば、hosts.denyに単に「ALL： ALL」を記述するだけでなく、次のような行を書き込むと、接続を許可されていないホストからアクセスされたときにそのホストに対してsafe_fingerを実行し、その結果をroot宛にメールで送信する。なお、hosts.allow、hosts.denyでは、1行が長くなった場合に「＼」をつけて改行することにより、1つの設定を複数行にわたって記述することができる。

・hosts.denyの設定

```
ALL: ALL: (/usr/sbin/safe_finger -l @%h \
| /bin/mail -s Illegal-access-%d-%h  root ) &
```

またスクリプトを起動するときに、twistのオプションを付けると、起動したtcpdのプロセスが指定されたスクリプトで置き換えられる。このとき、標準入力、標準出力、および標準エラーも置き換えられるので、接続が拒否された場合にクライアントに警告メッセージを送るなどの用途に利用することができる。たとえば、hosts.denyに次のような行を書き加えることで、接続が拒否されたクライアント側に任意のメッセージを送ることができる。

```
in.ftpd : 拒否ホスト : twist /bin/echo メッセージ
```

メッセージの内容は、たとえば「421 Connection not allowed」などとしておけばいい。なお、twistを使用する場合は、TCPでの接続に限る。UDPのプロトコルではtwistは正しく動作しない。

■起動スクリプトと設定ファイルについて

サーバといっても、基本的には端末を必要としないというだけで、普通のUNIXプログラムと大きな違いはない。

したがって、起動時に起動スクリプトでコマンド名を書けばサーバが起動する。しかし、いうまでもなく、起動したサーバにユーザーが期待する動作をさせるためには、起動の条件をはじめ、さまざまな動作の指定が必要になる。

Windowsを中心に利用してきたユーザーは、しばしばUNIXやLinuxの設定が難しいと言うが、ちょっと経験を積んだWindowsユーザーなら、config.sysやautoexec.bat、そしてさまざまなINIファイルを編集したことがあるはずだ。UNIXの起動スクリプトや設定ファイルも、WindowsのバッチファイルやINIファイルと基本的な発想は同じである。ただ、比較的書式が一定しているWindowsのINIファイルと違って、UNIXの場合、設定ファイルの書式がサーバによって異なっていること、起動スクリプトで指定できるオプション引数の種類もWindowsより多く、やや複雑だというだけである。

以下、サーバの起動と終了、そして設定ファイルの書き方など、サーバを管理するうえで知っておきたい基本的な常識について順に説明していくことにしよう。

■起動スクリプト

起動スクリプトは、BSDスタイルのrcであればrc.localに直接コマンドラインを書き込み、System Vスタイルであれば/etc/rc.d/以下にインストールされる。

RPMパッケージなどのバイナリファイルをインストールする場合、起動スクリプトも自動的に適当なディレクトリにインストールされ、ほとんどの場合、起動スクリプトの内容をユーザーが変更する必要はないが、簡単にSystem Vスタイルの起動スクリプトの動作を説明しておこう。なお、ここではRed Hat Linuxでのディレクトリツリーを前提に説明するが、ほかのディストリビューションでも基本的な動作は同じである。

さて、起動スクリプトは、/etc/rc.d/以下のディレクトリにインストールされる。起動スクリプトの本体は、/etc/rc.d/init.d/以下にインストールされ、「start」、「stop」、「restart」などの引数によってそれぞれ起動、停止、そして再起動をすることができるようになっている。

init.dに保存されているスクリプトは、一見複雑に見えるが、基本的な構成は次のようなものだ。

```
# サブルーチンの読み込み
. /etc/rc.d/init.d/functions

# 実行ファイルのチェック
[ -x プログラムへのパス ] || exit 0

# 起動
start () {
    daemon プログラム名
    return $?
}

# 停止
stop () {
    killproc プログラム名
    return $?
}

# 再起動
restart () {
    stop
    start
    return $?
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart}"
        RETVAL=1
esac

exit $RETVAL
```

この起動スクリプトの中で使われているdaemonとkillprocという関数は、/etc/rc.d/init.d/functionsの中で定義されている関数で、サーバの起動と停止を実行するための汎用の関数である。

このように、起動と停止の関数を使う理由は、サーバを

起動したり停止させる方法は、サーバのプログラムによって微妙に異なっているためだ。たとえば、あるサーバは、単にコマンドを起動しただけではバックグラウンドで動作せず、端末からデタッチするためにコマンド名の最後に「&」を付けなければならないことがある。また、停止させる場合は、サーバのプロセスIDを指定して、killコマンドなどで停止シグナルを送るのだが、指定するプロセスIDの取得の方法が必ずしも同じではない。そのため、Red Hat Linuxなどでは汎用の関数で起動と停止ができるようになっているが、もちろん、直接コマンドラインを指定して起動し、適切な方法で停止させてもかまわない。

ちなみに、サーバを停止させるには、/var/run/以下に保存されているプロセスIDのファイルを参照して、次のようなコマンドを実行するのが一般的だ。

```
kill -TERM `cat /var/run/プログラム名`
```

同様に、サーバの再起動、または設定ファイルを変更したあとのファイルの再読み込みは、次のようにHUPシグナルを送る。

```
kill -HUP `cat /var/run/プログラム名`
```

/var/run/以下にプロセスIDのファイルが作成されないサーバでは、次のようにpsコマンドを使ってサーバのプロセスIDを探し、killコマンドを実行することになる。

```
ps wax|grep プログラム名
kill -TERM プロセスID
```

なお、TERMシグナルを送っても、プログラムが何らかのエラーで終了できない場合は、「kill -KILL プロセスID」で強制終了させる。

さて、System V系のシステムでは、ランレベルによって、マシンの状態を変更できる。ランレベル0はシステム停止、ランレベル1はシングルユーザーモード、そしてランレベル6はシステムの再起動を意味しており、そのほかのランレベルは/etc/inittabで動作の内容が定義されている。ランレベル2から5までの意味はシステムやディストリビューションによって異なるが、Red Hat Linuxの場合、ランレベル2はNFSを起動しないマルチユーザーモード、ランレベル3は通常のマルチユーザーモード、ランレベル4は未使用で、ランレベル5がX Window Systemを使ったグラフィカルログインに割り当てられている。

これらのランレベルに応じてサーバを起動、停止するために、ランレベル0から6までに対応して/etc/rc.d/rc0.dからrc6.dまでのディレクトリが用意され、たとえばランレベルが3なら、/etc/rc.d/rc3.d/以下のスクリプトを読み出して、必要なサーバの起動や停止を行うわけだ。

具体的には、それぞれのランレベルごとのディレクトリには、init.dに収められた起動スクリプトから、スタート（S）または停止（K）、2桁の優先順位、コマンド名という形でシンボリックリンクが張られている。たとえば、“testserver”というサーバを新しくインストールした場合は、まず“testserver”という名前の起動スクリプトが/etc/rc.d/init.dにインストールされる。そして、それぞれのランレベルのディレクトリに適当なシンボリックリンクを張るのだが、このサーバをランレベル3で起動させたい場合は、「ln -s /etc/rc.d/init.d/testserver /etc/rc.d/rc3.d/S00testserver」のコマンドで/etc/rc.d/rc3.dに「S00testserver」というシンボリックリンクを張ればいい。また、たとえばシングルユーザーモードではこのサーバを停止させたいのであれば、「ln -s /etc/rc.d/init.d/testserver /etc/rc.d/rc1.d/K00testserver」で/etc/rc.d/rc1.dに「K00testserver」というリンクを作ればよい。

なお、ここで指定する優先順位は、慣例として00から99までの2桁の数値で、値が小さいものから先に処理される。つまり、先に起動させたいサーバは小さい数値を設定し、あとで起動させたいサーバは大きな数値を設定する。たとえばNFSでマウントしたボリュームの情報が必要になるようなサーバは、NFSよりも大きな数値を指定すればいいというわけだ。

■設定ファイルのいろいろ

次に、設定ファイルだが、UNIXのサーバの多くは/etc/以下のディレクトリにサーバの起動時に読み込まれるファイルを保存し、起動時に参照するようになっていることが多いが、これらのファイルにはいくつかのパターンがある。

/etc/ディレクトリ以下を見ればわかるように、/etc/にはさまざまな種類のファイルが収められており、サーバの動作を設定するためのファイルの多くは、拡張子が「.conf」、または「.cfg」、「.config」などになっている。ただし、UNIXの場合、拡張子は単なる符丁にすぎず、必ずしも「.conf」だけが設定用のファイルとは限らないし、拡張子に特別な意味もないのでそのつもりで。

これらのファイルを順に開いて見てみよう。それぞれ、内容や書式は異なっているが、じっくりと見ていると、設定ファイルの書式や書き方に一定のパターンがあることがわかるはずだ。必ずしもすべてのサーバに共通するというものではないが、ここでごく一般的なサーバの設定ファイルでよく使われる書式について説明しよう。

まず、ほとんどの設定ファイルに共通するのがコメント行の書き方だ。つまり、「#」で始まる行は、それ以後の文字列がコメントとみなされる。たとえば、設定のメモや一時的に無効にしたい設定の行の前に「#」を付けることで、その行の設定は無視されるようになる。

ただし、次のような行で「#」以後をコメントとみなすかどうかはプログラムによって異なる。

```
foo = bar   # fooをbarにする
```

プログラムによっては、「#」以後を読み飛ばすが、プログラムによっては「foo」の値に「bar # fooをbarにする」を代入しようとして失敗することがあるので注意が必要だ。特に、変数に空白を含む文字列を代入することができる場合、つまり「foo = bar and baz」といった書式が許される場合、上記のようなコメントは許されないことが多い。それに対して「foo = "bar and baz"」のように文字列は「"」で囲むような書式では、上記のようなコメントが許されることもある。

また、プログラムによっては、たとえばnamedのようにコメントを示す記号が「;」のように「#」以外の記号になっていることがある。

ところで、ほとんどの設定ファイルは、1つの設定を1行で書くようになっているが、1行が長くなるような場合、次のように行末に「\」記号を入れることで、複数の行にわたって設定内容を記述できることがある。

```
foo: /usr/local/share/foo/bar/baz \
hogehoge, fugafuga ....
```

次に空白の扱いだが、設定ファイルに書き込まれるキーワードとキーワードは、通常空白、またはタブなどで区切られるが、ほとんどの場合、空白の個数やタブの個数は任意である。つまり、「a＝b」も「a　＝　b」も「a　　＝　　b」も同じように取り扱われる。ただし、いうまでもなく日本語の全角空白文字は空白とはみなされないので、編集時には日本語の空白文字を入れないようにしたい。

さて、次に設定ファイルそのものの構文について説明しよう。設定ファイルの構文や書き方は、サーバごとに大きく異なっているが、細かい違いを無視すれば、おおざっぱに次の3つのパターンに分類することができる。

●パターンA：「変数＝値」のパターン

このパターンは、「A＝B」の形で設定内容を列挙していくもので、たとえばlilo.confなどで使われている設定ファイルの書式である。たとえば、lilo.confでは、「default＝linux」という行で起動するカーネルのラベルが「linux」であることを指定している。

・パターンAの例：rsyncd.conf

```
motd file=/etc/motd
pid file = /var/run/rsync.pid
max connections= 2

[misc]
path=/usr/local/public
comment=Public Directory
read only=yes
```

●パターンB：「キーワード　値」のパターン

このパターンは、ある動作を示すキーワードの値を指示することでサーバの動作を指定する。この種の設定ファイルを使うプログラムは多く、たとえばinetd.confをはじめ、resolv.confやsyslog.confなどがこのパターンの設定ファイルを使っている。また、inetd.confのように、キーワードに対する値は、複数の値が列挙される場合も多い。

なお、このパターンと変形には、「キーワード：値」のように「：」などでフィールドを区切るような書式を持つ設定ファイルもある。

・パターンBの例：sshd_conf

```
Port 22
Protocol 2,1
PermitRootLogin without-password
IgnoreRhosts yes
StrictModes yes
PrintMotd no
SyslogFacility AUTH
LogLevel INFO
RhostsAuthentication no
```

```
RhostsRSAAuthentication no
RSAAuthentication yes
DSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
CheckMail no
UseLogin no
```

●パターンC：ブロック構造のパターン

このパターンの設定ファイルは、たとえばnamed.confのように「ラベル { 変数＝値； 変数＝値}」の形で、ラベルに続く「{」と「}」で囲まれたブロックの構文を持つ。また、smb.confなどでは、「[ラベル] 変数＝値；変数＝値」のようにラベルを「[」と「]」で囲み、次のラベルまでをブロックとする構文を、あるいはhttpd.confなどHTMLのように「<ラベル>」と「</ラベル>」をブロックとする構文を持つファイルもこのパターンだ。

この種の設定ファイルは、パターンBの書式の拡張と考えることもできるが、パターンBでは指定する値の順番や数が固定されているのに対して、値の数や順番は自由に記述できることが多い。

・パターンCの例1： named.conf

```
options {
    directory "/etc/namedb";
    allow-transfer {
            192.168.10.1;
            127.0.0.1;
    };
};

zone "example.co.jp" in {
    type master;
    file "example.co.jp.zone";
};

zone "0.168.192.in-addr.arpa" in {
    type master;
    file "0.168.192.rev";
};

zone "." in {
    type hint;
    file "named.root";
};

zone "0.0.127.IN-ADDR.ARPA" in {
    type master;
    file "localhost.rev";
};
```

・パターンCの例2： proftpd.conf

```
ServerName              "Linux mag FTP server"
ServerType              standalone
DefaultServer           on
Port                    21
Umask                   022
TimesGMT                FALSE
MaxInstances            30
User                    nobody
Group                   nobody

<Directory /*>
  AllowOverwrite        on
</Directory>

<Anonymous ~ftp>
  User                  ftp
  Group                 ftp
  UserAlias             anonymous ftp
  MaxClients            10
  RequireValidShell     no
  DisplayLogin          welcome.msg
  DisplayFirstChdir     .message
  <Limit WRITE>
    DenyAll
  </Limit>
</Anonymous>
```

実際には、これらのパターンが組み合わされることも少なくない。また、このほかにもLISPのような書式で設定ファイルを記述するものや、独自の複雑な書式を持つファイルを使うプログラムもある。このあたりが、サーバの個性というのか、UNIXの難しさといわれる部分なのかもしれないが、だいたい上記のパターンさえマスターしておけば、たいていのサーバの設定はできるはずだ。

Oracle8iで作るWebサイト入門

Oracle8iなどのRDBを使うときにどうしても難しく感じるのは、ユーザー（スキーマ）や各種権限、データベースやテーブルの管理が複雑で多岐にわたるところではないでしょうか。もちろんその気になればかなり複雑な管理もできるのですが、まずは第一歩としての基礎知識を勉強してみましょう。

第5回　Oracle8iの管理

文：おもてじゅんいち／かざぐるま
Text : Junichi Omote/Kazaguruma

書籍検索サイト

本連載では、**画面1**のような書籍情報を検索できるサイトを構築します。ここで表示されている項目と表紙の画像をデータベースで管理することになるのですが、基本的には、データの読み出し（SELECT文）、登録（INSERT、UPDATE文）、削除（DELETE文）ができれば実現できますので、データ項目（列）は増えますが、前回使ってみたSCOTTユーザーのEMPテーブルと同じようなテーブルが使えればよいことになります。

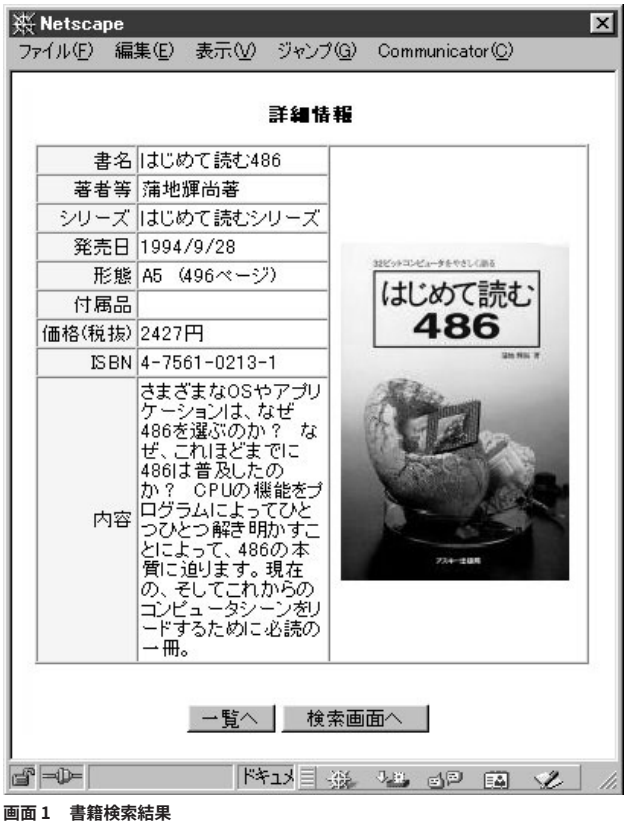
SCOTTユーザーをそのまま使ってもかまわないのですが、せっかくですから今回は専用のユーザーを新たに作成するところから始めてみます。

書籍検索サイトのテーブルを使えるようにするまでの手順は**図1**のようになります。ユーザーを作成したり、テーブルを作成したりするために、Oracle8iの製品版ではEnterprise Managerというツールがありますが、まず最初はデータベース管理とはどういうものなのかを理解してもらうために、コマンドラインで進めてみます。Oracleでも少し前のバージョンまではGUIツールがありませんでしたし、他のRDBに移行する場合にも、ここでの経験と理解が役に立つでしょう。

管理ツールには、おなじみのSQL＊Plusを使います。データベースの管理は基本的にすべてSQL文になります。

■ユーザーの作成

図1の手順にしたがって、まずは新しいユーザー（スキーマ）を作成します。作成時にパスワードも決めておきます。ここでは、



画面1　書籍検索結果

ユーザー名：ascii
パスワード：linuxmag

とします。ユーザーを作成する前に、現在どんなユーザーが存在するかを見てみます。

ユーザー一覧を見たいときは、**画面2**のようにSQL＊Plusを管理ユーザー（デフォルトではSYSかSYSTEM）で起動し、all_usersというビュー（テーブルのようなもの）をSELECT文で表示します。SYSTEMのデフォルトパスワードはmanagerです。

SELECT文の結果は15行になりました。つまりorclデータベースには現在15名のユーザーが登録されていることになります。では新しいユーザーを作成してみましょう。

ユーザーを作成できるのは管理ユーザーですから、同様にSYSTEMユーザーでSQL＊Plusを起動して、**画面3**のように実行してください。書式は、

```
create user ユーザー名 identified by パスワード
```

です。作成されたというメッセージが表示されればOKです。これでユーザーすなわち新しいスキーマが作成できましたが、このユーザーはまだ何の権限ももっていないので、データベースに接続することさえできません。試しにSQL＊Plusをいったん QUIT して、新しいユーザーで起動してみてください（**画面4**）。

CREATE SESSIONという権限がないためにログオンができないというエラーメッセージが表示されます。CREATE SESSIONとはまさにデータベースとの接続（SESSION）を作ることができなかったということですから、これでは何もできません。そこで**図1**の手順に戻って、新ユーザーasciiのシステム権限を設定します。

図1　テーブル作成までの手順

システム権限というのは、データベースと接続したり、テーブルを作成したり変更したりできる権限で、Oracle8iには80近い種類の権限が用意されています。すべてを扱う必要はないのですが、最低限、テーブルを作成したり、その内容を変更したりする権限は必要です。いくつかの例を**表1**に挙げてみました。しかし、80もの権限のどれが必要かを考えるのは大変なので、前回のユーザーSCOTTを参考にして、SCOTTと同じ権限を与えてみましょう。

あるユーザーの権限を調べるためには、そのユーザーでデータベースに接続する必要があります。接続したら、session_privsというビューをSELECT文で表示します（**画面5**）。

SCOTTのシステム権限が表示されました。ちょっとわ

```
$ sqlplus system/manager@orcl
SQL> select * from all_users;

USERNAME                  USER_ID  CREATED
------------------------- ------- --------
SYS                             0 00-04-17
SYSTEM                          5 00-04-17
OUTLN                          11 00-04-17
DBSNMP                         16 00-04-17
TRACESVR                       19 00-04-17
AURORA$ORB$UNAUTHENTICATED     25 00-04-17
ORDSYS                         26 00-04-17
ORDPLUGINS                     27 00-04-17
MDSYS                          28 00-04-17
CTXSYS                         31 00-04-17
SCOTT                          33 00-04-17
ADAMS                          34 00-04-17
JONES                          35 00-04-17
CLARK                          36 00-04-17
BLAKE                          37 00-04-17

15行が選択されました。
```

画面2　ユーザー一覧

```
$ sqlplus system/manager@orcl
SQL> create user ascii identified by linuxmag;

ユーザーが作成されました。
```

画面3　ユーザー（スキーマ）の作成

```
$ sqlplus ascii/linuxmag@orcl

SQL*Plus: Release 8.1.6.0.0 ……
(c) Copyright 1999 Oracle ……

ERROR:
ORA-01045: user ASCII lacks CREATE SESSION
privilege; logon denied
```

画面4　接続エラー

からない用語もあるかもしれませんが、何となく理解できるでしょうか。CREATE SESSIONは先ほどユーザーasciiにないと言われたものですし、CREATE TABLEは文字どおりテーブルを作成する権限です。

SCOTTの権限と同じでよいので、これらの権限をasciiに設定したいのですが、14個の設定をひとつずつ行うのは面倒です。そこで、Oracle8iにはロールというものが用意されています。ロールとは、いくつかの権限をグループとしてまとめたものです。あるロールを1つ設定するだけで、そのロールに含まれている権限を一度に設定できます。デフォルトで用意されているロールの例を**表2**に挙げます。

ではSCOTTにはどんなロールが設定されているのでしょうか。画面5の最後のSELECT文でロールを調べています。

SCOTTに設定されているのは、CONNECTとRESOURCEという名前のロールです。表2とSCOTTの持つシステム権限（画面5）を見比べてみてください（UNLIMITED TABLESPACEについては後述）。ということで、asciiにもCONNECTとRESOURCEロールを設定すればよいことがわかりました。

せっかくですから、一気に2つのロールを設定するのではなく、ひとつずつロールを設定していって、そのときの権限の状態を見てみましょう。

```
$ sqlplus scott/tiger@orcl
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
UNLIMITED TABLESPACE
CREATE TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE

14行が選択されました。

SQL> select * from session_roles;

ROLE
------------------------------
CONNECT
RESOURCE
```

画面5　scottのシステム権限とロール

まず、CONNECTロールを設定します。ロールの設定は管理ユーザーで行います。

```
$ sqlplus system/manager@orcl
SQL> grant connect to ascii;
```

これはCONNECTロールをユーザーasciiに設定するというもので、GRANT文は個々の権限を設定するときにも、ロールを設定するときにも使えます。

権限が付与されたら、今度はユーザーasciiでSQL＊Plusを起動してください。SCOTTのときと同じように、権限を調べてみます（**画面6**）。

この時点ではまだCONNECTロールに含まれる権限しか付与されていませんから、SCOTTのもつ権限より少なくなっています。では同様にRESOURCEロールも設定します。

```
$ sqlplus system/manager@orcl
SQL> grant resource to ascii;
```

CREATE USER	ユーザーの作成
ALTER USER	ユーザーの変更
DROP USER	ユーザーの削除
CREATE ROLL	ロールの作成
CREATE TABLE	テーブルの作成
ALTER TABLE	テーブルの変更
DROP TABLE	テーブルの削除
ALTER SESSION	現行セッションの変更
ALTER SYSTEM	Oracleインスタンスの変更
CREATE TABLESPACE	表領域の作成
UNLIMITED TABLESPACE	表領域を無制限に使用
SYSDBA	システム管理（データベース作成など）

表1　システム権限の例

CONNECT	CREATE SESSION
	ALTER SESSION
	CREATE TABLE
	CREATE CLUSTER
	CREATE SYNONYM
	CREATE VIEW
	CREATE SEQUENCE
	CREATE DATABASE LINK
RESOURCE	CREATE CLUSTER
	CREATE SEQUENCE
	CREATE TABLE
	CREATE PROCEDURE
	CREATE TRIGGER
	CREATE TYPE
	CREATE OPERATOR
	CREATE INDEXTYPE
DBA	すべてのシステム権限

表2　ロールの例と含まれる権限

設定後のasciiの権限とロールを見てみましょう(**画面7**)。ほぼSCOTTと同じになっていますが,UNLIMITED TABLESPACEという権限が1つ足りません。UNLIMITED TABLESPACEというのは表領域(テーブルなどを作成するために確保された領域)を無制限に使えますよという権限なのですが、この権限がない場合は、別途QUOTAという、領域の使用可能量を設定しなければなりません。UNLIMITED TABLESPACE権限がある場合は無制限ですから、QUOTAの設定は必要ないのです。つまり、UNLIMITED TABLESPACE権限もQUOTA設定もない場合は、そのユーザーはテーブル等をいっさい作れない(作る領域がない)ということになります。

本格的なシステムならば、QUOTA設定を適切に行わないと、各スキーマのディスクの使用量などが問題になってくるのですが、とりあえずデータベースが1つで、稼動するスキーマ(ユーザー)も少ないのなら、UNLIMITED TABLESPACE権限を与えるほうが簡単ですから、ユーザーasciiにもUNLIMITED TABLESPACE権限を与えます。

```
SQL> grant unlimited tablespace to ascii;
```

これで、asciiの持つ権限はSCOTTと同じになりました(**画面8**)。

ところで、画面2のあたりからビューというものが出てきました。all_users、session_privs、session_rolesというのがそれで、あたかもテーブルを読み出すようにSELECT文で内容を表示させていました。これらは、ディクショナリビューというもので、ユーザーやテーブルの各種情報を保持しています。データを格納する通常のテーブルとは異なるものなのですが、これらの管理情報もSQLで扱えるほうが都合が良いので、テーブルと同様にSELECT文で読み出すことができるようになっています。ただし、ビューという名の通り、読み出すだけで書き換えはできません。

表3がおもなディクショナリビューです。SELECT ＊で読み出せば内容が表示されますが、USER_TABLESなどいくつかは列の数が多いので、列名を指定したほうが見やすくなります。たとえば今接続しているユーザーのテーブル一覧を見る場合は、

```
select table_name from user_tables;
```

となります。SCOTTで試してみてください。

■テーブルの作成

ではasciiユーザーでテーブルを作成してみましょう。テーブルの作成時には、そのテーブルに含まれる列名と各列のデータ型をすべて指定しなければならないので、ちょっと長い文になりますが、一度は手で書いてみることをお勧めします。書式は**リスト1**です。1行ですべて書き切ってしまってもかまいませんし、SQL＊Plusでは改行するごとに行番号を返してくれますから、分割して順に入力することができます。最後に「;」を入力すれば完了です。

では、**画面9**のように入力してみましょう。間違えると

```
$ sqlplus ascii/linuxmag@orcl
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
CREATE TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK

8行が選択されました。
```

画面6　CONNECTロールの権限

```
$ sqlplus ascii/linuxmag@orcl
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
CREATE TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE

13行が選択されました。

SQL> select * from session_roles;

ROLE
------------------------------
CONNECT
RESOURCE
```

画面7　RESOUCEロール追加後の権限

エラーが出て打ち直しになります。また、改行は**画面9**のように行ってください。変なところで改行するとエラーになることがあります。

データ型は、

number(n)	数値、()内は桁数
char(n)	固定長文字列、()内は文字数
varchar2(n)	可変長文字列、()内は最大文字数
date	日付/時刻型

の4種類を使っていますが、ほとんどの場合はこの4種類でこと足ります。char（）とvarchar2（）は、実際のデータの文字数（バイト数）が（）内の数よりも少ないときにふるまいが異なります。char（）は残りの部分に空白をつめて、必ず（）内の文字数だけ領域を確保します。varchar2（）は実データの格納に必要な分だけしか領域をとりませんから経済的です。

USER_USERS	接続ユーザーの数
ALL_USERS	全ユーザーの簡易情報
DBA_USERS	全ユーザーの詳細情報
SESSION_PRIVS	接続ユーザーの権限リスト
SESSION_ROLES	接続ユーザーのロールリスト
USER_ROLE_PRIVS	接続ユーザーのロール
USER_SYS_PRIVS	ユーザーのシステム権限
ROLE_SYS_PRIVS	ロールに対するシステム権限
USER_TABLES	接続ユーザーのテーブル一覧

表3　ディクショナリ・ビューの例

```
$ sqlplus ascii/linuxmag@orcl
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
UNLIMITED TABLESPACE
CREATE TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE

14行が選択されました。
```

画面8　UNLIMITED TABLESPACE権限を設定した

CREATE TABLEが無事完了したら、user_tablesでテーブルが追加されていることを確認します（**画面9**）。**画面10**のようにINSERT文やDELETE文でデータを格納してみてテーブルが正しく動いているかをテストすることもできます。

テーブルの情報を表示させるには、**画面11**のように、

```
desc book
```

を実行してください。これだけはSQL文ではなくSQL＊Plusのコマンドです。

■列の変更、追加、削除

テーブルを作成したあとでも、列のデータ型を変更したり、列を追加、削除したいことがあります。その場合はALTER TABLE文を使います。列のデータ型の変更は、

```
alter table テーブル名 modify 列名 データ型
```

という書式です（**画面12**）。列を追加するときは（**画面13**）、

```
alter table テーブル名 add 列名 データ型
```

```
$ sqlplus ascii/linuxmag@orcl
SQL> create table book
  2  (
  3  id number(6),
  4  title varchar2(100),
  5  author varchar2(100),
  6  series varchar2(100),
  7  seldate date,
  8  type varchar2(50),
  9  appendix varchar2(50),
 10  price number(6),
 11  isbn char(13),
 12  contents varchar2(1000),
 13  image varchar(50)
 14  )
 15  ;

表が作成されました。

SQL> select table_name from user_tables;

TABLE_NAME
------------------------------
BOOK
```

画面9　BOOKテーブルの作成

▼リスト1　CREATE TABLEの書式

```
SQL> create table テーブル名 ( 列名 データ型, 列名 データ型, …… );
```

列を削除するときは（**画面14**）、

```
alter table テーブル名 drop (列名)
```

となります。削除のときだけ列名を（）でくくることに気をつけてください。

書籍検索テーブルを作る

テーブルの作成は完了しましたが、このテーブルにはまだデータが入っていません。とりあえずはSQL＊Plusでデータを手入力してみてください。その場合、すべての列のデータを入力するのは面倒ですから、

```
insert into book (id,title,price) values (1,'Oracle
入門',2500);
```

```
SQL> insert into book (id,title,price) values
(1,'Oracle入門',2500);

1行が作成されました。

SQL> select id,title,price from book;

        ID
----------
TITLE
----------------------------------------
     PRICE
----------
         1
Oracle入門
      2500

SQL> delete book where id=1;

1行が削除されました。
```

画面10　BOOKテーブルのテスト

```
SQL> desc book
 名前                      NULL?    型
 ------------------------- -------- ----------
 ID                                 NUMBER(6)
 TITLE                              VARCHAR2(100)
 AUTHOR                             VARCHAR2(100)
 SERIES                             VARCHAR2(100)
 SELDATE                            DATE
 TYPE                               VARCHAR2(50)
 APPENDIX                           VARCHAR2(50)
 PRICE                              NUMBER(6)
 ISBN                               CHAR(13)
 CONTENTS                           VARCHAR2(1000)
 IMAGE                              VARCHAR2(50)
```

画面11　BOOKテーブルの情報

というように列を指定して入力すればよいでしょう。

データが正しく格納されたかを確認するにはSELECT文を使うのですが、SQL＊Plusでは何行にもわたって表示されるため非常に見にくい表示になります。そこで、ブラウザでテーブルの中身を確認できるようにスクリプトを書いてみます（**リスト2**）。

これは、前回使ったリスト表示スクリプトをもとに作ったものです。変更するのは、ログオン時のユーザー名とパスワード、SELECT文のテーブル名と列名（＊にしている）、そして表示したい列と同名の変数との対応、表示部分です。基本的な構造はそのままですから、自由に表示する列を変えてみてください。先ほどSQL＊Plusで入力したデータが表示されればOKです。

ただし、この場合SQL＊Plusでのログオンユーザーと、

```
SQL> alter table book modify image char(50);

表が変更されました。

SQL> desc book
 名前                      NULL?    型
 ------------------------- -------- ----------
 ID                                 NUMBER(6)
  ⋮                                  ⋮
 CONTENTS                           VARCHAR2(1000)
 IMAGE                              CHAR(50)
```

画面12　BOOKテーブルの変更

```
SQL> alter table book add comm varchar(100);

表が変更されました。

SQL> desc book
 名前                      NULL?    型
 ------------------------- -------- ----------
 ID                                 NUMBER(6)
  ⋮                                  ⋮
 CONTENTS                           VARCHAR2(1000)
 IMAGE                              VARCHAR2(50)
 COMM                               VARCHAR2(100)
```

画面13　列の追加

```
SQL> alter table book drop (comm);

表が変更されました。

SQL> desc book
 名前                      NULL?    型
 ------------------------- -------- ----------
 ID                                 NUMBER(6)
  ⋮                                  ⋮
 CONTENTS                           VARCHAR2(1000)
 IMAGE                              VARCHAR2(50)
```

画面14　列の削除

このスクリプトのユーザーが同じなので、必ずSQL＊Plusのほうの接続をQUITしてからスクリプトを実行しないと、データベースが正しく更新されません。

■大量のデータを流し込む

このテーブルが検索サイトとして動くためには、あらかじめデータが十分格納されていなければなりません。SQL＊Plusで入力していくのはいくらなんでも大変ですし、データ登録用PHPスクリプトを用意して、ブラウザから入力していくのもよいのですが、それでも1件ずつの入力になるため、最初のデータ入力手段としては面倒です。

そこで、スクリプトによるSQLの自動実行という実験も兼ねて、あらかじめ容易してあるテキストデータを一括でテーブルに読み込ませるスクリプトを考えてみます。

テキストデータはCSV（カンマ区切り）ファイルとしてスクリプトと同じディレクトリに置いてあるものとして、そのファイルを読み出して、順次テーブルに格納していくスクリプトです。コマンドラインで起動するスクリプトでもよかったのですが、これも前回のINSERT文を使ったスクリプトを修正して作るので、ブラウザから起動するものとしました。

読ませるCSVファイルは**リスト3**のようなものとします。内容はなんでもよいのですが、データの順番は、テーブルの列に合わせておいてください。books.txtというファイル名で保存しておきます。

リスト4がそのスクリプトです。まずCSVファイルをオープンして、1行ずつ$lineという変数に読み込みます。

```
$line = fgets($fd,2000);
```

は、ファイルから1行読み込むという関数です。2000は最大の読み込みバイト数を指定しています。改行文字があればそこまでしか読み込まないので、大きめの数字を指定しておきます。

```
$line = i18n_convert($line,"EUC-JP","SJIS");
```

は、CSVファイルがシフトJISコードだったので、EUCに変換しています。CSVファイルがEUCで用意されていればこの行は必要ありません。

```
$dat = split(",",$line);
```

では、$lineの中身をカンマで区切って分割し、それぞれを$datという配列変数に格納しています。つまりここで列

▼リスト2　確認用book_list.php3

```
<?php
    $conn = OCIlogon("ascii","linuxmag","orcl");
?>
<html>
<head>
<title>Books List</title>
</head>
<body>
<center>
<b>Books List<b><br>
<table border=1>

<?php
  $stmt = OCIparse($conn,"select * from book
order by id");
  OCIdefinebyname($stmt,"TITLE",&$title);
  OCIdefinebyname($stmt,"AUTHOR",&$author);
  OCIdefinebyname($stmt,"SELDATE",&$seldate);
  OCIdefinebyname($stmt,"PRICE",&$price);
  OCIdefinebyname($stmt,"ISBN",&$isbn);
  OCIexecute($stmt);
  while(OCIFetch($stmt)) {
    print("<tr>");
    print("<td nowrap> $title </td>");
    print("<td nowrap> $author </td>");
    print("<td nowrap> $seldate </td>");
    print("<td nowrap> $price </td>");
    print("<td nowrap> $isbn </td>");
    print("</tr>\n");
  }
?>

</table>
</center>
</body>
</html>
```

▼リスト3　CSVデータの例

```
11,はじめて読む486,蒲地輝尚著,はじめて読むシリーズ,1994/9/28,A5,,2427,4-7561-0213-1,さまざまなOSやアプリケーションは、なぜ486を選ぶのか？　なぜ、これほどまでに486は普及したのか？　CPUの機能をプログラムによってひとつひとつ解き明かすことによって、486の本質に迫ります。現在の、そしてこれからのコンピュータシーンをリードするために必読の一冊。,4-7561-0213-1
12,はじめて読む8086,蒲地輝尚著，村瀬康治監修,はじめて読むシリーズ,1989/5/18,A5,,1602,4-87148-245-6,「はじめて読むマシン語」の8086CPU版解説書です。16ビットパソコンの多くに使用される8086CPUの基礎から、マシン語命令の実習まで、読み進めるうちに自然とプログラミングの知識が身につきます。マシン語プログラミングの入門に最適です。,4-87148-245-6
  ⋮
```

ごとのデータに分割されて$dat[0]、$dat[1]…に順に格納されるのです。その後、配列$datの各データを各々テーブルの列名と同じ名前の変数にコピーしています。もし、CSVファイルのデータの並びがテーブルの列順と異なるなら、ここで移す順番を調整します。そして念のため、列idが空の場合はその行が空データだと判断し、テーブルへの格納は行いません。次の行へ進みます。

次にINSERTのSQL文を作るのですが、列が多くSQL文が長くなるようなので、$sqlという変数にSQL文を順次作っていきます。

```
insert into book values (
```

までは固定なのでそのまま$sqlに入れていますが、よく見ると前回学習したINSERT文とは少し違っていて、valuesの前に（）でくくった列名リストがありません。列名リストを省略したときは、すべての列が対象となります。ですから、valuesのあとの（）にはすべての列に対するデータを並べることになります。順番はテーブルの列順なので間違えないようにしましょう。

もうひとつ気をつけなければならないのは、values （）の値リストでは、文字列型（char、varchar2）と日付時刻型（date）には、「' '」を付けなければならず、数値型（number）には、「' '」を付けてはいけないことです。「'」と「,」が混在していて混乱しそうですが、じっくりと確認してください。ここでは、$idと$priceに「' '」が付きません。ひとつでも間違えるとエラーになるので、注意してください。

何度やってもOracleエラーが出る場合は、

```
//  print("$sql <br>");
```

のコメント（//）をはずしてみましょう。$sqlの中身がブラウザに表示されるので、SQL文が正しいかどうかをチェックできます。

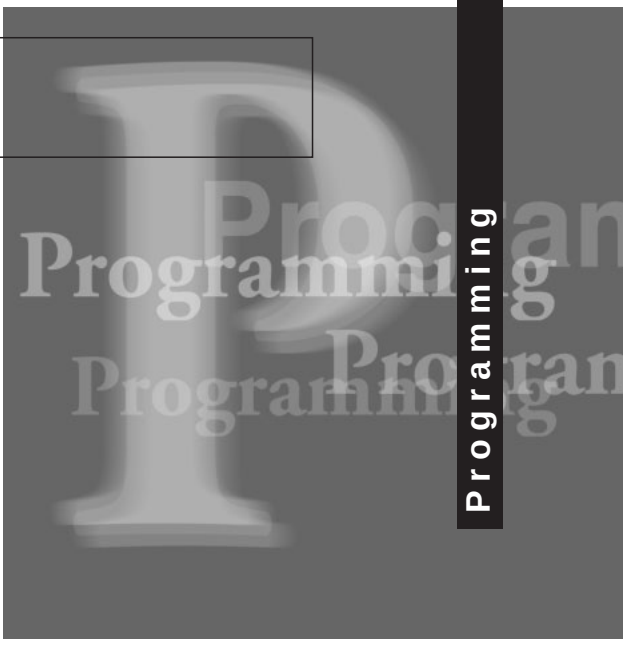
あとは、前回のスクリプトと同じです。ただ今回はINSERT文を実行するたびにOKと表示するので、CSVファイルの行数だけOKが表示されます。実行の確認にもなるのでそのままにしておきました。

このスクリプトをブラウザから起動して、CSVファイルの行数分のOKが表示されれば登録は完了です。再びリスト2の確認スクリプトを実行してみてください。登録されたデータが表示されます。

▼リスト4　CSV読み込みbook_upload.php3

```
<?php
    $conn = OCIlogon("ascii","linuxmag","orcl");
?>
<html>
<head>
<title>Books List</title>
</head>
<body>
<center>
<b>Books List<b><br>
<?php
  $fd = fopen("books.txt","r");
  while(!feof($fd)) {
    $line = fgets($fd,2000);
    $line = i18n_convert($line,"EUC-JP","SJIS");

    $dat = split(",",$line);

    $id       = $dat[0];
    $title    = $dat[1];
    $author   = $dat[2];
    $series   = $dat[3];
    $seldate  = $dat[4];
    $type     = $dat[5];
    $appendix = $dat[6];
    $price    = $dat[7];
    $isbn     = $dat[8];
    $contents = $dat[9];
    $image    = $dat[10];
    if ( $id == "" )    continue;

    $sql = "insert into book values (";
    $sql = $sql . "$id,";
    $sql = $sql . "'$title',";
    $sql = $sql . "'$author',";
    $sql = $sql . "'$series',";
    $sql = $sql . "'$seldate',";
    $sql = $sql . "'$type',";
    $sql = $sql . "'$appendix',";
    $sql = $sql . "$price,";
    $sql = $sql . "'$isbn',";
    $sql = $sql . "'$contents',";
    $sql = $sql . "'$image')";
    //  print("$sql <br>");

    $stmt = OCIparse($conn,$sql);
    if ($stmt) {
      OCIexecute($stmt);
      echo("OK<br>");
    }
    else {
      echo("Error<br>");
      break;
    }
    OCIfreestatement($stmt);
  }
  fclose($fd);
?>
</center>
</body>
</html>
<?php
    OCIlogoff($conn);
?>
```

プログラミング工房



コンピュータがどのように動作しているかを知らないと、なかなかC言語でプログラムを書けるようにならないものである。今月号では、機械語やアセンブラのコードを実際に実行しながら、コンピュータの動作の本質について理解を進める。

第23回　C言語を理解するための機械語入門

文：藤沢敏喜
Text : Toshiki Fujisawa

コンピュータの基本的な動作

20年ほど前にコンピュータを自作するというのは、秋葉原でCPUやメモリなどのチップを買ってきて、数十個のチップを等間隔に穴が開いた汎用の基板に載せ、チップの端子を無数の導線で半田付けすることであった。

時代は変わり、PCの自作をするには、マザーボードを買ってきて、それにCPUとメモリを挿し、ケースに格納すればよいという便利な世の中になった。そして、PCを組み立てることにより、CPUやメモリがどういう形状をしているかを知っている人は増えたのだ。

しかし、実際にプログラムがどのようにしてメモリに記録され、どのようにCPUと通信し、CPUの中でどんな演算が行われているかは、意外に知られていないような気がする。C言語は、CPUが扱う機械語に密接に関係しているため、このあたりの事情をよく理解しないと深い理解は難しい。特に、メモリにおけるアドレスとデータの関係が理解できないと、C言語の特徴であるポインタや構造体などがわからずに挫折することになる。

そこで今月号では、まず最初にコンピュータの基本的な動作を解説する。そして、C言語で書いたプログラムがどのようにして実行されるかを、実際にプログラムを作成し、それを実行することによって理解していく。

コンピュータの歴史とその構造

1936年にアランチューリングが万能チューリング機械の論文を発表し、1945年にフォン・ノイマンがプログラム内蔵式のコンピュータを提案して（**脚注1参照**）、これがコンピュータの基礎となった。そのため、現在のコンピュータはノイマン型コンピュータと呼ばれるが、このタイプのコンピュータでは、プログラム（命令）をメモリに保存しておき、その命令をCPUが順番に読み出し、その内容に従ってCPU内部にある演算ユニットで計算が行われる（**図1**）。また、命令に従い、演算するべきデータをメモリ

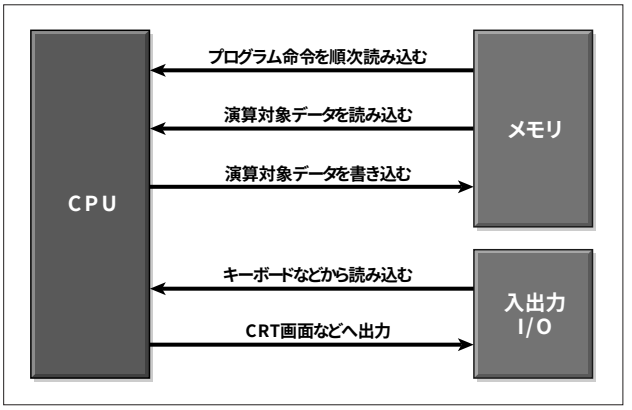


図1　ノイマン型コンピュータ

脚注1
プログラム内蔵式コンピュータのアイデアは、ENIACの開発者モークリーとエッカートが先だという説もある。

から読み出したり、演算した結果のデータをメモリに保存したり、CRT画面など外部の装置へ出力したりする。

ノイマン型コンピュータの大きな特徴は、データとプログラムを区別しないで1つのメモリ中に混在させるという部分である。データとプログラムが混在しているという具体的な状態が想像しにくい読者もいるだろう。したがって、この概念を理解するため実際にプログラムを書いて実験してみる。

メモリの構造とCPUからのアクセス

メモリとは、その名前のとおりデータを記録する装置である。電気的に情報を記録するためには、フリップフロップと呼ばれる回路を用いて記録する方法と、コンデンサに電荷をためてその電荷の有無により、0と1を記録する方法があるが、大容量を実現できる後者が現在では主流となっている。

この電荷をためる方法では、放電によりその内容が薄れていくため、動的（Dynamic）に充電をしてリフレッシュする必要がある。そのため、この方法を採用したメモリは「DRAM」と呼ばれる。RAMというのはRandom Access Memoryの略であるが、これはテープ装置などのシーケンシャルな記憶媒体と違い、記憶領域の任意のデータに瞬時にランダムにアクセスできるという意味である。メモリにおいては、アクセスするデータの1つ1つに番地を割り当て、このアドレスを元にデータを特定している。

メモリは前述のように、電荷の有無で1ビットを表し、CPUからは1ビットずつ1本の電線を通じてアクセスする。64本の電線でアクセスする場合、データバス幅が64ビットであるといい、CPUは物理的には64ビットのデータに同時にアクセスできることになる。

一方、CPUの論理的なデータの最小アクセス単位は8ビット（＝1バイト）であり、そのデータを指定するためのアドレス範囲は多くの場合32ビットである。このアドレスの範囲は空間とも呼ばれ、そのサイズは約4Gバイト（2の32乗）である。

ここで、説明の都合上メモリのバス幅が8ビットであると仮定してみる。この場合、CPUがメモリからデータを読み出すには、まず、**図2**のようにアドレスを示す32本の電線の電圧を0Vか電源電圧のどちらかに設定する。すると、メモリはアドレスにあるデータに従い、8本のデータを示す電線の電圧を変化させ、CPUはそのデータを読み取るということになる。メモリに書き込む場合は、最初にCPUが書き込みたいアドレスを出力し、次に書き込みたいデータをメモリに対して出力することになる。

以上のように、メモリとは、ソフトウェアの面から見れば、32ビットのアドレスを与えるとそのアドレスに対応するデータ8ビット分を読み書きすることができる装置である、と考えることができる。そしてプログラムの実行というのは、このメモリから読み込んだり、書き込んだりする操作であると言える。

CPUの内部にあるレジスタ

CPUがメモリにアクセスするには、上記のように電線を通じて行うが、その通信速度には限界がある（最近のよう

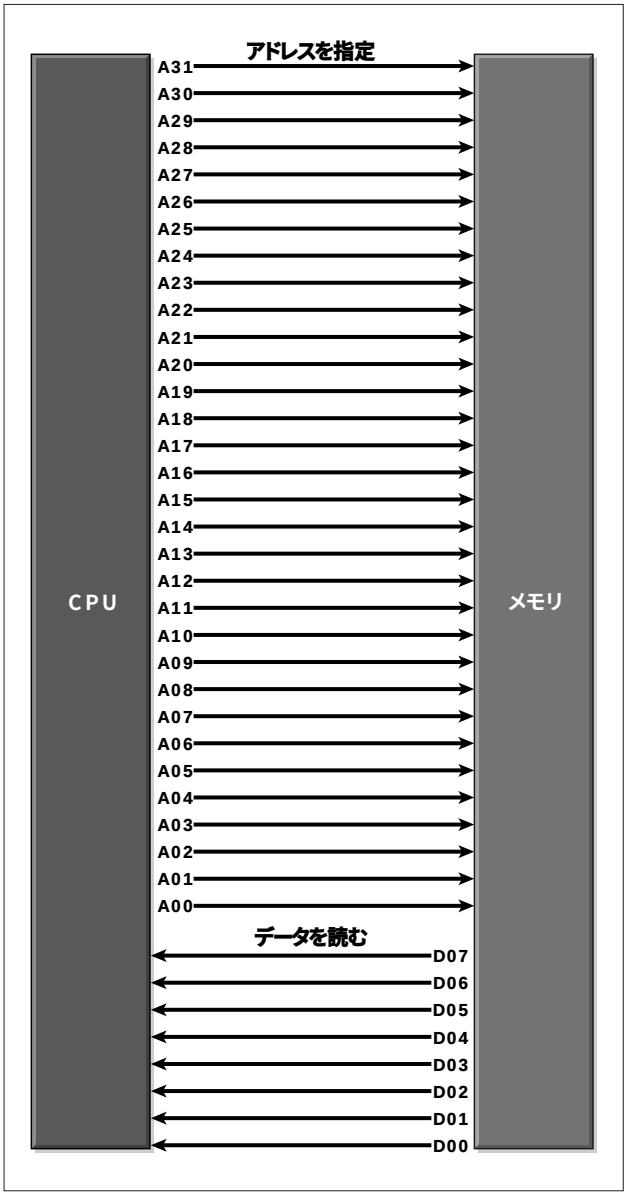


図2　CPUからメモリのデータを読み出す

にCPUのクロックが2GHzになると、1クロックの間には真空中の光でさえ15cmしか進めない)。したがって、高速化のために、CPUの内部には一時的な記憶領域が設けられている。これをレジスタと呼ぶ。インテルのi386以降のCPUでは、レジスタ構成は**図3**のようになっている(実際にはセグメントレジスタなどもあるが、ここでは省略する)。

eax、edxは通常の算術演算などに使われ、esi、edi、xcxは文字列操作のソースインデックス、ディストネーションインデックス、カウンタなどに用いられる。また、espは、スタックポインタとして用いられ、eipは現在実行している命令(インストラクション)の位置(ポジション)を記憶するために用いられる。

eipは、プログラムの実行という意味では、たいへん重要なレジスタである。CPUはeipで指定されるアドレスから命令データを読み込み、その命令データに従った処理を行う。そして、命令の処理が終わるとeipをインクリメントして、次の命令実行に備えるのである。また、命令を実行した結果、サブルーチン呼び出しを行うことになった場合は、そのサブルーチンの先頭アドレスがeipにセットされる。

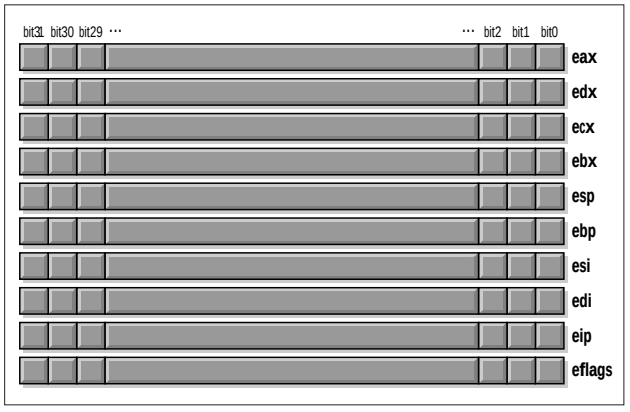


図3　386アーキテクチャにおけるレジスタ構成

機械語

前述のように、ノイマン型コンピュータはメモリに記憶した命令を順番に読み出し、それを逐次実行していくことが特徴である。このメモリに記憶する命令というのは、単なる数値の列であるが、機械が理解するのに便利な言葉であるため、「機械語」と呼ばれている。

さて、概念説明はこれくらいにして、機械語でプログラミングをしてみよう。まず最初に、かけ算を実行するC言

▼リスト1　C言語のかけ算関数を機械語で記述

```
 1 #include <stdio.h>
 2
 3 /*
 4  *  以下の機械語は、この関数と同様の動作をする。
 5  *
 6  *  int
 7  *  mul(int arg1, int arg2)
 8  *  {
 9  *          int d;
10  *
11  *          d = arg1;
12  *          d = d * arg2;
13  *          return d;
14  *  }
15  *
16  */
17
18 unsigned char machine_language[] = {
19         0x8b, 0x54, 0x24, 0x04,       /* d = 1番目の引数      */
20         0x0f, 0xaf, 0x54, 0x24, 0x08, /* d = d × 2番目の引数 */
21         0x89, 0xd0,                   /* 戻値 = d             */
22         0xc3,                         /* return               */
23 };
24
25 #define mul ((int (*)(int,int))machine_language)
26
27 int
28 main(void)
29 {
30         printf("%d\n", mul(8,9) );
31         return 0;
32 }
```

語の関数を機械語で書いてみる。**リスト1**がそのプログラムソースコードで、**リスト1**の6～14行が今回機械語で作成する関数の内容である。1番目の引数の数値と2番目の引数の乗算を行い、その結果を返すという単純な関数であるが、ここでは機械語との対比のため、dというローカル変数を使って計算をするようにしている。

この関数を機械語で書いたものが、19～22行めの16進数の列である。この関数を16進数で10000000番地からのメモリに配置する場合、メモリの中には次のような形で記憶されることになる。

```
アドレス      機械語データ
10000000:    8b
10000001:    54
10000002:    24
10000003:    04
10000004:    0f
10000005:    af
10000006:    54
10000007:    24
10000008:    08
```

Column

逆アセンブル

機械語を人間がわかりやすいニーモニック表示にしてくれるプログラムが「逆アセンブラ」である。逆アセンブルをするには、http://biew.sourceforge.net/で配布されるBIEWなどを使用することもできるが、GNUのbinutilに含まれるobjdumpを使い、

```
objdump -D a.out
```

のようにするのが便利である。なお、GNUのデバッガであるgdbを使うと、逆アセンブルしながら機械語レベルで1ステップずつ命令を実行することもできる。

Column

スタックとフレームポインタ

C言語の関数のように、呼び出して、呼び出し元に戻ってくるプログラムでは、戻り番地を保存しておかなければならない。戻り番地を保存するためには、スタックと呼ばれるRAM領域が使用される。このRAM領域は、アドレスが大きい番地からアドレスの小さい番地へ向かって使われてゆき、現在まで使用している位置が「スタックポインタ」と呼ばれるレジスタに保存される（図4）。i386アーキテクチャでは、espがスタックポインタに使われるレジスタである。

また、関数では引数やローカル変数を使用するが、これらの変数のRAM領域は関数の終了とともに使われなくなる領域であるためスタックに保存される。なお、スタックポインタは関数実行中に動的に変わってしまうことがあるので、関数の開始時にスタックポインタの位置を「フレームポインタ」と呼ばれるレジスタ（i386ではebpレジスタ）に記憶し、ローカル変数をアクセスする場合はフレームポインタからの相対アドレスを用いてアクセスすることが多い。

i386では、フレームポインタ設定を高速にするため、関数の最初に実行するenter命令と、関数のリターン時に実行するleave命令を持っている。

スタックを使用すると、関数が呼ばれるたびにスタックは消費され、スタックポインタはアドレスの小さいほうを指すことになる。また、関数がリターンするときにスタックポインタを元に戻せばよいので、RAMを有効に活用することが可能になる。

```
int grandchild(int a, int b)
{
        long i = 111;
        long j = 222;

        return a * b;
}

void child(void)
{
        long x = 7;
        long y = 8;

        grandchild(x,y);
}

void parent(void)
{
        child();
}
```

アドレスの例	スタックの内容	
1FFFFD0	未使用	
1FFFFD4	未使用	
1FFFFD8	孫関数のローカル変数(long j)	ebp-8
1FFFFDC	孫関数のローカル変数(long i)	ebp-4
1FFFFE0	子関数用フレームポインタを保存	←孫関数実行時のebpの位置
1FFFFE4	子関数への戻り番地	←孫関数開始時にespが指す位置
1FFFFE8	孫関数の第2変数(long a)	ebp+8　esp+4
1FFFFEC	孫関数の第1引数(long b)	ebp+12　esp+8
1FFFFF0	子関数のローカル変数(long y)	
1FFFFF4	子関数のローカル変数(long x)	
1FFFFF8	親関数用フレームポインタを保存	
1FFFFFC	親関数への戻り番地	

図4　スタックポインタの例

```
10000009:    89
1000000A:    d0
1000000B:    c3
```

この16進数の数値列を見ても、人間にはなんのことやらさっぱりわからないが、CPUはこれを理解して忠実に実行していくのである。

CPUは、まずeipレジスタに命令の先頭アドレスである10000000を代入し、そのアドレスのデータ8bを読み込む。すると、そのデータから続く4バイト、すなわち「8b 54 24 04」の部分が、1番目の引数が存在するメモリにあるデータをCPU内部のedxレジスタに読み込む命令であることを理解する。そして、eipのアドレスを4つ進め、「0f af 54 24 08」という命令列で、2番目の引数が存在するメモリにあるデータを読み込み、そのデータとedxレジスタの内容の乗算を行い、結果をedxに保存する。

なお、i386のgccでは戻値をeaxレジスタに記憶することになっているため、「89 d0」という機械語命令により、乗算結果が保存されているedxレジスタの内容をeaxレジスタへとコピーする。そして、「c3」という機械語命令により、この関数が呼び出された親関数へとリターンすることになる。

リスト1の25行めには、

```
#define mul ((int (*)(int,int))machine_language)
```

という記述があるが、これはmachine_languageで示されるアドレスから保存される機械語命令を、int型の2つの引数を持ち、int型の結果を返す関数へのポインタとなるようにキャストしている。

ちょっとわかりにくいかもしれないが、要するに、

```
mul(8,9)
```

と記述した場合、上記の機械語が書かれている先頭アドレスからの命令群がサブルーチンコールされ、機械語の先頭アドレスの命令にジャンプし、上記の機械語の命令列が実行されることになる。その結果、8と9の乗算結果である72が、printf関数に渡されて表示が行われるのである。

プログラムによる自分自身の書き換え

先ほど、ノイマン型コンピュータでは、プログラム命令とデータが1つのメモリに混在することが特徴であると述べたが、**リスト1**を見ればこの機械語部分は、まぎれもなく単なるデータであることがわかるはずだ。

ここで、もう少しこの概念を理解するために、**リスト2**のプログラムを実行してみよう。このプログラムでは、先ほどの**リスト1**のプログラムと同じ機械語が書かれているが、17行めから21行めで機械語命令の5番目から9番目の部分を、

▼リスト2　機械語命令の書き換えの例

```
 1 #include <stdio.h>
 2
 3 unsigned char machine_language[] = {
 4         0x8b, 0x54, 0x24, 0x04,       /* d = 1番目の引数      */
 5         0x0f, 0xaf, 0x54, 0x24, 0x08, /* d = d × 2番目の引数 */
 6         0x89, 0xd0,                   /* 戻値 = d             */
 7         0xc3,                         /* return               */
 8 };
 9
10 #define mul ((int (*)(int,int))machine_language)
11
12 int
13 main(void)
14 {
15         printf("%d\n", mul(8,9) );
16
17         machine_language[4] = 0x03;
18         machine_language[5] = 0x54;
19         machine_language[6] = 0x24;
20         machine_language[7] = 0x08;
21         machine_language[8] = 0x90;
22
23         printf("%d\n", mul(8,9) );
24         return 0;
25 }
```

```
03 54 24 08
90
```

に、書き換えている。「03 54 24 08」は、

```
d = d + 2番目の引数
```

という操作の機械語命令である。また、90は何もしないNOP（Non Operation）命令である。

つまり、リスト1では乗算命令だった部分を、加算命令に置き換えているのである。乗算命令は5バイトであったが、加算命令は4バイトであり、1バイト足りないためNOP命令で1バイト分を埋めている。

リスト2をコンパイルして実行すると、最初に8と9の乗算結果である72が表示され、そのあとに8と9の加算結果である17が表示されるはずだ。

この例を実際に実行してみれば、プログラムというのはメモリ中に記憶された単なる数値データであることがよくわかっていただけるのではないかと思う。

ニーモニックとアセンブリ言語

月刊アスキーが創刊された今から20数年前には、上記で説明したような機械語が雑誌に掲載され、それを人間が入力してゲームすることが日常的であった（**脚注2参照**）。

当時は、主流のZ80や6809といったCPUの機械語をすべて暗記し、16進数をキーボードから入力して、プログラミングする強者がたくさんいたのである。

しかしながら、機械語をそのまま扱うのはあまりにも非人間的行為である。そのため、機械語と1対1に対応する、人間にわかりやすい単語を用いた言語が使われるようになった。この単語は「ニーモニック」と呼ばれ、それを使って書かれたものを「アセンブリ言語」と呼び、アセンブリ言語を機械語に変換するプログラムを「アセンブラ」と呼ぶ。

先ほどの乗算関数の機械語とニーモニックの対応は次のようになる。

```
8b 54 24 04     ⇔ movl   0x04(%esp),%edx
0f af 54 24 08  ⇔ imul   0x08(%esp),%edx
89 d0           ⇔ movl   %edx,%eax
c3              ⇔ ret
```

上記で、movlは「move long」の略で、imulは「integer Multiplication」、retは「return」の略である。

また、0x04（%esp）は、スタックポインタespから4バイト足したアドレスが指すデータを示している。C言語の関数引数は、スタックと呼ばれる領域に順番に格納されるが、int型が4バイトであるため、0x04（%esp）が第1引数、0x08（%esp）が第2引数を指すことになる（上記はGNUアセンブラであるasを使用する場合の表記である

脚注2
アスキーから2001年4月1日に出版された「パロディ版BSDマガジン」には、昔流行した「南青山アドベンチャー」のパロディである「南新宿アドベンチャー」というゲームが、細密base64ダンプリストで掲載されている（アスキーは南青山から南新宿に移転した）。

Column

アセンブラでのHello World

アセンブリ言語で、「Hello World」と表示するプログラムを作成するには、**リスト3**をhello.sとして保存し、

```
$ as hello.s -o hello.o
$ ld hello.o
$  ./a.out
```

というように、アセンブルとリンクを行う。

なお、Linuxでは、システムコールを呼び出すためにint命令（interrupt命令）を使用するので、eaxレジスタにシステムコール番号を入れ、ebx、ecx、edxに引数を順番に入れてから、int $0x80を呼び出すことになる。

▼リスト3　アセンブリ言語によるHello World

```
 1          .global _start
 2  _start:
 3          movl $4,%eax     ※4はwriteシステムコール
 4          movl $1,%ebx
 5          movl $buf,%ecx
 6          movl $len,%edx
 7          int  $0x80       ※write(1,buf,len)
 8
 9          movl $1,%eax     ※1はexitシステムコール
10          int  $0x80
11
12          .data
13  buf:    .string "Hello World\n"
14          .equ    len,     .-buf
```

が、「MOV EDX, DWORD PTR [ESP+4]」と書くアセンブラもある）。

アセンブラを使えば、機械語を覚えなくてもプログラミングが可能になり、

```
movl   0x04(%esp),%edx
imul   0x08(%esp),%edx
movl   %edx,%eax
ret
```

というファイルを作成して、mul.s として保存し、アセンブルリストを作成する「-a」オプションを付けて、

```
as -a mul.s
```

を実行すると、**リスト4**のようにアセンブラが機械語への翻訳を行ってくれる。

Cコンパイラ

アセンブリ言語は、機械語に比べればはるかにわかりやすいが、機械語と一対一に対応しているため、人間にとってはわかりやすい言語とはいえない。このため、アセンブラよりもわかりやすい言語を、アセンブラ言語に翻訳するプログラムが開発された。この翻訳をするプログラムが「コンパイラ」である。

ここで、**リスト5**の内容をmul.cというファイル名で保存し、gccにソースを出力するオプション「-S」と、その出力結果を画面に表示するためのオプション「-o -」を付け、

```
gcc  -S -o - mul.c
```

として実行する。すると、**リスト6**のようなアセンブリ言語に翻訳される。

上記のように、C言語からのアセンブラ出力は簡単に行えるので、いろいろなサンプルコードを書いてみて、そのコードが実際にどのようなアセンブリ言語に翻訳されるかを見てみると、C言語に対する理解が深まるはずだ。

まとめ

C言語によるプログラミングを行ううえで、CPUがどのようにして命令を実行し、どのようにしてメモリにアクセスするかということを理解することは重要である。この概念をしっかりと理解したあとで、CPUがアドレスを出力し、メモリの読み書きを行う状況を想像してみてほしい。そうすれば、一見難しく見えるポインタや構造体というものが、本質的にどういうものであるかがよく理解できるのではないかと思う。

なお、次号のプログラミング工房は、特集記事執筆のため休載します。

▼リスト4　アセンブラの翻訳結果

```
GAS LISTING mul.s                       page 1


   1 0000 8B542404       movl   0x04(%esp),%edx
   2 0004 0FAF5424       imul   0x08(%esp),%edx
   2      08
   3 0009 89D0           movl   %edx,%eax
   4 000b C3             ret

GAS LISTING mul.s                       page 2

NO DEFINED SYMBOLS

NO UNDEFINED SYMBOLS
```

▼リスト5　mul.c

```
int
mul(int arg1, int arg2)
{
        return arg1 * arg2;
}
```

▼リスト6　コンパイラの出力結果

```
        .file   "mul.c"
        .version        "01.01"
gcc2_compiled.:
.text
        .align 4
.globl mul
        .type    mul,@function
mul:
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%edx
        imull 12(%ebp),%edx
        movl %edx,%eax
        jmp .L1
        .p2align 4,,7
.L1:
        leave
        ret
.Lfe1:
        .size    mul,.Lfe1-mul
        .ident  "GCC: (GNU) egcs-2.91.66
19990314/Linux (egcs-1.1.2 release)"
```

特別講座

ステップアップC言語

asm（）文

■C言語中でアセンブラを使用するには

C言語のソース中でアセンブリ言語を使用する必要がある場合、本文で説明したように機械語をそのまま埋め込むという方法もあるが、これではあまりに不便である。

一方、アセンブリ言語ファイル（＊.s）で関数を記述して、それをC言語で書いたものと一緒にリンクする方法もあるが、その場合には関数単位でしかアセンブラの記述ができないことになる。

C言語のソースコードの一部分だけにアセンブラを表記したい場合は、どうしたらよいだろうか？

■asm文

gccを使っている場合には、asm文を使うことによりC言語の中でアセンブリ言語を記述することができる。

たとえば、

```
#include <stdio.h>

asm("mul:                  ");
asm("  movl 0x04(%esp),%edx");
asm("  imul 0x08(%esp),%edx");
asm("  movl %edx,%eax      ");
asm("  ret                 ");

extern int mul(int a, int b);

int
main(void)
{

    printf("%d\n", mul(8,9) );
    return 0;
}
```

のようにすると、本文で解説した乗算を行うmul関数を定義することができる。なお、上記で、

```
extern int mul(int a, int b);
```

というプロトタイプ宣言を行っているのは、コンパイラには、asm（"mul:"）というラベルが認識できないからである。

また、asm文は関数の中にも書くことができる。

```
#include <stdio.h>

int
main(void)
{
    int a;

    asm(" movl $12345678,-4(%ebp)");

    printf("a=%d\n", a );
    return 0;
}
```

というようなプログラムを書いて実行すると、「a=12345678」という表示結果が得られる。

■asm文の中からの変数参照

上のプログラムでは、ローカル変数に定数を代入するために、

```
asm(" movl $12345678,-4(%ebp)");
```

という文を用いた。ローカル変数はスタック上に確保されるため、フレームポインタebpを通してアクセスできるのである。

一方、グローバル変数は、

```
int global;

asm(" movl $12345678,global");
printf("a=%d\n", global );
```

というようにしてアクセスできる。

■asm文使用時の注意

asm文は気軽に使用できるものの、注意しなければならない点がある。まず、asm文で使用するレジスタに制限があるということに気をつけなければならない。コンパイラがループの数をカウントするのに特定のレジスタを使用している場合、ループの中のasm文でそのレジスタを破壊すると壊滅的なことになるのである。このような場合、asm文で使用するレジスタをスタックにプッシュして保存するなどの対応が必要になる。

また、コンパイラの最適化にも注意が必要である。ローカル変数がasm文の中だけでしかモディファイされていない場合、コンパイラがその変数を消去することがある。

■asm文のメリット

gccなど、最近のコンパイラは高度に最適化され、下手に人間がアセンブラで書くよりもコンパイラにまかせたほうが高速なことも多い。

しかしながら、場合によってはやはり人間がギチギチとアセンブラで書いたほうが速いこともある。このようなケースでは、本当に速くする必要があるのはほんの一部分だけであることが多いが、そのような部分にだけasm文を使うと便利である。

また、I/O命令のように、C言語で記述できない命令を使用したい場合にも、asm文は有効である。

さらに、1行に複数命令を書くことができ、強力な#defineマクロも使えるため、アセンブラで書くよりも、読みやすいソースを作成することができる。

そして、複数ファイルのソースコードを管理するうえでは、＊.cだけのほうが都合がよいというのも見逃せない点である（筆者には、苦労して作成した＊.sファイルを、間違ってrm ＊.sコマンドで消してしまったという苦い経験がある）。

Perlスクリプト入門
すぐにできる実践Perl

Perlで作るCGIのおもしろさは、GUIなどの画面まわりの能力を持たないどちらかというと縁の下の処理ツールであったPerlが、非常に身近なWebインターフェイスを縦横無尽に駆使できるということにあるのでしょう。

第9回　実践CGI（2）

文：おもてじゅんいち／かざぐるま
Text : Junich Omote/Kazaguruma

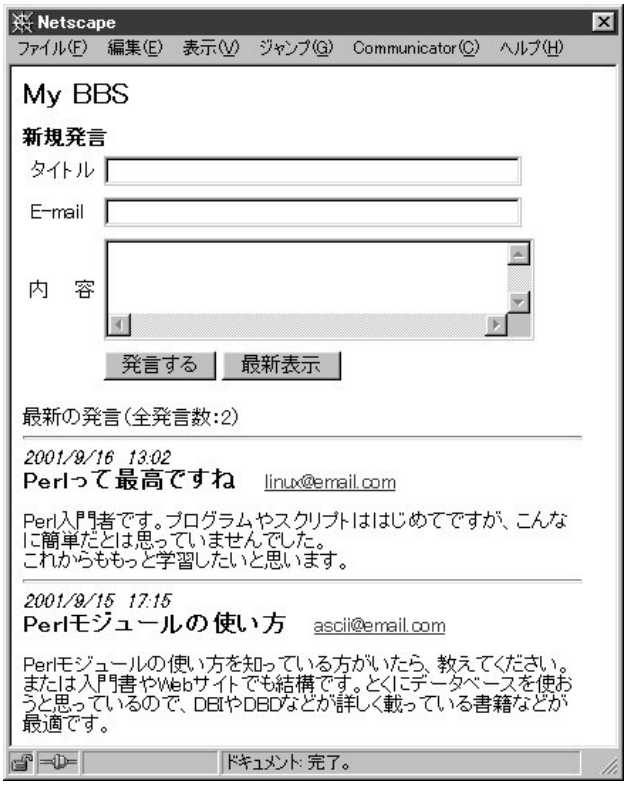
今回は、Webでおなじみの掲示板をCGIで作ってみます。Webの掲示板は説明する必要もないと思いますが、要するに発言が登録できて、登録された発言が順次表示されていくといった仕組みのものです。便利な機能として、登録された発言に対するコメントを通常発言とは別に「Re:〜」という形で扱うものや、管理側の機能として発言された内容がいったん管理者の元に届き、管理者が確認してはじめて掲示板に表示されるものなど、インターネットで目にする最近の掲示板にはさまざまな機能が付加されています。

完成品としての掲示板CGIがフリーソフトとして使えたり、CGIの入門書などで機能豊富な掲示板CGIのソースが提供されたりしているようですが、いずれも簡単に導入できてすぐに使える反面、自分で作り上げたものではないのでちょっとしたCGIの修正などが行いづらく、かゆいところに手が届かないものです。

そこで今回は、機能としては必要最小限で実運用に使うにはちょっと物足りないかもしれませんが、一から自分で理解しながら掲示板を作り上げてみましょう。自分で理解できていれば、新しい機能を付け加えたり、ちょっとした修正を行うことが簡単にできます。

掲示板を作る

画面がその掲示板です。上部に発言を書き込むフォームがあり、下部に発言された内容を表示しています。通常は書き込むフォームと発言の表示部分をフレームで分けていたり、別の画面にしているものも多いのですが、今回はできるだけ簡単に作れるように1画面でその両方をまかなっています。



画面　掲示板CGI

この掲示板は発言を書き込んで登録するとすぐに画面が更新され、下部の一番上にその発言が表示されていきます。CGI自体は1ファイルですから、発言を登録するときにこのCGIが再度自分自身を呼び出す方法を採っています。

CGIは2つの機能からなっています。1つはフォームに書き込まれた発言内容をサーバ内のテキストファイルに書き込む機能、もう1つはテキストファイル内の発言データを順次画面に表示する機能です。

この2つがうまく連携して動けば、掲示板として働くのです。発言を書き込んで登録ボタンを押したときは、

A フォームの内容をテキストファイルに書き込む
B テキストファイルのデータを順次画面に表示

という2つの処理を行えばよいですし、発言を書き込まずにCGIを呼び出したときは、

B テキストファイルのデータを順次画面に表示

という処理だけを行えばよいことになります。つまり、「フォームのデータが送られてきたときはAの処理を行い、フォームデータがないときはAの処理を行わないで、いずれの場合にもBの処理は行う」と考えればよいのです。

ではさっそくCGIスクリプトを見てみましょう（**リスト1**）。今回もフォーム入力を使うので、cgi-lib.plが必要になり、フォームデータを処理するために&ReadParse;を実行しておく必要があります。

リスト1の①はヒアドキュメントで、Content-typeヘッダと、フォーム部分のHTMLを出力しています。フォームそのものは常に同じ形で表示されるものなので、固定したHTMLとして出力します。Content-typeヘッダの行の次に空行を1行入れるのを忘れないようにしてください。

フォームの構成は、タイトルとEメール欄がテキストボックス、内容欄は複数行ですからtextarea、そして「発言する」というsubmitボタンとなります。

このフォームに文字を入力して発言ボタンを押すと、画面が更新されてフォームの中身もクリアされるのですが、その次にブラウザの「再表示」ボタンを押すとブラウザによってはもう一度フォームの内容を送り直してしまいます。送り直してしまうと、同じ発言が2度登録されることになりますから、それを避けるためにフォーム上に「最新表示」というボタンを用意しておきました。本当は、ブラウザの「再表示」ボタンを押しても問題が起こらないようにスクリプトを修正しなければならないのですが、ちょっと複雑になってしまうのでとりあえずの解決策として「最新表示」ボタンにしました。

「最新表示」ボタンはフォームデータを送り直さずに画面だけを再表示できればよいのですから、「最新表示」ボタンが押されると単に自分自身であるbbs.cgiにジャンプするようになっています。フォームデータは何も送られないので、登録処理は行われずにデータを再表示します。

「最新表示」をボタンでなくリンクにしておけば、

```
<a href='bbs.cgi'>
```

というHTMLだけで済むのですが、デザイン的にボタンのほうがよかったので、ボタンを押したときにリンク先にジャンプするように、次のようにJavaScriptを使って書い

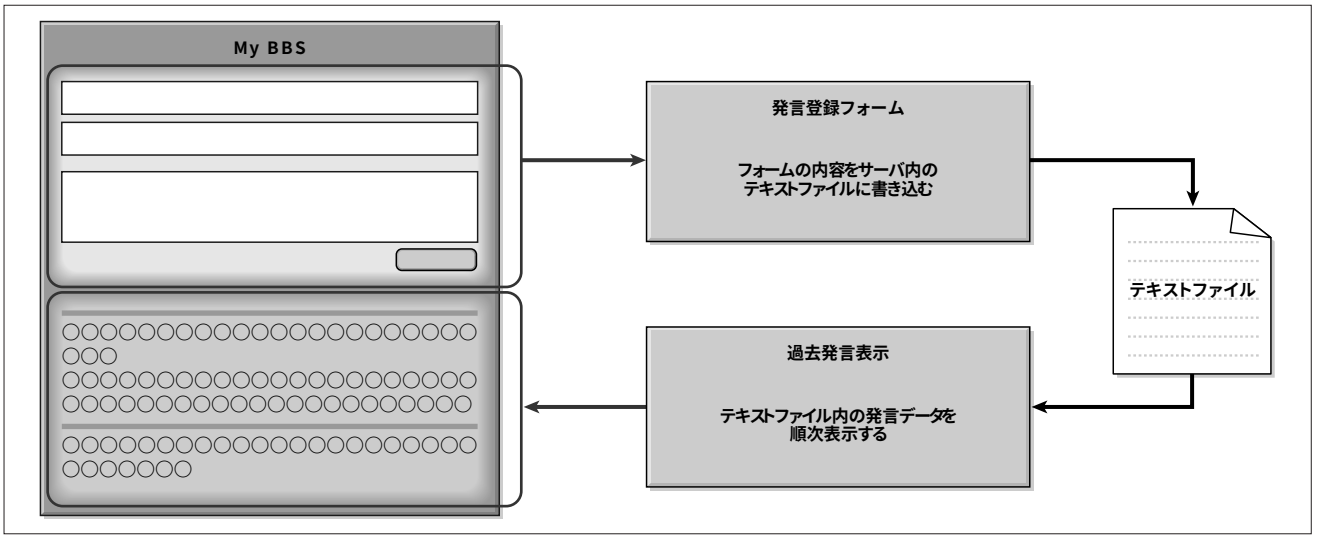


図1　掲示板CGIの動き

てあります。

```
onclick="JavaScript:location.href='bbs.cgi'"
```

これは、クリックされたら表示するURLを変更せよという意味になります。ブラウザ側を制御するためにクライアントスクリプトとしてのJavaScriptがよく使われるので

▼リスト1　掲示板CGIスクリプト

```
#!/usr/bin/perl
require 'cgi-lib.pl';
&ReadParse;

print <<'EOD1';
Content-type:text/html

<html><head></head><body>
<h2>My BBS</h2>
<form method="post" action="bbs.cgi">
<b>新規発言</b><br>
<table>
<tr><td>タイトル</td><td><input type="text" size=40 name="title"></td></tr>
<tr><td>E-mail</td><td><input type="text" size=40 name="email"></td></tr>
<tr><td>内　容</td><td><textarea cols=40 rows=3 name="cont"></textarea><br></td></tr>
<tr><td></td><td><input type="submit" value="発言する">
<input type="button" value="最新表示" onclick="JavaScript:location.href='bbs.cgi'"></td></tr>
</table>
</form>
EOD1
        # フォームの内容をテキストファイルに書きこむ
        $title = $in{'title'};
        $email = $in{'email'};
        $cont = $in{'cont'};
        if($title) {
                ($sec,$min,$hour,$day,$mon,$year) = localtime();
                $mon++;
                $year += 1900;
                $tm = sprintf("%02d:%02d",$hour,$min);
                $title =~ s/,/%%/g;
                $cont  =~ s/,/%%/g;
                $cont  =~ s/¥n/<br>/g;
                $line = "$year/$mon/$day  $hour:$min,$title,$email,$cont\n";
                open(OUT,">>bbs.txt");
                print OUT $line;
                close(OUT);
        }

        # 発言の表示
        open(IN,"bbs.txt");
        while(<IN>) {
                ($dtime,$title,$email,$cont) = split(/,/,$_);
                $title =~ s/%%/,/g;
                $cont  =~ s/%%/,/g;
                print "<hr><i>$dtime</i><br>";
                print "<font size=+1><b>$title</b></font>   ";
                print "<a href='mailto:$email'>$email</a><br><br>¥n$cont";
        }
        close(IN);
print <<"EOD2";
</body>
</html>
EOD2
```

①
②
③
④
⑤
⑥

すが、ここではJavaScriptについては詳しく触れません。

フォーム部分をHTMLで構成したあとはスクリプト処理部分になります。スクリプトで処理するのは、前述のように2つの処理になります。

まずフォームの内容をテキストファイルに書き込む処理ですが、テキストファイルの形式はCSVとしますので、基本的にはフォームに入力された各データを取得し、それらをカンマで区切りながら並べて1行にしてファイルに出力すればよいだけです。ただし、発言が書き込まれた日時も情報として記録しておきたいので、スクリプト側で自動的に処理します（**リスト2**）。

リスト1の②が日付と時間の処理で、localtime()で取得した日付と時間を使います。月（$mon）は0～11が返ってくるので1を加え、年（$year）には1900を加えます。これで正しい西暦と月の表示になります。時分は11:08のように1桁の場合でも0を付けて2桁にしますので、

```
$tm = sprintf("%02d:%02d",$hour,$min);
```

という処理を行っています。sprintf()は、書式付き出力のprintf()と同様の処理をする関数ですが、結果の出力先を画面でなく、文字列として返します。「%02d」という書式は、「2桁の数字で、2桁に満たないときは上位の桁を0で埋める」ということですから、時と分をその書式にしてその間に「:」を挟むことによって、正しい時分の表記になるのです。こうしてできあがった結果は$tmに文字列として格納されます。

③の部分ではフォームで入力された文字をちょっと加工しています。データを保存するテキストファイルはCSVですから、「,（カンマ）」は区切り文字として特別な意味を持っています。フォームで入力されたデータ内にカンマがあるとテキストファイルに保存されたあとに区切り文字との見分けがつかなくなって、正しい処理が行われません。ですから、最初の2行でタイトルと内容のデータ内のカンマを「%%」にいったん置き換えています。これはあとでテキストファイルからデータを読み込んだときに逆に置換し直しますので、データの内容が変わるということはありません。あくまでもテキストファイル内でだけカンマを使わないようにしておくのです。

最後は同じようにテキストデータ内で改行がそのまま保存されるとまずいので、改行を
というタグに置き換えます。これは最終的にブラウザに表示するときに必要なタグなので、このほうが都合がよいのです。

日時の取得とデータの加工が済んだら、テキストファイルに保存するだけです（**リスト1**の④）。ここでは、フォームで入力されたデータと日時の情報をCSVデータ1行分としてまとめています。$lineにカンマ区切りで1行にまとまったデータが格納されますので、それをテキストファイルbbs.txtに追加書き込みします。

ところで、ここまでの処理は発言が書き込まれたとき、すなわちフォームデータが入力されたときだけ行う処理で、フォームデータが送られてこないときは行ってはいけない処理ですから、⑤でその判別を行っています。判別は簡単で、タイトル文字が入力されていれば発言ありとみなし、なければ発言の登録でなく表示のみであると判断しています。

あとは、テキストファイル内の発言データを順次表示していくだけです。発言の表示部分のHTMLのひな型は、**リスト3**のようなものになりますから、これを元にデータの部分を挿入していきます。ただし、その前にテキストファイルから読み込んだCSV形式の1行データをsplit()で各変数に分割し、先ほど%%に置き換えた文中のカンマを元に戻します（**リスト1**の⑥）。そして、最後にHTMLのフッタ部分をヒアドキュメントで出力します。

▼リスト2　CSVファイルbbs.txtの例

```
2001/9/15  17:15,Perlモジュールの使い方,ascii@email.com,Perlモジュールの使い方を知っている方がいたら、教えてください。<br>または入門書やWebサイトでも結構です。とくにデータベースを使おうと思っているので、DBIやDBDなどが詳しく載っている書籍などが最適です。<br>
2001/9/16  13:02,Perlって最高ですね,linux@email.com,Perl入門者です。プログラムやスクリプトははじめてですが、こんなに簡単だとは思っていませんでした。<br>これからももっと学習したいと思います。
```

▼リスト3　発言表示部分のHTMLひな形

```
<hr><i>2001/9/16  13:02</i><br>
<font size=+1><b>Perlって最高ですね</b></font>  <a href='mailto:linux@email.com'>linux@email.com</a><br>
<br>
Perl入門者です。プログラムやスクリプトははじめてですが、こんなに簡単だとは思っていませんでした。<br>
これからももっと学習したいと思います。
```

掲示板の改良

さて、これだけで簡単な掲示板CGIが完成しました。しかし、これを動かしてみるとわかるのですが、

①発言が古いものから並んでいる
②発言が何件あっても1画面に表示される

という点が気になります。最低限の機能といっても、一応実際に使える掲示板を作るのが目的ですから、もう少しがんばってこれらの問題を解決しなければなりません。

①の問題ですが、通常、掲示板は新しい発言が一番上にあり、新しい発言を登録したらそれが一番上に挿入されて次々と下にずれていくものです。そうでないと、最新の発言を見るためにずっとスクロールすることになります。

次に②の問題はやはり、1画面あたり20件くらいの表示が適当で、それ以上、ましてや全発言がいつも1画面に表示されたのではすぐに長いページになってしまいます。

ということで、この2点に取り組んでみましょう。**リスト4**はその改良部分です。修正するのは**リスト1**の、

```
#発言の表示
```

以降です。新しい発言から表示するときに困るのは、テキストファイルに新しい発言が追加保存されていくときに、ファイルの最後にどんどん追加されてしまっていることです。テキストファイルからデータを読み込むのは必ずファイルの先頭からですから、普通に読み込んだのではどうしても古い発言から読み込むことになってしまうのです。Perl自体にはファイルを最後から読んでいくという機能はありませんから、自分で工夫しなければいけないようです。

そこで、ファイル内のデータをどのように読んでいけば新しい発言から、しかも1画面20件ずつに分割して読めるかを考えたのが**図2**です。

図2にあるように、まず最初の画面ではファイルの最後の20行（20件）のデータを読み込み、次の画面ではその手前の20行、その次にはそのまた手前の20行と読み込んでいけばよいことがわかります。ただし、「最後の20行」というのは人間の表現で、スクリプトを作るには具体的に「（先頭から数えて）何行めから何行めまで」ということがわからなければなりません。

ファイル全体の行数がわかれば、最後の20行が先頭から数えて何行めから何行めまでであるかというのは計算できそうです。ですから、その結果を$start、$endという変数に格納するものとします。そうなれば、各画面表示のときに読み込むのは、「（先頭から数えて）$start行めから$end行めまで」ということになります。

スクリプトで実現するにはもう少し言い換えて、「先頭から各行を読み込んでいって、$start行めになったら表示を始め、その後、$end行めになったら表示をやめる」という処理になります。

では実際のスクリプトを見て行きましょう。**リスト4**の①でまずテキストファイルの全行数を調べています。<IN>で1行ずつ読み込みますが、ここではまだ読み込んだデータは使わずに、読み込むたびに$cntを+1していきます。ループが終わった時点で$cntの値は全行数になっていることになります。

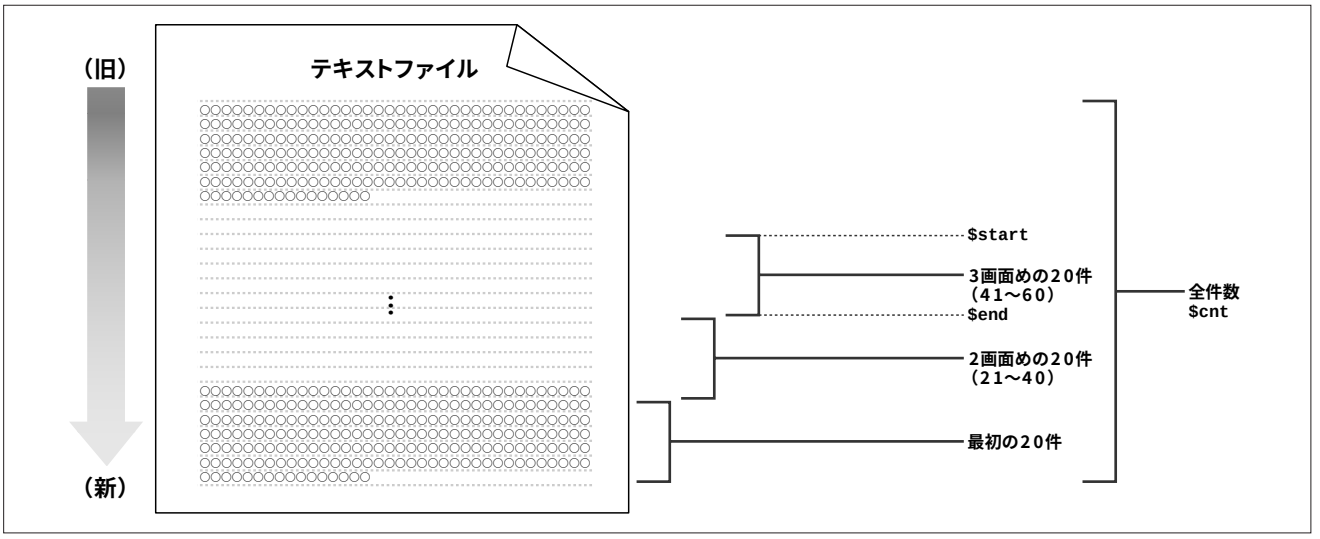


図2　テキストファイルと画面表示の対応

全行数がわかったところで$startと$endを算出するのですが、②の$nextというのは今から表示するのが何画面めかという切り替えに使います。

今はとりあえず最初の画面（$nextは0）として話を進めていきます。

③で$startと$endを計算します。

最初の画面ですから、**図2**を見ると$startは全行数から20行手前のところになります。となると、$nextは今考えないものとして、

```
$start = $cnt - 20
```

となります。$endは常に$startよりも20行先を指しているのですから、

▼リスト4　改良した掲示板CGI

```
        (ここまではリスト1と同じ)

        # 発言の表示
        $cnt = 0;
        open(IN,"bbs.txt");
                while(<IN>) {  $cnt++;     }  ①
        close(IN);

        open(IN,"bbs.txt");
        $next = $in{'next'};                    ②
        $start = $cnt - 20 - $next;
        $end = $start + 20;                     ③
        $c = 0;
        while(<IN>) {
                $c++;
                if ($c <= $start) {  next;  }   ④
                if ($c >  $end)   {  last;  }

                @line = (@line,$_);             ⑤
        }
        close(IN);

        if ($start < 0)  {      $start2 = $cnt;   }
        else                    {       $start2 = $cnt - $start; }
        $end2   = $cnt - $end + 1;
        if ($end2 == 1)  {      print "最新の発言";  }
        else                    {       print "$end2番め〜$start2番めの発言";  }   ⑥

        print "(全発言数：$cnt) ";

        foreach (reverse @line) {               ⑦
                ($dtime,$title,$email,$cont) = split(/,/,$_);
                $title =~ s/%%/,/g;
                $cont  =~ s/%%/,/g;
                print "<hr><i>$dtime</i><br>";
                print "<font size=+1><b>$title</b></font>    ";
                print "<a href='mailto:$email'>$email</a><br><br>¥n$cont";
        }

        if ($start > 0) {                       ⑧
                $next += 20;
                print "<hr><a href='bbs.cgi?next=$next'>前の発言</a><br>";
        }
print <<"EOD2";
</body>
</html>
EOD2
```

```
$end = $start + 20
```

ということになりますね。これで$startと$endが計算できましたから、実際にファイルを読み込みます。

④によって、前述の「先頭から各行を読み込んでいって、$start行めになったら表示を始め、その後、$end行めになったら表示をやめる」という処理を実現しています。

while(<IN>)ループはファイルの全行を順に読み込んでいくのですが、ここでは1行読むたびに$cを+1しています。つまり、$cは「今、何行めを読んでいるか」を表していることになります。そこで、$cが$startの値以下であるあいだは、それ以降の処理をせずループだけを繰り返します。next;はそれ以降の処理をスキップしてwhile(<IN>)に戻り、次の行を読み込むということになります。

$cが$startを超えたら、それは表示すべきデータということになりますから以降の処理に進みます。ですが、それ以降ずっと表示し続けてもいいというわけではありませんから、今度は$cと$endの関係をチェックします。$cが$endを超えたら、つまり$startから表示を始めて20行分を超えたら、last;でループを抜ければよいということです。

結局、ループ内のその後の処理、すなわち⑤の部分には$start行めから$end行めまでの20行の間だけ到達する仕組みになっているのです。

あとはその20行を表示すればよいだけなのですが、実際には⑤の部分ではまだ表示を行っていません。というのは、その20行のデータは古いほうから順に並んでいるので、そのまま表示するとやはり古いほうから表示してしまうからです。そこで、この20行分のデータをいったん配列に格納しておき、逆順に取り出して表示するようにします。

⑤は、ファイルから読み込んだ各行を順に配列に格納する部分です。ちょっと特殊な書き方ですが、右辺は、

```
(@line,$_)
```

となっています。これは、「@lineという配列のうしろに、今読み込んだ$_をリスト（配列）として追加する」ことになりますから、これを再び@lineに格納してやれば結果として@lineは最後の要素が増えることになるのです。ちょっとわかりにくいかもしれませんが、配列とリストをもう一度復習してみてください。

とにかくこれで、@lineには読み込んだ20行のデータが順に格納されています。次に、格納されているデータを逆順に並べる必要があります。

次に⑥の部分は後回しにして、⑦を見てください。foreach()は配列@lineの要素を順に取り出してゆくのですが、ここで、

```
foreach (reverse @line)
```

と書かれています。この「reverse」が何を隠そう配列の要素を逆順にするものです。ここで、やっと古いほうから並んでいたデータを新しいほうから取り出していくことができるのです。foreach()ループの中では1行ずつ表示を行っています。これは**リスト1**とまったく同じです。

どうです？　ファイルから読み出すのはあい変わらず古いほうからでも、それをいったん配列に入れて、foreachとreverseで取り出してやれば簡単に逆順で表示することができました。蛇足ですが、「reverse」の代わりに「sort」と書くと、この20行がアイウエオ順に並び替えられて表示されます。掲示板ではあまり意味がありませんが、名簿などでは利用価値がありそうです。

さて、ここまでは最初の画面、すなわち最新の20件の発言の表示について考えてきました。次に考えなければならないのは次の20件の表示です。

2画面めには、最新から数えて21～40件めが表示されなければなりません。**図2**を見てみると、そのためには$startを手前に20行分ずらす必要があることになります。ということは、2画面めでは$startから20を引き、3画面めではそれからまた20を引き……というように、画面が進むごとに$startから20を引いていけばよいことになります。

ここで活躍するのは先ほど無視した$nextです。画面が進むごとに20ずつ加えた値を、

```
bbs.cgi?next=40
```

というようにクエリー文字列として渡し、スクリプト内では$nextでその値を取得して、$startから$nextを引いてやれば画面が進むごとに$startの値が20ずつ減っていくことになります。

説明はややこしいので、実際にスクリプトを見てみましょう。まず、最初の画面はクエリー文字列の、

```
?next=xx
```

という部分なしでbbs.cgiを呼び出すので、②の$nextは空文字列、すなわち0になります。ですから③では、

```
$start = $cnt - 20 - 0;
```

ということになり、ファイル内の最後の20行が読み込めるのでしたね。では⑧のif文の中を見てください。ここで$nextには20が加えられ、次の行に「前の発言」というリンクがあるのですが、リンク先として、

```
bbs.cgi?next=$next
```

というように指定されています。$nextは値に展開されるので、これは、

```
bbs.cgi?next=20
```

という意味になります。つまり、2画面めにはnext=20が渡されるので、2画面めの③の処理は、

```
$start = $cnt - 20 - 20;
```

になります。最初の画面のときと比べてみてください。$startが20行分手前になるのがわかるでしょうか。その後⑧のところで、$nextにまた20が加えられますから、今度は$nextは40になり、「前の発言」のリンク先は、

```
bbs.cgi?next=40
```

になります。では次の3画面めは？　と考えてゆくと、うまいぐあいに$startが20ずつ減算されて、その結果、画面が進むごとにファイル内の20行手前を読み込んでいくことになるのです。⑧のif文はもうそれ以上古い発言がなくなったときに、「前の発言」というリンクを表示させないためのものです。

あとは⑥の部分ですが、画面をよく見ればわかりますが、現在の画面で表示しているのが何件めから何件めであるかを表示しています。その数値はもちろん計算すればわかるのですが、どう計算すればよいのでしょうか。

$startや$endはそれらしいところを指してはいるのですが、これらは値としてはファイルの先頭からの行数です。たとえばファイル内のデータ数が43行あったとして、最初の画面の表示の場合、画面上の件数表示としては「1～20件め」と出なければならないところが、実際には$startの値は23、$endの値は43となっています。これを「$start件め～$end件め」と表示するわけにはいきません。では**図2**をにらみながら考えてみましょう。

図2によると、最初の画面の場合は1～20件め、2画面めでは21～40件め……となればよいのですから、それはファイルの最後から数えていったときの$startと$endの位置であるということがわかります。ということは、ファイルの最後＝全行数ですから、全行数である$cntから$start、$endの値を引いてやれば最後から数えて何件めかという数字が計算できることになります。ただし、$endは一番最後を指しているときに$cntと同じ値になりますから、$cntから引いただけでは0になりますので、最初を1件めと数えるのなら＋1しておく必要があります。

この計算が⑥の部分です。$start、$endからそれぞれ$start2、$end2という変数を計算しています。ここでしていることは基本的には、

```
$start2 = $cnt − $start;
$end2 = $cnt − $end + 1;
```

という計算なのですが、$startがマイナスの時は例外的に$cntと同じ値にしています。なぜそうしなければいけないかは、ぜひともこのCGIを実行してみて、$startがマイナスの時の処理をしない場合にどういう結果になるのかを画面で確認してください。一目瞭然ですから、どんな説明よりも納得できます。

ところで、「何件めから何件め」という表示は、最初の画面の場合はいつも「1～20件め」で、2画面めは「21～40件め」なのだから、$nextを使って、$next+1件めから$next+20件めという表示をすればいいのじゃないかということに気づいた方もいるかもしれません。そのほうがずっと簡単ではないかと。確かにその通りなのですが、実はその方法にはひとつだけ問題があって、一番古い発言を表示する最後の画面のときで、表示している発言が20件に満たない場合、たとえば全43件のときの3画面めでは、本来は「41～43件め」と表示しなければならないのに、「41～60件め」と表示してしまうのです。

リスト4の方法だと、そのあたりは正しく表示されますのでご安心を。

●

今回の掲示板はあくまでもPerlとCGIを学習するための題材ですから、なんとか実用になるようにと作ってみましたが、本格的な運用にはまだまだ力不足かもしれません。しかし、たったこれだけのスクリプトでも一応掲示板として機能するものが作れるということを実感してもらえたと思います。

[超] 入門シェルスクリプト

今回は文字列の検索や置換の際に使われる正規表現を覚えよう。文字の位置や繰り返しなどを指定するメタ文字や、各種ツールでの検索方法などについて説明したのち、日本語テキスト中の助詞の重複を検出するスクリプトを作成する。

正規表現で検索や置換を行う（前編）

文：大池浩一
Text:Koichi Oike

「正規表現」（Regular Expression）とは、テキストの検索や置換といったマッチ処理に使われるきわめて強力なパターン記述言語だ。日本語では「正規式」や「正則式」、英語のマニュアルなどでは「RE」や「regex」などと表記されることもある。

正規表現のパターンには、通常の文字に加えて、「行の先頭（末尾）」「任意の文字」「列挙した文字のいずれか」「文字（列）の繰り返し」「複数の文字列からの選択」といった情報を埋め込める。この仕組みにより、柔軟な検索や置換処理が可能になるのだ。

優れた特徴を持つ正規表現は、OSを問わずさまざまなプログラムで広く使われている。特にLinuxを含むUNIX系OSには、grep、egrep、exprなどの単機能ツールをはじめ、AWK、Perl、Ruby、sed、Tclといったさまざまなスクリプト言語や、viやEmacsなどのエディタ、さらにはPostgreSQLなどのデータベースに至るまで、正規表現の機能を内蔵したプログラムが数多く存在する。

bashなどのBourne系シェルには、正規表現の機能は内蔵されていない（別系統のシェルであるKornシェルには正規表現によるファイル名マッチ機能が内蔵されている）。しかし、grepやsedなどのツールを利用して、正規表現による検索や置換処理を行うシェルスクリプトは多数存在する。こうしたスクリプトがどのような処理を行っているのか理解するには、シェルに関する知識だけでは不十分で、正規表現についての知識がどうしても必要になる。

正規表現を使って柔軟な検索を行う

使い慣れれば便利な正規表現だが、初心者が習得するのはなかなか難しい。位置、範囲、繰り返し、選択といった情報を表す特殊な記号「メタ文字」をパターン中に埋め込むため、実際のパターンは暗号のような文字と記号の羅列になってしまうからだ。実際、他人が作った複雑なパターンの意味を解釈するのは上級者でも大変だ。

さらに、直前に「¥」を書くかどうかなど、メタ文字の記述方法がプログラムごとに微妙に異なっていたり、あるプログラムで使えるメタ文字が別のプログラムには用意されていないこともある。

今回は、たいていのツールで共通して使える基本的な正規表現と、それを利用したテキストの検索処理について説明する。sedやPerlなどを使ったテキストの置換処理や、シェルスクリプト中で用いられることの多いexprによる分岐処理については来月取り上げる予定だ。

以下では、

- **grep、egrep、Perl、Rubyを利用した検索**
- **基本的なメタ文字の機能と使い方**

などを説明したのち、日本語テキスト中の助詞の重複を検出するスクリプト「checkjyosi」を作成する。

●通常の文字も正規表現の一部

特殊な意味を持つメタ文字について説明する前に、見た目どおりの通常の文字「リテラル文字」について説明しておこう。正規表現のパターンは、リテラル文字とメタ文字の組み合わせで構成される。

ある文字がリテラル文字なのか、それともメタ文字なのかを一言で説明するのは難しい。同じ文字でも、前後の文脈によってリテラル文字になったり、メタ文字になったりすることがあるからだ。一般には、アルファベットと数字はリテラル文字で、記号の一部だけがメタ文字として使われる。詳しくはのちほど説明する。

一方、漢字やひらがな、カタカナなど日本語の文字は、例外なくリテラル文字として扱われる。なお、日本語の文字は、日本語EUCでは2バイト、UTF-8では3バイトで表わされるマルチバイトコードなので、「1文字＝1バイト」を前提とするツールでは、期待される結果が出力されないことがある。日本語を含むテキストを検索、置換する場合は、マルチバイトコードに対応したツールを利用しよう。

さて、正規表現のパターン中に現れるリテラル文字は、その文字1文字にマッチする。たとえば、パターン中の「a」という文字は、検索するテキスト中の「a」1文字にマッチするのだ。正規表現では大文字と小文字は厳密に区別されるので、「A」にはマッチしない。

当然のことながら、リテラル文字が複数並んで文字列を構成すると、テキスト中の同じ文字列にマッチする。たとえば、「Linux」というパターンは、テキスト中の「Linux」にマッチする。

つまり、リテラル文字だけで構成されるパターンの検索結果は、単なるテキスト検索の場合と同じだ。パターンの各部にメタ文字を埋め込むことによって、正規表現ならではの柔軟な検索処理が実現される。

要点をまとめると、

- **正規表現のパターンは、見た目どおりの「リテラル文字」と、特殊な意味を持つ「メタ文字」で構成される。**
- **一般的には、アルファベット、数字、日本語の文字はすべてリテラル文字として扱われる。**
- **パターン中のリテラル文字は、テキスト中の同じ文字1文字にマッチする。**
- **パターン中のリテラル文字の並びは、テキスト中の同じ文字列にマッチする。**

ということになる。

●grep、egrep、Perl、Rubyを使って検索する

正規表現を利用した検索が可能なツールとして、grep、egrep、Perl、Rubyを取り上げ、それぞれの簡単な使い方と日本語への対応について解説する。

(1) grep、egrepを利用した検索

正規表現による検索を行うコマンドとして第一に挙げられるのが、ファイル内の文字列検索を行うgrepとegrepだ。これらのコマンド名に含まれる「re」は、正規表現（Regular Expression）に由来する。

オリジナルのgrepとegrepには、内蔵されている正規表現エンジンの方式などさまざまな違いがあるのだが、Linuxのディストリビューションが採用しているGNU版のgrepとegrepの場合、メタ文字の記述方法が一部異なるだけで機能自体はまったく同じだ。以下では、より自然にメタ文字を記述できるegrepを利用することにする。

egrepの書式は、

```
egrep 'パターン' ファイル名
```

というもので、検索に利用する正規表現のパターンと、検索対象となるファイル名を指定する。ファイル名は複数利用でき、パイプ経由で他のコマンドの出力から検索することも可能だ。なお、パターン中に登場するメタ文字には、「＊」や「｜」などが含まれることがあるため、シェルがこれらをワイルドカードやパイプとして処理しないよう、パターン全体を「 ’」で囲む必要がある。

ファイルやパイプから入力されたテキストは、1行ずつパターンにマッチするかどうか調べられ、マッチする場合は行全体が標準出力（通常は端末画面）に出力される。

たとえば、

```
$ egrep 'Linux' hoge.txt
```

とすると、hoge.txtの内容から、「Linux」という文字列を含む行をすべて検索して端末画面に表示する。

日本語を含むテキストに対して、正規表現によるマッチを正しく行うには「マルチバイトパッチ」と呼ばれる修正が必要だ。GNU grep／egrepの最新版（バージョン2.4.2）用のマルチバイトパッチは、「WILLs' trash can (PROGRAM)」（http://www.hinadori.dyn.to/~wills/program.html）などから入手できる。

Linuxのディストリビューションに含まれるGNU grep

やegrepは、このマルチバイトパッチによる修正が行われていない場合が多いため、入手したファイルを使ってソースファイルにパッチを当て、コンパイルし直すなどして、自分で修正を加えたgrepやegrepを作成する必要がある。それができない場合、日本語テキストはPerlやRubyで検索したほうがいいだろう。

(2) Perlを利用した検索

有名なスクリプト言語Perlは、WebサーバのCGIなどにも広く用いられている。Perlの特徴のひとつは強力な正規表現にあり、他のプログラムには見られない便利なメタ文字が数多く用意されている。ただし、今回はこれらについてはほとんど触れない。

Perlを使ってegrepと同様の検索を実現するには、指定したファイルの内容を1行ずつ読み込んで処理する-nオプションと、スクリプトの内容を直接コマンドラインに記述する-eオプションを使って、

```
perl -ne 'print if /パターン/' ファイル名
```

と書く。

たとえば、egrepで挙げた例と同じ検索は、

```
$ perl -ne 'print if /Linux/' hoge.txt
```

とすれば行える。

正規表現の日本語対応については、Perlのバージョンによって状況が異なる。1つ前の安定版で、現在でも広く使われているバージョン5.005_03には日本語パッチが用意されており、国産ディストリビューションの多くがこのパッチで修正されたPerlを利用している。

```
$ perl -v
```

として、「Japanization patch…」という行が含まれていれば、日本語パッチで修正済みのPerlだ。

ただし、正規表現などを日本語対応にするには、Perlを「jperl」というコマンド名で起動する必要がある。もし、/usr/binなどにjperlが含まれていない場合は、「su -」としてrootになった状態で、

```
# ln -s /usr/bin/perl /usr/bin/jperl
```

として、perlからjperlにシンボリックリンクを張っておく。日本語テキストを検索する場合は、perlの代わりにこのjperlを利用して、

```
$ jperl -ne 'print if /ほげほげ/' hoge.txt
```

などとすればいい。

なお、今回は取り上げないが、Kondara 2.0などに含まれる最新のPerl（バージョン5.6.0）では、ユニコードをサポートしており、UTF-8でコーディングされた日本語テキストをそのまま扱うことができる。

(3) Rubyを利用した検索

国産のオブジェクト指向スクリプト言語として人気の高いRubyは、Perlに匹敵する正規表現クラスライブラリを有している。また、日本語に最初から対応している点も見逃せない。

Rubyを使ってegrepと同様の検索を実現するには、Perlと同じ-nオプションと-eオプションを使って、

```
ruby -ne 'print if /パターン/' ファイル名
```

と書く。これらのオプションには、意図的にPerlと同じ機能が割り当てられている。

たとえば、egrepで挙げた例と同じ検索を行うには、

```
$ ruby -ne 'print if /Linux/' hoge.txt
```

とすればいい。

日本語への対応はPerlの場合よりさらに簡単だ。最初から日本語に対応しているので、パッチによる内容の修正やファイル名の変更は必要なく、起動時に-Kオプションを指定するだけでいい。

-Kオプションのあとには、処理する日本語テキストのコーディングに応じて、「e」（日本語EUC）、「s」（シフトJIS）、「u」（UTF-8）を付ける。「n」（日本語処理なし）を付けると、デフォルトと同じ動作になる。

たとえば、日本語EUCで書かれたテキストを検索する場合は、「-Ke」を付けて、

```
$ ruby -Ke -ne 'print if /ほげほげ/' hoge.txt
```

とすればいい。

●1文字にマッチするメタ文字

以下では、基本的な正規表現の機能と、それを実現するためのメタ文字について順を追って説明する。例として挙げられているパターンを使ってegrepやPerl、Rubyによる検索を行い、ファイルに対する検索結果を確かめながら読むと理解が進むだろう。

シェルのワイルドカード「?」に相当する正規表現のメタ文字は「.」で、テキスト中の任意の1文字にマッチする。たとえば、

```
マ.チ
```

というパターンは、リテラル文字「マ」の次に、メタ文字「.」による任意の1文字、次にリテラル文字「チ」が続く文字列を表す。ありそうな例では「マルチ」や「マッチ」などにマッチする。「.」自体を検索する方法についてはコラムを参照されたい。

マッチする文字を「.」よりも制限したいときには「文字クラス」を使用する。文字クラスは、大カッコ「[」と「]」の間に文字を並べたメタ文字列で、列挙した文字のいずれか1文字にマッチする。たとえば、

```
[0123456789]
```

は、数字1文字、

```
[02468]
```

なら偶数1文字にそれぞれマッチする。

なお、文字コードが連続している部分は、「−」を使って範囲を指定できるので、数字1文字は、

```
[0-9]
```

とパターンを簡略化できる。一方、偶数1文字の例では文字コードが連続していないため「−」は使えない。

また、列挙する文字の先頭に「^」を付けると「否定文字クラス」になり、列挙した文字を除く1文字にマッチするようになる。マッチさせたい文字を直接列挙すると記述が複雑になる場合に使うといい。

たとえば、

```
[^a-zA-Z]
```

というパターンは、数字や記号、漢字、かな文字など、英字以外の1文字にマッチする。

なお、文字クラスの内側と外側とではメタ文字の扱いがまったく異なるので注意されたい。たとえば、「−」は文字クラス内ではメタ文字だが、文字クラス外ではリテラル文字だ。逆に、文字クラス外で使われる「.」などのメタ文字の多くは、文字クラス内では特殊な意味を失ってリテラル文字となる。さらに、「^」のように、文字クラスの内外でまったく意味が異なるメタ文字もある。

この節での要点をまとめると、

- **メタ文字「.」は、任意の1文字にマッチする。**
- **文字クラス「[…]」は、列挙した文字のいずれか1文字、否定文字クラス「[^…]」は、列挙した文字以外の1文字にそれぞれマッチする。**
- **（否定）文字クラスでは、文字コードが連続する部分は「ー」で範囲指定できる。**

ということになる。

●繰り返しを表すメタ文字と最長最左マッチ

電話の市外局番は、「03」「043」「0794」「01452」「082488」といったように、地域によって桁数が異なる。こうした文字列を1つのパターンでまとめて検索するためには、繰り返しを意味するメタ文字をパターンに埋め込む必要がある。

繰り返しを表す代表的なメタ文字には、

+…1回以上の無制限の繰り返し
＊…0回以上の無制限の繰り返し
?…0回または1回の繰り返し

Column

メタ文字として使われる記号自体を検索するには

正規表現を使って、「.」などの記号そのものを検索したい場合はどうすればよいだろうか。一般に、メタ文字を「¥」の直後に書くと、メタ文字としての意味が失われ、リテラル文字として扱われる。このため、「.」自身を検索するならパターン中に「\.」と書けばいい。

逆に、リテラル文字を「¥」に続けて書くと、たいていはリテラル文字のままだが、なかには特別な意味を持つ「メタ文字列」になるものもある。たとえば、単語の境界を表わす「\b」などだ。PerlやRubyにはこの手のメタ文字列が豊富に用意されている。

の3つがある。

さらに、繰り返す回数を細かく指定したいなら、

{m,n}…m回以上n回以下の繰り返し
{m,}…m回以上の繰り返し
{,n}…1回以上n回以下の繰り返し
{m}…m回の繰り返し

といったメタ文字列も使用可能だ。ただし、ツールによってはこれらのメタ文字列を利用できないものもあるので注意しよう。GNU egrep、Perl、Rubyはどれも対応している。

繰り返しという動作を表すこれらのメタ文字（列）は単独では使用できず、直前の文字や文字列にくっついてパターンを形成する。

文字の繰り返しにマッチさせる場合は、その文字の直後に繰り返しを表すメタ文字を書く。たとえば、

```
あう+っ
```

というパターンは、「あ」と「っ」の間に「う」が1回以上繰り返された、「あうっ」「あううっ」「あううううっ」…といった文字列にすべてマッチする。

一方、文字ではなく文字列の繰り返しにマッチさせる場合は、対象となる文字列をカッコ「(」と「)」で囲んでグループ化し、その直後に繰り返しを表すメタ文字を記述する。たとえば、

```
(こく)+
```

というパターンは、「こく」自体が1回以上繰り返された、「こく」「こくこく」「こくこくこく」…といった文字列にすべてマッチする。

それでは、この節の冒頭に挙げた市外局番にマッチするパターンを、繰り返しを表すメタ文字を使って表現してみよう。先頭の0に続けて、任意の数字が1回から5回繰り返されるので、

```
0[0-9]{1,5}
```

と書けばいい。市外局番の2桁めには0はこない（「00」で始まるのは通信業者を選択する番号）といった細かい点に着目すると、さらに複雑なパターンになる。

また、正規表現でよく使われるパターンとして、シェルのワイルドカード「＊」に相当する「.＊」がある。これは「任意の1文字の0回以上の繰り返し」、すなわち（0文字以上の）任意の文字列を意味する。

たとえば、

```
マ.*チ
```

というパターンは、「マ」に続いて、「.＊」がマッチする（0文字以上の）任意の文字列が続き、「チ」で終わる文字列、すなわち「マチ」「マルチ」「マサルのハンカチ」…などにマッチする。

それでは、「マルチがポーチからマッチを取り出した」のように、「マ.＊チ」にマッチする文字列が複数あるテキストの場合、実際にはどの部分にマッチするだろうか。マッチの候補となるのは、テキストの左から順に、「マルチ」「マルチがポーチ」「マルチがポーチからマッチ」「マッチ」の4つだ（**図**）。

ここで覚えてほしいのは、「＊」や「＋」などを使ったパターンが複数の候補にマッチする場合、対象となるテキストのできるだけ左側から始まる候補のうち、できるだけ長い候補とマッチする性質があることだ。これを「最長最左マッチ」と呼ぶ。このため、さきほどの例で実際にマッチするのは、最も左側から始まる候補のうちで最も長い「マルチがポーチからマッチ」になる。

以上をまとめると、

- **メタ文字「＊」は0回以上、「＋」は1回以上、「？」は0回または1回の繰り返しを表す。**
- **メタ文字列「{m,n}」はm回以上n回以下、「{m}」はm回の繰り返しを表す。**
- **文字列の繰り返しは、文字列をカッコで囲んでグループ化し、その直後に繰り返しを表すメタ文字を書く。**
- **「.＊」は、（0文字以上の）任意の文字列にマッチする。**
- **マッチ候補が複数ある場合、できるだけ左側から始まる、できるだけ長い文字列とマッチする性質がある。**

ということになる。

●複数文字列からの選択とマッチ位置の指定

両側のパターンからの選択を意味するメタ文字「｜」を使うと、複数の文字列のいずれかにマッチするパターンを簡単に作成できる。たとえば、

```
リヌクス|リナックス|ライナックス
```

というパターンは、「リヌクス」「リナックス」「ライナックス」のいずれの文字列にもマッチする。いわゆるOR検索を行えるわけだ。

選択の対象を限定するには、カッコ「(」と「)」で囲んでグループ化すればいい。たとえば、

```
(リヌ|リナッ|ライナッ)クス
```

と書くと、「リヌ」「リナッ」「ライナッ」のいずれかに「クス」が続く文字列にマッチする。パターン自体は異なるが、得られる結果は最初の例と同じだ。また、パターンには「+」などの繰り返しを表すメタ文字や、文字クラスなどが含まれていても構わない。その場合、メタ文字「|」は一番最後に解釈される。

このほか、文字そのものではなく、テキスト内での位置にマッチするメタ文字も用意されている。行の先頭を意味する「^」や、末尾を意味する「$」などだ。これらは単独では使えないので、「^」なら直後、「$」なら直前に少なくとも1文字が必要だ。たとえば、

```
^●
```

というパターンは、行頭にリテラル文字「●」がある場合にだけマッチし、

```
です。$
```

というパターンは、行末にリテラル文字列「です。」がある場合にだけマッチする。

このほか、テキスト中の単語（英数字の文字列）の境界にマッチするメタ文字列「¥b」も用意されており、単語単位でのマッチを行いたい場合に重宝する。

たとえば、

```
Linux
```

というパターンは、「Linuxconf」や「TurboLinux」のように、「Linux」を単語の一部に含む場合にもマッチしてしまう。これに対し、「¥b」を取り入れて、

```
¥bLinux¥b
```

というパターンに変更すると、独立した単語の「Linux」にのみマッチするようになる。

要点をまとめると、

- **メタ文字「|」を使って、両側のパターンのいずれかにマッチする選択を実現できる。**
- **メタ文字「^」は行の先頭、「$」は行の末尾という位置にマッチする。**
- **メタ文字列「¥b」は、単語の境界にマッチする。**

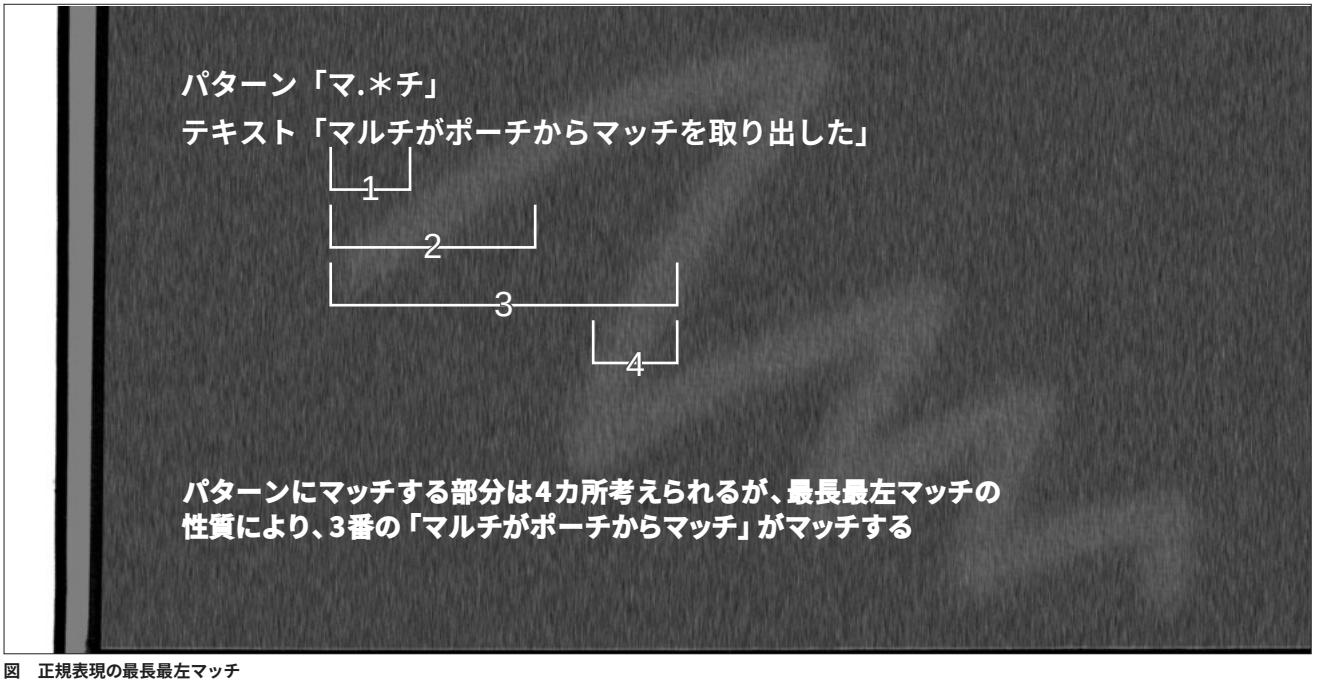


図　正規表現の最長最左マッチ

ということになる。

●前方参照（バックリファレンス）について

正規表現の重要な概念として、パターンの一部分にマッチした文字列を、同じパターンのそれよりあとの部分で指定する「前方参照（バックリファレンス）」がある。

日本語（前方）と英語（バック）がうまくかみ合っていないのは、参照に関する前後のとらえ方が異なるからだ。日本語では「前に読んだ」という意味で、すでに過ぎた部分を「前」と表現するのに対し、英語では読み進む向き（forward）とは逆なので「backward」と表現する。

まずは、パターン中の覚えておきたい部分をカッコ「(」と「)」で囲んでグループ化する。これで、カッコ内のパターンにマッチしたテキスト中の文字列が、専用の記憶領域に格納される。

格納した文字列を呼び出すには、「¥1」「¥2」…というメタ文字列を使う。「¥」に続く数字は、格納の際に使用した開きカッコ「(」が、パターンの左から数えて何番目なのかを意味している。

前方参照を利用したパターンの例として、

```
([ウド]キ)¥1
```

を考えてみよう。まず、カッコ内の「[ウド]キ」というパターンに実際にマッチした文字列が1番目の記憶領域に格納される。続く「¥1」では、格納した文字列が呼び出されてリテラル文字として使われる。

たとえば、カッコ内のパターンが「ウキ」にマッチした場合は「¥1」も「ウキ」、「ドキ」にマッチした場合は「¥1」も「ドキ」となる。つまり、パターン全体としては「ウキウキ」か「ドキドキ」のいずれかにマッチするわけだ。これは、カッコ内のパターンを単に2つ並べた、

```
[ウド]キ[ウド]キ
```

に対するマッチとは異なる。こちらは、「ウキドキ」や「ドキウキ」にもマッチしてしまうからだ。

なお、PerlやRubyなどでは、カッコを使って記憶領域に格納された文字列は、パターンのマッチが終わったあとで置換処理などを行う際にも利用できる。こちらについては次回に説明することにしよう。

これまでの説明に出てきたように、カッコ「(」と「)」を使ったグループ化には、

(a) 繰り返しの際に文字列を対象とする
(b) 選択の際に対象範囲を限定する
(c) 前方参照や後の置換処理のために、カッコ内のパターンにマッチした文字列を記憶領域に格納する

という3通りの使い方がある。

(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	メタ文字の場合はリテラルとして扱う。通常文字の場合はメタ文字列になるものと、そのままリテラルとして扱われるものがある
[…]	列挙した文字の1文字にマッチ（文字クラス）
[^…]	列挙した文字を除く1文字にマッチ（否定文字クラス）
(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（m、nの一方は省略可）
{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
(…)	選択範囲を変更する
(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
(…)	カッコ内の部分パターンにマッチした文字列を記憶領域に格納する

表1　egrep、Perl、Rubyで使える基本的なメタ文字（列）

egrepはこれらをいっさい区別しないので、(a)や(b)のためだけにカッコを使った場合でも部分パターンにマッチした文字列の格納処理が行われる。その結果、前方参照で使用する「¥」に続く数字が飛び飛びになってわかりにくくなるうえ、扱うテキストが大規模になると格納にかかるメモリやCPUの負担も馬鹿にならない。このため、PerlやRubyには格納を行わない開きカッコ文字列「(:?」が用意されている。

最後に、これまでに登場した正規表現のメタ文字（列）を**表1**にまとめておく。

今月のスクリプト

後半は、与えられたテーマを実現するスクリプトの作成手順を説明する。今月は、Rubyの正規表現による検索機能を利用して、

日本語テキスト中の1文字の助詞の重複をチェックするスクリプト「checkjyosi」

を作成する。

日本語の助詞は、述語に対する関係を表す「格助詞」、主題の提示などを行う「副助詞」、語や節を接続する「接続助詞」、文末に現れる「終助詞」など、数多く存在する。その中でも、「が」や「を」といった1文字の助詞は、テキスト入力時の打ち間違いや、編集作業などによってしばしば重複することがあり、見逃しやすい誤りのひとつだ。こうした助詞の入力ミスをチェックするスクリプトを作成するわけだ。

たとえば、Vine Linuxに付属する「JF」のドキュメントを今回作成したスクリプトでチェックしたところ、273個のファイルに対し、明らかに誤りとわかる助詞の重複が50個近く見つかった。

1文字だけで構成される助詞は、「が」「を」「に」「と」「で」「へ」「の」「は」「も」「か」「さ」「ね」の12個ある。このうち、「と」「で」「の」「も」「さ」「か」については、実際にチェックを行ったところ、「こととする」や「〜でできる」、「するものの」といったように、連続していても助詞の重複ではないケースが多いため、今回の検索対象からは除外することにした。

さて、残りの6文字「が」「を」「に」「へ」「は」「ね」のいずれか1文字にマッチするパターンは、文字クラスを使えば、

```
[がをにへはね]
```

と書くことができる。

同じ文字が2回続くことを表現するには、文字クラスをカッコで囲んでグループ化し、マッチした部分を前方参照する「¥1」をそのあとに書けばいい。

```
([がをにへはね])¥1
```

さらに、めったにないだろうが、同じ文字が3個以上続くような場合も一応考慮して、1回以上の繰り返しを表すメタ文字「＋」を付け、

```
([がをにへはね])¥1+
```

とすればパターンは完成だ。

これをRubyによる検索のパターンとして指定し、

```
$ ruby -Ke -ne 'print if /([がをにへはね])¥1+/'
hoge.txt
```

とすると、hoge.txtの内容のうち、助詞が重複している行がすべて出力される。

こうしたパターンを毎回記述するのは面倒なので、シェルスクリプトを作成しよう（**リスト1**）。単にRubyを起動するだけではなく、コマンドライン引数を1つも指定しなかった場合は、使い方（usage）を標準エラー出力に表示してスクリプトを終了するようにしている。なお、他のコマンドの出力などをパイプ経由で読み込む場合は、標準入力を意味するファイル名「−」を指定すればいい。

今回作成したスクリプトの出力は、注意深く見ないとどの部分が重複しているのかわからない。それほど助詞の重複は見逃しやすいのだ。そこで、来月号の本連載では、sedやPerl、Rubyなどで正規表現を利用した置換処理を行うことで、マッチした部分が目立つように工夫してみることにしよう。

▼リスト1　日本語テキスト中の1文字の助詞の重複をチェックするスクリプトcheckjyosi

```
1: #!/bin/sh
2: if [ $# = 0 ]; then
3:   echo "usage: checkjyosi file..." > /dev/stderr
4:   exit 1
5: fi
6: ruby -Ke -ne 'print if /([がをにへはね])¥1+/' "$@"
```

彼方からの手紙

— IMAP4メール環境で遊ぶ —

第2回　メッセージ入りのデジタル付箋紙を画面に貼り付ける

文：塩田紳二
Text : Shinji Shioda
Program: Ryo Shimizu
illustration : M.T

さて今回より、実際にPerlのスクリプトを使い、IMAP4サーバのアクセスなどをやってみることにいたします。とりあえずは、サーバにあるメールを表示させることにしましょう。前回、IMAP4の解説などいたしましたが、世の中は便利なもの、こういうときのためのライブラリというものがございます。このようなものを使うことで、プログラマーは、IMAP4の細かいプロトコルを考えることなく、メインテーマのアルゴリズムの実現に専念できるわけでございます。

IMAP4をアクセスする前に

IMAP4は比較的簡単なプロトコルではありますが、メッセージを扱うスクリプトではやはり細かな制御が必要になります。これは、IMAP4自体がさまざまなメッセージの取り扱いを想定したものになっている以上、しかたのないことです。

逆に、メッセージを扱う部分をパターン化することも可能です。こうしたところに、いわゆる「ライブラリ」というものが作られるわけです。Perlのライブラリは、Perlスタイルのデータの取り扱いになじむようにIMAP4などのプロトコルを包み込み、プロトコル自体はあまり見えないようになっています。

スクリプトプログラミングは、ある意味「お手軽」なところがなくてはいけません。ここで真面目にソケットを使ってプログラミングするのもいいのですが、手を抜けるところはどんどん抜くべきでしょう。これはPerlでなくとも、ほとんどのスクリプト、いや、ほとんどのプログラミングに共通することです。手抜きなくしてソフトウェアの進歩はないとさえいえます（ちょっとおおげさか？）。

前回でも紹介したように、実際にスクリプトを作成したのはプログラマーS氏ですが、彼は最初、IMAP4のプロトコルが簡単なのでソケットを使って真面目にIMAP4を使っていました。しかし、“Mail:IMAPClient”というライブラリを見つけてプログラムを書き直したあとは、メールで「# 最初からこっち使えばよかった…^^;」と語っております。実際、その効果はかなり大きなものです。270行のスクリプトが110行程度、つまり半分以下になったのですから。最初からこれを使っていれば、半分以下の手間でプログラムが完成していたわけです。

さて前回では、この連載の前提となる環境についても解説しましたが、この連載のスクリプトを試すにはPerlが必要です。といっても、Linuxの多くのディストリビューションでは、Perlのパッケージをインストールするだけなので、なんの問題もないと思います。ここでは、簡単にPerlのモジュールとライブラリをインストールする一般的な手順を解説しておくことにしましょう。

Perlでは、さまざまなライブラリが「モジュール」という形で提供されます。これは、素のままでなく、インストール（場合によってはコンパイル作業など）の手順を記述したMakefileやドキュメントなどとともに、tar＋gzipなどのアーカイブファイルで提供されます。

多くの場合、モジュールのインストールは、アーカイブを展開したディレクトリで、

```
perl Makefile.PL
make
make install
```

という手順で行います。これで、目当てのモジュールが適切なディレクトリにコピーされ、そのオンラインマニュアルなどがインストールされます。実際には、附属のドキュメント（README とかINSTALL とか）を読んで、そこに書いてあるとおりにしなければならないのですが、一般的な手順を知っていれば迷うこともないでしょう。特に、初めてモジュールをインストールするような場合には。

筆者も、Perl プログラマーではないので、最初はどうやっていいのかちょっととまどいました。なにせ、『初めてのPerl』という本を読んだだけなので……。そういえば、『初めてのC』という本が出たとき、中山美穂の「C」という曲が流行ったあとで、なにか非常に赤面したことを思い出しますが、『初めてのPerl』ってのもなんかへんな感じ。衣料用洗剤ブルーダイヤの「金銀パールプレゼント」で当たったみたいで……。

というわけで、こういうふうにインストールしてください。それから、Perl のモジュールなどはCPAN（http://www.cpan.org/）から入手します。とりあえず、Perl関係はここを探すのが基本中の基本（らしい）です。

IMAP4からメッセージを取り出してみよう

という経過をたどりつつ、「有明逆ピラミッドの夏の陣」（おそらく、コミケというやつだったのではないでしょうか）で忙しかったS氏が作り上げたのが、**リスト1**のプログラムです。

これは、メールボックスのなかから指定したメッセージ

▼リスト1

```
#!/usr/bin/env perl
#
# imapcat - cat messages with imap
#
# $Revision: 1.10 $
#
;$debug = 0;

use Mail::IMAPClient;

die "Usage: imapcat [user[:password]@]host
[Mailbox/][sequence]\n" if ($#ARGV<0);

#
# コマンドライン文字列の解析
#
$host = shift if ($#ARGV >= 0);  # imap server ―― ①

▼ユーザ名@ホスト名の形式ならば分割
($user,$host) = ($host =~ m/([^@]*)@(.*)/) if ($host =~
m/\@/);
▼パスワード付きならさらに分割
($user,$password) = split(/:/,$user,2) if ($user =~
m/:/);

▼ユーザ指定がなければ login名から取得
($user = getlogin || (getpwuid($<))[0]) if ($user eq
'');
▼パスワードがなければ ~/.imapcatrc から取得
$password = &getconfig("$user\@$host") if ($password eq
'');
▼それもなければconsoleから取得
$password = &getpass if ($password eq '');
▼mailの指定
$sequence = shift if ($#ARGV >= 0);
▼デフォルトのメールボックス名
$mailbox = "INBOX";
▼指定があればそのメールボックス名を使う
($mailbox,$sequence) = ($sequence =~ m|(.*)/(\d*)$|) if
($sequence =~ m|/|);

▼imapサーバに接続
$imap = Mail::IMAPClient->new( ―― ②
        Server            => "$host",
        User              => "$user",
        Password => "$password",
        Port              => 143,
        Debug             => "$debug",
        Clear             => '1',
)
|| die ("cannot connect to $user\@$host\n");

$imap->Uid(1);
▼メールボックスを選ぶ
$imap->select($mailbox); ―― ③

▼リスト表示 or メール内容表示?
if ($sequence ne '') { ―― ④
        ▼メッセージの中身を取得
        $msg = $imap->fetch($sequence,"RFC822");
        ▼それを整形して表示
        print join("\n",&fetch2msg($msg));
} else { ―― ⑤
        ▼すべてのメッセージのリストを取得
```

を取り出すもので、今後メッセージを扱うときの基本となるべき「コマンド」です。

この連載では、主題となるアルゴリズムをわかりやすくするためと、のちのさまざまな応用を考えて、まず最初に重要な機能を実現した「コマンドラインから利用するスクリプト」を作ることにします。これを使えば、すぐにktermなどのコマンドラインから動作を確認できます。というわけで、作ったのがimapcatです。ちなみに、このプログラムの命名はS氏。たぶん、IMAP用のcatという意味だと思います（まさか、IMAP用のネコってわけではないでしょう）。

まず、IMAP4サーバへのアクセス開始は、**リスト1**の②のところから始まります。簡単にいえば、Mail:IMAPClientを使ってIMAP4サーバへの接続を行う部分です。言い方を換えると、Mail:IMAPClientオブジェクトのインスタンスを生成するところでもあります。余談ですが、Perlのオブジェクト指向は、かなり独特なものです。もともと手続き型言語であったものにオブジェクト指向（のようなもの）を導入したため、見かけ上はオブジェクト指向になっていますが、解説書（『プログラミングPerl』Larry Wall他著、オライリー・ジャパン刊）などによれば、「クラスはパッケージにすぎない」、「メソッドはサブルーチンにすぎない」、「オブジェクトはリファレンスで指された『物』にすぎない」などと書かれており、オブジェクト指向的にプログラムを解釈することもできるけど、単なるライブラリ呼び出しと解釈することも可能です。リストを見る場合には、適当に自分で理解しやすいほうに解釈してください。

ここでは、サーバ名やユーザー名、パスワードなどを指定して、IMAP4のプロトコルで接続を開始させています。これが成功すれば、以後はMail:IMAPClientのメソッド（あるいはサブルーチン）が呼び出し可能になります。

IMAP4でメッセージをアクセスするには、

▼リスト1

```
        @mails = $imap->search("ALL");
        ▼ふつうに表示
        print join(" ",@mails),"\n";
}

$imap->close();  ⑥
$imap->logout();

# imap->fetch で返ってきたメッセージからMailbox形式の
# 中身を取り出す（imapの余計なヘッダが付いているので）
sub fetch2msg
{
        my ($msg) = shift;
        grep(/: /,@$msg);
}

# consoleからパスワードを入力させる。
# （echoなしにするためにsttyコマンドを使う）
sub getpass  ⑦
{
        my ($stty);

        if (-t STDIN) {
                ▼現在の状態を取得
                chop($stty = `stty -g`);
                ▼echoなし、cbreakモード
                `stty -echo cbreak`;
        }

        print STDERR "Password: ";
        chop($password = <STDIN>);
        print STDERR "\n";
        ▼状態を戻す
        `stty $stty` if (-t STDIN);


        $password;
}

# ~/.imapcatrc から password を得る。
# ~/.imapcatrc のフォーマットは、
#
#     username@hostname <TAB> password
#           :
# とする。コマンドラインで指定された username@hostname ## に一
致する行を探して、それをpasswordとして返す
sub getconfig
{
        my ($userhost) = @_;
        my ($result);

        open(IMAPCATRC,(getpwuid($<))[7] .
"/.imapcatrc");
        while (<IMAPCATRC>){
                if ($userhost eq
(split(/\s+/,$_,2))[0]) {
                        chop($result =
(split(/\s+/,$_,2))[1]);
                }
        }
        close(IMAPCATRC);
        $result;
}
```

1.ログインを行う（LOGIN）
2.メールボックスを選択する（SELECT）
3.メッセージのUIDなどを指定してメッセージを取り出す（FETCH）
4.メールボックスの選択を解除する（CLOSE）
5.ログアウトする（LOGOUT）

という手順になります。これは、Mail:IMAPClientでは、

1.newで接続開始
2.selectでメールボックスを選択
3.fetchでメッセージ取得
4.closeで選択終了
5.logoutでログアウト

となります。基本的には、IMAP4のコマンドを使うようにメソッド（サブルーチン）を呼べばいいわけです。

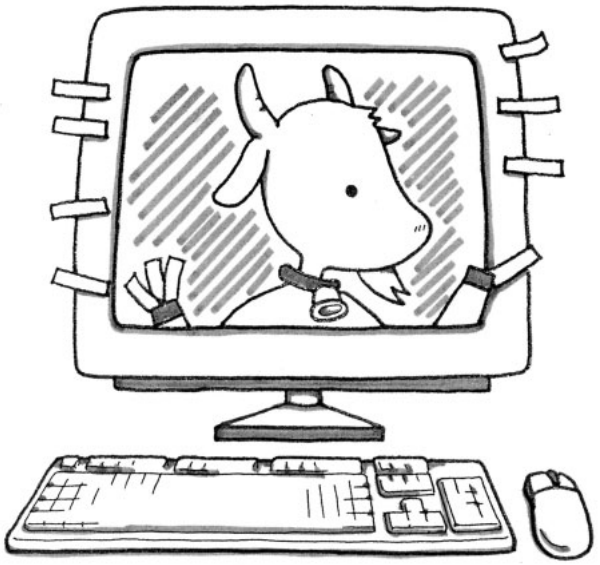
IMAP4を操作したあとは、終了手続きをきちんとしておく必要があります。というのも、メールの削除などはコマンド（DELETEコマンド）が発行された時点でマークだけつけておいて、メールボックスがクローズされるときに実際に削除するからです。今回の例では、メールボックス内のメッセージに対して削除などの操作をせずに参照しているだけなので、どう終わっても問題はないのですが、メッセージを操作する場合には、正しい手順で終了させないと操作が反映されません。

メールボックスの選択は、**リスト1**の③で行っています。これは、単にメールボックス名を引数として渡しているだけです。

これでメールボックスをアクセスする準備が整いました。次にやることは、メッセージリストの表示（UIDの取得）かメッセージの取り出しです。このimapcatは、UIDが指定されない場合には、単に該当メールボックスのUIDをプリントするだけです。

UIDが指定されれば、該当のメッセージをfetchで取り出します（④）。IMAP4のFETCHコマンドは、さまざまな形でメッセージを取り出すことができます。たとえば、ヘッダのみ、本文のみ、といった具合です。実際、IMAP4サーバにはMIMEを解析する能力があり、メッセージ全部を取り出さずに添付ファイルだけ取り出す、といったことも可能です。ここでは、RFC822形式でメッセージをそのまま取り出しています。

最後は、フォルダの選択をやめ、logoutします（⑥）。この時点で、サーバがセッションを切断するはずなので、以後のことは考える必要はありません。



imapcat全体の構成

imapcatは、以下のような書式で利用します。

```
imapcat　[ユーザー名[：パスワード]@]ホスト名　[メールボックス名/][シーケンス]
```

ただし、いくつかのパラメータは省略が可能です。まず、ユーザー名を省略すると、現在のユーザー名が利用されます。メールボックス名を省略すると、“INBOX”が使われます。シーケンスにUIDを指定するとメッセージが表示され、省略するとUIDのリストが表示されます。

パスワードを省略した場合、まずユーザーのホームディレクトリにある.imapcatrcからパスワードを探します。しかし、このファイルがない場合には、標準入力からパスワードを受け取ります。このため、

```
echo パスワード | imapcat ……
```

とすることも可能です。

.imapcatrcは、以下のような形式になっているテキスト

ファイルです（“〈TAB〉”はタブコードを表す）。

```
ユーザー名@ホスト名〈TAB〉パスワード
```

このファイルを作っておけば、パスワード自体を直接指定する必要がないので、できるだけこの方法を使うようにしておいてください。なお、セキュリティに配慮をして、このファイルは自分だけが読めるようにパーミッションを設定しておいてください。

基本的にimapcatは、「引数の処理」、「IMAP4によるメッセージの取得」、「パスワード処理」の3つの部分から構成されています。引数の処理はスクリプトの前半部分（**リスト1**の①から）で、ここでIMAP4接続に必要なパラメータを抽出したら、あとはさきほど解説した手順でメッセージを取り出すだけです。なお、リスト1の後ろのほう（⑦よりも後ろ）は、メッセージの整形とパスワード処理のためのサブルーチンです。

メッセージをWebブラウザで表示させる

さて、IMAPサーバからメッセージを取り出せるようになったら、これを表示させてみることにしましょう。目標は、メッセージを付箋紙のように画面に貼り付けることです。単純にテキストにして画面に置くだけでは芸がないので、ここはHTTPサーバを使って、どこからでも付箋紙を見ることができるようにします。ただ、通常のWebブラウザのウィンドウは、メニューバーやツールバーなどが煩わしく、とても「付箋紙」という雰囲気ではありません。そこで、タイトルバーのみの単純なウィンドウにしたいところですが、これがけっこう面倒な仕事。今回は、JavaScriptを使って解決することにしました。

実際には、imapcatの出力であるテキストのメッセージを、JavaScriptを含むHTMLページに変換します。このために、mail2jsというプログラムを作りました。誌面の関係で、mail2jsのスクリプトの掲載は省きます（付録CD-ROMのDisc2およびLinux magazineサイトからダウンロードが可能です）。

mail2jsの出力は、**リスト2**のようになります。mail2jsが生成するものはHTMLファイルですが、その中にはメッセージ入りのウィンドウを描画するJavaScriptが入っています。このmail2jsの出力をWebブラウザで表示させれば、別途、タイトルバーのみのシンプルなウィンドウが表示され、そこにメッセージが表示されます。mail2jsは、

```
imapcat ユーザー名@サーバ名 メールボックス名/UID | mail2js -w 幅 -h 高さ -f フォントサイズ
```

のように利用します。幅と高さには、ウィンドウの幅と高さをピクセル単位で指定します。また、フォントサイズは表示するメッセージの文字の大きさで、1～7の数字で指定します。これは、FONTタグのSIZEで指定する数値となります。

とにかく、上記のようなコマンドラインで、小さなウィンドウにメッセージを表示するHTMLが生成できます。あとは、これをWebから扱えるようにするだけです。そのために、ここではHTTPサーバが持つCGIを使うことにして、**リスト3**のshow.cgiというCGIスクリプト（もちろんPerlで記述してあります）を用意してみました。この

▼リスト2　mail2jsの出力

```
<HTML><HEAD><TITLE>mymail</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
mailwin = window.open("","mymail","width=300,height=300,scrollbars,resizable,top=900,left=1100");
mailwin.document.open();
mailwin.document.writeln("<HTML><HEAD><TITLE>ASCII24 Mail Service 2001-09-14");
mailwin.document.writeln("</TITLE><META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text/html; charset=x-sjis\"></HEAD>");
mailwin.document.writeln("<BODY>");
mailwin.document.writeln("<FONT SIZE=3>From: ASCII24 Mail Service &lt;info@ascii24.com&gt;<br>");
mailwin.document.writeln("Date: Fri, 14 Sep 2001 03:41:13 +0900 (JST)<br>");
mailwin.document.writeln("Subject: ASCII24 Mail Service 2001-09-14<br>");
mailwin.document.writeln("<br>");
mailwin.document.writeln("<br>");
mailwin.document.writeln("□□□□□□□□□□　ASCII24電子メールサービス 2001-09-14　□□□□□□□□□<br>");
```

CGIでは、以下の変数を受け付けます。

server　IMAPサーバ名
user　ユーザー名
pass　パスワード（.imapcatrcがあれば省略可能）
mbox　メールボックス名
seq　表示したいメッセージのUID
width　ウィンドウの幅（ピクセル）
height　ウィンドウの高さ（ピクセル）
font　表示フォントのサイズ（1～7）

となっています。簡単にいえば、最初の5つがimapcatの引数で、後ろの3つがmail2jsの引数です。

たとえば、**リスト4**のshow.htmlのようなHTMLを作れば、フォームで指定したメッセージを表示させることができます。実際にLinux上のNetscapeで表示させてみたのが画面1です。もちろん、クライアントがWindowsのInternet Explorerであってもちゃんと動作します。

あるいはもうひとつの方法として、

```
imapcat 引数 | mail2js -i 引数 > mymail.html
```

などとしてメッセージをHTML化してしまい、サーバサイドインクルード（SSI）などで表示させることもできま

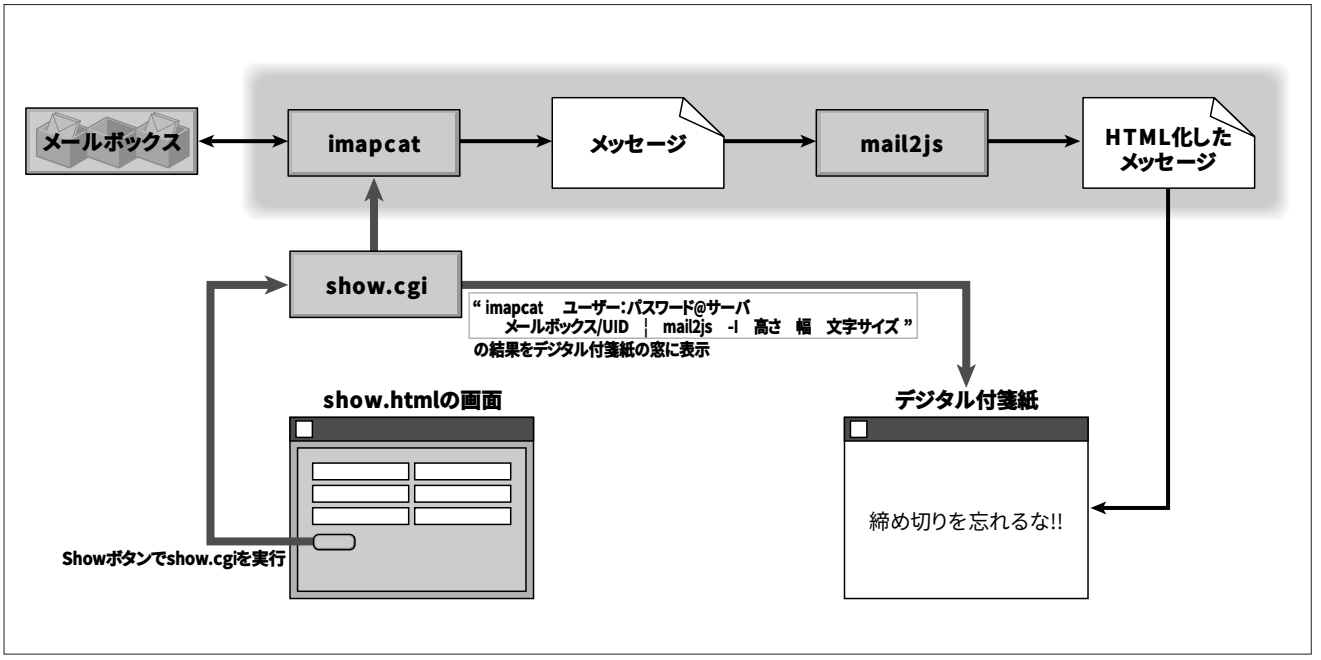


図1　デジタル付箋紙が画面に貼り付けられるまでのしくみ

▼リスト3　show.cgi

```
#!/usr/bin/perl

print "Content-Type: text/html\r\n\r\n";

read(STDIN,$_,$ENV{'CONTENT_LENGTH'});
for (split(/&/,$_) ) {
        ($name,$value) = split(/=/,$_);
        $args{$name} = $value;
}

$server = $args{'server'};
$user   = $args{'user'};
$pass   = $args{'pass'};
$mbox   = $args{'mbox'};
$seq    = $args{'seq'};
$height = $args{'height'};
$width  = $args{'width'};
$font   = $args{'font'};

$h = "-h $height" if ($height+0);
$w = "-w $width"  if ($width+0);
$f = "-f $font"   if ($font+0);

open(IMAPCAT,"imapcat '$user:$pass\@$server'
'$mbox/$seq'|mail2js -i $h $w $f|");

print "<HTML><HEAD><TITLE>show
mail</TITLE></HEAD><BODY>\n";
print "display $mbox/$seq on $user\@$server<br>\n";
print <IMAPCAT>;
print "</BODY></HTML>\n";
```

す。ただし、この場合、ApacheでSSIが使えるように設定しておかねばなりません。

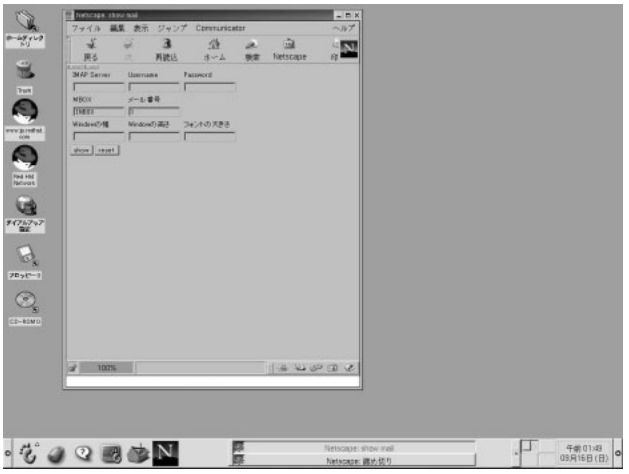
ユーザーのホームページでCGIを使う

なお、このshow.cgiを使うには、ひとつ制限があります。パスワードをユーザーのホームディレクトリから読み込むようにするには、suExecと呼ばれる機能をApacheに組み込み、このCGIをユーザーのホームディレクトリにあるpublic_html以下に置かねばなりません。

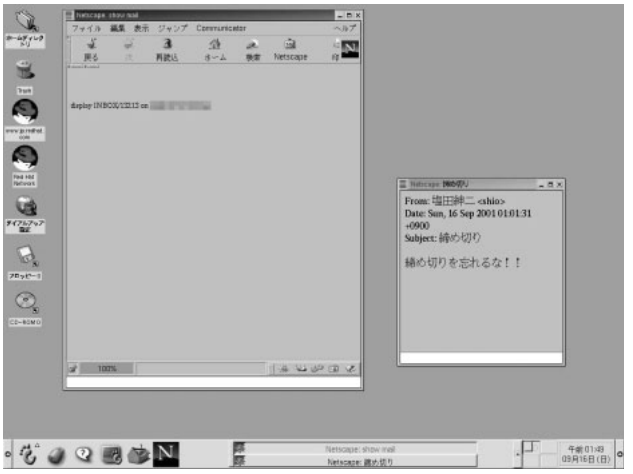
Webブラウザからのアクセスに対して、httpサーバはnobodyというユーザーとして動作します。しかしこれでは、imapcatはimapcatrcからパスワードを読み込むことができません（もちろん、直接パスワードを指定するのな

▼リスト4　show.html

```
<HTML>
<HEAD>
<TITLE>メールの表示</TITLE>
<META http-equiv="Content-Type" content="text/html; charset=iso-2022-jp">
<BODY>

<FORM method=post action="show.cgi">
<TABLE>
        <TR>
                <TD>IMAP Server</TD>
                <TD>Username</TD>
                <TD>Password</TD>
        </TR>
        <TR>
                <TD><INPUT type=text     name="server"  size=12 maxlength=64></TD>
                <TD><INPUT type=text     name="user"    size=12 maxlength=64></TD>
                <TD><INPUT type=password name="pass"    size=12 maxlength=64></TD>
        </TR>

        <TR>
                <TD>MBOX</TD>
                <TD>メール番号</TD>
                <TD></TD>
        </TR>
        <TR>
                <TD><INPUT type=text     name="mbox"    size=12 maxlength=64 value="INBOX"></TD>
                <TD><INPUT type=text     name="seq"     size=12 maxlength=64 value="1"></TD>
                <TD></TD>
        </TR>

        <TR>
                <TD>Windowの幅</TD>
                <TD>Windowの高さ</TD>
                <TD>フォントの大きさ</TD>
        </TR>
        <TR>
                <TD><INPUT type=text     name="width"   size=12 maxlength=64></TD>
                <TD><INPUT type=text     name="height"  size=12 maxlength=64></TD>
                <TD><INPUT type=text     name="font"    size=12 maxlength=64></TD>
        </TR>

</TABLE>
<INPUT type=submit value="show">
<INPUT type=reset  value="reset">
</FORM>
</BODY>
</HTML>
```



画面1　show.htmlの表示



画面2　show.cgiの表示

ら問題はありません)。このため、Apacheのユーザーごとの URL（http://SERVER/~USER）からアクセスしてきたユーザーを判別し、suExecを使ってそのユーザーとしてCGIを起動する必要があるのです。

この設定ですが、Apacheはデフォルトで各ユーザーごとのホームページ機能を働かせているはずです。httpd.conf（/etc/httpd/confなどにある）に、

```
UserDir  public_html
```

という行があればOKです。次にsuExecですが、おそらくApacheとともにインストールされていて、たとえば/usr/sbinなどに置かれているはずです。

```
rpm -ql apache | egrep suexec
```

などで確認してください。Apacheを起動したとき、error_logファイル（/etc/httpd/logsなどにある）に、

```
[notice] suEXEC mechanism enabled
```

と表示されればOKです。起動しない場合、おそらくsuexecコマンドのパーミッションが問題と思われます。

```
chmod 4711 /usr/sbin/suexec
```

としてsticky bitを立てておきましょう。そのあと、Apacheを再起動すれば動くはずです。

次にやるべきことは、~/public_html以下に置いたCGIを起動できるようにすることです。これには、

```
<directory /home/*/public_html>
 Options +ExecCGI +Includes
</Directory>
```

などという設定をhttpd.confに入れ、CGIの実行を許可しておきます（ちなみに上記の設定で、サーバサイドインクルードも許可されます)。また、拡張子.cgiがCGI実行ファイルであることを認識させるには、

```
AddHandler  cgi-script  .cgi
```

という行が必要です。最近のApacheでは、この行の行頭に“＃”を付けて注釈にしていることが多いので、こちらも修正しておきます。

全部の修正が終わったら、Apacheを再起動させます。

◆

というわけで、メッセージをウィンドウに表示させることができました。表示部分にWebを使ったのは、どのマシンからでも見られるようにしたかったから、というだけの理由です。すべてLinuxですませたいのなら、ほかの方法（たとえばTkを使うとか）もあったのですが、筆者はこのほうが原稿を書くのがラクだから、というのが真相です（なにせ、Windowsマシンで原稿を書いているので)。さて、出来上がったスクリプトは、実用で使うにはいろいろと行き届かないところがあります。UIDでメッセージを指定するのは現実的ではありませんし、フォルダからメッセージを消してもデジタル付箋紙は消えないようにしたいもの。というわけで、次回はこのスクリプトをさらに発展させて、ちゃんとした付箋紙アプリケーションを完成させる予定です。

Linux日記

第26回　メール配送（13）

前回はPOPサーバの話をしました。今回は、IMAPサーバと電子メールのセキュリティについて説明します。

文：榊 正憲
Text : Masanori Sakaki

illustration ; Aki

もう秋口なのだが、夏前から妙に忙しい状態が続いている。忙しいと現実逃避をしたくなるのは人間の常である（決して筆者だけではないはず）。忙しい時期が長引いたため、現実逃避も充実（？）したものとなり、6石スーパーラジオの組み立てだけでは済まなかった。しばらく前にアナログシンセサイザの話をして、そのときにいつかハモンドオルガンも欲しいと書いたのだが、トーンホイールをモデリングした音源を使っているKORGのコンボオルガンをついふらふらと買ってしまった（さすがに本物は価格も置き場所も辛すぎる）。

古いハモンドオルガンは、トーンホイールという音源を使っている。一定速度で回転する金属歯車のそばに、エレキギターなどで使っているのと同じようなピックアップを置くと、通過する歯の数に応じた周波数の交流が発生する。トーンホイールオルガンには、この歯車とピックアップのセットがたくさん（少なくともキーの数以上）並んでいる。これで、押したキーに応じて音を出すことができる。さらに、音に厚みを加えるために、いろいろなバランスで倍音を混ぜることができるのである。典型的な、倍音加算形式の音源である。さらにハモンドオルガンには、レズリーというスピーカを接続するのがお約束となっている。これはキャビネットの中でスピーカが回転するという構造になっていて、音に揺れや回転感が加わる。

筆者が購入したオルガンは、こういったトーンホイールオルガンと回転スピーカの構造を、デジタル的にモデル化し、DSPでリアルタイムにオーディオ信号を合成している。本物の音をサンプリングしたのではなく、本物の構造をモデル化しているのである。ご丁寧に、近接したトーンホイールからのクロストーク（要するにノイズ）や、回転スピーカに対するマイクの位置などまでモデリングしており、パラメータとして調整できるようになっている。いわばハイテクでローテクを実現しているのである。

ソフトウェアで実現している仕組みは古くても、現在の楽器である。ちゃんとMIDIは備えている。そこで、いわゆるDTMソフトというのを買ってみた。ノートPCにインストールすれば、オルガンの所に持っていって使えるかなと思ったのだが、ちとノートには荷

Column

今の楽器

久々に楽器のカタログを眺めたり、メーカーのサイトを見たりしたのだが、ずいぶん驚かされた。ソフトウェアによるモデリング音源などはあたり前だし、オプションにSCSIインターフェイスや外付けHDD、MO、CD-Rなどがある。本体のメモリやDSPをSIMMモジュールで増設したりなどなど。

今のソフトウェアや半導体の技術を考えれば不思議でもなんでもないと頭ではわかっても、実際に目の当たりにすると、やはりびっくりである。

が重いようで、ソフトウェアシンセサイザやウェーブ音源まで組み合わせると処理能力が不足し、まともに動かないということがわかった。単なるMIDIシーケンサとしてなら使えるのだが。とりあえずパワーのあるデスクトップ機にインストールしたが、肝心のオルガンにケーブルが届かないという情けない状況になっている。

■お盆なので

病院でチューブだらけにされた状態で死ぬというのは勘弁してもらいたいというのが多くの人の希望だろう。できれば、普通に生活しているまま、ある朝見たら、本人すら気付くことなく、苦しまずに安らかに死んでいたというのは、ひとつの理想型かもしれない。しかし、相手が現役ばりばりのコンピュータともなると、「それはよい最期でしたね」などといってもいられない。というわけで、お盆の頃に、人間なら理想的ともいえる最期をとげたマシンを彼岸から連れ戻す作業を行った。

ずっと懸案事項だった、スリープしたまま死んでしまったPentium Ⅲ×2マシンの復活に向けて作業を始めた。主電源を入れるとファンだけ回り、電源スイッチは効かないという症状なので、電源かマザーボードの故障だろう。マザーボード交換だと、デュアルだから高いよなと思いつつ、まず電源をほかのマシンのものにつなぎ換えてみた（この辺は、お盆の行事というよりはフランケンシュタイン博士的である）。幸い、これで電源スイッチが効くようになり、LEDが点灯、起動メッセージが表示され、BIOS Setupまではたどり着くようになった。

試験に使った電源はケーブルが短く、ディスクドライブに届かないので、OSの起動までは確認できなかったが、たぶんこれでOKとして、秋葉原に出かけ、適当な電源を仕入れ、無事に復活させることができた。今まで、ファンのベアリングが固まって電源が故障したというパターンは何度か経験しているが、ファンが動いているまま電源が壊れたのは初めてである。

さて、前回はsendmailから少し離れて、POPの話をした。

POPサーバを運用することで、随時稼働のクライアントが常時稼働のSMTPサーバからメールを受信できるようになる。また、SMTPサーバ、POPサーバは、ユーザーの組織で運用する必要はなく、プロバイダ側で稼働していればよいので、メールシステムのアウトソーシングが可能になる。

しかし、POPはクライアントが1台であることを前提にしているので、会社のデスクトップPCと持ち歩いているノートPCを組み合わせて使っているというユーザーには不便である。こういった問題を解決するのがIMAPである。

■IMAP

複数のMUAからメールにアクセスしたいという要望に応えて作られたのが、IMAPである。POPサーバが受信メールのダウンロード用ツールという位置付けであるのに対して、IMAPサーバは、メールサーバホスト上で動作するローカルMUA（ucbMailなど）に近い動作をする。IMAPサーバの基本動作は、MTAによりスプールされた受信メールを、そのサーバコンピュータ上で保存、管理することである。これらの操作は、ネットワークで接続されたMUAクライアントからリモートで行われる。そして操作の一環として、受信メールの閲覧／クライアントへのダウンロードが行われるのである。

IMAPも、POPと同じようにまずユーザー認証を行う。そして、システムメールボックス（MTAが受信したメールをスプールするメールボックス）の中にあるメールの一覧、特定のメールのダウンロードなどを行うことができる。ここまではPOPと同様である。しかしIMAPの機能はこれだけでなく、ユーザーのプライベートなメールボックスをサーバホスト上に作成し、管理できるのである。たとえば、現在やっている仕事ごとのメールボックス、さまざまなプライベートなメールボックスなどをホスト上に作成し、受信したメールをこれらのメールボックスに保管することができる。もちろん、プライベートメールボックスに保存したメールは、いつでもクライアントから閲覧／ダウンロードできる（**図1**）。

Column

ファン

安いPC互換機を使い始めた頃は、よくファンによるトラブルをくらった。電源のファンモータのベアリングが固まってファンが止まり、過熱により電源が昇天というパターンだ。この頃の電源は、コスト低減のために軸受にボールベアリングを使わず、ただのグリース潤滑のプレーンベアリングが多かったためである。

この種の軸受は、24時間運転をしていると、早いもので数カ月、普通は1年ないし2年程度で回らなくなる。電源が昇天する前に気付けば、ファン交換でことは済む。国産のボールベアリングタイプのファンに換えた後は、数年間問題なく運転できた。

最近は、ファンの質が向上したのか、以前ほどは壊れなくなったような気がする。しかし、ケースに付いている補助ファンやCPUクーラなどには、まだけっこう怪しげなものも残っているので、油断はできない。

UNIXをIMAPサーバとして使う場合であれば、UNIXのユーザーアカウントで認証を行い、各ユーザーのホームディレクトリの下にこれらのメールボックスを作成することができる。

IMAPの重要な点は、プライベートメールボックスをサーバ側に置けることである。つまり、アクセスするクライアントが何台あろうと、すべてのクライアントで同じメールボックスを（各クライアントマシンごとに個別にコピーを持つことなく）見ることができるのである。

クライアントから接続され、リクエストに応じてさまざまな処理を行うのが、IMAPサーバである。IMAPにもいくつかバージョンがあり、現在広く使われているのはIMAP4である。IMAPの主要なコマンドを**表1**に示す。

IMAPの難点は、サーバホスト側に、プライベートメールボックス用のディスク領域が必要になることである。たとえばUNIXホストで数十人程度のユーザーをサポートするのであれば、たいした容量ではないが、プロバイダのメールサーバなど、ユーザー数が桁外れに多い場合、これをまかなうのは容易なことではない。たとえ容量に制限を設けて運用するとしても、制限を超えた際の対処が難しい。POPサーバであれば、古いメールから順に消していくといった単純な処理で済むが、プライベートなメールボックスがある場合、どのように対処すればいいのか？

またPOPに比べて、サーバに対する接続時間が長くなるという点もある。POPはバッチ的に処理を行うので、接続している時間は、事実上受信メールをダウンロードする時間だけである。それに対してIMAPは、メールセッション中はずっと接続しているという形になるだろう。個々のプロセスは軽いものだが、ユーザー数によってはサーバ側の同時接続セッション数が大幅に増えることになるだろう。

こういった理由から、多数のユーザーを抱えるプロバイダのメールサービスでは、IMAPをサポートしているものはほとんどない。会社組織内にサーバを置き、ユーザーのニーズに応じてサーバ構成を自由に設定できる環境なら便利に使うことができるが、不特定多数のユーザーを相手にするプロバイダでは、運用が難しいのである。

■POPプラスローカルMTA

複数のコンピュータを使っているユーザーにしてみれば、POPよりIMAPが便利なのは明らかである。しかし、

機能	説明
LOGIN	サーバにログイン
AUTHENTICATE	ユーザー認証
SELECT	メールボックスの選択
CREATE	メールボックスの作成
DELETE	メールボックスの削除
LIST	メールボックス一覧
APPEND	メールボックスにメッセージを追加
CLOSE	メールボックスをクローズ
SEARCH	メールボックスからメッセージを検索
FETCH	メッセージを取得
COPY	メッセージをメールボックスにコピー
UID	メッセージIDの取得

表1　IMAPの主要な機能

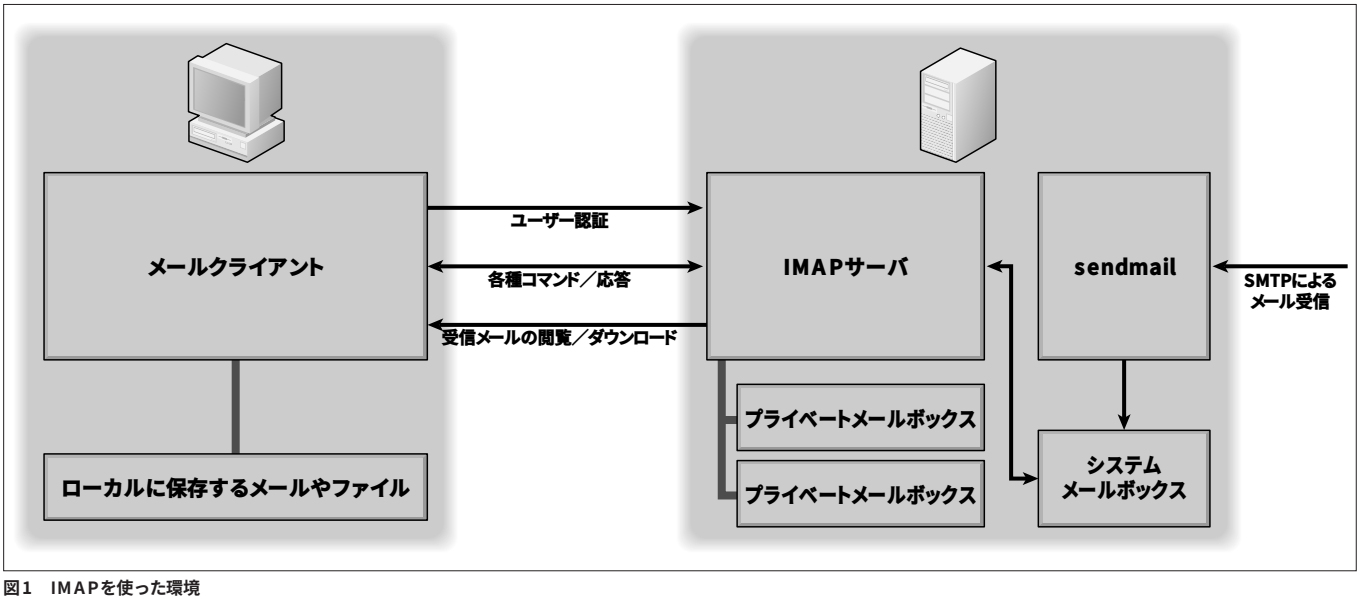


図1　IMAPを使った環境

商用プロバイダのメールサービスを使っている場合、選択肢はPOPだけである。どうにかならないだろうか？

一般に、メールをPOPでダウンロードするのは、ユーザーの手元のマシン上で動作するメールクライアントプログラム、つまりMUAであるが、POPを使うのはMUAでなければならないというわけではない。MTAでも構わない。メールがsendmailなどにより受信メールボックスにスプールされた時点で、SMTPによるメール配信は完了し、エンベロープ情報は消滅するが、ヘッダ類はすべて残っている。これをPOPでダウンロードしても、ヘッダ情報は失われない。このヘッダ情報を利用して、POPで受信したメールを、ローカルMTAを使って再配信することが可能である。

メール配信に使われるSMTPは、発信側から接続を行うプロトコルであり、常時稼働を前提としている。一方、POPは受信側から接続を行うバッチ指向のプロトコルである。また、SMTP配信を行うMTAを運用する場合は、ドメイン名やDNS、グローバルIPアドレスなども関与してくるが、POPにはこれらは関係ない。この2つのプロトコルを組み合わせることで、独自のドメインなどを運用していない場合であっても、メール配信の自由度を高めることができる。

POPとローカルMTAを組み合わせると、次のような処理が可能になる。

・受信したメールを別のアドレスに転送する

たとえばローカルMTA側のエイリアス機能やフォワード機能を使って、受信したメールを別のアドレスに転送することができる。あるいはメーリングリストを運用したり、データベースで処理したりすることも可能だ。

・POPでダウンロードしたメールをIMAPサーバで管理する

ダウンロードしたメールをホスト上でローカル配信し、ローカルホスト上の受信メールボックスにスプールし直せば、IMAPサーバを使って管理することが可能になる。

sendmailなどのSMTP用MTAは、POPをサポートしていないので、このような形のシステムを構築するためには、ローカルMTAのフロントエンドとなるプログラムが必要である。UNIXの世界では、この種のプログラムとして、fetchmailが有名である。このフロントエンドプログラムは、POPサーバに接続し、認証を行い、そのユーザー用のメールをダウンロードする。そして受信したメールを適当な配信エージェントに渡す。あとは、配信エージェントがメールをよろしく取り扱うという形になる（**図2**）。受信したメールがどのように処理されるかは、指定した配信エージェント次第である。適当なメールプログラムを指定すれば、受信したメールを別のアドレスに転送することができる。適当なスクリプトやプログラムを指定すれば、メーリングリストを運用したり、データベースアクセスなども可能である。

このようなシステム運用で広く使われているパターンは、IMAPサーバと組み合わせるということだ。フロントエンドプログラムの配信エージェント指定で、ローカル配信エージェントを指定すれば、POPで受信したメールは

Column

閲覧とダウンロード

サーバ上に置かれたメールをクライアント側で表示し、ユーザーが見るのが閲覧で、サーバ上のメールのコピーをクライアント上に作成するのがダウンロードだが、ネットワークを介してやり取りされる情報という意味ではこれらの間にほとんど違いはない。クライアント側がメールをファイルとして保存すればダウンロードであり、画面に表示するだけなら閲覧である。

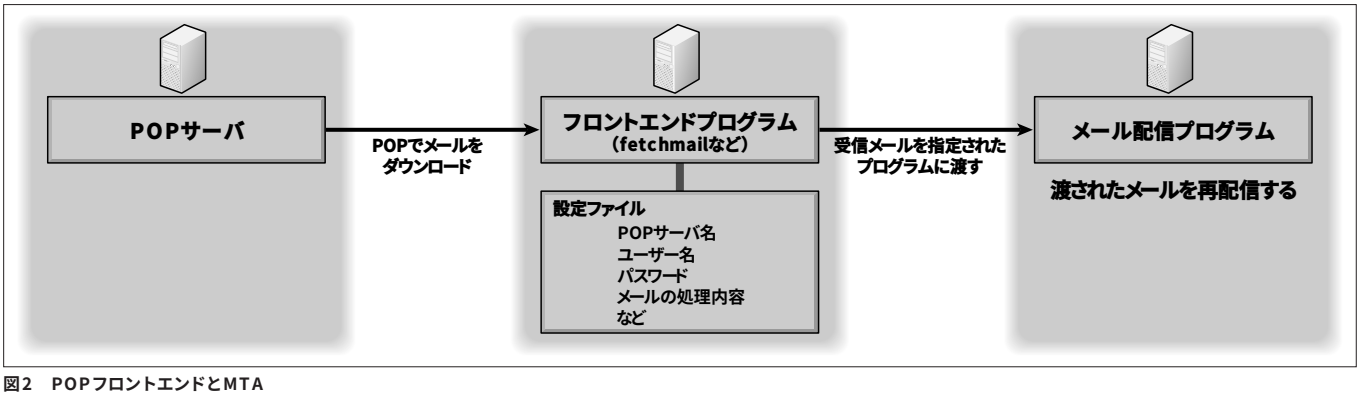


図2　POPフロントエンドとMTA

ローカルホスト上の受信メールボックスにスプールされる。メールが受信メールボックスに入ってしまえば、そのホスト上でIMAPサーバを運用し、メールクライアントからのIMAPリクエストに応えることができる（**図3**）。このホストにアクセスできるクライアントは、メールボックスを効率的に共有できるのである。

別アドレスへのメール転送などを行わず、IMAPを使いたいだけという場合は、sendmailやucbMailなどではなく、直接ローカル配信エージェントを指定すればよい。ローカル配信を行うだけならsendmailは必要ないので、sendmail.cfと格闘する必要もない。

個人ないしユーザー数のさほど多くない組織なら、このような形でプロバイダ側のPOPサーバと、個人や組織内で運用するIMAPサーバを組み合わせて、事実上IMAPの形でメールを使うことができる。出先からも使いたいのであれば、ダイヤルアップ環境を用意すればいい。

もちろん、この種のPOP／IMAPシステムを単独のプログラムとして作成することもできる。実際、Windowsなどの環境であれば、1つのプログラムとしたほうが簡単かもしれない。しかしUNIX環境であれば、配信エージェント、IMAPサーバなど、必要なツールはあらかた揃っている。そしてこういった基本ツールを組み合わせて使うというのは、UNIXの基本哲学である。したがって、ちょっと複雑そうに思えるかもしれないが、フロントエンドプログラムを使うというのが、UNIX流のやり方なのである。

■POP／IMAPサーバの動作

POPやIMAPは、クライアントに対してサービスを提供するためのプロトコルであり、pop3d、imapdといったデーモンプログラムによって実装されている。これらのプログラムは、外部からのネットワーク接続リクエストに応じて、inetdなどのネットワークデーモンにより起動される。メール受信モードのsendmailのように、常時動作し、必要に応じてforkするという形ではない。また、これらのプログラムは、起動後にユーザー認証を行い、ユーザーとのセッションを確立し、ユーザーサイドから送られたコマンドを処理するという面で、シェルセッションと似ている。実際、ポート番号（POP3は110、IMAP4は143）を指定して、telnetで接続することも可能だ。

■セキュリティ

MUAとMTAの対話の話はこのくらいにして、今日電子メールサーバを運用するうえで欠かせないセキュリティの話に入る。sendmailとは直接関係ないPOPやIMAPの話をしたのは、インターネット電子メールシステムにおける重要な要素という点もあるのだが、セキュリティの話にも多少関係するからである。

インターネットが広く普及するにつれて、悪人も増えてきた。研究ベースで運用されていた当時は、多数の組織を相互に接続するネットワークであるものの、インターネットは基本的に性善説に基づいて運用されていた。「こういうことができれば便利だろう」という発想に基づいていろいろなサービスが設計、実装され、システムは善良な管理者によって管理され、ユーザーは自分の研究や仕事のために、良心的にコンピュータやネットワークを活用す

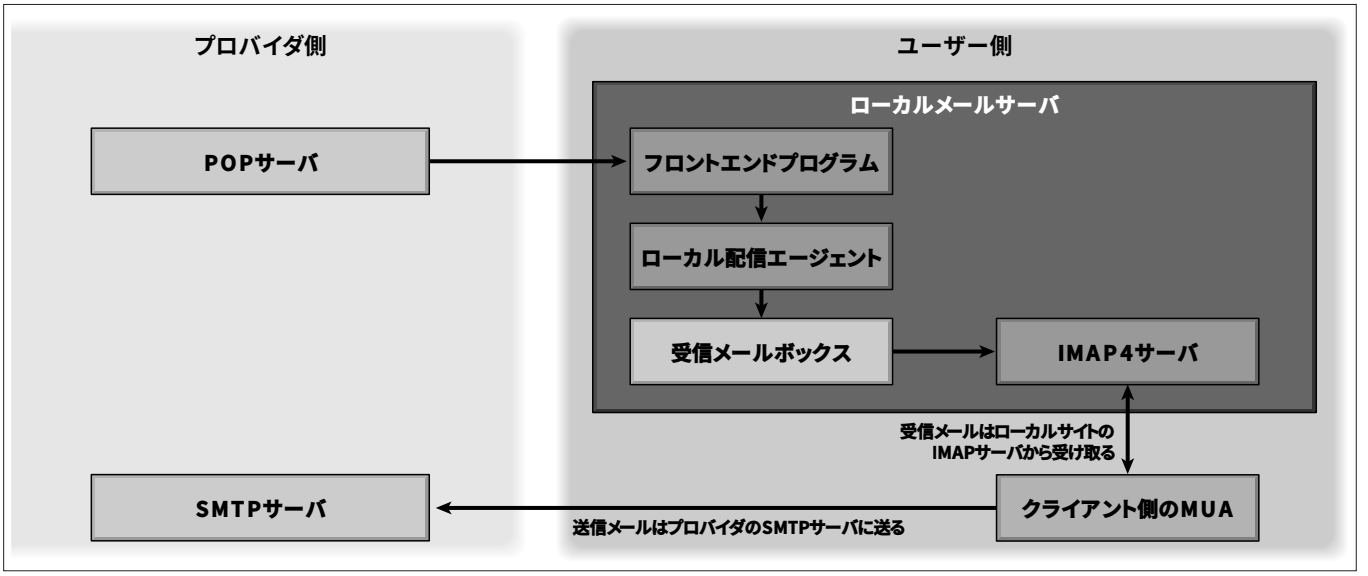


図3　フロントエンドと組み合わせたIMAPサーバ

るという前提だったのである。実際、インターネットが研究レベルで運用されていた頃は、この前提でもさほど問題は起こらなかった。

インターネットが普及していくにつれ、この前提が崩れてきた。そしてインターネットの商用利用が解禁され、パソコン通信系の業者の参入により、ユーザー層も大きく変化した。また、コンピュータの高性能化が進み、従来の中型や大型のコンピュータをTSSで使う、きちんと管理されたワークステーションを使うといった状況も変化し、個人レベルで管理されている（あるいはまったく管理されていない）PCが大量にネットワークに接続された。その結果、悪意をもってネットワーク通信を妨害する、ほかの組織のコンピュータに侵入してデータを盗んだり破壊したりする、個人に対する攻撃や中傷を行うといったインターネットを悪用する事例が増えたのである。このような治安の悪化は、インターネットに限らず、コミュニティの参加メンバーが特定少数から不特定多数になっていく過程においては、残念なことだが珍しくない。

これに対抗するには結局のところ、各個人、組織がみずから防衛するという方策しかない。性善説的な前提は完全に崩れ、「こういうことができれば便利だが、こんな危険がある」、あるいは「これは危険だから使わない」というように考えなければならなくなった。

組織や個人がインターネット上にサーバを設置した場合、当然のことながら、これは万人がアクセスできるものとなる。メールを送るとか、Webページを見るなど、本来の目的でそのサーバにアクセスしてくるのは問題ない。しかし、中には別の目的を持ってアクセスしてくるユーザーもいる。サーバの非公開の部分を覗いたり破壊したりするとか、サーバの本来の用途以外の使い方を試みるといったことである。

■電子メール関連のセキュリティ

電子メールシステムの運用も例外ではない。SMTPやPOP、IMAPなど、各種電子メールサーバを運用する場合も、セキュリティ対策を講じておく必要がある。各種のインターネットサーバに共通する要素としては、次のようなものがある。

・システムへの侵入

インターネットからアクセス可能なサーバは、特定のサービスしか提供していないつもりであっても、正規の使い方以外のアクセスを試みてくる悪人がいる。

たとえば外部からtelnetで接続できて、パスワードが破られれば、システムの中を自由に見て回り、ファイルを盗んだり改竄することができてしまう。特にrootのパスワードが破られたら、システムはもう相手の思うままである。

たとえtelnetで接続できないようになっていても、システムに侵入する方法はいろいろある。

・過負荷攻撃

特定のサーバに多数のリクエストを送り、サーバやネットワークなどの負荷を過度に高め、本来のサービスに支障をきたすような攻撃である。

困ったことに、サーバに送るリクエストは正当なものなので、不特定多数のサイトから攻撃を受けた場合、対応策はほとんどない。サーバやネットワークを停止して、ほとぼりが冷めるのを待つ程度である。

電子メールシステムに関連する要素としては、次のようなものがある。

・盗聴／改竄

電子メールの場合、ホストコンピュータ上の情報の窃盗、改竄のほかに、配信されるメールの盗聴や改竄という問題がある。

・不正な中継処理

SMTP配信を行うMTAは、受信したメールを別のMTAに転送する機能を持っている。メールシステムの悪用事例のほとんどは、MTAの中継処理に関連するものである。

・コンピュータウィルス

現在、コンピュータウィルスの主要感染経路のひとつは電子メールである。多くのウィルスは、Windowsなどのクライアントコンピュータに対して悪さを働き、MTA自体に影響するものは少ない。そのためウィルス対策は一般にクライアント上で行うのだが、MTA側で対処することも可能だ。

ここまで書いたところで誌面が尽きてしまった。次回は、電子メールシステムに関連するセキュリティ上の具体的な問題、特に不正中継について解説する予定だ。

Column

ネームサーバのセキュリティ

サイトを運営し、SMTPサーバを運用している場合は、ネームサーバのセキュリティにも気を付けなければならない。たとえば、MXレコードを改竄してしまえば、あるドメイン宛のメールをすべて不通にしたり、盗聴、改竄することができてしまう。

Linux ガベコレ

Linux Garbage Collection

目からウロコの用語辞典

文: しのはらひろあき
Text : Hiroaki Shinohara

第19回
【ワイヤレス】（わいや・れす）
【ワイフレス】（わいふ・れす）
【無銭LAN】（むせん・らん）
【無心LAN】（むしん・らん）
【IEEE802.11b】
（あいとりぷるいーはちまるにーてんいちいちびー）
【bluetooth】（ぶるーとぅーす）

ワイヤレス
【わいや・れす】

細長いものでつながっていないのに、実はつながっているかのように体感できる技術。たとえば、高度なワイヤレス技術の一例としては、「ふふふ。『モーニング娘。』では、やっぱり加護ちゃんがイチバンだよね。ほかのメンバーなんてスカだよ、スカ。ボ、ボクがいつも応援してあげるからね。ゴマキにいじめられたって、くじけちゃだめだよ。あっ、“ハロー！　モーニング”が始まった！　って言っても、ボクはMPEG-2で完全自動録画してあるから慌てる必要ないんだけどね。やっぱりリアルタイムで見て、加護ちゃんと生きた時間を共有しないとね。おおっ、加護ちゃん登場〜！　今日も輝いてるなぁ。ハァハァハァ。えっ！？　あれれっ！？　もしかして、いまコッチを見て笑ったのは、ボクのため！？　そうだ、そうだよね！　ああー、やっぱり加護ちゃんにはボクの愛が通じてるんだ！　おおっと、いまの感動を反芻すべく、タイムシフトで巻き戻しチェーック！　うんうん。ボクの小指の赤い糸、加護ちゃんとつながってるんだなぁ、やっぱり」といった思い込み（このケースの場合、特別に「ストーカー技術」とも呼ばれる）が挙げられる。

卑近な例では、Linux boxを利用する際に邪魔きわまりないケーブル類を排除していこうという動きも、またワイヤレス化であるといえる。現在ではワイヤレスマウス、ワイヤレスキーボード、ワイヤレスバンジージャンプ、ユウキレスEE JUMPなど比較的ワイヤレス化が容易なデバイスはもちろん、ワイヤレスLANさえも普及が進んでいる。そんななか、次世代のワイヤレス技術として注目を集めているのが「電源ケーブルの排除」である。この問題については世界中の素人がオープンソースの名の下にテキトーな意見を交換しあっているが、ディーゼルエンジンを使った自家発電機搭載（発生する排気ガスの清浄度が東京都規準を満たさないため、山手線より内側での使用には税金が課せられる）、PCケースへの水力発電機構の搭載（ケース上部に設置された水タンクからの落水によって発電。落ちた水は発生した電力によりタンクに揚水され永遠に発電の用を足す）などの案を押しのけ、「電力マイクロウェーブ送信技術」こそが有力と目されている。これは、家庭のAC100Vコンセントに電力を2.4GHzの超高周波へと変換するアダプタを設置、PC本体でこれを受けて熱エネルギーから電力へと再変換するというものである。民生の電子レンジの技術がそのまま利用できるので、製品化も間近いといわれる。ただし、この機能を利用している部屋には濡れたネコを入れてはいけないという些細な欠点もある。

ワイフレス
【わいふ・れす】

「KDEの2.2が出た！」、「GNOMEを1.4にアップデートしなきゃ」、「bindにまたセキュリティホールが見つかっちゃってさ」、「qmailがハヤリだからチャレンジしてみたいんだよね」などとどうでもいいインストール作業にかまけていて、肝心の夜のインストールのお勤めをないがしろにしていると、いつのまにかテーブルのうえに書き置きが一枚置かれてワイフレスになっていたという都市伝説。

誕生日にオーダーメイドのスーツを新調してくれるという

奥さんに「ぼくはフリーランスライターだからスーツは要らない。それより、Pentium 4の2GHz買ってよ」と言ったら同じ現象が起きたという未確認報告事例もある。

無銭LAN
【むせん・らん】

難癖つけ

有料のインターネットカフェに入り込んで、さんざんネットに接続したあげく、「ちょっと用を足したい」と言ってトイレから逃げたり、「待ち合わせている知り合いの到着が遅いので電話をかけてくる」と偽って逃げたり、「この店はゴキブリの入ったPCを客に出すのか！」と難癖をつけたりして金を払わずにすまそうとする犯罪行為。

無心LAN
【むしん・らん】

ブロードバンド導入を機に家庭内LANを構築、さっそく息子のパソコンもLANにつなげたのはいいが、メールやIRCで父親を呼び出しては「金くれよ、オヤジ」、「なんだ。何に使うんだ。金ならこづかいを渡しているはずだぞ」、「なんだっていいだろ、このクソじじい」、「父親に対して、その言葉はなんだ。世話になっているという意識はないのか」、「ねえよ、そんなもん。あんたのこと父親だなんて思ってねえもん」、「これまでの16年間、誰に育ててもらってきた思っているんだ」、「育ててくれって頼んだ覚えはねえ」と、そこへ祖母がログインしてきて「よしお。お父さんに謝りなさい」、「じゃまするなクソババア」、「なんだと。おまえ、おばあちゃんに向かってその口のきき方はなんだ。出て行け」、「ああ、出ていくよ。アンタが追い出した、母さんと同じようにな」、「よ、よしお」、「なんだよ、ババア」、「出ていくならそれでもいいわ。でも、それはあなたの誤解を解いてからにして」、「なに？　誤解？」、「あなたはお父さんを恨んでいるけど、それは誤解なのよ。あなたのお母さんは、お父さんに追い出されたわけじゃない。あなたのお母さんはね」、「義母さん、やめてください。そんなことこいつに言う必要はないんだ」、「うるせえ、黙ってろ。ばあちゃん、オレの母さんはどうしていなくなったんだ」、「あなたのお母さんはね、よし子は……」などというディープなやりとりもキーボードを介して交わすようになったので、血を見ることが少なくなり、家庭生活が円満になったという。

「しのはらさん、無心の話は最初の数行だけで終わりですか。オチもついてないし」、「あ、いやそれはですね、ここのところ世界情勢が不安定になっているじゃないですか。それで、一晩中ニュースを見ていると原稿を書く時間が……いえいえ、仕事をおろそかにしていたというわけではなくて、えーと、うーんと」、「なるほど、よくわかりました。この項目は書き直してくださいね」、「はあ。ところで、来月の原稿料を前借りしたいんですけど……」という殺伐とした会話もキーボードを介することで、ほのぼのとしたものに……ならなかった。失敗。

IEEE802.11b
【あいとりぷるいーはちまるにーてんいちいちびー】

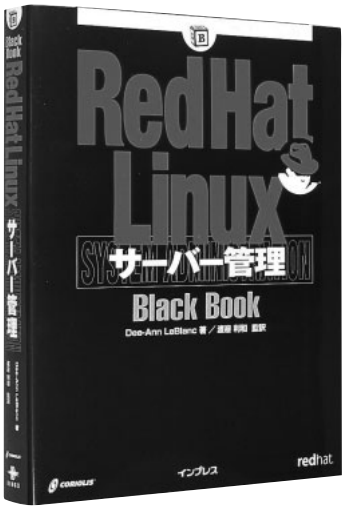
あいとりぷるいーはちまるにーてんいちいちびーと呼ばれるこの技術は、無線LANの一規格であり、昨今のワイヤレスブームに一助をなす存在であるが、とにかくあいとりぷるいーはちまるにーてんいちいちびーと読むらしいその読み方が難解であり、まどろっこしい。これはもしかしたらあいとりぷるいーはちまるにーてんいちいちびーではなく、あいいーいーいーはちまるにーてんいちいちびーとか、いえええはちまるにーてんいちいちびーとか読むのではないか、いやいや、「てん」ではかっこわるいので「ドットコム」のようにあいとりぷるいーはちまるにーどっといちいちびーと読む可能性もあるぞ、などと考えてしまう。

そういえば、あいとりぷるいーはちまるにーてんいちいちびーと最後だけ違う、あいとりぷるいーはちまるにーてんいちいちえーと読む規格や、あいとりぷるいーはちまるにーてんいちいちじーと読む規格も、じきに実用化されるらしい。読み方についてちょっと書くだけでこんなに原稿用紙が埋まってしまうとは、まじめなライター泣かせである。あいとりぷるいーはちまるにーてんいちいちびーって。

bluetooth
【ぶるーとぅーす】

より低コストで無線対応の機器を作れるように策定されたワイヤレスの規格。bluetoothの名は、北欧の伝説の王である「青歯王」から。この王はオープンな北欧出身らしくヌーディストで、厳寒の中でも素っ裸だったので体を壊し、歯槽膿漏を併発して歯茎が青紫色であった。不健康な王の名を冠したために普及が遅々として進まない不景気な状況に陥っているのがbluetoothの現状だ。今後どのように展開していくのか、よしおのお母さんのよし子はいったいどこへいったのか、来月の原稿料を前借りできるのか、気になるところである。

Books



Red Hat Linuxサーバー管理 Black Book

Dee-Ann LeBlanc 著／渡邉利和 監訳
インプレス
B5変形判／592ページ　　本体価格 3800円

Linuxが優れたサーバ機能を持つOSであることは言うまでもない。しかし、UNIXの流れをくむLinuxにおいて、サーバを構築することは簡単ではない。設定ファイルが多岐に渡わたるだけでなく、ネットワークやUNIXの前提知識が必要となるからだ。こうした点がUNIX系OSに初めて触れる管理者を悩ます要因となっている。

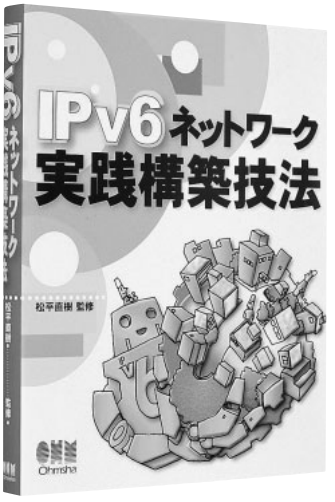
本書は、Red Hat Linux 7.1をベースに、BIND、sendmail、Apacheといった各種サーバを構築する際に必要となる前提知識と設定手順を順を追って解説している。このほかにも、ファイルシステムやパッケージの管理法、インストール手順を自動化するKickStartなど、管理面についての解説が充実している。構成も、テクニカルリファレンスをうたうBlack Bookシリーズらしく、知りたい情報にすぐにたどり着けるよう配慮されている。脱初心者を目指す管理者にとって、たいへん重宝する一冊となるだろう。

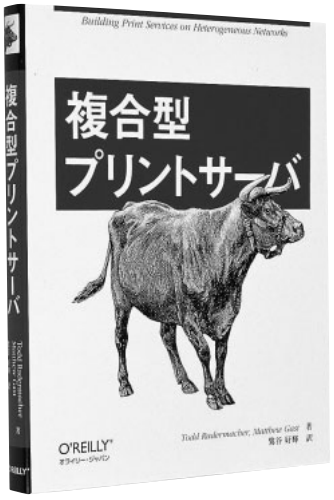
IPv6ネットワーク実践構築技法

松平直樹 監修
オーム社
B5変形判／336ページ　　本体価格 4400円

現在のインターネットの基盤となっているIPv4は、1981年に仕様が固まったプロトコルだ。今年で20年が経過したことになる。策定当時に比べインターネットは爆発的に普及し、各ホストに振られるIPアドレスがすでに枯渇状態にあることは周知の事実である。このような状態を解消すべく、新しいIPv6の普及が切望されている。IPv6は、膨大な数のIPアドレスが利用できるうえ、通信内容を暗号化するIPsecを標準で取り込むなど、今後のインターネットの発展に欠かせない技術であることは疑いようもない。

本書は、IPv6の技術を詳しく解説しながら、実際にIPv6ネットワークを構築する手順を解説している。さらに、各種IPv6対応ルータの設定方法が実例とともに紹介されているなど、IPv6によるネットワーク構築を検討している管理者、IPv6対応のアプリケーションあるいはサービスを開発しようとしている技術者にとって十分実践的な内容となっている。





複合型プリントサーバ

Todd Randermacher、Matthew Gast 著／鷲谷好輝 訳
オライリー・ジャパン／オーム社
B5変形判／336ページ　　本体価格 3500円

ちょっと中身が想像しづらい邦題だが、複数のOSが混在するネットワーク環境におけるプリントサーバの設定・管理について解説した書籍である（ちなみに原書のタイトルは『Network Printing』）。UNIXの伝統的なプリントスプーラであるlpdとより多機能でマルチプラットフォーム環境との親和性の高いスプーラLPRngの設定方法をはじめ、Windowsとのプリンタ共有に欠かすことのできないSamba、Macintoshとのプリンタ共有を実現するnetatalkを使ったプリントサーバの構築について、またSNMPやLDAPによるネットワークプリンタの管理、さらにはセキュリティ、アカウンティング（課金）、チューニングまで、プリントサービスに関するトピックを網羅的に取り上げている。各トピックについてそれほど深く掘り下げてはいないが、要点は簡潔にまとめられている。巻末には付録として、PrintcapのリファレンスとSNMPのプリンタMIBに関するRFC文書を収録。

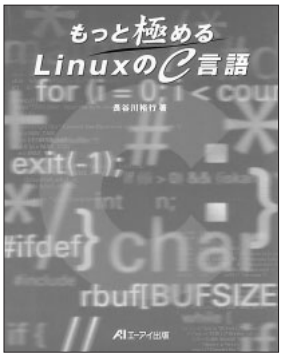
3000万円のムダ遣い

スタパ齋藤 著
アスキー
四六判／168ページ **本体価格 1400円**

著者のスタパ齋藤氏は、ハードウェア記事を中心に（というか、ほとんど全部が「お買い物」系企画のような気もするが）アスキーを始めとするパソコン雑誌で活躍中のライターである。思いっきり好き嫌いの分かれそうなアクの強い文体とコワモテの容貌でもって、パソコン系のライターとしては珍しく「キャラ立ち」のする存在だ。

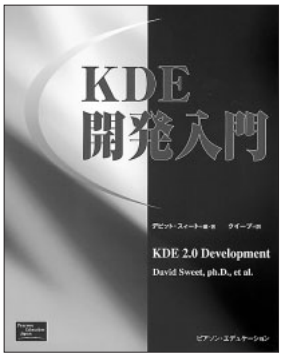
そんなスタパ氏のもうひとつの名が「物欲番長」。本書によると社会人になってから購入したパソコンが90台、携帯電話が63台、デジカメは36台、ビデオカメラとPDAが各12台、おまけにDVDソフト400本、という恐るべき買いっぷりである。これに周辺機器やら何やらを合わせると、その金額が本書のタイトルにある3000万円となるわけだ。こうなると、もはや仕事云々の枠をはるかに越えている。純粋に己が物欲に従う姿は爽快でさえある。日本人の何割かがスタパ氏を見習えば、景気なんてあっという間に回復しそうである。

もっと極めるLinuxのC言語



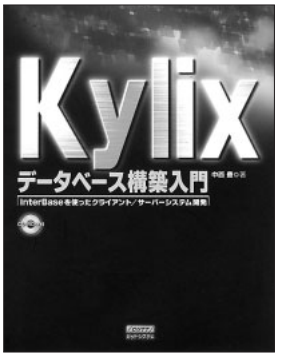
長谷川裕行著
エーアイ出版
B5変形判／352ページ
本体価格 2600円

KDE開発入門



デビット・スィート 著／クイープ 訳
ピアソン・エデュケーション
B5変形判／586ページ
本体価格 5200円

Kylixデータベース構築入門



中西豊著
カットシステム
B5変形判／280ページ／CD-ROM1枚付き
本体価格 2800円

プログラミングRuby

デビット・トーマス+アンドリュー・ハント 著／田和 勝 訳／まつもとゆきひろ 監修
ピアソン・エデュケーション
B5変形判／696ページ
本体価格 4800円

読者の声

俺にもいわせろ！

担当の家には、モルモットだけでなく、2年目に突入するミドリガメもいます。ミドリガメはミドリと名がつきながらも、最近ミドリ色でなくなってきました。怪しく思った担当が、Webで検索すると……。そこには！　思いもよらない事実がぁっ！　ミドリガメにはミシシッピアカミミガメという本名があり、甲長だけで20〜30センチにもなるというのです。それに加えて、20〜30年も長生きしちゃうらしいのです！　なんということでしょう。でも、ちょっと楽しみです。

パーティション

Turbolinuxインストール時のパーティション検索結果の表示は DOSしか知らない人間にはわかりづらかった。UNIXもすでに組んであるものを少し触った程度だったので。

（大土居 俊章さん）

ちょっとした興味本位でLinuxなるものを入れてみたくなり、生まれて初めてこの雑誌を買ってみました。本音を言うと小学生の時に初めて算数の教科書を見たときぐらい知らない言葉だらけで、泣きそうになりました。でもインストールの方法がとても詳しく書いてあり、チョー初心者の私でもインストールができました！　すごいです！！　でも「パーティション」ってものがよくわからなくって、どうやらドライブが1つ丸ごとLinuxに奪われたようすです。まぁ初めてだし、これもひとつの経験だな、と思って諦めました（笑）。ちょっと毎月買うにはお値段がきついけどそのぶん内容がギッシリあって、別の雑誌と比べて買ったんですけど得した気分です。今後ともよろしくお願いしますね。

（ごまごまさん）

たとえドライブごとデータを失おうとも、「まぁ、初めてだし」で済ませられるごまごまさんとは、なんて素晴らしい人なんでしょう。

パーティションとは、1台のドライブを論理的に複数台に分割することです。通常、1台のマシンに複数のOSをインストールする際には、パーティションを切って、それぞれにインストールします。1つのOSが占領しているハードディスクに、ほかのOSをインストールする場合には、1．パーティションを切って、インストールしなおす。2．PartitionMagicなどのソフトを使って、パーティションのサイズを変更する。3．Virtual PCなどのソフトを使う。などの方法があります。

Virtual PCとは、1台のコンピュータの中にいくつも仮想のコンピュータを作って、OSをインストールすることができるソフトです。複数のOSをインストールするために、ドライブを分けたり、パーティションを切ったりする必要がありません。アプリケーションを切り替えるように、すべてのOSを同時に起動することもでき、大変便利です。

今月号の読者プレゼントであるハードディスク切り替え器を使って、デュアルブートに見切りをつけるのも、またよいでしょう。

アンケート

GIGABYTEのグッズ当たりました。ありがとうございます。大須の街（名古屋在住なので）をグッズを身につけながら闊歩してきました。痛いぐらいの注目を集めてしまったので途中からキャップはやめてこっそりストラップを前面に押し出して楽しんできました。

（梅本 満さん）

久しぶりにアンケートページに来ています。毎月、Linux magazineを買って読んでいるのですが、最近仕事が忙しかったりして、読み切らないうちに、次号が出てしまって追いつけません。ちょっと前までは、次号が出るのが楽しみだったのに、 最近は、もう少しゆっくり出てくれないかなー、と思っています。

（松村 岳さん）

各プレゼント倍率の順位結果を見るのは、担当の楽しみのひとつです。8月号の読者プレゼントには、Itaniumグッズが2点入っていました。キーホルダのほうは、特によい品だったので、間違いなく

Itaniumキーホルダがプレゼント希望者数ダントツの1位だろうと予想していました。話題性良し、デザイン良し、時期良しなんですから。予想にだいぶ自信があったのです。ところが、どうしたことか、静音キットがダントツの1位でした。応募者の皆さんの裏読みによる結果だったのでしょうか？　キーホルダの倍率が4倍だったとは……。担当も応募したかったなぁ。

消えたファイル

GNOME、KDEの特集はどんどんしてほしいです。あまりインターネットを閲覧している時間もないので、頼るのはLinux magazineというところです。

ところで、本文中にWebからダウンロードして下さい。なんて記事が多いのですが、私がゆっくり学習しているせいか、時すでに遅しの時がいっぱいあり、ページが変更になったりしてダウンロードできなくて困っています。できれば、付属CD-ROMに入れてほしいのですが……。InterBaseの記事も勉強がてらに頑張っていたのですが、途中で断念せざるを得なかったし。容量の問題で難しいとは思いますが、ご検討ください（一初心者を助けると思って）。

（三好哲史さん）

できるだけCD-ROMに入れるように努力しているのですが、CD-ROMの容量や、ソフトの権利のために、CD-ROMに入れることができず、URLのみをご紹介することがあります。ご迷惑お掛けして申し訳ありません。Webページやムックなどで、極力フォローしていきますので、ご了承くださいませ。

ご指摘のInterBase 6.0の連載記事の場合ですと、たとえば第5回のfirebird 0.9 for Linuxのファイルが移動されています。その代わりに、firebirdのWebページ（http://sourceforge.net/projects/firebird）からバージョン1.0をダウンロードできるようになっていました。そちらからダウンロードしてください。

アクセス速度

Linuxはハードディスクのアクセス速度がWinに比べて どうも遅いように感じるんですが、なんででしょう？　ドライバのせいですか？

（河村 智さん）

データの転送モードがDMAになっていないのかもしれません。

hdparm /dev/hdaを実行して、using_dmaがoffになっていたなら、hdparm -d 1 /dev/hda でonにしてみましょう。速くなったでしょう？　-tオプションで、ハードディスクの読み取り速度を計測して確かめてみましょう。hdparmには、ほかにもパラメータがいろいろあるので、試してみてください。

いろんな場面で

大学の研究室でLinuxをメインに活用しています。Linux magazineのおかげでいろいろとLinuxに関する知識が増えました。これからもわかりやすい内容でお願いします。

（中西康介さん）

2000年2月から購読を始めたが、ここに来てやっと誌面の3分の1程が少し読めるようになった（何とかキーを叩いて誌面をなぞれるくらい）。それまで色々雑誌を取り換えていたが、結局時間をかけるしかなかったんだと今思う。でもこれからも、先が長そー。途中で投げ出さなさなければ、これからもやさしく面白くを期待します。

（takeさん）

転職したソフト会社でLinux用のソフト（ドライバ？）の開発をやることになり、まずはLinuxの勉強を、と思い貴誌を購入しました。これからの勉強用にしたいと思いますので、初心者にもわかりやすい誌面作りをよろしくお願いします。

（海潮さん）

貴誌の7月号を購入したのをきっかけに（それから毎号購入しております）Linuxを使い始めたものの、まだどのディストリビューションに固定しようか迷っていました。

そんなときに10月号の「ディストリビューション最新カタログ」は非常にありがたいものでした。特に目的があるわけではなくただLinuxを使ってみたいという理由で使っています。

今はまだ、記事を読みながらそれに従っていじっているだけなのですが、それだけでも面白いです。少しずつ勉強していこうと思います。これからもよろしくお願いします。

（宮原 眞一郎さん）

ご購読ありがとうございます。こちらこそ、これからもよろしくお願いします。

宮原さん、どのディストリビューションにするか決まりましたでしょうか？　編集部のメンバーが使っているディストリビューション第1位は……、と統計をとろうとしたのですが、皆きれいにバラバラでした。debian愛好家以外は、そのとき近くにあったものをインストールしたみたいです。記事を書くために、徹底調査していろんなディストリに愛着があるのでしょうか？　どれか1つには決められないみたいです。

付録CD-ROMに収録した
Turbolinux7 Workstation(FTP版)のインストール

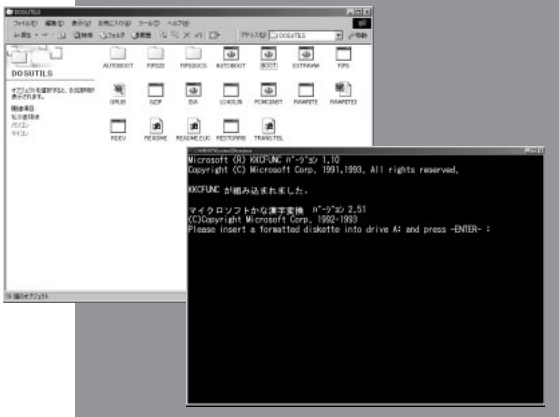
本誌の付録CD-ROMに収録したTurbolinux7 WorkstationはFTP版です。非商用ソフトだけが含まれており、ターボリナックスジャパン株式会社や編集部からサポートを受けることはできません。インストールなどのサポートが必要な場合は、製品版Turbolinux7 Workstationのご購入をお勧めします。
　また、CD-ROMのメディアに不良があった場合は、代替メディアをお送りしますので、お手数ですが住所、氏名を明記のうえ（linux-cd@ml.ascii.co.jp）宛にご連絡くださるようお願いいたします。

1 インストールの前準備

　これからTurbolinux7 Workstationのインストールを始めます。多くのネットワークカード、ビデオカード、モニタは自動認識されますが、自動認識されない場合に備えて、あらかじめハードウェアのマニュアルなどを用意しておくとインストール中にあわてずにすみます。
　また、レスキュー用のフロッピーを作成するために、空のフロッピーを1枚用意しておいてください。CD-ROMブートできないマシンを使う場合はインストーラ起動用のフロッピーも用意します。

2 ブートフロッピーの作成

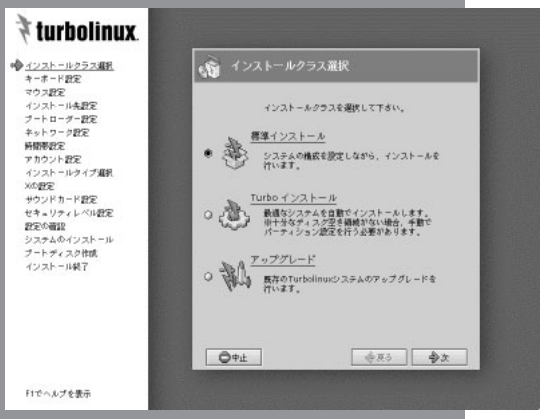
　CD-ROMブートできないマシンにTurbolinuxをインストールする場合は、以下の手順でブートフロッピーを作成します。
　まず、Windowsマシンに付録CD-ROMのDisc 1と空のフロッピーディスクをセットします。次に、エクスプローラを使ってCD-ROMの［DOSUTILS］を開き、［BOOT］をダブルクリックします。するとDOS窓が立ち上がるので、［Enter］をタイプしてインストーラ起動用のブートディスクを作成します。



3 インストーラの起動と使用言語の選択

　CD-ROMやブートフロッピーをセットしてマシンを起動すると、Turbolinuxのインストーラが起動します。［boot:］プロンプトが表示されたら［Enter］を押します。
　使用する言語の選択場面に切り換わったら、［↓］キーで［Japanese］を選択して［Enter］キーをタイプします。
　しばらくすると、Xを使ったグラフィカルなインストール画面が表示されます。
　なお、［boot:］の箇所で［text］と入力して［Enter］を押すと、テキストベースのインストーラが起動します。また、インストーラがグラフィカルな画面の表示に失敗した場合は、自動的にテキストベースのインストール画面が起動します。

既知の問題として、ラトックシステム社製のSCSIカードが接続されているPCにTurbolinux7をインストールすると、SCSIカードに不具合が起きることが判明しております。同社製のSCSIカードを利用されている場合は、以下のURLをご確認ください。
http://www.turbolinux.co.jp/knowledge/public/454.html



インストールクラスの選択

ここでは［標準インストール］を選択して次に進みます。

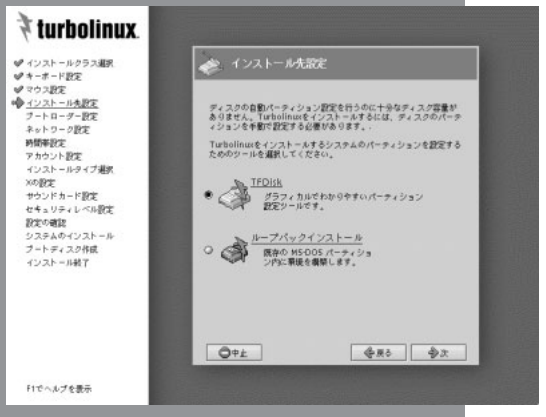
4

キーボードとマウスの設定

使用するキーボードタイプをプルダウンメニューで選択します。キーボードを選択したら、［ここで選択した設定をテストしてください］の欄でキーボード入力をテストします。［Shift］キーを押しながら［1］、［2］、［3］を押して、キーボードの刻印どおりの記号が表示されたら、キーボードの設定は成功しているでしょう。

次にマウスを設定します。使用するマウスタイプをプルダウンメニューから選択します。マウスを選択したら、［マウスのテスト］の領域でマウスのボタンを押して、マウスの設定が反映されているかどうかをチェックしてください。

5



パーティション作成ツールの選択

マシンをLinux専用に使う場合は［TFDisk］を選択します。また、Windowsと共存させる場合は以下のように選択します。

■Windows 9x／Meと共存させる場合

ディスクがすべてWindows領域である場合は、［ループバックインストール］を選択するか、［TFDisk］を選択し［手順8］から始まるPartedの使い方を参考にしてLinux用の領域を作成します。［ループバックインストール］はLinuxのインストールとアンインストールが簡単ですが、［TFDisk］を選択する場合と比べて動作が遅くなります。

■Windows NT／2000と共存させる場合

多くのWindows NT／2000環境で使われているNTFSというファイルシステムにはループバックインストールできないうえに、PartedもNTFSには未対応なので、Linux用の領域を作成するには、PartitionMagicなどの商用ソフトを使うか、Linux用に使えるディスクを1台追加する必要があります。

6

ディスクの設定（[手順6]で［ループバックインストール］を選択した場合）

［手順6］で［ループバックインストール］を選択した場合は、まず、インストールするディスクを選択して［次］を押します。

次に、Linux用に使う領域のサイズを指定します。［rootファイルシステムの容量］も［スワップ領域の容量］もデフォルトのままでよいでしょう。

7

Partedの起動（[手順6]で［TFDisk］を選択した場合）

［手順6］で［TFDisk］を選択して、Partedを使ってLinux領域を作成する場合は、棒グラフ状に表示されている［/dev/hda］を押して、［Parted］をクリックします。［Parted］をクリックすると、画面（手前）のようにPartedが起動します。

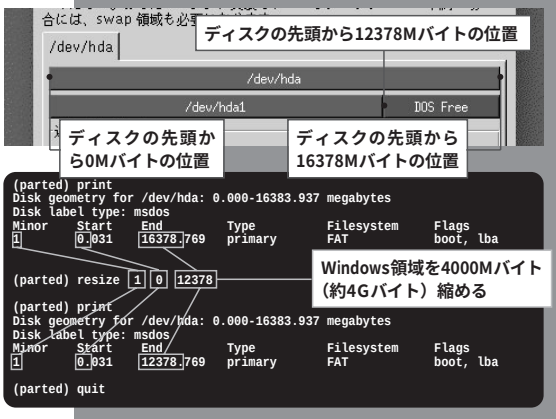
すでにLinux用の領域がある場合は、Partedを起動する必要はありません。

Partedを使ったLinux領域の作成

Partedを使って、たとえば画面（上）のように約16GバイトあるWindows 9x／Me領域を約12Gバイトに変更します。Windows領域を縮めることで作成された約4Gバイトの領域をLinux用に使います。

まず、(parted)というプロンプトのところで、画面（下）のように［print］コマンドを実行します。すると、ハードディスクの使用状況が表示されます。次に、［resize］コマンドを使ってWindows領域のサイズを変更します。

Partedは、［quit］コマンドを実行すると終了します。



パーティションの作成

Linuxの使用には、最低限Swap用とシステム用の2つのパーティションが必要です。

まず、［DOS Free］を選択して、［パーティションの追加］をクリックします。すると、別のウィンドウが表示されるので、画面（左）の例のようにSwap用とシステム用（Linux ext2）の2つのパーティションを作成します。

［swap］にはマシンに搭載しているメモリの1～2倍程度のサイズを、［/］には4Gバイト程度のサイズを指定します。

Linuxパーティションのフォーマットとブートローダの設定

［OK］をクリックして次に進みます。［フォーマット中に不良ブロックをチェックする］は、特にチェックする必要はありません。

次にLILOのインストール先を設定します。Windows NT／2000と共存させる場合は、［LILOのインストール］のチェックを外し、Windows 9x／Meと共存させる場合とマシンをLinux専用に使う場合は、デフォルトのまま［次］を押します。

［ブートディスクを作成する］には、必ずチェックを入れておきます。

ネットワークとタイムゾーンの設定

TurbolinuxをDHCPサーバからネットワーク情報を取得する環境で使う場合は、［DHCPを使用して設定する］を、DHCPサーバのない環境で使う場合は［DHCPを使用して設定する］のチェックを外して、IPアドレスなどのネットワーク情報を入力します。なお、この場面ではダイヤルアップ接続の設定はできません。

次にタイムゾーンを設定します。デフォルトでは日本時間がセットされているので、たいていのユーザーはタイムゾーンを設定する必要はありません。［システムクロックでUTCを使用］は世界標準時に合わせるためオプションです。Windowsなどと共存させる場合は、このオプションを外しておくのがよいでしょう。

NTPを使ってマシンの時刻を設定する場合は、［タイムサーバー］タブをクリックして使用するNTPサーバのホスト名かIPアドレスを入力します。

ユーザーアカウントの作成

まずはLinuxの管理者用ユーザー（ログイン名はroot）のパスワードを設定します。［rootパスワード］と［確認］に同じパスワードを入力します。［確認］欄の右側に「確認しました」と表示されれば設定は成功しています。

次に、メールの読み書きなどを行う一般ユーザーを作成します。まずは［ユーザー作成］タブをクリックして画面を切り替えます。［ユーザー名］に一般ユーザーのログイン名を入力して、［パスワード］と［パスワード（確認）］に一般ユーザー用のパスワードを入力します。［パスワード（確認）］の右側に「確認しました」と表示されたら［追加］を押し、さらに［次］を押して次に進みます。

インストールタイプとモニタの選択

インストールするパッケージグループを選択します。各グループを選択すると、そのグループの説明が下の欄に表示されます。ここでは、デフォルトの［標準デスクトップ］を選択してインストールします。［標準デスクトップ］を選択した場合は、ハードディスクを約1.5Gバイト消費します。

次はモニタの設定です。使用するモニタが自動認識されなかった場合は、リストの中からモニタを選択します。使用するモニタがリストにない場合は、［製造元］で［Generic］を選択し、［モデル名］で、モニタのマニュアルに記載されている解像度を超えない範囲のモードを選択します。

なお、液晶ディスプレイを使用する場合は、［モデル名］で［Laptop Display Panel］から適切な解像度のものを選択します。

Xの設定

ほとんどのビデオカードは自動認識されるでしょう。ひとまずインストーラが自動設定した［デスクトップカラー］と［デスクトップサイズ］のままで、［この設定をテストする］を押してXの表示をテストします。表示に成功したら、Turbolinuxのデスクトップ画面が表示されるので、表示される質問に［はい］と答えてテストを終了します。

テストに失敗した場合や、解像度を変更する場合は、［デスクトップカラー］と［デスクトップサイズ］を手動で選択して、Xの表示テストを繰り返します。

［グラフィカルログインの使用］は、Xの表示テストに成功した場合のみにチェックしてください。［グラフィカルログインの使用］をチェックすると、Linuxの起動後にグラフィカルなログイン画面が表示されます。

サウンドカードの設定

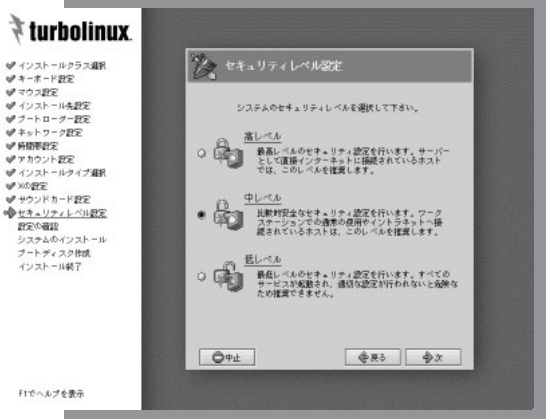
サウンドカードが自動認識されたら、［ドライバの読込］を押して、次に進みます。自動認識されない場合は、そのまま次に進みます。

16

セキュリティレベルの設定

セキュリティレベルの設定で、どれを選択してよいのかわからないユーザーは、デフォルトの［中レベル］を選択したまま［次］を押しましょう。

17

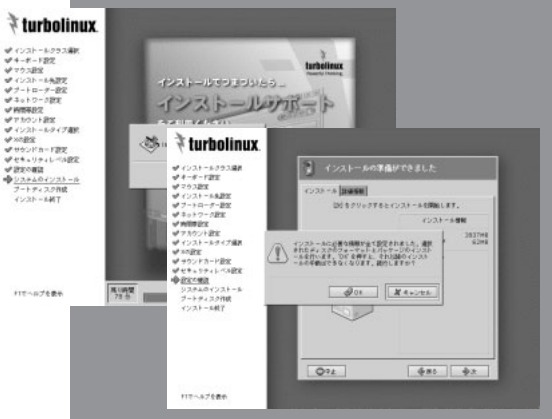


パッケージインストールの開始

セキュリティレベルを設定したら、いよいよパッケージのインストールを開始します。［インストールの準備ができました］の画面で［次］をクリックすると、画面（手前）のようにインストールの開始を確認するウィンドウが表示されるので、［OK］をクリックしてパッケージのインストールを始めます。

［インストールタイプの選択］で［標準デスクトップ］を選択した場合は、パッケージのインストールに、128MバイトのメモリをAMD K6-II/400MHzマシンで約40分かかります。パッケージインストールの途中で、画面（奥）のように［Install CD2を挿入してください］というメッセージが表示されたら、付録CD-ROMのDisc 2をセットしてインストールを続けてください。

18



インストール完了とブートディスクの作成

パッケージのインストールが終了すると、追加パッケージのインストールに進みますが、本誌付録のCD-ROMには追加パッケージを収録していません。［追加パッケージのインストールをスキップする］をチェックしたまま次に進みます。

Linuxをハードディスクから起動できなくなった場合などに備えて、レスキュー用のフロッピーディスクを作成します。空のフロッピーディスクをセットして、［次］を押してください。フロッピーディスクの作成が終わるとインストールは終わりです。［完了］を押し、CD-ROMとフロッピーディスクを抜いてください。お疲れさまでした。

19